Technology Final Report
Privacy-Preserving Data Dissemination and Adaptable Service Compositions in Trusted & Untrusted Cloud

September 2016

Prepared for
The Northrop Grumman Cyber Research Consortium
As part of IS Sector Investment Program

Prepared by
Bharat Bhargava
CERIAS, Purdue University
Table of Contents

1 Executive Summary ............................................................................................................. 1
   1.1 Statement of Problem .................................................................................................... 1
   1.2 Current State of Technology ......................................................................................... 2
   1.3 Proposed Solution ......................................................................................................... 5
   1.4 Technical Activities, Progress, Findings and Accomplishments ................................. 18
   1.5 Distinctive Attributes, Advantages and Discriminators .............................................. 27
   1.6 Tangible Assets Created by Project .............................................................................. 28
   1.7 Outreach Activities and Conferences .......................................................................... 29
   1.8 Intellectual Property Accomplishments ....................................................................... 29
2 General Comments and Suggestions for Next Year .......................................................... 29

List of Figures

Figure 1. Solution architecture.............................................................................................. 5
Figure 2. Cloud service anomaly detection ......................................................................... 6
Figure 3. Classification example using one-class SVM....................................................... 9
Figure 4. AOP example for service policy enforcement ....................................................... 11
Figure 5. Structure of active bundle ................................................................................... 12
Figure 6. Secure data dissemination model ....................................................................... 13
Figure 7. Authentication ticket creation and validation ....................................................... 14
Figure 8. Publish – Subscribe architecture for data dissemination .................................. 17
Figure 9. Isolated AB execution in a Docker container ........................................................ 18
Figure 10. Effect of number of services on dynamic service composition time .................. 19
Figure 11. Effect of number of QoS constraints on dynamic service composition time ....... 19
Figure 12. Effect of number of service categories on dynamic service composition time ... 20
Figure 13. Throughput of service monitoring for different concurrency levels .................... 21
Figure 14. Percentage of requests served within a certain time with service monitoring ...... 21
Figure 15. EHR Dissemination in cloud .............................................................................. 22
Figure 16. Hospital information system main page .............................................................. 23
Figure 17. Password-based authentication of a client at AS ............................................. 23
Figure 18. EHR retrieved for doctor with high level of browser’s crypto capabilities .......... 24
Figure 19. EHR retrieved for doctor with low level of browser’s crypto capabilities .......... 24
Figure 20. Detailed EHR of a patient retrieved for doctor .................................................. 25
Figure 21. Patient’s data accessible by Insurance Company .............................................. 25
Figure 22. Patient’s data accessible by University Researcher ......................................... 26
Figure 23. Flow visualization panel for EHR prototype ..................................................... 26

List of Tables

Table 1. Executive Summary ................................................................................................. 1
Table 2. Service performance/security parameters .............................................................. 7
Table 3. Amazon EC2 testbed characteristics ................................................................... 20
1 Executive Summary

<table>
<thead>
<tr>
<th>Title</th>
<th>Privacy-Preserving Data Dissemination and Adaptable Service Compositions in Trusted &amp; Untrusted Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Bharat Bhargava</td>
</tr>
<tr>
<td>Principal Investigator</td>
<td>Bharat Bhargava</td>
</tr>
<tr>
<td>Funding Amount</td>
<td>$210,000</td>
</tr>
<tr>
<td>Was this a continuation of Investment Project?</td>
<td>Yes</td>
</tr>
<tr>
<td>TRL Level</td>
<td>3</td>
</tr>
<tr>
<td>Key Words</td>
<td>Cloud security, data privacy, resiliency, adaptability, dynamic reconfiguration</td>
</tr>
</tbody>
</table>

Table 1: Executive Summary

1.1 Statement of Problem

Cloud computing creates new security challenges with a large attack surface. Micro-services based cloud system architectures for DoD applications require systematic monitoring of service operations to satisfy their resiliency (withstand cyber-attacks, and sustain and recover critical function) and antifragility (increase in capability, resilience, or robustness as a result of mistakes, faults, attacks, or failures) goals.

When clients interact with a cloud service, they expect certain levels of Quality of Service (QoS) guarantees, expressed as service performance, security and privacy policies. Controlling compliance with service level agreements (SLAs) requires continuous monitoring of services in an enterprise, as sudden changes in context can cause performance to deteriorate, if not result in the failure of a whole composition. To provide optimal performance in the enterprise cloud architecture under varying contexts, we need context-awareness and adaptation mechanisms for SOA and cloud service domains. Cloud platforms are vulnerable to increasingly complex attacks that could violate the privacy of data stored on them or shared with web services, which is especially detrimental in case of mission-critical operations. In order to mitigate these problems, cloud enterprise systems need to integrate proactive defense mechanisms, which provide increased resiliency by treating potentially malicious service interactions and data sharing before they take place.
There is a need for development of unified models for performance and security monitoring of operations that provide valuable input for achieving situation-awareness, dynamic adaptability and restoration of services in the face of changes in context, and effective mechanisms for detection and sharing of threat data, as well as enforcing cross-domain security and Quality of Service (QoS) constraints. Controlled privacy-preserving data dissemination and filtering models are needed to ensure protection of the privacy of sensitive data in trusted and untrusted clouds.

1.2 Current State of Technology

Service Monitoring & Adaptable Compositions

Current industry-standard cloud systems such as Amazon EC2 provide coarse-grain monitoring capabilities (e.g. CloudWatch) for various performance parameters for services deployed in the cloud. Although such monitors are useful for handling issues such as load distribution and elasticity, they do not provide information regarding potentially malicious activity in the domain. Log management and analysis tools such as Splunk [16], Graylog [17] and Kibana [18] provide capabilities to store, search and analyze big data gathered from various types of logs on enterprise systems, enabling organizations to detect security threats through examination by system administrators. Such tools mostly require human intelligence for detection of threats and need to be complemented with automated analysis and accurate threat detection capability to quickly respond to possibly malicious activity in the enterprise and provide increased resiliency by providing automation of response actions.

Development of runtime-auditing systems for mobile and web-based services has been the focus of many research efforts. Li et al. [11] describe a system for auditing runtime interaction behavior of web services. They use finite state automata to validate predefined interaction constraints, where message interception is bound to the particular server used for deploying Web services. Simmonds et al. [10] present a more comprehensive auditing solution for checking behavioral correctness of web service conversations. They use UML 2.0 Sequence Diagrams as a property specification language, which are then transformed to automata by multiple monitors that check the validity of safety and liveness properties. Their proposal is for a specific application server, since they utilize an event mechanism provided by that server. Guinea et al. [4] present an auditing system that is able to report and monitor functional requirements and quality-of-service constraints of BPEL (Business Process Execution Language) processes. Their approach leverages data collection, including message interception for auditing. However, their solution is technology-dependent and they do not address the auditing of cloud-based service interactions.

To support flexible auditing of the behavior pattern for composite services, Wu et al. [12][13] demonstrate an “aspect extension” to WS-BPEL, in which history-based pointcuts specify the pattern of interest within a range, and advices describe the associated action to manage the process if the specified pattern occurs. The solution they provide addresses specific orchestration engines, which is not a generic solution for modern cloud-based and mobile services. In [12] and [11] the identification of trusted services and dynamic trust assessment in SOA are studied. Malik et al.
[12] introduce a framework called RATEWeb for trust-based service selection and composition based on peer feedback. It is based on a set of decentralized techniques for evaluating reputation-based trust with ratings from peers. However, they do not take into account initial service invocations and the secondary services in compositions. Spanoudakis et al. [13] present an approach to keep track of trusted services to address the compliance of promises expressed within their service level agreements (SLAs). The trust assessment is based on information collected by monitoring services in different operational contexts and subjective assessments of trust provided by different clients (consumers) situated in specific operational context. They further address the issue of unfair ratings by combining user rating and quality of service monitoring. Approaches like [11] and [13] are not suitable for compositions with many services, because the monitoring system would need to collect intensive information from a lot of peers and clients, which would make it very expensive.

Gamble et al. [14] present a tiered approach to auditing information in the cloud. The approach provides perspectives on auditable events that may include compositions of independently formed audit trails. Filtering and reasoning over the audit trails can manifest potential security vulnerabilities and performance attributes as desired by stakeholders. [15] introduces a system to model the essential security elements and define the proper message structure and content that each service in the composition must have, based on a security meta-language (SML) that models the security relevant portions of the standards for their consistent, comprehensive, and correct application. Both of these approaches focus on how services can comply with established standards, but their implementation requires strong support from the cloud provider (and hence dependency on a certain technology) and extensive changes in the current infrastructures. On the other hand, the solution proposed for agile and resilient monitoring in this work is generic and can easily be applied to different technologies.

**Secure Data Dissemination**

Cryptographic solutions such as Public Key Infrastructure (PKI) [3], Identity-based Encryption (IBE) [4], Attribute-based Encryption (ABE) [5] can be used to ensure authorized data access in cross-domain data sharing. However, these solutions typically rely on a Trusted Third Party (TTP) to issue, manage, and verify the cryptographic keys. TTP must be highly trusted as it could be capable of decrypting data without authorization. It becomes a high-value target and a single point of trust and failure. If the keys of TTP are breached, the data protected by the keys issued and managed by the TTP can also be compromised. Furthermore, the access policies in these mechanisms are limited to a simple policy i.e. the recipient has the private key, identity, or attribute that would allow decryption.

Existing Web service standards address security extensions for Service Object Access Protocol (SOAP)-based services. WS-Security provides specifications for credential exchange, message integrity, and message confidentiality during service interactions. WS-Policy provides specifications for advertising policies of services and policy requirements of clients. These standards could be sufficient for point-to-point security and policy enforcement in static service invocations, but they fall short of the policy transmission and enforcement in dynamic service invocations.

Access control in web services is well-studied. Security Assertion Markup Language (SAML) is an open-standard specification and a framework for exchanging authentication and authorization
information between parties in XML [6]. It defines three parties: (a) the principal requesting a service, the Identity provider (IdP) authenticates the principal and issues identity assertions, and the service provider (SP) uses the assertions and authorization policies for making access control decisions. It has been used to provide single sign-on (SSO) in a single domain. The implementation uses browser cookies to maintain the authentication state information. However, cross-domain implementation is a problem because cookies cannot be transferred across different DNS. Domain specific proprietary implementations further lead to non-interoperable solutions. Shibboleth defines an architecture based on SAML and provides an open-source implementation that allows federated identity management, authentication, and authorization [7]. It enables cross-domain single sign-on. The solution is prone to the TTP related issues because of its dependence on the IdP.

Several general approaches have been proposed for controlling access to shared data and protecting its privacy. DataSafe is a software-hardware architecture that supports data confidentiality throughout their lifecycle [8]. It is based on additional hardware and uses a trusted hypervisor to enforce policies, track data flow, and prevent data leakage. Applications running on the host are not required to be aware of DataSafe and can operate unmodified and access data transparently. The hosts without DataSafe can only access encrypted data, but it is unable to track data if they are disclosed to non-DataSafe hosts. The use of a special architecture limits the solution to well-known hosts that already have the required setup. It is not practical to assume that all hosts will have the required hardware and software components in a cross-domain service environment.

A privacy preserving information brokering (PPIB) system has been proposed for secure information access and sharing via an overlay network of brokers, coordinators, and a central authority (CA) [9]. This solution is based on the notion that the brokers may not be trusted and could collude to correlate, infer, or leak information. To provide protection, it proposes an approach to divide and allocate the responsibility and processing among multiple brokers so that no single component has enough control to make a meaningful inference from the information disclosed to it. However, the ownership, distribution, and management of components are a challenge in a large cross-domain system. The approach does not consider the heterogeneity of components such as different trust levels and policy conflicts among them. It uses a centralized TTP to manage metadata, joining/leaving of brokers, and key management. The use of TTP creates a single point of trust and failure. The information can be leaked if the TTP is attacked and compromised.

Other solutions have been proposed that address secure data dissemination when the recipients are not known in advance. Pearson et al. present a case study of EnCoRe project that uses sticky policies to manage the privacy of shared data across different domains [10]. The main idea of sticky policies is to make data and policies inseparable so that an unauthorized recipient cannot access the data without satisfying the policies. In the EnCoRe project, the sticky policies are enforced by a TTP and allow tracking of data dissemination. Other methods of implementing sticky policies include using Identity-based Encryption (IBE) or Trusted Platform Module (TPM). The approach is prone to TTP-related issues. The sticky policies are vulnerable to attacks from malicious recipients.
1.3 Proposed Solution

In this project, we propose a novel approach that uses a distributed network of *service activity monitors* to audit and detect service behavior and performance changes, adaptively update service compositions and securely share data in a cloud enterprise. By integrating components for service performance monitoring, dynamic service reconfiguration, and adaptable data dissemination, the proposed model aims to provide a unified architecture for agile and resilient computing in trusted and untrusted clouds. The overall architecture of the proposed model is demonstrated in Figure 1.

General characteristics of the solution are as follows:

- Each service domain, such as a cluster of machine instances in the cloud or a set of mobile services in a close proximity to each other, has a service monitor that tracks interactions among the services in the domain as well as outside the domain.

- The local service monitors (Monitor A, Monitor B etc.) gather performance and security data including response time, response status, authentication failures, etc., among other parameters for each service by intercepting service requests and utilizing available performance monitoring software. The data collected are logged in the database of each local monitor, and mined using unsupervised machine learning models to detect deviations from normal behavior. The analysis results are reported to a central monitor in the form of summary statistics for the services.

- The central monitor utilizes information submitted by local monitors to update trust values of services and reconfigure services/service compositions to provide resiliency against attacks and failures.

- Detection of service failures and/or suboptimal service performance triggers restoration of optimal behavior through dynamic reconfiguration of service compositions.
Privacy-preserving dissemination of data between services is achieved using active bundles. Likewise, data services in the cloud utilize active bundles for protected data storage that enforces fine-grain security policies associated with the usage of the data items when authorizing access.

While we focused on the development of a distributed monitoring system, dynamic service composition algorithms and active bundles in the previous phase of this project, our efforts were concentrated on the extension of active bundles to support a richer feature set to provide privacy-preserving data dissemination in untrusted clouds, as well as detection of anomalies through finer-grain monitoring in cloud service domains in this project period.

1.3.1. Cloud Service Anomaly Detection & Adaptable Compositions

In this section we present our system architecture for the monitoring of cloud services and detection of anomalies in order to provide adaptable and resilient service operation in a cloud enterprise. Figure 2 shows a high-level overview of service monitoring and anomaly detection in the proposed architecture.

Monitoring in the enterprise architecture is distributed in the sense that each service domain, such as a cluster of machine instances in the cloud, has a service monitor that tracks interactions among the services in the domain as well as interactions with services or users outside the domain. When a service is deployed, it is registered with the local monitor of its domain in order to be discoverable by other services or users. The local monitors have access to all interactions with the services registered in their domain and they gather interaction/performance data streams containing items for response time, response status, authentication failures etc. among other parameters for each service using interceptors transparent to each service implementation. Services in each domain are also tracked using aspect-oriented programming (AOP)-based software monitors for parameters requiring finer-grained control. The data collected are mined by the anomaly detection module of...
the domain and reported to the central monitor in the form of summary health statistics and trust values for the services. These statistics are utilized by the dynamic service composition module when making decisions about which services to include in a specific orchestration to meet user requirements. Below we provide details on the two major components of the adaptability framework: Unsupervised learning for anomaly detection and monitoring/enforcement using AOP.

1.3.1.1. Unsupervised learning for service anomaly detection

Research in machine learning has resulted in various models for detection of outliers in different types of data. While supervised and unsupervised classification models have been applied with success to a variety of domains [19], robust models for detecting anomalies and failures in service operation are still lacking. The main shortcoming of supervised anomaly detection models is that they require a large amount of training data and can only provide accurate results on anomalies that were previously observed in the system. This makes such models unable to capture threats/anomalies that are completely new, which is essential in an environment of ever-growing security vulnerabilities and attacks.

In this project we focused on the investigation and development of unsupervised models for outlier/anomaly detection in service behavior. A significant advantage of unsupervised models is that the training data required is gathered from the behavior of services operating under normal conditions (possibly in an isolated environment/private cloud); i.e. no attack data is required to train these models. Specifically, we focused on two unsupervised learning models, k-means clustering and one-class support vector machines (SVM), as described below. Training of the models is performed with data gathered under normal system operation (i.e. isolated execution under a controlled runtime environment).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Cloud services</th>
<th>Cloud data services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of requests/sec</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Bytes downloaded/sec</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Bytes uploaded/sec</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Total error rate</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CPU utilization</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Memory utilization</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Number of authentication failures</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Number of connection failures</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Number of disk reads/writes</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Network latency</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Service response time</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Disk space usage</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Number of database connections</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 2: Service performance/security parameters
Service performance and security parameters that are used in the learning process for general cloud-based services and data services are shown in Table 2. Note that this is not an exhaustive list and various other relevant parameters that can be obtained during service runtime through monitoring can be integrated into the learning algorithms easily.

**K-means Clustering**

K-means clustering partitions \( n \) observations into \( k \) clusters in which each observation belongs to the cluster with the nearest mean [20, 21]. When applied to the service anomaly detection problem, k-means clustering finds clusters of parameter values of normal service behavior during the training phase, using the data obtained with service monitoring under normal operation. During the online anomaly detection process, data gathered by the service monitors are utilized to measure the distance of the service behavior (i.e. values of performance/security parameters) at each time point to all clusters found by the algorithm, as shown in Algorithm 1. If the value does not fall in any cluster, an anomaly signal is raised.

**Training:**

- **Input:** Matrix \( V_{d \times t} \) of service performance record
- \( d \): number of performance parameters
- \( t \): number of time points observed
- **Cluster each set of performance parameter values using k-means algorithm**

**Testing (system operation):**

- for each service interaction log
  - measure distance of performance parameter values to each cluster
  - assign time point to closest cluster
  - if latest interaction does not belong to any cluster
  - raise anomaly signal

**Algorithm 1. K-means clustering for service anomaly detection**

**One-class Support Vector Machines (SVM)**

One-class SVM [23] is an extension of the well-known support vector machines (SVM) classification algorithm, where training is performed using only positive examples and test instances are classified as belonging or not belonging to the single (positive) class. Essentially, one-class SVM learns a decision function for novelty detection, which is what we try to achieve in service anomaly detection to mitigate attacks with no well-known signature. SVM constructs a decision hyperplane boundary as shown in Figure 3, based on normal runtime conditions of the service it is trained for. During the online anomaly detection phase, instances lying outside the boundary for normal operation are classified as anomalous, resulting in an anomaly signal.
1.3.1.2. AOP-based actions for request interception

Aspect-oriented programming (AOP) was originally proposed to complement the object-oriented paradigm by separating and encapsulating crosscutting concerns from the main logic of applications [31]. AOP enables us to inspect and modify the functionality of a program at compile time or runtime. More specifically, we can add features, orthogonal to the current logic of a program. In this context, orthogonality means we do not need to modify the existing source code of a program.

AOP is built around the following concepts:

**Joinpoint.** A joinpoint is a specific event (or point) in the control flow of a program. There are different types of joinpoints including method, field, and constructor joinpoints. For example, in Java, events such as calling of a method, accessing or modifying a field, and execution of a constructor are joinpoints.

**Pointcut.** A collection of joinpoints is called a pointcut. Pointcuts are AOP expressions (similar to regular expressions) that match particular joinpoints. More specifically, they define at what points during the execution of a program, we are interested to be notified to inject our custom logic (which is called an advice) to the control flow of that program.

**Advice.** Advice (similar to an event handler) is a method that is executed in response to a pointcut match event.

**Aspect.** An aspect is a reusable component that encapsulates crosscutting concerns and it is a bundle of one or more pointcuts and advices. Logging is not tied to the logic of programs and crosscuts almost all components in a program. Therefore, it is an example of a crosscutting concern that can be implemented as an aspect.

**Weaving.** The process of injecting the aspect code into the main functionality of a program is called weaving.

**Invocation.** An invocation is a class that represents a joinpoint and its relevant context (including information about the called method, its arguments, etc.) at runtime.

The AOP-based service monitoring model involves a number of policy monitoring and
enforcement components and their interactions with the trust manager service. The monitoring components are deployed in target service domains to monitor the execution of services. These components are extensible and pluggable and allow the addition or removal of new security aspects. Security aspects enable the monitoring framework to support new policies flexibly [33].

In Figure 4 [33], we illustrate how we can use AOP to intercept all service invocations within a service. This figure shows that the SampleWebService invokes another web service through the wsInvokerMethod(). This operation is performed by calling a library method called invokeService(). To intercept this service invocation, we create an aspect called WSMonitorAspect. Every aspect has a pointcut definition and one or more advices. We define a pointcut to precisely capture the invocations of invokeService(). The next step is to define proper advices to encapsulate the logic of what we want to do once this pointcut is matched. We define an around advice which encapsulates the target service invocation. The order of events is as follows:

1. Before the invocation of the invokeService() method, the corresponding pointcut (call(invokeService (..))) is matched and therefore, aroundAdvice() is called.
2. The aroundAdvice() method encapsulates what we want to inspect before calling the invokeService method. In this example, we extract the context information and then call the trust manager (TM) service. TM on behalf of the monitoring component contacts the policy engine (through a XACML request with relevant metadata) and returns the response. At the end of this step, the control flow returns to the invokeService method to be executed. At this point, the pointcut gets matched again and the aroundAdvice() method is called. If the report flag is enabled, it means that the service has violated the policies and it must be reported back to the trust manager (with relevant metadata for further analyses).
3. In the last step, the control flow of the execution returns back to the service to continue its execution.

The potential actions that an AOP-based service monitoring component is capable of performing include the following [33]:

- **Aborting the service.** In this strategy, the monitoring component prevents the service from continuing its execution by raising an unchecked exception or calling the System.exit() method. This strategy is interruptive and terminates the whole application. In this strategy, the untrusted service would be added to a blacklist to prevent future selection by other clients.

- **Replacing or redirecting the service.** In this strategy, instead of terminating the service and the whole application, the monitoring component replaces the untrusted service with a trusted service in the same category during the runtime. This mechanism enables the security framework to isolate the malicious services by redirecting all their incoming requests.

- **Throttling the service (rate control).** This strategy will only pause a service invocation for a limited time. This may be helpful to control the traffic to a target service. Another use case would be to slow down the invocations when the trust manager observes a malicious pattern. This delay may last until the unusual activities are analyzed.
• **Rearranging the topology dynamically.** This strategy may replace multiple services at once to maintain a required property. For example, if a highly trusted service gets compromised, the security framework may decide to replace two services with the lowest trustworthiness with two other services to maximize the trustworthiness of the overall service composition.

• **Retrying service invocation.** In case of high system load and intermittent service outage, this strategy may retry to invoke the next service for a predefined number of times.

Monitoring and policy enforcement using AOP in the adaptability framework is complementary to the function of local monitors deployed in every service domain to gather service performance and security parameters for anomaly detection. In addition to helping gathering of fine-grain performance/security parameters, AOP advices allow for faster predefined responses to well-known attacks/anomalies as discussed above to quickly recover normal system operation.

### 1.3.2. Privacy-Preserving Data Dissemination in Cloud

We propose a policy–based distributed data dissemination model, which provides secure data dissemination, i.e. every service gets access only to those parts of data for which it is authorized. The goal of the proposed solution is to selectively disclose information based on policies, minimize the unnecessary disclosure and ensure security and privacy of the information. Our solution uses Active Bundle (AB) to achieve this [27, 28, 29]. The use of AB ensures that the security policies are communicated and enforced.
Active Bundles Overview

An active bundle (AB) is a self-protecting data mechanism that includes sensitive data, metadata (policies) and a policy enforcement engine (Virtual Machine) for policy enforcement as illustrated in Figure 5. Clients interact with services by sending an AB, which contains encrypted data about their request and the policies associated with the data. AB is a data protection mechanism, which can be used to protect data at various stages throughout its lifecycle. AB is a robust and an extensible scheme that can be used for secure cross-domain data dissemination. Our implementation also supports context-based and trust-based data dissemination.

Figure 5. Structure of active bundle.

AB includes the following components:

**Sensitive data:** It is the digital content that needs to be protected from privacy violations, data leaks, unauthorized dissemination, etc. The digital content can include documents, pieces of code, images, audio, video files etc. This content can have several items, each with a different security/privacy level and an applicable policy to ascertain its distribution and usage.

**Metadata:** It describes the active bundle and its policies. This can include information such as AB identifier, information about its creator and owner, creation time, lifecycle etc. It also includes policies that govern AB’s interaction and usage of its data, such as access control policies, privacy policies, dissemination policies etc.

**Policy Enforcement Engine (or Virtual Machine, VM):** It is a specific-purpose VM used to operate AB, protect its content and enforce policies (for example, disclosing to a service only the portion of sensitive data that it requires to provide service).

Further details of the active bundle solution can be found at [27].

In this project period, we extended our previous work on active bundles to support a richer feature
set to provide privacy-preserving data dissemination in trusted and untrusted clouds. The enhanced solution provides data dissemination based on cryptographic capabilities of a browser (designed in collaboration with W3C/MIT group), in addition to Role-Based Access Control policies [24].

Data access depends on the following:
- Subject’s role
- Cryptographic capabilities of a browser
- Authentication method: Password-based authentication is considered to be less secure than hardware-based or fingerprint.
- Trust level of subject (User or Service)
- Source network: If an authorized user logs in from an unknown network, then she will get less data compared to when she logs in from the corporate intranet.
- Client’s device: When an authorized client tries to access data from his/her mobile phone, then less data will be accessible, compared to access from a desktop computer.

![Diagram of Secure Data Dissemination Model](image)

**Figure 6. Secure data dissemination model**

Below we describe the main extensions to the operation of active bundles to provide privacy-preserving data dissemination in trusted and untrusted clouds.

### 1.3.2.1. Authentication Ticket Creation and Validation

The initial data request from an unauthenticated client is redirected from the Cloud Provider to the Authentication Server (AS). Figure 7 shows the details of the protocol for an example scenario demonstrating dissemination of electronic health records (EHR).
Before describing the protocol, it is necessary to list the assumptions of the system:

- **Secure channel**: Communication among all parties will be encrypted. No eavesdropping attacks are possible.
- The client is a Web browser, and the data owner is the Hospital entity.
- The pair of keys $K_m$ is set before the system starts. The authentication server (AS) will be responsible for maintaining the private key. The data owner will have access to the public key to verify the signature of AS.
- The Hospital entity runs 4 Node.js servers: Hospital, Doctor, Insurance, and Researcher.
- The servers Doctor, Insurance, and Researcher interact with the Active Bundle.
- The server Hospital receives the first request from the client.
- Clients do not know about the authentication server. So, they will always initiate communicating with the Hospital server.

The protocol operates as follows:

**Step 1**: The client sends an HTTP GET request to Hospital. Hospital expects to receive a valid ticket and an AB request. In this implementation the client is redirected to the authentication server. In the future, clients will directly be redirected to the appropriate service (Doctor, Insurance or Researcher) if a valid ticket is included in the request.

**Step 2**: Hospital redirects the client’s request to the client to authenticate against the authentication server and receive a valid ticket. The redirection message includes the service
(Doctor, Insurance, or Researcher) to which the client must connect after getting the ticket and the role of the user.

Step 3: The client sends an HTTP GET request to the authentication server to obtain the HTML of the web page as a result of the redirection. As it was mentioned before, the HTTP GET request includes the service (Doctor, Insurance, or Researcher) to which the client must connect after getting the ticket and the role of the user.

Step 4: The authentication server sends the HTML file of its web page to the client, which includes in a hidden form the information received in the redirection url.

Step 5: The client sends an HTTP POST with the credentials to the AS. If the credentials are valid, AS produces a valid ticket for the client. If credentials are not valid no ticket will be generated.

Step 6: The authentication server redirects the client to the proper service with the new valid ticket.

Step 7: The client sends an HTTP GET request to the corresponding service. The corresponding service (Doctor, Insurance, or Researcher) expects to receive a valid ticket and an AB request. In this occasion the ticket will always be valid. The server, however, will go through the following validation process:

- If ticket is not present, discard request.
- If ticket is present but is not correctly signed, then discard the request.
- If ticket is correctly signed, then check Client_ID (e.g. IP address). If Client_ID does not match the source IP address, then discard the request.
- If Client_ID is correct check Expiration_Time. If the ticket has already expired, then go to Step 2.
- If the ticket has not expired, then perform the steps required to process the request with the Active Bundle and go to Step 8 to provide data to the client.

Step 8: The service (Doctor, Insurance, or Researcher) will disseminate the data approved by the Active Bundle after processing a valid ticket.

### 1.3.2.2. Privacy-Preserving Publish/Subscribe Cloud Data Storage and Dissemination

Figure 8 (created by Dr. Leon Li at NGC) shows a public cloud that provides a repository for storing and sharing intelligence feeds from various sources. The use of a public cloud requires the stored data to be encrypted. The cloud includes three main components:

- **Collection Agent** that is used to gather data from multiple distributed intelligence feeds. It maps similar data across different feeds and logically orders them for storage.
- **CryptDB** that stores encrypted data and provides SQL query capabilities over encrypted data using a search engine based on SQL-aware encryption schemes.
- **Subscription API** that provides various methods for data consumers to enable authorized access to data.

The incoming data may have different applicable data access and usage policies due to multiple source feeds and multiple data classifications in a single feed. For instance, the data may be classified as top secret, secret, confidential, and public. In order to ensure secure access to and usage of data, the policies specified by the source must be evaluated and enforced at all times...
including storage, access, and usage. We use Active Bundles to provide data confidentiality and privacy-preserving data disclosure in a publish/subscribe environment that may use trusted or untrusted cloud with multiple data sources and subscribers.

During AB creation process, data owner, in addition to sensitive data, specifies keywords (keyword phrase) along with access control policies. The identifier for AB, keywords and policies are grouped and stored in CryptDB in encrypted form, and access control policies are defined using the policy annotation feature provided by CryptDB. This allows us to encrypt data differently based on policies [26], so only authorized subjects for the target data are allowed to search over the database for AB meta information. Then, based on those mapped information, keyword phrases are indexed by a Lemur Indri [25] engine in order to provide data identification and efficient search. The index returned by Lemur will be stored in CryptDB. To search for an AB, Lemur should query CryptDB. This approach prevents leaking information on AB by hiding indexing information and it allows to make use of efficient search. The search result returns identifiers for ABs and ABs with matching identifiers are fetched from the object store. To query an AB, subjects are required to authenticate with CryptDB to retrieve sensitive information. For an unauthorized subject AB will be invisible and will not be a part of document corpus, i.e. will not participate in retrieving matching AB objects. CryptDB allows to execute SQL queries over encrypted keywords using a collection of efficient SQL-aware encryption schemes. Therefore, the public cloud environment would never get access to decrypted data and even if the cloud environment is compromised, the attackers would not be able to decrypt the data. Database of AB objects has an anonymized schema with encrypted data. Currently, we are using CryptDB with MySQL server, and with the manually created index, inserting new metadata into CryptDB and retrieving identifier for target AB is possible with manual SQL commands, which have been verified to work with node.js REST services.

A subject (user or a process) can access specific data by using APIs or performing a search using SQL. Password-based authentication is handled by the front-end Web service and after fetching AB from CryptDB the data access is authorized by executing the related Active Bundles. For example, if an authenticated insurance service issues a query to get billing information for all patients with keywords “Graduate Student Purdue Hospital”, then the first $N$ relevant AB objects are retrieved from the database. $N$ objects are chosen based on results given by Lemur Indri Retrieval Engine. The underlying retrieval model can be configured and selected between TF-IDF or Latent Semantic Indexing (LSI). Then retrieved AB objects can evaluate data access requests against the applicable policies and disclose data only if the requesting service (insurance in our example) is authorized for the billing data attribute.
Key Management: Cryptographic keys are needed for data encryption and decryption for AB and to decrypt the results of search over encrypted data stored in the cloud. These keys need to be dynamically issued and revoked. We use a trusted service for key management.

1.3.2.3. Isolated Execution of ABs

ABs currently require execution on the host machine, and the service which receives AB from an unknown source might be reluctant to execute it. Therefore, we propose to execute AB in Linux containers, a lightweight virtual machine. Using namespaces allows us to create a separate view from the host, and the host machine can consider the AB process running as a user process, even if privileged commands can be executed inside the container. Therefore, we will create a lightweight sandbox to execute AB.

To manage Linux containers, we use Docker [30]. Docker allows us to create a private registry, where we can create and export internal purpose VMs. The central authority service manages the private registry and it can export VMs via a REST service. Other services such as Doctor and Hospital services can fetch the VMs from the central private registry [see Figure 9].

Also, Docker provides tools to validate containers. It is possible for an attacker to deliver incorrect code inside the container with a man-in-the-middle-attack. However, the private registry can create SSL connections, so this issue can be resolved. Docker also provides features to check integrity of virtual machines. Cryptographic hashes can be generated from the container’s snapshot and it can be signed with the PKI system. This allows reluctant users to execute AB in an isolated environment.
1.4 Technical Activities, Progress, Findings and Accomplishments

The main technical activities for this project consist of the implementation of distributed service monitoring/adaptable service compositions and active bundles for secure data dissemination, experiments with the prototype system, integration with two other NGCRC projects, as well as preparation of a demo for NGC TechFest 2016 and a poster for the CERIAS Security Symposium at Purdue. The details of the technical activities and our main findings are described below.

1.4.1 Implementation of Service Monitoring & Adaptable Service Compositions

In the prototype distributed service monitoring system, each local service monitor has been implemented using Apache Axis2 valves for intercepting all service requests in the domain and each service domain includes a MySQL database, in which useful data (response time, response status, CPU usage, memory usage) about each service gathered by the monitor is logged. Additionally, AOP-based service interceptors were added in this project period to allow for finer-grain monitoring and policy enforcement capability. The central monitor is implemented as a web service on Amazon EC2, which has its own database to store health, endpoint address and category data for various services. While each service invocation leads to an update in the local monitor’s database, summary data for all services in a specific domain is reported to the central monitor periodically by each local monitor. One of the benefits of cloud computing is that there can be multiple options for services to achieve a specific task. We define a service category as an abstraction for a set of services that provide similar functionality. A service is the actual implementation of the functionality for a specific service category. The dynamic service composition module utilizes information from the central monitor’s database to create service orchestrations that comply with users’ performance and/or security requirements on-the-fly. The goal of dynamic service composition is to maximize the resiliency and trustworthiness of the system based on selecting the best individual services, while meeting the constraints (security and SLA requirements).
We performed experiments to evaluate the overhead of dynamic service composition using testbeds in the Amazon EC2 cloud. Note that the problem here is finding an optimal service composition (i.e. selecting a service from each service category required in the composition) subject to a set of QoS and security constraints. In the first experiment, we investigated the effect of the number of services to choose from for each service category, on the performance of dynamic service composition. In this experiment, we set the number of service categories to 5 and the number of QoS constraints to 3. Figure 10 shows the response time of the dynamic service composition module for scenarios with total number of services from 25 to 125. The results show that the execution time changes almost linearly. Even for 125 services in 5 categories (which is unlikely to be surpassed in any practical SOA scenario), the dynamic service composition module performs very well and the average response time is 22ms.

![Figure 10. Effect of number of services on dynamic service composition time](image)

In the second experiment, we investigated the effect of the number of service constraints on the performance of dynamic service composition module. In this experiment, we set the number of services to 50 and the number of service categories to 5. According to Figure 11, the effect of the QoS constraints on performance is sublinear. Even after increasing the input size by a factor of 5, the response time only increases by 50%.

![Fig 11. Effect of number of QoS constraints on dynamic service composition time](image)

In the third experiment, we investigated the effect of the number of service categories on the
performance of dynamic service composition. In this experiment, the number of services was set to 50 and set the number of constraints was set to 3. Similar to the previous results, based on Figure 12, the response times do not exceed 20ms even for 25 categories.

![Figure 12. Effect of number of service categories on dynamic service composition time](image)

<table>
<thead>
<tr>
<th>Service instance</th>
<th>c3.xlarge (4 vCPU, 7.5GB memory, SSD storage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client instance</td>
<td>t2.micro (1 vCPU, 1GB memory, EBS storage)</td>
</tr>
<tr>
<td>Operating system</td>
<td>Amazon Linux 2015.03 64-bit OS</td>
</tr>
<tr>
<td>Geographical region</td>
<td>US-w2 (Oregon region)</td>
</tr>
</tbody>
</table>

**Table 3: Amazon EC2 testbed characteristics**

To verify the scalability of the proposed AOP-based service monitoring solution, we conducted experiments with a testbed in the Amazon Elastic Compute Cloud (EC2). The details of the testbed are provided in Table 3.

For the performance measurement, we used the Apache Benchmark [32] tool to generate requests with varying levels of concurrency. In each run, we sent 1000 requests to the service under monitoring. The number of concurrent requests sent to the service was varied from 1 to 8. Therefore, if the level of concurrency is 4, the Apache Benchmark keeps 4 active concurrent requests at any given time during the runtime of the experiment. Once one of the requests is finished, it replaces it with a new request.

Figure 13 provides a comparison of the throughputs achieved under different concurrency levels under three different scenarios: A baseline with no security measure in the service domain, monitoring with AOP involving no policy enforcement, and monitoring of the service with policy enforcement using AOP. We observe that monitoring and policy enforcement slightly decreases the throughput for a concurrency level of 1, but the decrease in the throughput is much less noticeable for higher concurrency levels, demonstrating the efficiency of the monitoring solution.
Figure 13. Throughput of service monitoring for different concurrency levels

We further studied the distribution of response times in the testbed. The results for the concurrency-level 8 are presented in Figure 14. This figure shows that the difference between response times before the 80%-percentile is not significant. Therefore, less than 20% of requests are responsible for the most of the difference between average response times of all three scenarios. Interestingly, the major increase happens at the 10% (or even at the top 1%) of the requests, which mostly happen at the beginning of the experiment, where the AOP applies advanced bytecode rewriting and optimizations. The other root cause of this observation is a random fluctuation in the available computational resources and network delays in the cloud that happen every once in a while.

Figure 14. Percentage of requests served within a certain time with service monitoring
1.4.2 Implementation of Privacy-Preserving Data Dissemination with Active Bundles

Based on the showcase scenario created by Dr. Leon Li, NGC, we implemented a software prototype for Electronic Health Record (EHR) dissemination using active bundles in collaboration with the W3C group at MIT. The implementation is based on an extension of the active bundle prototype implemented in the previous phase of the project. This prototype for the WAXEDPRUNE project was demonstrated at NGC TechFest’2016.

The basic idea is that the data owner creates an EHR for a patient and stores the EHR in the form of an Active Bundle in the cloud, which hosts a hospital information system. Subjects with different roles are able to request access to the patient’s EHR. Access to data field(s) is given based on role-based access control policies as well as client’s attributes, i.e. security level of client’s browser, client’s authentication method (password-based vs. hardware-based vs. fingerprint), type of client’s device (Mobile or Desktop) and source network (corporate Intranet vs. unknown). Figure 15 provides a high-level visualization of the scenario.

![Diagram](image)

**Figure 15. EHR Dissemination in cloud (by Dr. Leon Li, NGC)**

Firstly, the client needs to select her role and create a data request (‘All’ or particular data attribute) on the hospital information system’s main page as demonstrated in Figure 16.
Then the client is redirected to the Authentication Server (AS), where she authenticates herself. For instance, in the password-based authentication scheme the client needs to enter her username and password as shown in Figure 17.

![Figure 17. Password-based authentication of a client at AS](image-url)
If the entered credentials are correct then the cryptographic capabilities of the browser are checked. If the level of cryptographic capabilities is high then the doctor will get the EHR of 3 patients (see Figure 18), whereas if the level of cryptographic capabilities is low then the doctor will get data on for only 1 patient (see Figure 19). The idea behind this is that if the level of cryptographic capabilities is high then the doctor can get access to data for patients assigned to other doctors, in addition to the data of a patient (Monica Latte) assigned to the given doctor.

![Figure 18. EHR retrieved for doctor with high level of browser’s crypto capabilities](image1)

![Figure 19. EHR retrieved for doctor with low level of browser’s crypto capabilities](image2)

Each party (corresponding to the Role) interacts with an Active Bundle (AB), running on a cloud platform, to access only those data from AB for which the party is authorized. If a client (e.g.
Doctor) clicks on a patient’s record, then detailed information for that patient is displayed (see Figure 20).

![Figure 20. Detailed EHR of a patient retrieved for doctor](image)

According to the specified Role-Based Access Control Policies, the doctor can get access to Contact, Medical and Billing Information of a Patient; **Insurance company** can get access to Contact and Billing information only (see Figure 21); and a **Researcher** can get access to Medical and Billing Information (see Figure 22).

![Figure 21. Patient’s data accessible by Insurance Company](image)
On the right side of the web pages a flow visualization panel is displayed, as seen in Figure 23. This shows the current phase of a data access process. Initially, the client’s authentication phase is in progress. Then, after the client enters correct credentials, data request is transferred to the Active Bundle and the ‘Policy application’ phase starts. Then role-based access control policies specified in the Active Bundle are applied and, finally, data for which the client is authorized (according to client’s role) are retrieved and sent to the client.
1.5 Distinctive Attributes, Advantages and Discriminators

The distinctive attributes of the cloud security/resiliency approach we proposed in this project are the following:

- **Policy-based and attribute-based data access authorization**: Applications have different security requirements in different context and an adaptable system is highly desirable. Existing SOA/cloud infrastructures lack mechanisms that can be used by clients to specify, communicate and dynamically enforce security policies for their data and service requests. The proposed approach closes this gap by giving to the client increased control over cross domain data sharing. Data owner can specify various security levels for data such as top secret, secret, confidential, unclassified and then specify role-based access control policies. Active Bundle can deploy different code for policy execution to provide data access to various types of applications. The proposed approach allows proactive treatment of potentially malicious data sharing with services, as opposed to the “after-the-fact” detection using audit logs. Also, to control data sharing our data dissemination model, in collaboration with W3C/MIT group, uses client’s attributes, i.e. security level of client’s browser, client’s authentication method (password-based vs. hardware-based vs. fingerprint), type of client’s device (Mobile or Desktop) and source network (corporate Intranet vs. unknown). Active Bundle is tamper-resistant: if malicious user tries to modify policies or AB code or bypass policy check then tamper attack will be detected and data access will be denied. Context-based and trust-based data dissemination are also supported by Active Bundles.

- **Resilient and adaptable service compositions**: Existing cloud enterprise systems lack robust mechanisms to monitor compliance of services with security and performance policies under changing contexts, and to ensure uninterrupted operation in case of failures. This work demonstrates that it is possible to enforce security and performance requirements of cloud enterprise systems even in the presence of anomalous behavior/attacks and failure of services. Service monitors include components that enable the adaptation of the systems in response to detected anomalies, such that the non-stop system operations continue and comply with security requirements. The resiliency is accomplished through dynamic reconfiguration and restoration of services. Our approach is complementary to functionality provided by log management tools such as Splunk in that it develops models that accurately analyze the log data gathered by such tools to immediately detect deviations from normal behavior and quickly respond to such anomalous behavior in order to provide increased automation of threat detection as well as resiliency. Our approach allows for proactive mitigation of threats and failures in cloud-based enterprise systems through active monitoring of the performance and behavior of services, promising achievement of resiliency and antifragility under various failures and attacks.

- **Unified model for agile computing**: The proposed approach offers a unified model for agile and resilient distributed computing, based on standardized technologies for monitoring and sharing of performance and threat data, promising for easy adoption in industry. The proposed performance and security policy enforcement model enables integration of various types of policies and optimization algorithms as well as filtering capabilities (e.g., high-quality vs. lower-quality data) for various data types, which is needed for fine-grain control over dissemination, searches, analytics, and operations in cross domains of privacy.
1.6 Tangible Assets Created by Project

The tangible assets created by this project include active bundle software, a tutorial describing the use of the developed software, as well as demos for secure data dissemination and dynamic service composition.

Our code repository for the “privacy-preserving data dissemination using active bundles” project is available at:
http://github.com/Denis-Ulybysh/absoa16

The tutorial explaining the use of the active bundle software is available at:

The following demo video has been created to demonstrate privacy-preserving data dissemination capabilities of the system:
https://www.dropbox.com/s/30scw1srqsmyq6d/BhargavaTeam_DemoVideo_Spring16.wmv?dl=0
https://www.youtube.com/watch?v=SIUupq5V6zk&feature=youtu.be

We have also created the following demo videos to demonstrate the capabilities of the adaptable service compositions and policy enforcement system for cloud enterprise agility:

- Introduction to the user interface and a composite web service for ticket reservations:
  https://www.youtube.com/watch?v=nucRjKZBtM

- Composite trust algorithms for cloud-based services:
  https://www.youtube.com/watch?v=6uHEfoxjEgs

- Trust update mechanisms:
  https://www.youtube.com/watch?v=xnm0-MzGBzw

- Enforcing client’s QoS and security policies:
  https://www.youtube.com/watch?v=ePtAM0N7jdY

- Service redirection policy:
  https://www.youtube.com/watch?v=e8xkCgZcQ1s

- Adaptable service composition:
  https://dl.dropboxusercontent.com/u/79651021/Adaptable_Service_Composition_Bhargava.wmv
1.7 Outreach Activities and Conferences

We have disseminated the results of the project with presentations at various conferences. The following list of publications resulted partially or wholly from the research work involved in this project:

- N. Ahmed and B. Bhargava. Disruption-Resilient Publish and Subscribe. 6th International Conference on Cloud Computing and Services Science (CLOSER'16), April 2016 (Best poster award).

1.8 Intellectual Property Accomplishments

None.

2 General Comments and Suggestions for Next Year

The TechFest is a great place to learn and develop ideas. More people should be encouraged to attend. Collaboration and visits with other NGCRC universities is a great step forward for the future. The coordinators at NGC are very dedicated.
References


