Blockchain and Git Repositories for Sticky Policies Protected OOXML

Grzegorz Spyra
The Cyber Academy
Edinburgh Napier University
Edinburgh, UK
g.spyra@napier.ac.uk

Prof William J Buchanan
The Cyber Academy
Edinburgh Napier University
Edinburgh, UK
w.buchanan@napier.ac.uk

Dr Elias Ekonomou
The Cyber Academy
Edinburgh Napier University
Edinburgh, UK
e.ekonomou@napier.ac.uk

***Abstract*—The paper discusses possible cloud-based Information Rights Management (IRM) model extension with enhanced accountability for both a sticky policy and an attached data. This work compliments research on secure data sharing with Office Open XML (OOXML) package extended by a sticky policy in eXtensible Access Control Mark-up Language (XACML) format. Research used Identity Based Encryption (IBE) primitive to securely bind the policy and the data together. High availability required from cloud service is here achieved using distributed system components. The Git repository and the Blockchain, leveraged technologies are not new, however their application for IRM system is novel and brings it closer to universal approach and open architectural construct.**

Keywords—Sticky policies; git repositories; Blockchain

I. Introduction

This work is focused on a sticky policy protected data access accountability and non-repudiation. With emerging cloud technology, data, once released into the cloud, may be stored in several locations, across several legal jurisdictions and can be hosted by various cloud service providers, and that not necessarily share the consistent information required to take valid access control decisions. Sectors such as healthcare, governments and finance seek for secure cloud systems, where not only data confidentiality could be protected but also contains information on authorship [1]. These new extended data boundaries require new security safeguards i.e. data accountability and auditing that currently are adapted from legacy on-premises environments.

Secure Patient Health Record (PHR) architecture consists of historical fingerprint [2]. When such record or its part leaves a medical system boundary it should be accompanied by well-defined security components to ensure access control, integrity and non-repudiation in semi-trusted or non-trusted environments. Unfortunately neither homogeneous systems nor data sharing standard exists that guarantees such safeguards across different security boundaries.

Companies and health institutions processing, storing and sharing sensitive information like PHR run various data protection programmes aiming to prevent data leakage. Information Rights Management (IRM) implementation is one of the technical safeguards helping to protect documents. Where information integrity and changes history are required to comply with data protection directives this solution delivers a core functionality for IRM system.

This work contributes by adding a new simple module and it compliments other works on sensitive documents sharing where information authenticity and non-repudiation are ensured within a single document boundary by recording all changes made upon commit operation of a new version.

II. Background

Office Open XML (OOXML) as XACML policy wrapper is a ZIP package file consisting of one or more file sections followed by a central directory. The main document part is defined by multiple XML document elements. Each file section consists of an actual embedded file, and a local metadata file that includes information such as a filename, a file directory, a timestamp, compression used and a data descriptor that includes a valid file checksum. Most of OOXML internal sections could be protected by built-in OOXML encryption. However there are sections that are not covered by a native OOXML cryptographic techniques [3]. An XACML policy is added as an additional package content that remains in unencrypted XML format. This policy defines access rules over resource and implements Attribute-based Access Control (ABAC) with attribute values defining legitimate data processing subject, Role-based Access Control (RBAC) [4] where business or institutional roles define who can access the data or finally Risk-Aware Access Control (RAAC) expressions [5], the most dynamic access control technique making access decisions upon dynamically calculated risk.

OASIS empowered XACML with health-care system authentication architectures [6] and defined entities, i.e. Access Control Service (ACS) responsible for taking access control decisions. Proposed here model integrates existing architectures with Identity Provider and Identity Based Encryption (IBE) key generator. The IBE as a preferred encryption method leverages XACML policy as an encryption key that attached to the OOXML package remains in plain text and follows the package ensuring data confidentiality prior to successful data access authorization.

III. OOXML Accountability and Auditing

Limited sources define possible non-repudiation assurance applied to an Information Rights Management (IRM) system where OOXML is used as a protected rich data wrapper. Furthermore if this is simultaneously processed, several different versions of the same distributed data may become available on the cloud. While the OOXML standard built-in functionality delivers integrity with well-defined granular digital signature model [7] it does not deliver information non-repudiation other than one used for revision tracking [8]. While the revision tracking could be leveraged to keep an audit record on changes it was not designed for such a purpose. The entire OOXML data-sharing scheme has features enabling it for changes non-repudiation and auditing and with properly configured editor application the required safeguard could be secured.

Non-repudiation could be secured by introducing data versioning for properly hosted and referenced data. Where non-repudiation of all the changes is not required data versioning functionality without historical versions will still ensure non-repudiation of the latest registered version. However a single chain of data versioning is must to consistently maintain data shared in the cloud. Where data storage cost has to be considered, OOXML internal structure allows package decomposition and only changed elements extraction. In this way rich text element, i.e. media files that may not change as often as the text are not becoming redundant and unnecessarily stored all over the chain of changes. However, the last approach requires highly trusted service provider as none of the historical changes could be stored in encrypted form, considering the fact that each historical revision or version requires different encryption key derived from policy therefore generates different cipher text.

IV. XACML Accountability and Auditing

Furthermore not only data but also access policy may require accountability allowing incident identification showing when, how and by whom the initial data owner access rights were tampered or simply legitimately changed. XACML policy could be signed, however it does not guarantee non-repudiation and does not provide any historical information [9].

Same functionality used for OOXML could be leveraged for XACML policy as XACML data incorporated as a part of OOXML package inherits security safeguards from its wrapper.

There are various functional disadvantages of such a safeguard where entire sticky policies implementation may loose its flexibility and increase maintenance costs. However where thorough accountability is required Trust Authority (TA) or data owner may decide to enforce higher security policies.

V. Merkle Trees Applications

There are various applications leveraging Merkle Trees construct designed to ensure distributed data or database integrity like in [10], although block-chain and git repositories, the most popular, have the required functionality as well as many implementations available already as a cloud-based service.

Consistent OOXML data versioning requires a single globally available chain of all the changes. Document changes have to be consistent and relate only to one previous version. Users should not be able to simultaneously commit same version updates with two different contents.

Merkle Trees allow quick and efficient verification of data and its version in large data structures. Hashing, a cryptographic primitive ensures the integrity of the current and the preceding tree leaf.

A. Data Versioning with Blockchain

Blockchain maintains one central chain of all the transactions. The single chain normally consists of the latest blockchain hash. XACML policy instance and OOXML package versions can be located on a centralized blockchain what guarantees document integrity. Data editor who wishes to commit a new version has to ensure that the version committed is a direct ascendant from the latest committed version. In the case where new version from a different version ancestor has to be committed to the chain, a new transaction for version cancelation has to be added to the blockchain by authorized actor. Classical blockchain implementation maintains basic transaction meta-data unlike Git repositories, where actually entire data history is stored. Excluding consensus available in blockchain the both are very similar.

B. Changes History via Git Repository

In Git everyone may have several branches ascendant from the same data. Consequently everyone could commit the latest version in a chain by resolving conflicts with latest committed version. Unlike in blockchain, in Git the content matters regardless of the branch while in blockchain the final consensus matters regardless of the content. Entire OOXML package and XACML policy history can be stored and hosted simultaneously using single Git repository. Package data could be either stored in unencrypted format, what has many functional features compared to single branch consisting only encrypted versions.

VI. Conclusions

Where only policy and data versioning has to be maintained the blockchain technology is sufficient, however where non-repudiation and changed history is required a constrained Git repository have to be used. Basic evaluation of Git repository for described purpose shows that encrypted OOXML data and unencrypted OOXML data require more storage and cannot respond to diff requests compare to unpacked and unencrypted OOXML data. Additionally, a constrained built-in OOXML revision control could deliver granular changes information required for high accountability although it requires custom OOXML editor implementation. With both Git repository and OOXML revision control model could provide revision history via Git required for data maintenance and non-repudiation via amended internal OOXML revision control functionality.

Automated Git repository evaluation was completed using default version control repository configuration. The Blockchain evaluation for versioning control with a single version revocation upon author request is not ready and requires components development on client and server side.

References

[1] D. Shackleford, “Orchestrating Security in the Cloud,” 2015.

[2] D. Daglish and N. Archer, “Electronic personal health record systems: A brief review of privacy, security, and architectural issues,” in CONGRESS 2009 - 2009 World Congress on Privacy, Security, Trust

and the Management of e-Business, 2009, pp. 110–120.

[3] S. L. Garfinkel and J. J. Migletz, "New XML-Based Files Implications for Forensics," IEEE Secur. Priv., vol. 7, no. 2, pp. 38–44, 2009.

[4] Soceanu, M. Vasylenko, A. Egner, and T. Muntean, "Managing the Privacy and Security of eHealth Data," in 2015 20th International Conference on Control Systems and Computer Science, 2015, pp. 439–446.

[5] L. Gasparini, "XACML and Risk-Aware Access Control," Aberdeen, 2013.

[6] Mohammad Jafari and D. DeCouteau, "Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of SAML v2 . 0 for Healthcare Version 2 . 0 Committee Specification Draft 01 /," 2014.

[7] Apple, Barclays Capital, BP, The British Library, Essilor, Intel, Microsoft, NextPage, Novell, Statoil, Toshiba, and the United States Library of Congress, "Information technology — Document description and processing languages — Office Open XML File Formats — Part 2: Open Packaging Conventions," no. December. ISO/IEC, Geneva, p. 138, 2006.

[8] Apple, Barclays Capital, BP, The British Library, Essilor, Intel, Microsoft, NextPage, Novell, Statoil, Toshiba, and The United States Library of Congress, "Information technology — Document description and processing languages — Office Open XML File Formats —Part 1: Fundamentals and Markup Language Reference," vol. 2012. ISO/IEC, Geneva, p. 5030, 2012.

[9] Saldhana, A. Tappetla, A. Anderson, A. Nadalin, B. Parducci, C. Forster, D. Arumugam, C. David, S. Dilli, D. DeCouteau, E. Rissanen, G. Richards, H. Lockhart, J. Herrmann, J. Tolbert, L. Seitz, M. Kudo, N. Itoi, P. Tyson, P. Mishra, R. Levinson, R. Jacobson, S. Proctor, S. Muppidi, T. Moses, and V. Murdoch, "eXtensible Access Control Markup Language (XACML) Version 3.0," 2013.

[10] N. Parab and A. Brown, "Cloud Storage Using Merkle Trees," 20160110261, 2016.