Privacy – Preserving Data Dissemination in Untrusted Cloud

Denis Ulybyshev, Bharat Bhargava, Miguel Villarreal-Vasquez, Aala Oqab Alsalem
Computer Science Department, CERIAS
Purdue University
West Lafayette, United States of America
dulybysh, bbshail, mvillar, alsalema@purdue.edu

Donald Steiner, Leon Li, Jason Kobes
Northrop Grumman
McLean, United States of America
Donald.Steiner, Leon.Li, Jason.Kobes@ngc.com

Harry Halpin
W3C / MIT
Boston, United States of America
hhalpin@w3.org

Rohit Ranchal
IBM Watson Health Cloud
Cambridge, United States of America
ranchal@us.ibm.com

***Abstract*—B2B (business-to-business) systems often use service-oriented architecture (SOA) with decomposed business services. These services can interact and share data among each other. Service might use a cloud – hosted database, such as a non - relational encrypted key – value store. However, the cloud platform hosting the database can be untrusted. Data owner needs to be sure that each service can access only those segments of a shared database for which the service is authorized. Furthermore, data requests can come from a service also hosted by untrusted cloud. Hence, there is a need for designing a cloud enterprise framework that can ensure privacy-preserving data dissemination in SOA and accurately detect data leakages. We design and prototype a solution that ensures privacy – preserving dissemination of data. The solution is based on (a) role-based access control, (b) cryptographic capabilities of client's browser, (c) authentication method, (d) subject's trust level. The prototype enables privacy – preserving dissemination of Electronic Health Records (EHRs) hosted in an untrusted cloud.**

Keywords—privacy; trust management; data dissemination; access control; SOA; database privacy; cloud computing

I. Introduction

Non-relational databases in the form of encrypted key-value pairs can be hosted by an untrusted cloud. Cloud platforms are vulnerable to large attack surface that could violate the privacy of data stored in cloud or shared with web services. Services can interact and share data with each other, including services from untrusted environments. The problem statement is to ensure for the data owner that each service can access only those data items for which the service is authorized. A mechanism which guarantees that unauthorized data accesses are denied is needed. In our approach we rely on an Active Bundle [5, 6] in order to store non-relational database in encrypted form. An Active Bundle (AB) is a self-protecting structure that consists of key-value pairs in encrypted form, access control policies and policy enforcement engine (Virtual Machine) [1]. The novelty of our approach is that, in addition to access control policies, used in role-based access control, our cloud data dissemination model depends on client's attributes. These attributes are:

1. Level of cryptographic capabilities of client's browser, which sends data request by means of https message.
2. Client's authentication method (password – based vs. hardware – based vs. fingerprint). Password – based authentication method is considered to be least secure.
3. Client's network (trusted vs. unknown network).
4. Type of the client's device (Mobile vs. Desktop).

Our approach, compared to Attribute-Based Encryption, has the following advantages: (a) it does not rely on Trusted Third Party (TTP) to issue keys for the recipient services; (b) it supports complex policies that can be written in Java language, whereas ABE policies are expressed as boolean and threshold operations over a set of attributes [1]. Such operations have limited ability to express access control policies.

II. Related Work

A mechanism of micro-policies [15] enforced at a browser's side was proposed to provide confidentiality and integrity of web sessions. The mechanism, implemented as a Google Chrome extension *Michrome* [16], can be used to ensure secure access to web data by means of http(s) protocol. Micro-policies are specified in terms of tags, used to label URLs, network connections, cookies, etc; and a transfer function, which monitors security-relevant operations based on these tags and defines which operations are permitted by the browser. In our approach, in contrast, the policies are enforced in the Active Bundle, by its policy enforcement engine. Our solution provides the following advantages: (1) role – based access control; (2) trust level of clients is constantly monitored and recalculated; (3) modification of browser's source code is not required.

A privacy – preserving information brokering (PPIB) system was proposed for secure sharing and information access via overlay network of brokers, coordinators, and a central authority [2]. This approach relies on centralized TTP to manage keys, metadata, joining and leaving brokers. Centralized TTP creates a single point of failure. Suggested methodology does not consider trust levels of services. [4]

Approved for public release by Northrop Grumman, Case #17-0995.

Pearson et al. proposed a solution for secure data dissemination when the recipients are not known in advance. '*EnCoRe'* project uses sticky policies to manage the privacy of shared data across different domains [3]. Data and policies are made inseparable. Sticky policies are enforced by a TTP and data dissemination provenance is supported. Sticky policies are prone to tamper attacks from malicious recipients and the approach itself is prone to TTP – related issues [4].

III. Core Design

A. Active Bundle

Our solution relies on Active Bundle (AB) [1] for secure data exchanges between services. Active Bundle is a self – protecting structure that incorporates sensitive data in encrypted form, access control policies and policy enforcement engine (Virtual Machine). Sensitive data is a non-relational database stored in the form of encrypted key-value pairs. Here is the example of key – value pair stored in the Active Bundle:

*{ "ab.patientID" : "***Enc***(0123456)" }*. Patient ID is 0123456 and it is stored in the Active Bundle in encrypted form. Each data item is encrypted with a separate symmetric key, which is generated on-the-fly based on execution flow. When service requests data from an Active Bundle, the identity of the requesting service requesting is verified, as a first interaction step. Authentication is based on signed digital certificates. The services present their X.509 certificates signed by a trusted Certificate Authority (CA) to the Active Bundle to verify their authenticity [1]. After service authenticates itself, its attributes (trust level, cryptographic capabilities of a browser, etc) and the context (e.g. emergency or attack) are evaluated and enforced by the policy enforcement engine embedded into an Active Bundle. Then evaluation of applicable access control policies determines what data can be disclosed to the requesting service. Symmetric decryption keys will be generated to decrypt those data items for which the authenticated service is authorized, based on access control policies, stored in the Active Bundle. Service requests keys from key – value pairs and corresponding values are decrypted, based on derived decryption keys. Decrypted values will be sent to the client using https protocol, provided client's trust level is sufficient, its browser's cryptographic capability level is sufficient and client's authentication method is secure. Symmetric key generation is based on the unique information generated in the execution control flow path of an Active Bundle [1]. This information depends on the Active Bundle modules and their resources: authentication code; subject's role (e.g. doctor, insurance agent, researcher), extracted from the X.509 certificate of the subject (service); authorization code; applicable access control policies and policy evaluation code. To ensure proper entropy in the key, hash of this information is transformed into a secret. This secret is then used to derive the symmetric key, using *SecretKeyFactory, PBEKeySpec* and *SecretKeySpec* methods from javax.crypto library. During AB creation, the data owner's policies are first embedded into the Active Bundle template, which is then executed to obtain the information and to derive the corresponding symmetric keys for each data item [1]. Decryption key derivation procedure is similar to encryption key derivation. Active Bundle execution control steps generate the information, based on the Active Bundle modules and their resources. Hash of this information is then used to derive the symmetric key, employing *SecretKeyFactory*, *PBEKeySpec* and *SecretKeySpec* methods. This symmetric key is able to decrypt the corresponding data item. Active Bundle is tamper-resistant so that modification of any items from the list below:

a) authentication code (when attacker try to bypass authentication phase); service certificate (when attacker tries to impersonate his identity or use wrong certificate);

b) authorization code (when attacker tries to bypass evaluation of access control policies);

c) applicable access control policies (when attacker tries to modify them to gain access to data he is not authorized for);

will result in digest change and lead to the incorrect decryption key derivation. Assumptions behind an "untrusted" cloud include direct access to confidential data and to encryption keys by the curious or malicious cloud provider. In our solution decryption keys are not stored neither on a cloud provider nor inside Active Bundle nor on TTP. Tamper – resistance mechanism provides data integrity and storing data in encrypted form provides data confidentiality in untrusted cloud. Active Bundle protects data communications between services from man-in-the-middle and masquerade attacks.

Active Bundle is written in Java and implemented as a Java Executable Archive (JAR). Access control policies can be specified either using Javascript Object Notation (JSON) [10] or Extensible Access Control Markup Language (XACML) [11] policy language. We use JSON-based policies since they impose less performance overhead, compared to XACML-based policies. WSO2 Balana [14] is used for policy evaluation.

Assumptions: Hardware on a site where AB is hosted / executed is trusted. OS kernel is trusted, as well. The services (e.g. Doctor, Insurance, Researcher) interact with the Active Bundle on a server side. For communication between all the web services https protocol is used. It provides protection against eavesdropping attacks.

In our implemented prototype, we provide secure dissemination of Electronic Health Records (EHRs), stored in the form of Active Bundles in untrusted cloud, one Active Bundle per one EHR. JSON [10] is used to store key – value pairs. Hospital Information System has all the EHRs and is hosted by a cloud provider. There are three clients: doctor, insurance and researcher. Access control policies specify that doctor can access medical data (test results, diagnosis, prescriptions, etc) of a patient, as well as contact and billing information. Table 1 shows access control policies for the Medical and Contact Information of a Patient, correspondingly.

TABLE 1. ACCESS CONTROL POLICY FOR MEDICAL DATA

ALLOW		
Resource	Patient's Medical Data	Patient's Contact Info
Subject's Role	Doctor, Researcher	Doctor, Insurance
Action	Read	Read

Insurance can access contact and billing information only, access to medical information is not allowed. Researcher can

only access anonymized records of patients, i.e. medical and billing information. Three services (Doctor, Researcher, Insurance), as well as Hospital service, that serves data requests, are running as NodeJS servers (daemons) at http://www.waxedprune.cs.purdue.edu:3000 on a cloud provider and are listening to the corresponding opened ports. Initial data request from unauthenticated client is redirected from Cloud Provider to Authentication Server (AS) where client needs to authenticate itself in order to receive a valid Ticket. Along with the authentication procedure, the level of cryptographic capabilities of client's browser and client's authentication method are determined and are included to the authentication ticket, which is signed by AS.

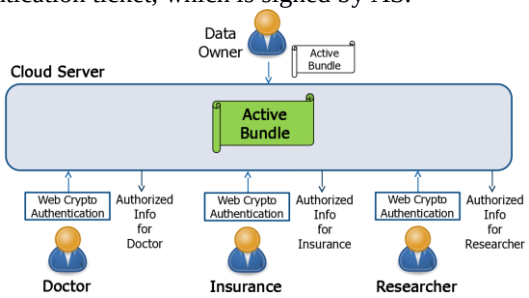

Fig. 1. EHR framework architecture (created by Dr. Leon Li, NGC)

If client authentication procedure is successful then AS redirects client's data item request to the proper service, corresponding to client's role, with the new valid Ticket. Once a corresponding service running in cloud receives data request and authentication ticket from the client, the ticket (signature, client ID, expiration time) and access control policies are evaluated, based on client's role. Based on evaluation of access control policies, of the client's browser cryptographic capabilities and of client's authentication method, Active Bundle responds to the requesting service with the authorized data. Then service transfers authorized data to the client. For a doctor, who logs in and requests for data from an insecure browser with WebCrypto not enabled (see Fig.2), the set of retrieved data is smaller than for the same doctor who logs in from a secure browser with WebCrypto enabled. Cryptographic capabilities, i.e. "WebCrypto enabled", assume existence and support of certain cryptographic libraries in the client's browser. The threshold of sufficient amount of cryptographic libraries supported by the browser, can be tuned, depending on the context. Demo video for our prototype is available [9].

B. Attribute-based Data Dissemination

In addition to access control policies, our selective data dissemination model is based on cryptographic capabilities of client's browser, requesting data from an Active Bundle; and on client's authentication method. Thus, if client requests data from an old browser with low level of cryptographic capabilities, then either limited or no data can be retrieved from an Active Bundle even if the client's authorization level and trust level are sufficient. The W3C Web Cryptography API [12] provides generic cross-browser access to cryptographic primitives such as AES and ECDSA in browsers. In our application, we are aiming at multi-level access where available services are selected based on the users attributes, including geolocation and the level of security given by their authentication using the browser. Once client opens the browser, the cryptographic capabilities of this browser are assessed by the server to determine their level of trust. Web Cryptography and future APIs that support advanced authenticators then is used to detect the capabilities of their device for authentication. Hardware and biometric support is in progress via the FIDO-based Web Authentication API and future work will continue to add hardware token-based authentication as soon as the W3C Web Authentication API [13] is deployed in modern browsers. Currently, the Web Cryptography API is supported by all modern browsers, but many users still use out-of-date browsers that are less trusted. If the Web Cryptography API is supported, a 'secure authentication' is enabled that uses a protocol beyond just user-names and passwords. In detail, the Web Cryptography API allows high-value authentication via Secure Remote Password (SRP), a zero-knowledge proof protocol that essentially is a form of password-based key derivation where the private key never leaves the client, and only a verifier database is required on the server (as given by IETF RFC 2945). Thus, if a server is compromised, the database of user passwords is still secure against popular "cloud-cracking" tools for password encryption using weak hashing algorithms such as MD5. This makes Secure Remote Password the most secure password – based authentication scheme available, and by taking advantage of WebCrypto's AES functions, we are able to make it much faster and more secure than a pure Javascript implementation. SRP also has the advantage as a key-based authentication protocol that key material stored on hardware tokens that can be accessed by future browser-based versions of WebCrypto could be easily incorporated into the key generation of SRP, or even replace the password component by deriving the initialization of the 'password' from secret key material in combination with the domain name of the origin. In general the user's (subject's) role and the cryptographic capabilities of their device in terms of the support of WebCrypto allow them to access different kinds of data, with more and sensitive data being visible only if the client device supports authentication using SRP.

IV. Evaluation

We evaluated performance overhead for service requesting data from an Active Bundle. Round-trip time (RTT) is measured between the moments when service issues data request and data retrieved from Active Bundle are received by the service. Thus, it includes authentication, authorization, key derivation and data disclosure phases. *ApacheBench, ver.2.3* utility is used for RTT measurements. Detailed configuration setup can be found in the prototype tutorial [8].

Experimental setup 1

Hardware: Intel Core i7, CPU 860 @2.8GHz x8, 8GB DRAM
OS: Linux Ubuntu 14.04.5, kernel 3.13.0-107-generic, 64 bit
Browser: Mozilla Firefox for Ubuntu, ver. 50.1.0

In the following experiment (see Fig.2), we use Active Bundle which, in addition to tamper – resistance, supports client's browser cryptographic capabilities and authentication method detection. Local request for Patient's Contact Information is sent to an Active Bundle, which represents EHR and runs on a Purdue University Server *waxedprune.cs.purdue.edu:3000*. We use an Active Bundle with 8 access control policies, similar in terms of complexity. Examples of access control policies are

given in table 1. Data request is issued from the service running on the same host with an Active Bundle. Thus, we measure RTT for a local data request to an Active Bundle. Network delays, that would be imposed if the request is issued from the remote service and that can affect the measurements, are excluded.

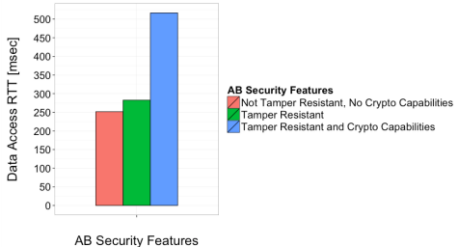

Fig.2. AB performance overhead with browser's crypto capabilities on / off

Tamper-resistance support adds 12.3% performance overhead since the hash value of an Active Bundle and its modules (code, access control policies, certificates) is verified by an Active Bundle when the data request comes. Support of client's browser cryptographic capabilities and authentication method detection, imposes additional performance overhead of 82.8%. RTT for data request increases because now, before responding to the client's request, Active Bundle needs to check the cryptographic capabilities of the browser and authentication method, which are sent to AB by means of https message.

Experimental setup 2: Google cloud Platform

OS Version: Linux Debian 3.16.39-1 (2016-12-30) x86 64 bit Hardware: n1-standard-1 (1 vCPU, 3.75 GB memory) CPU platform: Intel Sandy Bridge Zone us-central1-a External IP: ab-cloud-server (104.198.68.1)

In this experiment, request for Patient ID is sent to an Active Bundle, which represents EHR of a patient and runs on a Google cloud server. Google Cloud provider hosts a Hospital Information System, i.e. database of EHRs. Active Bundle with embedded tamper – resistance mechanism is evaluated. We vary number of access control policies included into Active Bundle, from 1 to 16. To make experiment reasonable, these policies are made similar in terms of complexity. Active Bundle runs on a Google Cloud Server, instead of Purdue Server.

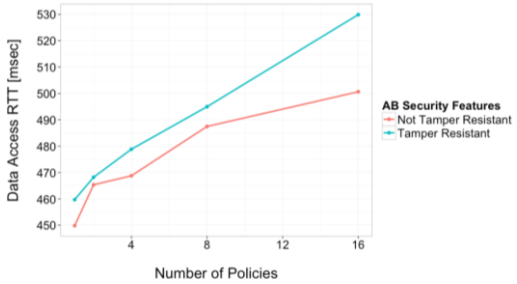

Fig.3. Performance overhead of AB, hosted by Google Cloud

Fig. 3 illustrates that RTT grows linearly. When the number of policies grows, the number of required checks before data can be retrieved, grows correspondingly. Support of tamper-resistance adds up to 5.8% (for 16 policies) performance overhead, depending on the number of policies, since the hash value of an Active Bundle and its modules need to be verified by an Active Bundle when the data request comes.

V. Conclusion

We presented a privacy – preserving dissemination model, that provides confidentiality and integrity for data, hosted in untrusted cloud. Here are the contributions of our approach:

1. It does not rely on TTP to issue secret keys (decryption keys) for the recipient services
2. Support of complex policies that can be written in Java [1]
3. It does not require data owner's availability
4. Data can be updated by multiple parties (services)
5. Dissemination depends on context and the client's attributes.

Acknowledgment

This work was funded by the Northrop Grumman Cybersecurity Research Consortium. The prototype was implemented in collaboration with Northrop Grumman and W3C / MIT and presented internally to Northrop Grumman in April, 2016. We are thankful to Prof. Leszek Lilien and Prof. Weichao Wang for their collaboration and valuable feedback.

References

[1] R. Ranchal, "Cross-domain data dissemination and policy enforcement," PhD Thesis, Purdue University, 2015

[2] F. Li, B. Luo, P. Liu, D. Lee, and C.-H. Chu, "Enforcing secure and privacy- preserving information brokering in distributed information sharing," IEEE Transactions on Information Forensics and Security, vol. 8, no. 6, pp. 888–900, 2013.

[3] S. Pearson and M. C. Mont, "Sticky policies: an approach for managing privacy across multiple parties," IEEE Computer, no. 9, pp. 60–68, 2011.

[4] B. Bhargava, "Privacy – preserving data dissemination and adaptable service composition in trusted and untrusted cloud," NGCRC Project Proposal, CERIAS, Purdue University, Aug.2015

[5] L. Ben Othmane and L. Lilien, "Protecting privacy in sensitive data dissemination with active bundles," 7-th Annual Conf. on Privacy, Security and Trust (PST 2009), Saint John, New Brunswick, Canada, Aug. 2009, pp. 202-213

[6] L. Lilien and B. Bhargava, "A scheme for privacy-preserving data dissemination," IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 36(3), May 2006, pp. 503-506.

[7] B. Bhargava, "Secure/resilient systems and data dissemination/provenance," NGCRC Project Proposal, CERIAS, Purdue University, Aug.2016

[8] D. Ulybyshev, B. Bhargava, L. Li, J. Kobes, D. Steiner, H. Halpin, B.An, M.Villarreal, R.Ranchal, T.Vincent, "Secure dissemination of EHR in untrusted cloud," Project Tutorial, Purdue University, 2016.

[9] D. Ulybyshev, B.Bhargava, "Secure dissemination of EHR," demo video https://www.dropbox.com/s/30scw1srqsmyq6d/BhargavaTeam_DemoVideo_Spring16.wmv?dl=0 , accessed: Feb.2017

[10] "Lightweight data-interchange format JSON," http://json.org/ , accessed: Oct.2016

[11] "eXtensible access control markup language (XACML) version 3.0," http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html, accessed: Oct. 2016

[12] "W3C Web Cryptography API," https://www.w3.org/TR/WebCryptoAPI/ , accessed: Oct.2016

[13] "Web authentication: an API for accessing scoped credentials," http://www.w3.org/TR/webauthn, accessed: Oct.2016

[14] "WSO2 Balana Implementation," https://github.com/wso2/balana , accessed: Oct.2016

[15] S. Calzavara, R. Focardi, N. Grimm and M. Maffei, "Micro-policies for web session security". *Computer Security Foundations Symp. (CSF), 2016 IEEE 29th* (pp. 179-193), June, 2016

[16] Anonymus, "Micro-policies for web session security," 2016, available at https://sites.google.com/site/micropolwebsese, accessed: Feb.2017