

ABLS - An Attribute Based Logging System for the Cloud

Christopher A. Wood
Department of Computer Science
caw4567@rit.edu

ABSTRACT

User-based non-repudiation is a system security property that provides indisputable evidence that links specific actions to individual users (or entities) that trigger such actions. Cryptographically speaking, non-repudiation requires that the integrity and origin of all data should be provable. In essence, this enables system audits to be conducted that can identify data misuse (and thus, potential security policy violations) by comparing the sources of system events with all entities authorized to invoke these events. Therefore, treating non-repudiation as a required system quality attribute in the SaaS architecture is likely to become a common trend in the commercial, government, and even more specifically, the health-care domain.

System audits typically use system log files to determine the cause and effect of events that took place during the system's lifetime. In order to provide accurate information for non-repudiation purposes, it is often necessary to place some amount user-sensitive data in these log files that can be used to trace data back to its origin. As such, log data generated by a client that is being served must maintain data confidentiality and integrity should the system be compromised. These goals are commonly achieved using a combination of encryption and signature techniques. However, traditional approaches to encryption and signature generation and verification are no longer applicable in the context of cloud systems. Furthermore, naive approaches to log security that are based on tamper-resistant hardware and maintaining continuous secure communication channels between a log aggregator and end user are no longer useful in the context of cloud-based applications.

Symmetric-key and public-key encryption of log entries have been the most common confidentiality techniques proposed in the literature. However, in cloud systems, these schemes are becoming out-dated technologies. There is a need for robust access policy control so that user revocation and addition can be done on the fly with minimal overhead (i.e. re-encrypting a subset of the log database should be

avoided). Both symmetric- and public-key cryptosystems lack in that access policies must be tied directly to keys used for encryption and decryption. If the access policy for a set of log messages needs to be changed, then both the keys used to encrypt and decrypt such log entries will need to be regenerated and distributed, and the entries must also be re-encrypted. Both of these can be very expensive tasks.

In addition, symmetric-key cryptosystems require keys to be shared among users who need access to the same set of logs, which requires a secure and comprehensive key management and distribution policy to be in place. Enforcement of access policies to certain log entries may require re-keying and encryption of an entire set of log files in the database if a user's access is revoked. From a storage perspective, public-key cryptosystems (e.g. RSA) suffer from the extra data transfer and storage requirements for large cryptographic keys. It may be too expensive to maintain a public-key infrastructure (PKI) for managing keys and digital certificates for all users.

In terms of log file integrity, aggregate signature schemes that support forward secrecy through the use of symmetric- and public-key cryptosystems are also becoming outdated. Symmetric-key schemes may promote high computational efficiency for signature generation, but they do not directly support public verifiability for administrators and auditors. This means that robust key distribution schemes or the introduction of a trusted third party (TTP) are needed to ensure all required parties can access the necessary log information. Such schemes also suffer from high storage requirements and communication overhead. Public-key schemes have similar issues, as the increased key size leads to even larger storage requirements and less computational efficiency.

From this we see that a balance between encryption and signature generation and verification performance is needed to support the unique scalability and resource usage requirements for cloud applications. Attribute-based encryption (ABE), a new cryptographic scheme that uses user attributes (or roles, in certain circumstances) to maintain the confidentiality of user-sensitive data, has an appealing application to logging systems in the cloud for a variety of reasons.

In this project we will continue the work on ABLS, an attribute-based logging system that supports ciphertext-policy attribute-based encryption (CP-ABE) [1] and authenticated hash-chain constructions for log file confidentiality and integrity. The preliminary design of ABLS was weak with regards to the scalability of encryption operations under heavy traffic loads, the relational schema to store log-data, and the

interactions with database servers. To address these issues, we will modify ABLS in the following ways:

1. The CP-ABE encryption scheme will be replaced with a hybrid cryptosystem in which the Advanced Encryption Standard (AES), the standardized symmetric-key encryption algorithm, is used to encrypt user session data by a key that is protected with CP-ABE encryption. In the event that a user generates log messages with varying sensitivity levels during a single session, multiple symmetric keys will be produced to maintain the confidentiality of information in different security classes.
2. The relational database storage scheme will either be changed to include a masking column in the appropriate log table to hide user identities or replaced entirely with a document- or object-based database. Both of these modern database systems are similar in semantics (if documents are conceptually treated as serialized objects), so a product and literature survey will be conducted prior to selecting an adequate replacement that satisfies the necessary security and performance requirements. Though, based on preferences and architectural experience, a document-based database such as MongoDB will be the likely candidate. Also, since the ABLS architecture is highly structured around a relational schema to store data, this change will require modifications made in the logging, attribute authority, and auditing modules in order to maintain functional correctness.
3. Database security mechanisms, including client authentication to support granular access policies for pro-

tected databases, will be implemented. This is particularly important for the databases that store cryptographic keys. Since preference is given to document-based databases such as MongoDB, third-party services such as zNcrypt (provided by Gazzang) that are specifically tailored to this DBMS will be used to enforce access control to sensitive databases.

4. The auditing module will be extended to include audits of the database operation and the data contained therein. Currently, the auditing module is only designed (and partially implemented) to support strategy-based audits on database contents. However, in this type of application, database performance and security are almost as critical. Thus, with this change, there will be two audit techniques with that can roughly be equated to data and policy audits, both of which will only be accessible to clients with the appropriate privileges. This access control will be likely be enforced using username and password credentials.

With these changes, we will re-evaluate the ABLS system at an architectural, security, and performance perspective to determine its usefulness in cloud-based settings.

TODO: security features to be explored in the analysis!

1. REFERENCES

- [1] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP '07, pages 321–334, Washington, DC, USA, 2007. IEEE Computer Society.