Formal Verification of the xDAuth Protocol

Quratulain Alam, Saher Tabbasum, Saif U. R. Malik, *Member, IEEE*, Masoom Alam, Tamleek Ali, Adnan Akhunzada, Samee U. Khan, *Senior Member, IEEE*, Athanasios V. Vasilakos, *Senior Member, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*— Service-oriented architecture offers a flexible paradigm for information flow among collaborating organizations. As information moves out of an organization boundary, various security concerns may arise, such as confidentiality, integrity, and authenticity that needs to be addressed. Moreover, verifying the correctness of the communication protocol is also an important factor. This paper focuses on the formal verification of the xDAuth protocol, which is one of the prominent protocols for identity management in cross domain scenarios. We have modeled the information flow of xDAuth protocol using high-level Petri nets to understand the protocol information flow in a distributed environment. We analyze the rules of information flow using Z language, while Z3 SMT solver is used for the verification of the model. Our formal analysis and verification results reveal the fact that the protocol fulfills its intended purpose and provides the security for the defined protocol specific properties, e.g., secure secret key authentication, and Chinese wall security policy and secrecy specific properties, e.g., confidentiality, integrity, and authenticity.**

***Index Terms*— Cross domain access control framework, formal methods, high-level Petri nets (HLPN), information security, modeling, and verification, SMT, service oriented architecture (SOA), xDAuth protocol, Z3.**

I. Introduction

THE internet has revolutionized the ways of communication. Before the advent of Service Oriented Architecture (SOA), it was not easy to connect organizations and to achieve interoperability among organizations having heterogeneous environments. The SOA provides the facility in which a function of an application can be called as a service by another application using a particular communication protocol. Moreover, the SOA has benefited both the private and the public sector organizations as it has enabled them to share information within and outside the organizational boundaries [1]. However, as information flows out of an organization boundary, it has its own security implications and necessary measures are required to make the communication secure [2]–[4]. Some of the major challenges of modern-day information sharing among different organizations are: handling cross domain resource sharing among distinct organizations over HTTP or HTTPS, making a wise decision about which information must be allowed for access by which user, dealing with cross domain access control policies, providing strong privacy protection to user data and the issue of building trust on services offered by unseen organizations [4]–[6].

Different standards and protocols have been developed to address the security concerns mentioned above. Some of the well-known standards and protocols are Shibboleth [7], OAuth 2.0 [8], and OpenID [9]. All of these standards provide users the facility to Sign-In on a system or a website using a single identity, such as Single Sign On (SSO). This can be used in a distributed system such as a federation of different organizations. The federation can be of universities, public or private sector organizations. The federation with respect to Identity Management Systems (IMS) is a joint venture of two or more trusted organizations. Such organizations are bound together with some business or technical contract in which users from either side can access restricted resources [10]. Although, the identity management frameworks and standards provide the capability of SSO, the frameworks are limited on providing certain features of cross domain security such as Conflict of Interest (CoI). Such standards focus on the format in which information must be written. The specifications level security at both collaborating bodies is still a challenge. As an alternate solution to these standards, an access control framework-xDAuth was presented in [11]. The xDAuth protocol overcomes two of the major issues in the SOA and access control, namely Privacy and Trust. The privacy is handled at two levels: one is at the service requester end by hiding user attributes from the service provider and user authentication credentials from the delegation service. The other level is achieved at the service provider end in which delegation service has no prior knowledge of the published resources and service provider uses pseudonyms for the published resources on the delegation service. Trust is achieved by exposing

Manuscript received December 17, 2015; revised March 28, 2016; accepted April 17, 2016. Date of publication May 3, 2016; date of current version June 27, 2016. This work was supported by the National ICT Research and Development Fund under the Project CDACDEA (http://www.ictrdf.org.pk/index.php/component/tprojects/project/19). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Wanlei Zhou.

Q. Alam and T. Ali are with the Department of Computer Sciences, Institute of Management Sciences, Peshawar 25000, Pakistan (e-mail: quratul.alam30@gmail.com; tamleek091@gmail.com).

S. Tabbasum, S. U. R. Malik, and M. Alam are with the Department of Computer Sciences, COMSATS Institute of Information Technology, Islamabad 45550, Pakistan (e-mail: saher.tabbasum@comsats.edu.pk; saif.rehmanmalik@gmail.com; masoom.alam@gmail.com).

A. Akhunzada is with the Center for Mobile Cloud Computing Research, University of Malaya, Kuala Lumpur 50603, Malaysia (e-mail: a.adnan@siswa.um.edu.my).

S. U. Khan is with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58108-6050 USA (e-mail: same.khan@ndsu.edu).

A. V. Vasilakos is with the Department of Computer Science, Electrical and Space Engineering, Lule University of Technology, Skellefteå SE-931 87, Sweden (e-mail: th.vasilakos@gmail.com).

R. Buyya is with the Cloud Computing and Distributed Systems Laboratory, Department of Computing and Information Systems, The University of Melbourne, Parkville, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2016.2561909

the unique secret key provided by the delegation service (mediating service) to each service requester and service provider involved in the cross domain resource sharing environment. Detailed information on the xDAuth protocol is provided in Section II of the paper.

Security standards and protocols of federations are vulnerable to network attacks, as advocated in [5] and [12]–[14]. Therefore, an effort must be made to analyze the correctness of the protocols, so as to avoid network attacks. The improvements should be formally verified to ensure that the protocols are free of security problems. Formal methods are applied to check the correctness of the protocol. Formal methods have been found to be very useful in the verification of protocols. Formal methods provide techniques and tools for specifying the system and checking the security properties for system verification [15]–[17], [54]. Some cryptographic protocols such as Needham-Schroeder, TMN, Kerberos and a formal foundation for web security have been successfully verified using Formal methods approaches, which exposed the security flaws [18], [19].

xDAuth is considered as one of the prominent protocol for SOA and cross domain access control scenarios. Therefore, verification of the correctness of xDAuth protocol is imperative to avoid network attacks. The correctness of the xDAuth protocol ensures that the protocol is free of security problems. To the best of our knowledge, this is the first effort made on modeling, analysis and formal verification of the xDAuth protocol. We preferred model checking than simulation, testing and deductive verification due to the reason that it consists of an exhaustive exploration of the system and can be used to verify finite state concurrent systems, which benefit as automatic verification. We have used High Level Petri Nets (HLPN) and Z language for modeling and analysis of the cross domain access control protocol xDAuth [11]. The HLPN presents a graphical representation of the system and provides the mathematical representation to analyze the behavioral and structural properties of the system. With the help of these two techniques, we are able to understand and analyze the interconnections between the system entities, granular details, and processing of information. Verification of the model is done by translating the HLPN model using bounded model checking techniques through Satisfiability Modulo Theories Library (SMT-Lib) and Z3 solver. The security properties are also translated into the SMT and checked whether the model satisfies the security properties. As the focus of this paper is proving the correctness of xDAuth protocol flow, in the said perspective, we defined certain security properties and verified that the security properties must always be satisfied throughout the system. For each of the system's behavior, such as the sequence of inputs, outputs, and state changes, we clearly determine that whether a desired security property holds or not. The proposed method can be applied for the evaluation or verification of other security protocols. We have applied the similar method for the formal analysis and verification of FADE protocol, which is used for the security of files stored on the cloud (complete details can be seen in [55]). Moreover, some of the literature work that depicts the application of the method that we adopt to verify our protocol can be seen in [56] and [57]. The method can also be applied to other kinds of security protocols, such as OAuth and Shibboleth for the verification. The main contributions of the paper are listed below.

- Detail modeling and analysis of cross domain access control protocol, i.e. xDAuth protocol with the help of HLPN and Z language respectively.
- Formal verification of cross domain access control protocol xDAuth on defined system specifications and security properties, using SMT-Lib and Z3 Solver.
- Proving the correctness of the xDAuth protocol by defining certain security properties and verifying that the security properties must always be satisfied throughout the system.
- We perform the automated verification of the model by following the bounded model checking technique using Satisfiability Modulo Theories Library (SMT-Lib) and Z3 solver. During the analysis of xDAuth, we realized that the formal verification of the security properties, such as confidentiality, integrity and authenticity, is not possible without incorporating certain attributes as part of xDAuth protocol. Therefore, we augmented the attributes in xDAuth to verify the security properties. This formal verification technique made us able to augment certain attributes in the xDAuth protocol which are necessary for the verification of the security properties.
- The delegation service (DS) returns a domain key (30-byte public string to uniquely identify the domain) and the secret key (10-byte shared secret between the DS and domain) after the registration process. Our analysis has revealed that the said keys can create a single point of failure, which will lead to the compromise of the whole federation.

The rest of the paper is organized as follows. Section II presents the background concepts such as xDAuth Protocol, Boolean Satisfiability Solvers, and Petri Nets. Section III elaborates related work. Section IV describes the Modeling and Analysis of the xDAuth protocol. Section V constitutes the verification of the xDAuth protocol. Finally, we provide the concluding remarks in Section VI.

II. Background

A. An Overview of the xDAuth Protocol

The xDAuth protocol presents a general framework for the realization of cross domain access and delegation control of the resources. In the xDAuth, there is a single trusted Delegation Service (DS) that acts as the decision making body for cross domain resource access request. The Service Provider (SP) is the organization that agrees to share its resources for cross domain resource sharing. The Service Requester (SR) is the organization from which users access the resources. The xDAuth provides strong privacy protection to both the SP and the SR domains. The SR domain user's authentication credentials are kept private from the DS. Similarly, the SP can define policies to hide the information of shared resources from the DS. The xDAuth can support many

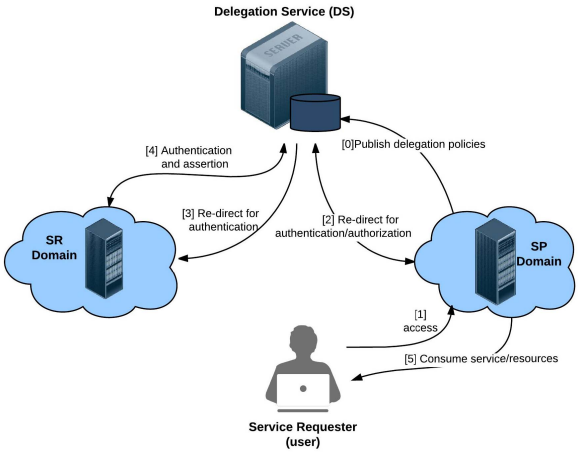

Fig. 1. An Overview of the xDAuth Protocol [11].

access control and delegation constraints in cross domain environment, such as Dynamic Separation of Duty (DSoD) [20] and the Chinese Wall Security Policy (CWSP) [21]. A pictorial representation of the xDAuth protocol is depicted in Figure 1.

1) Responsibilities of the xDAuth Entities: The xDAuth is implemented in OpenERP by the "Cross Domain Access Control and Delegation in Enterprise Applications (CDACDEA)" project [22]. Three main entities constitute the xDAuth protocol: the service provider (SP), the service requester (SR), and the delegation service (DS). However, both the SP and the SR are implemented in the OpenERP; whereas, the DS is implemented as a web service. Responsibilities of each of the entity are as follows:

- The SP Responsibilities: The SP is responsible to publish the cross domain policies on the DS. The SP receives resource access requests from the SR domain user and redirects the user's client application to the DS for further procedure.
- The DS Responsibilities: The DS receives resource access request redirected from the SP. Once the request is received the DS performs two main operations: (a) Managing request redirections for authentication and authorization, and (b) Implementing the CWSP.
- The SR Responsibilities: The DS redirects the access requests to the SR domain for authentication. Once the access request is received, the responsibility of the SR is to authenticate the user in the home domain. As a response to the DS request, the SR domain sends the required attributes, user authentication response (valid or invalid), and domain authentication response.

2) Accessing Cross Domain Resources via the xDAuth Protocol: The overall process of accessing a shared resource within the xDAuth protocol is divided into the following steps:

Forming a federation is considered to be a prerequisite of step 0 i.e. Publishing Policies. A federation is formed by the registration of SP and SR domains at the DS to become part of the federation. Once the federation is formed, a local user (from the SP domain) makes a delegation request to the Internal Authorization (IA) service of the SP for creating cross domain policy.

- Step 0: Publishing Policies In step 0, After the creation of cross domain policy, the SP publishes the policy on the DS.
- Step 1: Resource Request The first step towards accessing a shared resource is initiated by an external user (from the SR domain). The external user initiates a request for resource access that is available on the SP resource page through a client application, such as a web browser.
- Step 2: Request Redirection to the DS The SP domain receives a resource access request from the SR user. Since SP is not managing the identity of external users; therefore, the request is redirected to the DS for authentication and authorization decisions.
- Step 3: SR Home Domain Selection The DS receives the redirected resource access request from the SP. In order to get the user authenticated from her parent organization, the DS provides a list of domains in the federation to the SR user, for selecting the home domain.
- Step 4: SR Authentication Once redirected to the SR domain, the user authenticates herself by providing already issued credentials. After successful authentication, the user is redirected to the DS along with the domain identity and required attributes (which were requested by the DS).The reason for requesting the attributes is because the DS has to make an authorization decision for the SP domain based on the cross domain access control policy.
- Step 5: DS Redirection to SP On receiving the required attributes, the DS evaluates them against the published policy. A positive evaluation takes the user's client application to the SP along with the authorization result.
- Step 6: Resource Release After receiving the authorization result from the DS, the SP provides access to the resource to the user. A fine-grained description of each step along with the security constraints will be discussed in Section IV.

B. SMT-Lib and Z3 SMT Solver

In the context of automated reasoning and formal verification, Boolean Satisfiability Solvers (SAT) are used. However, now the decision problems are encoded and solved as Satisfiability Modulo Theories (SMT) [23]. The SAT are propositional satisfiability solvers. The SMT takes the decidability problem as first order logic formula and decide for its satisfiability based on the decidable background theory. There are a number of theories supported by the SMT solvers, such as equality and un-interpreted functions, linear arithmetic over rationals, linear arithmetic over integers, non-linear arithmetic over reals, over arrays, bit vectors, and combinations [24]. The SMT-Lib provides a common input platform for many solvers used for the verification of systems. Behavioral specifications of a system can be represented using abstract models. The SMT solvers are then used to perform bounded model checking to explore a bounded symbolic execution of the model [25], [26]. A number of solvers are available that support the SMT-Lib such as the Beaver, the Boolector, the CVC4, the MathSAT5, the Z3, and the OpenSMT [25].

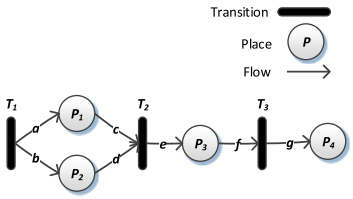

Fig. 2. An Example of the HLPN.

The differentiating feature of solvers can be the underlying logic (First Order Logic (FOL) or Temporal Logic), supported theories, input formulas, and interfaces [24], [25].

In this paper, we used the Z3 constraint solver, which is an efficient automated SMT solver by Microsoft Research Labs [27]. The Z3 solver is mostly used in the analysis and verification of software systems. The underlying verification theory for our system's model is the theory of array that is used to prove the satisfiability of our model's logical formulae. The array theory is frequently used in software modeling domain [28].

C. The High-Level Petri Nets (HLPN)

Petri Nets are very useful for mathematical and graphical modeling of wide range of systems, such as distributed, parallel, concurrent, stochastic, non-deterministic, and asynchronous systems [29]. However, a tradeoff must be kept between modeling generality and analysis capability. Even a modest model can become too large for analysis process. In this work, we have used a variant of the conventional Petri Nets, termed as High-Level Petri Nets (HLPN) for the formal verification of the proposed migration technique. The HLPN is a set of 7-tuple [30], [31], $N = (P, T, F, \varphi, R, L, M_0)$, where:

1) P is a set of finite places,
2) T is a set of finite transitions such that P and T are two distinct sets $P \cap T = \phi$,
3) F represents the flow relation from place to transition or transition to place such that $F \subseteq (P \times T) \cup (T \times P)$,
4) ϕ represents the mapping function that maps places to data types, such that ϕ: P $\rightarrow$ Data types,
5) R defines the set of rules that maps T to logical formulae such that R: T $\rightarrow$ Formula,
6) L represents the labels that are mapped on each flow in F, such that L: F $\rightarrow$ Label,
7) M_0 represents the initial state/marking where flow can be initiated, such that M: P $\rightarrow$ Tokens.

The first three variables *(P, T, F)* provide information about the structure of the Petri Net. The next three variables *(ϕ, R, L)* provide the static semantics of the Petri Net, which means that the information does not change throughout the system. To build an understanding of a Petri Net, we demonstrate a small example. Figure 2 represents a simple Petri Net, having 4 places *($P = P_1, P_2, P_3, P_4$)*, 3 transitions *($T = T_1, T_2, T_3$)*, and 7 flows *($F = a, b, c, d, e, f, g$)*. In HLPN, each place has some tokens to enable adjacent transitions, which means that the preconditions must hold for the transition to fire. The tokens can correspond to one type or a cross product of different data types. In Table I, we have mapped places to the following data types.

TABLE I
PLACES TO DATA TYPE MAPPING

Places	Mapping	Description
$\varphi(P_1)$	$\wp$ (int, bool)	P_1 holds variables of type int and bool
$\varphi(P_2)$	$\wp$ (char)	P_2 holds variable of type char only
$\varphi(P_3)$	$\wp$ (string, int, bool)	P_3 holds variables of type string, int and bool
$\varphi(P_4)$	$\wp$ (bool)	P_4 holds variables of type bool only

Let α and β be the nodes of the HLPN N if and only if $\alpha, \beta \in P \cup T$. A node α is an input node of another node β if and only if there is a directed arc from α to β such that $(\alpha, \beta) \in F$. Node α is an output node of β if and only if $(\beta, \alpha) \in F$. The precondition is $\bullet$ $P_1 = (P_2 \mid (P_2, P_1) \in F)$ and post condition is $\bullet$ $P_1 = (P_2 \mid (P_1, P_2) \in F)$. The precondition must hold to enable the transition. For example in the Figure 2, precondition for T_2 will use c and d as input. Similarly, post-condition will take values from outgoing flow to enable further transitions.

III. RELATED WORK

The concerns such as privacy, security, authorization, and authentication in the cross domain access control applications calls for a mechanism that can be used to ensure correctness, reliability, robustness, and safety of the system. The formal methods, which involves formal modeling, analysis, and verification technique, is one such mechanism that can be used to achieve the aforesaid attributes by introducing rigor, performing proofs, and the verification of the underlying systems [23], [25], [32]. The formal analysis of the security and the access control protocols has been addressed by many researchers [33], [36], [41], [48]. The goal of using the formal analysis and the verification techniques is to provide a formal base for the protocol verification that helps in finding flaws and improving the design. One such analysis is performed in [33], where a symbolic safety analysis framework for the Administrative Role Based Access Control (ARBAC) policy model is presented. The authors of [33] used the SMT for the symbolic backward reachability procedure that can be used to solve the user-role reachability problem. However, the work done in [33] is not focused on the incremental analysis of the access control policies and does not provide automated analysis and proof generation in the case of policy resistance. The authors in [34] performed the automated analysis of ARBAC model.

The work presented in [34] shows an abstract rule-based framework for the integration of the attributes in the RBAC model. However, the analysis performed in the aforesaid paper was only focused on the centralized policies of the RBAC model. Research work done in [35] has considered the administration of multiple attributes including the atomic attributes. A systematic root cause analysis to study the security vulnerabilities of the OAuth 2.0 protocol is presented in [36]. The authors used an attacker model to unveil the vulnerabilities such as the replay attacks, the network eavesdropping, the forced-login CSRF attacks, and the impersonation attacks in the OAuth 2.0 protocol. An abstraction technique for an

automated error finding in the ARBAC polices is presented in [37]. The proposed technique is evaluated in an access control policy verification tool MOHAWK. The results obtained are scalable with respect to complexity of the policies and also in the magnitude. Similarly, a methodology for the bounded analysis of the security and the cryptographic protocols using a model checking tool, named the Mur-phi, is presented in [18]. The authors of [18], analyzed the Needham-Schroeder protocol, the Kerberos, and the TMN protocol to identify known bugs. However, the tool (Mur-phi) is rarely used for analyzing network protocols where the number of topologies are large. In another study [38], the analysis of the Needham-Schroeder protocol is performed using the Failures Divergences Refinement (FDR) model checker to test whether the protocol correctly achieves the authentication and discovered an attack on the protocol. In [39], the authors have given a protocol formalization, first in the Alice-Bob notation, and then in the High Level Protocol Specification Language (HLPSL). Moreover, the authors has used the AVISPA, the state-of-the-art verification tool for the security protocols to identify two security flaws, a replay attack and a masquerade attack. However, the formal specification of the Facebook Connect protocol was not discussed in detail. Therefore, the model was inferred only by observing the message exchange process during a valid protocol run. In another study [19], the authors worked on the formal modeling of the web security and the authentication protocols using the Alloy model checker.

In order to identify some known and unknown vulnerabilities of the login CSRF attack, the authors of [19] have modeled the web-based authentication protocol the *WebAuth*, which is a Kerberos based SSO web solution. However, the approach followed by the authors is limited to the HTTP communication only and does not take into account the application layer messages such as the SOAP, as discussed in [40]. The formal model presented by [41] is a good set of advice to the secure web service communication. A symbolic protocol model of the Needham Schroeder protocol is generated from the C implementation code in [16], using the Csur tool (a project about the automatic analysis of the cryptographic protocols). Only the secrecy properties were considered for the protocol modeling. However, the authors did not provide any verification results. Apart from the model checkers and the theorem provers, the researchers also developed the logics of knowledge, the logics of belief, such as the BAN, and improved them to provide more formal ways for analyzing the security and the authentication protocols [42].

An abstract representation of the cryptographic protocols using the prolog rules is presented in [15]. Such prolog rules are used for proving the secrecy properties of the protocols using the ProVerif [15] tool. The main difficulty faced while using such methods is the termination of the analysis process. Moreover, such methods do not address the decidability of the secrecy with the Cipher Block Chaining (CBC) encryption or the blind signature. An automated unbounded symbolic verification and analysis tool the Scyther is presented in [43]. The scyther tool is used for the verification of the security and the cryptographic protocols. Such tool has the ability to automatically find attacks in a protocol. Other tools for the same purpose are the AVISPA [44], the ProVerif [15], and the TAMARIN [45].

A role-based provenance mechanism for services aggregation and trustworthiness is presented in [46]. However, the association of roles from different domains are handled solely by manual process that can be tedious and lack agility. Also, assuring access control needs to be considered as one significant part of security requirement in service oriented architecture.

Zhao *et al.* [47] discussed about constructing authentication web in cloud computing which is also a promising direction for cross domain resource sharing. The focus was to propose a single sign-on mechanism to accept user identities from various domains. They also made a formal mathematical model to analyze security issues of the proposed mechanism and verified it to support hybrid authentication protocols. However, the proposed mechanism gave authority to service providers to verify the user identities and took authorization decisions for releasing resources in a cross domain resource sharing environment.

In [48], the Alloy modeling language is presented to cast the resource accessibility problem as a constrained graph reachability problem. Also, an Alloy model is created for each domain policy, and these models are then composed in a sound and efficient manner. However, [48] has discussed only the access control policiy'aspect of cross domain paradigm.

Our work is different in the context that along with the formal analysis of the xDAuth protocol we have modeled the protocol in the HLPN and verified the model within the Z3 [27], that is an efficient SMT solver. Using the Z3, we performed bounded verification of the xDAuth protocol. The reason for selecting the Z3 SMT solver for the verification of the xDAuth protocol is that Z3 is currently an area of the research for many researcher for the software verification. The Z3 is a powerful solver used for solving the real world problems that require greater computational power and the complex constraint solving abilities. Moreover, its ability to support several FOL theories brings in it more flexibility when reasoning about the security polices and their properties. Other state-of-the art the SMT solvers are the BarceLogic, the CVC4, the MathSAT, and the Yices [25]. However, the Z3 solver(and many others automated provers) use classical logics, the proof terms produced in this manner are not constructive in nature. Moreover, the Z3 does not have any precise decision procedures for the string [49] and the floating point arithmetic [50]. In the coming section, we will model the xDAuth protocol using the HLPN and define all the granular level details. We will also define the rules that contain conditions on which transition will be enabled/disabled.

IV. Modeling and Analysis of xDAuth Protocol

A HLPN model for initiating a cross domain resource access request through the xDAuth protocol is demonstrated in Figure 3.

As stated in the formal definition of the Petri Net, presented in the background section, the HLPN is a 7-tuple $N = (P, T, F, \varphi, R, L, M_0)$. The first step to model a system

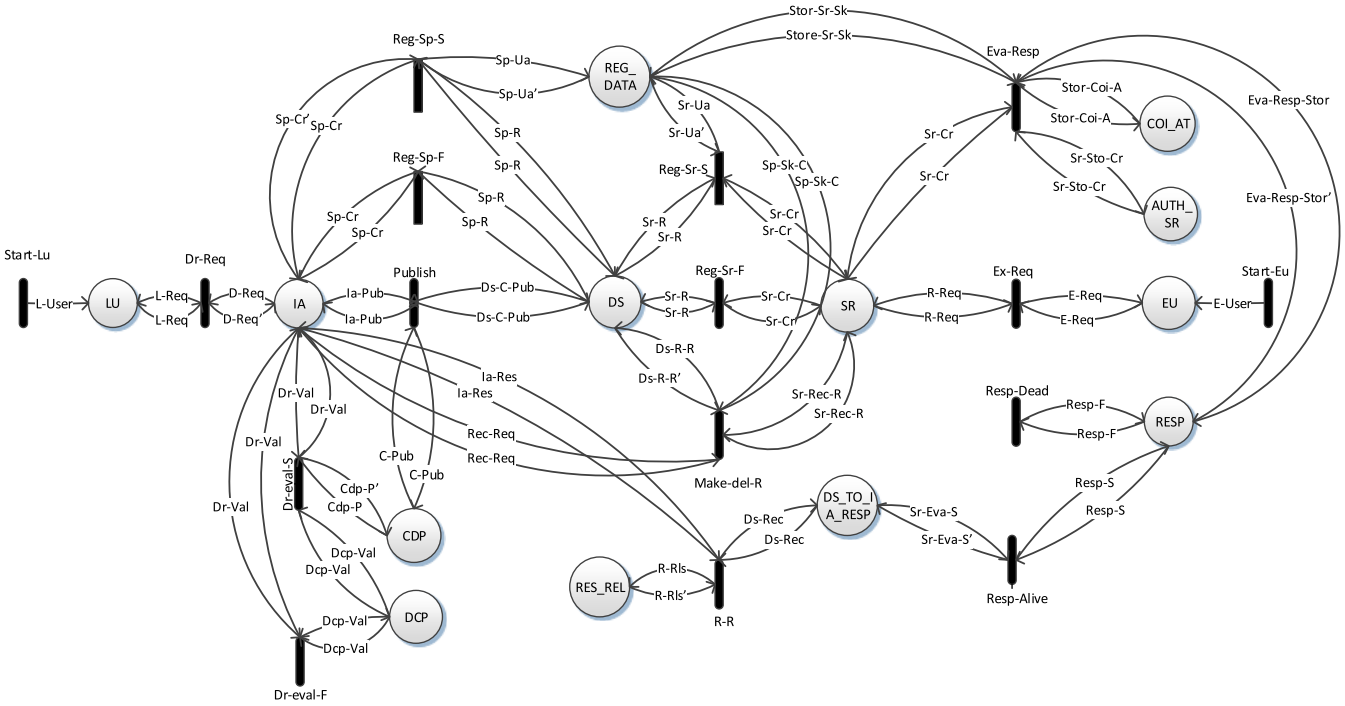

Fig. 3. A HLPN Model for the Resource Access Request in the xDAuth Protocol

in the HLPN is to define a set of P (Places) and the associated data types. As depicted in Figure 3, there are 13 places in the model. The names and the mapping of P is shown in Table II. The data types used in the model are shown in Table III.

The steps for initiating a cross domain resource access request are discussed in the background section. In this section, we will model and analyze the granular level details of the protocol. Moreover, we will define the set of rules, the preconditions, and the post-conditions to map to the T. The set of the transitions T is : T = *{Start-Lu, Dr-Req, Reg-Sp-S, Reg-Sp-F, Publish, Dr-eval-S, Dr-eval-F, Start-Eu, Reg-Sr-S, Reg-Sr-F, Make-del-R, Eva-Resp, Ex-Req, Resp-Dead, Resp-Alive, R-R}*.

A. Modeling and Analyzing Step 0 of the Protocol

The Step 0 of the protocol has two sub-steps:

- The Domain Registration, and
- The Policy Publication.

In the following subsection, we will elaborate the aforesaid sub-steps in detail.

The Domain Registration: The organization that needs to be a part of a federation must register the domain on the DS. In our case, we have two organizations, the SP and the SR. An organization registers on the DS through a client application, such as a web browser. On successful domain registration, the DS stores the domain information. A portion of the HLPN model shown in Figure 3 is depicted in Figure 4, which represents the registration process of domains on the DS.

The registration outcome of domains can be successful or failed. The transition *Reg-Sp-S* depicts the successful registration of the SP shown as IA place, where the DS assigns a unique domain-ID and secret-key to the newly registered domain and stores the record at the REG-DATA for later

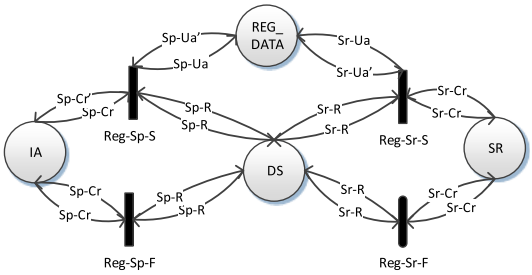

Fig. 4. Model for the Domain Registration.

usage, as shown in (1). Moreover, in (1), the transition registers the domain as the SP, based on the domain administrator's credentials and the domain type, that is done by matching the credentials stored at the place REG-DATA with the one provided by the administrator.

$$
\begin{aligned}
\boldsymbol{R}\ (\textit{Reg-Sp-S}) \ =\ & \forall \textit{sp-cr} \in Sp-Cr, \quad \forall \textit{sp-r} \in Sp-R, \\
& \forall \textit{sp-ua} \in Sp-Ua \mid \textit{sp-cr}[17] \\
& \quad = \textit{sp-r}[6] \vee \textit{sp-cr}[10] \neq \textit{sp-r}[1] \\
\rightarrow\ & \textit{sp-r}[1] := \textit{sp-cr}[10], \quad \textit{sp-r}[2] := \textit{sp-cr}[11], \\
& \textit{sp-r}[3] := \textit{sp-cr}[12], \quad \textit{sp-r}[4] := \textit{sp-cr}[13], \\
& \textit{sp-r}[10] := \textit{sp-cr}[14], \quad \textit{sp-r}[13] := \textit{sp-cr}[9], \\
& \textit{sp-r}[25] := \textit{sp-cr}[24], \\
& \textit{sp-r}[11] := CR-REG-DID(\textit{sp-cr}[17], \textit{sp-cr}[10]), \\
& \textit{sp-r}[12] := CR-REG-SK(\textit{sp-cr}[17], \textit{sp-cr}[10]), \\
& \textit{sp-cr}[15] := \textit{sp-r}[11], \quad \textit{sp-cr}[16] := \textit{sp-r}[12], \\
& \textit{sp-ua}[1] := \textit{sp-r}[11], \quad \textit{sp-ua}[2] := \textit{sp-r}[12], \\
& Sp-Cr' \ =\ Sp-Cr \cup (\textit{sp-cr}[15], \textit{sp-cr}[16]) \\
\wedge\ & Sp-R' \ =\ Sp-R \cup (\textit{sp-r}[1], \textit{sp-r}[2], \textit{sp-r}[3], \textit{sp-r}[4], \\
& \textit{sp-r}[10], \textit{sp-r}[13], \textit{sp-r}[11], \textit{sp-r}[12], \textit{sp-r}[25]) \\
\wedge\ & Sp-Ua' \ =\ Sp-Ua \cup (\textit{sp-ua}[1], \textit{sp-ua}[2])
\end{aligned} \tag{1}
$$

TABLE II

PLACES AND MAPPING OF THE xDAUTH PROTOCOL

Places	Mapping	Description
$\varphi(LU)$	℘(lu-subject-delegator-id×lu-subject-delegator-name×lu-subject-delegator-email×lu-resource-name×lu-action×lu-subject-delegatee-email×lu-delegatee-domain-name×lu-designation-delegatee)	Holds local user attributes
$\varphi(IA)$	℘(ia-subject-delegator-id×ia-subject-delegator-name×ia-subject-delegator-email×ia-resource-name×ia-action×ia-subject-delegatee-email×ia-delegatee-domain-name×ia-designation-delegatee×ia-delegator-domain-name×ia-delegator-domain-type×ia-delegator-domain-description×ia-delegator-domain-resurl×ia-delegator-domain-weburl×ia-cof×ia-domain-key×ia-secret-key×ia-admin-email×ia-admin-pass×ia-cdp-id×ia-id×ia-sp-auth-result×ia-request-token×ia-mess-cont×ia-timestamp)	Stores Service provider attributes
$\varphi(DCP)$	℘(dcp-subject-delegator-email×dcp-resource-name×dcp-action×dcp-subject-delegatee-email×dcp-delegatee-domain-name×dcp-designation-delegatee)	Holds delegation control policy attributes
$\varphi(CDP)$	℘(cdp-subject-delegator-email×cdp-resource-name×cdp-action×cdp-subject-delegatee-email×cdp-delegatee-domain-name×cdp-designation-delegatee)	Captures cross domain policy attributes
$\varphi(DS)$	℘(ds-domain-type×ds-domain-description×ds-delegator-domain-resurl×ds-delegator-domain-weburl×ds-delegatee-domain-authurl×dc-delegator-admin-email×ds-delegator-admin-pass×ds-delegatee-admin-email×ds-delegatee-admin-pass×ds-coi×ds-domain-key×ds-secret-key×ds-delegator-domain-name×ds-delegatee-domain-name×ds-subject-delegator-email×ds-subject-delegatee-email×ds-resource-name×ds-action×ds-designation-delegatee×ds-coi-history×ds-sr-auth-results×ds-sp-auth-results×ds-request-token×ds-cdp-id×ds-timestamp)	Stores Delegation service attributes
$\varphi(EU)$	℘(sr-subject-delegatee-name×sr-subject-delegatee-email×sr-designation-delegatee×sr-resource-name×sr-action×sr-subject-delegator-email×sr-delegatee-domain-name×sr-delegator-domain-name)	Holds external user attributes
$\varphi(SR)$	℘(sr-subject-delegatee-name×sr-subject-delegatee-email×sr-designation-delegatee×sr-resource-name×sr-action×sr-subject-delegator-email×sr-delegatee-domain-name×sr-delegatee-domain-type×sr-delegatee-domain-authurl×sr-delegatee-domain-weburl×sr-delegatee-domain-description×sr-admin-email×sr-admin-pass ×sr-subject-delegatee-pass×sr-delegator-domain-name×sr-cof×sr-auth-results×sr-domain-key×sr-secret-key×sr-username-ent×sr-pass-ent×sr-mess-cont×sr-request-token×sr-timestamp)	Holds Service requester attributes
$\varphi(REG-DATA)$	℘(rd-domain-key×rd-secret-key×nu)	
$\varphi(COI-AT)$	℘(coiat-delegatee-email×coiat-resource-name×coiat-action×coiat-delegator-domain-name×coiat-coi)	Maintains a log against resource requests
$\varphi(AUTH-SR)$	℘(sr-uname×sr-pass)	Holds service requester authentication credentials
$\varphi(RESP)$	℘(auth-res-sr×coi-r)	Stores response of attributes evaluation for resource release
$\varphi(DS-TO-IA-RESP)$	℘(cr-sp-rqt-r×sr-req-token)	Holds response for resource release when get a success after attribute evaluation
$\varphi(RES-REL)$	℘(resrelese-resource-name×resrelease-action×resrelease-subject-delegatee-email×resrelease-delegator-domain-resurl×resrelese-mess-cont)	Hold value for resource release

The rule for the domain registration failure is depicted in (4). Similar constraints are applied to the SR registration as were applied to the SP domain, where the administrator can only register one domain with the same email ID and the domain type. If the constraints are not followed, then the registration should not be successful and the process should terminate.

TABLE III

DATA TYPES USED IN THE xDAUTH PROTOCOL

Data types	Description
subject-delegator-id	An integer type for the delegator identification
subject-delegator-name	A string type for the delegator name
subject-delegator-email	A string type for the delegator email
resource-name	A string type for a resource name
action	A Boolean type for action to perform
subject-delegatee-email	A string type for the delegatee email
delegate-domain-name	A string type for the delegatee domain name
designation-delegatee	A string type for the delegatee designation
delegator-domain-name	A string type for the delegator domain name
delegator-domain-type	A string type for the delegator domain type
delegator-domain-description	A string type for the delegator domain description
delegator-domain-resurl	A URL type for the delegator domain resource URL
delegator-domain-weburl	A URL type for the delegator domain website
coi	A string type category for both the delegator and the delegatee
domain-key	A unique integer value for the domain identification
secret-key	An alpha-numeric value for a unique identification of the delegator and the delegate
admin-email	A string type for the administrator email
admin-pass	A string type for the administrator password
cdp-id	An integer type unique identification value for the cross domain policy
ia-id	An integer type unique identification value for the delegator domain administrator
sp-auth-result	A Boolean type authentication response
request-token	An integer type request token for the resource request
mess-cont	A string type message content
delegatee-domain-authurl	A URL type delegatee domain authentication URL
coi-history	A string type value for the COI history
sr-auth-results	A Boolean type response from the SR
sp-auth-results	A Boolean type response for the SP
subject-delegatee-name	A string type name of the delegatee
delegatee-domain-type	A string type domain type of the delegatee
Delegatee-domain-weburl	A URL type for the delegatee domain website
Delegatee-domain-description	A string type domain description of the delegatee domain
Username-ent	A string type username entered by the SR user
Pass-ent	A string type password entered by the SR user
Auth-res-sr	A Boolean type the SR authentication response
Coi-r	A string type resource name release after all authentications
timestamp	An integer type attribute part of a log mechanism

$$\begin{aligned} \boldsymbol{R}\ (Reg\text{-}Sp\text{-}F) &= \forall sp\text{-}cr \in Sp - Cr, \quad \forall sp\text{-}r \in Sp - R \mid \\ sp\text{-}cr[17] &= sp\text{-}r[6] \wedge sp\text{-}cr[10] \\ &= sp\text{-}r[1] \rightarrow Sp - Cr' := Sp - Cr \end{aligned} \tag{2}$$

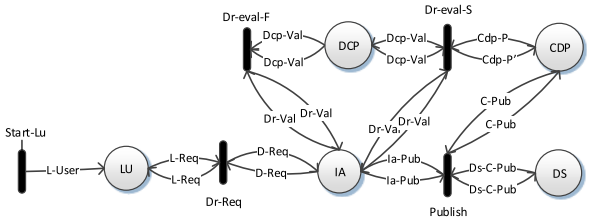

Fig. 5. Model for the Policy Publication.

Like the SP, the SR domain registration can be successful or failed. The transition *Reg-Sr-S* depicts the successful registration of the SR domain, where the DS assigns a unique domain-ID and secret-key to the newly registered domain and keeps the record at REG-DATA for later usage, as shown in (3). Moreover, in (3), the transition registers the domain as the SR, based on domain administrator's credentials and domain type. The aforesaid is done by matching the credentials stored at the place REG-DATA with the one provided by the administrator.

$$\begin{aligned}
\boldsymbol{R}\ (\textit{Reg-Sr-S}) &= \forall \textit{sr-cr} \in Sr - Cr, \quad \forall \textit{sr-r} \in Sr - R,\\
\forall \textit{sr-ua} &\in Sr - Ua \mid \textit{sr-cr}[12]\\
&= \textit{sr-r}[8] \vee \textit{sr-cr}[8] \neq \textit{sr-r}[1]\\
\rightarrow \textit{sr-r}[1] &:= \textit{sr-cr}[8], \quad \textit{sr-r}[2] := \textit{sr-cr}[11],\\
\textit{sr-r}[4] &:= \textit{sr-cr}[10], \quad \textit{sr-r}[5] := \textit{sr-cr}[9],\\
\textit{sr-r}[10] &:= \textit{sr-cr}[16], \ \textit{sr-r}[14] := \textit{sr-cr}[7],\\
\textit{sr-r}[25] &:= \textit{sr-cr}[24],\\
\textit{sr-r}[11] &:= CR - REG - DID(\textit{sr-cr}[12], \quad \textit{sp-cr}[8]),\\
\textit{sr-r}[12] &:= CR - REG - SK(\textit{sr-cr}[12], \quad \textit{sr-cr}[8]),\\
\textit{sr-cr}[18] &:= \textit{sr-r}[11], \textit{sr-cr}[19] := \textit{sr-r}[12],\\
\textit{sr-ua}[1] &:= \textit{sr-r}[11], \textit{sr-ua}[2] := \textit{sr-r}[12]\\
Sr - Cr' &= Sr - Cr \cup (\textit{sr-cr}[18], \textit{sr-cr}[19])\\
\wedge Sr - R' &= Sr - R \cup (\textit{sr-r}[1], \textit{sr-r}[2], \textit{sr-r}[4], \textit{sr-r}[5],\\
&\quad \textit{sr-r}[10], \textit{sr-r}[14], \textit{sr-r}[11], \textit{sr-r}[12], \textit{sr-r}[25])\\
\wedge Sr - Ua' &= Sr - Ua \cup (\textit{sr-ua}[1], \textit{sr-ua}[2]) \qquad (3)
\end{aligned}$$

The rule for the domain registration failure is depicted in (4). Similar constraints are applied to the SR registration as was applied to the SP domain, where the administrator can only register one domain with the same email ID and the domain type. If the constraints are not followed, then the registration will not be successful and the process will terminate.

$$\begin{aligned}
\boldsymbol{R}\ (\textit{Reg-Sr-F}) &= \forall \textit{sr-cr} \in Sr - Cr, \quad \forall \textit{sr-r} \in Sr - R \mid\\
\textit{sr-cr}[12] &= \textit{sr-r}[8] \wedge \textit{sr-cr}[8]\\
&= \textit{sr-r}[1] \rightarrow Sr - Cr' := Sr - Cr \qquad (4)
\end{aligned}$$

The Policy Publication On successful registration of the SP domain on the DS, the IA create and stores a cross domain policy in response to the delegation request made by the local user. A Petri Net model for the policy publication on the DS is shown in Figure 5.

The local user provides information through the transition *Start-Lu*, as shown in (5). There are no pre-conditions for the transition to be satisfied. All required values move to the place LU seamlessly.

$$R\ (\textit{Start-Lu}) = \exists \textit{l-user} \in L - User \mid \bullet \textit{l-user} = \phi \qquad (5)$$

The delegation request is performed on the transition *Dr-Req*, as shown in (6). The IA stores the information received in the request for the evaluation purpose.

$$\begin{aligned}
\boldsymbol{R}\ (\textit{Dr-Req}) &= \forall \textit{l-req} \in L - Req, \quad \forall \textit{dr-req} \in D - Req \mid\\
\textit{dr-req}[1] &:= \textit{l-req}[1], \quad \textit{dr-req}[2] := \textit{l-req}[2],\\
\textit{dr-req}[3] &:= \textit{l-req}[3], \quad \textit{dr-req}[4] := \textit{l-req}[4],\\
\textit{dr-req}[5] &:= \textit{l-req}[5], \ \textit{dr-req}[6] := \textit{l-req}[6],\\
\textit{dr-req}[7] &:= \textit{l-req}[7], \quad \textit{dr-req}[8] := \textit{l-req}[8],\\
D - Req' &= D - Req \cup (\textit{dr-req}[1], \textit{dr-req}[2], \textit{dr-req}[3],\\
&\quad \textit{dr-req}[4], \textit{dr-req}[5], \textit{dr-req}[6], \textit{dr-req}[7],\\
&\quad \textit{dr-req}[8]) \qquad (6)
\end{aligned}$$

The outcome of the evaluation of the delegation request can be successful or failed. The transition *Dr-eval-S* depicts the successful evaluation of the delegation request where the IA assigns a unique CDP-ID to the newly created cross domain policy and keeps the record at CDP for later use, as shown in (7). Moreover in (7), the evaluation is done by checking the values of the local user's email address, the resource name, the action, and the email address of the resource requester in organization's Delegation Control Policy (DCP). An additional constraint is also checked, where the ID of the request initiator and the request approver must not be the same.

$$\begin{aligned}
&\boldsymbol{R}\ (\textit{Dr-eval-S})\\
&= \forall \textit{dr-val} \in Dr - Val, \ \forall \textit{dcp-val} \in Dcp - Val,\\
\forall \textit{cdp-p} &\in Cdp - P \mid \textit{dr-val}[1] \neq \textit{dcp-val}[20]\\
\wedge \textit{dr-val}[3] &= \textit{dcp-val}[1], \quad \textit{dr-val}[4] = \textit{dcp-val}[2],\\
\textit{dr-val}[5] &= \textit{dcp-val}[3], \quad \textit{dr-val}[6] = \textit{dcp-val}[4]\\
\rightarrow \textit{cdp-p}[1] &:= \textit{dr-val}[3], \textit{cdp-p}[2] := \textit{dr-val}[4],\\
\textit{cdp-p}[3] &:= \textit{dr-val}[5], \ \textit{cdp-p}[4] := \textit{dr-val}[6],\\
\textit{cdp-p}[5] &:= \textit{dr-val}[7], \quad \textit{cdp-p}[6] := \textit{dr-val}[8],\\
\textit{cdp-p}[7] &:= CR - CDID(\textit{dr-val}[3], \textit{dr-val}[4],\\
&\quad \textit{dr-val}[6])\textit{dr-val}[19] := \textit{cdp-p}[7]\\
Dr - Val' &= Dr - Val \cup (\textit{dr-val}[19])\\
\wedge Cdp - P' &= Cdp - P \cup (\textit{cdp-p}[1], \textit{cdp-p}[2], \textit{cdp-p}[3],\\
&\quad \textit{cdp-p}[4], \textit{cdp-p}[5], \textit{cdp-p}[6], \textit{cdp-p}[7]) \qquad (7)
\end{aligned}$$

The rule for the delegation request failure is depicted in (8), where the failure occurs if: (a) The IDs of the local user and the administrator are the same, or (b) The delegated action corresponding to the requested resource is not defined in the DCP.

$$\begin{aligned}
\boldsymbol{R}\ (\textit{Dr-eval-F}) &= \forall \textit{dr-val} \in Dr - Val,\\
&\quad \forall \textit{dcp-val} \in Dcp - Val \mid\\
\textit{dr-val}[1] &= \textit{dr-val}[20] \vee \textit{dr-val}[3] \neq \textit{dcp-val}[1],\\
\textit{dr-val}[4] &\neq \textit{dcp-val}[2], \quad \textit{dr-val}[5] \neq \textit{dcp-val}[3],\\
\textit{dr-val}[6] &\neq \textit{dcp-val}[4] \rightarrow Dcp - Val' := Dcp - Val \qquad (8)
\end{aligned}$$

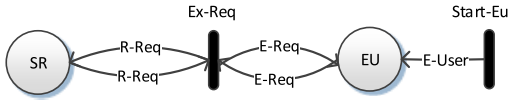

Fig. 6. Model for an External User.

Finally, after successful evaluation of the delegation request, the IA administrator publishes the newly created policy on the DS, as shown in (9). The administrator must authenticate himself with the DS before policy publication. This is done by matching the credentials stored at the place REG-DATA, with the one provided by the administrator. A distinct ID is also assigned to each published policy, which is then stored on both the SP and the DS.

$$
\begin{aligned}
\boldsymbol{R}\ (Publish) \ &= \ \forall c\text{-}pub \in C-Pub,\ \ \forall ia\text{-}pub \in Ia-Pub,\\
&\forall ds\text{-}c\text{-}pub \ \in \ Ds-C-Pub \mid ia\text{-}pub[17] = ds\text{-}c\text{-}pub[6]\\
\rightarrow\ & ds\text{-}c\text{-}pub[15] := c\text{-}pub[1],\quad ds\text{-}c\text{-}pub[17] := c\text{-}pub[2],\\
& ds\text{-}c\text{-}pub[18] := c\text{-}pub[3],\quad ds\text{-}c\text{-}pub[16] := c\text{-}pub[4],\\
& ds\text{-}c\text{-}pub[14] := c\text{-}pub[5],\quad ds\text{-}c\text{-}pub[19] := c\text{-}pub[6],\\
& ds\text{-}c\text{-}pub[13] := ia\text{-}pub[9],\quad ds\text{-}c\text{-}pub[24]\\
& \qquad := CR-CDP-ID(c\text{-}pub[2]),\\
& ia\text{-}pub[19] := ds\text{-}c\text{-}pub[24]\\
& Ia-Pub' \ = \ Ia-Pub \cup (ia\text{-}pub[19])\\
& \wedge\ Ds-C-Pub'\\
& \qquad = \ Ds-C-Pub \cup (ds\text{-}c\text{-}pub[15],\\
& \qquad\quad ds\text{-}c\text{-}pub[17], ds\text{-}c\text{-}pub[18], ds\text{-}c\text{-}pub[16],\\
& \qquad\quad ds\text{-}c\text{-}pub[14], ds\text{-}c\text{-}pub[19], ds\text{-}c\text{-}pub[13],\\
& \qquad\quad ds\text{-}c\text{-}pub[24]) \qquad (9)
\end{aligned}
$$

B. Modeling Step 1, 2, and 3 of the Protocol

The request for a cross domain resource acquisition in a federation is initiated by an external user (coming from the SR domain). The resource request flow starts on the transition *Start-Eu*, as shown in figure 6. The rule for the transition *Start-Eu* is shown in (10), which depicts that there are no pre-conditions for the transition and all the required values are provided by the user.

$$\boldsymbol{R}\ (Start\text{-}Eu) = \exists e\text{-}user \in E-User \mid \bullet e\text{-}user = \phi \qquad (10)$$

The resource access request reaches the SR from the transition *Ex-Req*, as shown in (11). The transition updates the SR by transferring the information from the place EU.

$$
\begin{aligned}
\boldsymbol{R}\ (Ex\text{-}Req) \ &= \ \forall e\text{-}req \in E-Req,\quad \forall ex\text{-}req \in Ex-Req \mid\\
& ex\text{-}req[1] := e\text{-}req[1],\quad ex\text{-}req[2] := e\text{-}req[2],\\
& ex\text{-}req[3] := e\text{-}req[3],\quad ex\text{-}req[4] := e\text{-}req[4],\\
& ex\text{-}req[5] := e\text{-}req[5],\quad ex\text{-}req[6] := e\text{-}req[6],\\
& ex\text{-}req[7] := e\text{-}req[7],\quad ex\text{-}req[8] := e\text{-}req[8],\\
& E-Req' \ = \ E-Req \cup (ex\text{-}req[1], ex\text{-}req[2], ex\text{-}req[3],\\
& \qquad ex\text{-}req[4], ex\text{-}req[5], ex\text{-}req[6], ex\text{-}req[7],\\
& \qquad ex\text{-}req[8]) \qquad (11)
\end{aligned}
$$

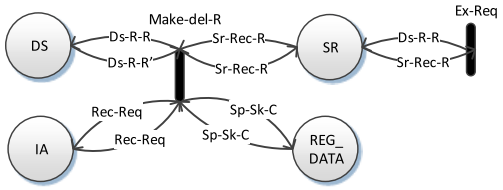

Fig. 7. Model for the Delegation Request.

On the transition *Make-del-R*, which is shown in Figure 7, the request is forwarded to the DS through a web interface which provides the details of all the shared resources of the SP. By selecting any of the listed resources, a request is initiated and forwarded to the DS for the authentication, and authorization decisions. A Petri Net model for resource access request is depicted in Figure 7 below. The rule for the *Make-del-R* transition is shown in (12).

As soon as the request reaches the DS, the DS first authenticates the SP domain. The authentication is performed by comparing the shared secret key provided to the SP at the time of registration. If the comparison results in a failure, then the resource request is denied by the DS. Alternatively, the DS creates a request token for the resource request and keeps it for further processing.

$$
\begin{aligned}
\boldsymbol{R}\ (Make\text{-}del\text{-}R) \ &= \ \forall sr\text{-}rec\text{-}req \in Sr-Rec-R,\\
&\forall rec\text{-}req \ \in \ Rec-Req,\quad \forall ds\text{-}rec\text{-}req \in Ds-R-R,\\
&\forall sp\text{-}sk\text{-}c \ \in \ Sp-Sk-C \mid sp\text{-}sk\text{-}c[2] = rec\text{-}req[16]\\
\rightarrow\ & ds\text{-}rec\text{-}req[15] := rec\text{-}req[3],\\
& ds\text{-}rec\text{-}req[17] := rec\text{-}req[4],\\
& ds\text{-}rec\text{-}req[18] := rec\text{-}req[5],\\
& ds\text{-}rec\text{-}req[16] := sr\text{-}rec\text{-}req[2],\\
& ds\text{-}rec\text{-}req[14] := sr\text{-}rec\text{-}req[7],\quad rec\text{-}req[22]\\
& \qquad := CR-REQ-TKN(rec\text{-}req[4],\\
& \qquad\quad rec\text{-}req[6])ds\text{-}rec\text{-}req[23]\\
& \qquad := rec\text{-}req[22],\quad DS-R-R'\\
& \qquad = \ DS-R-R \cup ds\text{-}rec\text{-}req[14],\\
& ds\text{-}rec\text{-}req[15],\quad ds\text{-}rec\text{-}req[16],\ ds\text{-}rec\text{-}req[17],\\
& ds\text{-}rec\text{-}req[18], ds\text{-}rec\text{-}req[23]\ Rec-Req'\\
& \qquad = \ Rec-Req \cup rec\text{-}req[22] \qquad (12)
\end{aligned}
$$

C. Modeling the Step 4, 5, and 6 of the Protocol

After the resource request is forwarded to the DS by the SP, the DS evaluates the request on constraints such as (a) The SR authentication, (b) Checking the CoI history for the resource request, (c) The user authentication response, and (d) The required user attributes. The DS stores the evaluation result at place RESP. If the result is positive, then the DS forwards the result to the SP for the resource release. If the response is negative, the resource request should be discarded. A Petri Net model for the process is shown in Figure 8 below. The rule for the aforesaid behavior is mapped on the transition *Eva-Resp* and shown in (13). In the rule, the dual

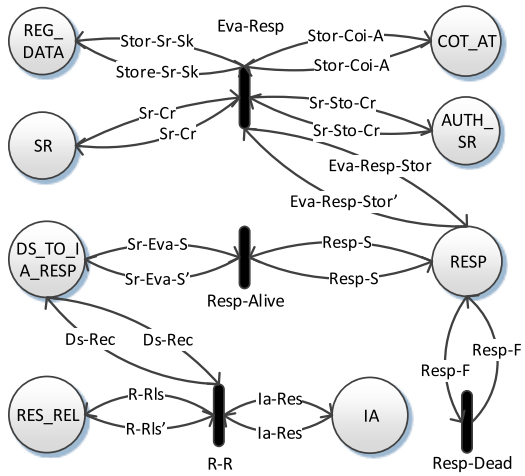

Fig. 8. Model for the Resource Release.

authentication is performed, i.e. (a) the user authentication, which is performed by comparing the credentials, such as the user name and the password, with the users home domain and (b) the domain authentication, to ensure that the domain is the part of a federation. Moreover, the DS maintains a CoI-history at the end to cross check whether the user has accessed the same resource in the past or some other resource of the same CoI class. The user provided attributes are compared for the CoI checking. After all the comparisons are performed, the result is stored at a place RESP.

$$
\begin{aligned}
\boldsymbol{R}\ (\textit{Eva-Resp}) &= \forall \textit{sr-sto-cr} \in Sr - Sto - Cr, \textit{sr-ent-cr} \\
&\forall \in Sr - Cr, \quad \forall \textit{eva-resp-stor} \in Eva - Resp - Stor, \\
&\forall \textit{stor-sr-sk} \in Stor - Sr - Sk, \\
&\forall \textit{stor-coi-a} \in Stor - Coi - A \mid \\
\textit{sr-sto-cr}[1] &= \textit{sr-ent-cr}[20], \quad \textit{sr-sto-cr}[2] = \textit{sr-ent-cr}[21], \\
\textit{stor-sr-sk}[2] &= \textit{sr-ent-cr}[19], \quad \textit{sr-ent-cr}[2] = \textit{stor-coi-a}[1], \\
\textit{sr-ent-cr}[4] &= \textit{stor-coi-a}[2], \quad \textit{sr-ent-cr}[5] = \textit{stor-coi-a}[3], \\
\textit{sr-ent-cr}[15] &= \textit{stor-coi-a}[4], \quad \textit{sr-ent-cr}[23] = \textit{stor-coi-a}[5] \\
&\rightarrow \textit{sr-ent-cr}[17] \\
&:= CR - AUTH - RS(\textit{sr-sto-cr}[1], \\
&\quad \textit{sr-sto-cr}[2], \quad \textit{stor-sr-sk}[2], \ \textit{stor-coi-a}[1], \\
&\quad \textit{stor-coi-a}[2], \ \textit{stor-coi-a}[3], \\
&\quad \textit{stor-coi-a}[4], \ \textit{stor-coi-a}[5]), \\
&\quad \textit{eva-resp-stor}[1] \\
&:= \textit{sr-ent-cr}[17], \\
Sr - Cr' &= Sr - Cr \cup (\textit{sr-ent-cr}[17]) \\
Eva - Resp - Stor' &= Eva - Resp - Stor \cup (\textit{eva-resp-stor}[1])
\end{aligned}
\tag{13}
$$

The results of the authentication process can be positive or negative. If the response is positive, then the transition *Resp-Alive* is fired. The rule for the said transition is shown in (14), where the DS generates the response as *true* against the request token and stores it at a place DS-TO-IA-RESP.

$$
\begin{aligned}
\boldsymbol{R}\ (\textit{Resp-Alive}) &= \forall \textit{resp-s} \in Resp - S, \\
&\quad \forall \textit{sr-eva-s} \in Sr - Eva - S \\
&\mid \textit{resp-s}[1] = True \rightarrow \textit{sr-eva-s}[1] := \textit{resp-s}[1] \\
Sr - Eva - S' &= Sr - Eva - S \cup (\textit{sr-eva-s}[1])
\end{aligned}
\tag{14}
$$

If the response is negative, the transition *Resp-Dead* is fired that depicts the failure, resulting in a termination of the process. The rule for the said transition is shown in (15).

$$
\begin{aligned}
\boldsymbol{R}\ (\textit{Resp-Dead}) &= \forall \textit{resp-f} \in Resp - F, \\
&\quad \forall \textit{sr-eva-f} \in Sr - Eva - F \\
&\mid \textit{resp-f}[1] = False \rightarrow \textit{sr-eva-f}[1] := \textit{resp-f}[1] \\
Sr - Eva - F' &= Sr - Eva - F \cup (\textit{sr-eva-f}[1])
\end{aligned}
\tag{15}
$$

Finally, the transition *R-R* will be triggered only if (13) is fired that complete the process of a secure resource release in a federation. The rule for the transition *R-R* is shown in (16).

$$
\begin{aligned}
\boldsymbol{R}\ (\textit{R-R}) &= \forall \textit{ia-res} \in Ia - Res, \quad \forall \textit{ds-rec} \in Ds - Res, \\
&\forall \textit{r-rls} \in R - Rls \mid \textit{ia-res}[21] = \textit{ds-rec}[1] \wedge \textit{ia-res}[22] \\
&= \textit{ds-rec}[2] \rightarrow \textit{r-rls}[1] := \textit{ia-res}[23], \\
R - Rls' &= R - Rls \cup (\textit{r-rls}[1])
\end{aligned}
\tag{16}
$$

V. Verification of the xDAuth Protocol

Verification is the process of demonstrating the correctness of an underlying system. To prove the correctness of the system under consideration, it is important to verify the system on certain parameters. Generally, system specifications and system properties parameters are used for the system verification. Every system has some specifications that define the system; whereas, system properties act as the axioms that must be proved by the system.

A. The xDAuth Model Verification Using the Z3 Solver

We have followed 3 main steps towards the verification of the xDAuth protocol. As a first step, we have modeled the xDAuth protocol in the HLPN. Secondly, we have defined the transition rules using the Z formal specification language. Such rules are then transformed into the array theory of the SMT-Lib. Finally, using a bounded model checking technique [51], [52], the Z3 solver [27] verify the system properties (in our case these are the security properties) of the protocol model to check for the satisfiability of the logical formulae over the system specification. The Z3 solver performs the computation and provides result as satisfiable *sat* or unsatisfiable *unsat*. If the generated result is *sat*, it means that there is a violation of the asserted property. The solver will generate a counter example for it. Alternatively, if the result is *unsat*, then this indicates that the property holds and the correctness of the system is proved. To verify the xDAuth protocol model, we have defined five security properties of which two are protocol specific and the other three are secrecy properties.

1) Security Properties: The protocol specific security properties are: (a) The Secure Secret Key Authentication, and (b) The Chinese Wall Security Policy. The secrecy properties that define the granular level security are: (c) The Confidentiality, (d) The Integrity, and (e) The Authenticity.

```
(assert (not (or (= (select IA-req-token-update 15) (select ds-ia-request 11))
(= (select sr-req-response 18) (select ds-ia-request 11))))) (check-sat)
```

Fig. 9. The Secure Secret Key Authentication.

```
(assert (not (or (= (select coi-response 0) coi-r)))) (check-sat)
```

Fig. 10. The CWSP Assertion.

```
(assert (not (or (= (select IA-req-Fin-content 22) (select srr-con 21))
(not (= (select IA-req-Fin-content 22)(select sr-mess 21)))))) (check-sat)
```

Fig. 11. The Confidentiality Assertion.

a) The secure secret key authentication: In the xDAuth protocol, at the time of the registration, a secure secret key is shared between the DS and each of the SP and the SR domain respectively. The secret key is used for the purpose of domain authentication. For the secure secret key authentication, we have used the secret keys and asserted this as a property. The code snippet for the secure secret key property is shown in Figure 9. The property holds if the secret key held by the SP and DS is similar and the secret key held by the SR and DS is similar, respectively. If the keys are matched, the solver generates unsat result, which indicates that the xDAuth protocol is secure and the DS can successfully authenticate each organization that is a part of a federation. Also, xDAuth authentication mechanism ensures the fact that the unauthorized user cannot pass the authentication due to secure secret key mechanism and user authentication mechanism in its home domain.

b) The chinese wall security policy (CWSP): The next property is the CWSP. The CWSP is about avoiding the CoI among clients in a federation. The CWSP maintains a set of the objects, each containing information of different companies that fall in one of the CoI classes [21]. Asserting the same for the xDAuth protocol, where different organizations may have CoI with each other, we have defined this as a security property. The code snippet for the CWSP is shown in Figure 10 below. In the property, we asserted that if a user has accessed the resource of a company of a specific CoI class and has performed some action on it such as read or write, he/she will not be allowed to access the resource of another company which is in the same CoI class.

c) The confidentiality: For a system to be secure, it is important to consider the confidentiality constraints. The confidentiality is defined as a liaison between two or more persons in which the communicated information is not disclosed deliberately to another person [53]. By following the definition of the confidentiality, we have made an assertion for testing the xDAuth protocol. The code snippet as shown in Figure 11 exemplifies the assertion. In the xDAuth protocol, the resource itself is confidential information in the SP domain. Therefore, the resource content of the SP domain must not match with the resource content of the SR domain before the resource information is released properly following the protocol flow. The result that we obtained against this assertion is *unsat*.

```
(assert (not (or (= (select coi-history 0) (select ia 4)) (= (select coi-
history 0) (select ds-ia-request 17))))) (check-sat)
```

Fig. 12. The Integrity Assertion.

```
(assert (not (or (= (select ds-ia 14)
(select ia 2)) (= (select ia 2) (select sr 5))))) (check-sat)
```

Fig. 13. The Authenticity Assertion.

d) The integrity: Another secrecy property for a system to be secure is the integrity. The integrity is defined as the property of a system that a malicious user cannot alter or modify the contents of a secured message block once protected [53]. That is, a person or system which does not have a write access on the protected message must not be able to modify the contents. In the code snippet as shown in Figure 12, we have matched the SP user's action which he/she came to perform on the protected message. If the action is the same which is delegated to him/her then integrity of the system is maintained. The result that we obtained against this assertion is *unsat*.

e) The authenticity: By definition, authentication is the process of identifying and verifying the person or entity who is accessing the information of the system [53]. Any information that is coming from a valid user of the system is considered to be authentic. In Figure 13, an assertion has been listed for the xDAuth protocol to check its authenticity. In the code snippet, we have matched the source of the information i.e. the email address of the SP domain user known to the SR domain user with the sender's email address of the information. The result that we obtained from the assertion is *unsat*. From this we can say that the xDAuth protocol preserves the security constraint of authenticity.

B. Results

To verify, the model of the xDAuth protocol is translated to SMT. The properties are also translated and specified in SMT. The Z3 solver takes the model and the properties to check whether the model satisfies the properties or not. If the system model satisfies the aforementioned properties using SMT and Z3 solver then this indicates that the model specifications can be stated as correct. The objective that we want to achieve in this paper is to verify the correctness of the xDAuth protocol without analyzing quality attributes such as performance and reliability. Therefore, executing the xDAuth model along with the asserted properties in the Z3 solver, the model of xDAuth works fine and produces results as per our expectations. For our implementation using SMT-LIB, we used QF_AUFLIA logic, which is used for closed quantifier-free linear formulas over the theory of integer arrays extended with free sort and function symbols. The results are illustrated in the Table IV in execution time of the xDAuth protocol when verified on the security properties.

The results illustrate that the xDAuth protocol executes in a finite time on verification. The execution time here indicates that the time solver takes to compute the satisfiability of

TABLE IV
EXECUTION TIME OF THE SECURITY PROPERTIES

Security properties	Exec. time
Secure secret key authentication	0.02 Sec
CWSP	0.01 Sec
Confidentiality	0.03 Sec
Integrity	0.02 Sec
Authenticity	0.01 Sec

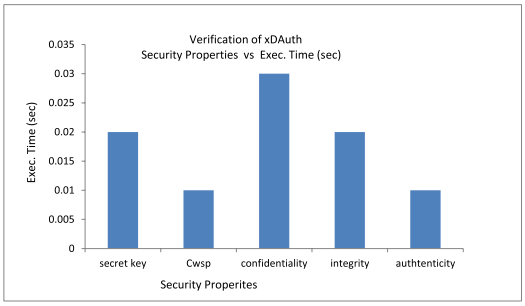

Fig. 14. Verification Results of the xDAuth Protocol.

the properties, which shows the correctness. Figure 14 plots the execution time taken by the Z3 solver on each security property of the xDAuth model.

VI. CONCLUSIONS

In a federation, where organizations are sharing their resources with other organizations, access to shared resources is readily governed by organizational and internal access control policies that define conditions on which access can be granted/denied. In this paper, we have used the HLPN and the Z language for the modeling and the analysis of the xDAuth protocol which is used for cross domain access control. The HLPN presents a graphical representation of the system whereas the Z specification language provides the mathematical representation which helps in the analysis of the behavioral and the structural properties of the system. By using formal techniques, we could understand and analyze the interconnections of the xDAuth protocol. For providing the proof of protocol correctness, the modeled system is transformed into the Z3 SMT solver. The obtained results revealed the fact that the asserted protocol specific and the secrecy properties complement the secure release of the resource in the xDAuth protocol. However, the secret key between the DS and each respective domain after the registration process can also be a single point of failure and should be backed by a strong logging system in order to track and audit all the administrative (domain registration, policy update etc.,) and non-administrative (resource access) activities. In our future work, we intend to incorporate a secure key mechanism backed by a logging system in the xDAuth protocol.

REFERENCES

[1] K. A. Alam, R. Ahmad, A. Akhunzada, M. H. N. M. Nasir, and S. U. Khan, "Impact analysis and change propagation in service-oriented enterprises: A systematic review," *Inf. Syst.*, vol. 54, pp. 43–73, Dec. 2015.

[2] OASIS. *Web Services Security (WSS) OASIS TC Homepage*, accessed on May 20, 2016. [Online]. Available: https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[3] E. Yuan and J. Tong, "Attributed based access control (ABAC) for Web services," in *Proc. IEEE Int. Conf. Web Services*, Los Alamitos, Jul. 2005, pp.561–569, doi: 10.1109/ICWS.2005.25.

[4] L. Lowis and R. Accorsi, "Vulnerability analysis in SOA-based business processes," *IEEE Services Comput.*, vol. 4, no. 3, pp. 230–242, Jul./Sep. 2011.

[5] E. Bertino, L. Martino, F. Paci, and A. Squicciarini, *Security for Web Services and Service-Oriented Architectures*. New York, NY, USA: Springer, 2009.

[6] *Web Services Security: What's Required To Secure A Service-Oriented Architecture*, accessed on May 20, 2016. [Online]. Available: http://www.oracle.com/us/products/middleware/identity-management/059410.pdf

[7] R. L. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein, "Federated security: The shibboleth approach," *ERIC Educause Quart.*, vol. 24, no. 4, pp. 12–17, 2004.

[8] E. Hammer, Ed. *IETF, The OAuth2.0 Authorization Protocol Draft*, vol. 28. USA: OAuth Working Group, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-oauth-v2-28

[9] D. Recordon and D. Reed, "OpenID 2.0: A platform for user-centric identity management," in *Proc. 2nd ACM Workshop Digit. Identity Manage.*, New York, NY, USA, 2006, pp. 11–16, doi: 10.1145/1179529.1179532.

[10] M. S. Ferdous and R. Poet, "Analysing power consumption of different browsers & identity management systems in mobile phones," *Int. J. Distrib. Parallel Syst.*, vol. 3, no. 2, pp. 21–41, Mar. 2012, doi: 10.5121/ijdps.2012.3203.

[11] M. Alam, X. Zhang, K. Khan, and G. Ali, "xDAuth: A scalable and lightweight framework for cross domain access control and delegation," in *Proc. SACMAT*, Innsbruck, Austria, Jun. 2011, pp. 31–40, doi: 10.1145/1998441.1998447.

[12] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, "Securing software defined networks: Taxonomy, requirements, and open issues," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 36–44, Apr. 2015.

[13] A. Akhunzada *et al.*, "Man-at-the-end attacks: Analysis, taxonomy, human aspects, motivation and future directions," *J. Netw. Comput. Appl.*, vol. 48, pp. 44–57, Feb. 2015.

[14] A. Akhunzada *et al.*, "Secure and dependable software defined networks," *J. Netw. Comput. Appl.*, vol. 61, pp. 199–221, Feb. 2015.

[15] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," *Proc. 14th IEEE Comput. Secur. Found. Workshop*, Jun. 2001, p. 82–96, doi:10.1109/CSFW.2001.930138.

[16] J. Goubault-Larrecq and F. Parrennes, "Cryptographic protocol analysis on real C code," *Verification, Model Checking, and Abstract Interpretation*. Berlin, Germany: Springer, 2005, pp. 363–379, doi: 10.1007/978-3-540-30579-8_24.

[17] M. Cheminod *et al.*, "Detecting chains of vulnerabilities in industrial networks," *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 181–193, May 2009, doi: 10.1109/TII.2009.2018627.

[18] J. C. Mitchell, M. Mitchell, and U. Stern, "Automated analysis of cryptographic protocols using Murϕ," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 1997, pp. 141–151, doi: 10.1109/SECPRI.1997.601329.

[19] D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song, "Towards a formal foundation of Web security," *Proc. 23rd IEEE Comput. Secur. Found. Symp. (CSF)*, Edinburgh, Scotland, July. 2010, pp. 290–304, doi: 10.1109/CSF.2010.27.

[20] N. Li, M. V. Tripunitara, V. Mahesh, and Z. Bizri, "On mutually exclusive roles and separation-of-duty," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 2, p. 5, 2007.

[21] D. F. C. Brewer and M. J. Nash, "The Chinese wall security policy," in *Proc. IEEE Symp. Secur. Privacy*, May 1989, pp. 206–214.

[22] *Cross Domain Assess Control and Delegation in Enterprise Applications (CDACDEA) Funded by the National ICT R&D*, accessed on May 20, 2016. [Online]. Available: http://aserg.com.pk/otw-portfolio/cross-domain-assess-control-and-delegation-in-enterprise-applications-cdacdea/

[23] L. Moura and N. Bjørner, "Satisfiability modulo theories: An appetizer," in *Formal Methods: Foundations and Applications*. Berlin, Germany: Springer-Verlag, 2009, pp. 23–36.

[24] M. Frade and J. S. Pinto, "Verification conditions for source-level imperative programs," Dept. Comput. Sci. Technol. Center, Univ. Minho, Braga, Portugal, Tech. Rep. DI-CCTC-08-01, 2008.
[25] S. U. R. Malik, S. U. Khan, and S. K. Srinivasan, "Modeling and analysis of state-of-the-art VM-based cloud management platforms," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 50–63, Jan./Jun. 2013.
[26] (2015). *SMT-Lib*. [Online]. Available: http://smtlib.cs.uiowa.edu/
[27] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Germany: Springer 2008, pp. 337–340, doi: 10.1007/978-3-540-78800-3_24.
[28] L. de Moura and N. BjÃ¸rner, "Satisfiability modulo theories: Introduction and applications," *Commun. ACM*, vol. 54, no. 9, pp. 69–77, Sep. 2011.
[29] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989, doi: 10.1109/5.24143.
[30] Y. Lin, S. U. R. Malik, K. Bilal, Q. Yang, Y. Wang, and S. U. Khan, "Designing and modeling of covert channels in operating systems," *IEEE Trans. Comput.*, to be published.
[31] W. Reisig and G. Rozenberg, Eds., *Lectures on Petri Nets I: Basic Models*. Berlin, Germany: Springer-Verlag, 1998.
[32] S. U. R. Malik and S. U. Khan, "Formal methods in LARGE-SCALE computing systems," *Proc. ITNOW*, vol. 55, no. 2, pp. 52–53, 2013, doi: 10.1093/itnow/bwt025.
[33] A. Armando and S. Ranise, "Scalable automated symbolic analysis of administrative role-based access control policies by SMT solving," *J. Comput. Secur.*, vol. 20, no. 4, pp. 309–352, 2012, doi: 10.3233/JCS-2012-0461.
[34] A. Armando and S. Ranise, "Automated and efficient analysis of role-based access control with attributes," in *Data and Applications Security and Privacy XXVI*. Berlin, Germany: Springer 2012, pp. 25–40, doi: 10.1007/978-3-642-31540-4_3.
[35] X. Jin, R. Krishnan, and R. Sandhu, "Reachability analysis for role-based administration of attributes," *Proc. ACM Workshop Digit. Identity Manage.*, New York, NY, USA, 2013, pp. 73–84, doi: 10.1145/2517881.2517891.
[36] F. Yang and S. Manoharan, "A security analysis of the OAuth protocol," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. Signal Process. (PACRIM)*. Victoria, BC, Canada, Aug. 2013, pp. 271–276, doi: 10.1109/PACRIM.2013.6625487.
[37] K. Jayaraman, V. Ganesh, V, M. Tripunitara, M. Rinard, and S. Chapin, "Automatic error finding in access-control policies," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, New York, NY, USA, 2011, pp. 163–174, doi: 10.1145/2046707.2046727.
[38] G. Lowe, "Breaking and fixing the Needham–Schroeder public-key protocol using FDR," in *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Germany: Springer, 1996, pp. 93–102, doi: 10.1007/3-540-61042-1_43.
[39] M. Miculan and C. Urban, "Formal analysis of Facebook Connect single sign-on authentication protocol," *Proc. SOFSEM*, 2011, pp. 1–11.
[40] R. Wang, S. Chen, and X. Wang, "Signing me onto your accounts through Facebook and Google: A traffic-guided security study of commercially deployed single-sign-on Web services," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 365–379, doi: 10.1109/SP.2012.30.
[41] K. Bhargavan, C. Fournet, and A. D. Gordon, "Verifying policy-based security for Web services," in *Proc. 11th ACM Conf. Comput. Commun. Secur.*, New York, NY, USA, 2004, pp. 268–277, doi: 10.1145/1030083.1030120.
[42] W. Mao and C. Boyd, "Towards formal analysis of security protocols," in *Proc. IEEE 6th Comput. Secur. Found. Workshop*, Franconia, NH, USA, Jun. 1993, pp. 147–158, doi: 10.1109/CSFW.1993.246631.
[43] C. J. F. Cremers, "The Scyther tool: Verification, falsification, and analysis of security protocols," *Computer Aided Verification*. Berlin, Germany: Springer, 2008, pp. 414–418, doi: 10.1007/978-3-540-70545-1_38.
[44] L. Viganò, "Automated security protocol analysis with the AVISPA tool," *Electron. Notes Theor. Comput. Sci.*, vol. 155, pp. 61–86, May 2006, doi: 10.1016/j.entcs.2005.11.052.
[45] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," *Computer Aided Verification*. Berlin, Germany: Springer, 2013, pp. 696–701, doi: 10.1007/978-3-642-39799-8_48.
[46] W. She, I. Yen, F. Bastani, B. Tran, and B. Thuraisingham, "Role-based integrated access control and data provenance for SOA based net-centric systems," in *Proc. IEEE 6th Int. Symp. Service Oriented Syst. Eng. (SOSE)*, Dec. 2011, pp. 225–234.
[47] G. Zhao, Z. Ba, X. Wang, F. Zhang, C. Huang, and Y. Tang, "Constructing authentication Web in cloud computing," in *Security and Communication Networks*. New York, NY, USA: Wiley, 2015.
[48] M. C. Reynolds and A. Bestavros, "Formal verification of cross-domain access control policies using model checking," Dept. CS, Boston Univ., Tech. Rep. BUCS-TR-2011-031, Dec. 2011.
[49] N. Bjørner, N. Tillmann, and A. Voronkov, "Path feasibility analysis for string-manipulating programs," *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Germany: Springer, 2009, pp. 307–321, doi: 10.1007/978-3-642-00768-2_27.
[50] B. Korel, "A dynamic approach of test data generation," in *Proc. IEEE Conf. Softw. Maintenance*, San Diego, CA, USA, Nov. 1990, pp. 311–317, doi: 10.1109/ICSM.1990.131379.
[51] H. Huang and H. Kirchner, "Formal specification and verification of modular security policy based on colored Petri nets," *IEEE Trans. Dependable Secure Comput.*, vol. 8, no. 6, pp. 852–865, Nov./Dec. 2011, doi: 10.1109/TDSC.2010.43.
[52] L. Cordeiro, B. Fischer, and J. Marques-Silva, "SMT-based bounded model checking for embedded ANSI-C software," *IEEE Trans. Softw. Eng.*, vol. 38, no. 4, pp. 957–974, Jul./Aug. 2012, doi: 10.1109/TSE.2011.59.
[53] C. Chen, P. Grisham, S. Khurshid and D. Perry, "Design and validation of a general security model with the alloy analyzer," in *Proc. ACM SIGSOFT 1st Alloy Workshop*, 2006, pp.38–47.
[54] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 1999.
[55] M. Ali, S. U. R. Malik, and S. U. Khan, "DaSCE: Data security for cloud environment with semi-trusted third party," *IEEE Trans. Cloud Comput.*, to be published.
[56] S. U. R. Malik, S. K. Srinivasan, S. U. Khan, and L. Wang, "A methodology for OSPF routing protocol verification," in *Proc. 12th Int. Conf. Scalable Comput. Commun. (ScalCom)*, Changzhou, China, Dec. 2012, pp. 1–5.
[57] S. U. R. Malik, S. K. Srinivasan, and S. U. Khan, "Convergence time analysis of open shortest path first routing protocol in Internet scale networks," *IET Electron. Lett.*, vol. 48, no. 19, pp. 1188–1190, Sep. 2012.

Quratulain Alam received the master's degree in information technology. She is currently pursuing the Ph.D. degree in information security and formal methods. Her research interests include cross domain access control frameworks and formal verification. She has also worked on a funded project "Cross Domain Access Control and Delegation in Enterprise Applications (CDACDEA)" by ICT R&D. The project was executed in the Applied Security Engineering Research Group (ASERG) lab active at COMSATS Institute of Information Technology, Islamabad, Pakistan.

Saher Tabbasum received the B.S. degree in computer science. She is currently pursuing the M.S. degree in software engineering. Her research interests include software engineering and formal verification.

Saif U. R. Malik received the Ph.D. degree from the Department of Electrical and Computer Engineering at North Dakota State University, USA, in 2014. He is currently an Assistant Professor with the Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan. His research interest revolves around the application of formal methods in large scale computing systems, software engineering, and security protocols.

Masoom Alam received the Ph.D. degree in computer sciences from the University of Innsbruck, Austria. He is currently an Associate Professor with the COMSATS Institute of Information Technology, Islamabad, Pakistan. His areas of interest include access control systems, model driven architecture, and work flow management systems. He received the best Ph.D. thesis award at the Ph.D. Symposium of Models 2006 Conference Genoa, Italy (Invited for Springer LNCS publication). He had two consecutive projects from the National ICT R&D Fund, which is a highly reputable national funding agency for research and development projects in Pakistan.

Tamleek Ali received the master's degree in computer sciences, where he is currently involved in remote attestation. He is currently an Assistant Professor with the Institute of Management Sciences, Pakistan. His research interests include dynamic remote attestation through behavior detection and analysis, and privacy in behavioral attestation systems. His findings related to privacy in behavioral attestation systems have been published in conferences and journals of international repute.

Adnan Akhunzada is currently a Ph.D. Fellow and an Active Researcher with the Center for Mobile Cloud Computing, University of Malaya, Malaysia. He had a great experience teaching international modules. He is a Senior Lecturer with CIIT, Islamabad, since 2011. He is author/coauthor in several high-impact major journal publications, conferences, and a book chapter. His current research interests include secure and dependable software defined networks, man-at-the-end attacks, lightweight cryptography, human attacker attribution and profiling, and remote data auditing.

Samee U. Khan is currently an Associate Professor of Electrical and Computer Engineering with the North Dakota State University, Fargo, ND, USA. His work has appeared in over 275 publications. He is Fellow of the Institution of Engineering and Technology and a Fellow of the British Computer Society.

Athanasios V. Vasilakos is currently a Professor with the University of Western Macedonia, Greece. He has authored or coauthored over 200 technical papers in major international journals and conferences, five books, and 20 book chapters. He served as a General Chair, a Technical Program Committee Chair for many international conferences. He served or is serving as an Editor of the IEEE renowned transactions and, also a General Chair of the Council of Computing of the European Alliances for Innovation.

Rajkumar Buyya (F'15) is currently a Professor of Computer Science and Software Engineering, a Future Fellow of the Australian Research Council, and the Director of the Cloud Computing and Distributed Systems Laboratory with The University of Melbourne, Australia. He is also serving as the founding CEO of Manjra soft, a spin-off company of the University, commercializing its innovations in cloud computing. He has authored over 525 publications and five text books. He is one of the highly cited authors in computer science and software engineering worldwide.