Behavioral Attestation for Business Processes (BA4BP)

Masoom Alam[1], Mohammad Nauman[1], Xinwen Zhang[2], Tamleek Ali[1] and Patrick C.K. Hung[3]
[1] Institute of Management Sciences, Pakistan
{mmalam,nauman,tamleek}@imsciences.edu.pk
[2] Samsung Information Systems America, USA
xinwen.z@samsung.com
[2] University of Ontario Institute of Technology, Canada.
patrick.hung@uoit.ca

Abstract

Service Oriented Architecture (SOA) is an architectural paradigm that enables dynamic composition of heterogeneous, independent, multi-vendor business services. A prerequisite for such inter-organizational workflows is the establishment of trustworthiness, which is mostly achieved through non-technical measures such as legislation, and/or social consent that businesses, or organizations simply pledge themselves to adhere. In our viewpoint, a business process can only be trustworthy if the behavior of all services in it is trustworthy. Trusted Computing Group (TCG) has defined an open set of specifications for the establishment of trustworthiness through a hardware root-of-trust. This paper has three objectives: firstly, the behavior of individual services in a business process is formally specified. Secondly, in order to overcome the inherent weaknesses of trust management through software alone, a hardware root of-trust devised by the TCG, is used for the measurement of the behavior of individual services in a business process. Finally, a verification mechanism is detailed through which the trustworthiness of a business process can be verified.

1. Introduction

Service Oriented Architecture (SOA) with underlying technologies like web services and web service orchestration facilitates smooth interaction among independent, multi-vendor data sources and legacy applications running on heterogeneous platforms across distributed information networks. Such interactions require intelligently interfaced application software and dynamic integration with other connected cooperative environments. As a result, more applications and services have been deployed which bring new businesses and pervasive information sharing. With these trends, the paradigm of SOA opens new vistas for businesses in the form of dynamic collaborations; however, abstracting the internals behind a single interface makes SOA more prone to security vulnerabilities. For example, it is extremely difficult to verify that an electronic health record or credit card number input into a service is used or updated in a trustworthy way. We believe that, a prerequisite for the realization of SOA-based inter-organizational workflows is the establishment of trustworthiness. However, according to current best practices, trustworthiness is mostly achieved through non-technical measures such as legislation, or social consent that businesses, or organizations simply pledge themselves to adhere.

Existing approaches for secure composition of business processes [8, 12, 11, 20, 7] do not take the behavior of individual services into account while composing business processes. We define the *behavior* of a service as *a set of observable actions or reactions of the service in response to its execution environment*. The execution environment comprises input data and the security policy associated with a service. In our viewpoint, a given business process is trustworthy if the data input to services is trustworthy and if all services in the business process enforce their security policies in a trustworthy manner. All existing approaches for composition of business processes are focused on the issues of authentication and authorization only. Authentication and authorization are primarily concerned with the verification of service identity and checking permissions for calling a specific service. Behavioral Attestation is a third dimension that goes well beyond the traditional view of authentication and authorization. It is the process of verification of 1) the integrity of the input data – *service intake* – and 2) correct enforcement of the security policy attached to a service – *service processing*.

We have laid down the following three requirements for the behavioral attestation of business processes. Firstly, a framework is needed that can explicitly specify the behavior of individual services in a business process. A formal means of specification helps to abstract the complex details of the underlying hardware and software.

Secondly, a mechanism is needed that can measure the dynamic behavior of services in a trustworthy way. Existing approaches for trust management through software alone – by their very principle – are uncompromising and have inherent weaknesses [15, 10, 4]. Trusted Computing Group (TCG) has taken a quantum leap in security, "a hardware root of trust", which provides secure storage for data and cryptographic keys. Remote attestation is an essential characteristic of Trusted Computing that provides a hardware supported evidence in enciphered form that a trusted environment actually exists on a remote platform [15, 10]. However, existing approaches for the realization of remote attestation measure the trustworthiness from binaries, configurations and properties. All of these techniques are low-level attestation techniques only and none of them define what a trusted behavior actually is and how to specify it [6]. Thus, these approaches cannot be used as a ready-made solution for the behavioral attestation of business processes.

Finally, trusted third parties are needed that can play a role beyond traditional certification authorities and PrivacyCAs [1]. These attestation authorities should be able to attest the behavior of a business process and can issue credentials based on their behavior.

In this paper, we present *Behavioral Attestation for Business Processes (BA4BP)*, which is a framework for the identification, specification, and verification of different behaviors associated with a business process. The framework is realized in three steps 1) behavior specification, 2) behavior measurement and; 3) behavior verification.

In Step 1 (cf. Section 3), the framework specifies the behavior of a service in a formal way. Each service is characterized by a set of attributes, which include the data, input as parameters to a service. A change in the values of these attributes can influence the overall behavior of a service. Note that this change should be in accordance with the security policy attached to the service.

In Step 2 (cf. Section 4), the framework uses the Trusted Platform Module (TPM) for measuring the dynamic behavior of the services in a business process. The measurements are stored in the secure storage provided by the TPM.

In Step 3 (cf. Section 5), the framework uses a verification mechanism for the attestation of different recorded behaviors – called *enforcement behaviors*. In order to prove the trustworthiness of different services in a business process, the framework compares the expected behaviors with the enforcement behaviors and draws conclusions regarding the trustworthiness of individual services.

Outline: Some background information about Trusted Computing and Remote Attestation is provided in Section 2. In Section 3, we formally define the semantics of services behavior. Section 4 covers the measurement of these behaviors. Section 5 describes the verification of the measured behaviors. Section 6 highlights some of the related works carried out in the past in this area and finally, Section 7 provides an outlook on this contribution.

2. Remote Attestation

The TCG [19] has devised a mechanism for bringing trust to the different aspects of computing including the PC client, mobile platforms and storage media. The hardware chip is called the Trusted Platform Module (TPM) and is a secure co-processor responsible for providing a root of trust. It has the capability of storing platform configurations in shielded memory locations called the Platform Configuration Registers (PCRs). The PCRs are designed to store cryptographic hashes, specifically SHA-1 hashes of software loaded for execution. PCRs are tamper-resistant as they can only be set using a special operation called pcr_extend. The irreversibility of SHA-1 ensures that once a hash has been saved in a PCR, it cannot be removed by any software loaded after the extend operation has been performed. Therefore, the hashes in the PCR can be used to report the configurations of a platform to a challenging party in a trusted way. This concept of reporting the platform configurations to a challenger in order to provide trust in the behavior of a target platform is termed as remote attestation.

Remote attestation is an important feature of Trusted Computing [19], which allows a remote party to verify the integrity of another platform through trust tokens submitted by the TPM on the target platform. In a typical remote attestation scenario, a service provider verifies the integrity of the platform, before dispatching a resource to it. For example, an hospital verifies the integrity of the client platform before dispatching medical record of a patient. Existing approaches for the realization of remote attestation such as Binary Attestation [16], Policy-Reduced Integrity Measurement Architecture (PRIMA) [13] etc., are low-level attestation techniques only.

The main problem with these approaches is architectural. Rather than relying on the binaries, configurations or properties, it is important to capture the dynamic behavior of a platform. Behavior is a dynamic thing, which can only be measured at run time. Model-based behavioral Attestation (MBA) [6] is one of few approaches that define

a high-level framework, for the explicit specification of the behavior of a platform. BA4WS [5] is an extension of the MBA framework for the realization of remote attestation through XACML policies.

3. Services Behavior

A *Business Process* (BP) is composed of a finite number of services denoted by $\{s_1, s_2, \ldots, s_n\}$ where $s_i \in S$. A service is defined as a function call, which may take a number of inputs and may return an output. Formally:

$$s : x_1, x_2, \ldots, x_n \mapsto y$$

where $s \in S$, x_i are input parameters to the service and y is the output value.

An *activity* within a business process is defined as a scenario in which a service s_1 can access another service s_2 and is denoted by $s_1 \rightarrow^+ s_2$ where $+$ sign indicates that s_1 can access s_2 one or more times in a single (authenticated) session. Depending on the role of a service in an activity, it can be treated as a *subject* or *object service*. For example, in activity $s_1 \rightarrow^+ s_2$, s_1 is treated as a subject service and s_2 as an object service.

Example 1 *An online hospital service calls a laboratory service for further investigation regarding some patient. In this activity, we write* $hospital \rightarrow^+ lab1$ *where* $hospital$ *is the subject service and* $lab1$ *is the object service.*

Each service in an activity has a finite number of attributes $ATT = \{a_1, a_2, \ldots, a_n\}$, associated with it such as its start time, end time and location. Each attribute is defined as a $(name, value)$ pair and a particular attribute a of a service s is referred to as $s.a$ where $s \in BP$ and $a \in ATT$. Service parameters are also categorized as its attributes and are referred in the same fashion.

Example 2 *The laboratory service (introduced in Example 1) forwards the electronic health record to another laboratory service as input and retrieves test results for specific diseases as output from the second laboratory service. In this case, the health record is a parameter of the service and is thus treated as an attribute.*

Each activity can have a *security policy*, or simply *policy* associated with it. This policy defines a set of conditions C, under which a subject service s_1 can access object service s_2. Each $c \in C$ is composed of a set of subject service and object service attributes.

Example 3 *The laboratory service can have a policy that requires the verification of the supplied credit card information for sufficient credit. For this purpose, the laboratory service may call a billing service to verify that the supplied credit card number is valid.*

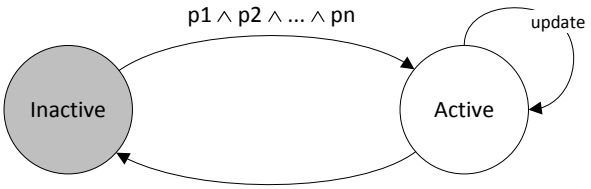

Figure 1. State Transition

An activity policy also contains a set of attribute update actions A_u, which update the attributes of services involved in the activity. Attribute update actions are also treated as internal/external services. For example, an attribute update action may update the supplied electronic health care record using an internal or external service.

Each service can be in one of the following states: *active* or *inactive*. Figure 1 describes the state transition diagram of a service. State *active* is associated with update actions and transition from state *inactive* to *active* is associated with a set of conditions. Whenever an object service authorizes its usage to a subject service after the evaluation of associated conditions, it is considered in the active state. Similarly, whenever a subject service invokes an object service for some processing, it is also considered in the active state.

Example 4 *The billing service (introduced in Example 3), before supplying the credit card information, verifies that the laboratory service is authorized to access this information. If the authorization conditions hold, the billing service as well as the laboratory service is activated.*

We define a business process as:

Definition 1 (Business Process) *A Business Process is a five-tuple,* $BP = (A_c, S, C, A_u, T)$ *where,*

A_c *is a set of activities,*
S *is a set of services,*
C *is a set of conditions,*
A_u *is a set of attribute update actions and;*
T *is a set of service states.*

The behavior of an individual activity is characterized by the behavior of its services' attributes. All attributes, which are used in the security policy of an activity affect the overall behavior of the activity. If an attribute changes due to an attribute update action or due to some environmental change, the behavior of the service in an activity may change. For example, a service may be deactivated, or activated due to an attribute change. Secondly, more than one activity sessions may exist for an object service. An *activity system* is therefore defined as a set of activities in which object services are same for each session. An

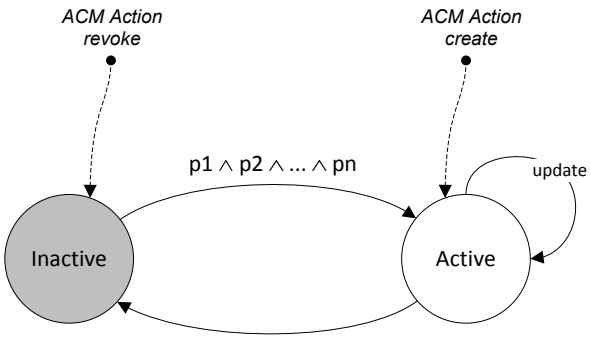

Figure 2. ACM Actions Associated with Each State

attribute change may cause a change in the whole view of an activity system. *Activity behavior* captures the dynamic behavior of an activity system. State transitions of a service are captured by the *state transition behavior*. Finally, *attribute update behavior* keeps track of all service attribute updates in an activity policy.

In general, for each of these behaviors, the formal specification takes the following form. First, the behavior itself is defined. Afterwards, the trustworthiness of the corresponding behavior is specified.

3.1. Activity Behavior

In order to capture the correct activity system state, *activity behavior* uses an Access Control Matrix (ACM). Two ACM actions have been formally defined. These are called ACM action *create* and *end*. These actions are associated with different states of a service. For example, ACM action *create* is associated with the state *active*. It adds a right r to the row s and column o in the ACM where s and o are subject and object services. Similarly, ACM action end is associated with states *inactive*. Below, we define the ACM and the two ACM actions formally:

Definition 2 (Access Control Matrix) *The ACM of an activity is defined as $A : O \times R \rightarrow 2^S$ where O is a set of active object services, R is a set of rights, and S is a set of active subject services.*

ACM Action $create$**:** Given the set of active subject services S and active object services O, ACM action $create$ inserts subject service s and object service o to S and O respectively. It also adds newly granted right r to the ACM. Formally,

$$create(s, o, r) = (O', S', R, A') \ \ where$$
$$S' = S \cup \{s\},\ O' = O \cup \{o\} \ \ and$$
$$A'(o, r) = A(o, r) \cup \{s\}$$

ACM Action end**:** When a right r is revoked from a subject service s on object service o, ACM action end removes r from the ACM. Moreover, if no other subject service is exercising any right on o, o is also removed from the set of active object services (O). Similarly, if s is not exercising any rights on any object, it is also removed from the list of active subject services (S).

$$end(s, o, r) = (O', S', R, A') \ \ where$$
$$S' = S \ominus \{s\} =^{def} \begin{cases} S & |A^{-1}(\{s\})| \geq 2 \\ S - \{s\} & otherwise \end{cases}$$
$$O' = O \ominus \{o\} =^{def} \begin{cases} O & \sum_{i=1}^{n} |A(o, r_i)| \geq 2 \\ O - \{o\} & otherwise \end{cases}$$
$$and \ \ A'(o, r) = A(o, r) - \{s\}$$

We define the trustworthiness of the activity behavior as follows:

Definition 3 (Activity Behavior Trustworthiness) *The activity behavior is trustworthy if all of the following conditions hold:*

- $create(s, o, r) \rightarrow o \in O' \wedge s \in S' \wedge s \in A'(o, r)$
- $end(s, o, r) \rightarrow S' = S \ominus \{s\} \wedge O' = O \ominus \{o\} \wedge A'(o, r) = A(o, r) - \{s\}$

3.2. State Transition Behavior

State *active* is associated with a set of conditions (cf. Fig 1). This set of conditions is primarily dependent on subject and object service attributes, which play a key role in sorting out state transition decisions for a service.

We define state transition behavior as follows:

Definition 4 (State Transition Behavior) *Let t_i be a service state and C be the set of conditions associated with its state transition. Transition from t_i to t_{i+1} is denoted by $t_i \rightarrow_C t_{i+1}$ and is allowed if and only if C is true.*

An attribute of a service is defined as a $(name, value)$ pair. In order to include the behavior property, it is treated as a triple $(name, value, behavior)$. We define the trustworthiness of state transition behavior as:

Definition 5 (State Transition Behavior Trustworthiness) *ATT is a set of attributes of services s_1 and s_2 where $s_1 \rightarrow^+ s_2$. State transition behavior is trustworthy if and only if the behavior of all attributes involved in the state transition of s_1 from state t_i to t_{i+1} is trustworthy and all conditions are true. Formally,*

$$\forall j.s.attr_j.behavior = trusted \wedge C = TRUE$$

where $attr_j \in ATT$ and C is the set of conditions associated with the service state transition.

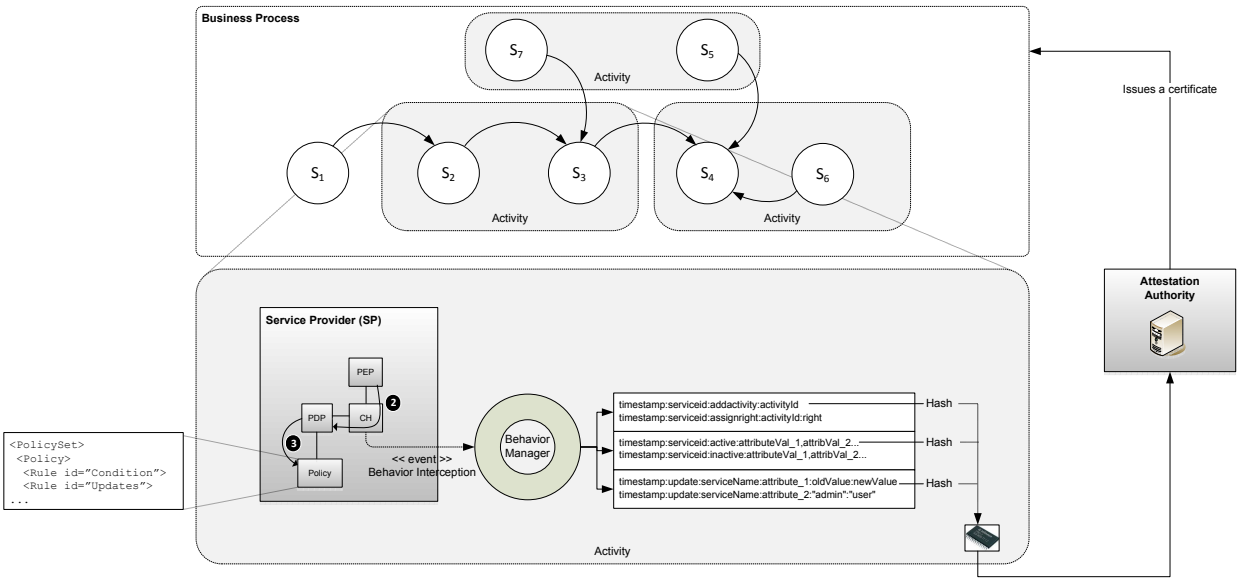

Figure 3. Target Architecture

3.3. Attribute Update Behavior

Attribute update actions are an integral part of the service behavior. We define attribute update behavior as follows:

Definition 6 (Attribute Update Behavior) *Given an attribute $s.a_i$, and an update service 'update', the attribute update behavior is defined as the application of the update service on the attribute. The application of the service yields a new value which is assigned to the attribute. Formally:*

$$update : s.a_i \mapsto s.a_i'$$

We define the trustworthiness of attribute update behavior as:

Definition 7 (Attribute Update Behavior Trustworthiness) *An attribute update behavior is trustworthy iff the corresponding attribute is in a trusted state and the service updating it is also trusted. Formally:*

$$s.a_i'.behavior = update.behavior \wedge s.a.behavior$$

We conclude that a behavior process can only be trustworthy if all the aforementioned behaviors are trustworthy. Below, we describe a framework for the measurement of these behaviors. Afterwards, their verification is detailed.

4. Service Behavior Measurement

Behavior measurement refers to a mechanism in which the behavior of the services is measured while the activities are being performed. Each service in an activity has an associated security policy that dictates how the activity should be performed in a business process. Basically, in this step, the trustworthiness of the *execution environment* of a service, which comprises input data and security policy, is measured. In our target architecture, XACML [21] data flow model is used as an abstract representation of the execution environment of an activity policy. We have extended the data flow model of XACML in order to capture the behavior of an activity. The reason is that, XACML is a de-facto standard for the specification of complex access control scenarios.

Figure 3 describes our target architecture. After authentication, each service is assigned a unique ID. All subsequent calls from the service are treated as being in a single authenticated session. After authentication, the *Policy Enforcement Point* forwards the request to the *Policy Decision Point* for access evaluation. We assume that a proper authentication mechanism has been employed such as public key infrastructure or biometric devices etc. for service authentication. The Policy Decision Point retrieves the applicable policy and required attributes for policy evaluation through the *Context Handler*. After successful evaluation, the Policy Decision Point sends the result of the policy evaluation to the Policy Enforcement Point. The Policy Enforcement Point maintains a separate session for each respective activity.

In our target architecture, a *reference monitor* is a component responsible for the evaluation and enforcement of an activity policy. Thus, the Policy Decision Point, the Policy Enforcement Point and the Context Handler all

together are referred to as a reference monitor. The Context Handler is responsible for gathering attributes from various sources and for performing updates on these attributes.

The *Behavior Manager* is an internal service responsible for measuring the behavior of the reference monitor during the evaluation and enforcement of an activity policy. Each access decision call from the Policy Enforcement Point to the Policy Decision Point is intercepted by the Behavior Manager service, which records the call and its parameters through the use of Trusted Platform Module (TPM) structures and capabilities. The TPM stores trust tokens in a shielded, tamper resistant memory locations referred to as Platform Configuration Registers (PCRs). The values of PCRs are secure SHA-1 hashes and can only be extended through a special operation pcr_extend. These values can be reported to authorized challenger through public key signature of the TPM using Attestation Identity Keys (AIKs). The private part of AIKs are accessible only to the TPM and the reported values can therefore be verified to be signed by a genuine hardware TPM and untampered. The Behavior Manager utilizes the PCRs for storing the information related to the intercepted events. The existing usage of PCRs is limited to hashes of files and executables [16, 13]. There has been no effort at utilizing the PCRs for storing run-time values of different data structures such as activity behavior and state transition behavior etc. Below, we describe how the Behavior Manager measures and stores the dynamic behavior of the reference monitor.

In essence, the Behavior Manager performs three actions: 1) Intercepts all calls within the Context Handler involving authorization decisions and attribute updates, 2) maintains a log for each behavior and; 3) extends the respective PCRs for each entry in the log.

For recording the runtime behavior of the reference monitor, the Behavior Manager maintains three *trust logs*: Activity log, 2) State transition log, and 3) Attribute update log. The log files represent the *enforcement behavior* of the corresponding reference monitor.

The *activity log* maintains the information regarding changes to the ACM of an activity system. The log is initialized with an empty ACM. Afterwards, every change to the ACM is recorded with an entry to the log. For instance, if a subject service is activated, it is added to the ACM and a corresponding entry is concatenated in the activity log. Similarly, assigning of a right to a subject service is also recorded in the log. The hash of each log entry is stored in the TPM using the pcr_extend operation. The pcr_extend operation takes the hash of the new entry, concatenates it with the existing value of the PCR, computes the SHA-1 of the resulting structure and stores the value in the PCR. Each entry P_{e_x} is computed as follows:

$$P_{e_x} := SHA1(sha-1(e_{x-1})||SHA1(e_x))$$

```
<ActivityLog>
 <DynamicEntry>
  timestamp:serviceid:addactivity:activityId
 </DynamicEntry>
</ActivityLog>
<StateTransitionLog>
 <DynamicEntry>
  timestamp:serviceid:active:attVal_1,attVal_2...
 </DynamicEntry>
</StateTransitionLog>
<AttributeUpdateLog>
 <DynamicEntry>
  timestamp:update:serviceName:att_1:oldVal:newVal
 </DynamicEntry>
</AttributeUpdateLog>
```

Figure 4. Enforcement Behaviors

This re-use of the PCR allows for the storage of an infinite number of hashes in the limited number of PCRs available to the TPM. Figure 4 shows an example activity log.

The *state transition log* stores the service state changes occurring within a reference monitor. Whenever a state change occurs in a reference monitor, the Behavior Manager creates a log entry in the activity log. This entry is composed of the timestamp of the event, the service identifier, the state change and the values of attributes of the state. Figure 4 shows an example state transition log.

Similar to the active state behavior, the Behavior Manager also records the attribute update behaviors. Changes to service attribute are recorded by the Behavior Manager in the *attribute update log*. Each entry in the log stores the timestamp, the name of the service to which the attribute belongs, old value of the attribute and the updated value. Figure 4 shows a sample of the attribute update log.

Once these logs and their corresponding PCRs are created and updated with the values of the individual service behaviors. They can then be sent out to appropriate authorities for their evaluation. A business process can have these logs for each corresponding activity. In the following, we highlight the role of an *attestation authority* that can verify the trustworthiness of these logs generated in the measurement step.

5. Behavioral Attestation

Behavioral attestation refers to a mechanism, which verifies that the behavior of individual services in a business process is trustworthy. An attestation authority collects the behavior logs such as activity log and attribute update log, of different services and their corresponding PCR values. All these logs and their corresponding PCRs are verified for the following three properties of a business process: 1) *measurement correctness*, 2) *individual behavior trustworthiness* and 3) *activities sequence*. In the following, we first present a high-level description of verification of these logs against their corresponding PCRs.

Afterwards, their combined effect of trust levels on the business process is presented.

The measurement correctness refers to a property that individual behaviors recorded for each activity are correct and trustworthy. For this, the following procedure is adopted: Let $E = \{e_1, e_2, \ldots, e_n\}$ be the set of dynamic entries in a log file representing an individual behavior. The measurement correctness procedure takes the hash of each individual entry and computes the final value according to the following procedure:

$$C_{e_x} := SHA1(SHA1(e_x - 1)||SHA1(e_x))$$

The resulting value for C_{e_n} must be equal to the TPM signed value of the PCR. If the value of the PCR matches with the computed value, the measurement can be considered trustworthy, i.e.:

$$C_{e_n} = P_{e_n}$$

where P_{e_n} is the PCR value returned by the TPM after the nth entry has been recorded.

The individual behavior trustworthiness can be verified by inspecting each individual behavior entry against the security policy. For example, it can be easily verified that attribute updates were trustworthy or not by looking at their individual values.

These individual behaviors can be combined to draw useful conclusion about different activities taking place in a business process. For example, the activity sequence can be verified with the help of state transition behavior of an activity. Each individual activity has a log that includes the timestamp of the attribute change etc. It can be verified with the help of these timestamps, the sequence in which these actions were performed was trustworthy.

The attestation authority can use these individual behaviors to compute a *trust vector*, which describes the trust level of the complete business process.

Definition 8 (Trust Levels) *Let $L = \{l_1, l_2, \ldots, l_n\}$ is a set of trust levels, where a partial order is defined on these trust levels, i.e. $l_1 \prec l_2 \prec \ldots \prec l_n$.*

Based on the verification of the individual behaviors, each service can be associated with one of these trust levels. During service invocation, the subject service can choose from among multiple paths based on the trust levels of object services. Figure 5 describes selection from among different execution paths based on the trust level of different services. Moreover, the aggregate of these levels can be used to specify the trustworthiness of the complete business process.

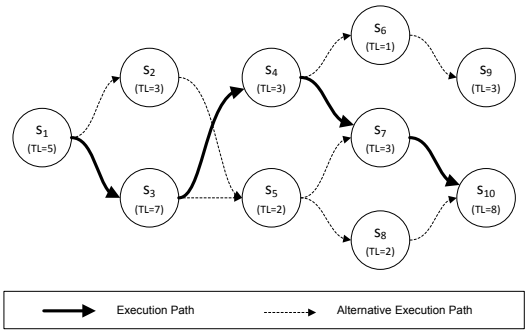

Figure 5. Alternative Execution Paths

6. Related Work

A big community is working on the issues related to secure document exchange among the inter-organizational workflows.

[18] presents an a web service architecture, in which current web services platforms split their processing in to two protection domains. The smaller part is responsible for executing security sensitive information such as credit card numbers etc. The larger domain is reserved for general processing. Also, the information flows taking place in the smaller part is analyzed.

[14] propose a secure access mechanism for remote services. In order to convey the web service security requirements such as specific encryption algorithms etc., an adaptive function module is introduced.

[17] enhances the current web services security framework to facilitate the control of information flow through service chains. It extends the basic security models by introducing the concepts of delegation and pass-on.

Bertino, Castano and Ferrari [8] deal with workflow security in centralized and closed environments. R. Bhatti et al. [9] have extended the web services access control models to incorporate contextual information, such as time, location, or environmental state etc. Different security requirement specifications in XML have been detailed in [11, 12, 20].

All these approaches are focused on the issues of authentication and authorization only. The only approach towards incorporating remote attestation on the web services level – WS-Attestation [22] – only deals with the communication aspects of the problem. To the best of our knowledge dynamic behavioral attestation of business processes has not been investigated before.

7. Outlook

Our project, *Dynamic Behavioral Attestation for Mobile Platforms (DBAMP)* [3, 2] is focusing on the attestation

of mobile services. The project is funded by National ICTR&D and its aim is to design and develop trustworthy electronic government services that can be deployed across different arms of the Pakistan government such as lodging tax returns and communication between citizens and other government agencies. BA4BP framework is part of the DBAMP project and the current prototype uses Openmoko mobile phone as the underlying mobile platform. We have not yet arrived at a perfect solution for measuring the trustworthiness of business processes. However, we believe that, as mobile devices are getting more and more sophisticated and powerful, this overhead is not going to hinder the process of attestation. The DBAMP framework will open new avenues for bringing trust into business processes. This will significantly improve dynamic collaboration beyond non-technical measures such as social consent, legislation and contracts etc. In order to improve the efficiency of behavior measurement and attestation, a software TPM, which can emulate the hardware root-of-trust is also under consideration. We are working on a step-wise implementation of the dynamic behavioral attestation of our approach.

References

[1] About IAIK/OpenTC PrivacyCA. `http://trustedjava.sourceforge.net/index.php?item=pca/about`.

[2] ICTR&D Fund, Pakistan. `http://www.ictrdf.org.pk`.

[3] Project: Dynamic Behavioral Attestation for Mobile Platforms. `http://serg.imsciences.edu.pk/projects/dbamp/`.

[4] M. Alam, J.-P. Seifert, and X. Zhang. A model-driven framework for trusted computing based systems. In *EDOC '07: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*, page 75, Washington, DC, USA, 2007. IEEE Computer Society.

[5] M. Alam, X. Zhang, M. Nauman, and T. Ali. Behavioral Attestation for Web Services (BA4WS). In *SWS'08: Proceedings of the ACM Workshop on Secure Web Services (SWS) located at 15th ACM Conference on Computer and Communications Security (CCS-15)*, New York, NY, USA, 2008. ACM Press.

[6] M. Alam, X. Zhang, M. Nauman, T. Ali, and J.-P. Seifert. Model-based Behavioral Attestation. In *SACMAT '08: Proceedings of the thirteenth ACM symposium on Access control models and technologies.*, New York, NY, USA, 2008. ACM Press.

[7] S. Anderson, J. Bohren, T. Boubez, et al. Web Services Trust Language (WS-Trust). *Public draft release, Actional Corporation, BEA Systems, Computer Associates International, International Business Machines Corporation, Layer*, 7.

[8] E. Bertino, S. Castano, and E. Ferrari. Securing XML Documents with Author-X. *IEEE INTERNET COMPUTING*, pages 21–31, 2001.

[9] R. Bhatti, E. Bertino, and A. Ghafoor. A Trust-Based Context-Aware Access Control Model for Web-Services. *Distributed and Parallel Databases*, 18(1):83–105, 2005.

[10] D. Grawrock. *The Intel Safer Computing Initiative Building Blocks for Trusted Computing*. Intel Press, `http://www.intel.com/intelpress/sum_secc.htm`, 2005.

[11] E. Gudes. Modelling, specifying and implementing workflow security in Cyberspace. *Journal of Computer Security*, 7(4):287–315, 1999.

[12] W. Huang and V. Atluri. SecureFlow: a secure Web-enabled workflow management system. In *Proceedings of the fourth ACM workshop on Role-based access control*, pages 83–94. ACM New York, NY, USA, 1999.

[13] T. Jaeger, R. Sailer, and U. Shankar. PRIMA: Policy-Reduced Integrity Measurement Architecture. In *SACMAT '06: Proceedings of the eleventh ACM Symposium on Access Control Models and Technologies*, pages 19–28, New York, NY, USA, 2006. ACM Press.

[14] H. Lufei, W. Shi, and V. Chaudhary. Adaptive secure access to remote services. In *SCC '08: Proceedings of the 2008 IEEE International Conference on Services Computing*, pages 173–181, Washington, DC, USA, 2008. IEEE Computer Society.

[15] S. Pearson. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[16] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, Berkeley, CA, USA, 2004. USENIX Association.

[17] W. She, I.-L. Yen, and B. M. Thuraisingham. Enhancing security modeling for web services using delegation and pass-on. In *ICWS*, pages 545–552. IEEE Computer Society, 2008.

[18] L. Singaravelu, J. Wei, and C. Pu. A secure information flow architecture for web services. In *SCC '08: Proceedings of the 2008 IEEE International Conference on Services Computing*, pages 182–189, Washington, DC, USA, 2008. IEEE Computer Society.

[19] Trusted Computing Group. `http://www.trustedcomputinggroup.org/`.

[20] J. Wainer, P. Barthelmess, and A. Kumar. W-RBAC-A WORKFLOW SECURITY MODEL INCORPORATING CONTROLLED OVERRIDING OF CONSTRAINTS. *International Journal of Cooperative Information Systems*, 12(4):455–485, 2003.

[21] XACML 2.0 Specification Set. Available at: `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`.

[22] S. Yoshihama, T. Ebringer, M. Nakamura, S. Munetoh, T. Mishina, and H. Maruyama. WS-Attestation: Enabling Trusted Computing on Web Services. *Test and Analysis of Web Services*, pages 441–469, 2007.