Future Generation Computer Systems 87 (2018) 488–501

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Secure policy execution using reusable garbled circuit in the cloud

Masoom Alam [a,*], Naina Emmanuel [a], Tanveer Khan [b], Abid Khan [a], Nadeem Javaid [a], Kim-Kwang Raymond Choo [c,1], Rajkumar Buyya [d,2]

[a] Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan
[b] Department of Computer Sciences, Office of Academic Research college of Engineering University of Qatar, Doha, Qatar
[c] Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA
[d] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia

ARTICLE INFO

Article history:
Received 11 July 2017
Received in revised form 16 October 2017
Accepted 31 December 2017
Available online 31 January 2018

Keywords:
Cross tenant access control
Formal specification
Cloud computing
Reusable garbled circuits

ABSTRACT

While cloud computing is fairly mature, there are underpinning data privacy and confidentiality issues that have yet to be resolved by existing security solutions such as cross domain access control policies. The latter necessitates the sharing of attributes with a Trusted Third Party (TTP), which in turn raises data privacy concerns. In this paper, we present a Privacy Aware Cross Tenant Access Control (PaCTAC) protocol for cross domain cloud users, based on reusable garbled circuit. We also propose the concept of a privacy aware Cloud Policy Decision Point (CPDP) that can be offered by cloud service providers. CPDP plays the role of a trusted third-party among its different tenants. We then formally specify PaCTAC to demonstrate its security.

1. Introduction

Cloud computing has been the subject of active research and innovation in both academia and industry in the last decade [1,2], as evidenced by the continuing interest in designing secure and efficient solutions for different cloud computing applications and scenarios. For example, cloud service providers (CSPs) have been working collaboratively with researchers to design solutions that allow cloud users to access the resources of the host CSP, as well as those of a collaborative CSP, efficiently and securely. Such collaboration allows the cloud computing and related industries to offer more sophisticated and advance services. However, such a collaborative distributed environment complicate efforts to ensure the privacy of data, services and infrastructure. One particular research direction is to design efficient techniques to preserve user privacy from (honest-but-curious or malicious) CSPs. Encryption for data-at-rest technique is an effective approach [3], but this limits the capability to provide other services. For example, how to effectively search on encrypted data remains a topic of ongoing research at the time of this research [4,5].

Achieving fine-grained and flexible access control is also another topic that is being explored in the literature, such as the scheme reported in [6]. The general requirement is for both the service provider and the data owner to be in the same trusted environment, which is not usually the case in practice. Presently, single sign-on methods are being used by CSPs to provide authentication and simple authorization in a collaborative cloud environment; however, fine grained authorizations are not fully supported. Role Based Access Control (RBAC) has been used in many diverse applications. However, RBAC does not support the extent of collaboration required by contemporary multi-tenant cloud computing environment. Unsurprisingly, the RBAC model has been extended in the literature in order to achieve effective access control in a collaborative environment [7,8]. However, these extended models require the existence of a centralized authority. In a collaborative cloud computing deployment where users are from different independent administrative domains, such a requirement may not be realistic. Electronic Health Record (EHR) is used to share health information via a cloud-based platform. However, data shared over the or stored in the cloud may be targeted by cyber criminals to violate a patient's privacy [9]. Existing privacy protection techniques can be broadly categorized into the following domains: privacy by cryptography, privacy by statistics, and privacy by police [10]. A number of (practical) solutions for EHR sharing have also been proposed in the literature, such as those that rely on a classification of the patient's attributes.

In this paper, we introduce the concept of secure policy execution in the cloud using a Reusable Garbled Circuit (RGC). RGC-based techniques allow computation to be performed over encrypted data, and include fully homomorphic encryption technique [11–13], functional encryption technique [14,15],

* Corresponding author.
E-mail addresses: masoom.alam@comsats.edu.pk (M. Alam), Qatatanveer.khan@qu.edu.qa (T. Khan), abidkhan@comsats.edu.pk (A. Khan), nadeemJavaid@comsats.edu.pk (N. Javaid), raymond.choo@fulbrightmail.org (K.R. Choo).
1 Senior Member, IEEE.
2 Fellow, IEEE.

https://doi.org/10.1016/j.future.2017.12.067
0167-739X/

and attribute based encryption technique [13,16–18]. In 2013, Goldwasser et al. [13] proposed a RGC using fully homomorphic encryption techniques. Another recent RGC-based approach is presented by Wang et al. [18] in 2016, who use random linear coding. In our protocol, the approach of Wang et al. [18] is used for garbling the policy. We remark that there has been no practical implementation of RGC on multi-party policy execution environment in the literature, at the time of this research. Also, in this paper, we construct the Privacy Aware Cross Tenant Access Control (PaCTAC) protocol for cross domain users. The protocol achieves secure policy execution and its storage at a privacy aware Cloud Policy Decision Point (CPDP) that can be offered by cloud service providers. In a typical RBAC scheme, if the centralized authority acts maliciously then the security of the system would be compromised. Thus, we propose a garbling scheme to act as roles under a secure manner in the untrusted environment. Specifically, we provide security against the execution of the policies where the cloud (or an unauthorized third party) does not have knowledge of the policies and user's attributes. In other words, this garbled paradigm provides protection against malicious activities on user data in the cloud. We remark that our proposal also support the extent of collaboration required in the contemporary multi-tenant cloud environment.

The rest of paper is organized as follows. In Sections 2 and 3, we respectively present the preliminaries and related work. In Sections 4 and 5, we describe the PaCTAC protocol and present a formal analysis of the proposed protocol, respectively. A distributed health care information system is used as a case study in Section 6. Section 7 provides the security analysis of the proposed scheme. The paper concludes in Section 8.

2. Preliminaries

The garbled circuit approach is first proposed by Yao [19] in 1986 for secure multiparty computation. There are two parties in such a protocol, and the protocol is shown to be secure in the semi-honest adversary model and can be extended to a polynomial number of parties.

Yao's garbled circuit can be defined as $GC = (GC.Garble, GC.Eval)$, which uses the Garbling algorithm and the Evaluation algorithm. In Yao's original scheme, one party generates the circuit and other evaluates it. Suppose we have a circuit C, having n input wires. The generator produces the garbled circuit and two labels for each input of the wire (0 *or* 1). The labels are used as encryption keys. The evaluator can evaluate the garbled circuit to obtain the circuit output [6]. The generator of the circuit submits its input. These inputs are the garbled inputs. The evaluator of the circuit submits its input and the output is generated, which further gets permuted. The evaluator then decrypts the output [7].

For many years, garbled circuits and variants have been used in a wide range of applications, such as two party secure protocols [16], verifiable computation [20], multi-party secure protocols [21], one-time programs and homomorphic computations [22]. However, a key limitation of garbled circuit is that it offers only one-time usage. Specifically, an encoding of more than one input compromises the security of the circuit and secrecy of the inputs. Thus, evaluating the circuit C for a new input requires a new garbled circuit.

The design of a RGC is an open problem for many years. It is a circuit garbling scheme in which garbled circuit is used multiple times for multiple policies. It takes any polynomial number of inputs without compromising the privacy of circuit and its inputs. The construction of RGC, say C, can be presented as follows: First, the inputs are encrypted before $C(x)$ is computed. After the computation, C encrypts the output $C(x)$, where x is the input value to the circuit. A RGC takes three algorithms, namely: $RGC = (RGC.GARBLE, RGC.ENC, RGC.EVAL)$, for circuit C. The security of the scheme intuitively says that a garbled circuit can be used with many encodings while still hiding the circuit and the inputs. A number of techniques have been proposed for the generation of RGC, such as those presented in [13,18].

Definition 1 (*Security Parameters*)**.** In our proposed scheme, κ is a security parameter while other parameters n, κ, η and q are determined from this security parameter κ, where $[n/4] \geq \eta > k$; $n = 17$, $k = 2$, $\eta = 4$ and $q = 19$. A linear code e $([n, k])$ is a sub-space of $\mathrm{GF}(q)^n$, where $n =$ length of code and $k =$ dimensions.

Definition 2 (*Galois Field*)**.** Let $GF(q)^n$ be a Galois field, where q is a prime number, giving $GF(19)$. Using $GF(19)$ we derive the following:

- Let A be a non-singular randomly selected matrix of 17×17 taken over $GF(19)$.
- Let G be the generating matrix of 2×17 obtained from $GF(q)^{k\times n}$, where $G = I_k/p$. (I_k an identity matrix and p a non-identity matrix).
- $H_{f,0}$ be a 17×2 parity check matrix with 5 to 7 being zero rows, where $H = [-p^T/I_{n-k}]$ (T is a transpose). We remark that either G or H can be changed as necessary.
- e, a linear code $[n, k]$, is a sub-space taken from $\mathrm{GF}(q)^n$.
- $x \in GF(q)^k$, where x is the message of length k.

Definition 3 (*Garbled Input, Output and Function*)**.** Let $(c_x+c_y+a_{\pi,i'})$ be the output of the garbled circuit, which is calculated as follows:

- Let c_1 and c_2 be the plain inputs given by client c_1 and $c_2 \in [0, 1]$. Both c_1 and c_2 are labeled with k_x^i and k_y^i for encryption.
- Inputs are randomized with shared row vectors (r_x, r_y, k_s), where $r_x, r_y \epsilon GF(q)^k$ and $k_s \epsilon_R GF(q)^n$, sets $e_s[j] = \{k_s[j-3]\eta\}$ if $2\eta \leq j \leq 3\eta - 1\}$, 0 otherwise. It produces garble inputs c_x and c_y, which then produces garble output $(c_X + c_y + a_{\pi,i'})$, where $a'_{\pi,i'}$ is randomly order tuple $i \in [0, 1]$ of row vector. The garble gate π is described by randomly ordered tuple and b_π, where b_π is a column vector $b_\pi = A^{-1}H_{\pi,0}c_\pi$, and the evaluator checks whether $(c_x + c_y + a_{\pi,i})b_\pi$ is 0 or 1.

3. Related work

Conventional RBAC [23,24] model supports authorization within an organization. In the RBAC model, a user of a service can perform an action only if a role has been assigned to the user and the role has the authorization to perform that action. A number of extensions to the RBAC model have been proposed to provide access control among multiple organizations. Some extensions of the RBAC model [25,26] introduced the concept of a centralized entity, which is responsible for mediating cross domain policies. However, these approaches are not applicable in a distributed cloud computing environment, since the CSP is responsible for maintaining the (generic) policies for all its tenants who may not be long-term and are self-service oriented. Extended models using decentralized authorities have also been proposed, such as delegation-based models [27,28] where a user delegates permission to other users. A graph data structure is used to maintain all delegations of a permission. If any of the nodes changes due to a user change, then the entire graph of authorization is reconstructed. In other words, such an approach can become overwhelmingly complex and does not scale well, particularly when deployed in a distributed cloud environment.

RBAC model has also been extended by mapping roles among multiple domains [20,21,29–31]. In order to support collaboration in a distributed environment, different identity and authorization services have been proposed in the literature. Federated identity, for example, allows strangers to be authenticated through sharing

of identity information among federated entities who trust each other. However, such approaches are costly and do not support agility typical of a cloud computing environment. Authorization services for virtual organizations have also been developed to manage secure resource sharing using cryptographic credentials (see [23–27]). Although such approaches are secure and effective, it is known that maintaining the public key infrastructure to securely store cryptographic credentials is an expensive exercise. Calero et al. [28] presented a centralized multi-tenancy authorization system for cloud deployment. This work is also an extension of RBAC model using a trust model which is coarse-grained and easily extensible as demonstrated by authors in [20–22,29]. In the near future,it is expected that Internet of Things (IoT) will permeate different facets of our society [32]. For example, in recent times there have been efforts to integrate cloud computing and IoT on same platform [33].

Other studies on providing cross tenant access control in a cloud computing environment include the work of Ngo et al. [34], Shen et al. [35], Hingwe et al. [36], Almutairi et al. [37], Syler et al. [38], and Pustchi et al. [39]. Ngo et al. [34] proposed an attribute based access control model for multi-tenant cloud based services. The authors extended the inter-cloud scenario using an exchange of tokens and by applying an efficient mechanism to transform complex logical expression in policies to compact decision diagrams. Shen et al. [35] introduced a collaborative cloud computing platform, which is designed to integrate resource management and reputation management. Three important features were provided by the platform, namely: integrated multi-faceted resource/reputation management, multi-QoS-oriented resource selection, and price assisted resource/reputation control. Hingwe et al. [36] extended RBAC to database as a service architecture, which can be used to manage role hierarchy and secure session management for query transaction in databases. The proposed approach uses a partial homomorphic encryption scheme. Data is protected from privilege escalation and SQL injection. The proposed approach assigns a query with minimum access privileges and a session key is used for session management. Almutairi et al. [37] proposed the notion of sensitivity of multi-tenant datacenters in terms of the degree of data sharing among tenants. Here, limited sharing implies a high sensitive datacenter and high sharing of data implies a low sensitive datacenter. The authors presented a virtual resource sharing approach in which risk is minimized. Three assignment heuristics are presented and their relative performance were compared. The authors also argued that for a given size of datacenter, risk should not exceed if the datacenter has high sensitivity as compared to the case if the datacenter has low sensitivity. Syler et al. [38] proposed a framework to handle both the dynamic nature of cloud data centers and optimized inter-tenant communication. The proposed network security framework can automatically respond to the frequent changes in the virtualized data center topologies while requiring minimal configurations. Pustchi et al. [39] proposed a multi-cloud trust model, where resources can be shared between the clouds. In the model, at the lower administrative level, CSPs can share their resources among their customers within multiple clouds. A prototype based on the administrative model of OpenStack was provided.

However, the above mentioned schemes do not consider the privacy of shared attributes or cater to the Chinese Wall policy among different organizations (who may have conflicting interests).

4. Proposed PaCTAC protocol

In this section, we describe the proposed PaCTAC protocol, based on RBC. In the protocol, a resource-providing tenant is also a service provider and generator of a circuit. The tenant who consumes resources is considered an attribute generator in the garbled circuit, as it provides the attributes. For instance, T2 accesses the resources of T1, provides the user attributes required by T1 and is evaluated by CPDP on behalf of T1. The role of CPDP is to proxy the policy evaluation for the tenant holding the resource.

RGC is used in the privacy preserving policy evaluation for the CPDP. In our example, T1's published policy on CPDP is hidden from CPDP due to the inherent characteristic of garbled function. This helps in protecting the privacy of user and its attributes, in the event that CPDP is compromised.

Formal Specification of PaCTAC: The three main underlying modules in PaCTAC are as follows:

- Tenant 1 (T1) who is the generator of RGC and policies for resources that it wishes to share with other tenants,
- Tenant 2 (T2) who is the consumer of resources and,
- CPDP (Proxy Cloud) who evaluates the circuit and policy on behalf of T1.

Algorithms 1, 2 and 3 define the functionality of the above three modules. IDs are associated with these users, and a user's cross-domain identity differs from their identity with the parent domain. In other words, a user has multiple IDs rather than a single ID for each domain.

Both users of local-domain and cross-domain can activate the permissions using attribute verification algorithm *ActQ*. Prior to the activation of permissions for the local user, their permission-assignment-relation are checked. Once the set of permissions are assigned to local users, they can issue the delegation of permissions to cross-domain. Delegation of permission can be performed in two ways, namely: user level and domain level. At the user level, permissions are delegated to users. At the domain level, permissions are delegated to a domain. From a privacy perspective, attributes from T2 are encrypted using garbled circuit and hence, not exposed to either CPDP or T1. T1 only receives the grant signal from CPDP, who then obtains the attributes in the form of garbled inputs. The components of the PaCTAC mechanism are as follows:

• T_l (set of local-domain users), T_j and T_k (set of cross-domain users), M_i (set of permissions) where ($m_l \in M_i$).

A user who belongs to T_l is a local-domain user known as the permission delegator, and a user belonging to T_j or T_k is a cross-domain user that can act as the delegator (in further delegation case) or delegatee (in case permissions are delegated to it).

• $T_{d=l,j,k} : D \rightarrow 2^{Sub}$, where $Sub = \bigcup_{d \in D} T_d$ (i.e. set of users in a domain), t_l denotes local users, t_j and t_k denote cross users, and m_l denote the set of permissions.

• $PAU_i \subseteq (t_l \times m_l)$.

Set PAU_i is the relation between the user of local-domain and permissions assigned to the user in ith domain.

• $PAU_a \subseteq (t_l \times m_l)$.

Set PAU_a is the relation between the user of local-domain and permissions activated by the user in ith domain.

• $LCP_a \subseteq (t_l \times t_j \times m_l)$.

Set LCP_a is the triple relation among the user of local-domain and cross-domain with permissions that activated them in ith domain.

• $CDP_a \subseteq (t_j \times t_k \times m_l)$.

Set CDP_a is the triple relation among the users of cross-domain and permissions activated them in ith domain.

• $LDP_a \subseteq (t_l \times d \times m_l)$.

Set LDP_a is the triple relation among users of the local-domain, the domain itself and permissions activated by the users of local-domain.

• $CDDP_a \subseteq (t_k \times d \times m_l)$.

Set $CDDP_a$ is the triple relation among the user of cross-domain, the domain itself and permissions activated by user of that domain.

• $LCPC \subseteq (t_l \times t_j \times m_l \times c')$.

Table 1
Summary of Notations.

Notation	Description	Notation	Description
C	Garbled circuit	sk_π	Secret key for π gate
n	No of wires of garbled circuit	G	Generating matrix
x	Input values of circuit	e^i_x, e^i_y	Random vectors for x and y inputs
T_l	Set of local-domain users	κ	Security Parameter
T_k, T_j	Set of cross-domain users	n, η, q	Values determined from κ
M_i	Set of permissions	A	Non-singular randomly selected matrix
π	Randomly selected gate	$H_{f,0}$	Parity check matrix
$GF(q)$	Galios Field where 'q' is prime no	λ	Yao's one time garbled circuit
$a_{\pi,0}...a_{\pi,3}$	Randomly order tuple	e	Linear code $[n, \kappa]$ is a subspace taken from $GF(q)^n$
r_π	Row vector	$\phi(c)$	Information including no.of i/p, no.of o/p, no.of gates and structure
b_π	Column vector	$\bar{\pi}$	Garbled version of π
ABE_2	Two-outcome based Attribute Based Encryption	msk	Master secret key
K_c	Conditional key	mpk	Master public key
k	Dimensions of vector	γ	RGC

Set *LCPC* is the quadruple relation among the user of local-domain, cross-domain, set of permissions and constraint (policy attributes).

• $UCPC \subseteq (t_j \times t_k \times m_l \times c')$.

Set *UCPC* is the quadruple relation among the user of cross-domain, set of permissions and constraints (policy attributes).

• $LDPC \subseteq (t_l \times d \times m_l \times c')$.

Set *LDPC* is the quadruple relation among the user of local-domain, the domain itself, permissions and constraints (policy attributes).

• $CDPC \subseteq (t_j \times d \times m_l \times c')$.

Set *CDPC* is the quadruple relation among the users of cross-domain, the domain, set of permissions and constraints (policy attributes).

The notations used in our proposed scheme are described in Table 1.

Definition 4 (*Attribute Verification Algorithm*)**.** This algorithm describes how permissions m_l (where m_l is the set of permissions) are activated for the user of either local-domain or cross-domain.

Attribute verification algorithm can be formally defined as:

$$ActQ : T_{d=l,j} \times d \times m_l \rightarrow \{PAU'_a, LCP'_a, CDP'_a, LDP'_a, CDDP'_a\}$$

If $d = i$, then the user of local domain activates the permissions, and if $i \neq d$, then the activation is by cross-domain users.

• Generation of attributes c_x, c_y and evaluation of the policy is also a part of *ActQ*.

Definition 5 (*Delegation Across the Domain*)**.** *Formally* delegation of cross-domain can be defined as (delegator, delegatee, permissions), where the role of delegator is being performed by users ($T_{d=l,j,k}$) from all domains, delegatee as user of cross-domain ($T_{d=j,k/d}$) and m_l sets of permissions showing:

$$(T_{d=l,j,k}, T_{d=j,k/d,m_l}) \epsilon \{T_{d=l,j,k \rightsquigarrow^{m_l} T_{d=j,k/d}}\}$$

Definition 6 (*Delegation of Domain Users*)**.** Domain-user delegation can be defined as triple relation among (d, $T_{d=j,k,m_l}$), where d is domain acting as "delegator", $T_{d=j,k}$ is a user acting as "delegatee" and m_l sets of permissions showing:

$$(d, T_{d=j,k,m_l}) \epsilon \{d \rightsquigarrow^{m_l} T_{d=j,k}\}$$

Definition 7 (*Policy Generation Algorithm*)**.** Generation of the policy for the resource R is the part of the policy generation algorithm along with delegation of user's permissions m_l to cross-domain users.

• Generation of the policy can be formally defined as:

R1→ Policy (x_i, y_i..) is the policy for that resources R1 having x_i, y_i as its policy attributes.

• Delegation of the permission can be formally defined as:

$$DelAlgo : (T_{d=l,k} \times T_{d=j}(d \times m_l \times c')) \epsilon \{T_l \rightsquigarrow^{m_l} T_j\}' | \{T_j \rightsquigarrow^{m_l} T_k\}|$$

Definition 8 (*Randomization*)**.** Random function are used in *DelAlgo* and during the generation of the policy, that includes the following steps:

• Let us take random vectors ($e^i_x, e^i_y \epsilon GF(q)^n$).

• Randomize the encrypted inputs with these random vectors giving $a_{\pi,i}$

$$randomize\{(e^i_x, e^i_y), E\}$$

Then permute the resulting values, where E is the encrypted input: $E = Enc\{(k^i_x, k^i_y), (x_i, y_i)\}$, where k^i_x, k^i_y are key labels and x_i, y_i are inputs of the RGC.

Definition 9 (*Permission Revocation Algorithm*)**.** Revocation of permissions includes revocation by local-domain user from cross-domain user. This results in the deactivation of policy and revocation of delegation when the attributes submitted by the delegatee are mis-matched with the published constraints of the policy. Permission revocation can be formally defined as:

$$RevAlgo : (T_{d=j,k} \times m_l \times (c_1, c_2)) \neq (x_i, y_i) \rightarrow (T_l \times m_l) \epsilon PAU_i$$

Definition 10 (*Secure Revealing Scheme*)**.** Submit two x, y inputs in the revealing scheme and it gives n output z. It is a black box, which uses Yao's garbled circuit '$\wedge$' where $z = \wedge(x, y)$.

• $\wedge$ is a Yao's one-time garbled circuit algorithm for producing garbled circuit, $\wedge$:$\{0,1\}^\lambda \longrightarrow \{0,1\}$ where λ is the length of garbled output.

• $\phi(c)$ is the information including the number of inputs, number of outputs, number of gates and structure of the circuit. It is "redundant" information that does not provide any information of circuit to adversary, if leaked.

Definition 11 (*Random Gate* π)**.** π is defined as randomly selected gate $\pi \epsilon_R \{OR, AND\}$. Let the garbled version of gate π be represented as $\bar{\pi} = (b_\pi, a_{\pi,0'}, a_{\pi,1'}, a_{\pi,2'}, a_{\pi,3'})$. Then, for guessing the gate type (π) from $\bar{\pi}$ and for every probabilistic algorithm D (not only polynomial time algorithm), we have the following [18]:

$$|Prob[D(\bar{\pi}, 1^\kappa) = \pi] - 1/2| = negl(\kappa)$$

Definition 12 (*Two-Outcome Attribute-Based Encryption Scheme* **ABE$_2$**)**.** For the class of predicate circuits ρ, the two-outcome attribute-based encryption scheme ABE_2 is represented as Boolean circuits having input bits ν and only one output bit. ABE_2 has the following four algorithms ($ABE_2.Setup$, $ABE_2.Enc$, $ABE_2.KeyGen$, $ABE_2.Dec$) [18]:

- $(msk, mpk) = ABE_2.Setup(1^\kappa)$: Given input 1^κ, $ABE_2.Setup$ outputs the master secret key msk and master public key mpk.
- $sk_p = ABE_2.KeyGen(msk, \rho)$: Given inputs msk and predicate circuit $P \in \rho$, $ABE_2.KeyGen$ generates secret key sk_p.
- $c = ABE_2.Enc(mpk, x, b_0, b_1)$: Given inputs attribute $x \in \{0, 1\}^*$, mpk and two messages b_0, b_1, $ABE_2.Enc$ generates ciphertext c.
- $b_i = ABE_2.Dec(sk_p, c)$: Given inputs sk_p and ciphertext c, $ABE_2.Dec$ generates the output b_i if $P(x) = i$, for $i = 0, 1$. Security of $ABE_2 scheme$ is that if sk_p is revealed then one of the two messages can be decrypted based on $P(x)$ value, where x represents the attribute.

There are three phases of PaCTAC protocol, namely: PaCTAC policy generation Phase (see Section 4.1), PaCTAC attribute value generation phase (see Section 4.2), and (iii) Proxy policy evaluation phase (see Section 4.3). The authorization revocation phase is presented in Section 4.4, and the protocol flow in Section 4.5.

4.1. PaCTAC policy generation phase

In PaCTAC, a user who belongs to tenant T2 is allowed to have access to resources from tenant T1, if and only if, it is allowed by the delegated policies or has cross domain access of T1. For cross tenant delegation, an administrator or a user in T1 makes a delegation request to an internal authorization service. At T1, the delegation request is checked against the defined delegation control policies. Therefore, PaCTAC policy is the combination of information in the delegation request and in the delegation control policies. T1 has its own internal authorization service to approve the requests from its local users. If the administrator wishes to delegate the permissions to any user in another domain, then it is assumed that the local authorization provides an interface for helping the user to make a decision on a set of access actions and objects the user wishes to delegate, along with constraints such as time validation. Now the authorization service approves the request based on pre-defined delegation policies.

Formally, the delegation control policy can be defined as a set of rules, where each rule defines the individual's permissions, constraints and delegation status. The delegation status is a Boolean value, which specifies if the permission is delegatable (or not) to the delegator (role or user) in T1. Every delegation request is garbled against delegation control policies in T1. Delegation request is a set of attributes. If the delegation request is allowed by the delegation control policy, then it is approved by T1's internal authorization service. Afterward, a cross tenant delegation policy is generated and encrypted using RGC.

4.1.1. Garble function generation

T1 generates the policy for the resources, say policy(a_1, a_2) is for resource R1, and shares attributes a_1 and a_2 with T2. T1 takes π as the function or garbled gate.

- The function π takes x_i, y_i as inputs, where $i \in [0, 1]$ and labels these inputs with $k_x^i, k_y^i \in_R GF(q)^n$.
- T1 takes random vectors $e_x^i, e_y^i \in GF(q)^n$, randomizes the vectors $(a_{\pi,0}, a_{\pi,1}, a_{\pi,2})$ defined by $r_\pi \in_R GF(q)^k$ and $c_\pi \in_R GF(q)^{\eta-k}$.
- Permuting the vectors produces the permuted vectors $(a_{\pi,0'}, a_{\pi,1'}, a_{\pi,2'}, \ldots)$. These permuted vectors are then combined with b_f (the column vector that describes the garbled gate π) giving $(b_f, a_{\pi,0'}, a_{\pi,1'}, a_{\pi,2'}, \ldots)$.
- $(mpk, msk) = ABE_2.Setup(1^k)$: On input of security parameter (1^k), $ABE_2.Setup$ outputs the master sk_π and mpk.
- $sk_\pi = ABE_2.KeyGen(msk, \pi)$: On input msk and garbled gate π, $ABE_2.KeyGen$ outputs the sk_π.
- The policy attributes $(a_1, a_2, ..)$ are embedded with k_c providing $(a_{1'}, a_{2'}..)$. Then, k_c checks the conditional operators $(>, <, \leq, \geq, =)$ for the policy attributes and is assigned to the matched attributes, which fulfill the policy requirements. Say a_1 attribute contains the age of the user and its condition is, age should be > 20.
- $c = ABE_2.Enc(mpk, \pi, a_{1'}, a_{2'}..)$: Given inputs msk, the garbled gate π, and attributes embedded with k_c, $ABE_2.Enc$ outputs the garbled function. T1 sends this garbled function to CPDP. When a user makes a request, T1 directs the authentication request to CPDP.

4.2. Attributes value generation phase

CPDP redirects the authentication request from T1 to T2, for the authentication of the user making the request. T2 submits the values of attributes to CPDP requested by T1. T1 sends its generated random vectors (r_x, r_y, k_s) to T2, which may be used later in producing the garbled inputs.

Garbled Input Generation:

- T2 takes as inputs the user's credentials $i_x, i_y \in [0, 1]$ and chooses the random vectors sent by T1, $r_x, r_y \in GF(q)^k$ and $k_s \in_R GF(q)^n$ and $e_S[j]$.
- T2 encodes these inputs with random vectors giving garbled inputs, $c_x = (r_x G + e_{x,i_x} + e_S)$ and $c_y = (r_y G + e_{y,i_y} + e_S)$, where c_x, c_y are the garbled inputs. Say $Y = c_x, c_y$, T2 sends these garbled inputs to CPDP for evaluation.

4.3. Proxy policy evaluation phase

CPDP, the proxy cloud of our protocol, evaluates the policy on behalf of T1 and sends the delegation status back to T1. This delegation status is a Boolean value indicating the granted or rejected status. For the evaluation of the attribute values submitted by T2 and the policy garbled function submitted by T1, T1 sends sk_π generated by $ABE_2.KeyGen$ and $k'_{c'}$ to CPDP.

As an example, CPDP ensures that the requested condition is met. Say $a_1 =$ age and $c_x = 22$, if the age of the user is more than 20 (the condition is embedded with the attributes using k_c), then the user is permitted access to the resource. Does the user actually meets the policy requirements? This is enforced by the CPDP and then this evaluation is passed to T1 having the delegation status (granted or rejected) as Boolean value.

Attribute Evaluation:

- $b_i = ABE_2.Dec(sk_\pi, c)$: On input sk_π for the garbled gate π and cipher text c which is calculated using embedded attributes (with k_c), $ABE_2.Dec$ outputs b_i where $b_i = X$.
- For $i' = 0, 1, 2, 3$, CPDP checks $((c_x + c_y + a_{\pi,i'})\, b_\pi) = 0$. Assuming $((c_x + c_y + a_{\pi,i'})\, b_\pi) = 0$, CPDP generates the garbled output: $(c_x + c_y + a_{\pi,i'})$.
- Using a probabilistic time algorithm, CPDP checks whether the values submitted by T2 (denoted as Y), and policy conditions asked by T1 (denoted as X), are indistinguishable using the following equation:

$$|P_r[D(X_k, 1^k) = 1] - P_r[D(Y_k, 1^k) = 1]| = negl(\kappa)$$

- After the successful evaluation, CPDP sends grant signal to T1. If the attributes match with the policy, then the request generated by the user of T2 is permitted access to the resource R1 held by T1, for which $policy = (a_1, a_2)$ is generated; otherwise, permission is denied.

- *Scalable Security and Flexible Constraints:* The PaCTAC protocol supports flexible policies in the cross-domain environment, especially in situations where the user who is not registered with the service provider to whom the resource request is being made. The framework also supports dynamic constraints such as Chinese Wall policy and Separation of Duty (SoD) among multiple domains.

4.4. Authorization revocation phase

Revocation in our PaCTAC protocol works on two levels, namely: access token revocation and policy revocation. Delegation policy's revocation is performed through the interface provided by the CPDP, as this is available to the registered domains. For policy revocation, policy *ID* and the domain name are sent to CPDP. After verifying the revocation request, the policy is removed from the CPDP's database. Revocation of the active session is performed directly by T1 via revoking of the access token associated with that session. This restricts the user's access to the particular resource. In the event that the user's attributes are changed in his/her home domain (e.g. revoked by T1), T1 will send the updated information of the user to CPDP, which can then evaluate the permissions of the user on the basis of the updated information. If user access needs to be revoked, then CPDP issues request of revocation to T1. The same session ID (active session) is used so that T1 can make the corresponding actions. *Formally:*

$$PaCTACpolicyGen : (DR \oplus p) \longrightarrow [PaCTACP|Error], \quad (1)$$

where PaCTAC policyGen is a mapping from the Delegation Request (DR) set and the set of control policy (P) to a set of PaCTAC policies (PaCTAC P) or error. The operator $\oplus$ verifies the matching of DR against the set of P.

There are multiple control policies, which can satisfy a single request from a user of T2; thus, generating many authorization policies. For example, one delegation request is being made by a researcher in an institute for accessing the research paper of another institute. This requires access to the resource of the institute whose research paper is to be accessed. The following query is made:
DelegatedQuery (User1, Inst1, Access Resource, Designation="MS Scholar" and Life Time=1 day),
where *User*1 is requesting access permission to a resource of another institute *Inst*1 with the constraint that user from the *Inst*1 should be MS scholar.

In this scenario, delegation control policy is verifying whether the PaCTAC policy exists and if so, whether this policy satisfy the above generated request. In the event that the PaCTAC policy does not exist, the delegation request will be denied.

4.5. Protocol flow

A user from T1 makes a request to access the resource of T2 by accessing http://T1.com. The work flow of PaCTAC is shown in Fig. 1. It is assumed that the user has not been granted access prior to this request at T1. Therefore, the user does not present any delegation permit with his/her access request. It is also assumed that T1 recognizes the user as an external user, through the IP address of the user.

In order to request authentication, T1 generates a request token and a policy for the resource a user wishes to access and sends these policy attributes to T2. T1 then generates the garbled circuit for that policy and makes the garbled function of that policy. T1 takes the function π and labels the inputs with keys $(k_x^i, k_y^i) \epsilon_R GF(q)^n$. The labeled inputs are randomized with random vectors and then permuted. Two-outcome attribute based encryption runs its setup phase and KeyGen algorithms as shown in Fig. 1.

Conditional attributes are embedded with K_c after the encryption of these embedded attributes. T1 produces the garbled function and sends it to CPDP. After placing the garbled function of the policy for resource R1, it redirects the request including request token, session-ID and permission required by the user's access.

After the redirection of user at CPDP, T1 sends random vectors (r_x, r_y, k_s) to T2 and sk_π along with K_c to CPDP. The user in turn is redirected by CPDP to T2. This redirection includes the name of domain (tenant consuming the resource) and session-ID. The user then attempts to authenticate by providing his/her credentials attributes to T2. T2 encodes the attributes (x_i, y_i) sent by T1. The credential attributes of user are embedded with k_c for conditional operators $(<, >, =, \leq, \geq)$ and then encrypted to c'_x, c'_y to maintain the privacy of user's information from CPDP. Finally, c'_x, c'_y are sent to CPDP for evaluation.

CPDP first decrypts the received values using RGC placed at CPDP. If the decryption is successfully performed using K_c, then the attributes provided by the user are valid and permission grant signal is sent to T1; otherwise, CPDP denies the access.

Enforcement of cross-domain security constraints is one of the functions of PaCTAC. CPDP has the capability to enforce dynamic and flexible constraints as a policy decision point. This capability is explained as the Chinese Wall policy, which states that two domains are in conflict of interest in a way that resource in domain 1 cannot be accessed by a user having an active session in domain 2 at the same time. PaCTAC maintains the lists of active sessions for its domains in order to address this issue. When a new request is generated from T2 to access the resource from T1, CPDP checks if the same T2 appears on the active list of T1. If so, then the request is denied.

Once the authenticity verification is performed, T1 generates the access token based on request token to allow the access to the resource and the session is terminated after the user's activities are completed.

5. Formal analysis of PaCTAC

PaCTAC and its algorithms are defined in Section 5.1 to 5.3.

5.1. Policy generation algorithm

In the delegation algorithm, permissions m_l can only be delegated by a user having the required permissions. Policies are generated in the delegation algorithm to satisfy the set of conditions for the resource being shared by T1. It checks whether any permission has one or more conflicts with another permission, and whether the user is from a local-domain or cross-domain and the associated set of permissions.

It contains the RGC with policy attributes as its input and makes a garbled function out of this to submit to CPDP. The algorithm uses k_c embedded with the policy attributes x_i, y_i to check the conditional operators $(=, <, >, \leq, \geq)$ associated with the conditions. It also runs the ABE protocol to generate the garble function.

The delegation algorithm is based on Definitions 7 and 8, and the permission delegation is performed at two levels, namely: user and domain levels. At the user level, a user delegates to another user, and at the domain level, a user delegates to a domain. The policy generation algorithm is described in Algorithm 1.

Let t_l belongs to local users and t_j, t_k belongs to cross-domain users and d represents the domain in the delegation algorithm. First, both input and output of the algorithm are defined. Then, a user is checked to determine whether the user belongs to local-domain or cross-domain in lines (1–6). If the user belongs to local-domain, then the permissions m_l are checked in lines (7–9). External to cross-user delegation is in lines (10–12). If a user belongs to the pre-defined set CDP_a, then inputs are labeled with keys k_x^i, k_y^i,

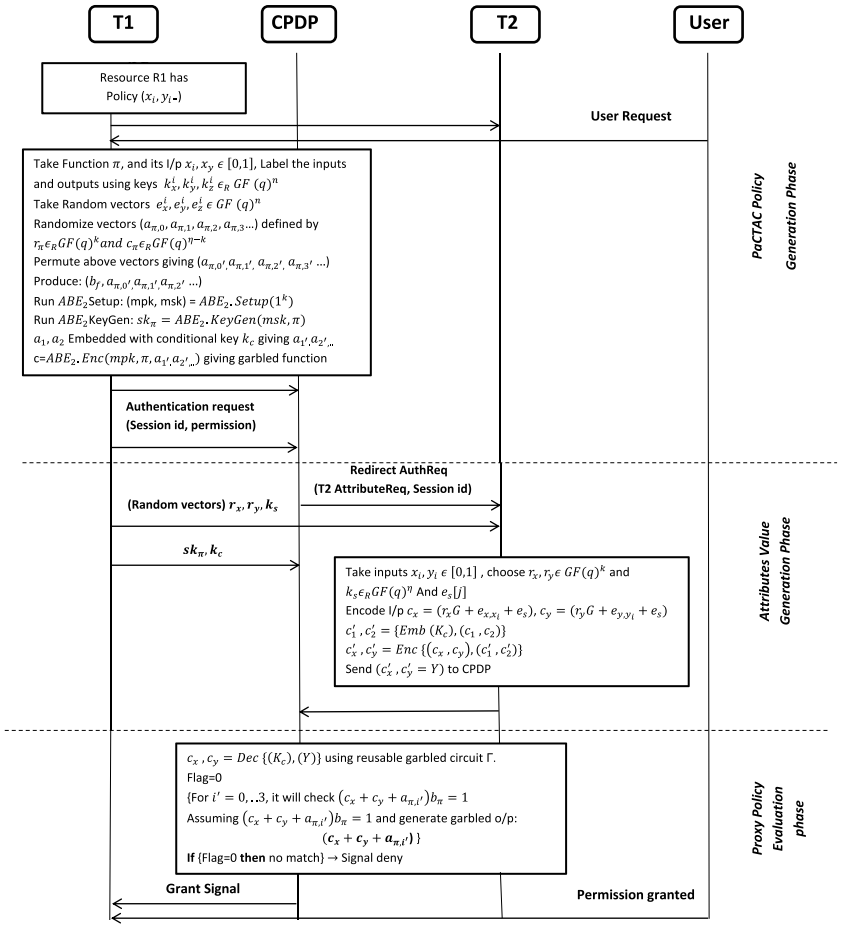


Fig. 1. Resource Access in Proposed PaCTAC Protocol.

and randomized with random vectors e_x^i, e_y^i before being permuted to give a garble function (b_f, $a_{\pi,i'}$) in lines (13–16). ABE_2 protocol is run, having three algorithms ($ABE_2.Setup$, $ABE_2.KeyGen$, $ABE_2.Enc$), and k_c is embedded with the policy attributes. $ABE_2.Enc$ is performed on the embedded attributes to form c, which is sent to CPDP in lines (17–21). If the user does not belong to the predefined set PAU_a and does not have the appropriate permissions assigned, then the algorithm returns false in lines (22–25).

5.2. Attribute verification algorithm

The attribute verification algorithm activates the permissions for both t_l and t_j, t_k users, generates the attributes and evaluates the attributes. Local-domain is for internal users who delegate the permissions and cross-domain users who obtain the appropriate permissions from internal users. Delegation of permissions is achieved by checking both local and cross domain users to determine whether the user belongs to the set of delegated permissions.

Attributes submitted by T2 are encrypted as a part of the RGC, which generates the garbled inputs (c_x', c_y'). These garbled inputs are given to CPDP which evaluates these inputs against the policy, making sure that submitted attributes match the policy attributes.

This algorithm is based on Definitions 4, 5 (for cross user delegation), and 9. The attribute verification algorithm is shown in Algorithm 2. The policy for resource R1 is generated and sent to T2 in lines (1–5). T1 checks if the user belongs to the set of delegated permission and directs the request for authentication to CPDP, where CPDP further redirects the request to T2 in lines (6–10). The attributes submitted by T2 belong to pre-defined set CDP_a. These attributes are embedded with the K_c and encrypted with garbled values (c_x, c_y) to form c_x', $c_y' = Y$. They are then sent to CPDP — see lines (11–19). If these submitted attributes by T2 is successfully decrypted, then the grant signal (true) is sent to T1 by CPDP (proxy cloud); otherwise, the false signal is returned — see lines (20–24).

5.3. Permission revocation algorithm

The permission revocation algorithm allows one to revoke permissions from the user by checking if the user belongs to PAU_i (where $PAU_i \subseteq (t_l \times m_l)$), or the attributes match the published policy. The algorithm is based on Definition 9.

If the user and permissions from PAU_i (where $PAU_i \subseteq (t_l \times m_l)$) and attributes do not match, then the algorithm remove the user and permission pair from this set in lines (1–5). Next, it checks whether the user has delegated the permissions to any other cross-domain user. If yes, then the corresponding delegation is removed in lines (6–8), before checking whether these cross-domain user(s) has/have further delegated the permissions. If it is determined that the domain user(s) has/have further delegated the permissions, then they are removed from the delegation triple (local user, cross-domain user and m_l) in lines (9–11). Now, the domain level permissions is checked. If any user belongs to one of these domains i.e. $t_{l,j,k}$ and delegates the permissions to any other domain(s), then the permissions assigned to local users of the delegated domain(s) is revoked at the user and at the domain level in lines (12–18).

Algorithm 1 Policy Generation Algorithm

procedure POLICY_GENERATION
  $DelAlgo\{(x_i, y_i), u_l|u_j, d, m_l, c^{'}\}$
  Output:$\{(b_f, a_{\pi,i'})\}$
  if (l=t) **then**
    if $(t_l, m_l) \epsilon PAU_a$ **then**
      $PAU^{'}_a = PAU_a \bigcup (t_l, m_l)$
    else{
      if $(t_l, t_j, m_l) \epsilon \{t_l \rightsquigarrow^{m_l} t_j\}$ **then**
        $LCP^{'}_a = LCP_a \bigcup (t_l, t_j, m_l)$ }
      else{
        if $(t_j, t_k, m_l) \epsilon \{t_j \rightsquigarrow^{m_l} t_k\}$
          $CDP^{'}_a = CDP_a \bigcup (t_j, t_k, m_l)$ **then**
          E = Enc $\{(k^i_x, k^i_y), (x_i, y_i)\}$
          $a_{\pi,i} = Randomize\{(e^i_x, e^i_y), (E)\}$
          $a_{\pi,i'} = Permute\{(a_{\pi,i})\}$ }
          $a_{\pi,i'} \rightarrow (b_f, a_{\pi,i'})$
          Run$ABE_2.Setup$
          Run$ABE_2.KeyGen$
          $a_{1'}, a_{2'} \rightarrow Emb\{(k_c), (a_!, a_2)\}$
          $c \leftarrow ABE_2.Enc(mpk, \pi, a_{1'}, a_{2'})$
          send c to CRMS }
        else
          if $(t_l, m_l) \notin PAU_a$ **then**
            if $(t_l, t_j, m_l) \notin \{t_l \rightsquigarrow^{m_l} t_j\}$ && $(t_j, t_k, m_l) \notin (t_j \rightsquigarrow^{m_l} t_k)$ **then**
              return false
            end if
          end if
        end if
      end if
    end if
  end if
end procedure

Algorithm 2 Attribute Verification Algorithm

procedure ATTRIBUTE_VERIFICATION
  $ActQ : \{(c_1, c_2), u_j|u_k, d, m_i\}$
  Output:$\{(Y), GrantSignal\}$
  if $R1 \rightarrow Policy(x_i, y_i)$ **then**
    send to T2
  else{
    if $(t_j, t_k, m_l) \epsilon \{t_j \rightsquigarrow^{m_l} t_k\}$
      $CDP^{'}_a$=$CDP_a \bigcup \{t_j, t_k, m_l\}$
      CPDP $\leftarrow$ AuthReq (Session ID, permission)
      T2 $\leftarrow$ Redirect (T2AttrReq , Session ID) } **then** {
      if $(c_1, c_2) \epsilon CDP_a$ **then**
        $Enc^{'} = c_x \leftarrow (r_x G, e_x, e_{x,ix}),\ c_y \leftarrow (r_y G, e_y, e_{y,iy})$
        $c^{'}_1, c^{'}_2 = Emb\{(K_c), (c_1, c_2)\}$
        $c^{'}_x, c^{'}_y = Enc\{(c_x, c_y), (c^{'}_1, c^{'}_2)\}$
        $Y \leftarrow (c^{'}_x, c^{'}_y)$
        send Y to CPDP }
      else
        if $(c_1, c_2) \notin CDP_{[a]}$ **then**
          Go to return
        else
          if { **then**$c_x, c_y = Dec\{(K_c, (Y))\}$
            send true $\rightarrow$ T1 }
          else
            return false
          end if
        end if
      end if
    end if
  end if
end procedure

6. Case study: a distributed health care information system

6.1. Scenario

Dr. John is a cancer specialist with Hospital X, and one of his new patients (Eve) presents some cancer symptoms. Hence, Dr. John wishes to obtain a second opinion from another specialist, say Dr. Bob from Hospital Y. The patient's medical data is highly sensitive, and therefore, Dr. John uses the proposed PaCTAC framework deployed at Hospital X. First, Dr. John obtains access to the system and selects Eve's data. An administrator (e.g. a clerk) at Hospital X grants the permission, and the policy is stored at CPDP in the form of garbled function using RGC. The CPDP does not know what is the policy. After Dr. Bob has been assigned the authority to view Eve's data, the relevant attributes are generated and evaluated on CPDP. Once the evaluation has been successfully completed, a grant signal from CPDP is given to Hospital X and Dr. Bob is then able to read the data (and in this context, the Electronic Health Record (EHR) of patient Eve) and perform his review. The system diagram of the case study can be seen in Fig. 2.

We formally specify the components in this case study to be as follows:

• $T_X(Dr.John), T_Y(Dr.Bob), M_X(read\_EHR)$

Dr. John belongs to T_X, where T_X is a local-domain. Dr. John is a local user who is acting as a delegator of the permissions. The permissions m_X belong to the set M_X, where $(read\_EHR)$ is the permission for reading the patient's EHR. Dr. Bob belongs to T_j set, a set of cross-domain users, and acts as a delegator (in further delegation of permission) or delegatee (when permissions are delegated to it).

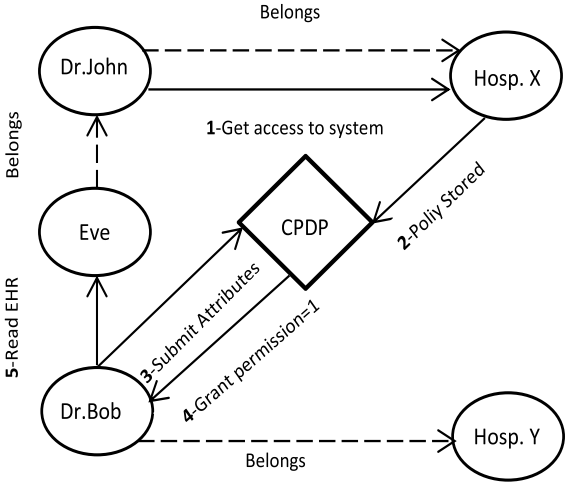


Fig. 2. Flow diagram of Case Study.

• $T_{d=X} : D \rightarrow 2^{(sub)}$, where $sub = T_{X \epsilon D} T_X$ (i.e, the set of users in X domain (local domain) are $\{\{\}, Dr.John\}$).

• $T_{d=Y} : D \rightarrow 2^{(sub)}$, where $sub = T_{Y \epsilon D} T_Y$ (i.e, the set of users in Y domain (cross domain) are $\{\{\}, Dr.Bob\}$).

• $PAU_i \subseteq (T_X \times m_X)$. A binary relation between the local-domain and the set of permissions assigned in the X domain (i.e., $PAU_i \subseteq (Dr.John, m_X(\{read\_EHR\}))$).

Algorithm 3 Permission Revocation Algorithm

procedure PERMISSION_REVOCATION
  $RevoAlgo : \{(c_1, c_2), u_j | u_k, d, m_i\}$
  Output: $s_7^{'}, s_8^{'}, s_9^{'}, s_{10}^{'}$,
  if $[(c_1, c_2) \neq (x_i, y_i)]$ && $[(t_l, m_l) \epsilon PAU_i]$ **then**
    $PAU_i^{'} = PAU_i - (t_l, m_l)$
    if $(t_l, t_k, m_l) \epsilon \{t_l \rightsquigarrow^{m_l} t_j\}$ **then**
      $\{t_l \rightsquigarrow^{m_l} t_j\}' = \{t_l \rightsquigarrow^{m_l} t_j\} - (t_l, t_j, m_l)$
      $LCPC^{'} = LUED - (t_l, t_j, m_l)$
      if $(t_l, t_k, m_l) \epsilon \{t_j \rightsquigarrow^{m_l} t_k$ **then**
        $\{t_j \rightsquigarrow^{m_l} t_k\}' = \{t_j \rightsquigarrow^{m_l} t_k\}' - (t_j, t_k, m_l)$
        $UCPC^{'} = UCPC - (t_j, t_k, m_l, c^{'})$
        if $(t_l, d, m_l) \epsilon \{t_l \rightsquigarrow^{m_l} d$ **then**
          $\{t_l \rightsquigarrow^{m_l} d\}' = \{t_l \rightsquigarrow^{m_l} d\} - (t_l, d, m_l)$
          $\{d \rightsquigarrow^{m_l} t_j\}' = \{d \rightsquigarrow^{m_l} t_j\} - (d, t_j, m_l)$
          $LDPC^{'} = LDPC - (t_l, d, m_l, c^{'})$
          if $(t_k, d, m_l) \epsilon \{t_k \rightsquigarrow^{m_l} d\} - (t_k, d, m_l)$ **then**
            $\{d \rightsquigarrow^{m_l} t_j\}' = \{d \rightsquigarrow^{m_l} t_j\} - (d, t_j, m_l)$
            $CDPC' = CDPC - (t_k, d, m_l, c^{'})$
          else
            do nothing
          end if
        end if
      end if
    end if
  end if
end procedure

• $PAU_a \subseteq (T_X \times m_X)$. A binary relation between the local-domain and the set of permissions activated in the X domain (i.e., $PAU_i \subseteq (Dr.John, m_X(\{read\_EHR\}))$).

• $LCP_a \subseteq (T_X \times T_Y \times m_X)$. A triple relation among the local-domain user, cross-domain user and the set of permissions assigned in the X domain (i.e., $PAU_i \subseteq (Dr.John \times Dr.Bob \times m_X(\{read\_EHR\}))$).

• $LCPC \subseteq (T_X \times T_Y \times m_X \times c')$. A quadruple relation among the local-domain user, cross-domain user, the set of permissions assigned in the X domain and the constraints (i.e., $PAU_i \subseteq (Dr.John \times Dr.Bob \times m_X(\{read\_EHR\}) \times c')$), where c' represents any age or specialization, etc.

• Let (Secret Key for Input) be $sk_i = E.KeyGen(\kappa)$ and (Secret Key for Output) be the $sk_o = E.KeyGen(\kappa)$. Circuit C is revised in such a way that C takes encrypted inputs $E.Enc(sk_i, x)$ instead of plaintext input x. Before computing $C(x)$, C decrypts the encrypted inputs $E.Enc(sk_i, x)$. After the computation, C encrypts $C(x)$ as garbled output $E.Enc(sk_o, C(x))$.

• Security Parameter:(κ) are used as the security parameter. Other parameters n, k, η and q are determined from it, where $[n/4] \geq \eta > k$; $n = 17, k = 2, \eta = 4$ and $q = 19$.

• Let λ be the length of garbled output $E.Enc(sk_o, C(x))$ where $\lambda = \lambda(\kappa)$ under κ.

$$ABE_2 \sum_{m=0}^{\lambda}(m) \quad \text{where } m = 0, 1, 2, 3, \ldots, \lambda$$

$ABE_2.Setup(1^k)$ runs λ times.

$sk_i \leftarrow ABE_2.KeyGen(msk, C^{-i}(.))$ for $i < \lambda$, where $C^{-i}(.)$ represents the output bits of the circuit C.

• We have K_c that is associated with the condition, or the conditional operator $(<, >, =, \leq, \geq)$ if the policy constraint has conditions such as $age \geq 20$. Then, this K_c is embedded with the attributes provided by hospital Y and can only be decrypted when the required condition is met. For example, when Dr. Bob's expertise level is greater than XXX, then the provided attributes are decrypted and he is allowed to read the patient's data.

• RGC $\gamma = (sk, \bar{C})$ and $sk = (mpk, e_{i,j}, G, A, sk_i, sk_o)$ as secret key, taking mpk, row vector, matrices G and A, sk_i and sk_o as its inputs respectively. Let $\bar{C}$ denotes the linear code garbled circuit.

• User Activation: Two activation queries for distributed health system are defined. First, when Dr. John activates the permission as an activation for local users as:

$$ActQ : T_X \times X \times m_X \rightarrow \{PAU_a'\}$$

For activation of local user, it takes Dr. John, Hospital X and the permission for reading the patient's data. Second activation occurs when Dr. Bob activates the permission. Activation for cross users are:

$$ActQ : T_Y \times Y \times m_X \rightarrow \{LCP_a'\}$$

Activation query for cross-domain user takes Dr. Bob, Hospital Y and the permission for reading the patient's data for activation.

• Delegation Across the Domain: According to Definition 5, cross-domain delegation is formally defined as a triple (T_X, T_Y, M_X), where (T_X) is a set of user acting as delegator, (T_Y) set of users as delegatee and (M_X) as permissions. Formally, it can be defined as:

$$(T_X, T_Y, m_X) \epsilon \{T_X \rightsquigarrow^{m_X} T_Y\}$$

i.e

$$(Dr.John, Dr.Bob, read\_EHR) \epsilon \{T_X \rightsquigarrow^{m_X} (\{read\_EHR\}) T_Y\}$$

Dr. John is a delegator, Dr. Bob is a delegatee and *read_EHR* are permissions that are delegated from delegator to delegatee.

• Delegation Query: According to Definition 7, delegation can formally be defined as:

$$DelAlgo : (T_X \times T_Y \times m_X \times c') \rightarrow \{T_X \rightsquigarrow^{m_X} T_Y\} | \{LCPC'\}$$

Input as $(T_X \times T_Y \times m_X \times c')$ shows delegator, delegatee, delegated permissions and constraint respectively.

• Permission Revocation: The permission revocation algorithm is activated when permissions are revoked or when attributes of the delegatee mismatch. Permission revocation can formally be defined as:

$$RevAlgo : (T_X \times m_X \times (i_x, i_y)) \neq (a_1, a_2) \rightarrow (T_X \times m_X) \epsilon PAU_i,$$

where T_X is a local user of Hospital X, T_Y is a cross user that belongs to Hospital Y, m_X are the permissions, (x_i, y_i) are inputs (attributes of cross user) and (a_1, a_2) are the policy attributes.

• Randomization: Random function used in *DelAlgo* and during the generation of Dr.Bob's policy includes the following steps:

- Random vectors $(e_x^i, e_y^i \epsilon GF(q)^n)$ are taken.
- Randomization of the encrypted inputs with these random vectors giving $a_{\pi,i}$

  $$randomize\{(e_x^i, e_y^i), E\},$$

where E is the encrypted input: $E = Enc\{(k_x^i, k_y^i), (x_i, y_i)\}$

The cross-domain resource access for this distributed health scenario is implemented using the following algorithms that are specified for PaCTAC:

• Activation for Dr. John (local-user)
• Activation for Dr. Bob (cross user)
• Permission Validation
• Reusable Garbled Circuit for Policy
• Attribute Evaluation at CPDP
• Permission Revocation

6.1.1. Activation for Dr. John (local-domain user)

Activation for local users is shown in Algorithm 4. Dr. John is checked to determine whether it belongs to local domain Hospital X - see line (3). Since Dr. John is determined to be local, PAU_i is checked to determine if Dr. John has permission to *read_EHR* - see line (2). If yes, then Dr. John can activate the permission and the set is updated in line (3).

Algorithm 4 Activation for DR.John (Local-User)

1: **procedure** ACTIVATION_LOCAL-USER
2: **if** (Dr.John is a local user in domain "X") **then**
3: **if** $(Dr.John, \{read\_EHR\}) \epsilon PAU_a$ **then**
4: $PAU_a^{'} = PAU_a \bigcup (Dr.John, \{read\_EHR\})$
5:
6: **end if**
7: **end if**
8: **end procedure**

6.1.2. Activation for Dr. Bob (cross-domain user)

Activation for cross user is shown in Algorithm 5. In line (1), it is checked whether Dr. Bob is a local user prior to checking in *CDP* whether Dr. Bob has the required permissions ($\{read\_EHR\}$) in line (2). If yes, then the set is updated in line (3), and Hospital X sends (c_x, c_y) to Y in line (4). Then, the policy inputs are embedded with K_c and encrypted with the ciphertext. The results are sent to CPDP in lines (5–7).

Algorithm 5 Activation for Dr. Bob (cross-user)

1: **procedure** ACTIVATION_CROSS-USER
2: **if** (Dr. Bob is not a local user in domain "X") **then**
3: **if** $(Dr.Bob, \{read\_EHR\}) \epsilon CDP$ **then**
4: $CDP^{'} = CDP \bigcup (Dr.Bob, \{read\_EHR\})$
5: X sends (c_x, c_y) to Y
6: where $c_x = (r_x G + e_{x,i_x} + e_s)$, $c_y = (r_y G + e_{y,i_y} + e_s)$
7: $c_1^{'}, c_2^{'} = Emb\{(K_c), (c_1, c_2)\}$
8: $c_x^{'}, c_y^{'} = Enc\{(c_x, c_y), c_1^{'}, c_2^{'}\}$
9: **send** $(c_x^{'}, c_y^{'})$ to CPDP
10: **end if**
11: **end if**
12: **end procedure**

6.1.3. Permission validation

The permission validation algorithm (see Algorithm 6) checks whether particular permissions are assigned to the users. If the user is determined to belong to local domain X, then the set PAU_a is checked to determine whether it has permissions. If yes, then this set is updated in lines (1–4). If Dr. John belongs to a local-domain, then the permission checking occurs in lines (5–7), and external to cross-user delegation occurs in lines (8–9).

6.1.4. RGC for policy

The generation of RGC for the policy is shown in Algorithm 7. Encryption of the policy attributes (x_i, y_i) with the keys (k_x', k_y') is performed. Those encrypted values are randomized with random vectors e_x^i, e_y^i and then permuted to form $(b_f, a_{\pi,i'})$ in lines (1–4). $ABE_2.Setup$ and $ABE_2.KeyGen$ are run in lines (5–6). The policy attributes are embedded with K_c and $ABE_2.Enc$ is performed at lines (7–8). The resultant c is then sent to CPDP in line (9).

Algorithm 6 Permission Validation

1: **procedure** PERMISSION_VALIDATION
2: **if** (X is a domain of local user) **then**
3: **if** $(Dr.John, \{read\_EHR\}) \epsilon PAU_a$ **then**
4: $PAU_a^{'} = PAU_a \bigcup (Dr.John, \{read\_EHR\})$
5: **else**
6: **if** $(Dr.John, Dr.Bob, \{(read\_EHR)\}) \epsilon LCP_a$ **then**
7: $LCP_a^{'} = LCP_a \bigcup (Dr.John, Dr.Bob,$
8: $\{read\_EHR\})$
9: **else**
10: **if** $(Dr.Bob, \{(read\_EHR)\}) \epsilon CDP$ **then**
11: $CDP^{'} = CDP \bigcup (Dr.Bob, \{read\_EHR\})$
12: **end if**
13: **end if**
14: **end if**
15: **end if**
16: **end procedure**

Algorithm 7 RGC for Policy

1: **procedure** REUSABLE_GARBLED
2: $E = Enc\{(k_x^{'}, k_y^{'}), (x_i, y_i)\}$
3: $a_{\pi,i} = Randomize\{(e_x^{'}, e_y^{'}), (E)\}$
4: $a_{\pi,i'} = Permute\{(a_{\pi,i})\}$
5: $a_{\pi,i'} = (b_f, a_{\pi,i'})$
6: **Run** $ABE_2.Setup$
7: **Run** $ABE_2.KeyGen$
8: $a_1^{'}, a_2^{'} \rightarrow Emb\{(K_c, (a_1, a_2))\}$
9: c $\leftarrow ABE_2.Enc(mpk, \pi, a_1^{'}, a_2^{'})$
10: **send** c to CPDP
11: **end procedure**

6.1.5. Attribute evaluation at CPDP

The evaluation of the attributes with the policy is performed at CPDP and its algorithm is shown in Algorithm 8. The garbled inputs c_x', c_y' are submitted by Hospital Y, which are the credentials of Dr. Bob, and are decrypted with K_c at CPDP giving c_x, c_y in line (1). If decryption is performed, then the submitted attributes are according to the policy and as a result grant signal is sent to Hospital X. If decryption is not performed, then the submitted credentials are incorrect. Consequently, a reject signal is sent to Hospital X in lines (2–6).

Algorithm 8 Attribute Evaluation at CPDP

1: **procedure** ATTRIBUTE_EVALUATION
2: $c_x, c_y = Dec\{(K_c), (c_x^{'}, c_y^{'})\}$
3: **if** (Decryption is performed) **then**
4: **send** granted to X
5: **else**
6: **send** not granted to X
7: **end if**
8: **end procedure**

6.1.6. Permission revocation

The permission revocation algorithm is shown in Algorithm 9. If the credentials submitted by Dr. Bob (c_1, c_2) do not match with the policy attributes submitted by Hospital X to the garbled circuit (c_x, c_y) or Dr. John does not belong to the set of assigned permissions, then permissions are revoked from the setting in lines (1–3). If Dr. John has further delegated any permissions $\{(read\_EHR)\}$ to

Dr. Bob, then they are removed from the set *LCPC* and set is updated in lines (4–6).

Algorithm 9 Permission Revocation

1: **procedure** PERMISSION_REVOCATION
2: **if** $[(c_1, c_2) \neq (c_x, c_y)]$&& $[(Dr.John, \{read\_EHR\}) \epsilon PAU_i]$ **then**
3: $PAU_i^{'} = PAU_i - (Dr.John, \{read\_EHR\})$
4: **if** $(Dr.John, Dr.Bob, \{read\_EHR\}) \epsilon \{Dr.John \rightsquigarrow^{\{read\_EHR\}} Dr.Bob\}$ **then**
5: $\{Dr.John \rightsquigarrow^{\{read\_EHR\}} Dr.Bob\}' = \{Dr.John \rightsquigarrow^{\{read\_EHR\}}$
6: $Dr.Bob\} - (Dr.John, Dr.Bob, \{read\_EHR\})$
7: $LCPC^{'} = LCPC - (Dr.John, Dr.Bob, \{read\_EHR\})$
8: **end if**
9: **end if**
10: **end procedure**

7. Security and performance analysis

The security analysis of the claimed properties stated in the following theorems is defined as follows.

Theorem 1. *CPDP is capable of enforcing flexible constraints via the Chinese wall policy, which states that a user having access to one domain cannot access the resource in another domain at the same time.*

Proof. Consider a new user that enters the system having *Conflict Class: {Microsoft, Google, Linux}*. This user is free to choose what he/she wishes to know. If he/she first chooses one domain, say *Google*, and then decides to obtain information from another domain, then this would raise a problem. Formally, the Chinese wall policy security is enforced through the following rules:

- *Simple Security Rule (Read Rule):* A user is allowed to access the organization, if and only if, the organization belongs to the same dataset, as the user has already obtained access within the wall or the user belongs to another conflict class.
- *Property (Write rule):* Write access is allowed to the user if the user has the read permission for the same organization and has not read any other organization's information which is in a different dataset to the one on which the write permission is performed.

Theorem 2. *Policies and attributes of users placed at the CPDP are secure because they are in garbled form.*

Proof. It is said that ABE_2 scheme is secure if for all probabilistic polynomial time (PPT) adversary A and for each sufficiently large security parameter κ, the following experiment holds:

$$Pr|Exp_{ABE_2}.A(1^{\kappa}) = 1| \leq 1/2 + negl(\kappa)$$

Messages are being encrypted with respect to the subsets of attributes and policies defined over the set of attributes.

This IBE scheme performs *probabilisticencryption* using Elgamal-like approach (i.e. Boneh–Franklin Scheme), and the BF-IDE scheme has been proven secure. Thus, the security proof rests on new assumption about the hardness of problems in certain elliptic curve group.

Theorem 3. *Policies are being executed in a secure manner as the CPDP performs computations on encrypted data.*

Proof. In the cloud, CPDP data is stored in encrypted form as discussed in Theorem 2, and this paper deals with random linear coding [18]. RGC at CPDP takes Y (Garbled attributes of users) and these attributes are embedded with K_c prior to being encrypted with c_x, c_y (encoded inputs with random vectors r_x, r_y).

The cloud provider decrypts Y with K_c to obtain c_x and c_y in encoded form, which are then evaluated. Specifically, it checks $(c_x + c_y + a_{\pi,i'})b_{\pi} = 0$ prior to granting the permissions.

In this paper, we provide the best application of RGC proposed by Wang et al. in [18]. Specifically, our proposed scheme is based on garbled circuit and provides reusability (i.e., inputs to the circuit are reusable). This reusability is presented by Goldwasser et al. [13] previously but the (original) scheme has a limitation of noise growth on computation on encrypted data. Our scheme uses Random Linear Coding that has reduced computational cost and is reusable at the same time. Not all schemes support reusability (e.g., [12–14,19] in Table 2). In most of the schemes, the user ID is revealed on execution of the setup and in our proposed scheme user identity remains unrevealed.

A comparative summary is presented in Table 2. The performance cost for existing Fully Homomorphic Encryption (FHE) schemes is higher due to the "noisy" computations; thus, the increase in/focus on RGC. The efficiency of our proposed RBC-based scheme is straightforward to observe, and reusability does not leak the inherent information about the input or the circuit itself (Definition 3.8) [18]. Thus, the efficiency of RGC remains that same as that of one-time garbling scheme.

Further performance comparison can be made with Advanced Encryption Standard (AES) encryption. To construct AES through the scheme discussed in [13], a universal circuit of depth 16 is needed. Homomorphic decryption through AES would require approximately 30 mins for one block decryption (128-bits); thus for big programs, say 40,000 bits, it could take 6.5 days for decryption of the circuit [18]. Other optimal existing algorithms for bootstrapping, such as the algorithm presented by Halevi and Shoup [40] takes 5.5 mins for each bootstrapping and for AES it could take 5.5 years to evaluate, at a security level of 76 bits. Our proposed technique uses random linear coding in which evaluation of each gate is converted into inner product of four vectors (i.e., each gate can be evaluated in $4n$ multiplications). Furthermore, our RGC can compute AES decryption of 35,495,055,200 gates in 2^{35} multiplications. We remark that the same AES evaluation would take approximately 16.9 years using Goldwasser et al.'s technique [13].

8. Conclusion and future work

Cloud computing is likely to have broader applications in our society, ranging from civilian to military (e.g. cloud and Internet of Military Things). Thus, the capability to offer authentication and authorization to access services, data and resources in a collaborative cloud environment between different domains will be increasingly crucial.

In this paper, we proposed a Privacy Aware Cross Tenant Access Control (PaCTAC) protocol for cross domain users, which is based on a Reusable Garbled Circuit (RGC) scheme. Our protocol allows the execution and evaluation of policy at a third party entity, while preserving the privacy and storage of policy at the Cloud Policy Decision Point (CPDP). In other words, the cross domain constraint evaluation is achieved in a privacy preserving manner. We also leveraged the widely used Chinese wall policy, which is used to handle conflicts of interest. Furthermore, a garbled circuit was designed for the policy that is placed at the CPDP. The use of RGC allows us to share resources securely across different cloud tenants.

As part of the future work, we plan to extend the design of RGC to include the use of digital comparators for handling conditional

Table 2
Security of existing techniques: A comparative summary.

Paper	Technique	Security	Limitations	Reusability	User ID
Fuzzy IBE [12]	IBE	Security against collusion attack, Error tolerant	Lack expressibility	×	Unrevealed
ABE for fined grained access control of encrypted data [13]	Key policy attribute based encryption		Key issuer is the untrusted party	×	Revealed
A Hierarchical attribute based solution for flexible and scalable access control in cloud computing [15]	HASBE	Efficient and Flexible with access control, Scalable due to hierarchy structure	Compound attributes are not supported efficiently, Not support multiple value assignments [15]	×	Unrevealed
How to generate and exchange secrets [16]	Random number generation from prime numbers	Achieve privacy, validity and fairness, Secure against passive adversaries	Exchanging secrets, Not secure on multiple inputs	×	Unrevealed
Reusable garbled circuit and succinct functional encryption [23]	Succinct single-key functional encryption scheme	Reusable garbled circuit, Secure against passive adversaries	Homomorphic encryption scheme is expensive, Homomorphic operation increases noise [14].	✓	Unrevealed
Cloud computing security using encryption technique [11]	RSA algorithm	Change and update the key frequently, Key is encrypted	Functionality of cloud computing is limited	×	Revealed
Candidate indistinguishability obfuscation and functional encryption for all circuits [17]	Program obfuscation and functional encryption	Functional encryption for all circuits, Indistinguishability obfuscation for all circuits	Not aware of any attack without additional safeguard	×	Revealed
Virtual black box obfuscation for all circuits via generic graded encoding [18]	Graded encoding scheme	No information is leaked, Satisfy indistinguishability obfuscation	Homomorphic encryption scheme is used for obfuscation which is expensive, Time and size complexity	×	Revealed
Attribute-Based Encryption for Circuits [24]	ABE scheme	Secure against standard learning with errors.	Large attribute size	✓	Revealed
Obfuscating Circuits via Composite-Order Graded Encoding [19]	Candidate Obfuscator	Secure against generic algebraic attacks, Robust model against adversary	Size Complexity	×	Revealed
Proposed technique	Random linear coding	Secure against policies, resources and user attribute leakage	All-to-Nothing security [18]	✓	Unrevealed

policies, as well as developing and evaluating a prototype of the enhanced scheme in a real-world environment.

References

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.

[2] Q. Liu, A. Srinivasan, J. Hu, G. Wang, Preface: Security and Privacy in Big Data Clouds, Elsevier, 2017.

[3] D. Borthakur, The hadoop distributed file system: Architecture and design, Hadoop Project Website 11 (2007) (2007) 21.

[4] X. Liu, R. Choo, R. Deng, R. Lu, J. Weng, Efficient and privacy-preserving outsourced calculation of rational numbers, IEEE Trans. Dependable Secure Comput. (2016).

[5] X. Liu, R.H. Deng, K.-K.R. Choo, J. Weng, An efficient privacy-preserving outsourced calculation toolkit with multiple keys, IEEE Trans. Inform. Forensics Secur. 11 (11) (2016) 2401–2414.

[6] J. Li, N. Li, W.H. Winsborough, Automated trust negotiation using cryptographic credentials, in: Proceedings of the 12th ACM conference on Computer and communications security, ACM, 2005, pp. 46–57.

[7] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping, ACM Trans. Comput. Theory (TOCT) 6 (3) (2014) 13.

[8] M. Van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, Fully homomorphic encryption over the integers, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2010, pp. 24–43.

[9] J. Liu, X. Huang, J.K. Liu, Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption, Future Gener. Comput. Syst. 52 (2015) 67–76.

[10] J.-J. Yang, J.-Q. Li, Y. Niu, A hybrid solution for privacy preserving medical data sharing in the cloud environment, Future Gener. Comput. Syst. 43 (2015) 74–86.

[11] D. Boneh, A. Sahai, B. Waters, Functional encryption: Definitions and challenges, in: Theory of Cryptography Conference, Springer, 2011, pp. 253–273.

[12] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2010, pp. 62–91.

[13] S. Goldwasser, Y. Kalai, R.A. Popa, V. Vaikuntanathan, N. Zeldovich, Reusable garbled circuits and succinct functional encryption, in: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, ACM, 2013, pp. 555–564.

[14] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters, Candidate indistinguishability obfuscation and functional encryption for all circuits, SIAM J. Comput. 45 (3) (2016) 882–929.

[15] Z. Brakerski, G.N. Rothblum, Virtual black-box obfuscation for all circuits via generic graded encoding, in: Theory of Cryptography Conference, Springer, 2014, pp. 1–25.

[16] S. Gorbunov, V. Vaikuntanathan, H. Wee, Attribute-based encryption for circuits, J. ACM 62 (6) (2015) 45.

[17] S. Goldwasser, Y.T. Kalai, R.A. Popa, V. Vaikuntanathan, N. Zeldovich, How to run turing machines on encrypted data, in: Advances in Cryptology–CRYPTO 2013, Springer, 2013, pp. 536–553.

[18] Y. Wang, Q.M. Malluhi, K.M. Khan, Garbled computation in cloud, Future Gener. Comput. Syst. 62 (2016) 54–65.

[19] A.C.-C. Yao, How to generate and exchange secrets, in: Foundations of Computer Science, 1986., 27th Annual Symposium on, IEEE, 1986, pp. 162–167.

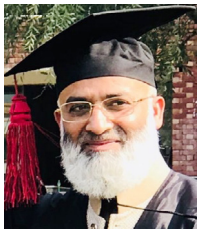
[20] N. Baracaldo, A. Masoumzadeh, J. Joshi, A secure, constraint-aware role-based access control interoperation framework, in: Network and System Security (NSS), 2011 5th International Conference, IEEE, 2011, pp. 200–207.
[21] M. Shehab, E. Bertino, A. Ghafoor, Serat: secure role mapping technique for decentralized secure interoperability, in: Proceedings of the tenth ACM symposium on Access Control Models and Technologies, ACM, 2005, pp. 159–167.
[22] R. Bhatti, E. Bertino, A. Ghafoor, An integrated approach to federated identity and privilege management in open systems, Commun. ACM 50 (2) (2007) 81–87.
[23] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, Proposed nist standard for role-based access control, ACM Trans. Inform. Syst. Secur. (TISSEC) 4 (3) (2001) 224–274.
[24] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, Role-based access control models, Computer 29 (2) (1996) 38–47.
[25] Q. Li, X. Zhang, M. Xu, J. Wu, Towards secure dynamic collaborations with group-based rbac model, Comput. Secur. 28 (5) (2009) 260–275.
[26] Z. Zhang, X. Zhang, R. Sandhu, ROBAC: scalable role and organization based access control models, in: Collaborative Computing: Networking, Applications and Worksharing, 2006, CollaborateCom 2006, International Conference on, IEEE, 2006, pp. 1–9.
[27] E. Freudenthal, T. Pesin, L. Port, E. Keenan, V. Karamcheti, dRBAC: distributed role-based access control for dynamic coalition environments, in: Distributed Computing Systems, 2002. Proceedings 22nd International Conference on, IEEE, 2002, pp. 411–420.
[28] X. Zhang, S. Oh, R. Sandhu, PBDM: a flexible delegation model in rbac, in: Proceedings of the Eighth ACM Symposium on Access control Models and Technologies, ACM, 2003, pp. 149–157.
[29] B. Shafiq, J.B. Joshi, E. Bertino, A. Ghafoor, Secure interoperation in a multidomain environment employing RBAC policies, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1557–1577.
[30] Q. Li, J. Ma, R. Li, X. Liu, J. Xiong, D. Chen, Secure, efficient and revocable multi-authority access control system in cloud storage, Comput. Secur. 59 (2016) 45–59.
[31] Y. Wang, F. Li, J. Xiong, B. Niu, F. Shan, Achieving lightweight and secure access control in multi-authority cloud, in: Trustcom/BigDataSE/ISPA, 2015 IEEE, Vol. 1, IEEE, 2015, pp. 459–466.
[32] M. Henze, L. Hermerschmidt, D. Kerpen, R. Häußling, B. Rumpe, K. Wehrle, A comprehensive approach to privacy in the cloud-based Internet of Things, Future Gener. Comput. Syst. 56 (2016) 701–718.
[33] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of cloud computing and internet of things: A survey, Future Gener. Comput. Syst. 56 (2016) 684–700.
[34] C. Ngo, Y. Demchenko, C. de Laat, Multi-tenant attribute-based access control for cloud infrastructure services, J. Inform. Secur. Appl. 27 (2016) 65–84.
[35] H. Shen, G. Liu, An efficient and trustworthy resource sharing platform for collaborative cloud computing, IEEE Trans. Parallel Distrib. Syst. 25 (4) (2014) 862–875.
[36] K.K. Hingwe, S.M.S. Bhanu, Hierarchical role-based access control with homomorphic encryption for database as a service, in: Proceedings of International Conference on ICT for Sustainable Development, Springer, 2016, pp. 437–448.
[37] A. Almutairi, M. Sarfraz, A. Ghafoor, Risk-aware management of virtual resources in access controlled service-oriented cloud datacenters, IEEE Trans. Cloud Comput. (2015).
[38] A. Sayler, E. Keller, D. Grunwald, Jobber: Automating inter-tenant trust in the cloud, in: HotCloud, 2013.
[39] N. Pustchi, R. Krishnan, R. Sandhu, Authorization federation in iaas multi cloud, in: Proceedings of the 3rd International Workshop on Security in Cloud Computing, ACM, 2015, pp. 63–71.
[40] S. Halevi, V. Shoup, Bootstrapping for helib, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2015, pp. 641–670.

Muhammad Masoom Alam received his P.h.D in information security from University of Innsbruck Austria, in 2007. He currently leads cyber security lab which is involved in various industrial projects in blockchain, threat intelligence and container security. He is an Associate Professor in the Department of Computer Science, CIIT Islamabad, Pakistan. His Ph.D. thesis has got the best thesis award at the Models 2006 conference. He has a total of 17 years experience in teaching, system administration and research and development. His research lab is hosting over 35 graduates and undergraduates.

Naina Emmanuel is the research scholar at Cyber Security lab, Computer Science Department, Comsats Institute of Information and Technology (CIIT), Islamabad. Pursued B.Sc. Computer Engg in (2015). She has recently completed her M.S. degree in Information Security from CIIT in (2017). Her research interests include Homomorphic Encryption, Garbled Computations, Cloud Security and Network Security.

Tanveer Khan received the Master degree in Information Security from COMSATS University Islamabad Pakistan. He is currently working as a Research Associate with the Department of Computer Sciences, Office of Academic Research college of Engineering University of Qatar, Doha, Qatar. His interests include Cryptography and Security.

Abid Khan is working as an assistant professor in computer science department at COMSATS Institute of Information Technology (CIIT), Islamabad. He was a postdoc fellow at Politecnico de Torino, Italy from 20092011. He did his Ph.D. from Harbin Institute of technology, Harbin, P.R. China in 2008. His research interest includes applied cryptography, security and privacy issues in distributed systems, secure provenance. He has more than 10 years of teaching, research and development experience.



Nadeem Javaid received the Bachelors degree in Computer Science from Gomal University, D. I. Khan, KPK in 1995, Masters degree in Electronics from Quid-I-Azam University, Islamabad, Pakistan in 1999, and the Ph.D. degree from the University of Paris-Est, France, in 2010. He is currently an Associate Professor and the founding Director of ComSens (Communications over Sensors) Research Lab, Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan. He has supervised 7 Ph.D. and 75 Masters theses. He has authored over 500 papers in technical journals and international conferences. He is also an Associate Editor of the IEEE Access Journal and an Editor of the International Journal of Space Based and Situated Computing. His research interests include: information security, energy optimization in smart grid clouds and in IoT enabled wireless sensor networks, big data analytics in smart grids, etc.

Kim-Kwang Raymond Choo received the Ph.D. degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed professorship with The University of Texas. He serves on the editorial board of Cluster Computing, Digital Investigation, IEEE Access, IEEE Cloud Computing, IEEE Communications Magazine, Future Generation Computer Systems, Journal of Network and Computer Applications, PLoS ONE, etc. He also serves as the Special Issue Guest Editor of ACM Transactions on Embedded Computing Systems (2017), ACM Transactions on Internet Technology (2016), Computers & Electrical Engineering (2017), Digital Investigation (2016), Future Generation Computer Systems (2016), IEEE Cloud Computing (2015), IEEE Network (2016), IEEE Transactions on Dependable and Secure Computing (2017), Journal of Computer and System Sciences (2017), Multimedia Tools and Applications (2017), Personal and Ubiquitous Computing (2017), Pervasive and Mobile Computing (2016), Wireless Personal Communications (2017), etc. He was named one of 10 Emerging Leaders in the Innovation category of The Weekend Australian Magazine/Microsoft's Next 100 series, and Cybersecurity Educator of the Year—APAC (produced in cooperation with the Information Security Community on LinkedIn) in 2009 and 2016, respectively. He and his team won Germany's University of Erlangen–Nuremberg (FAU) Digital Forensics Research Challenge 2015. He is the recipient of ESORICS 2015 Best Research Paper Award,

Highly Commended Award by Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, the British Computer Society's Wilkes Award for the best (sole-authored) paper published in the 2007 volume of The Computer Journal, and ACISP 2005 Best Student Paper Award. He is also a Fellow of the Australian Computer Society, an IEEE Senior Member, and an Honorary Commander of the 502nd Air Base Wing at Joint Base San Antonio-Fort Sam Houston, USA.

Dr. Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a Future Fellow of the Australian Research Council during 2012–2016. He has authored over 625 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. He also edited several books including "Cloud Computing: Principles and Paradigms" (Wiley Press, USA, Feb 2011). He is one of the highly cited authors in computer science and software engineering worldwide (h-index = 114, g-index = 245, 66,600+ citations). Microsoft Academic Search Index ranked Dr. Buyya as #1 author in the world (2005–2016) for both field rating and citations evaluations in the area of Distributed and Parallel Computing. "A Scientometric Analysis of Cloud Computing Literature" by German scientists ranked Dr. Buyya as the World's Top-Cited (#1) Author and the World's Most-Productive (#1) Author in Cloud Computing. Dr. Buyya is recognized as a "Web of Science Highly Cited Researcher" in 2016 and 2017 by Thomson Reuters, a Fellow of IEEE, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing.

Software technologies for Grid and Cloud computing developed under Dr. Buyya's leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world. Manjrasoft's Aneka Cloud technology developed under his leadership has received "2010 Frost & Sullivan New Product Innovation Award". He served as the founding Editor-in-Chief of the IEEE Transactions on Cloud Computing. He is currently serving as Co-Editor-in-Chief of Journal of Software: Practice and Experience, which was established over 45 years ago. For further information on Dr. Buyya, please visit his cyberhome: www.buyya.com.