Privacy-Preserving Multi-Party Access Control

Mina Sheikhalishahi
Eindhoven University of Technology
m.sheikhalishahi@tue.nl

Gamze Tillem
TU Delft
g.tillem@tudelft.nl

Zekeriya Erkin
TU Delft
z.erkin@tudelft.nl

Nicola Zannone
Eindhoven University of Technology
n.zannone@tue.nl

ABSTRACT

Multi-party access control has been proposed to enable collaborative decision making for the protection of co-owned resources. In particular, multi-party access control aims to reconcile conflicts arising from the evaluation of policies authored by different stakeholders for jointly-managed resources, thus determining whether access to those resources should be granted or not. While providing effective solutions for the protection of co-owned resources, existing approaches do not address the protection of policies themselves, whose disclosure can leak sensitive information about, e.g., the relationships of co-owners with other parties. In this paper, we propose a privacy-preserving multi-party access control mechanism, which preserves the confidentiality of user policies. In particular, we propose secure computation protocols for the evaluation of multi-party policies, based on two privacy-preserving techniques, namely homomorphic encryption and secure function evaluation. An experimental evaluation of our approach shows its practical feasibility in terms of both computation and communication costs.

CCS CONCEPTS

• **Security and privacy → Access control**; • **Theory of computation → Cryptographic protocols**.

KEYWORDS

Collaborative systems; homomorphic encryption; secure function evaluation.

ACM Reference Format:
Mina Sheikhalishahi, Gamze Tillem, Zekeriya Erkin, and Nicola Zannone. 2019. Privacy-Preserving Multi-Party Access Control. In *18th Workshop on Privacy in the Electronic Society (WPES'19), November 11, 2019, London, United Kingdom.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3338498.3358643

1 INTRODUCTION

Over the last decade, collaborative systems (e.g., social networks, content sharing platforms) have gained momentum. These systems provide platforms and tools to support the interplay among their users and facilitate the sharing of information and resources. Users can establish online relationships that resemble real-life interpersonal relationships and, based on them, cooperate to create, manage and protect their resources.

As collaborative systems have emerged and their popularity is significantly increased, the need of proper mechanisms for protecting sensitive resources shared in these systems has become of paramount importance [34]. Multi-party access control has been proposed as a new paradigm to overcome the inherent limitations of traditional access control models in dealing with the demands of collaborative systems. Specifically, multi-party access control aims to support the collaborative protection of co-owned resources by providing a means for collaborative decision-making, in which the access requirements provided by all entities involved in the protection of a jointly-managed resource are accounted for and (possible) conflicts arising from conflicting access requirements are reconciled to determine whether access to the resource should be granted or not.

Several solutions for collaborative decision-making have been proposed over the past years, ranging from authoritative approaches, in which the access requirements provided by the various entities are combined in a predefined manner [14, 29], to approaches aiming at a mutual agreement among the entities involved in the protection of co-owned resources [19, 36]. Regardless how collaborative decisions are made, to the best of our knowledge, no prior work has addressed the protection of the policies themselves within collaborative systems.

Policies might contain sensitive personal information, e.g., about the relationships among entities, or confidential corporate information and, thus, their disclosure could raise concerns for users and organizations [14, 41]. For instance, users might not define their privacy settings freely if they are visible to (or can be inferred by) other users due to *social pressure*. Indeed, several studies show that in social networks users' behavior is largely influenced by other users [18, 24]; for instance, it has been shown that adolescents are more likely to share a photo – even showing improper behavior – if that photo had received several likes from their peers [38].

At the same time, users might also want to minimize the amount of information to be disclosed to the collaborative platform itself. Several cases have been reported where online platforms and services used and/or disclosed their personal data (including their privacy preferences) without user consent [9], or were impacted by data breaches. These concerns are becoming particularly critical with the advancement in data science and the increasing number of available data analysis tools that can be used to correlate user policies with other pieces of personal information and therefore reveal much more insights about users. For instance, knowledge of the interpersonal relations between users along with their privacy settings and other metadata can be used to profile users (e.g., identifying individuals alienated from the community), and this information exploited for targeted advertising.

We argue that minimizing the information a user has to disclose allows reducing the trust that she has to pose on the platform. In this light, users should be able to specify their policies in a private form while the collaborative platform should still remain able to make access decisions based on those policies.

In this work, we address this challenge and propose a privacy-preserving framework for multi-party access control, in which users provide their policies in a private form and policy evaluation is performed on private inputs, thus preserving the confidentiality of user policies. Following the multi-party policy model in [29], we build a 'collaborative' policy, specifying both positive and negative authorizations for a given co-owned resource, by combining user policies using policy combining operators. Intuitively, policy combining operators define strategies specifying how policy conflicts should be resolved (*e.g.*, permit overrides, deny overrides, first applicable). For the evaluation of collaborative policies while protecting the confidentiality of users' policies, we employ secure computation protocols that implement policy combining operators on private inputs.

Secure computation has been applied in a large range of domains, spanning from medical data analysis [6] and face recognition [16], to constructing classifiers [7] and clustering algorithms [37]. However, no prior work has addressed the protection of inputs for three-valued logic operations, which are required for the specification and evaluation of collaborative policies that support both positive and negative authorizations.

To realize a practical mechanism for multi-party access control supporting policy evaluation on private inputs, we investigate the use of two well-known privacy-preserving techniques, namely homomorphic encryption (HE) [35] and secure function evaluation (SFE) [13], for the design of secure computation protocols tailored to operate on three-valued logic. We also investigate different encodings for these operators. Based on these privacy-preserving technologies and encodings, we design three variants for each policy combining operator: one based on homomorphic encryption, one based on SFE using a direct encoding of three-valued logic operators and one based on SFE using a Boolean logic encoding of the operators. The proposed protocols can be used as building blocks for the evaluation of arbitrary collaborative policies.

We demonstrate the security of the proposed protocols, as well as of their composition, in the presence of a semi-honest adversary, thus guaranteeing that policy evaluation does not leak any unintended information. We have implemented the proposed protocols and evaluated their computation and communication costs through experiments. The results shows that the SFE-based protocols defined over a Boolean logic encoding of the operators outperform the protocols defined over the other approaches and provide a foundation for the practical realization of privacy-preserving mechanisms for multi-party access control.

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 policy evaluation and Section 5 discusses their security. An experimental evaluation of their computation and communication costs is presented in Section 6. Finally, Section 7 discusses related work and Section 8 concludes the paper.

2 BACKGROUND

This section provides the background knowledge used in this work, including a multi-party policy model, homomorphic encryption and secure function evaluation.

2.1 Multi-Party Policy Model

In this work, we adopt the data governance model proposed in [29] for the specification of multi-party access control policies, *i.e.* policies regulating the access to co-owned resources. This model provides a general framework to reason on the level of authority that users have over shared resources and to build a multi-party access control policy based on their authorization requirements. Specifically, it captures the relations that users have with a shared resource and, based on these relations, determines policy combination strategies to resolve possible policy conflicts. Compared to other works (see [34] for a survey), the model in [29] allows a more fine-grained governance of shared resources by representing and ordering levels of authority, which can be instantiated using arbitrary conflict resolution strategies.

To formalize access control policies regulating co-owned resources, we use a policy combination algebra inspired by PTaCL [10, 30], which provides an abstraction of XACML [33], the de facto standard for attribute-based access control.

Let $\mathcal{U}$ be the set of users and $\mathcal{S}$ the set of their (individual) policies.[1] Policies for regulating access to co-owned resources are specified using a policy language $\mathcal{P}$. Formally, a multi-party policy $p \in \mathcal{P}$ is defined by the following grammar:

$$p = s \mid \text{op}(p_1, \ldots, p_n)$$

Intuitively, a multi-party policy p is an expression built over user policies $s \in \mathcal{S}$ using policy combining operators. op is an n-ary operator, defined over decision set $\mathcal{D}_3 = \{P, D, NA\}$ where P denotes *permit*, D denotes *deny* and NA that the policy is *not-applicable* to a given query. Here, we adopt the operators defined in PTaCL, which are presented in Table 1. These operators represent largely-used policy conflict resolution strategies like XACML operators *permit-overrides* ($\triangledown$), *deny-overrides* ($\triangle$) and *first-applicable* ($\triangleright$) as well as variations of these operators based on different interpretations of the *not-applicable* decision (we refer to [11] for a detail discussion on these interpretations and corresponding operators). 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 $\{\neg, \sim, \tilde{\sqcup}\}$ is canonically complete [27], *i.e.* any three-valued operator can be constructed using these three operators.

Given the set of policies $\mathcal{P}$, the set of access queries Q and decision set $\mathcal{D}_3$, we represent policy evaluation as a function $[\![\cdot]\!] : \mathcal{P} \times Q \rightarrow \mathcal{D}_3$ such that, given a query q and a policy p, $[\![p]\!](q)$ represents the decision of evaluating p against q. The evaluation of composite policies $\text{op}(p_1, \ldots, p_n)$ is obtained by combining the

[1]Here, we abstract from the specification of users' individual policies and focus on the way in which these policies are combined for the protection of co-owned resources.

d_1	d_2	$\neg d_1$	$\sim d_1$	$d_1 \tilde{\sqcap} d_2$	$d_1 \sqcap d_2$	$d_1 \triangle d_2$	$d_1 \tilde{\sqcup} d_2$	$d_1 \sqcup d_2$	$d_1 \nabla d_2$	$d_1 \rhd d_2$
P	*P*	*D*	*P*	*P*	*P*	*P*	*P*	*P*	*P*	*P*
P	*D*	*D*	*P*	*D*	*D*	*D*	*P*	*P*	*P*	*P*
P	*NA*	*D*	*P*	*NA*	*NA*	*P*	*P*	*NA*	*P*	*P*
D	*P*	*P*	*D*	*D*	*D*	*D*	*P*	*P*	*P*	*D*
D	*D*	*P*	*D*	*D*	*D*	*D*	*D*	*D*	*D*	*D*
D	*NA*	*P*	*D*	*D*	*NA*	*D*	*NA*	*NA*	*D*	*D*
NA	*P*	*NA*	*D*	*NA*	*NA*	*P*	*P*	*NA*	*P*	*P*
NA	*D*	*NA*	*D*	*D*	*NA*	*D*	*NA*	*NA*	*D*	*D*
NA	*NA*	*NA*	*D*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*

Table 1: Operators on decision set $\mathcal{D}_3 = \{P, D, NA\}$

evaluation of the policies forming that policy with respect to the semantics of the policy combining operators given in Table 1.

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 use an *additively* homomorphic cryptosystem, e.g., the *Paillier* cryptosystem [35]. We now briefly review the essential concepts behind additively homomorphic encryption and present the cryptographic building blocks used in this work.

2.2.1 *Additively Homomorphic Encryption.*

Additively homomorphic encryption allows for addition in the ciphertext domain while preserving the result of the operation in the plaintext domain. 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, m_2 be two messages and e a scalar value, both represented as integers. The homomorphism is defined as follows:

$$\mathcal{D}_{s_k}(\mathcal{E}_{p_k}(m_1) \cdot \mathcal{E}_{p_k}(m_2)) = m_1 + m_2,$$
$$\mathcal{D}_{s_k}(\mathcal{E}_{p_k}(m)^e) = e \cdot m.$$

Hereafter, we denote the encryption of a plaintext m, encrypted with a unique 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 when it is clear from the context and we simply write $[m]$.[2] Moreover, we use symbol $\oplus$ to denote homomorphic addition and $\ominus$ to denote homomorphic subtraction. Specifically, $[m_1] \oplus [m_2] = [m_1 + m_2]$ and $[m_1] \ominus [m_2] = [m_1 - m_2]$.

2.2.2 *HE Building Blocks.*

Additive homomorphic encryption enables us to perform addition and scalar multiplication operations without decrypting the ciphertexts. However, performing more complex operations in homomorphic encryption requires interactive two-party protocols. In such protocols, one party holds the secret key and helps the other party to perform intermediary computations and decryptions. Below we introduce four secure computation protocols that we use as building blocks for the construction of protocols for the secure evaluation of multi-party policies.

[2]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 cryptosytem, this is guaranteed by the fact that the encryption function is probabilistic.

Secure Equality Protocol: Secure equality is used to determine the equality between two encrypted values [31]. Given two ciphertexts $[a]$ and $[b]$, the secure equality test between $[a]$ and $[b]$ is defined as:

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

Secure Comparison Protocol: Secure comparison is used to compare two encrypted values [32]. Given two ciphertexts $[a]$ and $[b]$, the secure comparison between $[a]$ and $[b]$ is defined as:

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

Secure Multiplication Protocol: Secure multiplication is used to compute the multiplication between two encrypted values [17]. Given two ciphertexts $[a]$ and $[b]$, the secure multiplication of $[a]$ and $[b]$ is defined as:

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

Secure Matching Protocol: Secure matching is used to determine whether an encrypted value belongs to a set of encrypted values [20]. Given a ciphertext $[a]$ and a set of ciphertexts $\mathcal{B} = \{[b_1], \ldots, [b_n]\}$, the secure matching protocol is defined as:

$$[P(a, \mathcal{B})] = ([b_1] \ominus [a]) \otimes \ldots \otimes ([b_n] \ominus [a])$$

Intuitively, this protocol returns $[0]$ if $a \in \mathcal{B}$.

These building blocks can be combined for the definition of other protocols. For instance, they can be used to define a protocol for the secure computation of the XOR operation. Specifically, given two values $a, b \in \{0, 1\}$, the secure XOR protocol can be defined as:

$$[a]\ \mathtt{XOR}\ [b] = ([a] \oplus [b]) \ominus ([2] \otimes [a] \otimes [b]).$$

2.3 Secure Function Evaluation

Despite allowing computations in the ciphertext domain, homomorphic encryption is computationally expensive. *Secure function evaluation* (*SFE*) is an alternative method to homomorphic encryption for privacy-preserving computations. It enables several parties to compute a function on their private inputs without revealing any information apart from the result of the function. Secure function evaluation was introduced by Yao as a secure two-party computation to solve millionaires' problem such that Alice and Bob try to decide who is richer without revealing their wealth to each other [44]. Later, it was generalized to a multi-party setting in [22].

In this work, we implement secure function evaluation in two-party setting using the ABY framework [13]. ABY provides the constructions for $\underline{\text{A}}$rithmetic circuits [5], $\underline{\text{B}}$oolean circuits [22], and $\underline{\text{Y}}$ao's garbled circuits [44]. For the definition of our protocols, 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 is 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 $\langle x_i \rangle_1 \boxplus \langle x_i \rangle_2 \equiv x_i \mod 2$, where $\boxplus$ represents bitwise XOR operation. The result of the function is reconstructed by combining the secret shares obtained by each party through a bitwise XOR operation. For more details on the mechanism and implementation of Boolean circuits, we refer the reader to [13].

2.3.1 SFE Building Blocks. We adopt seven Boolean gates from the ABY framework as building blocks for the design of our SFE-based secure protocols. Next, we present these gates.

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

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

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:

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

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 $\mod 2^\ell$, 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:

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

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:

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

AND Gate: We perform a bitwise AND operation between two secret shared inputs using an AND ($\wedge$) gate. 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: We perform a bitwise OR operation between two secret shared inputs using an OR ($\vee$) gate. Given the secret shared inputs $\langle a\rangle$ and $\langle b\rangle$, the OR gate is defined as:

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

3 A PRIVACY-PRESERVING FRAMEWORK FOR MULTI-PARTY ACCESS CONTROL

This section illustrates the challenges in multi-party access control through a running example within the social network domain and presents an overview of our framework to address these challenges.

3.1 Motivating Example

Consider an online social network that provides users with a platform to host and share their contents and to build communities of users with common interests. Users can upload contents on their profile as well as on the profile of other users. The contents uploaded by a user can be about the user herself or about other users. Following the data governance model in [29], we identify three main stakeholders based on their relation with the contents: the *data host* (H), which is the user in whose profile the contents are posted; the *data provider* (P), which is the user who uploaded the contents; and the *data subjects* (S), which are the users to whom the contents refer. The contents can refer to multiple data subjects.

Users can specify privacy preferences that define who is authorized to access their contents; however, their preferences can conflict with the preferences of other users. To enable the collaborative management of contents, the social network should employ a multi-party access control policy that combines the privacy preferences of individual users while accounting for their level of authority over the contents. The level of authority that users have over the contents typically depends on their relation with the contents [12]. The data governance model in [29] allows one to explicitly reason on the level of authority that users have based on their relation with the contents and to specify a multi-party policy by capturing the different levels of authority using the combining operators in Table 1. Combining operators dictate how policy conflicts are resolved and their choice is influenced, for instance, by the requirements imposed by privacy and data protection regulations.

In our scenario, we assume that the social network adopts the following (abstract) multi-party access control policy:

$$p = (s_{S_1} \vartriangle \ldots \vartriangle s_{S_m}) \rhd (s_H \vartriangle s_P) \rhd s_{SN}$$

where $s_{S_1},\ldots,s_{S_m}, s_H, s_P$ are placeholders denoting the privacy preference of the data subjects $S_1,\ldots,S_m$, data host H and data provider P, respectively. s_{SN} is a default policy used by the social network to handle the situation in which none of the user preferences is applicable. Intuitively, the multi-party policy states that the privacy preferences of the data subjects have priority over the ones of the data host and data provider.[3] In turn, the preferences of the data host and data provider have priority over the social network's default policy. The preferences of the data subjects (and the ones of the data host and data provider) are combined using the deny-overrides operator ($\vartriangle$) that returns a deny decision if at least one of the user policies denies access (we refer to Table 1 for the exact definition of the operator).

Suppose that a user – Alice – uploads a photo representing some friends – Carly and David – on the profile of another user – Bob. Each of these users defines his/her privacy preferences stating which users can view the photo. For the sake of exemplification, we illustrate their privacy preferences using a simple policy language but our approach is applicable to any policy language that supports the specification of both positive and negative authorizations.

We model the privacy preferences of a user as a pair (X, Y), where X represents the set of users to which access is granted and Y represents the set of users to which access is denied. Let

[3]This is in line with most data protection regulations, which empower data subjects regarding their own personal data [23].

$s_o^u = (X_o^u, Y_o^u)$ be the privacy preferences of a user u regulating the access to a data object o. Given an access query q made by a user r for o, the evaluation of q against s_o^u is defined as follows:

$$[\![s_o^u]\!](q) = \begin{cases} P & \text{if } r \in X_o^u \setminus Y_o^u \\ D & \text{if } r \in Y_o^u \\ NA & \text{otherwise} \end{cases} \quad (1)$$

Let us assume that the privacy preferences of Alice, Bob, Carly and David for the photo are defined as follows:

$$s_{photo}^{Alice} = (Public, \emptyset) \qquad s_{photo}^{Bob} = (Colleagues, \{Evelyn, Hope\})$$
$$s_{photo}^{Carly} = (Friends, \emptyset) \qquad s_{photo}^{David} = (Friends, \{Grace\})$$

where *Friends* is used as a shortening to represent the set of friends of a given user; similarly, *Colleagues* represents the set of colleagues. *Public* denotes that access is granted to all users.

Grace, a friend of Carly and David, accesses the profile of Bob, who is her colleague. To determine whether Grace is allowed to view the photo, the social network instantiates the multi-party policy with the privacy preferences of the users involved in the management of the photo. Specifically:

$$p_{photo} = (s_{photo}^{Carly} \vartriangle s_{photo}^{David}) \rhd (s_{photo}^{Bob} \vartriangle s_{photo}^{Alice}) \rhd s_{SN}$$

The instantiated policy is evaluated with respect to a query q asking whether Grace is allowed to view the photo. First, the individual privacy preferences are evaluated, obtaining:

$$[\![s_{photo}^{Alice}]\!](q) = P \qquad [\![s_{photo}^{Bob}]\!](q) = P$$
$$[\![s_{photo}^{Carly}]\!](q) = P \qquad [\![s_{photo}^{David}]\!](q) = D$$

Based on these evaluations (and assuming that the social network's default policy always grants access), we obtain:

$$[\![p_{photo}]\!](q) = (P \vartriangle D) \rhd (P \vartriangle P) \rhd P = D$$

Thus, Grace is not allowed to view the photo. This is because David denied her access.

In order for our scenario to work as intended, users should be able to freely define their privacy preferences. For instance, David might not want Grace to learn that he did not give her the permission to view the photo. If this is not possible, he could require the removal of the photo from the social network, thus reducing his willingness to share new contents. Moreover, privacy preferences can reveal interpersonal relationships with other users, which a user might want to keep private. For instance, Carly might not want that other users know he has a friendship relation with Grace. Even if their privacy settings are not made visible to other users, users might prefer to minimize the information to be disclosed to the social network. In particular, they might not want to disclose their interpersonal relations (contained in their privacy preferences) as this information could be used for profiling and targeted advertising [15].

In this work, we focus on the protection of user policies and propose a framework that allows users to disclose their privacy preferences in a private form while the social network still remains capable of making access decisions based on user preferences.

3.2 Framework

To enable the evaluation of multi-party policies while protecting the confidentiality of user policies, we design an access control mechanism that supports policy evaluation on private inputs. An overview of our mechanism is presented in Figure 1.

Our framework comprises four main entities:

- *Data holders* (●) share their resources and data along with policies for their protection and want their policies to remain private.
- *Access requester* (●) requests access to resources and data.
- *Data Server (DS)* stores the data holders' resources and is responsible for their protection (the social network in our example). The Data Server evaluates data holders' policies to determine whether access to their resources should be granted.
- *Semi Trusted Party (STP)* is a semi-honest entity that assists DS in the secure evaluation of data holders' policies.

The data holders provide their policies in a private form (represented by $(s_1), \ldots, (s_n)$ in Figure 1) to DS. 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 in order for a decision to be made. This task is performed by DS together with STP.

The 'private form' in which policies are provided and the protocols used for policy evaluation are dictated by the underlying privacy-preserving technique. In HE, STP generates public (p_k) and private (s_k) keys and sends the public key p_k to the data holders and to DS. Data holders encrypt their policies using the public key received by STP and send the encrypted policies to DS. On the other hand, in SFE users generate two secret shares of their policies and provide one share to DS and one to STP. Upon receiving an access request, DS's *policy evaluation point* evaluates the request against each user policy in private form with the assistance of STP, resulting in an access decision (in private form) for each policy (represented by $(d_1), \ldots, (d_n)$ in Figure 1).

The *policy combination point* evaluates the multi-party policy by combining the (protected) access decisions resulting from the evaluation of user policies. We propose secure computation protocols that enable the policy combination point to perform policy evaluation over policies in private form (see Section 4). These protocols implement the combining operators in Table 1 and can be used as building blocks for the secure evaluation of any multi-party policy expressed using those operators. Once the multi-party policy has been evaluated (with the assistance of STP), the policy combination 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, DS should derive the decision in plaintext from the decision in private form returned by the policy combination point. Yet, how this step is performed depends on the underlying privacy-preserving technology. In SFE, DS can derive the decision in plaintext by recombining the secret share obtained by evaluating the multi-party policy with the one obtained by STP (see Section 2.3). Deriving the decision in plaintext is trickier in HE. DS should not have access to the decryption key; otherwise, it will be able to learn data holders' policies. In order for DS to decrypt the encrypted decision without STP learning it, DS adds random noise r to the encrypted decision $[d]$ and sends $[d + r]$ to STP. STP decrypts the ciphertext and sends $d + r$ to DS. DS can obtain the decision to be enforced by removing the added noise r from $d + r$.

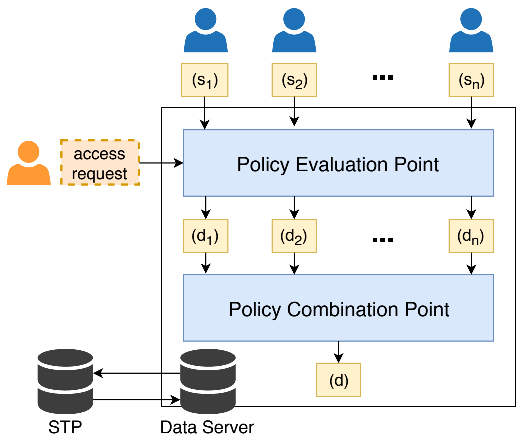

Figure 1: Architecture

3.3 Security Assumptions

We assume a semi-honest security model [21] where all participants are assumed to be honest-but-curious. It means 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.

With respect to the semi-honest security assumption, our goal is to design protocols that provide security against honest-but-curious non-colluding DS and STP. Accordingly, the policies provided by data holders should be kept hidden from DS and STP. They can only see the encryption of the policies (if HE is used) or their private shares of the policies (if SFE is used). At the end of policy evaluation, the decision should be revealed to DS (as it has to enforce the decision) but not to STP as this entity does not need to know whether access was granted to the requester or not (minimality principle [23]).

We do not include data holders and access requester in the security assumptions since they do not actively contribute to the computations. We assume the data holders provide the correct protected inputs (*i.e.*, their policies) to the corresponding parties at the beginning of the protocol execution. Also, we assume access requester can only observe the final decision of policy evaluation.

The non-colluding two-server setting is used to reduce the workload on the client side. Without the use of a second server, all computations have to be performed between the client and DS. This is, however, not desirable because it requires the client to have enough computational resources and also to be online during computations. Finally, we assume that all parties communicate over an authenticated channel. This assumption aims to prevent attacks coming from outside the framework.

4 PRIVACY-PRESERVING PROTOCOLS FOR POLICY EVALUATION

For the realization of a practical mechanism able to evaluate multi-party access control policies while preserving the confidentiality of user policies, we need efficient secure computation protocols that compute the policy combining operators in Table 1 on private inputs. To this end, we have investigated how to design such protocols using two well-known privacy-preserving techniques, namely (additively) homomorphic encryption (HE) and secure function evaluation (SFE). We also investigate alternative encodings for the specification of the three-valued logic operators in Table 1.

This section presents three alternative privacy-preserving realizations of the secure computation protocols, leveraging *(i)* homomorphic encryption, *(ii)* SFE based on a direct encoding of three-valued logic operators and *(iii)* SFE based on a Boolean logic encoding of the operators. These protocols serve as building blocks and can be used to evaluate any multi-party policy built over the operators in Table 1. We evaluate the proposed protocols through an experimental analysis in Section 6.

4.1 HE-based Protocols

The HE-based protocols are defined over the four cryptographic building blocks presented in Section 2.2.2 – equality test, comparison, multiplication and matching – along with homomorphic addition and subtraction. Before presenting the protocols, recall that homomorphic encryption operates on integer numbers. Thus, we need to represent the decisions in $\mathcal{D}_3$ as integer numbers. For the design of our protocols, we map access decisions D, NA and P to 0, 1 and 2, respectively. It is worth noting that this encoding leads to a total order over access decisions where $P > NA > D$.

For the evaluation of user policies as defined in Eq. 1 (Section 3.1), we use the secure matching protocol to determine whether the requester belongs to the set of allowed and prohibited users. To obtain an output suitable for policy evaluation, we have adapted the protocol as follows. Given an encrypted value $[a]$ and a set of encrypted values $\mathcal{B} = \{[b_1], \dots, [b_n]\}$, we define

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

Intuitively, this protocol returns [1] if $a \in \mathcal{B}$ and [0] otherwise. We use this protocol for the evaluation of user policies. Let $s = (X, Y)$ be a user policy, where X and Y are finite sets of user IDs denoting the users allowed and denied to access a given object, respectively. We assume that a user encrypts her policy by encrypting every ID individually.[4] We denote the encrypted policy as $[s]$. Note that, if a user wishes to make an object publicly available, denoted by *Public* in the example of Section 3.1, the corresponding access constraint does not require listing the IDs of all users within the social network, but it can be simply encoded with a simple condition that always holds or omitted. Thus, its verification does not introduce any computational burden.

Given an encrypted user policy $[s]$ (with $s = (X, Y)$) and an access request q consisting of the ID of the requester r, we define the protocol for the evaluation of user policies as:

$$eval^H([s], q) = ([1] \ominus [r \overset{?}{\in} Y]) \otimes ([1] \oplus [r \overset{?}{\in} X])$$

It is easy to verify that $eval^H([s], q) = [d]$ iff $d = [\![s]\!](q)$.

We now present the protocols for the combining algorithms in Table 1. As the set of operators $\{\neg, \sim, \widetilde{\sqcup}\}$ is canonically complete,

[4]To limit the information that can be inferred from the policy, fix-length lists can be used for the specification of the users for which access is granted of denied. Empty slots can be filled with 'dummy' values that do not match any user ID.

Operator	Protocol	*Definition*
$\neg a$	$not^H([a])$	$([a \overset{?}{=} 1] \otimes [a]) \oplus (([1] \ominus [a \overset{?}{=} 1]) \otimes ([2] \ominus [a]))$
$\sim a$	$wea^H([a])$	$[a \overset{?}{=} 2] \otimes [a]$
$a \widetilde{\sqcup} b$	$smax^H([a],[b])$	$(([1] \ominus [a \overset{?}{\leq} b]) \otimes [a]) \oplus ([a \overset{?}{\leq} b] \otimes [b])$
$a \widetilde{\sqcap} b$	$smin^H([a],[b])$	$([a \overset{?}{\leq} b] \otimes [a]) \oplus (([1] \ominus [a \overset{?}{\leq} b]) \otimes [b])$
$a \sqcup b$	$wmax^H([a],[b])$	$([a \overset{?}{=} 1] \ \mathtt{XOR}\ [b \overset{?}{=} 1]) \oplus \Big(([1] \ominus ([a \overset{?}{=} 1] \ \mathtt{XOR}\ [b \overset{?}{=} 1])) \otimes smax^H([a],[b])\Big)$
$a \sqcap b$	$wmin^H([a],[b])$	$([a \overset{?}{=} 1] \ \mathtt{XOR}\ [b \overset{?}{=} 1]) \oplus \Big(([1] \ominus ([a \overset{?}{=} 1] \ \mathtt{XOR}\ [b \overset{?}{=} 1])) \otimes smin^H([a],[b])\Big)$
$a \nabla b$	$po^H([a],[b])$	$([a \overset{?}{=} 1] \otimes [b \overset{?}{=} 1]) \oplus \Big((([a \overset{?}{=} 2] \otimes [b \overset{?}{=} 2]) \oplus ([a \overset{?}{=} 2] \ \mathtt{XOR}\ [b \overset{?}{=} 2])) \otimes [2]\Big)$
$a \triangle b$	$do^H([a],[b])$	$([a \overset{?}{=} 1] \otimes [b \overset{?}{=} 1]) \oplus \Big((([a \overset{?}{=} 2] \otimes [b \overset{?}{=} 2]) \oplus ([a \overset{?}{=} 2] \otimes [b \overset{?}{=} 1]) \oplus ([a \overset{?}{=} 1] \otimes [b \overset{?}{=} 2])) \otimes [2]\Big)$
$a \rhd b$	$fa^H([a],[b])$	$(([1] \ominus [a \overset{?}{=} 1]) \otimes [a]) \oplus ([a \overset{?}{=} 1] \otimes [b])$

Table 2: HE-based protocols

we first propose secure computation protocols for these operators. The definition of these protocols is given at the top of Table 2.

Negation protocol (not^H): Given an encrypted decision $[a]$, protocol $not^H([a])$ computes the negation of a (Table 1) under encryption, denoted by $[\neg a]$. The protocol consists of two terms: the first term (*i.e.*, $[a \overset{?}{=} 1] \otimes [a]$) is used to deal with the case where a is the *non-applicable* decision and the second term (*i.e.*, $([1] \ominus [a \overset{?}{=} 1]) \otimes ([2] \ominus [a])$) with the case where a is *permit* or *deny*. Intuitively, if a is 1 (*i.e., non-applicable*), the second term evaluates $[0]$ because of $([1] \ominus [a \overset{?}{=} 1])$ and the first term evaluates $[1]$. Then, $[0] \oplus [1] = [1]$, *i.e.* the *non-applicable* decision (under encryption). On the other hand, if a is 2 (*i.e., permit*), the first term evaluates $[0]$, and the second term can be reduced to $[2] \ominus [2]$, which gives $[0]$, *i.e. deny*. The homomorphic addition of these two terms gives $[0]$. Similar reasoning applies when a is 0.

Weakening protocol (wea^H): Given an encrypted decision $[a]$, protocol $wea^H([a])$ computes the weakening of a (Table 1) under encryption, denoted by $[\sim a]$. Intuitively, if a is 2, the comparison protocol $[a \overset{?}{=} 2]$ returns $[1]$ and, thus, the weakening protocol returns $[2]$. Otherwise, if a is 0 or 1, the comparison protocol returns $[0]$ and so the weakening protocol returns $[0]$.

Strong disjunction protocol ($smax^H$): Given two encrypted decisions $[a]$ and $[b]$, the strong disjunction protocol $smax^H([a],[b])$ computes $a \widetilde{\sqcup} b$ (Table 1) under encryption. It is worth noting that this operator corresponds to the *max* operator with respect to total order $P > NA > D$. Accordingly, the strong disjunction protocol computes the maximum between a and b in encrypted form. If a is strictly greater than b, the first term (*i.e.*, $([1] \ominus [a \overset{?}{\leq} b]) \otimes [a]$) returns $[a]$ whereas the second term (*i.e.*, $[a \overset{?}{\leq} b] \otimes [b]$) returns $[0]$, and thus the protocol returns $[a]$. Otherwise, if a is lower than (or equal to) b, the first term returns $[0]$ and the second $[b]$, and thus the protocol returns $[b]$.

Since the set of operators $\{\neg, \sim, \widetilde{\sqcup}\}$ is canonically complete, the protocols above can be used as building blocks for the definition of secure computation protocols for the other operators in Table 1. For instance, strong conjunction can be express as $a \widetilde{\sqcap} b = \neg(\neg a \widetilde{\sqcup} \neg b)$ and, thus protocol $not^H(smax^H(not^H([a]), not^H([b]))$ can be used to securely compute strong conjunction of $[a]$ and $[b]$. However, the resulting protocols are computationally expensive and require significant bandwidth usage. Thus, we have designed specific protocols for operators $\widetilde{\sqcap}, \sqcup, \sqcap, \triangle, \nabla, \rhd$ (Table 2), which minimizes the number of building blocks used. Next, we present these protocols.

Strong conjunction protocol ($smin^H$): Given two encrypted decisions $[a]$ and $[b]$, protocol $smin^H([a],[b])$ computes $a \widetilde{\sqcap} b$ (Table 1) in encrypted form. This operator corresponds to the *min* operator with respect to total order $P > NA > D$. Thus, protocol $smin^H$ is the dual of protocol $smax^H$.

Weak disjunction protocol ($wmax^H$): Given two encrypted decisions $[a]$ and $[b]$, protocol $wmax^H([a],[b])$ computes $a \sqcup b$ (Table 1) under encryption. Basically, this operator returns *non-applicable* if at least one of a and b is *non-applicable*. Otherwise, the maximum between a and b is returned. We designed protocol $smax^H$ per cases using the `XOR` protocol (we refer to Section 2.2 for its definition). If either a or b is 1 (*i.e., non-applicable*), term $([a \overset{?}{=} 1] \ \mathtt{XOR}\ [b \overset{?}{=} 1])$ evaluates $[1]$, and the second term (*i.e.*, $[1] \ominus ([a \overset{?}{=} 1] \ \mathtt{XOR}\ [b \overset{?}{=} 1]) \otimes smax^H([a],[b])$) returns $[0]$. If both a and b are 1, or neither a nor b is 1, then $([a \overset{?}{=} 1] \ \mathtt{XOR}\ [b \overset{?}{=} 1])$ evaluates $[0]$, and consequently the second term returns $smax^H([a],[b])$, *i.e.*, the maximum between a and b in encrypted form.

Weak conjunction protocol ($wmin^H$): Given two encrypted decisions $[a]$ and $[b]$, protocol $wmin^H([a],[b])$ computes $a \sqcap b$ (Table 1) under encryption. This operator differs from weak disjunction for the fact that when none of the decisions (or both) is (are) *non-applicable*, then the minimum is returned in encrypted form. Based on this observation, protocol $wmin^H$ deals with the case where at least one of a and b is 1 as in protocol $wmax^H$, *i.e.* the second terms becomes $[0]$. On the other hand, the two protocols differ for the last term, in which $smax^H([a],[b])$ is replaced with $smin^H([a],[b])$.

Deny-override (do^H): Given two encrypted decisions $[a]$ and $[b]$, protocol $do^H([a],[b])$ computes $a \triangle b$ (Table 1) under encryption. This operator returns *deny* whenever one of its arguments is *deny* and returns *permit* if at least one of its arguments is *permit* and none is *deny*; otherwise, if both arguments are *not-applicable*, it returns *not-applicable*. Protocol do^H follows this intuition. Specifically, the first term of protocol do^H (*i.e.*, $[a \overset{?}{=} 1] \otimes [b \overset{?}{=} 1]$) is used to check if both decisions are *not-applicable*. If this is the case, it returns $[1]$; also, it is easy to verify that in this case the second term returns $[0]$. Thus, the protocol returns $[1]$. On the other hand, the second term deals with the case where at least one of a and b is 2 (*i.e.*, *permit*) and both are not 0 (*i.e.*, *deny*). In this case, the first term evaluates $[0]$ and the second term reduces to $[1] \otimes [2]$, which gives $[2]$. In all other cases, *i.e.*, when at least one of a and b is 0, the protocol returns $[0]$.

Permit-override (po^H): Given two encrypted decisions $[a]$ and $[b]$, protocol $po^H([a],[b])$ computes $a \nabla b$ (Table 1) under encryption. This operator is similar to deny-override but in this case *permit* has precedence over *deny*. Accordingly, protocol po^H uses a similar structure of protocol do^H. However, note that the second term of po^H checks whether at least one of the arguments is $[2]$. This is done by checking if both a and b are 2 (using $[a \overset{?}{=} 2] \otimes [b \overset{?}{=} 2]$) and if exactly one of a and b is 2 using the `XOR` protocol.

First-applicable (fa^H): Given two encrypted decisions $[a]$ and $[b]$, protocol $fa^H([a],[b])$ computes $a \triangleright b$ (Table 1) under encryption. This operator evaluates its arguments in the order in which they are specified and returns the decision of the first policy that differs from *not-applicable*. Accordingly, the first term of protocol fa^H (*i.e.*, $([1] \ominus [a \overset{?}{=} 1]) \otimes [a]$) checks whether a is different from 1. If this is the case, the first term evaluates $[a]$; it is also trivial to see that, in this case, the second term evaluates $[0]$. Since $[a] \oplus [0] = [a]$, the protocol returns $[a]$, *i.e.*, the first argument. Otherwise, if a is 1 (*i.e.*, *not-applicable*), the first term evaluates $[0]$ and the second term evaluates $[b]$. Since $[0] \oplus [b] = [b]$, the protocol returns $[b]$, *i.e.*, the second argument.

4.2 SFE-based Protocols (Three Valued Logic)

As an alternative to homomorphic encryption, we design protocols that implement the evaluation of multi-party access control policies using secure function evaluation. Similar to HE-based protocols, we use three-valued logic for the representation of the decisions and use 0, 1, and 2 to represent D, NA and P, respectively. We use the SFE building blocks in Section 2.3.1 for the design of the protocols.

For the evaluation of user policies, we design a protocol similar to the one used for homomorphic encryption. Given a secret shared input $\langle a \rangle$ and a set of secret shared inputs $\mathcal{B} = \{\langle b_1 \rangle, \ldots, \langle b_n \rangle\}$, the SFE secure matching protocol can be defined as:

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

Then, we define $\langle a \overset{?}{\in} \mathcal{B} \rangle = \langle P(a, \mathcal{B}) \overset{?}{=} 0 \rangle$ to check whether a secret shared input a belongs to the set of secret shared inputs $\mathcal{B}$. The secret share of a user policy $s = (X, Y)$, denoted as $\langle s \rangle$, is obtained from s by secret sharing every element of X and Y. Given a secret shared policy $\langle s \rangle$ and an access control request q consisting of the user ID of the requester r, policy evaluation can be performed using the following protocol:

$$eval^T(\langle s \rangle, q) = (\langle 1 \rangle - \langle r \overset{?}{\in} Y \rangle) \times (\langle 1 \rangle + \langle r \overset{?}{\in} X \rangle)$$

Yet, it is easy to verify based on Eq. 1 that $eval^T(\langle s \rangle, q) = [d]$ iff $d = [\![s]\!](q)$.

For the evaluation of multi-party policy, we designed SFE-based protocols implementing the operators in Table 1, as shown in Table 3. Compared to HE-based protocols, SFE-based protocols are less complex. The reason is that SFE provides the implementation of logic gates like inverse, AND, OR, etc., whereas under HE these gates should be designed using additions and multiplications.

Negation (not^T): Given the secret shares of a decision $\langle a \rangle$, protocol $not^T(\langle a \rangle)$ computes its negation, which is represented as $\langle \neg a \rangle$. We do not use an inverse gate for the computation of the negation operator. The reason is that, when we compute the negation $\mod 2^2$, the inverse of 0, 1, and 2 become 3, 2, and 1 despite the correct values should be 2, 1, and 0. Instead, we make use of equality check and subtraction gates. Accordingly, first the protocol checks whether the value of secret shared input $\langle a \rangle$ is $\langle 1 \rangle$. If so, it maps it to itself with a multiplication gate. Otherwise, it performs a subtraction operation $\langle 2 \rangle - \langle a \rangle$. The result of subtraction maps $\langle 0 \rangle$ to $\langle 2 \rangle$ and $\langle 2 \rangle$ to $\langle 0 \rangle$.

Weakening (wea^T): Protocol $wea^T(\langle a \rangle)$ computes the weakening operator $\langle \sim a \rangle$ on the secret shares of a decision a. This operator returns *deny* for decisions *deny* and *not-applicable*, and leave decision *permit* unchanged. Thus, using an SFE equality gate, $wea^T(\langle a \rangle)$ protocol secretly checks whether the decision is $\langle 2 \rangle$. If so, it returns the secret shared decision itself, *i.e.* $\langle 2 \rangle$, otherwise it returns $\langle 0 \rangle$.

Strong Disjunction ($smax^T$): Given the secret shares of two decisions $\langle a \rangle$ and $\langle b \rangle$, protocol $smax^T(\langle a \rangle, \langle b \rangle)$ computes secret shares $\langle a \,\widetilde{\sqcup}\, b \rangle$. As stated previously, strong disjunction performs a maximum operation with respect to total order $P > NA > D$. Using the ABY library, we can compute the maximum of two secret shared decisions using an OR ($\vee$) gate.

Strong Conjunction ($smin^T$): Given the secret shares of two decisions $\langle a \rangle$ and $\langle b \rangle$, protocol $smin^T(\langle a \rangle, \langle b \rangle)$ computes secret shares $\langle a \,\widetilde{\sqcap}\, b \rangle$. As the opposite of strong disjunction, the protocol performs a minimum operation with respect to total order $P > NA > D$. Under SFE, we can compute the minimum of two secret shared decisions using an AND ($\wedge$) gate.

Weak Disjunction ($wmax^T$): Given the secret shares of two decisions $\langle a \rangle$ and $\langle b \rangle$, protocol $wmax^T(\langle a \rangle, \langle b \rangle)$ computes secret shares $\langle a \sqcup b \rangle$. If any of the decisions is *not-applicable* (encoded as 1), the protocol returns $\langle 1 \rangle$, otherwise it returns the maximum of the two secret shared decisions. Accordingly, $wmax^T$ checks if any of the decisions is *not-applicable* using the equality check functions $\langle a \overset{?}{=} 1 \rangle$ and $\langle b \overset{?}{=} 1 \rangle$. If the equality check returns true at least for one of the decisions, then the second part of the definition $(\langle 1 \rangle - (\langle a \overset{?}{=} 1 \rangle \vee \langle a \overset{?}{=} 1 \rangle)) \times smax^T(\langle a \rangle, \langle b \rangle)$ becomes $\langle 0 \rangle$ and the protocol returns $\langle 1 \rangle$. Otherwise, it returns the result of $smax^T(\langle a \rangle, \langle b \rangle)$.

Weak Conjunction ($wmin^T$): Given secret shared decisions $\langle a \rangle$ and $\langle b \rangle$, protocol $wmin^T(\langle a \rangle, \langle b \rangle)$ computes secret shares $\langle a \sqcap b \rangle$. The protocol works similarly to the weak disjunction protocol. Different from $wmax^T$, weak conjunction returns $smin^T(\langle a \rangle, \langle b \rangle)$ when none of the secret shared decisions are *not-applicable*.

Permit-override (po^T): Given secret shared decisions $\langle a \rangle$ and $\langle b \rangle$, protocol $po^T(\langle a \rangle, \langle b \rangle)$ computes secret shares $\langle a \nabla b \rangle$. The protocol checks if any of the secret shared decisions is $\langle 2 \rangle$, in which case returns $\langle 2 \rangle$. Otherwise, it returns the minimum between $\langle a \rangle$ and $\langle b \rangle$.

Operator	**Protocol**	*Definition*
$\neg a$	$not^T(\langle a\rangle)$	$\langle a \overset{?}{=} 1\rangle \times \langle a\rangle + (\langle 1\rangle - \langle a \overset{?}{=} 1\rangle) \times (\langle 2\rangle - \langle a\rangle)$
$\sim a$	$wea^T(\langle a\rangle)$	$\langle a \overset{?}{=} 2\rangle \times \langle a\rangle$
$a \mathbin{\widetilde{\sqcup}} b$	$smax^T(\langle a\rangle, \langle b\rangle)$	$\langle a\rangle \vee \langle b\rangle$
$a \mathbin{\widetilde{\sqcap}} b$	$smin^T(\langle a\rangle, \langle b\rangle)$	$\langle a\rangle \wedge \langle b\rangle$
$a \sqcup b$	$wmax^T(\langle a\rangle, \langle b\rangle)$	$(\langle a \overset{?}{=} 1\rangle \vee \langle a \overset{?}{=} 1\rangle) + (\langle 1\rangle - (\langle a \overset{?}{=} 1\rangle \vee \langle a \overset{?}{=} 1\rangle)) \times smax^T(\langle a\rangle, \langle b\rangle)$
$a \sqcap b$	$wmin^T(\langle a\rangle, \langle b\rangle)$	$(\langle a \overset{?}{=} 1\rangle \vee \langle a \overset{?}{=} 1\rangle) + (\langle 1\rangle - (\langle a \overset{?}{=} 1\rangle \vee \langle a \overset{?}{=} 1\rangle)) \times smin^T(\langle a\rangle, \langle b\rangle)$
$a \nabla b$	$po^T(\langle a\rangle, \langle b\rangle)$	$(\langle a \overset{?}{=} 2\rangle \vee \langle b \overset{?}{=} 2\rangle) \times \langle 2\rangle + (\langle 1\rangle - (\langle a \overset{?}{=} 2\rangle \vee \langle b \overset{?}{=} 2\rangle)) \times smin^T(a, b)$
$a \vartriangle b$	$do^T(\langle a\rangle, \langle b\rangle)$	$(\langle 1\rangle - (\langle a \overset{?}{=} 0\rangle \vee \langle b \overset{?}{=} 0\rangle)) \times smax^T(a, b)$
$a \vartriangleright b$	$fa^T(\langle a\rangle, \langle b\rangle)$	$\langle a \overset{?}{=} 1\rangle \times \langle b\rangle + (\langle 1\rangle - \langle a \overset{?}{=} 1\rangle) \times \langle a\rangle$

Table 3: SFE-based protocols over three-valued logic encoding

Deny-override (do^T): Given secret shared decisions $\langle a\rangle$ and $\langle b\rangle$, protocol $do^T(\langle a\rangle, \langle b\rangle)$ computes secret shares $\langle a \vartriangle b\rangle$. It checks if none of the secret shared decisions is $\langle 0\rangle$, in which case returns the maximum between $\langle a\rangle$ and $\langle b\rangle$. Otherwise, it returns $\langle 0\rangle$.

First-applicable (fa^T): Given secret shared decisions $\langle a\rangle$ and $\langle b\rangle$, protocol $fa^T(\langle a\rangle, \langle b\rangle)$ computes secret shares $\langle a \vartriangleright b\rangle$. The protocol checks if a is *not-applicable*, in which case it returns $\langle b\rangle$; otherwise it returns $\langle a\rangle$.

4.3 SFE-based Protocols (Boolean Logic)

The protocols in Table 3 use Boolean gates for addition, subtraction, multiplication and equality test, which usually have high computation and communication costs compared to the inverse, AND and OR gates. Therefore, inspired by previous work [30, 42], we provide an alternative representation of the three-valued operators in Table 1 and represent them as Boolean formulae.

To transform policies defined over three-valued logic into an equivalent Boolean logic representation, we adapt and extend the transformation rules proposed in [30]. These rules can be used to transform a policy formula built over the operators in Table 1 into a triple of propositional formulae (π_P, π_D, π_{NA}) that can be trivially represented as Boolean circuits. Intuitively, propositional formula $\pi_d(p)$ denotes the set of queries $Q_d \subseteq Q$ satisfying $d = [\![p]\!](q)$ whenever $q \in Q_d$.

The base case comprises user policies. Specifically, given a user policy $s = (X, Y)$ and the ID of the requester r, from Eq. 1 we obtain:

$$\begin{aligned}
\pi_P(s) &= (r \in X) \wedge \neg(r \in Y) \\
\pi_D(s) &= r \in Y \\
\pi_{NA}(s) &= \neg(r \in X) \wedge \neg(r \in Y)
\end{aligned}$$

The corresponding secure computation protocols can be implemented using protocol $\langle r \overset{?}{\in} \mathcal{B}\rangle$ defined in Section 4.2.

The transformation rules for the policy combining operators in Table 1 are presented in Table 4. Since operations are performed on a single bit, arithmetic operations are not needed in the implementation of the operators. We can implement all operators using inverse gates ($\neg$), AND gates ($\wedge$) and OR gates ($\vee$) along the lines of the formulae in Table 4. We use modulus 2 ($\ell = 1$) for the implementation of these protocols.

5 SECURITY ANALYSIS

This section provides a security analysis of our framework for multi-party access control (Section 3.2) and of the underlying secure computation protocols (Section 4). We design our multi-party access control framework to be secure against semi-honest adversaries. In this security model, the participants execute the protocols without deviation. However, they might try to obtain additional information by observing inputs, outputs and intermediary messages. Accordingly, in our design, we assume both STP and DS to be semi-honest non-colluding parties that try to obtain additional information from the execution of the protocols without any deviation. Since data holders do not actively participate in computations, we do not include them into the security analysis. We assume that data holders provide the protected data (*i.e.*, encrypted data for HE-based protocols and secret-shared data for SFE-based protocols) to the intended parties at the beginning of the protocol execution.

We propose three privacy-preserving approaches for the evaluation of multi-party policies, which are based on two different cryptographic techniques, homomorphic encryption and secure function evaluation. We choose Paillier's cryptosystem for the computations of homomorphic encryption based operators. The security of the Paillier cryptosystem is based on computational difficulty of solving the decisional composite residuosity assumption [35]. We use four building blocks to compute complex operations under homomorphic encryption, namely secure equality [31], secure comparison [32], secure multiplication [17] and secure matching [20] protocols. These building blocks are proven secure in the presence of semi-honest adversaries. We refer readers to the corresponding papers for the details of the security analysis for each building block.

For the evaluation of multi-party policies under secure function evaluation, we use Boolean circuits [22]. The proposal of Boolean secret sharing in [22] provides information theoretic security against semi-honest adversaries and malicious adversaries in the existence of honest majority. For our protocols, we use the semi-honest variant of Boolean circuits whose implementation is provided in the ABY framework [13]. The building blocks we use for SFE are also functionalities provided by ABY. Thus, for the details of secure implementation and security proofs, we refer readers to [13, 22].

In the design of our protocols, we use a combination of the building blocks that are proven secure. According to the universally

Operator	π_P	π_D	π_{NA}
$\neg a$	$\pi_D(a)$	$\pi_P(a)$	$\pi_{NA}(a)$
$\sim a$	$\pi_P(a)$	$\pi_D(a) \vee \pi_{NA}(a)$	$false$
$a \mathbin{\widetilde{\sqcup}} b$	$\pi_P(a) \vee \pi_P(b)$	$\pi_D(a) \wedge \pi_D(b)$	$(\pi_{NA}(a) \wedge \neg\pi_P(b)) \vee (\pi_{NA}(b) \wedge \neg\pi_P(a))$
$a \mathbin{\widetilde{\sqcap}} b$	$\pi_P(a) \wedge \pi_P(b)$	$\pi_D(a) \vee \pi_D(b)$	$(\pi_{NA}(a) \wedge \neg\pi_D(b)) \vee (\pi_{NA}(b) \wedge \neg\pi_D(a))$
$a \sqcup b$	$(\pi_P(a) \wedge \neg\pi_{NA}(b)) \vee (\pi_P(b) \wedge \pi_{NA}(a))$	$\pi_D(a) \wedge \pi_D(b)$	$\pi_{NA}(a) \vee \pi_{NA}(b)$
$a \sqcap b$	$\pi_P(a) \wedge \pi_P(b)$	$(\pi_D(a) \wedge \neg\pi_{NA}(b)) \vee (\pi_D(b) \wedge \neg\pi_{NA}(a))$	$\pi_{NA}(a) \vee \pi_{NA}(b)$
$a \nabla b$	$\pi_P(a) \vee \pi_P(b)$	$(\pi_D(a) \wedge \neg\pi_P(b)) \vee (\pi_D(b) \wedge \neg\pi_P(a))$	$\pi_{NA}(a) \wedge \pi_{NA}(b)$
$a \vartriangle b$	$(\pi_P(a) \wedge \neg\pi_D(b)) \vee (\pi_P(b) \wedge \neg\pi_D(a))$	$\pi_D(a) \vee \pi_D(b)$	$\pi_{NA}(a) \wedge \pi_{NA}(b)$
$a \rhd b$	$\pi_P(a) \vee (\pi_{NA}(a) \wedge \pi_P(b))$	$\pi_D(a) \vee (\pi_{NA}(a) \wedge \pi_D(b))$	$\pi_{NA}(a) \wedge \pi_{NA}(a)$

Table 4: Transformation of three-valued logic to Boolean logic.

composable security theorem of Canetti [8], a protocol that is obtained by arbitrary combination of secure subprotocols guarantees security. Therefore, we can conclude that the protocols we designed for the operator in Table 1 as well as their combination are secure since they are composed from subprotocols proven secure.

Given that our protocols based on both 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.
- STP learns neither the data holders' policies and their evaluation nor the final decision. In HE-based protocols, STP generates the public and private keys and sends the public key to the data holders and DS. Moreover, this entity interacts with DS to evaluate the multi-party policy. As discussed above, the building blocks used in the design of our protocols are secure and, thus, the semi-honest STP is not able to learn the data holders' individual policies. Moreover, STP receives the encrypted final decision, which it decrypts using the private key. However, because of the random noise added by DS to the encrypted message (see Section 3.2), STP is not able to infer the final decision. On the other hand, in SFE-based protocols, STP only receives secret shares of the data holders' policies. Also, decryption is not needed. Therefore, neither the data holders' policies nor the result of policy evaluation is revealed to STP.
- DS learns the final decision but not the data holders' policies or their evaluation. In HE-based protocols, DS receives the policies from the data holders in encrypted form. However, since it does not have the private key, it is not able to learn these policies. On the other hand, in SFE-based protocols, DS only receives secret shares of the data holders' policies. DS evaluates the multi-party policy through communication with STP, which has been proven secure. The final decision, which DS derives with the help of STP, is not considered as a secret information for DS since this entity is responsible for its enforcement.

It is worth noting that, by knowing the final decision and the multi-party policy, the entities involved (requester, data holders, *DS*, *STP*) may be able to learn some information about the user policies. For example, given the multi-party policy p in Section 3.1, if the final decision is *not-applicable*, one can infer that all user policies forming the multi-party policy 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.

6 PERFORMANCE ANALYSIS

We implemented the protocols presented in Section 4 in C++. We used the GMP multiprecision library for the implementation of big integer operations. For the HE-based protocols, we used a cryptographic key of length 4096 bits as recommended by the NIST [4]. For the SFE-based protocols, we used 2-bit Boolean circuits for three-valued logic and 1-bit Boolean circuits for Boolean logic.

To assess the practical feasibility of our mechanism for privacy-preserving multi-party access control, we performed a number of experiments. A first set of experiments was used to evaluate each protocol individually; then we performed another set of experiments to study the scalability of our mechanism by varying the number of user policies forming the multi-party policy. Performances were evaluated in terms of computation and communication costs. In particular, we used *computation time* to measure computation costs and *bandwidth usage* to measure communication costs.

The experiments were performed on a single machine running Ubuntu 18.04 LTS with a 64-bit microprocessor and 16 GB of RAM, with Intel Core i7-4770, 3.40 GHz x 8.

6.1 Single Operator Policies

The goal of this first set of experiments is to assess the computation and communication costs of the secure computation protocols proposed in Section 4. This provides an indication of the resources required by the policy combining operators in Table 1 when implemented using different privacy-preserving techniques.

Setting. For the experiments, we have created three multi-party policies for each operator in Table 1, one multi-party policy for each approach: homomorphic encryption (HE), Boolean circuits using the three-valued encoding (SFE-T) and Boolean circuits using the Boolean encoding (SFE-B). Each multi-party policy consists of one or two individual policies depending on the number of arguments required by the operator. For each multi-party policy, we tested all possible evaluations of the user policies forming the multi-party policy (3 for the protocols with one argument and 9 for the protocols with two arguments). We repeated the experiments 50 times.

Protocol	*HE*	*SFE-T*	*SFE-B*
not	262.710	3.106	**0.314**
wea	182.827	2.903	**0.787**
smax	216.687	**0.782**	0.803
smin	215.137	**0.768**	0.788
wmax	576.736	3.290	**0.835**
wmin	578.821	3.208	**0.821**
po	697.489	3.229	**0.800**
do	746.158	3.065	**0.783**
fa	236.091	3.292	**0.851**

Table 5: Computation time (in ms) for single operator policies. The best results for each protocol are reported in bold.

Protocol	*HE*	*SFE-T*	*SFE-B*
not	7681	67326	**42**
wea	6144	62866	**4122**
smax	7169	4230	**4072**
smin	7169	4242	**4125**
wmax	19459	66394	**4078**
wmin	19459	67122	**4090**
po	24580	71210	**4130**
do	26117	61369	**4071**
fa	7681	65428	**4124**

Table 6: Bandwidth usage (in bytes) for single operator policies. The best results for each protocol are reported in bold.

Results. Table 5 reports the average computation time of each protocol over 150 runs for unary protocols and over 450 runs for binary protocols. We can observe that the SFE-based protocols (based on both three-valued and Boolean encoding) outperform the HE-based protocols (HE) in terms of computation time. On average, the computation time of HE-based protocols is approximately 160 times worse than the one of SFE-based protocols based on the three-valued encoding (SFE-T) and 500 times worse than the one of SFE-based protocols based on the Boolean encoding (SFE-B).

The difference in computation cost is twofold. The first reason lies in the complexity of the operations performed. HE protocols require performing encryptions, decryptions and homomorphic operations. These operations require modular exponentiations, which are computationally expensive. On the other hand, SFE operations are simpler, encompassing low level circuit operations such as AND, OR, inverse and equality check. The second reason is the message size used in those schemes. In HE, all operations are performed on 4096 bits ciphertexts. On the other hand, in SFE, we use 1 bit (Boolean logic) and 2 bits (three-valued logic) message sizes.

We also observe that SFE-B protocols are, in general, significantly more efficient compared to the ones based on the three valued encoding. On average, the former performs four times better than the latter. This is expected since the Boolean encoding only uses AND, OR and inverse gates while the three-valued encoding requires more complex gates such as equality and multiplication gates. The only exceptions are the strong conjunction (*smin*) and strong disjunction (*smax*) protocols, for which the SFE-T protocol performs slightly better than the one based on the Boolean encoding. For these cases, SFE-T protocols are less complex compared to SFE-B protocols. Computing *smin* and *smax* protocols requires a single AND and a single OR gate operation in the three-valued encoding, respectively, while the Boolean encoding requires 3 AND, 2 OR and 2 inverse gates for each protocol (cf. Table 4).

It is worth noting that all SFE-B protocols, with the exception of the negation protocol (*not*), require a similar computation time, whereas for the other approaches there is a notable difference between protocols. The similarity of computation times is caused by the comparable number of gates used in the implementation of each protocol. Negation protocol differs from the others since it only requires swapping values rather than operating on logic gates.

Table 6 reports the average bandwidth usage of the proposed protocols. We can observe that the SFE-B protocols outperform the protocols implemented using the other approaches also for bandwidth usage. On average, the bandwidth usage required by the SFE-B protocols is approximately 30 times lower than the one required by the HE-based protocols and 190 times lower compared to the one required by the SFE-T protocols. For most operators, HE-based protocols perform better than SFE-T protocols. However, strong disjunction (*smax*) and strong conjunction (*smin*) protocols show a different trend since their implementation in three-valued logic requires a single logic gate while under HE the implementation is more complex.

From these results, we conclude that a direct implementation of the three-valued logic operators in homomorphic encryption and secure function evaluation has severe limitations in terms of computation time and bandwidth usage. In contrast, the SFE-B protocols provide a viable solution for the realization of mechanisms for privacy-preserving multi-party access control.

6.2 Complex 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.

Settings. For the experiments, we generate random multi-party policies by varying the number of user policies. Intuitively, the number of user policies represents the number of users that jointly manage a given object. We varied the size of the generated multi-party policies (*i.e.*, the number of user policies forming the multi-party policies) from 2 to 50. This allows us to test our approach is a situation representative for many real-world systems where resources are controlled by a large but limited number of users. We repeated each experiment 20 times.

Results. Figures 2 and 3 show respectively the average computation time and bandwidth usage (in log scale) for each size of multi-party policy protocols over 20 rounds of the experiments. The results confirm that the SFE-based protocols based on the Boolean encoding (SFE-B) outperform the HE-based protocols (HE) and the SFE-based protocols based on the three-valued encoding (SFE-T) in terms of both computation time and bandwidth usage. In particular, the evaluation of the multi-party policy comprising 50 user policies required 3.8 ms using SFE-B protocols, showing the practical feasibility of this approach in realistic scenarios. In general, we can observe that, for all approaches, both computation time and bandwidth usage are log-linear in the size of the multi-party policy.

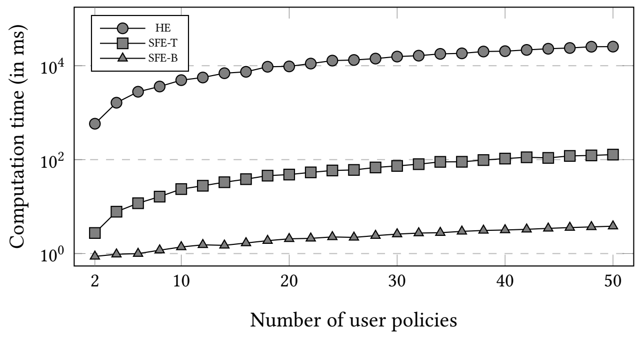

Figure 2: Computation time (in ms) for complex policies. Y-axis is in log scale.

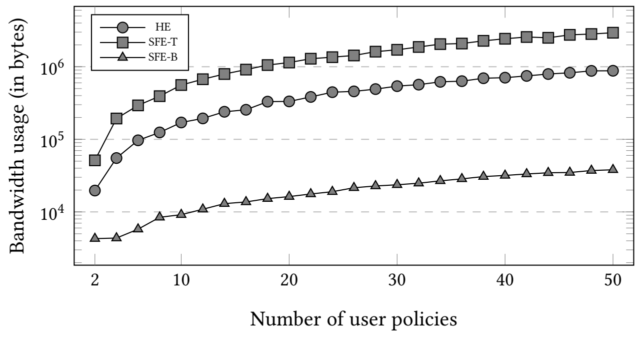

Figure 3: Bandwidth usage (in bytes) for complex policies. Y-axis in log scale.

The results show that our approach, when using SFE-B protocols, is scalable and provides an effective foundation for the realization of privacy-preserving mechanisms for multi-party access control.

7 RELATED WORK

In recent years, multi-party access control has received increasing attention [34]. This interest has resulted in several access control solutions for the protection of jointly-owned and jointly-managed resources, either through co-owners' negotiation [19] or automatic built-in interface [26]. In particular, conflict resolution approaches have been largely investigated through the application of: game theory [36], computational mechanisms [39], social-friend circle model [45], and veto voting [40]. These approaches, however, do not consider users' policies as sensitive information by themselves.

The confidentiality of policies is typically addressed in trust negotiation [43, 46]. The aim of trust negotiation is to establish mutual trust between parties that do not have a pre-existing relationships 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. [41] 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. Then, these policies are evaluated in privacy-preserving way using secure computation protocols.

A large body of research has investigated the use of cryptographic techniques in the context of access control [1, 3, 28]. However, most works aims at the protection of resources and do not address the confidentiality of policies. An exception is the work in [2], which proposes the concept of secure policy execution using reusable garbled circuits. However, this approach assumes that each resource is under the control of a single entity.

An application of cryptographic techniques to multi-party access control can be found in [25], 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, which 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. To the best of our knowledge, our work is the first that addresses the problem of secure evaluation of composite policies encompassing both positive and negative authorizations.

8 CONCLUSION

Multi-party access control has emerged as a paradigm to support collaborative decision-making for co-owned resources. Several solutions have been proposed with the purpose of combining the access requirements of the users involved in the protection of co-owned resource. However, the problem of protecting the confidentiality of users' policies in multi-party access control has not been addressed.

In this paper, we have presented the first mechanism for the secure evaluation of multi-party access control policies, which preserves the confidentiality of user policies. To this end, we have proposed secure computation protocols based on three different privacy-preserving approaches: 1) homomorphic encryption, 2) Boolean circuits based on a three-valued encoding, and 3) Boolean circuits based on a Boolean encoding. An experimental evaluation of the proposed protocols shows that the latter approach outperforms the other two approaches in terms of both computation and communication costs. In particular, the results obtained using SFE-based protocols relying on the Boolean encoding demonstrate the practical feasibility of privacy-preserving mechanisms for multi-party access control.

In this work, we have assumed that only user policies are in private form whereas the multi-party policy, which specifies how user policies are combined, is in plaintext. As discussed in Section 5, this may be exploited by an attacker to learn some information on the undelying user policies. To minimize such a risk, we plan to extend our approach to also support the evaluation of multi-party policies in private form. Moreover, we plan to adapt and extend our approach for the privacy-preserving evaluation of more complex policies, for example specified in the XACML standard.

Acknowledgements. This work is funded by the ECSEL project SECREDAS (783119) and the ITEA3 project APPSTACLE (15017).

REFERENCES

[1] Ruqayah R. Al-Dahhan, Qi Shi, Gyu Myoung Lee, and Kashif Kifayat. 2019. Survey on Revocation in Ciphertext-Policy Attribute-Based Encryption. *Sensors* 19, 7 (2019), 1695.

[2] Masoom Alam, Naina Emmanuel, Tanveer Khan, Abid Khan, Nadeem Javaid, Kim-Kwang Raymond Choo, and Rajkumar Buyya. 2018. Secure policy execution using reusable garbled circuit in the cloud. *Future Generation Computer Systems* 87 (2018), 488–501.

[3] Muhammad Asim, Tanya Ignatenko, Milan Petkovic, Daniel Trivellato, and Nicola Zannone. 2012. Enforcing Access Control in Virtual Organizations Using Hierarchical Attribute-Based Encryption. In *Proceedings of International Conference on Availability, Reliability and Security*. IEEE, 212–217.

[4] Elaine B. Barker, Lidong Chen, Andrew R. Regenscheid, and Miles E. Smid. 2009. *SP 800-56B. Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography*. Technical Report. Gaithersburg, MD, United States.

[5] Donald Beaver. 1991. Efficient Multiparty Protocols Using Circuit Randomization. In *Advances in Cryptology (LNCS)*, Vol. 576. Springer, 420–432.

[6] Joppe W. Bos, Kristin Lauter, and Michael Naehrig. 2014. Private predictive analysis on encrypted medical data. *Journal of Biomedical Informatics* 50 (2014), 234–243.

[7] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. 2015. Machine Learning Classification over Encrypted Data. In *Proceedings of Annual Network and Distributed System Security Symposium*. Internet Society.

[8] Ran Canetti. 2001. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of Symposium on Foundations of Computer Science*. IEEE, 136–.

[9] Shuchih Ernest Chang, Anne Yenching Liu, and Wei Cheng Shen. 2017. User trust in social networking services: A comparison of Facebook and LinkedIn. *Computers in Human Behavior* 69 (2017), 207–217.

[10] Jason Crampton and Charles Morisset. 2012. PTaCL: A Language for Attribute-based Access Control in Open Systems. In *Principles of Security and Trust*. Springer, 390–409.

[11] Jason Crampton, Charles Morisset, and Nicola Zannone. 2015. On Missing Attributes in Access Control: Non-deterministic and Probabilistic Attribute Retrieval. In *Proceedings of Symposium on Access Control Models and Technologies*. ACM, 99–109.

[12] Stan Damen, Jerry den Hartog, and Nicola Zannone. 2014. CollAC: Collaborative access control. In *Proceedings of International Conference on Collaboration Technologies and Systems*. IEEE, 142–149.

[13] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *Proceedings of Annual Network and Distributed System Security Symposium*. Internet Society.

[14] Jerry den Hartog and Nicola Zannone. 2016. Collaborative Access Decisions: Why Has My Decision Not Been Enforced?. In *Information Systems Security (LNCS)*, Vol. 10063. Springer, 109–130.

[15] Raïssa Yapan Dougnon, Philippe Fournier-Viger, Jerry Chun-Wei Lin, and Roger Nkambou. 2016. Inferring social network user profiles using a partial social graph. *Journal of Intelligent Information Systems* 47, 2 (2016), 313–344.

[16] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. 2009. Privacy-Preserving Face Recognition. In *Proceedings of International Symposium on Privacy Enhancing Technologies*. Springer, 235–253.

[17] Zekeriya Erkin, Thijs Veugen, Tomas Toft, and Reginald L. Lagendijk. 2012. Generating Private Recommendations Efficiently Using Homomorphic Encryption and Data Packing. *IEEE Transactions on Information Forensics and Security* 7, 3 (2012), 1053–1066.

[18] Daniel C. Feldman. 1984. The Development and Enforcement of Group Norms. *The Academy of Management Review* 9, 1 (1984), 47–53.

[19] Ricard L. Fogues, Pradeep K. Murukannaiah, Jose M. Such, and Munindar P. Singh. 2017. Sharing Policies in Multiuser Privacy Scenarios: Incorporating Context, Preferences, and Arguments in Decision Making. *ACM Trans. Comput.-Hum. Interact.* 24, 1, Article 5 (2017), 29 pages.

[20] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. 2004. Efficient Private Matching and Set Intersection. In *Advances in Cryptology*. Springer, 1–19.

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

[22] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. 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, 218–229.

[23] Paolo Guarda and Nicola Zannone. 2009. Towards the development of privacy-aware systems. *Information & Software Technology* 51, 2 (2009), 337–350.

[24] Pan Hui and Sonja Buchegger. 2009. Groupthink and Peer Pressure: Social Influence in Online Social Network Groups. In *Proceedings of International Conference on Advances in Social Network Analysis and Mining*. IEEE, 53–59.

[25] Panagiotis Ilia, Barbara Carminati, Elena Ferrari, Paraskevi Fragopoulou, and Sotiris Ioannidis. 2017. SAMPAC: Socially-Aware collaborative Multi-Party Access Control. In *Proceedings of Conference on Data and Application Security and Privacy*. ACM, 71–82.

[26] Panagiotis Ilia, Iasonas Polakis, Elias Athanasopoulos, Federico Maggi, and Sotiris Ioannidis. 2015. Face/Off: Preventing Privacy Leakage From Photos in Social Networks. In *Proceedings of Conference on Computer and Communications Security*. ACM, 781–792.

[27] William H. Jobe. 1962. Functional Completeness and Canonical Forms in Many-Valued Logics. *The Journal of Symbolic Logic* 27, 4 (1962), 409–422.

[28] Cheng-Chi Lee, Pei-Shan Chung, and Min-Shiang Hwang. 2013. A Survey on Attribute-based Encryption Schemes of Access Control in Cloud Environments. *International Journal of Network Security* 15, 4 (2013), 231–240.

[29] Rauf Mahmudlu, Jerry den Hartog, and Nicola Zannone. 2016. Data Governance and Transparency for Collaborative Systems. In *Data and Applications Security and Privacy (LNCS 9766)*. Springer, 199–216.

[30] Charles Morisset, Tim A. C. Willemse, and Nicola Zannone. 2018. Efficient Extended ABAC Evaluation. In *Proceedings of Symposium on Access Control Models and Technologies*. ACM, 149–160.

[31] Majid Nateghizad, Zekeriya Erkin, and Reginald L. Lagendijk. 2016. Efficient and secure equality tests. In *Proceedings of International Workshop on Information Forensics and Security*. IEEE, 1–6.

[32] Majid Nateghizad, Zekeriya Erkin, and Reginald L. Lagendijk. 2016. An efficient privacy-preserving comparison protocol in smart metering systems. *EURASIP Journal on Information Security* 2016, 1 (2016), 11.

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

[34] Federica Paci, Anna Cinzia Squicciarini, and Nicola Zannone. 2018. Survey on Access Control for Community-Centered Collaborative Systems. *ACM Comput. Surv.* 51, 1 (2018), 6:1–6:38.

[35] Pascal Paillier. 1999. Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proceedings of International Conference on Theory and Application of Cryptographic Techniques*. Springer, 223–238.

[36] Sarah Rajtmajer, Anna Squicciarini, Christopher Griffin, Sushama Karumanchi, and Alpana Tyagi. 2016. 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, 680–688.

[37] Mina Sheikhalishahi and Fabio Martinelli. 2017. Privacy preserving clustering over horizontal and vertical partitioned data. In *Proceedings of Symposium on Computers and Communications*. IEEE, 1237–1244.

[38] Lauren E. Sherman, Ashley A. Payton, Leanna M. Hernandez, Patricia M. Greenfield, and Mirella Dapretto. 2016. The Power of the Like in Adolescence: Effects of Peer Influence on Neural and Behavioral Responses to Social Media. *Psychological Science* 27, 7 (2016), 1027–1035.

[39] Jose M. Such and Natalia Criado. 2016. Resolving Multi-Party Privacy Conflicts in Social Media. *IEEE Transactions on Knowledge and Data Engineering* 28, 7 (2016), 1851–1863.

[40] Kurt Thomas, Chris Grier, and David M. Nicol. 2010. unFriendly: Multi-party Privacy Risks in Social Networks. In *Privacy Enhancing Technologies (LNCS 6205)*. Springer, 236–252.

[41] Daniel Trivellato, Nicola Zannone, and Sandro Etalle. 2014. GEM: A distributed goal evaluation algorithm for trust management. *TPLP* 14, 3 (2014), 293–337.

[42] Fatih Turkmen, Jerry den Hartog, Silvio Ranise, and Nicola Zannone. 2017. Formal analysis of XACML policies using SMT. *Computers & Security* 66 (2017), 185–203.

[43] William H. Winsborough and Ninghui Li. 2002. Towards Practical Automated Trust Negotiation. In *Proceedings of International Workshop on Policies for Distributed Systems and Networks*. IEEE, 92–103.

[44] Andrew Chi Yao. 1982. Protocols for Secure Computations. In *Proceedings of Annual Symposium on Foundations of Computer Science*. IEEE, 160–164.

[45] Lingjing Yu, Sri Mounica Motipalli, Dongwon Lee, Peng Liu, Heng Xu, Qingyun Liu, Jianlong Tan, and Bo Luo. 2018. My Friend Leaks My Privacy: Modeling and Analyzing Privacy in Social Networks. In *Proceedings of Symposium on Access Control Models and Technologies*. ACM, 93–104.

[46] Ting Yu, Marianne Winslett, and Kent E. Seamons. 2003. Supporting Structured Credentials and Sensitive Policies Through Interoperable Strategies for Automated Trust Negotiation. *ACM Trans. Inf. Syst. Secur.* 6, 1 (2003), 1–42.