EMBEDDED DOCUMENT SECURITY USING STICKY POLICIES AND IDENTITY BASED ENCRYPTION

GRZEGORZ KAROL SPYRA

A THESIS IS SUBMITTED AS PARTIAL FULFILMENT OF THE REQUIREMENTS OF EDINBURGH NAPIER UNIVERSITY, FOR THE AWARD OF DOCTOR OF PHILOSOPHY IN THE CENTRE FOR DISTRIBUTED COMPUTING, NETWORKS, AND SECURITY

JANUARY 2019

ACKNOWLEDGEMENTS

Mojej żonie i dzieciom…

Prof Bill Buchanan you let me do research around all what’s written here and a wee bit more.

Declarations

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

CONTENT

FIGURES

TABLES

Appendixes

GLOSSARY OF ACRONYMS

Term	Definition	Term	Definition
AAA	authentication, authorisation, and accounting	KP-ABE	key-policy attribute-based encryption
AAL	authenticator assurance level	LDAP	lightweight directory access protocol
ABAC	attribute-based access control	LiBAC	lightweight break-glass access control system
ABE	attribute-based encryption	MA-ABE	multiple-authority attribute-based encryption
ACS	access control service	MAC	mandatory access control
AD DS	Microsoft Active Directory Domain Services	MFA	multi-factor authentication
AE	authenticated encryption	MHIBE	multiple hierarchical identity-based encryption
AEAD	authenticated encryption with authenticated data	MS	Microsoft
AES	Advanced Encryption Standard	NISC	National Institute of Standards and Technology
AES-CBC	Advanced Encryption Standard Cipher Block Chaining	Non-DAC	Non discretionary access control model
AES-CTR	Advanced Encryption Standard Counter Mode	NTRU	a cryptosystem that utilises lattice theory; acronym not clearly defined
AES-GCM	Advanced Encryption Standard Galois Counter Mode	OASIS	Organisation for the Advancement of Structured Information Standards
AIP	allowable intended purpose	OAuth	open authentication
API	application programming interfaces	OECD	Organisation for Economic Co-operation and Development
APP	application component	OLE	object linking and embedding
ARBAC 97	administrative RBAC '97	OOXML	Office Open XML format
ASL	algorithm security lifetime	OpenID	open identity
AuthN	authentication	OWL	Web ontology language
AuthZ	authorization	PBAC	purpose-based access control
B2B	business to business	PDF	portable document format
B2C	business to customer	PDP	policy decision point
BDHP	bilinear Diffie Hellman problem	PEP	policy enforcement point
BYOK	bring your own key	PFS	perfect forward secrecy
C-RBAC	cryptographic RBAC	PHR	personal health record
CA	certificate authority	PII	personal identifiable information
CIP	conditional intended purpose	PIP	prohibited intended purpose
CNG	Microsoft Cryptography Next Generation	PIP	policy information point
COM	Component Object Model	PKC	public key certificates
CP-ABE	ciphertext-policy attribute-based encryption	PKI	public key infrastructures
CRUD	Create, Read, Update, Delete event	PKG	private key generator
CSP	credential service provider in identity assurance context; otherwise cloud service provider	QI	quasi-identifier
CSP	cloud service provider	QRNG	quantum number generator
CSU	cloud service user	RAAC	risk-aware access control
CSV	comma-separated values format	RBAC	role-based access control
DAC	discretionary access control	RDF	resource description framework
DES	data encryption standard	RDFS	resource description framework schema
DH	Diffie-Hellman algorithm	RFID	radio-frequency identification
DLP	discrete logarithm problem; but not in data loss prevention context	RMS	Rights Management Services
DNA	deoxyribonucleic acid	RNA	ribonucleic acid
DRM	digital rights management	RNG	random number generator
ECC	elliptic-curve cryptography	RP	relying party
ECS	enterprise cloud subscriber	RSA	Rivest–Shamir–Adleman public key scheme
EHR	electronic health record	SAML	security assertion markup language
FAL	federated assurance level	SARBAC	scoped administrative role-based access control model
FIM	federated identity management	SCIM	cross-domain identity management
FISA	Foreign Intelligence Surveillance Act	SDK	software development kit
fs-HIBE	forward-secure hierarchical identity-based encryption	SFA	single-factor authentication
GDPR	The General Data Protection Regulation from EU	SMS	short message service
GP	general medical practitioner	SOA	service-oriented architecture
HIBE	hierarchical identity-based encryption	SOAP	simple object access protocol
HIBS	hierarchical identity-based signing	SP	service provider
HSM	hardware security module	SPIBE	sticky policy identity-based access control
HTTP	HyperText Transfer Protocol	SPML	service provisioning markup language
IAL	identity assurance level	SSO	single sign-on
IAM	identity and access management	TA	trust authority
IBC	identity-based cryptography	TCP	Transmission Control Protocol
IBE	identity-based encryption	TPA	third party auditor
IBE-BF	identity-based encryption from Dan Boneh and Matthew K. Franklin	UDP	User Datagram Protocol
IBE-IPG	identity-based encryption over isogenous pairing groups	UWD	Universal Windows Driver
IBS	identity-based signing	VBA	Visual Basic for Applications
ID2020	identity in 2020; ID2020 is a nonprofit public-private partnership committed to improving lives through digital identity	W3C	World Wide Web Consortium
IdP	identity provider	WWW	World Wide Web
IPS	isogenous pairing groups	X500	directory management protocol standard
IRM	information rights management	X509	public key certificates format
JSON	JavaScript Object Notation format	XACML	eXtensible Access Control Markup Language
KMS	key management system	XML	eXtensible Markup Language

ABSTRACT

Data sharing domains have expanded over several, both trusted and insecure environments. At the same time, the data security boundaries have shrunk from internal network perimeters down to a single identity and a piece of information. Since new EU GDPR regulations, the personally identifiable information sharing requires data governance in favour of a data subject. Existing enterprise grade IRM solutions fail to follow open standards and lack of data sharing frameworks that could efficiently integrate with existing identity management and authentication infrastructures. IRM services that stood against cloud demands often offer a very limited access control functionality allowing an individual to store a document online giving a read or read-write permission to other individual identified by email address. Unfortunately, such limited information sharing controls are often introduced as the only safeguards in large enterprises, health-care institutions and other organizations that should provide the highest possible personal data protection standards.

The IRM suffers from a systems architecture vulnerability where IRM application installed on a semi-trusted client truly only guarantees none or full access enforcement. Since no single authority is contacted to verify each committed change the adversary having an advantage of possessing data-encrypting and key-encrypting keys could change and re-encrypt the amended content despite that read only access has been granted. Finally, the two evaluated IRM products, have either the algorithm security lifecycle (ASL) relatively short to protect the shared data, or the solution construct highly restrained secure key-encrypting key distribution and exposes a symmetric data-encrypting key over the network. Presented here sticky policy with identity-based encryption (SPIBE) solution was designed for secure cloud data sharing. SPIBE challenges are to deliver simple standardized construct that would easily integrate with popular OOXML-like document formats and provide simple access rights enforcement over protected content. It leverages a sticky policy construct using XACML access policy language to express access conditions across different cloud data sharing boundaries. XACML is a cloud-ready standard designed for a global multi-jurisdictional use. Unlike other raw ABAC implementations, the XACML offers a standardised schema and authorisation protocols hence it simplifies interoperability. The IBE is a cryptographic scheme protecting the shared document using an identified policy as an asymmetric key-encrypting a symmetric data-encrypting key. Unlike ciphertext-policy attribute-based access control (CP-ABE), the SPIBE policy contains not only access preferences but

global document identifier and unique version identifier what makes each policy uniquely identifiable in relation to the protected document. In IBE scheme the public key-encrypting key is known and could be shared between the parties although the data-encrypting key is never sent over the network. Finally, the SPIBE as a framework should have a potential to protect data in case of new threats where ASL of a used cryptographic primitive is too short, when algorithm should be replaced with a new updated cryptographic primitive. The IBE like a cryptographic protocol could be implemented with different cryptographic primitives. The identity-based encryption over isogenous pairing groups (IBE-IPG) is a post-quantum ready construct that leverages the initial IBE Boneh-Franklin (IBE-BF) approach. Existing IBE implementations could be updated to IBE-IPG without major system amendments. Finally, by applying the one document versioning blockchain-like construct could verify changes authenticity and approve only legitimate document updates, where other IRM solutions fail to operate delivering the one single authority for non-repudiation and authenticity assurance.

1 Introduction

Just in the last couple of years, people experienced transformation from the analogue into the digital world. This is not only a beginning of a new technological phase, it has started the new digital era. Today, the international ID2020 alliance plans to deliver one global digital identity to all living people, including over one billion people who have never been officially registered. Global digitalisation brings new digital threats with an impact never seen on such a scale.

Reading news, we could observe a fascinating phenomenon, enormously rapid digital body growth driven by demands for more functionality, size, speed, and features. This new living organism is not alone and requires intelligence to protect itself from the surrounding environment. Without the immunity, it will experience more severe damages. The recent cyber-attack at German government systems [1] caused not only exposure of sensitive intelligence information but also weakened the trust citizens have for the institution that supposed to protect them. Furthermore, we see that trusted channels providing decent security boundary fail in trying to assure the data protection. Information lost its protection shield since it entered into a shared cloud space. It is not about network firewalls anymore, which like veins could keep the bit of information within a closed stream. In recent years information security has rapidly changed from a network being a security boundary to an identified information as the only true security boundary that has left. In 2017 one of the most significant cyber-attacks compromised the security of the major credit reporting agency in the United States [2] causing personal data leakage of over 140 million citizens. Just a few months later, the largest Swiss telecommunication company reported one million Swiss citizen records being stolen [3]. In both cases, attackers easily managed to take an advantage over the security of the system. Hardly any attacks on such a scale used sophisticated cryptographic techniques to gain an advantage within a protected system.

1.1 Author Publications

G. Spyra, "Next Generation Authentication Infrastructures With Role Based Security For Cloud Computing," Edinburgh Napier University, 2012.

G. Spyra, W. J. Prof Buchanan, P. Cruickshank, and D. E. Ekonomou, "Cloud-Based Identity and Identity Meta-Data: Secure and Control of Data in Globalization Era," Int. J. Reliab. Qual. E-Healthcare, vol. 3, no. 1, pp. 49–66, 2014.

G. Spyra, P. W. J. Buchanan, and D. E. Ekonomou, “Sticky policy enabled authenticated OOXML for Health Care,” in BCS Health Informatics Scotland Research Conference 2015, 2015, pp. 1–6.

G. Spyra, W. J. Buchanan, and E. Ekonomou, “Sticky policy enabled authenticated OOXML,” in SAI Computing Conference 2016, 2016.

G. Spyra and W. J. Buchanan, “Protecting Documents with Sticky Policies and Identity-based Encryption,” in 2016 Future Technologies Conference (FTC), 2017, no. January, pp. 1–5.

G. Spyra, W. J. Buchanan, and E. Ekonomou, “Blockchain and Git repositories for Sticky Policies protected OOXML,” in 2017 Future Technologies Conference (FTC), 2017, no. November, pp. 29–31.

G. Spyra, W. J. Buchanan, and E. Ekonomou, “Sticky policies approach within cloud computing,” Computers & Security, pp. 1–9, 2017.

1.2 Aim and objectives

This thesis aims to deliver a model for secure information sharing in the cloud, leveraging common information rights management (IRM) architectures and modern cryptographic and access control techniques.

The main research questions to answer in this thesis are:

- How to build a secure cloud-ready file sharing framework using sticky policy?
- Is there existing standard, mature and secure components ready to construct such a solution?
- What are the challenges for secure cloud information sharing especially for organizations such as health-care, governments and large enterprises?
- How the proposed sticky policy with identity-based encryption (SPIBE) construct could cope with secure cloud information sharing challenges in comparison to other IRM solutions?

The research first has to highlight current security related problems when it comes to data sharing, new data protection regulations and latest global technical trends. Crucial part to justify the selected components of SPIBE framework are existing already in use authentication and authorization standards. To synthetize the model design and produce evaluation with other existing IRM solutions research explains common cryptographic protocols and related security safeguards including advanced key management.

Work critically asses existing popular IRM solutions to identify potential weak points, architectural gaps and challenges related to cloud computing and quantum technology.

1.3 Contribution and novelty

This work provides an overview of secure models, which combined, could give a solid foundation for a new reliable cloud-based secure data-sharing model. Many identities and personal identifiable information (PII) data hosting models recognise security issues that come with Cloud computing. Researchers try to propose methods to enforce accountability over data stored in the cloud shared space [4]. Many focus on data protection in the health-care context and propose to find the most suitable encryption and access control model, along with delivering a mature framework for Cloud-based implementations [5], [6]. Various information rights management (IRM) systems deliver solutions based on symmetric and both symmetric and asymmetric cryptography [7], [8] or [9].

While symmetric key-based solutions suffer from difficulty to exchange keys in distributed implementations [8], the combined symmetric and asymmetric solutions suffer from a low public key algorithm security lifecycle (ASL) [7], [9].

Proposed model, sticky policy with identity-based encryption (SPIBE) uses unique document identifier together with policy to identify and protect OOXML document. Unlike typical IBE construct [10] this work delivers identity from a sticky policy protecting the data like in [11]. Access policy is formatted with eXtensible Access Control Markup Language (XACML) standard, which highly constrains the access attributes, rules and conditions. In compare with non-standardized attribute-based access control (ABAC) models the SPIBE is simpler to use and to implement due to many existing XACML schemas. While the ciphertext-policy attribute-based encryption (CP-ABE) schema [12] extends traditional key-policy attribute-based encryption (KP-ABE) [13] over access subject context, the SPIBE leverages concept of both, where global document identifier and its version together with role-based access control (RBAC) like rules are combined into a policy. Cryptographic construct currently uses AES symmetric encryption to protect the data and asymmetric identity-based encryption over isogenous pairing groups (IBE-IPG) [14] to encrypt the symmetric data-encrypting key. SPIBE contributes significantly by aligning existing IRM models with new data protection demands and newly developed security techniques.

1.4 Background

1.4.1 Data Loss Prevention (DLP)

A major problem of data protection within the Cloud is that a data, which seems to be local and personal is in fact unencrypted once stored in the Cloud. Sooner or later this data might become a subject of direct or indirect processing by third parties. Lack of encryption mechanisms used to protect the information and the global character of the Cloud causes data leaks without data owner control [4]. Even encrypted data is still exposed to several data leakage risks because the encryption techniques cannot compete with increasing processing power [15]. This new processing power is needed to effectively encrypt live data in memory generate, release and revoke encryption keys to achieve a truly protected personal data in the Cloud [16].

Data loss prevention (DLP) is a common data protection programme more or less successfully adapted by many organisations and corporations. Since the first OECD recommendation on data protection [17] the data loss has been not only a concern for financial institutions taking higher precautions worrying to lose its reputation but also for companies having anything to do with a personal data. Data loss or leakage occurs when data is no longer under control of the responsible data custodian. Data loss could be an effect of an external or an insider attack. Unfortunately often we see it is negligence at the highest corporate C-level [2] [3] that allowed an adversary to take advantage of unprotected systems. Data loss, as with other security incidents, could cost organisations a signification amount of money due to indirect loses like reputation, intellectual property, trade secrets or more direct like stolen bank accounts or credit cards details and crypto-currency units. Now financial losses will be also be relate to penalties that come into force on 25 May 2018 following the new Regulation (EU) 2016/680 of the European Parliament and the EU Council [18]. Furthermore, approved in June 2018, a new Assembly Bill 375 in California takes effect in January 2020 [19] will make companies that fail to follow the new privacy prior to data leakage incident liable for high civil penalties.

Recently the Massachusetts Institute of Technology referred to existing safeguards, i.e. cryptography, as insufficient to protect data [15], when it comes to personal data shared in a cloud by governments, medical institutions and others, actually disqualifies any encryption algorithms used. However, it does not encourage to protect data and to seek for modern information protection countermeasures.

1.4.2 Sticky Policies

Sticky policies group rules defining who, when, where and how can access the data. Unlike other access policy models, policies are bound to the data piece. This access control model could secure personal identifiable information (PII) with high accountability. Each personal data access attempt is a subject of extensive auditing where any security breach or a data leakage incident is reported by sticky policies framework. Detected incidents could be tracked and give solid evidence leading to legal consequences, e.g. where data is not properly governed under GDPR regulations.

The data owners can feel owning the data released into the Cloud. Policies associated with the data are protecting and enforcing the data owner consent. E.g. sticky policy added to medical report about a patient by data owner would cover a data owner consent and define any subject rights to process that data.

Data owner preferences regarding trust authority (TA) selection for policy management give possibility to choose only providers that ensure certification for all independent trusted parties behind TA, i.e. authorization authority and private key generator (PKG).

1.4.3 Identity

Identity can be seen as the unique information sufficient to perform operations on objects. The subject, owner of the digital identity, could be a living person or any digital actor that could initiate a digital transaction. Information related to an identity could be seen as a set of attributes. Attributes could be static in some identity models but can also vary depending on claims by specific service providers (SPs) [20]. Digital identity has to uniquely identify the subject in some contexts. However, not all contexts require unique identification [21]. Most common attributes used for identity identification are email address [22], international phone numbers [23], in some countries national identification number or national insurance number. Some governments and institutions conducted research on RFID implants to identify a living person uniquely.

Technologically advanced biometric measurements could deliver physical characteristics vector representing unique, immutable identity attributes [24]. The last could potentially replace all existing identity attributes required to initiate identification and authentication. Considering non-living subjects, the identity attributes could be constructed from legacy hardware numbering [25], [26]. However, identifying attributes of a device could be successfully derived similar to a living person biometrics vector, from a hardware physical characteristics vector [27]. The process where the subject claims a digital identity is called authentication (AuthN). Authentication verifies various authenticators required to complete a digital identity claim. Subject during authentication attempts to access an authentication service that is in control of authentication technologies. Digital authentication presents subject with a challenge to solve relevant to security boundaries and access context of the digital identity. The authentication process could use both physical and digital channels. The subject can fulfil authentication claims in person or over the network. Authenticators are also connected with all required entities over the network, very often this is an open Internet network.

1.4.4 Health-care

In healthcare, medical organisations store and process mostly sensitive personal information, and also need persistent access to a sensitive data [28] to save their patients life at any time, without technological and jurisdictional constraints [18]. Unfortunately, access to such a data is mostly restricted to one institution or very often a single building. Currently, healthcare services are still concentrated around medical institutions rather than the patient [29]. Furthermore, most of the legacy systems suffer from lack of standardisation, therefore, are neither ready for a global integration nor for full personal health record (PHR) and electronic health record (EHR) enablement.

Even when personal data is stored and processed within secure and well-defined boundaries, problems can arise because there is no oversight by the data subject (i.e. the patient). There are strong indications that PHR owners would also like to have full access to their information [30] and also to be able to control the rights of access to the records. PHRs and even more so the EHR, which aggregate them require a platform that will allow secure data exchange [31] preserving privacy across Cloud-based systems [32]. Similar problems can be found in educational institutions where pupil [33] or student information cannot be shared due to legal and technological limitations, despite the data subject's expectations.

Several works [34], [35] aim to deliver a unified model, which can be adapted under several security contexts like health-care, education institutions, enterprises and others. Here is worth mentioning the Microsoft HealthVault development project, which defines, in detail, several XML schemas ready to adapt in medical institutions. The World Wide Web Consortium (W3C) aims to address quality of XML language and related schemas what includes delivery of basic secure standards that guarantee integrity and confidentiality of information represented as an XML-structured piece of data. The W3C has created the W3C XML Signature Working Group focused on digital signatures and W3C XML Encryption Working Group specialised in encrypted content. The main technological problem with adequate data protection is the efficiency of encryption algorithms when the encrypted information requires e.g. indexing [4]. Several methods can be used in parallel for encryption to effectively index unencrypted XML data [36], although implementation would require further techniques to protect the health-care related part of personal identifiable information (PII) data which is an integral part of PHR and EHR data.

1.5 Thesis structure

The following chapters cover:

- Literature Review. This includes research background presenting the current situation demanding for secure cloud data sharing. It covers existing frameworks and open standards that integrate with the proposed Sticky Policies with Identity-Based Encryption (SPIBE) model. It covers current research around cryptographic primitives that are ready for modern computing.

- Sticky Policies Approach within Cloud Computing. This chapter covers the main contribution of this work shows how the proposed model protects, encrypts and decrypts data protected under SPIBE construct.

- Evaluation Methodology and Implementation. This covers actual evaluation that has been performed to support SPIBE model, recognise its advantages and disadvantages in comparison to other existing IRM solutions.

- Conclusions and Future work. Finally, the concluding chapter summarises the research and evaluation showing missing parts and space for further development and SPIBE model improvements.

The thesis also contains appendixes which contain not only the code used for evaluation but also the Visual Studio solutions as some of the C++ projects require specific configurations to be hosted on the MS Windows platform.

2 Literature Review

2.1 Introduction

This chapter provides the fundamentals for sticky policy with identity-based encryption (SPIBE) model evaluation. It should justify different standards and technologies used giving the reader a complete context. Starting from explaining the need for standardised identity, through secure authentication the model could finally authorise an individual to access the protected data. Protection requires access level countermeasures such as relevant access control model, access control expression language (i.e. XACML) and finally a cryptographic primitive that should address the actual demand for flexible and secure schemes. The final piece, the OOXML document format should introduce the reader to one but not the only possible SPIBE integration scenario. This chapter also discusses the IRM weaknesses and proposes potential architectural improvements by introducing single trusted blockchain tracking all legitimate information changes.

2.2 General Data Protection Regulation (GDPR)

European Union states agreed on adapting directives giving every person a right to the protection of own personal data [18]. Set of new regulations applied to all Member States aims to respond to new challenges brought by rapid sociotechnical development. Amount of collected personal data together with technological advance has exposed new threats. New scientific fields like the data science allow global processing of all sensitive data types, never available at this scale before. Both a private sector and public authorities could use that data for their advantage. Recent events show how new EU regulations may improve the legal perception of the personal data. British political consulting firm Strategic Communication Laboratories (SCL Group) / Cambridge Analytica illegally used personal data of millions of Facebook users [37]. Furthermore, the same company consulted current United States president Donald Trump during the presidential election. Facebook owner Mark Zuckerberg confirmed [38] that the data has been sold to Cambridge Analytica knowing how it will be processed. In the face of the different charges effectively at 1st of May 2018 SCL Group closed operations also having new EU regulations hanging above as Damocles' sword. GDPR highly constrains that freedom of uncontrolled data processing in favour of a natural person's freedom. The regulation also highly constrains personal data processing techniques such as profiling, where personal preferences, behaviours and attitudes are analysed.

The regulation requires the strengthening of the data subject rights as well as data recipient obligations. The data recipient means a natural, a legal person or another body to whom the personal data has been disclosed including legal authorities responsible for monitoring and compliance with data protection rules. Regulations ease data processing although not except regulation compliance for scientific research on personal data. Data subjects should give their explicit consent allowing data processing to be completed, but only to some extent. This includes deoxyribonucleic acid (DNA) or ribonucleic acid (RNA) analysis, which is also defined as personal data processing. Data processing obligations are tightened when data relates in any way to a child, person of age below 16. This, as with adult person personal data, applies to any information that relates directly or indirectly to a subject. Any data processing not excluding marketing purposes, creating a personality or user profile is strictly regulated and requires a consent given or authorised by the holder of parental responsibility for the child [18]. Furthermore, such consent has to be explicitly verified by the controller.

GDPR ensures and protects freedom in regard to religion, ethnic origin, political opinions, sexual orientation, philosophical beliefs and genetic data i.e. also historical ancestry data that could identify an individual. Any data processing that could reveal such information threatens these freedoms and is strictly prohibited by the regulation [18].

Regarding security countermeasures, the regulations give strong due diligence safeguards protecting the data at rest. Cryptographic algorithms security lifecycle makes any encrypted data vulnerable to tomorrow's technology. Unlike any other sensitive or confidential data, the personal data should be protected during time exceeding algorithm security lifecycle. While protection of critical financial report or country tactical, or strategic information, is necessary mostly within a relatively short period, i.e. days, months or decades but the personal data protection is not limited to any period. The regulation, although, does not apply to deceased persons, and the historical data processing such as ancestry analysis if related to a living person [18] might fall within the scope of the regulation.

2.3 IRM

Information rights management (IRM) the document protection related domain of wider in scope digital rights management (DRM). DRM aims to protect digital intellectual property such as movies, audio, patents, documents and all types of multimedia where legal protection require technological safeguards. In other words, it is a collection of hardware, software, services, and technologies that have been developed for persistently governing the authorized distribution and use of content and services according to their associated rights and managing consequences of that distribution and use throughout their entire lifecycle or workflow [39]. The IRM refers to safeguards for digital information i.e. digital documents that are either at rest or in transit. Digital documents could consist of actual documents like Office Open XML (OOXML) or Portable Document Format (PDF) package encapsulating multimedia content. The package is defined as a rich content combined into one single file. Depends on the context the digital information could also refer to an image, a simple text file, an email or even a short message service (SMS) message.

IRM prevents protected information package from being printed, forwarded, saved, edited or copied without prior authorisation, ensuring information confidentiality, integrity, and non-repudiation. The greatest challenge for IRM systems is its interoperability across different platforms with different editing applications. As with DRM, missing standards are allowing decent security enforcement for digital data protection [39]. Existing products are either very homogeneous hence too hermetic for common use or are highly heterogeneous and are often user-friendly. However, offering an apparent acceptable level of data protection.

IRM systems enforce data owner access preferences, where owner decides who, under what conditions and how could process the data. Access preferences follow the digital data that moves across different security boundaries. Such assigned rights could constrain access using discretionary access control (DAC) model, where data access is governed at all times by the data owner. Access could also be defined under mandatory access control (MAC) specifying document classification. Here, protected information is addressed to a group of individuals having sufficient level of privileges or the right clearance level that authorises the access. Finally, the access could be defined under non-discretionary access control (Non-DAC) model like role-based access control (RBAC) where the document is addressed to a group of people with a specific role within an organisation. For example, patient's medical examination results initially have the patient information assigned (i.e. data subject), together with laboratory access preferences (i.e. data owner) and general practitioner (GP) role, giving GP access to read and provide own comments about the medical test results. Data owner has full rights over the report, but data subject who has the full legal rights over that data within the IRM system, the data subject could only read the report. The general practitioner, although, has rights to read and write the report i.e. complete a diagnosis as the report is addressed for further medical analysis.

2.4 Identity management

Information does not stay within single corporate boundaries, but large, middle and small companies and organisations federate to collaborate efficiently. Still, digital identity moving across boundaries becomes often exposed to a threat of identity impersonation where the illegitimate subject becomes in control of a digital identity. It is important to design identity management systems following the latest recommendations for cryptographic techniques, authentication flows, and authentication protocols. To overcome new data protection challenges, there are existing assurance levels related to digital identity protection. The digital identity could be measured against several identity assurance levels [21].

There are two levels of assurance for non-federated identities the Identity Assurance Level (IAL) and the Authenticator Assurance Level (AAL). IAL consists of three levels, **IAL1** where digital identity does not have to identify a living person uniquely. Any attributes verification during authentication has only technical context without any actual relation to real unique identity identification. **IAL2** requires verification that digital identity is uniquely related to a real person identity. Identification of the identity provided during initial registration either delivered in person or remotely is a must. Either a credential service provider (CSP) or a relying party (RP) could assert attributes delivered as a claim for pseudonymous identity. **IAL3** requires that individual proofs identity in person upon registration. Identity meta-data attributes have to be verified by authorised and trained CSP representative. For pseudonymous identity, attributes could be asserted via CSP or RP. The AAL has three distinct assurance levels for authentication procedures. With **AAL1** a claimant needs an authenticator that is bound to the digital identity account [21]. Both single-factor authentication (SFA) or multi-factor authentication (MFA) techniques could be used.

To complete the authentication the claimant via secure authentication protocol proves he is a legitimate owner of the authenticator. **AAL2** proves that the claimant controls two distinct authenticators using MFA with a minimum of two distinct factors. Verification requires secure authentication protocols. All cryptographic techniques used at this level and above have to be approved. Unlike lower AAL levels the **AAL3** requires from the claimant a proof, using cryptographic protocols, of a possession of a key. A hardware authenticator and only approved cryptographic techniques could be involved at this level. Authentication methods used have to preclude any possible identity impersonation. Furthermore, **AAL3** requires minimum two distinct authentication factors via secure authentication protocols. For federated identities, there is a separate measurement component called Federated Assurance Level (FAL) [21]. In **FAL1** the relying party (RP) is allowed to receive the bearer assertion signed by the identity provider (IdP) using only approved cryptography. **FAL2** level allows RP to receive signed and encrypted bearer assertion from IdP. Only RP should have the possibility to decrypt the assertion. The highest possible assurance level for federated identity is the **FAL3**. It requires on top of the lower assurance level safeguards that a digital identity owner the subscriber proofs of possession of a cryptographic key referenced in the assertion.

2.4.1 SCIM Schema

System for cross-domain identity management (SCIM) was created [40] in order to standardise the digital identity management including all the processes related to identity provisioning, meta-directory provisioning, identification, authentication, and federation. The rapid development of global cloud-based systems requires a completely new approach to identity management. Legacy identity management systems are monolithic often based on X500 directories [41] and directory metadata [42], where schema evolved to store not only identity-related information but configuration metadata from various applications and systems. Furthermore, lack of consistency in legacy implementations makes cloud data and information sharing a difficult challenge. Identity-related attributes are not following a single standard; hence any cross-enterprise identity federation requires a non-standard approach. Identity provisioning was partially standardised with service provisioning markup language (SPML) [43]. However, it was never easily adaptable due to different identity management data schemes. The new SCIM approach highly simplifies the identity management including cross-domain identity provisioning due to standardised skimmed schema [44]. SCIM defines a schema for different resource types, i.e. users, groups, configuration, and so on.

For each resource type, it defines a standard attributes set, e.g. *userName*, *password*, *userType, timeZone* [44], which represent flat and non-complex attributes or constructed, complex or multi-valued attributes such as *name*, *groups*, *x509certificates*. Each standard attribute defined within SCIM schema has its type (e.g., *string*, *dateTime*), cardinality (e.g. *singular*, *complex*), mutability (e.g. *readWrite*, *immutable* or *writeOnly*), uniqueness (i.e. *none*, *global* or *server*), case-exactness (e.g. *caseExact* is equal to *false* when exactness is case-insensitive) and returnability (e.g. *password* attribute has returnability equal *never*, however, *id* attribute is returned during each request due to returnability equal to *always*) [44]. Identity management schema following these provisioning baselines is ready for global identity federations. This schema constrains existing systems giving developers some guidelines regarding a standard set of attributes required to handle security enabled resources like users and groups. SCIM also provides security best practices and recommendations for sensitive security data (e.g. *password* attribute) protection also in the context of other authentication and authorisation security recommendations [45].

SCIM defines a provisioning schema and a provisioning protocol [46]. It is another successful standard (see Figure 1) after Service Provisioning Markup Language (SPML) [43].

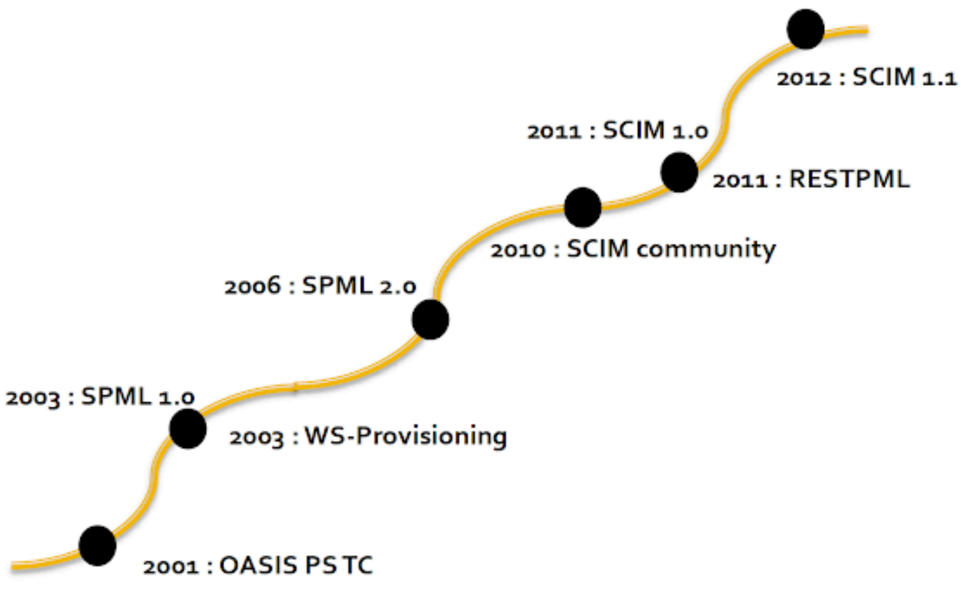


Figure 1 Brief history of Identity Provisioning [47]

This protocol operates at the HTTP application layer executing HTTP methods that represent CRUD (*Create*, *Read*, *Update*, *Delete*) activities exchanged between different parties [40]. SCIM 1.1 defines a cloud service provider (CSP), enterprise cloud subscriber (ECS) and cloud service user (CSU) as three distinct acting parties, actors that take part in protocol flow.

Only well-designed identity and access management (IAM) system can ensure secure access to the data. Empowered with SCIM schema existing enterprise systems could federate and define common data access policies understandable by all parties.

Consistently provisioned and revoked identities across all interconnected environments highly increase the data security. Standardised single SCIM schema helps to build global heterogeneous IAM solutions with all heterogeneous systems advantageous.

2.5 Identity metadata

Data held by the identity provider (IdP) can most often be classified as either personally identifiable information (PII) [48] or personal data. PII includes home addresses, social security numbers or maiden name, what is enough to allow unique identification of an individual. Depending on the authentication architecture supported by the IdP, the PII metadata is exchanged as claimed verified attributes [30] between IdP and SP.

Identity verification methods require a relevant level of assurance [21] with effective safeguards against unauthorised PII data divulgence. Secure cloud-based identity with corresponding related personal data requires a secure model that supports the personal responsibility of the object (i.e. data stored in the cloud) owner over their digital identity and its authenticity.

While performing research around identity and identity metadata, it is crucial to search for the possibility of separating PII data from identity itself as well as securely joining the shared data with identity using unique anonymous or ephemeral links. In modern identity management implementations, where identity consists of an obfuscated unique identifier [4] with policies required for further authorisation, only an obfuscated piece of information [46], should be exchanged during service provider (SP) authentication claim. Authentication that involves PII data flow needs to face a lack of single globally established security baselines and certification for SPs and authentication, authorisation, and accounting (AAA) mechanisms [49]. On the other hand, the model itself is still vulnerable to more sophisticated attacks such as data inference [50].

Identity accessing object can act as a subject, depending on the activity context as shown in Figure 2. At the same time personal identity (user) may act as an object when the subject requires further information about the personal identity metadata. Furthermore, the SP and the cloud-based service require an identity (user). All access control actors require identity and identity metadata to be properly accounted for performed activities in the shared cloud space. With a generic identity and access management framework, all activities of the subject over an access object are logged for further legal audits. The technology can benefit from a single, secure model where each entity of access control operation is equally accountable, as an identity instance inherits a generic schema whether it is a real person or an automated robot.

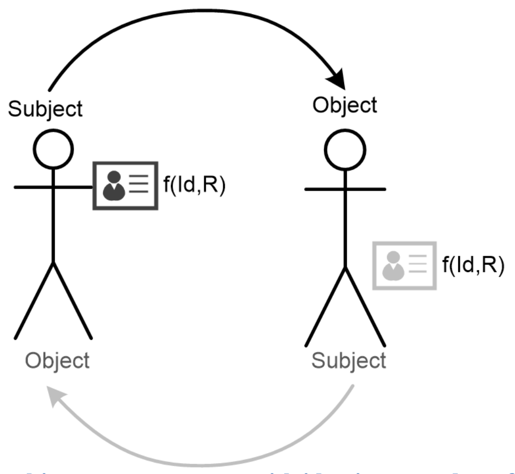


Figure 2 Two different subjects access contexts with identity meta-data; f – function matching identity Id with its rights R in given context

2.5.1 Identity Subject, Data Object and Predicate

Despite the discussed security control identity always remains in the centre. Over the years very static and naive account-based access management evolved into a global problem, where legacy security boundaries based on network perimeters were extended into a public space called the cloud. Account-based security became identity-based security, with a boundary that cannot be protected by legacy safeguards. To justify this statement, one simply has to look at the data sharing context. A subject (see Figure 3) within a strictly defined network environment does not access a data document. However, an object passes several different network and security boundaries before the subject also defined within a different context (i.e. medical, governments, enterprises or organisations) attempts to access it.

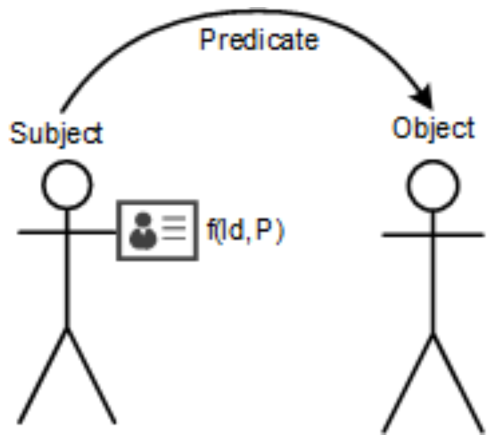


Figure 3 Access tuple with a subject (Identity), an object (data) and a predicate (permission).

Publicly exposed data requires several safeguards, and, in this field, research work related to purpose-based access control models play an important role as they aim to fill an existing gap [51], [52] not addressed with any legacy access control models (see Section

2.7). Currently in Europe OECD conducts the most crucial non-technically related work, which addresses legal aspects of data security, [53]. It aims to deliver legal frameworks to ensure data protection and address privacy concerns related to cloud-based computing era.

Data structure and access management model both play a crucial role in building a secure cloud-based data sharing framework [54]. Sticky-policies one of the most promising access control models effectively enforces owner rights assignments over the owned object. Recently the National Institute of Standards and Technology (NISC) published their access control framework called Policy Machine [55], which was one of the deliverables of Next Generation Access Control project and was proposed as a cloud-based implementation [56]. Its major architecture components include core access policy elements, assignments and relations definition, obligations and finally access request decisions. Single policy framework can not only enforce access control policies comprehensively across distributed and centralized operating environments, but also comprise aspects involving the characterization, distribution, and control of implemented capabilities. Policy Machine aims to dramatically simplify administrative effort, policy enforcement, data interoperability, and usability challenges faced by every large organization and enterprise today [55].

2.5.2 Secure Distribution of Identity Metadata

Because identity must securely span across different boundaries, it should provide cross-platform integration abilities and capacity for identity and access management (IAM) and federated identity management (FIM). Although, this is not the core part of the SPBIE framework it has to be integrated if the solution should work on a global scale. There are existing methods of secure data distribution, and also there are technologies that compound together can be used to deliver an integrated identity meta-data framework.

Considering XML as a standard for identity meta-data the XML schema would require a technique, which allows different parties to share an XML schema for a particular type of content that is attached to the main identity. As identity meta-data content will be spanned across different systems, it has to share the core identity element to maintain its unique reference to a single person, while also maintaining several different schemas for contextual interpretation. In this way, a person only needs to share a core identity element that subjects can use for self-identification in the process of accessing objects. For example, a patient registering for private medical treatment would not have to allow the medical institution to store own PII information but would grant access by reference to own identity metadata that is hosted centrally by legal institutions. The core identity ontology should not only identify but also represent identity access and operations entitlements in the cloud for various services enabled for different access control models.

Encryption with other safeguards delivers the security crucial for distributed identity metadata. Personal data before it is hosted by any cloud-based service should be encrypted by default. Furthermore, the encryption of identity metadata should be required for every single XML node [57], as they form a sensitive part of hierarchical identity metadata. Digital signatures should be used to ensure the authenticity of XML ontology definition and to ensure the access control granularity [58].

2.5.3 XML Schemas and Ontologies

One such XML-based ontology is the Web Ontology Language (OWL), which was designed to define the semantics of the relationships between entities. OWL defines what is semantically correct in XML, and both deliver data framework for the Web as well as for Cloud-based systems. OWL has been successfully used for access control systems implementation [59] as well as with encrypted distributed XML content [57]. There exists an older alternative approach for delivering structures, the resource description framework (RDF) together with RDF Schema (RDFS) defining classes for RDF. Although, while it seems to be easier to adapt, semantic limitations mean RDF may not satisfy all cloud-based identity metadata framework requirements, although it can be successfully combined with an OWL in some implementations scenarios and suffice its constraints [60].

Using OWL ontology [57], a single ontology is defined by a class, a sub-class, properties, and relationships. There is also a possibility to define OWL class relationships, where different ontologies can share a common parent. Within OWL it is possible to define the ontology for our identity metadata, which would allow identity to refer to other identities with simple predicate definitions (see Figure 4). The identity meta-data, here as a compound XML model defined under several ontologies, can be spanned across several contexts (see Figure 5). In other words, different parts of personal data can be stored and processed by different organisations. XML parts can be defined under various ontologies with different OWL-defined, XML schemas. A model where different parts of identity metadata are distributed to different service providers enables it to make use of a range of existing XML schemas, for instance, Microsoft HealthVault XML. Only such an approach can guarantee that data access to personal information can be distributed and efficiently maintained by different cloud service providers (CSP).

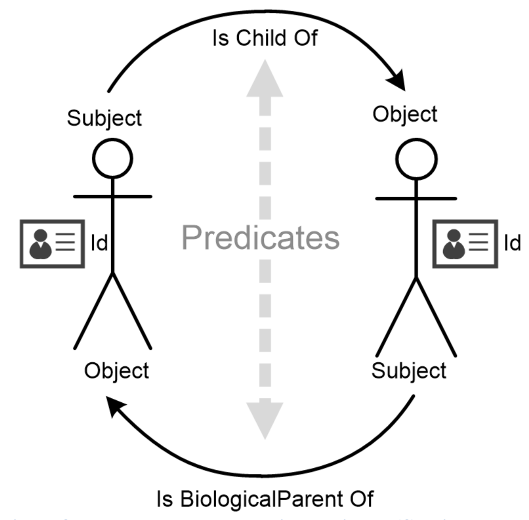


Figure 4 Two subjects in reference to each other in a triplet (Subject – Predicate – Object) represent linked identities

In the health-care sector, the identity metadata would align with the EHR concept, where a range of different and possibly competing health-care repositories can hold patient information [31]. Secure access to distributed data is possible thanks to one identity and identity metadata framework. Securely linked identity metadata is an assurance of data integrity and authenticity, which is what is required from new cloud-based systems.

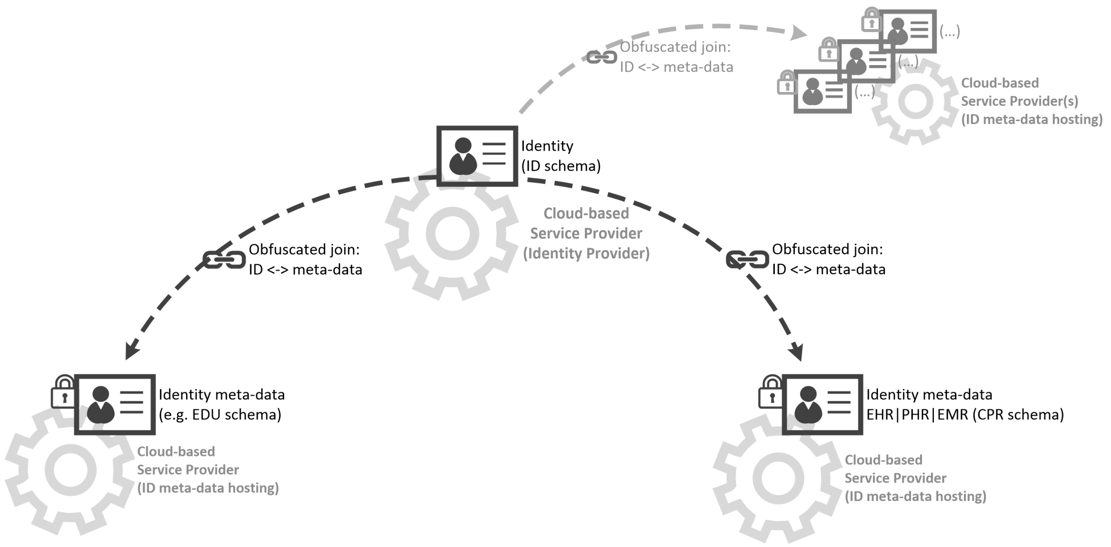


Figure 5 Linked identity with identity meta-data using obfuscated references across several security boundaries and contexts

Distributed XML identity metadata parts need to be linked to refer only to one identity. This secure XML linking defines that part of the identity metadata that belongs only to one identity. Without security, such linking is highly vulnerable to several types of attack including impersonation attack and man-in-the-middle attack. In response, the identity-based encryption (IBE) model has been successfully utilised to create a secure dynamic reference for hierarchical data structures as discussed later in this work.

In summary, individual parts of the identity metadata share a common ontology designed to support secure links. A mandatory obfuscated link is maintained from the main identity

XML to subparts of the identity metadata and back from identity metadata to the main identity. Requests from separate parts of identity metadata could be hosted in a service-oriented architecture (SOA) implementation as shown in Figure 6. Web service(s) exposed as part of dedicated cloud-based services could process distributed requests using encryption, obfuscation and anonymisation (see also Figure 7, which includes further SOA implementation details). Next, each cloud-based service could effectively support such distributed XML model with effective XML clusters [61], where a single XML document can be partitioned into several clusters.

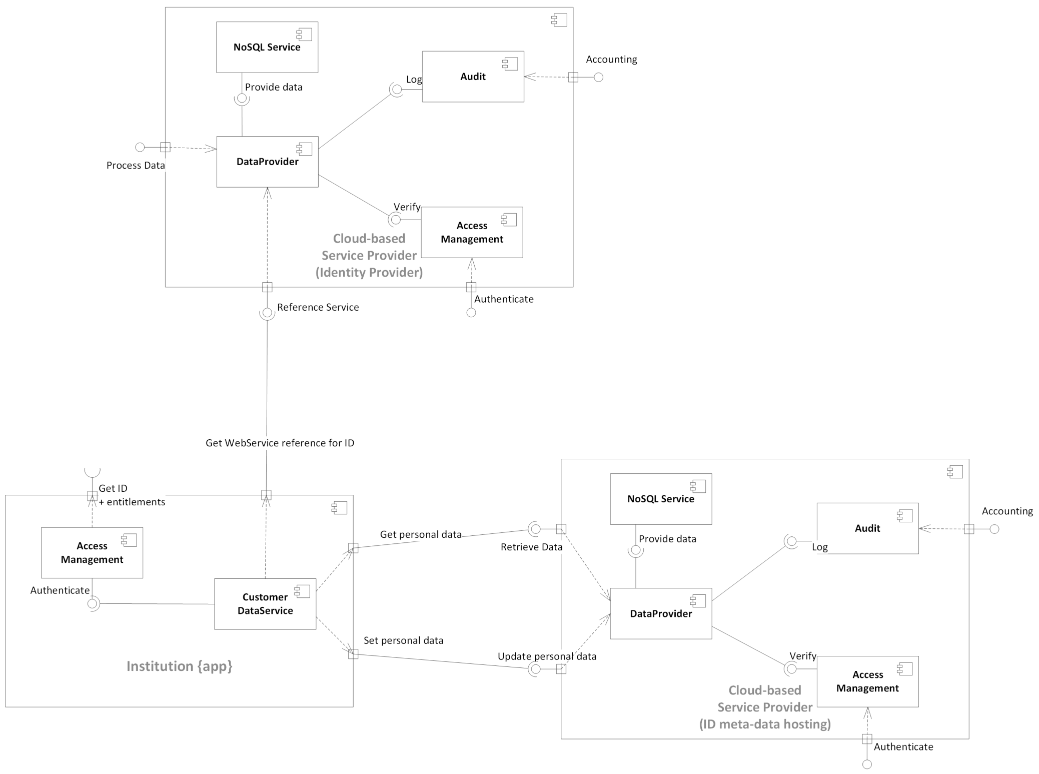


Figure 6 System architecture: calling identity meta-data by obfuscated reference under SOA (possible use case)

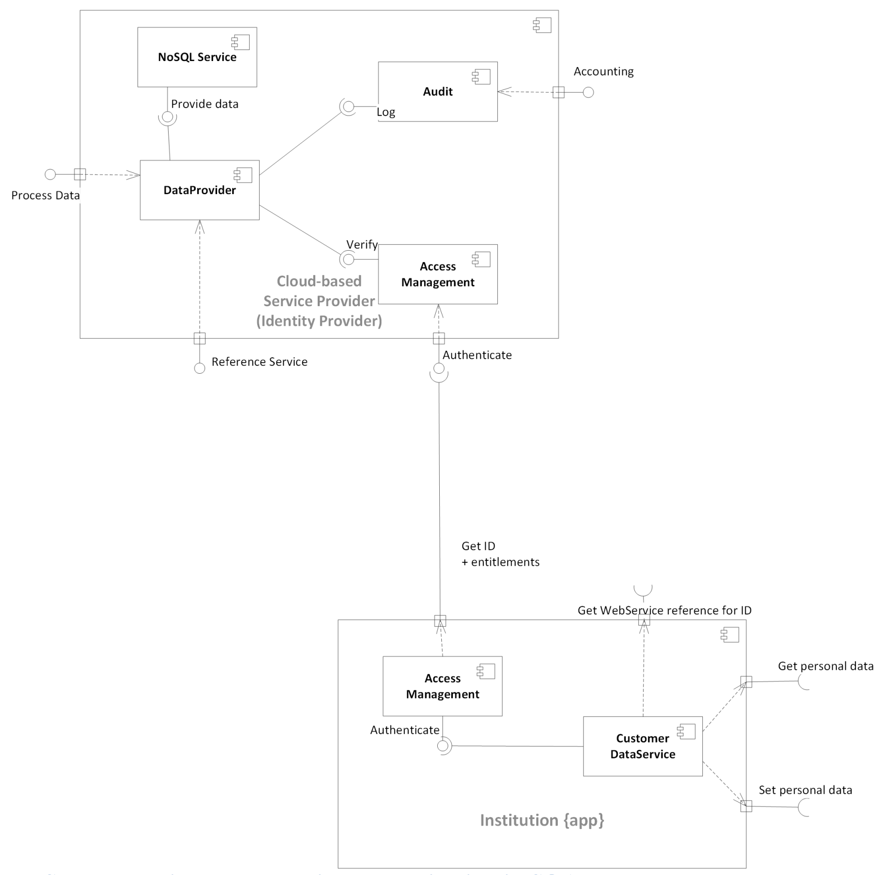


Figure 7 System architecture: Identity authentication in SOA

2.5.4 Office Open XML – Standard Schemas

Office Open XML (OOXML) standard is mostly built on top of XML files, which reference to each other to form a single document. XML files can be supplemented with other reach files to deliver graphic, multimedia and other elements [62]. OOXML data format can deliver data integrity using internal elements hashing, while confidentiality can be assured by a single ZIP wrapper password protection and content encryption. These techniques are sufficient to protect content that does not leave corporate network, however, when leaked this built-in protection may not be sufficient for personal data. Cloud-based identity metadata sharing solution to utilise OOXML standard would require additional safeguards from service providers.

Distributed identity metadata requires a data structure that will allow a comprehensive view of the data. E.g. doctor checking patient's medical record will look not only into a single laboratory result but would also need to see other medical opinions that the patient received after medical evaluations were made. Such an overview can be shared with the doctor using the standard functionality of OOXML. Parts of the overall report that could be edited could be natively controlled in the so-called Master document. Composite reports can be set to read-only state due to various factors related to document *checked* *out* state or related to access rights that doctor has over the entire report and its subsections.

Regarding OOXML data indexing, in large databases NoSQL-based it is an easy task for indexing engine as long as the document is not encrypted. This part requires further research related to OOXML data anonymisation and obfuscation for indexing purposes.

Office Open XML (OOXML), here a XACML policy wrapper, is a ZIP package file consisting of one or more file sections followed by a central directory. Multiple XML document elements define the main document part. 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, some sections are not covered by a native OOXML cryptographic techniques [63]. In sticky policy with identity-based encryption (SPIBE) concept, a XACML policy is added as an additional package content that remains in unencrypted XML format. This policy defines access rules over resources and implements attribute-based access control (ABAC)-like with attribute values defining legitimate data processing subject, also role-based access control (RBAC) [64] where business or institutional roles define who can access the data and finally risk-aware access control (RAAC) expressions [65], the most dynamic access control technique making access decisions upon dynamically calculated risk.

2.6 Cryptography overview

Every cryptographic algorithm could be characterized by keys, algorithm and the message [66]. The perfect key is a truly random group defined at {0,1} of size n. Key size is often predetermined by the cryptographic algorithm. Larger key size gives lower probability to break the encryption. However, a not truly random key highly lowers the security of the cryptographic algorithm as the predictability of the key is highly lowered due to a limited key space [67]. Common random key generators rely on physical deterministic hardware setup and machine boot up time to generate keys. These possible keys might seem hard to predict, however, does require only existing modern computing power to factorise all the possible keys generated by the underlying randomising computer library. Key length should be selected precisely for the model, considering key crypto-period, protected message characteristics such as other key or a data, a context where the message is either in transit or at rest, or the available processing power, and the allowed energy consumption.

2.6.1 Key Crypto-period

Keys could have either static or ephemeral life, although all keys have to be revoked at some point of time the ephemeral keys have rather very short lifecycle in compare to static keys. Although a shorter key lifetime results in better security, it also reduces performance. Diffie-Hellman (DH) is a good example of a cryptographic primitive where the key is used to provide perfect forward secrecy (PFS). This ephemeral key encryption is often used to exchange other static keys for long-term cryptographic algorithm lifecycle. PFS property ensures that having two cryptographic primitives one used to protect communication channel and the second used for the long-term message encryption, in the case where the long-term key is compromised, the ephemeral session will not be automatically compromised. Advanced Encryption Standard (AES) is an example of a static key use, where the single secret key could be kept in hardware security module (HSM) for a longer period of time. Such key requires increased safeguards as it is often used to encrypt a larger number of keys and messages. Another approach is to define different keys life-type within the same system are public key certificates (PKC) or public key infrastructures (PKI). Certificates consist of RSA key pair [68] and other key metadata such as a certificate expiry date and a key usage. If the key expiry date is defined, the system using such a certificate should reject it for further use and revoke it. Certificate templates based on different certificate purposes specify whether and when keys should expire. Normally such a key life-time is defined with days, months, sometimes years. Rarely RSA keys are set without an expiry date under the PKC.

2.6.2 Symmetric and Asymmetric Keys

The most common cryptographic key categorisation is derived from symmetric and asymmetric algorithms. In symmetric encryption such as AES [66], the same key could both encrypt and decrypt the message. This key is also referred to as a secret key because it requires higher security safeguards to keep it secret and exchange with parties to encrypt and decrypt the message. History shows that symmetric encryption has a significant disadvantage when it comes to the key exchange between parties. This cryptographic construct weakness helped Polish mathematicians during World War II breaking Enigma, the cryptographic hardware-based algorithm used by Germans to exchange strategic information [69]. French counterparts managed to steal symmetric shared keys what helped Polish cryptographers breaking the algorithm. Fortunately, modern symmetric algorithms follow the Kerckhoff's principle, where the encryption algorithm is known to parties, although decryption keys are kept secret. Unlike Enigma, the AES cryptographic primitive is an official open encryption standard, where the security boundary starts and ends at the key itself. AES algorithm does not leave much space for tampering giving limited adversary surface for backdoor implementation [70].

On the contrary to symmetric cryptography, the asymmetric algorithm uses two different keys, one to encrypt and the other to decrypt the message, or to sign and then verify the message authenticity. Keys used by this cryptographic primitive are often referred as a public and a private key. The divulged key used for initial encryption or signature verification is called a public key. The other that is kept a secret is referred to as a private key and could be used to decrypt or sign the message. DH, RSA and elliptic-curve cryptography (ECC) [71], [72] are one of the most commonly implemented public key cryptographic algorithms.

2.6.3 Algorithm Security Lifetime

From the data perspective, the data kept at rest is exposed not only to current cryptographic vulnerabilities. Encrypted data stored for a time exceeding algorithm security lifetime (ASL) [73] is exposed to potential future threats. The time length during which the data has to be protected should be taken into account [66] when selecting the cryptographic algorithm. ASL for asymmetric encryption is relatively short in comparison to symmetric encryption mostly due to a public key that by design is known to different parties. The greatest challenge that shortened ASL of all the commonly used cryptographic algorithms is quantum computing and the high computational power that comes with it.

2.6.4 Quantum Computing

Although quantum computing does not pose a direct threat to all modern cryptographic algorithms, it could be successfully used to break most of the commonly used asymmetric key algorithms [74] and also symmetric encryptions if too short key sizes are used. Quantum computers, unlike traditional binary machines, use quantum bits. The qubits could exist simultaneously in two states representing 0 and 1, what is called superposition. High parallel processing power quantum computer achieves via entanglement, a state of two initially completely independent qubits that became entangled [75]. Such qubits cannot change states independently hence if one qubit changes state the other changes the state as well. In n qubits [qb] computer the entanglement allows 2^n simultaneous operations. This implicates the most obvious attack on a cryptographic construct by efficiently factorizing all possible keys. A more sophisticated challenge for cryptography is a Shor's algorithm [76] allowing factorization of large prime numbers m is making RSA algorithm vulnerable to the first actual Shor's algorithm quantum implementation.

The RSA algorithm strength comes from a difficulty to produce large prime factors [68], therefore even with large keys, this algorithm is no longer secure under quantum processing paradigm [74]. Shor's algorithm could be used to solve discrete logarithm problems what could be leveraged to break elliptic curve cryptography (ECC) over Galois Fields (GF) [77]. It would require 1000[qb] to efficiently factorize 160-bit ECC GF keys. In regard to symmetric cryptography, the quantum computer could be used to factorise data encryption standard (DES) keys. Although DES is not considered to be a secure [78] it is still being used by many organisations. It could be broken using current computational power, however, with Grover's algorithm [79] applied on a quantum computer, the adversary could construct tables for every possible DES key finding all possible collisions with $\sqrt{n}$ searches over n unsorted database records.

2.6.1 Key Management System (KMS)

Security of cryptographic schema is not only about algorithms but mostly about the implementation. Insufficient safeguards behind key management give the adversary an advantage over the most sophisticated and secure cryptographic primitive [73]. Key management systems (KMS) aim to keep key management under strict governance. The key management should be aligned with a protected data type, algorithm security lifetime (ASL), a key crypto-period and a cryptographic schema itself. Quantum computing is the threat with the highest concern for cryptographic keys security [80]. It has been proven that the efficient keys factorisation [76] and highly efficient searching over unsorted data sets, i.e. factorised key tables [79], allow an attacker to compromise the security of legacy as well as cloud-based systems. KMS system consists of different Crypto modules that are components extending KMS cryptographic functions and constraining key security boundaries.

A hardware security module (HSM) provides both physical and logical security boundaries for the protected key. However, KMS could consist only of logical software layer module. HSM implements several cryptographic safeguards including cryptographic modules and random number generators (RNG) and some of the products already implement quantum number generators (QRNG) [81]. Unfortunately, mostly financial instructions and governments, rarely medical care and enterprises consider HSM implementation. Most often KMS boundary starts and ends at the single database level where either entire database is encrypted or only specific sensitive tables [82]. The keys, however, that encrypt this database is either cached locally, stored in rights-protected files or are kept in the system registry. HSM is an appliance that often keeps key–encryption keys in a dedicated security boundary with own security procedures and own network. From GDPR perspective, often legacy logical software KMS gives better protection than the protection offered by cloud service provider (CSP) due to legal and jurisdictional issues. Proper identity federation could provide sufficient security boundary for cloud identity, where actual credentials are kept either on-premises or are hosted by certified local CSP [21].

Considering commercial cloud-based KMS that is a part of some corporate grade cloud suite, Microsoft has implemented HSM support for its information rights management (IRM) solution, MS Rights Management Services Online or Azure RMS [7]. It supports key and data encryption using both cloud-based and on-premises HSM modules. Unfortunately, only limited data scope could be covered with on-premises HSM, what in most cases provides sufficient safeguards for keys integrity and confidentiality [73]. Keys stored outside of a legal jurisdiction in countries like Switzerland could cause concerns as the local data protection law is considered equal or even more restrictive than existing European Union regulations [83].

2.6.2 Identity-Based Encryption (IBE)

Encrypted data require secure key repositories able to perform revocation when necessary. In cloud-based implementations, IBE works efficiently by introducing ephemeral cryptographic keys. In IBE, with public key encryption, the public key can be derived from a unique identity identifier. Therefore, this approach reduces the need for certificate authorities and public key certificates [84]. While public key cryptography (PKC) successfully protected data in large environments offering high scalability across various security boundaries, the IBE can simplify the encryption management offering sufficient security at the same time [85].

As an example, one institution can release an encrypted report with keys to other institution, which can only access it during the strictly controlled period. Identity-based cryptography (IBC) is a technology already considered for secure data sharing in the cloud. Identity-based encryption (IBE) could be [10] easily scalable across several security boundaries. Considering the fact that most of private cloud service providers (CSP) offer trusted or semi-trusted platforms to store data, the combination of both native CSP security and IBE applied by data owner through access control framework might deliver an acceptable level of security required to protect personal and non-personal data.

The concept where the public key could be derived from any arbitrary text was proposed in 1984 [86], however, the first practical IBE implementation waited until early 2000. Different types of IBE schemes provide benefits for different system models. The first working IBE scheme relies on Weil pairings with security based on the computational Diffie-Hellman assumption [10]. Another IBE schema dated around the same time as BF is based on the difficulty of distinguishing quadratic residues from non-residues in the ring with RSA two large primes products [87]. This model outputs long ciphertexts as a product of relatively slow bit by bit encryption.

One approach to effective key lifecycle management is forward-secure hierarchical identity-based encryption (fs-HIBE) with self-expiring keys. fs-HIBE has been successfully used for several identities and access management for IAM implementations. It allows secure dynamic joins between identities, making use of time constraints and dynamic key revocation [84]. Multiple hierarchical ID-based encryption schemes (MHIBE) is another concept derived from a generalisation of fs-HIBE. MHIBE is not only highly suitable for federated identity management systems such as this but, because of the ability of encryption with multiple ID-tuples, it can be efficiently used with the role-based access control (RBAC) systems implementations [84].

While most of the currently used public key-based schemes are easily breakable using a quantum computing, there are public key primitives based on lattices that could stand the new computing power challenge. Lattice-based cryptography was proposed by Ajtai in 1996 [88] although it waited two decades to be recognised as one of the most promising long-term methods to protect information. Following lattices, the NTRU was the first open source public key system based on Ajtai work. By changing the setup, it was possible to create an IBE over lattices such as relatively small the key and the cyphertext sizes are acceptable for actual real-life implementation [89].

Last and the most promising approach to amend the IBE construct for post-quantum computing was made by defining IBE pairing over isogenous pairing groups (IPG) [14]. The IBE-IPG does not change the initial IBE construct but only amends the way the curves are evaluated for morphisms. Isogeny, in other words, looks at elliptic curves morphisms not only as regular curves but curves that have a specific shape. Thanks to this observation the Weil pairing used in IBE BF [10] could also be constructed under isogeny and with a reduced schema amendment the existing system could be adapted as the new quantum safe construct.

2.6.3 IBE with Authenticated Encryption

Authenticated encryption with authenticated data (AEAD) was initially introduced as an extension of authenticated encryption (AE) where cryptographic construction could deliver not only message confidentiality but also data integrity. AE ensures confidentiality against an active adversary that can decrypt the ciphertext. The product of AE authenticated message is only a ciphertext, while in AEAD an encrypted message is accompanied by a plain text data that can be used to efficiently evaluate the message authenticity before any other crypto techniques are involved.

Authenticated identity-based encryption (Authenticated IBE) delivers both message confidentiality and integrity on top of IBE [90]. To empower security of our model data integrity should have additional safeguards, this is where we decided to use Authenticated IBE. Authenticated IBE could ensure that both XACML policy and OOXML document content cannot tamper. In cloud space only digitally-signed documents give a non-repudiation assurance. In other words, author of a document, e.g. general medical practitioner (GP), can be sure that document content after being signed has not been falsified by any adversary [5]. The only bottleneck of Authenticated ABE is that unlike standard AE here encryption and signing are separate operations, therefore, are more expensive operations, although this crypto approach satisfies security requirements.

2.6.4 Obfuscation

Obfuscation methods aim to hide data, so it cannot be directly processed. Obfuscated data allows the object owner to reveal only the necessary information required to execute an operation on that information without exposing PII part. For example, a health cloud-based services provider (CSP) could introduce a technique where patient's identity is not a subject of an exchange between parties; instead, unique pseudonyms are exchanged between parties to securely satisfy claims [4].

Obfuscation uses basic cryptographic techniques to hide rather than encrypt a data. These methods use keys and functions to derive obfuscated information that corresponds to sensitive information, i.e. identity metadata, but which do not disclose actual information [4]. To decide which part of identity metadata should be obfuscated, policy-based obfuscation can be used, where different policies enforce obfuscation of specific fields before these are made available for cloud-based processing. The privacy manager implementation proposed in [4] for personal data obfuscation can almost transparently integrate with existing applications. Thus, this is a reasonable safeguard that ensures data security due-care principals.

2.6.5 Anonymisation

Another approach is anonymisation. Several types of research suggest that personal data requires further safeguards, where actual information cannot be simply linked with an individual, therefore, can protect personal rights and, at the same time, following relevant legal consent, deliver useful research or other materials. Access to the anonymised part of identity metadata mostly applies to medical research [91], where medical staff and research students can benefit from previous medical records to save peoples' lives.

Technically, in research, it is acceptable to process an anonymised data. K-anonymisation techniques are widely studied as part of artificial intelligence research. They apply to dataset processing where sophisticated attack techniques like data linking (data inference) can be used to uniquely identify individual from among other records that are not directly exposed for processing [92]. Quasi-Identifiers (QIs) can be derived using k-anonymity from the table of k number of records, where the k-anonymous table ensures anonymity of the QI from among other k-1 records [93]. K-anonymisation can be effectively used to deliver statistical data securely. Therefore, all personal data processing, which requires generalised information rather than identity-specific data should be delivered via anonymisation. Here, as an example, effectively anonymised information exchanged between parties or exposed to the general public for research purposes will help others to base their work on personal data, that when non-anonymised would be restricted for processing because of the data protection. Students who need to study patients' history rather than an individual patient's case would have access to extensive knowledge-base of securely indexed and anonymised data.

Existing XML obfuscation [4] and anonymisation [93] are techniques, which provide high-performance searching and indexing algorithms, ensuring the accessibility required. Both techniques could be potentially used for most of the implementation. However, complexity of efficient obfuscation or anonymisation platform may be much higher than any encryption applied on top of the protected data.

2.7 Integrity and Authenticity

Identity meta-data must provide the most accurate information possible. It should ensure not only data quality but also data integrity and authenticity. Data quality can be maintained with well-designed XML ontologies applied at different identity metadata contexts. Data integrity gives assurance that data has not been amended since the last valid data change was committed. Authenticity ensures that the subject identified as the last data processor initiated the data transaction. Because changes made over the identity metadata are not accountable at the identity metadata level, they require dedicated functionality responsible for accounting. Identity metadata itself needs to deliver a basic integrity and an authenticity assurance. This assurance could be guaranteed with a digital signature applied to the part of the information that requires data integrity. As the digital signature could be derived not only from the information but could be bound with a unique identifier, it is used for information, which requires data authenticity [94].

As an example, in medical report authenticity of information is crucial to verify that diagnoses made have not been amended by the illegitimate party. When a patient's personal record was updated by some medical personnel, and afterwards the same information was changed either by a patient or another healthcare staff member, this later change has to be uniquely distinguished from all previously made changes. For identity metadata, we need to ensure that a malicious or ignorant subject did not amend the information, that information was changed in the current identity context, and an entitled subject processed that information.

The access control models we described here use signing for non-repudiation and integrity enforcement; however, identity meta-data requires the same enforcement at the level of actual data. For instance, using an emergency access example, where a medical professional need to access a patient's data to check their medical history if an unauthorised subject (including the data owner) amended medical history, it may have critical consequences leading to patient's death.

Digital signing cryptography requires secure keys to derive a signature. Public-key infrastructure (PKI) and ID-based signing (IBS) are two different approaches we can use to deliver keys [95]. While PKI involves trusted certification authorities (CAs) to certify public keys and bind them with a digital identity, in IBS the public key consists of an identity unique identifier. Therefore, it simplifies the implementation model by eliminating CA entity from key management lifecycle However, it introduces other required by IBS system entities.

To keep the identity metadata model as homogeneous as possible and therefore potential framework simple, we will focus on IBS as a preferred digital signing technique. IBS and IBE share the same concept for secure key management. IBS, unlike PKI, can use certificates issued by an involved trusted authority (TA) based on identity identifier and the assigned public key. IBS certificate does not require a CA, as it is a simple digital signature derived from a public key and a unique identity identifier [95]. As an alternative to certificates, IBS can utilise the hierarchical ID-based encryption (HIBE) discussed above as a preferred identity metadata encryption method. Hierarchical IBS (HIBS) schemas become very useful when combined with HIBE [96] as HIBE schema derived from content encryption can be transformed into HIBS schema. The digital signing and verification processes are therefore simplified.

2.7.1 XACML Accountability and Auditing

Furthermore, not only data, i.e. OOXML, 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 [97]. Same functionality used for OOXML could be leveraged for XACML policy as XACML data incorporated as a part of the OOXML package inherits security safeguards from its wrapper.

2.7.2 Merkle Trees applications

Various applications are leveraging Merkle trees construct designed to ensure distributed data or database integrity like in [98]. Blockchain and git repositories, the most popular, have the required functionality available already as a cloud-based service. For secure construct, the 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 commit the same version updates with two different contents simultaneously. Merkle Trees could ensure that and allow quick and efficient verification of data and its version in large data structures. Simple hashing, a cryptographic primitive leveraged by Merkle trees ensures the integrity of the current and the preceding tree leaf.

2.7.2.1 Data Versioning with Blockchain

Blockchain maintains one central chain of all the transactions. The single chain usually consists of the latest blockchain hash. XACML policy instance and OOXML package versions could be located on a centralised 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 the authorised actor. Classical blockchain implementation maintains basic transaction metadata unlike Git repositories, where the entire data history is stored. Excluding the consensus available in blockchain, these are both very similar.

2.7.2.2 Changes History via Git Repository

In Git everyone may have several branches ascendant from the same data. Consequently, everyone could commit the latest version into a chain by resolving conflicts with the latest committed version. Unlike 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 a single branch consisting only encrypted versions.

2.8 Authentication

Identity as an access attempt subject identifies itself and initiates authentication flow. Before the authentication decision is made, a target system needs to meet basic requirement namely it has to support the same authentication protocol. Several factors will determine whether an authentication framework will suit end-user, medical institutions, organisation or enterprise needs. Multi-factor authentication model empowers authenticity and can be an enhancement to the most common username and password exchange. *Something you know* is the Type 1 authentication factor specifying anything that identity owner knows and what can be verified like password, passphrase, mother's maiden name, ATM PIN, and so on. *Something you have* constitutes the Type 2 factor allowing authentication authority to verify whether the person owns physical object during authentication flow. This can be any electrical device like a one-time token generator, memory stick, smart card, and so on. Finally, the last Type 3 factor, the *something you are* verifies the identity of the owner physical unique features, that includes all types of biometrics like fingerprints, hand geometry, retina and iris patterns, body movement, voice prints etc. [99]. On top of these three basic authentication factors, there is an additional identity specific factor *somewhere you are*, which, in a cloud environment, can be simply verified, and additional claims are often transparent to the authentication initiator.

Authentication system usability is often achieved with a single sign-on (SSO) model where the subject, here identity once authenticated is allowed to pass other trusted systems authentication challenges without being prompted. The bottleneck of SSO is that when implemented without well-defined authenticator policies and with lack of multi-factor authentication can expose even highly secure systems. If identity is compromised, all SSO-enabled systems will accept subject authentication claims as legitimate without further verification [99]. Recently Microsoft warned about new Zero-Day vulnerability allowing an attacker to take control over active end-user session through crafted MS Office document using built-in Object Linking and Embedding (OLE) functionality. Such compromised session in SSO enabled environment maintains unsecured access to all interconnected systems.

Cloud-based data sharing model to perform any activities as an access attempt subjects or simply as a passive accessed object should support several authentication technologies and methods including multi-factor authentication and SSO. Only by re-authentication or progressive identity authentication using different factors the data can be effectively protected and hosted in a shared cloud space.

In a cloud computing era any closed trusted environments can provide the best possible security in terms of information confidentiality and integrity, however, such environments fail to deliver high availability. Therefore, these solutions will have difficulties to sustain. New global identity and data sharing models require authentication that will bring data securely into the shared cloud space, where global organisations and institutions such as medical, financial, educational, government frameworks aiming to protect shared data require a reliable and efficient authentication method. Furthermore, cloud computing seeks for authentication protocols that are easy to integrate with existing internal infrastructures. Kerberos is one of the most popular authentication protocols, which successfully secured large infrastructures. However, since it was challenged with cloud-based authentication, it never adapted to new open cloud space [100] without the use of new protocols like Secure Assertion Markup Language (SAML) or WS-Security.

2.8.1 Kerberos

Kerberos is the most popular authentication protocol implemented by some of the largest world organisations and institutions. Many medical institutions use Kerberos as a standalone authentication technology or adapted it as a part of Microsoft Active Directory Domain Services (MS AD DS) product suite. Because of Kerberos architecture, it is often used for single sign-on solutions within relatively closed security boundaries. Any non-kerberized and not joined environment has to maintain its identity repository and authentication, and authorisation systems. Let assume a medical doctor or a general practitioner (GP) have an account in Kerberos federated institutions environment (see Figure 8) and is coming to use X-Ray medical facility where GP does not have any credentials, therefore due to Kerberos constrains the GP is not able to use the facility with currently owned federated account [100].

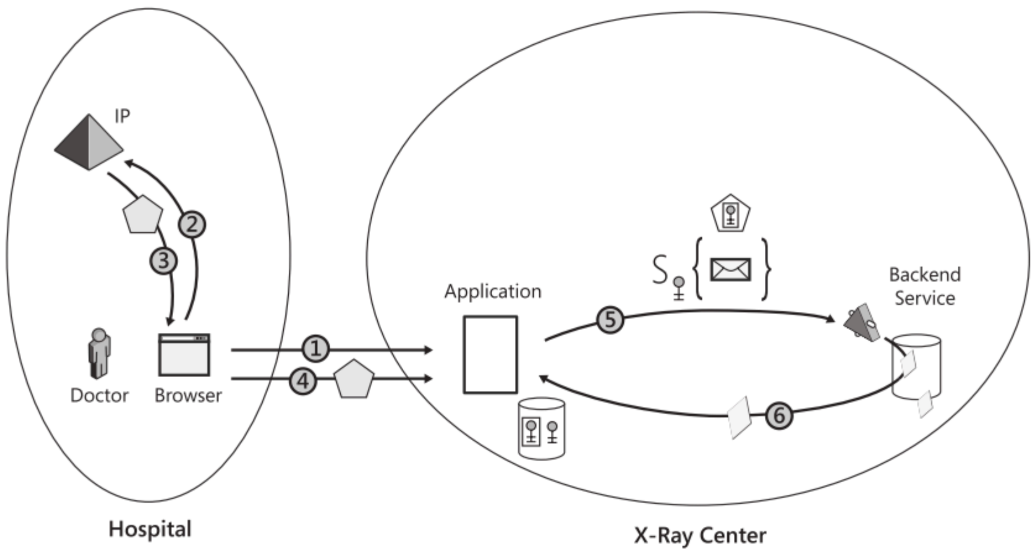


Figure 8. A trusted subsystem – no Kerberos token exchange possible (source Bertocci, 2011)

In terms of cloud-ready technologies, the Kerberos relies on private-key cryptography where all involved sides have to protect a shared private-key [99]. Even though technically cloud service can be integrated with existing internal Kerberos based authentication systems, in a real-life scenario, for a cloud-based service provider, it becomes difficult to obtain approval from customer's information security officer for firewall policy amendments. To operate Kerberos requires additional network ports to be open (TCP/UDP 88), what should include Microsoft Active Directory Domain Services (MS AD DS) LDAP ports (TCP 389, 636), as having these two technologies together customer can receive comprehensive access control service.

2.8.2 SAML

Looking at other modern authentication protocols SAML is one of the promising cloud-friendly standards offering very high cross-platform compatibility. SAML was successfully used for global single sign-on (SSO) solutions and was implemented in several products offering federated identity functions [100]. It is interoperable with various legacy authentication and authorisation systems, and it communicates at a high HTTP protocol layer. Therefore, it does not require any modifications to egress firewall access policies at the customer side. SAML will leverage TCP ports 80 and 443, which by default are open in most of the networks passing in the World Wide Web (WWW) contents. It defines a protocol and token XML structure schema including the use of simple object access protocol (SOAP) wrapper to exchange standardised messages between involved authentication flow parties [101].

SAML assertion consists of several features, and in the most generic simplification, it contains information about the *issuer* of the token and most often it is expressed using Distinguished Name format from X509. It has the *subject* information of all assertion statements including subject confirmation for relying party (RP) to verify the assertion relationship between the sender and the subject. Additionally, the assertion is a constraint with *conditions* that are part of the assertion. These constraints validate the time frame when assertion can be evaluated as well as define a recipient or an audience of the assertion (e.g. RP). SAML assertion also includes an *attribute statement* related to the subject, which in WS-Federation are equivalents of claims [100]. *Authentication statement* informs parties about initial authentication of the subject. Assertion also includes information about authorisation against the object for which access was claimed and this part is called an *authorisation statement*. It indicates whether a subject can access an object (resource) the way it is specified in the request. Finally, a digital *signature* verifies the integrity of the token contents, so RP is assured of the authenticity of the assertion. *Advice* is optional information related to the authorisation itself.

2.9 Access Control

2.9.1 Role Based Access Control

Most of the mature modern access control models that are ready to securely protect the asset (such as PII) from unauthorised access hardly span outside a simple boundary [42]. The well-known model, role-based access control (RBAC) can be easily adapted for highly secure end-to-end identity provisioning and revocation within specific security contexts. A role is assigned with an identity for a set of transactions, for example, as the ability for a general practitioner (GP) to access patient data and take further actions according to new circumstances and patient history [102] then update the patient record. This access control model controls the subject access over the object, based on roles assigned to the subject in the organisation, which defines a security boundary [58].

Often roles can span across several systems; here, well-integrated infrastructures can ensure role change enforcement on the end system. Furthermore, well-defined roles and consistent RBAC system implementation are safeguards against several security threats such as collusion, creeping privileges, and excessive privileges. Enforcement of separation of duties is a countermeasure for collusion attack [103], while the principle of least privilege overcomes problems of creeping privileges and excessive privileges [99]. Although RBAC ensures high security within an organisation, it does not introduce a namespace that can be implemented across organisations, for example, in open cloud space.

The identity metadata requires the RBAC concept with its several variations to enforce control and secure identity with its metadata in the cloud. To protect a policy that is applied to the object in the cloud and accessed as a part of RBAC transaction the part that exposes the policy can be encrypted [58].

The new concept of Cryptographic RBAC for the cloud addresses several security threats that have roots in early RBAC architectures, where this access control model had closed security boundaries such as enterprises, organisations and institutions. Role-based Encryption is a model that allows data encryption before it is handed-over to the cloud service provider (CSP), thus ensuring that only data owners and identities that hold the required access role can decrypt the information.

Identity metadata requires also clearly defined ontology to reach Cloud maturity for RBAC. Several approaches are emerging that introduce standardised RBAC in different sectors, one, the Enhanced RBAC, is focused on clinical education, biomedical research, and patient care [35]. This work highlights the fact that there is a need to define strict ontologies where Enhanced RBAC could be applied. These ontologies would constrain and help to define ontology dedicated for personal data and sensitive personal data (e.g. related to patient medical history or personal assets) to allow only secure access control over such data in the cloud.

RBAC implementations have a couple of disadvantages such as a lack of global public standard defining roles in a public sector, also roles aging, where business changes outrun actual roles implementations, and one the most important one is the single point of failure. In case of an attack, the adversary could compromise the central access management system. XML-based data where RBAC is used to control access can still be protected using a distributed access control system [104]. Distributed access control systems can be scaled and adapted for cloud-based implementations. This problem was also addressed with cryptographic RBAC (C-RBAC) [58] and, unlike the distributed access control approach; this model was designed for cloud-based IAM systems implementation. C-RBAC uses policies that are enforced via cloud services, which can be controlled in a decentralised manner by the data owner.

Finally, to deliver the fully homogeneous model, the ARBAC97 and SARBAC models can be used to provide control over RBAC systems including granular role hierarchy amendments, new policy definitions and all other administrative operations which are fully controlled via a dedicated roles set [58].

2.9.2 Attribute-Based Access Control

Recently widely discussed data protection approach is access models based on attribute-based encryption (ABE). The concept itself combines cryptography and elements of access control. The attribute-based access control (ABAC) provides another approach to govern access, by giving the data owner full control over their data. In the ABAC model, roles are bound to role attributes and are attached to a data element through attributes based encryption (ABE) [105]. The ABAC model can coexist with RBAC and easily enables RBAC beyond a single security boundary [42].

ABE allows the data owner to encrypt the personal data under specific attributes. Same attributes are attached to subjects who will process the data [6]. The identity metadata model and especially PII part have to use access control system with encryption applied to access control properties that are attached to data. The ABE model has been proposed as the most suitable technology for cloud-based global data access [105], although ABAC among other access control models described here have specific features in combination that can satisfy the identity metadata. There seems to be an increasing interest in ABE as demand on electronic health-care systems has grown in the last few years [6].

Attribute-based infrastructures have been proposed as ready for handling PII information, for instance, a special implementation of ABE called ciphertext policy ABE (CP-ABE) with message broadcasting enables an ABAC system to perform ad-hoc direct revocation [106]. As with RBAC, the main problem with CP-ABE is that a single trust authority (TA) that can be used to decrypt data. Key escrow enables a single TA to decrypt all the information and a compromised TA provides the potential attacker access to all the protected data. A way to overcome this problem is multiple-authority ABE (MA-ABE) where each TA releases only a partial secret key that is used to encrypt information. On the other hand key revocation under this approach creates a bottleneck where each TA needs to be involved in a keys lifecycle [6].

While CP-ABE allows data owners to decide on attribute structure defining permissions before encrypting data sent into the cloud, the other approach key-policy ABE (KP-ABE) uses policies to define permissions, and the data owner assigns attributes to define encrypted data [106]. Service managing policies for KP-ABE automatically generates access structure for the data then combines access policies into keys [6].

Early ABAC implementations [11] suffered from dynamic membership control. However, later [107], [108], [106] ABE was reviewed and empowered with attribute revocation functionality that enabled the fundamental access control functions required for cloud-based access management. Each ABE construct [11], [84], [108], [106], [105] concentrates on cryptographic operations under several attributes. ABE makes cloud-based authorisation a cryptography-centric due to highly constrained implementations by selected ABE primitives. ABE implementations leverage many fundamental access control techniques like Break-Glass [6] where data could be accessed in an emergency scenario with a post-factum approval or a justification. Also, time-constrained attributes [106], [105] technique that compliments the access control system using ABE. Despite the fact that ABE is functionally related to attribute-based access control (ABAC) model, it seems it has never been wider discussed in the context of standardisation to simplify global integration for secure and flexible access control.

2.9.2.1 XACML

XACML is a policies standard from OASIS. Attached to a data piece could represent a sticky policy. This policy model defines tuple relationships where a subject performs a particular action against an object (see Figure 9).

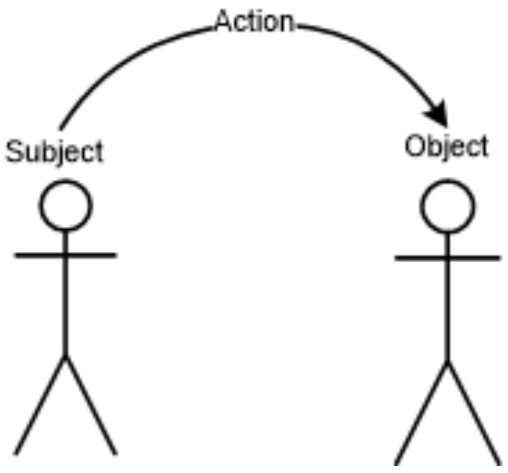


Figure 9. Access Tuple

Actions or access attempts are strictly controlled with XACML policy, which introduces the concept of obligations. Traditional discretionary access control (DAC) models come rather with static conditions where access decisions are made upon subject entitlements gathered in technically constraint security boundary. Here policy includes obligations separating stateless access conditions from stateful security context oriented conditions [109]. In other words, authorisation decisions can depend on subject attributes such as location-*somewhere you are* factor, relationship to other subjects, time, previous authorisation decision and others. It is also dependent on object state including a target system or resource state.

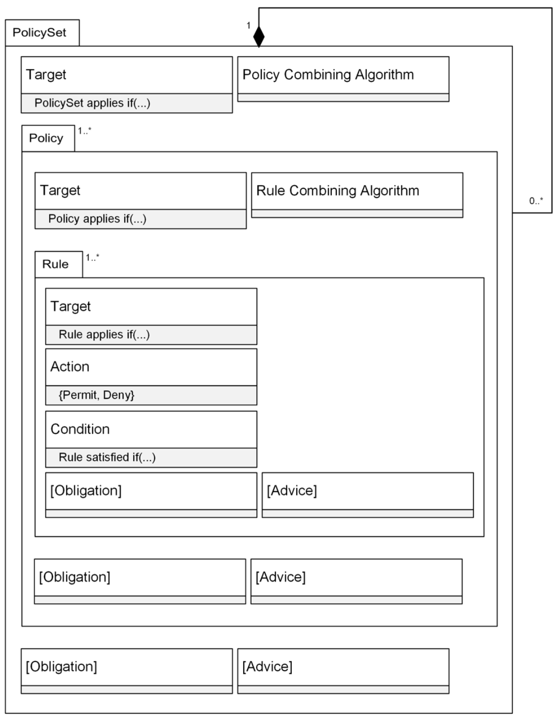


Figure 10 XACML Policy Construct

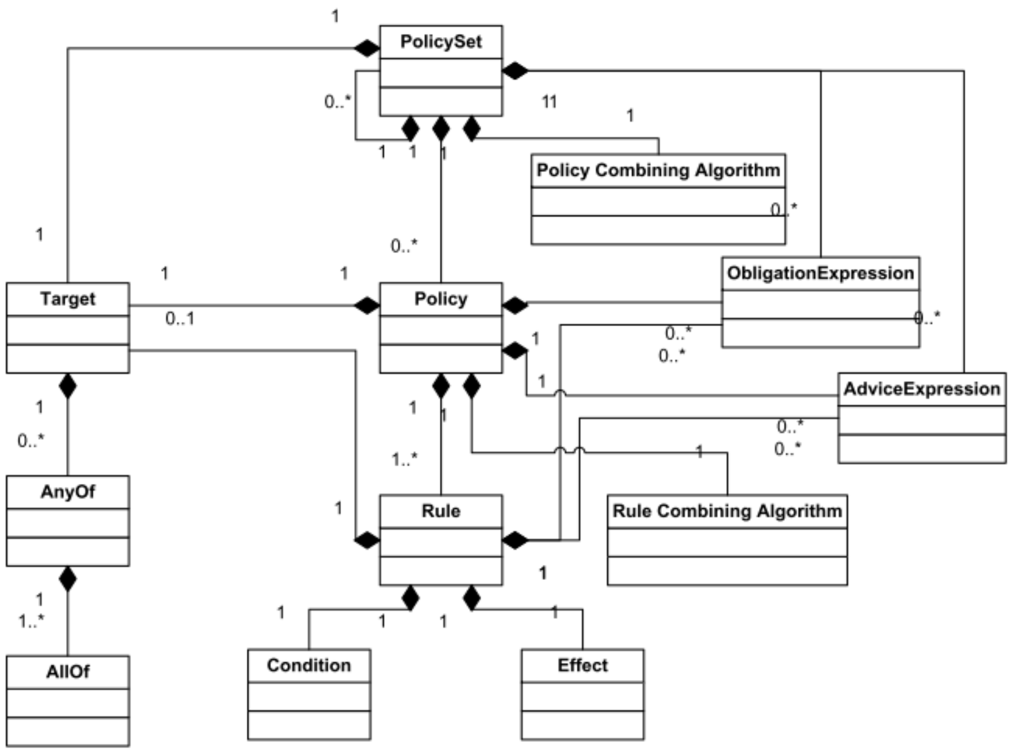


Figure 11 Policy language model [97]

This policy data model comprises of three elements (see Figure 10 and Figure 11): *rule*, *policy* and *policy set* [97]. The *rule* is the most fundamental piece of policy defining the target, which is an object in the access attempt tuple, the effect that can be expected after evaluation of the rule. E.g. rule ensures, that highly confidential content object, i.e. target can be processed by the subject located only in countries specified by the policy condition. XAML condition is representing a Boolean expression resulting in True or False. *Policy*, the next XACML element, is a rules wrapper that can be passed amongst data-flow entities. It is constructed with a policy target, where, in a sticky policy model, the Target delivers only additional classification meta-data for the enclosed document. The rule-combining algorithm defines how the composite rules results are combined. XACML version 3.0 defines several policies and rule-combining algorithms [97]:

- **Extended Indeterminate values** – used to hide potential inconclusive decisions allowing only Permit or Deny while the PDP needs to enumerate "indeterminate" values to combine rules and policies.

- **Deny-overrides** – used to take logical Deny decision if any rule or policy results in Deny. Execution of all rules is stopped immediately after the first encountered Deny result. Permit result is only possible when none of the evaluated policies and rules gave Deny result (see Table 5). Policies, policy sets or rules may be executed in any order.

- **Ordered-deny-overrides** – same as Deny-overrides used to take logical Deny decision if any rule or policy results in Deny. The only difference is that the rules evaluation order is predetermined by the policy containing rules listed in strictly predefined order (see Table 6). Same policies are evaluated in the order specified in the policy set.

- **Permit-overrides** – used to take logical Permit decision if any of rules or policies results to Permit. Evaluation of all rules or policies will stop immediately if at least one results to Permit (see Table 7). Policies, policy sets or rules may be executed in any order.

- **Ordered-permit-overrides** – same as Permit-overrides used to take logical Permit decision if any rule or policy results in Permit state. The difference is the rules evaluation order; it is predetermined by the policy containing rules listed in strictly predefined order (see Table 8). Same policies are evaluated in the order specified in the policy set.

- **Deny-unless-permit** – used to hide potential inconclusive decisions allowing only Deny decision before any rule or policy evaluates to Permit (see Table 9). Algorithm never results to either "Indeterminate" or "NotApplicable".

- **Permit-unless-deny** – unlike Deny-unless-permit, here all rules or policies evaluate to Permit unless the first Deny occurrence (see Table 10). Algorithm hides the non-conclusive results hence results to neither "Indeterminate" nor "NotApplicable".

- **First-applicable** – stops evaluation if any of the rules or policies resulted in conclusive Permit or Deny state (see Table 11). If an error occurs while evaluating the condition of a rule, then the evaluation stops, and the policy evaluates to "Indeterminate", with the appropriate error status.

- **Only-one-applicable** – unlike other combining algorithms this applies only to Policy Set; algorithm evaluates to Permit or Deny only when only one child returns valid Permit or Deny decision (see Table 12).

- **Legacy Deny-overrides** – designed for all the cases where Deny result should have precedence over Permit decision (see Table 13). This combining algorithm is depreciated.

- **Legacy Ordered-deny-overrides** – same as Deny-overrides, used where Deny result should take precedence over Permit decision (see Table 14). The rules evaluation order is predetermined by the policy containing rules listed in strictly predefined order. This combining algorithm is depreciated.

- **Legacy Permit-overrides** – here Permit result should take precedence over any Deny decision (see Table 15). This combining algorithm is depreciated.

- **Legacy Ordered-permit-overrides** – same as Permit-overrides, Permit result takes precedence over other Deny decisions (see Table 16). The rules evaluation order is predetermined by the policy containing rules listed in strictly predefined order. This combining algorithm is depreciated.

Finally, the last XACML element, the *policy set* (see Figure 10) is constructed with the target and set of policies. The possibility of Policy and PolicySet nesting gives many possibilities to represent access conditions, however, from an architectural perspective, it seems reasonable to keep the policy relatively flat and constrained by templates from a given TA context same as in [8]. Considering time required to evaluate complex policies and PolicySets the despite available XACML features, the policy words should be reduced to a minimum [110]. The interesting functional part that is defined by XACML is obligations and advice. The obligation is a *must* requirement compared to non-obligatory advice, which *can* be considered during access control decision.

Obligations are stateful actions that must be taken upon authorisation decision. Only the possibility to evaluate context distinguish obligations from regular stateless access conditions [111]. Obligation expressions are evaluated by Policy Decision Point (PDP) into obligations and passed onto PEP and the advice expressions, which is the same as obligation expressions that are resolved into advice and passed to PEP. The obligation for Policy Enforcement Point (PEP) enforces additional stateful conditions and similar to obligation the advice, which is optional (unlike obligations). Obligations and advice were distinguished in XACML Version 3 to separate the obligation that is must statement for PEP from the advice that can be considered by PEP, e.g. Bob can be denied access because he does not have a valid email address from the educational *ac.uk* domain.

OASIS empowered XACML with health-care system authentication architectures [112] and defined entities, i.e. Access Control Service (ACS) responsible for taking access control decisions. Proposed here model integrates existing architectures with an identity provider (IdP) 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 plaintext and follows the package ensuring data confidentiality before successful data access authorisation.

To enforce access rules in real productive implementations, it is reasonable to consider a way to efficiently transform XML-formatted policies from and to an abstract object-oriented construct that delivers all programming interfaces required to automate policy creation and evaluation. One of the most applicable techniques that could be used here is a data serialisation. This is the encoding method where objects of any type are effectively translated into series of bytes, words or even into higher level formatted JSON and XML language structure. Serialisation techniques are widely used in programming to share objects between services implemented under Service Oriented Architecture (SOA). Dedicated serialisation methods can improve the performance of data translation or transposition from relational databases into XACML [113]. Well-designed serialisation components can deliver interface between legacy systems used in medical institutions, enterprises or governments and modern cloud-based services. Frameworks or systems that provide cloud-based access control services should provide functionality that allows migrations or co-existence with legacy systems as the most of existing data, what includes medical data, is still hosted outside of the cloud.

2.9.2.2 XrML

XrML policy format created and patented by Xerox in 1994 with a purpose of digital rights management (DRM) [114]. Currently, XrML is owned by ContentGuard a private software company. Similar to traditional discretionary access control (DAC) XrML condition, access decisions are made in technically constraint security boundaries. XrML 2.0 policy defines a *principal* an access control subject that is given some *right* defining predicate actions over a *resource*. The *license* is the top policy level (Figure 12) under which all the *grants*, license issuer principals and other meta-data. *Grant* consists of the remaining policy elements (see Figure 13), subject *principal*, the *rights*, *resource* objects and *conditions*.

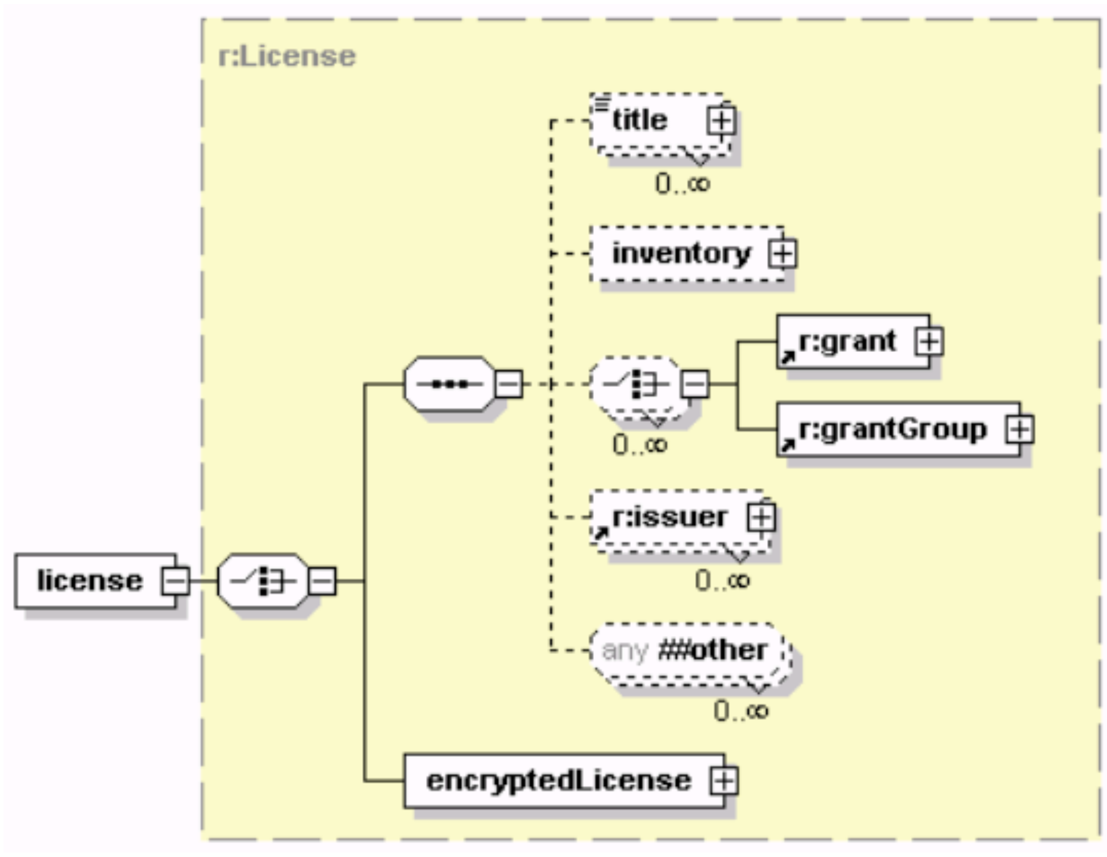


Figure 12. License Model [114]

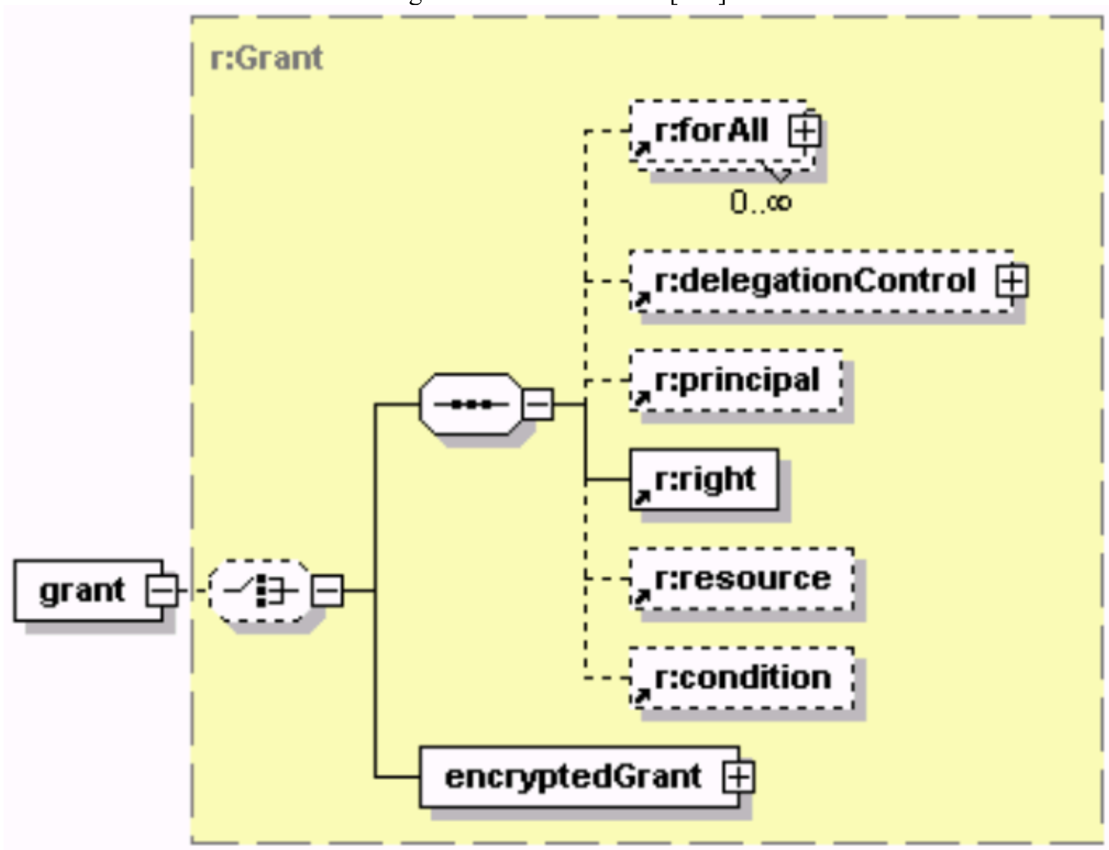


Figure 13. Grant Model [114]

A *Principal* could either define the license issuer or a subject. Among other accepted identifiers the principal accepts a public key cryptography-based identity allowing subject definition using public and private key pairs. The *principal* is then defined under *keyHolder* describing identity using the public key information (see Figure 14).

```
<license>
    <grant>
        <keyHolder>
            <info>
                <dsig:KeyValue>
                    <dsig:RSAKeyValue>
                        <dsig:Modulus>Fa7wo6NYfmvGqy4ACSWcNmuQfbejSZx7aCibIg
kYswUeTCrmS0h27GJrA15SS7TYZzSfaS0xR9lZdUEF0ThO4w==</dsig:Modulus>
                        <dsig:Exponent>AQABAA==</dsig:Exponent>
                    </dsig:RSAKeyValue>
                </dsig:KeyValue>
            </info>
        </keyHolder>
        <cx:print/>
        <cx:digitalWork>
            <cx:locator>
                <nonSecureIndirect URI="http://www.contentguard.com/sampleBook.spd"/>
            </cx:locator>
        </cx:digitalWork>
        <validityInterval>
            <notAfter>2018-12-31T23:59:59</notAfter>
        </validityInterval>
    </grant>
</license>
```

Figure 14. Principal a private keyHolder having temporary print rights over e-book under URI

Next, the principal is given certain legacy rights that are very specific for DRM systems. XrML 2.0 defines the following rights model: accessFolderInfo, backup, copy, delete, edit, embed, execute, export, extract, install, loan, manageFolder, play, print, read, restore, transfer, uninstall, verify and write (see Figure 15).

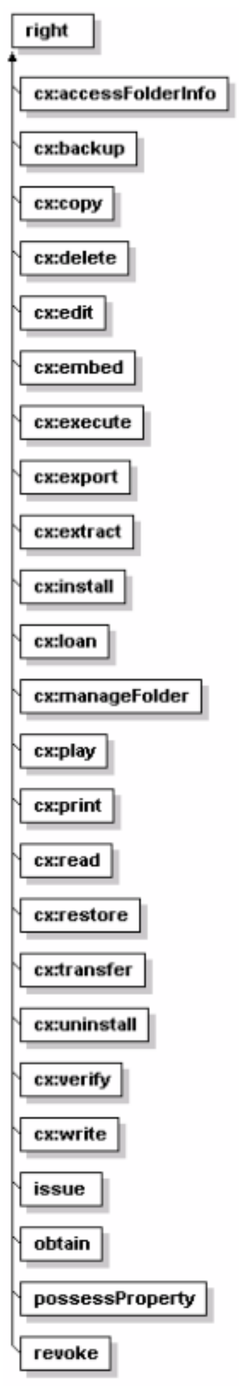


Figure 15. Right Model [114]

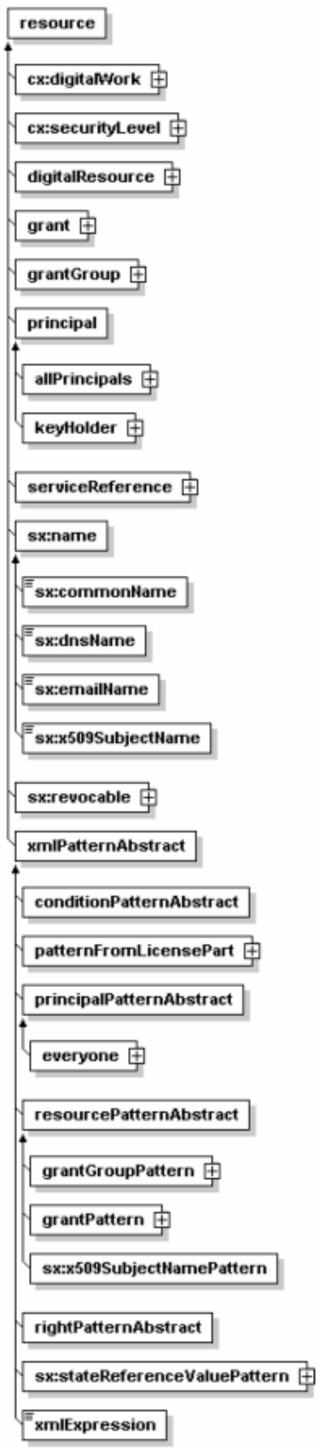


Figure 16. Resource Model [114]

A resource could be an e-book, audio, video or image file or any piece of information that could be owned by a principal (see Figure 16). Under custom setup, XrML could define access for an OOXML or a portable document format (PDF) file as well as email content [7].

A policy defines very basic conditions similar to XACML obligation, specifying circumstances under which subject could access the object (see Figure 17). Via conditions, the XrML could grant temporary access or access based on territory preferences to the subject, e.g. movie could be only watched in the United States of America.

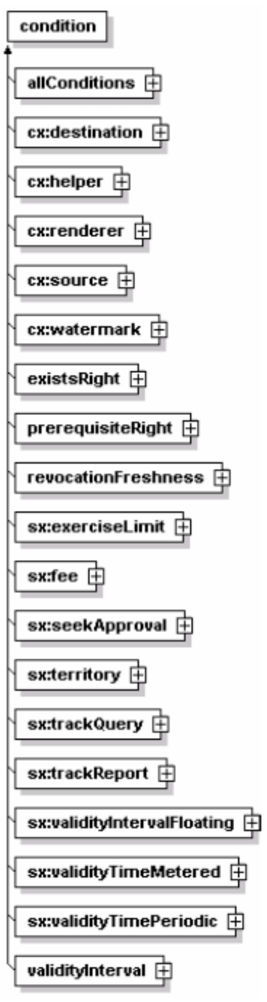


Figure 17.Condition Model [114]

2.9.3 Purpose-Based Access Control (PBAC)

All the above access models control access based on entitlements granted and detailed access policies. Another model, purpose-based access control (PBAC) allows long-term maintenance of access granted at some point in time [52] and efficiently enforces need-to-know and need-to-have principles. In more traditional access control model from the moment when access is granted to a subject via either role or direct assignment, this access relationship from a subject to an object is preserved over time unless relevant auditing procedures enforce access control review and revoke so-called creeping privileges [99]. This purpose justifies the subject to store, process or access an object [52]. It could be defined under intended purpose and access purpose categories. Therefore, the access decision is made based on the correlation between the intended purpose and the access purpose. The intended purpose falls into three components: allowable intended purpose (AIP), conditional intended purpose (CIP) and prohibited intended purpose (PIP) [51]. Where AIP defines unrestricted data access, CIP conditional data access and PIP denies any access for given purpose. Combined with access purpose, which could consist of a single RBAC assignment, the data access is enhanced by a very granular control [51].

As the RBAC model is successful in delivering effective access control functionality and became widely adopted in many enterprises, it is reasonable to consider the integration of RBAC with policy-based access control model [51].

The concept of access purpose is not an integral part of any of the previously described access control models. This does not mean that related security procedures cannot define circumstances where the subject becomes entitled to process data under the defined access control model. The purpose-based access control model shows that there is a need for legal baselines and guidelines for cloud-based IAM implementations. In a global context, there is a risk of inconsistencies between access management systems caused by conflicting definitions of legally justified access purposes. It may, therefore, be simplest for access to PII to be governed by a single legal framework, e.g. one based either in Europe or the USA. This is a challenge in a world of conflicting attempts by the USA through FISA, Privacy Shield [115] and similar and the EU through Data Protection law to establish a worldwide jurisdictional reach.

2.9.4 Break-Glass – Emergency Access

A complete identity and access management (IAM) system consist not only of technologies but also of relevant security policies and procedures built to support access control and provide reliable accountability of a subject’s activities over an object. In most generic scenarios, a subject is entitled to process data when it is granted rights at some point in time. Rights are granted based on subject roles assignment or based on direct permissions applied to the object. In a secured environment, before PII data can be processed, the subject requires a consent [116].

Now let us analyse a person’s experience of an accident abroad, and where a medical professional need to access the patient’s personal record, which is a part of identity metadata and due to serious injuries, the patient is unable to approve medical access to that data. This scenario requires a dedicated and strictly controlled Break-Glass process allowing access to personal data to be subject to post-processing approval [6]. Such an access attempt should trigger communication channels that inform the relevant authorities e.g. authorized personnel of a local health-care practice where the patient is registered. Next, in most cases, access needs to be justified by the person performing the emergency access, and then afterwards by the relevant authorities. Break-Glass action thus requires legal enforcement to account for each occurrence of the emergency access.

Lightweight Break-Glass Access Control System (LiBAC) has been designed explicitly to take any access decision upfront and next claim the legitimate approval [117]. Whichever access control model is used with new identity and access control models, there is an increasing demand to deliver positive authorisation decision before the service provider evaluates entire access context. Finally, a service provider (SP) that needs to deliver such specialised authorisation, depends on data classification, should obtain a legal approval and certification proofing that it meets all security requirements.

2.10 Conclusions

Cloud computing to serve its purpose requires compatible standards. Consistent identity implementation like SCIM schema across different cloud systems, together with support for XACML authorisation highly improves currently weak cloud systems interoperability. SCIM schema facilitates identification of access control entities expressed in XACML policy. Therefore, it simplifies actual implementation, reduces system customisations and hence imposes additional economic advantageous. Constrained XACML dictionary compliments this research objective. OOXML however, has been already adapted as a standard for word processing document format. A number of features supported by OOXML accelerates this research around IRM and impacts the final SPIBE model evaluation. Various cryptographic primitives depending on applications could improve the IRM security. From ASL perspective symmetric encryption has an advantage over other asymmetric schemes. However, any key exchange weakens the security of the entire model. The IBE as a cryptographic scheme seems very flexible in terms of ASL, where actual cryptographic algorithm could be replaced with a new crypto primitive. Finally, IBE requires a symmetric algorithm to encrypt the data. Therefore, it still could have an advantage of relatively long ASL. To complement the proposed SPIBE construct, the blockchain that addresses integrity, non-repudiation and authenticity of the information, safeguards the entire information lifecycle.

3 Sticky Policies Approach within Cloud Computing

3.1 Introduction

Since XML was accepted as a data structure among scientists, institutions and organisations there were several approaches to define globally accepted schemas and standards. The Organisation for the Advancement of Structured Information Standards (OASIS) works on building standards mostly based on XML. They have standardized many XML-based schemas and delivered a suite of namespaces for Web Services, authentication and authorisation [101]. OASIS created an open standard for access management XACML that almost entirely implements the concept of access policies [97]. While sticky policies can be easily expressed with any other XML schema and ontology [118] we see potential in using an open standard that can be adapted across several institutions, organisation and enterprises as a common language in the Cloud.

We have already mentioned the OASIS created several open standards, what includes security assertion markup language SAML, an open format suitable for authentication in the cloud [119]. Considering XACML [97] as a granular access control language SAML will complement the final access control framework [109] for secure data sharing in the cloud.

Following open standards, Office Open XML (OOXML) is a data format created by several parties [62]. OOXML content can be delivered to end-user as a standalone file or as an online rendered document, spreadsheet, presentation or other. This standard offers high transparency for data conversions and data storing [120]. This data structure is easy to index or pre-process for efficient indexing with various database types.

Combination of XACML, SAML, OOXML and optionally IBE delivers functionality for cloud-based access control framework where personally identifiable information can be securely stored in a public cloud space. These technologies support several best security practices for access control systems. In the literature review, we show that system suitable for medical institutions, organisations and enterprises should provide such information security functionality as: break-glass temporary access granted based on a policy owned by subject; dynamic access key revocation; and a key lease for a constrained period of time.

3.2 Identity-Based Cryptography

Proposed here construction rearranges IBE schema model entities like in [11], where author proposes new approaches for IBE schema. SPIBE does not leverage Fuzzy-IBE, however it shares the functional concepts with a non-primary Fuzzy-IBE application. Since, e.g. Alice's data access preferences, including document version are attached to a document in a form of a sticky policy they constitute a unique characteristic of this document, its identity under IBE schema. Alice, the author, does not use Bob's, the recipient's identity, but only the XACML-formatted sticky policy identifying the data in a security context is mapped into a public key. Finally, Bob does not use his private key, but after he is authorized by the trust authority (TA) policy engine the private key is calculated from the document sticky policy and a TA master key.

The major SPIBE feature also described in Fuzzy-IBE is that a document shared by Alice does not have to be stored on a trusted storage server instead it could be hosted on any untrusted server, which could be freely accessed by Bob and Eve. This is a status quo for efficient cloud data sharing. Data could securely change its primary location without a major data migration. Both Bob and Eve could maintain a local copy of the document securely protected and shared by Alice.

The content of the policy definitions could be encrypted using identity-based encryption (IBE) [85]. Both policies and data encrypted with IBE add security on top of the sticky-policies model. However, the bottleneck of this method is that encryption applied this way makes data *heavy-weighted* [121].
This specific IBE implementation shares the same sticky policy concept with cyphertext attributes based encryption (CP-ABE) [122], however while CP-ABE is almost entirely focused on subject access context the IBE here, used over XACML policies requires a XACML policy defining also object global unique identity and its distinguished version.

3.3 IBE-Enabled Sticky Policy

The proposed solution combines several identity-based encryption (IBE) and policy framework components into a simplified model [123]. XACML policy tightens the security boundary for IBE and constrains the involved parties to trusted and certified parties only. Simplified trust authority (TA) entity is responsible for both policy management as well as for key management, while in actual framework these responsibilities should be handle by two distinguished TAs. It maintains policy templates (see Figure 18) for TA responsible contexts, e.g. internal medical templates, private banking customer templates or human resources external candidate templates.

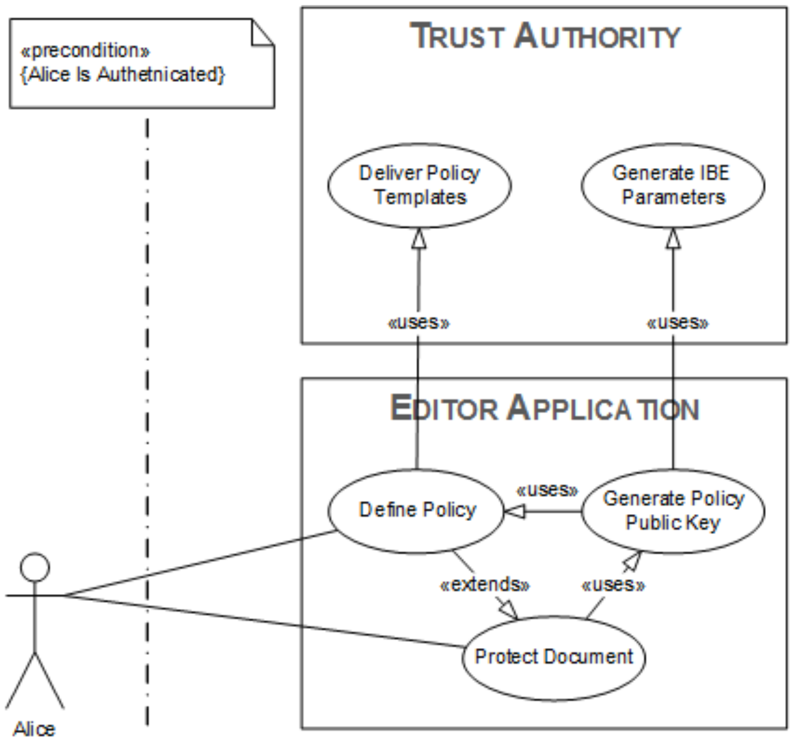


Figure 18 Sticky Policy IBE encryption

Policy templates together with the TA delivers IBE parameters to the editor application required to generate a policy public key. Based on a policy request the TA also makes an access decision or delegates part of the decision to a third-party TA. TA stores master key *{s}* for its domain (see Figure 19) and after positive access request decision it generates a policy private decryption key that is leased to the policy enforcement editor application. Client or server-based editor application handles a read or read-write document access based on the response from the TA.

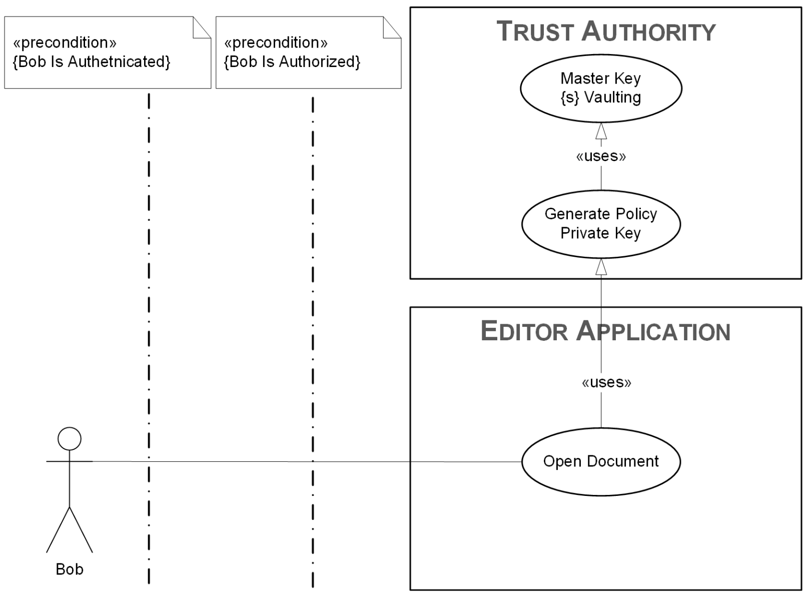


Figure 19 Sticky Policy IBE decryption

How Sticky Policy-enabled OOXML protects files? Authenticated against verified identity provider (IdP) Alice, in order to protect the document, selects preferred trust authority (TA) from a list of registered TAs, this way she receives a template of possible policy rules in a given security context (see Figure 20). After defining policy access rules, the policy set is extended by Alice rights, and together with document global unique identifier and a TA reference, the sticky policy is ready to protect the document.

Alice to encrypt the document using IBE BF [10] setup requires policy public key Q_{POL} therefore she generates

$$Q_{POL} = H_1(POL_{ID}) \tag{1}$$

where H_1 is a hash function defined on the group $\mathbb{G}_1$ of prime order q such as $H_1: \{0,1\}^* \longrightarrow \mathbb{G}_1^*$, which maps sticky policy POL_{ID} into a single point on an elliptic-curve.

Having $H_2: \mathbb{G}_2^n \longrightarrow \{0,1\}^n$ for some n where under bilinear map ê Alice generates random r via random generator P from group $\mathbb{Z}_q = \{0, \dots, q-1\}$ under modulo q and calculates parameters:

$$\begin{cases} U = rP \\ V = e\left(m, H_2\left(\hat{e}\left(R_{pkg}, rQ_{POL}\right)\right)\right) \end{cases} \tag{2}$$

where e is a symmetric AES encryption function over message m and bilinear map ê. r could be generated during initial setup and added to sticky policy as a document unique identifier. Bilinear map ê over Alice's public key Q_{POL} and a TA public key R_{pkg} generated from TA master key. Secret key as per IBE is derived from bilinear mapping ê where ê: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$.

Both AES algorithm and function modulo are symmetric, therefore used in IBE BF [10] modulo operation shown in (3) is replaced with symmetric encryption function (2) e.

$$c = m \oplus H_2\left(\hat{e}\left(R_{pkg}, rQ_{POL}\right)\right) \tag{3}$$

Next, both values U, V are stored inside the OOXML document wrapper and together with the embedded sticky policy are shared in the cloud. Parameters same as other OOXML wrapped data should be protected by cloud provider at rest and in transit.

How Sticky Policy-enabled OOXML opens the protected file? Authenticated by the verified identity provider (IdP) Bob accessing the document presents the policy with the access request to trust authority (TA) using TA reference from the sticky policy. TA takes access decision and assuming its positive TA uses secret master key s and computes private key (see Figure 21) for given sticky policy as follows:

$$S_{POL} = sQ_{POL}, s \in \mathbb{Z}_q \tag{4}$$

Next TA sends policy response together with sticky policy private key S_{POL} to Bob.

Bob can now use symmetric AES decryption function d on parameter V and hash function $H_2: \mathbb{G}_2 \rightarrow \{0,1\}^n$ and decrypt the document as follows:

$$m = d\left(V, H_2\big(\hat{e}(U, S_{POL})\big)\right) \tag{5}$$

The access right specific decision is made by policy framework based on policy response details. However, all possible permissions are interpreted as a read or read-write rights.

3.4 Symmetric Data-Encrypting Key

Within the discussed IBE model a ciphertext is a product of symmetric encryption over a plaintext message and a public key derivate in $H_2: \mathbb{G}_2 \rightarrow \{0,1\}^n$. Hashed value k is a symmetric key equal to:

$$k = H_2\left(\hat{e}\big(R_{pkg}, rQ_{POL}\big)\right) = H_2\big(\hat{e}(U, S_{POL})\big) \tag{6}$$

3.5 Security of Sticky Policies IBE

In evaluated Boneh-Franklin IBE the model security depends mostly on the difficulty of solving bilinear Diffie Hellman problem (BDHP) [10] and also correct parameters selection that is must for making discrete logarithm problem (DLP) hard to solve. Based on the assumption that probability Pr of finding message m using algorithm $\mathcal{A}$ is negligible it has an advantage ϵ defined as:

$$Pr[\mathcal{A}(P, aP, bP, cP) = e(P,P)^{abc}] \geq \epsilon \tag{7}$$

An adversary can get an advantage in the selected model if BDHP is easy despite of DLP security.

Having $\{P, aP, bP, cP\} \in \mathbb{G}_1$, find $e(P,P)^{abc}$ Alice encrypts a document m and selects Bob as a receiver using tailored policy under selected trust authority (TA). Document is shared via cloud services and now only Alice – the data owner and Bob should be able to decrypt the data.

Eve, the adversary, illegitimately obtained the protected document and unpacked the U, V parameters. Therefore, she knows the following:

$$\begin{cases} rP \Leftarrow Q \\ sP = R_{PKG} \\ hP = H_1(POL_{ID}) = Q_{pol} \Leftarrow \exists\, h \in \mathbb{Z}_q, Q_{pol} \in \mathbb{G}_1 \end{cases} \tag{8}$$

Because of (1), (2), (4), (8) having rP, sP and hP Eve now can derive $\hat{e}(P,P)^{shr}$ from the following:

$$\hat{e}(U, S_{POL}) = \hat{e}(rP, sQ_{pol}) = \hat{e}(rP, shP) = \hat{e}(P,P)^{rsh} \tag{9}$$

With BDHP easy to solve we compute message m:

$$m = d\Big(V, H_2((P,P)^{rsh})\Big) \Leftarrow \hat{e}(P,P)^{rsh}, \tag{10}$$

what shows that Eve can get an advantage in the selected model only if parameters are incorrectly selected.

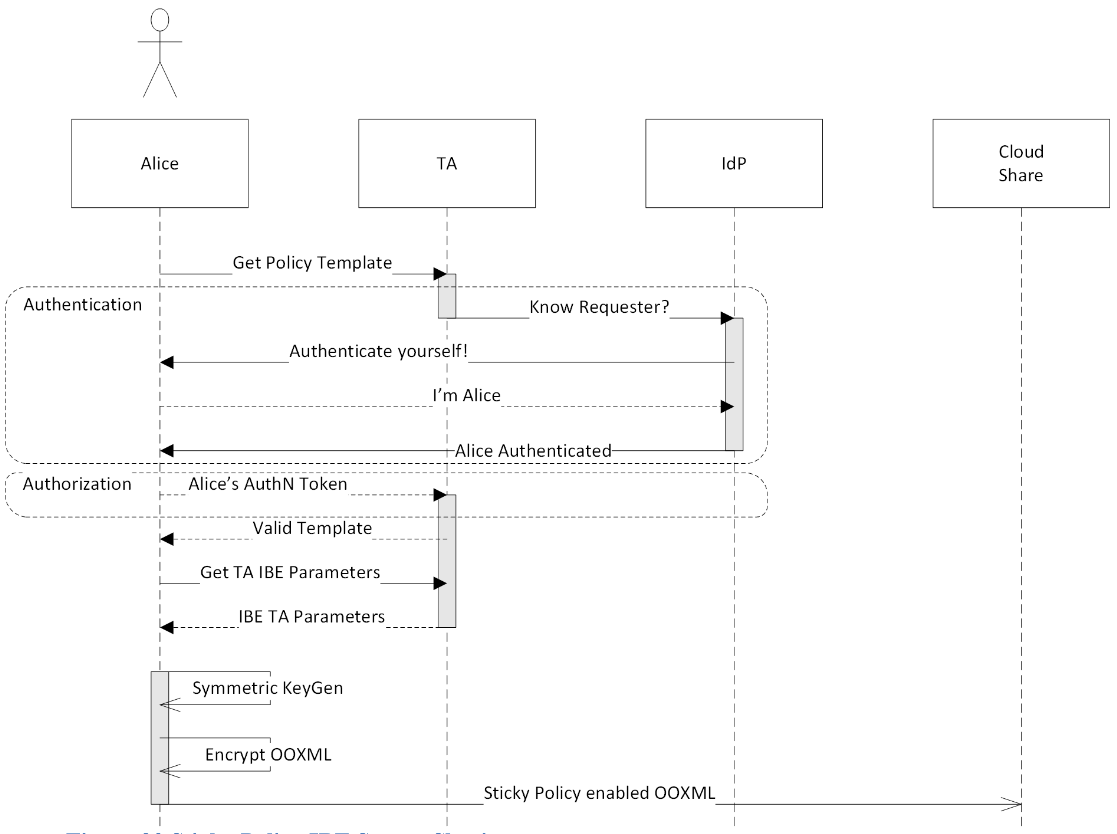


Figure 20 Sticky Policy IBE Secure Sharing

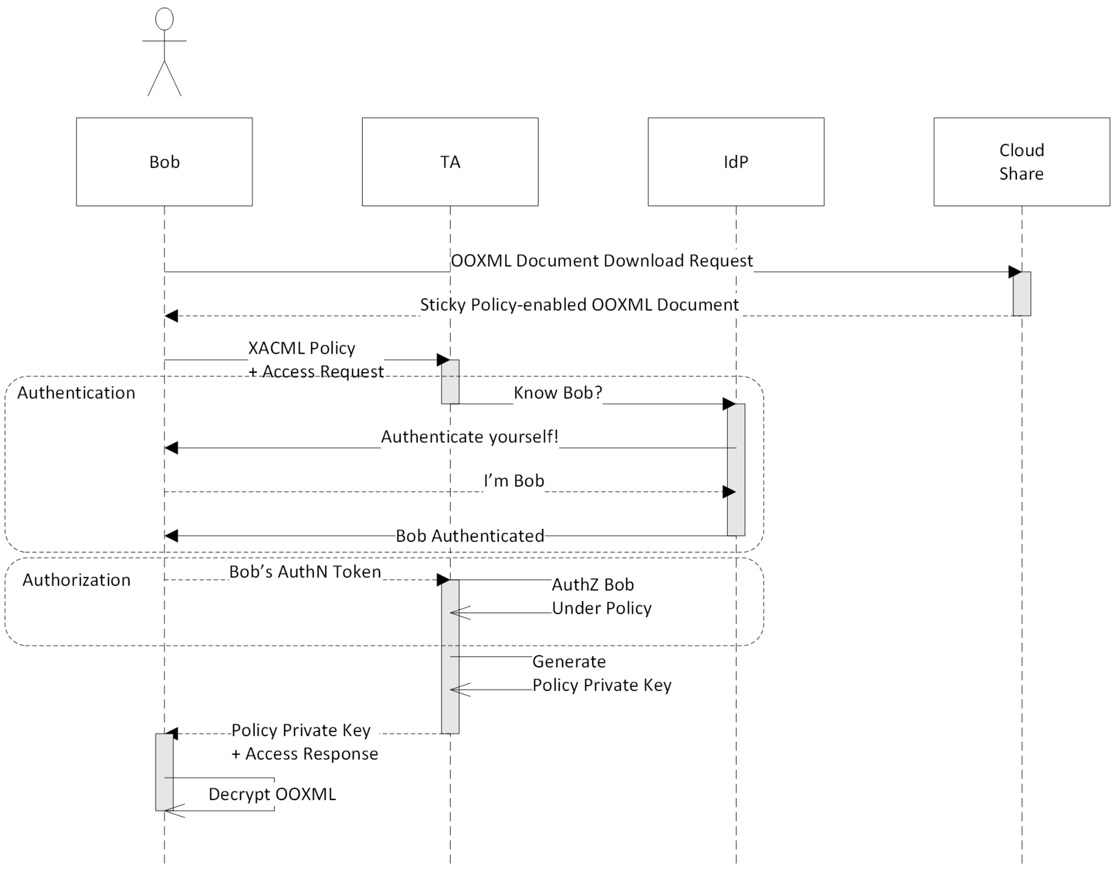

Bob
TA
IdP
Cloud Share
OOXML Document Download Request
Sticky Policy-enabled OOXML Document
XACML Policy + Access Request
Authentication
Know Bob?
Authenticate yourself!
I'm Bob
Bob Authenticated
Authorization
Bob's AuthN Token
AuthZ Bob Under Policy
Generate Policy Private Key
Policy Private Key + Access Response
Decrypt OOXML

Figure 21 Sticky Policy IBE Secure Access

3.6 Sticky Policies IBE Authenticity

An adversary cannot tamper a policy attached to the data using the proposed method. Acting as a public key, the sticky policy is authenticated by IBE scheme. As in any symmetric or asymmetric encryption, only the right key can be used to decrypt the cipher-text. IBE is a public key asymmetric cryptographic primitive therefore for a given public key encrypting the message exists one private key decrypting the cipher-text with this message. If an adversary tries to change the sticky policy attached to the data in this construct after trust authority (TA) authorises the request, the received private key cannot be used to decrypt the cipher-text. Adversary having the advantage in a policy engine authorisation flow, that is, the TA still cannot divulge the message by tampering the sticky policy.

The accepted security notion for the model [78] that could provide data non-repudiation assurance with an extra cryptographic operation is an Authenticated Identity-Based Encryption (Authenticated IBE). It delivers both message confidentiality and non-repudiation on top of IBE scheme [90]. To implement this authorship safeguard either sticky policy or OOXML document metadata should carry information about the data owner. Sender - Alice - using her own private key can authenticate the encryption. Albeit it requires policy private key being leased by the TA during the initial encryption. If data integrity is required, there are existing Identity-Based Signature (IBS) schemes [124].

3.6.1 IBE Signatures

IBE scheme can be immediately converted into a public key signature scheme. The private signing key is the master key *{s},* while the public key is a derivate from public IBE TA parameters. Verification of signature S_m where:

$$S_m = sQ_m, \tag{11}$$

is a result from both encryption of any random message m', e.g.:

$$POL_{ID} \Leftarrow m' \tag{12}$$

under IBE scheme:

$$c' = m' \oplus H_2\left(\hat{e}\left(R_{pkg}, rQ_m\right)\right) \tag{13}$$

and successful decryption using private key S_m as the decryption key:

$$m' = d\left(V, H_2\left(\hat{e}\left(U, S_m\right)\right)\right) \tag{14}$$

This safeguard is more expensive than non-repudiation as requires separate encryption and signing operations, while Authenticated IBE is faster also in compare to actual IBE encryption.

3.6.2 IBE Digital Signature for XACML

Access request and especially response, if to be available across different trust authorities require additional safeguards. XACML response wrapped with SAML object could be signed and securely exchanged between parties [125].

Furthermore, SPIBE leverages the SAML XML signing profile to deliver a private decryption key to policy enforcement point (PEP). Policy decision point (PDP) upon positive access decision sends to PEP not only signed response but also the sticky policy itself. Using fact that IBE digitally signed sticky policy under IBE digital signing is actually a SPIBE data decryption key.

PE after policy decision point (PDP) makes positive access decision, before it is sent to PEP (i.e. BOB), the PE requests PKG to sign the XACML sticky policy. Under IBE signing (see section 3.6) sticky policy signature generation results in deriving a private data decryption key sQ_{POL}. The SAML authorization response would contain not only XACML response but upon positive access decision, the actual XACML sticky policy and its signature (i.e. decryption key).

Finally, for higher security the XACML response could be also signed if other entities (i.e. PEP) require higher authenticity assurance.

3.7 Sticky Policies Authorisation

Sticky policies carry authorisation information required to protect the data. Unlike conventional policy framework where policy is centrally stored and referenced to data, here policy is attached to the data and follows it to enforce access control rules. Policy evaluation upon access request can check *who you are*, *what you have*, *what you know*, *where you are* and *when and how you can access* the data. For example, in countries that adapted OECD data protection directives [53] owner consent related to data access can be represented as an access rule and combined into a policy set. As mentioned before, data access can be constrained by time. For example, a sticky policy added to a financial report would define any subject rights to process the report within a defined time slot and before or after a specific date.

XACML policy defines multiple subjects construct with more than one subject involved in access control decision [97]. This technique implements separation of duties security principle. This non-cryptographic safeguard can have a functional application similar to attribute-based encryption (ABE), or Shamir shared secret [126] concept. While the entire access model document is not cryptographically protected the TA still can reject document access and its decryption if not all policy conditions are met. I.e. document could be accessed only if all subjects agree to open the file.

XACML access request construct represents access tuples, with the subject, object and predicate. The subject is the data owner or data processor who wish to access the object. The object is the resource document that can be represented by cloud data hosting provider path and a unique data identifier. Predicate defines an action that subject is entitled to, based on the policy rules. Because of its internal XML structure, XACML policies are defined via attributes represented by name/value pairs. XACML sticky policy subject can be constrained by a technical Role [127] represented as a group in a target system, where, e.g. Role is equal *BusinessEngineering*. Because sticky policy remains unencrypted, its attribute values could be anonymised or obfuscated as a further safeguard. *BusinessEngineering* role could be represented by a unique global identifier (see Figure 22) from within given trust authority (TA) context. Several attribute-based encryption (ABE) work with an attribute representation using binary-state attributes where attributes unlike in arbitrary-state binary attributes do not directly disclose any information about the content of the protected message. XACML rules may remain in arbitrary-state However, in a form that requires attributes mapping to some predefined encoded unique attributes.

```
<Policy>
    <Rule Effect="Permit">
        <Target>
            <Subject "GROUP(BusinessEngineering):{956EFF…}"/>
            <Resource "TA_URI/{8781F074-FAB1-4D5D-BBF0…}"/>
            <Action "Read"/>
        </Target>
    </Rule>
</Policy>
```

Figure 22 XACML rule example

The obligation is a directive specifying obligatory operation after access request decision. For example, an obligation can instruct to raise a security incident after Eve was denied access to the data. Advice can instruct Bob to use his academic email identity because he does not have a valid educational *ac.uk* domain address. An important feature of both obligations, as well as advice, is the fact that these can enforce data re-encryption under a larger key space or even different cryptographic method. In case of a newly discovered cryptographic vulnerability, the TA upon every policy evaluation request, may send respond to the access subject requiring policy templates update as well as the document re-encryption under a new cryptographic algorithm. TA to respond to the first successful implementation of quantum decryption computer may enforce IBE based on IBE - Isogenous Pairing Groups (IPS), which would result in re-encryption of all the data under new cryptographic safeguard.

Data access control implementation based on XACML sticky policy can efficiently secure confidential information and personal identifiable information (PII), provide high accountability, where single data access attempt is a subject of auditing [85]. Comprehensive implementation of sticky policy model could support advanced security auditing where security breach or a data leakage incident is reported and collected, giving significant evidence for a further legal investigation. The policy construction is highly simplified with policy templates that could be pre-defined by each TA. Policy template can represent a required access evaluation context to be included in the policy (see Figure 23).

```
<Apply xsi:type="AtLeastMemberOf" functionId="urn:oasis:names:tc:xacml:1.0:function:string-at-
least-one-member-of">
    <Apply functionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <AttributeValue ParameterId="location"
DataType=http://www.w3.org/2001/XMLSchema#string/>
    </Apply>
    <AttributeDesignator AttributeId="http://schemas.tscp.org/2012-03/claims/ISO-3166-2"
DataType="http://www.w3.org/2001/XMLSchema#string" />
```

Figure 23 Policy template part for the location-based access rule

Policy template (see Figure 24) will use attribute designators to set the correct rules in the right context, e.g. country jurisdiction.

```
<Parameter ParameterId="location">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">UK</AttributeValue>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">PL</AttributeValue>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">CH</AttributeValue>
</Parameter>
```

Figure 24 Policy template data representation defining access location in ISO 3166-2 for attribute designator

Constrained XACML policy template simplifies policy generation and reduces complexity on the client-side allowing only a pre-defined set of rules to be configured.

XACML policy defines which individual or group of individuals in what configuration (i.e. location, time) can be granted permissions to access the protected data in the cloud. Sticky policies implemented based on XACML suffice wide range of access control implementations. This includes modern cloud-enabled authentication and authorisation frameworks which leverage SAML and claims-based authentication like in [128] where individual prior to authorisation would be a subject of authentication (see Figure 20 and Figure 21) involving third-party identity providers (IdP).

3.8 Trusted Parties

Sticky-policies make use of trust authorities (TA), which validate compliance with policies in order to lease decryption keys. Policies likewise cover data owner consent give subject rights to process data. A model where a TA has to be contacted by the service provider (SP) to access the PII data delivers high accountability as each personal data access attempt is a subject of auditing [85] and could be tracked in case of a data leakage incident. The data owner can then feel that they own the data released into the cloud because of not only the policies associated with data following data owner approval, but also for the TA, which specifies where the policy can be interpreted and is pre-selected by the data owner [121]. Information about the TA is attached to the policy and is passed to the SP. An XML schema that can store sticky-policy definition can be easily integrated into identity metadata.

TA as an abstract entity needs to represent one or more actual trust parties that take part during initial authentication, authorization, key management and actual document management. Different TAs could exist independently and provide service do many other service providers, not necessary related to SPIBE. All trusted parties as recommended for Policy Machine architecture [55] have to authenticate one another. SPIBE recommends that trusted parties are certified and authenticated but that the actual sticky policy is signed. Furthermore, to the separation of duties principle recommends that one party should not be able to compromise model security.

The private key generator (PKG) [129] is the most sensitive TA as it could deliver decryption keys for every given public key. It is important to inspect the PKG system and its logs to ensure only that certified PKG implementations are used and only legitimate decryption requests are processed. The PKG should never hold the same master key for two independent frameworks, this will eliminate scenarios where private data decryption key is illegitimately generated without prior authorization. This is a part of core SPIBE architecture, however there are other safeguards required that would ensure framework compliance. It is necessary to allow a public audit through third party auditor (TPA) to certify all the trusted parties.

Policy engine and its sub-entities cannot decrypt data directly, however incorrect false positive access decisions made could also compromise security of the SPIBE model. Same as PKG it requires external TPA to verify whether the provider delivers secure service.

3.9 Comparison

Most of the currently studied IRM solutions are primarily focused around health care record privacy protection. Among them SPIBE has several distinguished features giving it functional advantage over other research works. Use OOXML for the actual implementation makes this solution adaptable by any organization or institution, or even individual using popular wordprocessing applications. Furthermore, CP-ABE based solutions [130], [131], [132], [133] , [134], where policy does not contain globally uniquely identified document with an incorrect setup could lead to situation that CP-ABE keys are same for different documents. When one of the keys is divulged it could compromise several unrelated documents. Finally, most of the compared solutions (see Table 1) do not consider quantum computing as a threat therefore it does not provide relevant safeguards. It is only possible to assume that research work that does not refer to any specific cryptographic primitive leaves space for cryptographic quantum hardening.

Solution	Strength	Weakness	Safeguards	Cloud-ready	Access Control	Cryptography	Confidentiality	Integrity	Accountability	Non-repudiation	Standards-based	Implementation	Quantum ready
CHISTAR [135]	Interoperability, scalability, maintainability	Inflexible access control	RBAC; AES-256; SSL; MAC; SSO	✓	✓	✓	✓	✓	✓	✗	✓	✓	N/A
VistA [136]	none	Client-server architecture	RBAC	✗	✓	✗	✗	✗	✓	✗	✓	✓	N/A
G. Hsieh and R.-J. Chen [130]	Flexible access control and integrity control	Lack of prototyped architecture	ABAC(XACML); XML Security (on policy); AES; CP-ABE	✓	✓	✓	✓	✓	✓	✓	✗	✗	N/A
F. Rezaeibagha and Y. Mu [131]	Scalability, confidentiality, and secure data outsourcing	Data sharing problems can be caused by increasing the complexity of EHR data policies	RBAC; CP-ABE	✓	✓	✓	✓	✗	✓	✗	✗	✗	N/A
U. Premarathne et al [137]	Perform access control through context and location awareness	Key exchange problems between various parties	RBAC; PKI	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗
M. Peleg et al [138]	Structured specification of patient data access scenarios via situation models	Scalability issues	Situation-based access control	N/A	✓	✗	✗	✗	✓	✗	✓	✗	N/A
R. Gajanayake et al. [139]	Combines three existing access control models	none	MAC; DAC; RBAC; PBAC	N/A	✓	✗	✗	✗	✓	✗	✗	✓	N/A
A. Lunardelli et al [140]	Provides solution to the situation of policy conflict	Lack of security aspect issue	ABAC(XACML)	N/A	✓	✗	✗	✗	✓	✗	✗	✓	N/A
J. Calvillo-	Flexible access control	No consideration of	ABAC(XACML)	N/A	✓	✗	✗	✗	✓	✗	✓	✗	N/A

Arbizu et al [141]		confidentiality or integrity issues											
P. Gope and R. Amin [142]	Practicality, Robustmess	Inflexible access control	ABAC, MAC	✓	✓	✗	✓	✗	✓	✓	✗	✗	N/A
S. Alshehri, S. P. Radziszowski, and R. K. Raj et al. [132]	High performance over time overhead and storage overhead	Lack of non-repudiation	ABAC; ECC; CP-ABE	✓	✓	✓	✓	✓	✓	✗	✓	✗	N/A
K. Yang et al [143]	Flexible access control, dynamically changing user attributes	Lack of implementation	ABAC; Time-domain ABE	✓	✓	✓	✓	✗	✓	✗	✗	✗	N/A
A. Mohandas And S. S [133]	Fine grained access control, anonymization	None	ABAC; CP-ABE; k-anonymization	✓	✓	✓	✓	✗	✓	✓	✗	✓	N/A
T. Neubauer and J. Heurix [144]	Provides a methodology for the pseudonymization of medical data	No cover for digital signature	Pseudonymization; RSA-2048; AES-256	N/A	✓	✓	✓	✓	✗	✗	✓	✓	✗
M. T. SandIkkaya et al [145]	Performed the pseudonymization and can break-glass procedures; self-protect the data in case of breached access using biometrics	Inflexible access control	Encryption signature; RBAC; pseudonimization	N/A	✓	✓	✓	✓	✓	✓	✗	✓	✓
S. Sharma and V. Balasubramanian [146]	Self-protect the data in case of breached access using biometrics	none	Biometrics Encryption; SHA-256	✓	✓	✓	✓	✗	✓	✗	✗	✓	✗
R. Au and P. Croll [147]	Considers various factors for privacy protection	Inflexible access control	Pseudonymization; PKI; RBAC; digital signatures	N/A	✓	✓	✓	✓	✗	✓	✓	✗	✗
K. Seol et al [134]	Supports all evaluative requirements	none	ABAC(XACML); XML Security	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
SPIBE	Integrates into common word processing systems; resistant to non-or-full access rights vulnerability	none	ABAC(XACML); AES-256; IBE; IBE-IPG; CP-ABE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1 Comparison with existing privacy preservation studies in e-health [134].

3.10 Conclusions

Sticky policies in the described cryptographic setup carries the potential to decentralize data access definitions for modern applications. With IBE the entire construct is not limited to one specific cryptographic algorithm implementation. It could quickly adapt to new requirements not only by increasing key sizes but also by replacing actual cryptographic primitives. Under IBE-IPG the actual model is quantum computing ready and with sticky policy evaluated under a TA, the security upgrade would only require re-encryption under a new cryptographic protocol without actual change to entire IRM architecture. Having several certified TAs separating core responsibilities including authentication, authorization, PKG and key management system (KMS) ensures that the SPIBE framework cannot be compromised at a single point. Single TA can not have power to decrypt all protected documents without having legitimate request approved by all trusted parties, i.e. TAs. The added blockchain part would efficiently authenticate all the changes complementing model with document changes authenticity, non-repudiation, and integrity. OASIS authentication standards with possible XACML profiles show the selected sticky policies model has solid foundations.

4 Evaluation Methodology and Implementation

4.1 Introduction

Evaluation is focused on selected architecture components. Model evaluation has been prepared to see how the rights could be enforced and visible to the end user. The OOXML document wrapper was configured as a single standalone document and as a master document constructed with several subdocuments. The master document evaluation aims to show how a group of authors could collaborate in writing a single report made out of several subdocuments. Unlike the single document, the master document could be leveraged to simplify access management over different document elements. Otherwise, a single document required a custom approach if granular access authorisation to different document parts is required. The XACML policy embedding and unpacking into and from OOXML document is not a core part of the evaluation. However, it was added and removed for each of the other tests performed. The JSON formatted XACML policy comparison with XML format has to prove that same policy could be expressed to save the storage space as well as potentially suit future encryption schemes with relatively larger keys that are limited by plain text message length.

By implementing several XACML policy engine components, it is possible to identify problematic points and recognise advantageous and disadvantageous of both PEP designs with the fat-client application and with the web-based application.

SPIBE has been compared to other popular IRM solution from Microsoft, the Azure RMS and the IONIC Secure Files. Azure RMS same as SPIBE uses both asymmetric and symmetric encryption. IONIC is based only on symmetric AES encryption. Both other solutions show strong and weak sides of symmetric and asymmetric cryptography based IRM approach. Therefore, the comparison with SPIBE helps in to find how the construct could get the best from existing solutions.

4.2 Architecture

The evaluation of XACML sticky policy with IBE (SPIBE) has been conducted on a simple, single domain setup (see Figure 25). This is sufficient to see IBE applicability for sticky policies as well as a potential of leveraging OOXML as an efficient data wrapper. XACML entities could be deployed in a standalone or a distributed model. By distributing authorisation entities, dedicated components have to handle authentication to ensure request and response non-repudiation. Simplified policy engine (i.e. AUTHZ) delivers XACML policy templates within a given context. Assuming AUTHZ is deployed by an organisation with strictly defined access policies, policy templates could be retrieved from policy information point (PIP) based on relationship type between identity and organisation. OOXML editor application (APP) extended with a custom plug-in for embedded XACML policy management could retrieve the correct templates based on the authenticated user using OAuth or SAML tokens. For the evaluation purposes, XACML templates management has not been implemented on AUTHZ component nor on the APP side. XACML embedding functionality that is handled by OOXML plug-in is prototyped (see Appendix O and Appendix X) to pack and unpack predefined XACML policies.

The editor application could be deployed either at a client-side or at a web server side. Both APP implementations were evaluated. The first APP proof of concept (PoC) used Microsoft Word as the editor, where access decisions were enforced using file metadata permissions. Client-based editor implementation is a challenging part of the evaluation. There are various OOXML compliant editors available. In order to deliver a comprehensive solution, the client-side extension should support various OOXML processing suites. An implementation that would support all different applications including different versions would require the complete understanding of the application itself. In many cases, vendors do not deliver libraries or software development kit that would support such a complex integration. Therefore, due to limited supervision over an OOXML document and the policy the approach requires control at the low driver level. Under Microsoft Windows system the handler and buffer for file access management has to implement Universal Windows Driver (UWD) at the Minifilter Driver level. The only bottleneck of the solution is that driver has to be designed for the exact Windows distribution and CPU architecture. Despite these drawbacks, the driver-based implementation seems more consistent and secure as all cryptographic operations are performed at the lower level where any potential key tampering is relatively limited. It requires a skilled individual to take an advantage within the selected architecture.

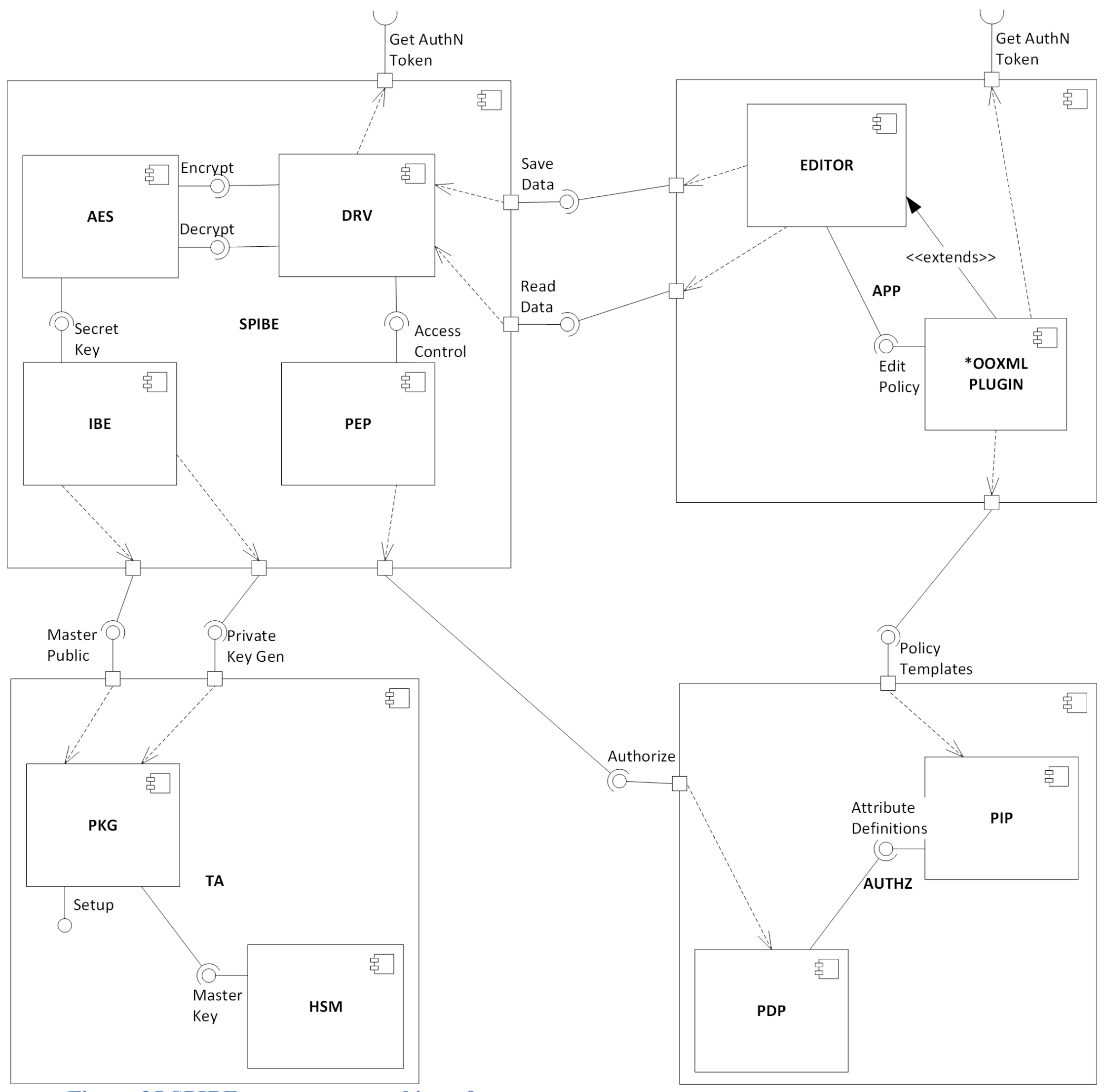


Figure 25 SPIBE components and interfaces

The second APP component PoC was conducted using the limited functionality of the online web-based OOXML editor (see Appendix Y) with both SPIBE and APP integrated as one component. Unlike Minifilter driver solution, the web-based editor could be hardly tampered to give adversary advantage in SPIBE. Direct access to memory that would give access to extract keys used to protect the document like in MS RMS [148] is limited or simply impossible as the hardware is not under control of the data processor.

4.3 Model Implementation

4.3.1 Introduction

SPIBE system component (see Figure 25) has been evaluated without modular implementation that is required to deliver fully functional customer ready framework. Only mandatory modules required for evaluation were prototyped and integrated. AES, IBE, PKG components were implemented together under a single program tool reading configuration from files stored under local file system. KMS and HSM have not been used. DRV module has been implemented as a standalone driver library, however integration with existing IBE libraries was not possible with current Visual Studio solution architecture. PEP, PIP together with PDP were running under separate solution instance and only the actual XACML policies were shared with IBE component over a file system. The EDITOR application was implemented as HTML editor, however only Microsoft Word successfully interpreted rights set on the file IO level (see Figure 40). OOXML PLUGIN component has not been required for the actual evaluation as all the XACML policies were pre-generated for actual RSA and IBE performance evaluation.

4.3.2 Policy Encapsulation

The main policy wrapper challenge is to allow efficient and visible to end-user access rights enforcement. While custom OOXML editor application would deliver the most comprehensive solution, its maintenance over years would become a bottle neck of the framework. That is why architecture considers existing Microsoft Office SDK or open source popular libraries. Microsoft libraries work relatively stable with Microsoft products and there are many other systems designs has to consider giving it higher priority due to main objectives, making actual framework standardized for secure cloud data sharing. The second way to handle OOXML wrapper is to leverage HTML based application, where the access to the document content would happen on a trusted sever side. While in a fat-client scenario the document access does not require high machine power, central server-side implementation needs very efficient back-end that would handle hundreds of thousands simultaneous read/write requests as well as access decision requests. Therefore, for server-side application it the SPIBE considers use of either flat databases that actually host basic document descriptor attributes or classic relational database. While the document descriptors might vary from the long-term database maintenance perspective it will be beneficial to use semi-structured No-SQL document database. For this research, however, we use transactional database with FileStream feature enabled. Databases work well with document data stored directly with other data, but it also depends on the size of the actual document and the frequency of access attempts. FileStream is not the best solution to handle large amount of small frequent updates, however OOXML is on the border where document is smaller than 1[MB] suitable for Binary Large Objects (BLOB) table field and document storing rich media meta-data highly exceeding 1[MB] is suitable to be hosted directly by the file system. Due to encrypted content that will be stored the direct updates are not recommended as it will cause various performance issues, where changed data before saving always has to be encrypted in the buffer. Actual initial access to the stored document should be highly efficient but the application need to use a cached document version in order to optimize actual operations on the document and align with overall database performance capacity. Modern No-SQL databases, like MongoDB have already various features that improve file access performance, therefore the used underlying database type will highly depend on the target market and the scale of the implementation. OOXML files, when encrypted cannot be efficiently hosted as an integrated part by semi-structured database, however the XACML policy and other unencrypted document meta-data could be efficiently stored and indexed in XML format. The entire document should be referenced directly from files. Same time, the policy before saved together with a document could be deserialized by the object layer and exported as a normalized set of attributes to be finally stored in the relational database next to the BLOB data-type.

The first part of the implementation reviewed possible ways of efficient storing, managing, opening and controlling access to OOXML document. To simply encapsulate and extract XACML policy into and from the OOXML document and to evaluate how the OOXML ZIP format aligns with standard libraries the basic compression functions were used (see Figure 26 and Figure 27).

```
#*******************************************************************************
#!/usr/bin/python


#************************************Functions**********************************

#************************************
#zip_word_doc_path encapsulates doc section into OOXML file
#************************************
def zip_word_doc_path(full_doc_path, full_word_doc_path, word_doc_path):
      with zipfile.ZipFile(full_doc_path, 'w') as zip_file:
        zip_file.write(full_word_doc_path, word_doc_path, zipfile.ZIP_STORED)
```

Figure 26 XACML policy encapsulation, Python

```
#*******************************************************************************
#!/usr/bin/python


#************************************Variables**********************************
CONST_WORD_DOC_PATH = 'word/document.xml'

#************************************Functions**********************************
#************************************
#extract_word_doc_path extracts doc section from OOXML file
#************************************
def extract_word_doc_path(full_doc_path):
      file_handle = open(full_doc_path, 'rb')
      with zipfile.ZipFile(file_handle) as zip_file:
        for arch_elem in zip_file.namelist():
                if arch_elem.startswith(CONST_WORD_DOC_PATH):
                        zip_file.extract(arch_elem)
```

Figure 27 XACML policy extraction, Python

For the initial evaluation, the policy enforcement point (PEP) has been implemented as a part of a Web-based editor application with policy related to a document as a reference. The application was editing OOXML word processing content after policy decision was made therefore access rights were already determined during the initial file access (see Figure 28).

```
/// <summary>
/// Creates an instance of a single OOXML document supporting basic access control
functionality
/// </summary>
/// <param name="singleStream">Document file stream</param>
/// <param name="fileAccess">Access type</param>
public XDocument(SingleStream singleStream, FileAccess fileAccess)
{
      this.singleStream = singleStream;
      this.WrapDocument(fileAccess);
}
```

Figure 28 XDocument Constructor, C#

Documents were stored in SQL FileStream for efficient access (see Figure 29). Every data request was followed by XACML policy retrieval.

```
private SqlFileStream GetData()
{
      const string SQL_TRANS_QUERY = @"SELECT GET_FILESTREAM_TRANSACTION_CONTEXT()";
            //byte[] buffer;
            //UInt32 position = 0;

      string sqlQuery = String.Format(@"
SELECT TOP 1
      [MetaDataFile].PathName()
FROM
      [NEHST].[dbo].[MetaData]
WHERE
      [MetaDataID] = '{0}'", this.metaDataID);
      if( this.fileStreamer.SqlConnection.State == System.Data.ConnectionState.Closed)
      {
        this.fileStreamer.SqlConnection.Open();
      }
      using (SqlCommand sqlCommand = new SqlCommand(sqlQuery,
this.fileStreamer.SqlConnection))
      {
        //using (SqlTransaction sqlTransaction
        this.sqlTransaction =
this.fileStreamer.SqlConnection.BeginTransaction(this.metaDataID.Replace("-", String.Empty));
        sqlCommand.Transaction = this.sqlTransaction;

        string filePath = (string)sqlCommand.ExecuteScalar();
        //SetRemoteSecurityContext(filePath);

        sqlCommand.CommandText = SQL_TRANS_QUERY;

        this.streamHandle = (byte[])sqlCommand.ExecuteScalar();
        return new SqlFileStream(filePath, this.streamHandle, this.fileAccess);
      }
}
```

Figure 29 Opens SQL FileStream with OOXML content, C#

This approach is suitable for HTML-based OOXML editing, however, to use Microsoft Office SDK the regular file stream suits better the `DocumentFormat.OpenXml.Wordprocessing` library.

4.3.3 Cryptography

The core evaluation of IBE-BF and RSA is based on a model construct that allows key-encrypting key operations. However, the complete end to end SPIBE cryptographic operations including AES256 data encryption are implemented. The main objective for evaluation was to create a single consistent solution where programming languages would not have impact on the overall performance measures. Decision regarding programming language for the actual solution is crucial because it has to consider further consequences when it comes to integration with different SPIBE components. If policy enforcement point (PEP) application would be running on Unix machine there should be existing not only set of OOXML editing libraries but also XACML libraries together with all cryptographic primitive implementations that could work under Unix system. If the editing application should have option to run as a client or server-side solution it is important to make underlying libraries generic written in the same programming language and configured for one single system. Such an approach simplifies the maintenance where critical changes could be quickly tested and deployed without need to maintain functionally identical solutions separately because of the programming architecture limitations.

Cryptographic libraries implemented as under Visual Studio solution allow further integration with XACML PEP component, but the aim is that popular MS based OOXML editor application part could be easily evaluated under the same software architecture.

There are two main C++ evaluation methods, one `ibe_eval` for IBE-BF with Sticky Policy mapping into key space and the other `rsa_eval` for RSA evaluation (see Appendix H). Actual implementation C++ methods could wrap the underlying C cryptographic libraries and expose them for all other programming language projects under one single Visual Studio solution.

```
static int ibe_bf_set_public_key(const unsigned char *id, long id_size, unsigned char *key,
const int key_size, char *err)
{
	const int HASH_LEN = 32;
	unsigned char hash[HASH_LEN] = { 0 };

	key = (unsigned char *)malloc(key_size+1);

	if (SHA256(id, id_size, hash) == NULL)
	{
	  ERR_error_string(ERR_get_error(), err);
	  printf("%s\n", err);

	  return -1;
	}
	for (int i = 0; i < key_size; i++)
	{
	  key[i] = hash[i % HASH_LEN];
	}
	key[key_size] = '\0';

	return strlen((char *)key);
}
```

Figure 30 XACML Policy mapped into Public Key space via SHA256, C

```
static void ibe_eval(int argc, char **argv)
{
	//...
	for (int i = 0; i < 100; i++)
	{
	  /********
	   ++++BEG_TIMING
	   *********/
	  QueryPerformanceCounter(&t1);
	  /********
	   ----BEG_TIMING
	   *********/
	  element_from_hash(mapped_id_hash_Qid, key, pkey_sz);
	  /********
	   ++++END_TIMING
	   *********/
	  QueryPerformanceCounter(&t2);

	  time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC /
frequency.QuadPart;
	  printf("%f\n", time_spent);
	  fprintf(file_key_gen, "%f;[ms]\n", time_spent);
	  /********
	   ----END_TIMING
	   *********/
	}
	//...
}
```

Figure 31. IBE-BF Public Key generation evaluation, C

```
static void ibe_eval(int argc, char **argv)
{
	//...
	for (int i = 0; i < 100; i++)
	{
	  /********
	   ++++BEG_TIMING
	   *********/
	  QueryPerformanceCounter(&t1);
	  /********
	   ----BEG_TIMING
	   *********/
	  element_mul_zn(Ppub, gen_P, master_key_s);
	  element_printf("++s: %B\n", master_key_s);
	  element_printf("++P:  %B\n", gen_P);
	  element_printf("++Ppub: %B\n", Ppub);

	  /********
	   ++++END_TIMING
	   *********/
	  QueryPerformanceCounter(&t2);

	  time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC /
frequency.QuadPart;
	  printf("%f\n", time_spent);
	  fprintf(file_key_gen, "%f;[ms]\n", time_spent);
	  /********
	   ----END_TIMING
	   *********/
	}
	//...
}
```

Figure 32 IBE-BF Private Key Generation Performance Evaluation, C

```
static RSA *rsa_create_key_pair(unsigned char **public_key, int *public_key_size, unsigned
char **private_key, int *private_key_size)
{
	const int KEY_SIZE = 1024;
	const int PUB_EXP = 3;
	RSA *key_pair;

	key_pair = RSA_generate_key(KEY_SIZE, PUB_EXP, NULL, NULL);

	BIO *bio_private_key = BIO_new(BIO_s_mem());
	BIO *bio_public_key = BIO_new(BIO_s_mem());

	PEM_write_bio_RSAPrivateKey(bio_private_key, key_pair, NULL, NULL, 0, NULL, NULL);
	PEM_write_bio_RSAPublicKey(bio_public_key, key_pair);

	*private_key_size = BIO_pending(bio_private_key);
	*public_key_size = BIO_pending(bio_public_key);

	*private_key = (unsigned char *)malloc(*private_key_size);
	*public_key = (unsigned char *)malloc(*public_key_size);

	BIO_read(bio_private_key, *private_key, *private_key_size);
	BIO_read(bio_public_key, *public_key, *public_key_size);

	return key_pair;
}
```

Figure 33. RSA key pair factorization, C

```
static void rsa_eval()
{
	//...
	for (int i = 0; i < 100; i++)
	{
	  /********
	  ++++BEG_TIMING
	  *********/
	  QueryPerformanceCounter(&t1);
	  /********
	  ----BEG_TIMING
	*********/

	  int size_enc = RSA_public_encrypt(AES_KEY_SZ, aes_key, cipher, key_pair,
RSA_PKCS1_PADDING);
	  /********
	  ++++END_TIMING
	  *********/
	  QueryPerformanceCounter(&t2);
	  /********
	  ----END_TIMING
	  *********/

	  time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC /
frequency.QuadPart;
	  printf("%f\n", time_spent);
	  fprintf(file_key_gen, "%f;[ms]\n", time_spent);
    }
	//...
}
```

Figure 34 RSA key-encrypting key encryption evaluation, C

```
static void rsa_eval()
{
       //...
       for (int i = 0; i < 100; i++)
       {
         /********
          ++++BEG_TIMING
          *********/
         QueryPerformanceCounter(&t1);
         /********
          ----BEG_TIMING
          *********/
         int size_dec = RSA_private_decrypt(size_enc, cipher, aes_key_v, key_pair,
RSA_PKCS1_PADDING);

         /********
          ++++END_TIMING
          *********/
         QueryPerformanceCounter(&t2);
         /********
          ----END_TIMING
          *********/
       }
       //...
}
```

Figure 35 RSA key-encrypting key decryption evaluation, C

The cryptographic evaluation is mostly focused on key-encrypting key cryptographic operations. For IBE-BF the public key is derived directly from the policy (see Figure 30, Figure 31 and Figure 32) therefore there is no need to perform key-encrypting key operations like in RSA, where asymmetric keys have to be derived from prime numbers factorization (see Figure 33) and symmetric data-encrypting key encrypted or decrypted under RSA (see Figure 34 and Figure 35).

The further IBE-BF evaluation work itself is focused more on simple model workflows and looking at the actual performance of overall SPBIE cryptographic operations (see Figure 36, Figure 37, Figure 38). Here the policy into key space mapping under IBE-BF and data encryption using AES256 are evaluated together in various setups.

```
static int ibe_bf_aes256_encrypt(element_t r, element_t U, element_t P, element_t gid,
element_t mapped_id_hash_Qid, element_t Ppub, unsigned char *data, long data_len, unsigned
char *cipher, char *err)
{
	const int HASH_LEN = 32;
	char hash[HASH_LEN] = { 0 };
	unsigned char *gs;

	element_random(r);
	element_mul_zn(U, P, r);

	element_pairing(gid, mapped_id_hash_Qid, Ppub);

	element_pow_zn(gid, gid, r);
	gs = (unsigned char*)malloc(element_length_in_bytes(gid));

	element_to_bytes(gs, gid);

	if (SHA256((unsigned char*)gs, element_length_in_bytes(gid), (unsigned char *)hash) ==
NULL)
	{
	  ERR_error_string(ERR_get_error(), err);
	  printf("%s\n", err);
	}
	unsigned char iv[128] = { 0 };
	int cipher_len = aes_evp256_encrypt((unsigned char*)data, data_len, (unsigned
char*)hash, iv, cipher);

	free(gs);

	return cipher_len;
}
```

Figure 36 IBE-BF encryption over AES256, C

```
static void ibe_eval(int argc, char **argv)
{
	//...
	for (int i = 0; i < 100; i++)
	{
	  /********
	   ++++BEG_TIMING
	   *********/
	  QueryPerformanceCounter(&t1);
	  /********
	   ----BEG_TIMING
	   *********/

	  //ENCRYPTION
	  ibe_encrypt(r, U, P, gid, mapped_id_hash_Qid, Ppub, data, data_sz, cipher, err);
	  cipher_sz = ibe_bf_aes256_encrypt(r, U, gen_P, gid, mapped_id_hash_Qid, Ppub, data,
data_sz, cipher, err);
	  char *b64MsgHash;
	  to_base64(cipher, data_sz, &b64MsgHash);
	  element_printf("++m: %s\n", b64MsgHash);

	  /********
	   ++++END_TIMING
	   *********/
	  QueryPerformanceCounter(&t2);
	  ///********
	  //----END_TIMING
	  //*********/

	  time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC /
frequency.QuadPart;
	  printf("%f\n", time_spent);
	  fprintf(file_key_gen, "%f;[ms]\n", time_spent);
	}
	//...
}
```

Figure 37 IBE-BF decryption over AES256 evaluation, C

```c
static int ibe_bf_aes256_encrypt(element_t r, element_t U, element_t P, element_t gid,
element_t mapped_id_hash_Qid, element_t Ppub, unsigned char *data, long data_len, unsigned
char *cipher, char *err)
{
        const int HASH_LEN = 32;
        char hash[HASH_LEN] = { 0 };
        unsigned char *gs;

        element_random(r);
        element_mul_zn(U, P, r);

        element_pairing(gid, mapped_id_hash_Qid, Ppub);

        element_pow_zn(gid, gid, r);
        gs = (unsigned char*)malloc(element_length_in_bytes(gid));

        element_to_bytes(gs, gid);

        if (SHA256((unsigned char*)gs, element_length_in_bytes(gid), (unsigned char *)hash) ==
NULL)
        {
          ERR_error_string(ERR_get_error(), err);
          printf("%s\n", err);
        }
        unsigned char iv[128] = { 0 };
        int cipher_len = aes_evp256_encrypt((unsigned char*)data, data_len, (unsigned
char*)hash, iv, cipher);

        free(gs);

        return cipher_len;
}
```

Figure 38 IBE-BF decryption over AES256, C

4.3.4 Policy Engine

XACML policy engine was evaluated with 400 XACML policies to receive a valid access response (see Appendix P) and to see the actual policy decision point (PDP) behaviour. All policy engine components but policy enforcement point (PEP) could be easily deployed under single system architecture using the same programming language. The PEP if deployed on a client-side it brings many challenges. To simplify actual SPIBE deployment it might be easier to create server-side PEP first with web-based editor application. If PEP should be deployed to client, the actual cryptographic operations should be handled on as low level as possible to avoid any possibility of intercepting the keys and making illegitimate changes to the protected document and the access policy. The Windows system Minifilter driver acting as a PEP has been partially implemented to see the possible use cases and the implementation complexity (see Appendix JJ).

By comparing XML and JSON formatted XACML policy sizes it is possible to see some benefits of using JSON formatted expressions as well as possibilities of using some cryptographic primitives with larger keys that could additionally encrypt the policy to increase the model security. Larger mapping key space for IBE BF close to policy size would reduce possible collisions in the long term. Some cryptographic primitives limit the size of the plaintext to the key size. If encryption of the actual XACML policy should be considered it is important to see what the largest and the average policy size is. This part of the evaluation may have many implications for the future work and should be beneficial for other researchers working on similar problems.

5 Evaluation

5.1 Introduction

Evaluation of the SPIBE model is based on research progress where initial policy and later sticky policy implementation components, where possible, were tested and compared to other functional alternative components (see Table 3). Solutions simplicity was favoured over complexity. At the same time, easily adaptable standards that were previously successfully implemented on a global scale were preferred over homogeneous and closed systems. Finally, used algorithm security lifecycle (ASL) was considered as the highest priority due to IRM purpose and the emerging quantum computing. The concept of cloud-based data sharing will be only possible when open standards are used the construct is following Kerckhoff's principle for protection algorithms. The principle states that a cryptographic algorithm could be known to leave protected data safe under unknown cryptographic keys. Cloud protection cannot rely on corporate trade secrets as the only true cloud implementation enables different parties to provide the same standardised services.

5.1.1 Microsoft Rights Management Services (RMS)

Different available information rights management (IRM) products are currently available on the market, however, where data protection is a core competency either very homogenous custom solution is used to protect documents or companies use products from market leading vendors. In this group, Microsoft Rights Management Services (RMS) took over other similar products establishing its high position in this market niche providing services, e.g. to UK Ministry of Defence [148] but also to international financial institutions, governments, health-care institutions, and large enterprises. Although Microsoft has managed to introduce its product as a cloud service its internals seem, have never been changed except couple minor adjustments to make it more interoperable.

5.1.2 IONIC Security

On the cloud market, there are different information rights management (IRM) products that more or less successfully conquer legacy markets. However, their advantage over predecessors is the proper use of modern application programming interfaces (API) and cloud-ready standards. Among these IONIC products entered the existing large enterprises market successfully collaborating with Microsoft on empowering the cloud-based Azure suite [149]. IONIC seem to deliver a complete software development kit (SDK) for its products including documentation for multi-platform integration via API's and different programming languages.

5.2 Access Policy vs Sticky Policy

Sticky policy, unlike access policy, is attached as additional document metadata. Sticky-policies integrate into existing policy frameworks. It has a major advantage over another approach where the policy is kept separate from the object. It combines both a policy and an object (resource) under sticky policies model similar to discretionary access control (DAC) model. It reduces the number of model entities and also increases the database access response (see Figure 39). In transactional databases query response time t_p is equal to natural logarithm of total records number.

$$t_p = \ln(a), \tag{15}$$

In access policy-based access control model, implementation database maintains not only a document information, which is claimed by the subject, but it also maintains access policies. XACML-like policy keeps a reference to an actual resource, i.e. document. However, in access request scenario subject claims object based on resource information before the policy is evaluated. One can calculate query time t_p assuming we have to query policy each out of p policies and each document out of n documents separately as in:

$$\forall a = p \times n, \tag{16}$$

$$t_p = \ln(a) = ln(n) + \ln(p) \tag{17}$$

In the sticky-policy model, the policy is attached to the resource, and both are claimed in one single request. Therefore, the query time t_p is derived as following:

$$\forall\, p = n \tag{18}$$

$$t_s = ln(n) \tag{19}$$

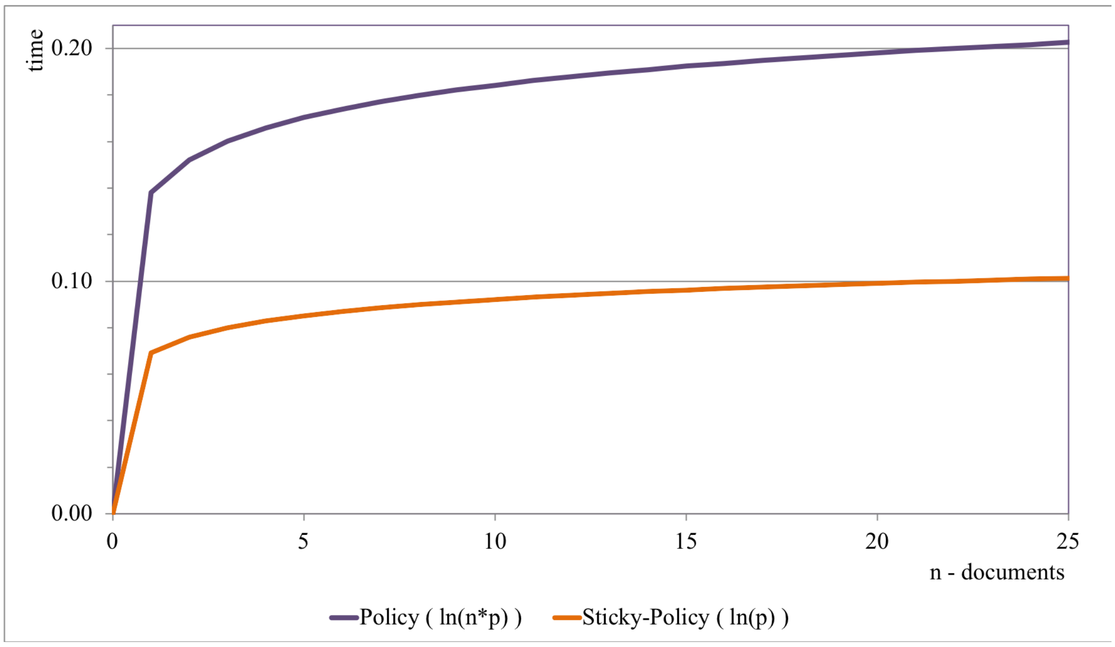


Figure 39 Access Policy (tp) and Sticky Policy (ts) DB query response time [150]

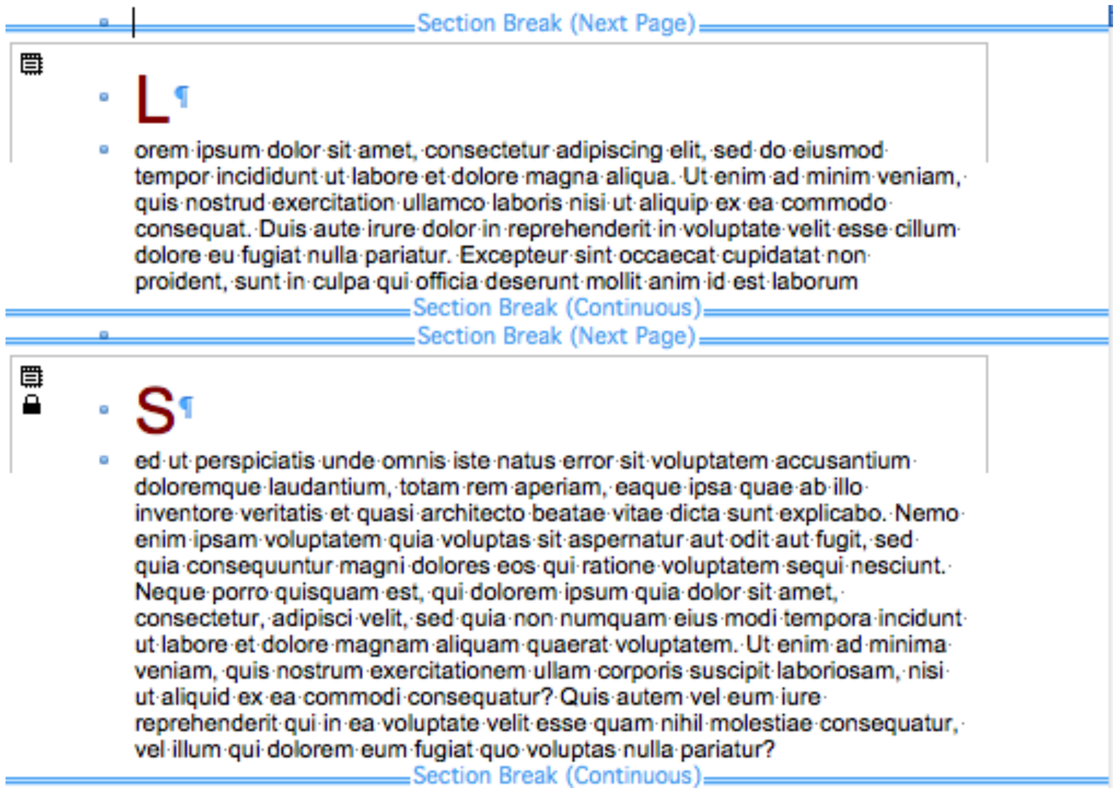
5.3 OOXML ZIP Wrapper

5.3.1 Policy Wrapper

Office Open XML (OOXML) standard is mostly built on top of XML files, which reference to each other forming a single document. XML files can be supplemented with other reach files to deliver graphic, multimedia and other elements [62]. OOXML data format can deliver data integrity using internal elements hashing, while confidentiality can be assured by ZIP wrapper password protection and content encryption. These techniques are sufficient to protect content that does not leave corporate network. However, when, leaked this built-in protection may not be sufficient for personal data. Cloud-based data sharing solution in order to utilize OOXML standard would require additional safeguards from service providers.

5.3.2 Master Document with a Single Sub-Document as a Security Boundary

As the additional OOXML with embedded XACML evaluation was performed using an explicit relationship and master document model [62] from WordprocessingML subclause. This part showed that described here granular access control method can be used to control access to sub-documents. Whereas policy access response denied resource Write access model added a read-only attribute to master document what was represented as a padlock on the document outline (see Figure 40). By representing OOXML sub-sections as individual documents, it is possible to control access to different document sections, i.e. sub-documents using one single access enforcement model. Other techniques would require a customised approach that leverages a single OOXML document internal XML schema.


Section Break (Next Page)

L¶

orem·ipsum·dolor·sit·amet,·consectetur·adipiscing·elit,·sed·do·eiusmod· tempor·incididunt·ut·labore·et·dolore·magna·aliqua.·Ut·enim·ad·minim·veniam,· quis·nostrud·exercitation·ullamco·laboris·nisi·ut·aliquip·ex·ea·commodo· consequat.·Duis·aute·irure·dolor·in·reprehenderit·in·voluptate·velit·esse·cillum· dolore·eu·fugiat·nulla·pariatur.·Excepteur·sint·occaecat·cupidatat·non· proident,·sunt·in·culpa·qui·officia·deserunt·mollit·anim·id·est·laborum

Section Break (Continuous)

Section Break (Next Page)

S¶

ed·ut·perspiciatis·unde·omnis·iste·natus·error·sit·voluptatem·accusantium· doloremque·laudantium,·totam·rem·aperiam,·eaque·ipsa·quae·ab·illo· inventore·veritatis·et·quasi·architecto·beatae·vitae·dicta·sunt·explicabo.·Nemo· enim·ipsam·voluptatem·quia·voluptas·sit·aspernatur·aut·odit·aut·fugit,·sed· quia·consequuntur·magni·dolores·eos·qui·ratione·voluptatem·sequi·nesciunt.· Neque·porro·quisquam·est,·qui·dolorem·ipsum·quia·dolor·sit·amet,· consectetur,·adipisci·velit,·sed·quia·non·numquam·eius·modi·tempora·incidunt· ut·labore·et·dolore·magnam·aliquam·quaerat·voluptatem.·Ut·enim·ad·minima· veniam,·quis·nostrum·exercitationem·ullam·corporis·suscipit·laboriosam,·nisi· ut·aliquid·ex·ea·commodi·consequatur?·Quis·autem·vel·eum·iure· reprehenderit·qui·in·ea·voluptate·velit·esse·quam·nihil·molestiae·consequatur,· vel·illum·qui·dolorem·eum·fugiat·quo·voluptas·nulla·pariatur?

Section Break (Continuous)

Figure 40.Master Document with two sub-documents

Granular access model built for this evaluation is just a proof of concept as various sources discourage using master document model due to several integrity problems with complex documentation.

5.4 XACML Evaluation

5.4.1 XML and JSON Formatted Policy

Following tendencies to simplify and reduce the size of a data, the JavaScript Object Notation (JSON) a skimmed alternative of the eXtensible Markup Language (XML) has been successfully used to format XACML policy [8]. This research leveraged XML based XACML policy engine libraries to evaluate the access control model. JSON formatted policies could reduce the size of the protected document, what considering protection attached to every piece of information could highly reduce the storage space required to host the data. Furthermore, the policy cannot be simply encrypted using basic key-encrypting RSA or ECC algorithms. Currently, the only model to encrypt the policy under public key algorithm that could be considered are either special public key algorithms allowing larger messages or partial policy encryption, or policy attributes anonymisation and obfuscation. With relatively large keys that might be required in the future considering increasing computing power, the JSON formatted policy could also be a subject for encryption [7]. To show size differences between JSON and XML formatted policy the evaluation took 400 XACML policies with internal complexity from low to complex. The policies fall within different size ranges (see Figure 41), where 3% of all evaluated XML formatted policies exceed the size of 16[KB] but most 70% of JSON formatted policies do not exceed the size of 2[KB].

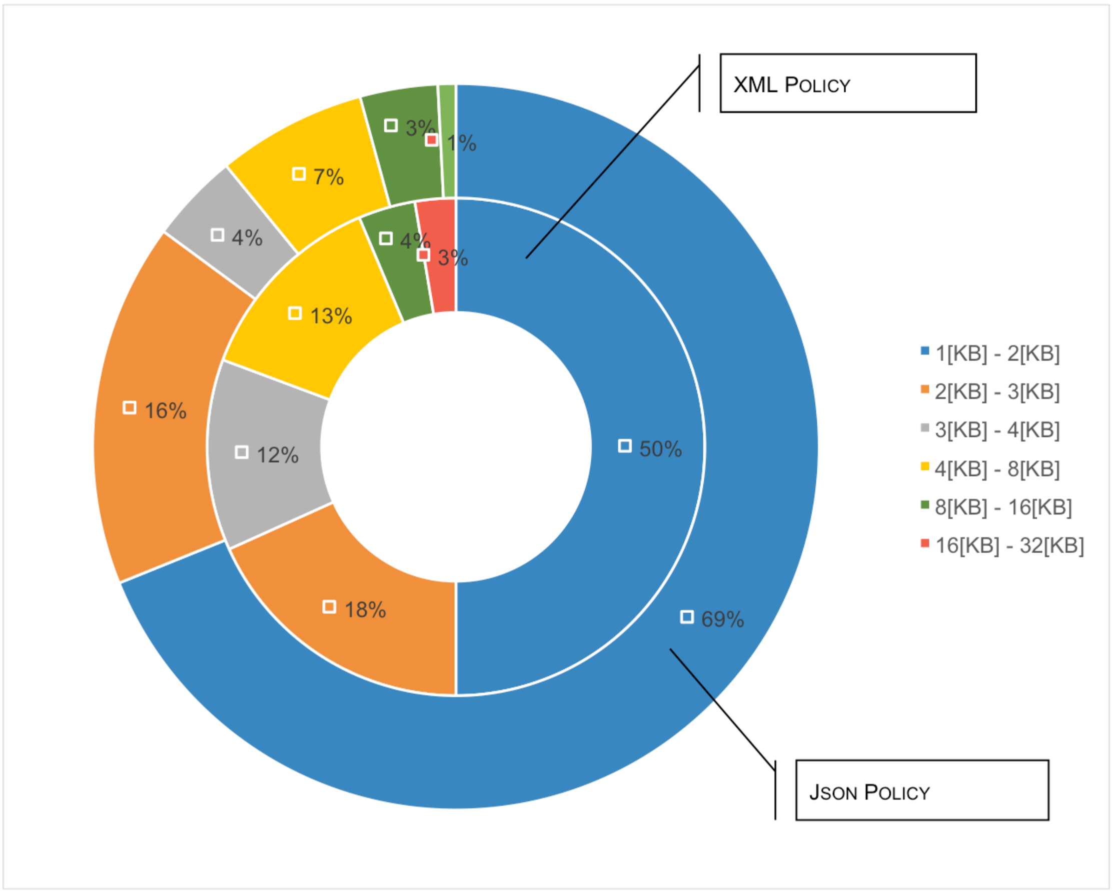


Figure 41 Policy sizes comparison formatted with XML and JSON. XACML policies [123].

5.4.2 Minifilter Driver – Security

The editing application will reject any OOXML modification requests if file metadata is set with a read-only permission. Access control implemented at mini driver level it is only a deterrent safeguard protecting a document from rights elevation. IRM systems fail to enforce strictly only the rights that are assigned to documents [148]. Minifilter driver will give sufficient protection at the client-side only if supported with additional safeguards like Blockchain ensuring data versioning. From an interoperability perspective overwriting file-level access control using a policy enforcement point (PEP) makes this part of SPIBE implementation transparent to the client editor application. Implementation at the driver level operates same as anti-virus software intercepting all open file events based on the specified filter at the file meta-data level. Driver development is not complicated, however, under Microsoft Windows systems, it requires exact Software Development Kit (SDK) version installed together with exact Windows Driver Kit

(WDK) and Microsoft Visual Studio version. Microsoft delivers official templates-like examples for WDK development where SwapBuffer File System Minifilter Driver covers exact scenario that is required for driver evaluation. Although driver development is relatively complex and requires rather a quality approach, the final solution reacted for OOXML document access requests despite editor application or editor application version.

The solution at this level could handle only limited interface implementation, although it exposed most of the operations related to policy enforcement point (PEP). Upon an open request (see Figure 42), it has to intercept a document access attempt, read the sticky policy, read the current identity information, send an access request to a trust authority (TA) for evaluation with the identity and enforce the response (see Figure 21) by either returning default empty document template upon deny response or replacing decrypted data buffer with file metadata flag set either to read-only or read-write. Upon an update request (see Figure 43), it contacts the TA for encryption parameters (see Figure 20) to encrypt the document and write it back to the buffer. Due to possible delays with productive implementations, minifilter should cache the TA parameters for later cryptographic operations. The part related to document policy management should be handled at the higher, application layer.

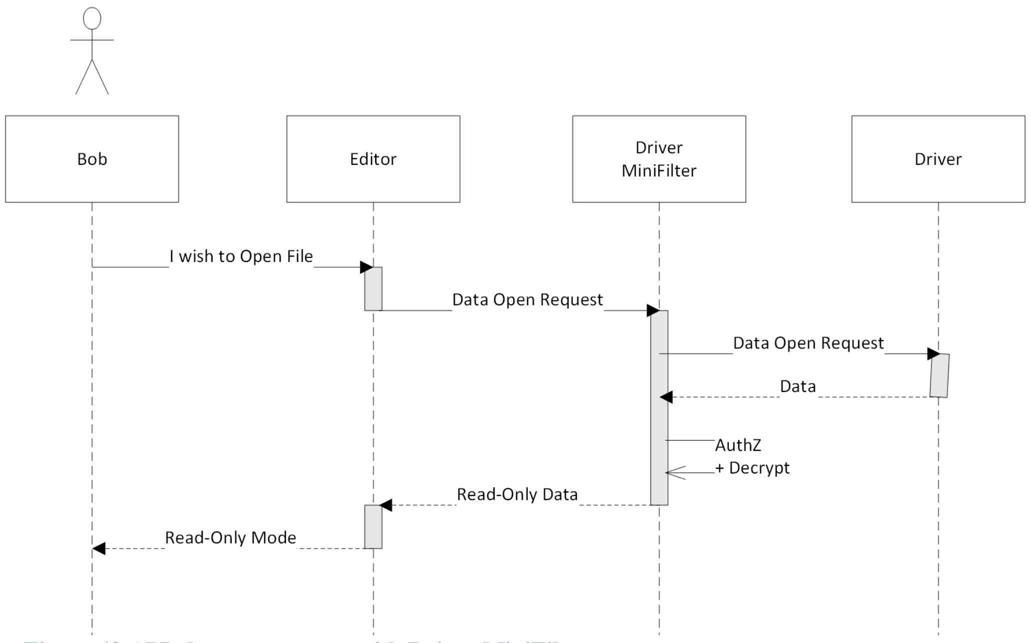


Figure 42 APP document access with Driver MiniFilter

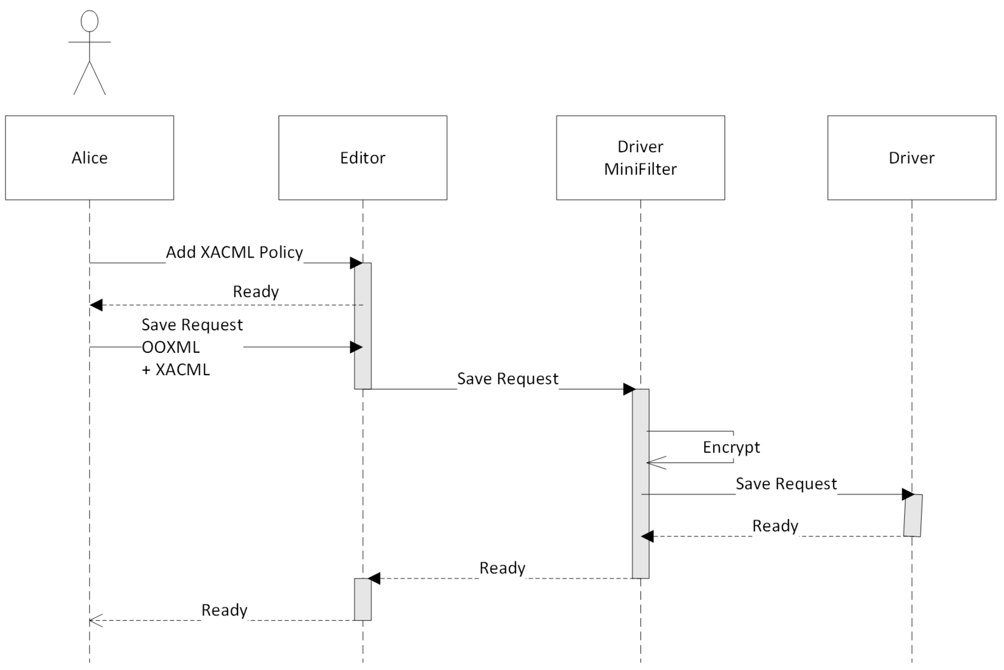


Figure 43 APP document protected with Driver MiniFilter

5.4.3 Web Application

To solve client systems interoperability problem, the OOXML document could be edited with a web-based application hosted online. Same as with the minifilter driver approach, any safeguards aiming to limit the possible data leakage after the document is decrypted have only a deterrent character. To limit illegitimate document amendments the online editor application, have to maintain document history metadata to ensure changes in authenticity and non-repudiation. This part could be achieved either via Git repositories or same as for minifilter driver; the data versioning could be hosted by blockchain (see 2.7.2).

5.4.4 Microsoft Office Add-in

Editor application plug-in is required to handle document policy edition based on TA templates. The only responsibility of the plug-in is to recognise the identity TA or allow addition TA registration to retrieve contextual policy templates. While some of the templates might be valid across different security boundaries and be accepted by different cloud services, some could be restricted to the specific trusted realm. Focusing on Microsoft Windows, a Word Add-in could be developed at the higher programming level, and, in comparison with previous MS Component Object Model (COM) or Visual Basic for Applications (VBA) implementations, the Add-in could be added to different MS Office distributions including distributions dedicated for different operating systems. Actual complete Add-in development is not part of the evaluation; however, the basic solution has been created to compare possible implementation scenarios.

5.5 RMS vs SPIBE

5.5.1 Key Management

The core Azure MS RMS feature is the support for bringing your key (BYOK) architecture. Customers could secure own root infrastructure keys leaving cloud service provider, i.e. Microsoft with almost no possibility to export keys outside of the strict boundaries. Using hardware security module (HSM) synchronisation functionality, the cloud-based HSM synchronizes keys only into specific regions. Technology is based on Thales Security World concept [151], although unlike with standard HSM Security World setup, Microsoft amended the master key sharing flow allowing non-physical master key transfer over the Internet. This model, if secure, would be already in place instead of physical safeguards. Hence authors assume this implementation weakens the model security as hardware HSM ensures security not only by protecting the keys in a secure appliance but also be enforcing strict procedures involving a physical, administrative key management.

Azure MS RMS by default uses RMS 2048 [bit] key sizes for applications. It seems currently secure, however, from the security perspective, the entire model has to be either completely reviewed or replaced with a construct that might be more quantum computing safe considering RMS algorithm security lifetime (ASL) as well as other vulnerabilities [148].

The SPIBE evaluation could only leverage soft key management system (KMS) vaulting via Microsoft Cryptography Next Generation (CNG) certificates with the private key-encrypting key. For actual implementation authors would leverage standard HSM implementation with no master key exchange over non-dedicated networks. The IBE scheme [10], however, is designed with the concept of using the distributed master key, which is never available in one piece in one single location. Using simple Shamir shared secret construct [126] master key could be distributed across different KMS locations and servers. Although authors have not yet evaluated this component on a large scale, there are several sources proposing this schema as a simple cryptographic primitive that solves complex security problems.

Another key management feature of SPIBE is that unlike other popular cryptographic constructs [7] for IRM the key s like shown in (8) does not have to be randomly generated prior IBE operations in order to encrypt the data. The XACML policy, a major secret key factor, follows the ciphertext. Upon an access request the XACML sticky policy authorizes the operation and authenticates the key generation.

5.5.2 Generate Keys Timing (Additional keys for RMS)

Prototyped sticky policies of size between 1[KB] and 32[KB] were used to protect the document, which was encrypted using IBE BF and AES256. Furthermore, IBE performance was compared to other more legacy RSA encryption - the same public key cryptographic model that Microsoft used for Azure RMS [7]. In the presented model sticky policy is used to generate a secret key under IBE for AES encryption of the data part. In MS RMS the AES secret key for data part encryption is generated separately and together with a policy to follow the data it is encrypted using RSA and then attached to the encrypted data. Therefore, here evaluation looks only into the initial process of policy setup including key-encrypting key operations without actual data-encrypting key operation (see Table 2).

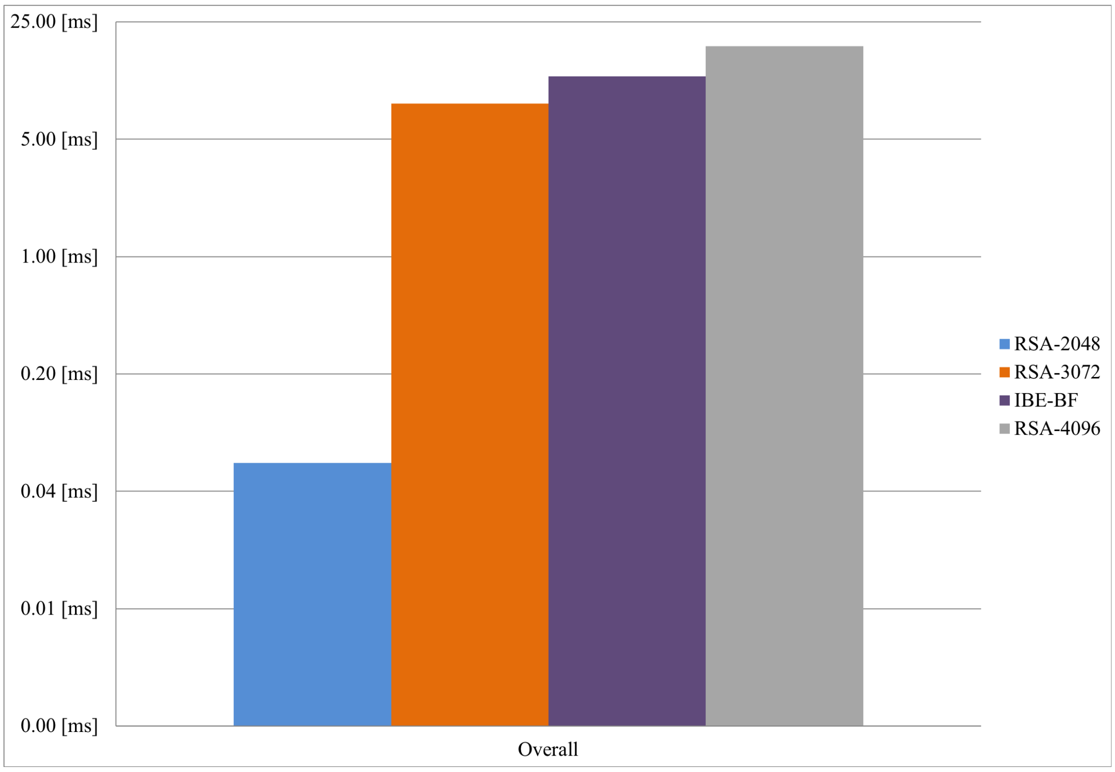


Figure 44. Times of Sticky policy mapping into 256 [bit] symmetric key space using IBE-BF compared to RSA3072 and RSA4096 operations applied to pseudo-random symmetric key

Results show (see Figure 44) that RSA with key size 4096[bit] requires more time than Pairing-based Cryptography, i.e. IBE to pair XACML policy of size between 1[KB] and 32[KB] into AES key space. RSA-3072 performs better and requires less time to complete cryptographic operations. However, soon it might need to be replaced with RSA of higher key size 4096 [bit] due to early quantum computing threats. Individual tests also show that RSA performed better during encryption compared to IBE pairing. RSA decryption, however, performed much slower, whereas IBE completes within similar time as in the previous pairing with the public test. The overall performance of RSA-4096 might be comparable to IBE. However, RSA-2048 performed much better overtaking all other evaluated cryptographic setups.

IBE WeilPairing Private 256 Key [ms]	IBE-WeilPairing Policy Public 256 Key[ms]	RSA 2048 Encrypt 256 Key [ms]	RSA 2048 Decrypt 256 Key [ms]	RSA 3072 Encrypt 256 Key [ms]	RSA 3072 Decrypt 256 Key [ms]	RSA 4096 Encrypt 256 Key [ms]	RSA 4096 Decrypt 256 Key [ms]	AES256GCM Encrypt 256 Key [ms]	AES256 GCM Decrypt 256 Key [ms]
12.745588	10.976774	0.055943	0.019828	0.274331	16.815242	0.402116	35.084968	0.011074	0.012746
12.330375	15.910127	0.018766	0.025493	0.135219	14.960057	0.20743	33.565706	0.012392	0.0131
11.795164	12.431612	0.019474	0.031158	0.135573	15.501286	0.193978	34.200738	0.015224	0.013808
10.782795	12.637626	0.026555	0.300961	0.134511	14.831918	0.225836	34.386575	0.009914	0.015932
13.138855	10.546694	0.018766	0.01133	0.133449	15.469782	0.192209	35.108331	0.013808	0.0131
11.489330	12.052504	0.010268	0.009914	0.134511	14.788379	0.194332	34.057732	0.012746	0.01487
13.028060	12.683996	0.013455	0.010622	0.133449	16.338083	0.190085	34.093837	0.0131	0.292448
11.161549	11.280838	0.022661	0.010622	0.133449	14.530685	0.211677	34.940546	0.013454	0.01487
10.695363	10.760141	0.022307	0.026201	0.133095	15.967471	0.20566	34.354363	0.033635	0.010976
13.482919	12.010735	0.288923	0.03045	0.133449	15.455269	0.202474	44.690088	0.012746	0.012746
13.012131	11.463136	0.023723	0.024077	0.133095	14.899174	0.191855	35.984419	0.011684	0.012038
12.004364	11.761891	0.020536	0.029034	0.133095	15.471552	0.191147	34.165694	0.011684	0.012038
10.922261	11.269865	0.025493	0.023723	0.133449	14.920412	0.191147	34.240029	0.016286	0.015578
34.011007	10.667399	0.075771	0.016641	0.133095	16.526752	0.190793	34.426574	0.013808	0.013808
11.915870	31.227346	0.023369	0.034699	0.133095	14.66909	0.189731	34.386929	0.016641	0.0131
11.063498	10.469881	0.021244	0.03045	0.150439	16.052779	0.193271	34.738072	0.015578	0.012746
12.421701	10.298203	0.021598	0.181993	0.135927	16.109061	0.192917	34.700905	0.013454	0.014162
11.721892	11.297829	0.322206	0.025493	0.134865	14.934217	0.192917	34.683914	0.01133	0.012392
11.273051	10.50988	0.282904	0.026201	0.134865	16.304102	0.193271	34.331355	0.017349	0.012746
11.783837	10.273779	0.033283	0.033991	0.135573	18.167781	0.205306	36.145832	0.015578	0.013454
11.873747	11.129691	0.256348	0.171017	0.134511	15.209964	0.191855	34.864796	0.012392	0.01487
10.767574	10.331477	0.026201	0.006727	0.136634	16.142689	0.22442	35.210983	0.012392	0.012746
12.568600	11.503135	0.027264	0.016995	0.134865	14.650683	0.189731	34.452414	0.0131	0.012392
10.912704	10.951641	0.036824	0.212089	0.134865	16.036496	0.190439	34.674003	0.012038	0.013808
13.487874	11.123673	0.01912	0.011684	0.145838	16.305871	0.190439	34.717896	0.014516	0.013454
12.207900	10.852174	0.022307	0.032575	0.134865	14.573162	0.190439	34.911874	0.014516	0.012392
12.035160	11.394464	0.03045	0.014163	0.135573	16.760376	0.191147	34.452414	0.012392	0.015578
11.021374	10.544924	0.01735	0.027618	0.135927	14.566437	0.190793	35.667966	0.009914	0.006019
11.361191	10.334662	0.016287	0.016287	0.145838	16.425869	0.189377	34.674357	0.016995	0.012038
10.630232	11.002614	0.026909	0.014517	0.134511	15.681459	0.191147	34.810991	0.020535	0.012392
12.091088	10.557667	0.024785	0.632019	0.13628	15.219521	0.191147	42.231729	0.008497	0.064792
11.284024	11.37181	0.014517	0.014871	0.151147	16.117557	0.191855	34.970988	0.005311	0.1048
10.638373	10.822086	0.00956	0.014517	0.135573	17.380894	0.192917	36.629008	0.006019	0.005311
13.198676	10.541738	0.008144	0.012747	1.448113	15.405358	0.189377	35.362485	0.008497	0.445045
10.828104	11.970028	0.00956	0.013101	0.204598	15.670132	0.189377	34.866919	0.013454	0.012392
12.010735	10.343866	0.00956	0.282904	0.183359	15.27368	0.191501	34.515422	0.006373	0.006019
11.191991	14.075473	0.212797	0.247497	0.156103	15.997559	0.242473	42.925521	0.011684	0.012746
10.653594	10.65855	0.021598	0.014871	0.139112	14.706611	0.271145	34.587279	0.013808	0.016286
12.367188	12.547008	0.019828	0.050278	0.150439	16.55153	0.217341	34.529581	0.01133	0.005665
12.232678	10.423156	0.026909	0.22519	0.141236	16.528522	0.191501	36.2053	0.007435	0.008851
12.237987	11.654636	0.270511	0.023015	0.139112	14.798291	0.227252	35.479651	0.017703	0.013808
10.887572	11.825606	0.029034	0.01133	0.14159	17.191872	0.194686	34.436132	0.012746	0.012392
12.459930	10.615719	0.0956	0.00779	0.136988	15.991541	0.196102	35.962473	0.01487	0.014162
11.140664	11.912684	0.008144	0.00779	0.153271	16.338437	0.199996	34.573828	0.016580	0.013454
12.124007	10.379971	0.009206	0.00779	0.139112	16.364631	0.192563	34.189411	0.014162	0.007435
12.172502	11.293227	0.008144	0.007436	0.13982	17.770267	0.208845	35.05488	0.006019	0.004957
11.929675	10.841909	0.033637	0.008144	0.263358	16.52746	0.192563	34.412415	0.008851	0.007789
12.645413	11.989143	0.008852	0.012393	0.134511	14.940943	0.335569	34.336664	0.004957	0.004249
27.699982	10.503155	0.568994	0.018412	1.179092	16.367463	0.190793	37.018735	0.005311	0.004603
12.848595	11.564019	0.035761	0.044613	0.177342	15.670486	0.204952	34.724267	0.004249	0.004249
10.639789	11.062436	0.035407	0.928378	0.17805	16.52392	0.189731	34.389761	0.004603	0.005665
11.859234	10.398732	0.020182	0.039656	0.145484	17.259127	0.191501	39.990712	0.004957	0.005311
12.300995	11.574638	0.259535	0.030096	0.143714	16.997185	0.189377	34.443919	0.004249	0.003895
10.982791	10.645453	0.026909	0.015225	0.134511	16.988336	0.22265	35.049925	0.030449	0.018057
12.480461	11.371102	0.145524	0.018766	0.134865	16.374897	0.192917	36.278573	0.007435	0.006727
10.798724	10.91235	0.007081	0.047092	0.134865	16.172423	0.213447	35.187267	0.006019	0.004603
12.028788	11.762245	0.008498	0.01735	0.134511	16.378436	0.190439	35.17948	0.007789	0.009914
10.882262	10.775008	0.00779	0.02089	0.14336	24.672785	0.189023	40.24522	0.007435	0.007081
12.563645	12.960805	0.00779	0.029034	0.150439	17.289569	0.189731	35.209214	0.004249	0.004249
10.633772	10.697487	0.026909	0.033637	0.153625	16.708341	0.235748	34.336664	0.004249	0.004249
11.941710	11.423136	0.320435	0.014871	0.16566	16.259147	0.344418	37.862612	0.003895	0.003895
10.994118	12.11268	0.020536	0.00779	0.135927	14.6344	0.221589	34.589049	0.004249	0.004249
10.817839	11.702069	0.239707	0.016641	0.157519	16.306933	0.191855	35.117888	0.021243	0.02443
12.432320	12.114804	0.033991	0.031867	0.134865	14.778468	0.202828	46.262446	0.005665	0.004603
10.645807	10.802264	0.01735	0.034699	0.134865	16.660909	0.204598	37.26581	0.004249	0.004249
11.909498	12.090734	0.02868	0.008852	0.135573	15.099878	0.207783	34.259144	0.005311	0.008143
11.567204	11.46172	0.021598	0.00956	0.164599	16.065522	0.223358	37.425099	0.004957	0.004957
10.697841	11.064913	0.015225	0.008852	0.14336	14.607498	0.20566	34.448875	0.005311	0.006019
13.119032	10.326875	0.018412	0.008144	0.142652	17.134882	0.259464	34.17596	0.004249	0.004603
13.227348	12.047903	0.021244	0.628832	0.135573	15.649247	0.207076	38.159244	0.004249	0.003895
12.138520	10.716602	0.33212	0.024431	0.135927	15.298812	0.202828	34.171712	0.004957	0.202873
11.855340	11.828792	0.007081	0.035407	0.143006	18.698036	0.201412	34.309762	0.004957	0.004249
12.523292	10.901023	0.236874	0.017704	0.157519	16.483567	0.191147	37.990751	0.004249	0.003895
10.979959	10.382449	0.019474	0.205362	0.163537	16.582326	0.191855	34.541616	0.004249	0.004957
12.844347	12.189847	0.195802	0.015933	0.156811	16.283925	0.191855	34.318966	0.012038	0.008497
10.691469	10.684744	0.011684	0.008498	0.159643	15.760042	0.189377	38.518528	0.004603	0.004249

11.763307	11.288272	0.020536	0.013101	0.140528	16.420559	0.191147	34.71117	0.004249	0.004249
11.070223	10.714832	0.020536	0.006727	0.20035	16.359676	0.20035	34.267639	0.004249	0.004249
12.171086	11.608619	0.044967	0.00956	0.138404	18.620162	0.189731	36.597859	0.004957	0.015224
10.669523	10.572534	0.02089	0.010976	0.136988	16.144105	0.189731	35.05488	0.014516	0.010268
11.975338	11.261016	0.029388	0.033283	0.136634	16.274014	0.190439	34.289232	0.004957	0.004603
11.133939	11.889322	0.00956	0.04001	0.13805	16.074726	0.191501	36.327775	0.004603	0.004249
10.842263	10.694655	0.009206	0.016287	0.137342	14.991561	0.193624	34.77701	0.004249	0.004249
12.416037	11.80543	0.009206	0.021598	0.136634	15.579868	0.228668	34.419849	0.004957	0.006727
10.770406	11.429154	0.01133	0.012038	0.135573	15.534913	0.3384	36.309369	0.009205	0.007435
12.058168	12.187723	0.21563	0.00779	0.134865	16.060567	0.258756	35.078951	0.005311	0.005311
11.369686	10.718372	0.035761	0.009914	0.134865	14.8036	0.212739	34.269763	0.004957	0.004603
10.638727	11.667025	0.044967	0.00779	0.217341	16.114371	0.190793	36.295918	0.014516	0.015224
12.532849	11.408269	0.01912	0.008144	0.171678	16.407816	0.223358	34.557899	0.007435	0.004249
10.934650	11.757643	0.018412	0.00779	0.1469	14.561835	0.194686	34.392239	0.003895	0.003895
11.948436	10.681558	0.009206	0.083561	0.144068	20.473222	0.19504	46.720844	0.003895	0.003541
12.651785	14.365379	0.009206	0.01133	0.144776	16.262332	0.198934	34.683914	0.003541	0.006727
12.060292	10.552357	0.00779	0.033637	0.13628	14.734221	0.192563	34.489228	0.007081	0.008143
12.846117	11.223494	0.009206	0.033991	0.157519	15.086073	0.192209	35.193639	0.008143	0.004957
11.300307	11.321545	0.009206	0.038948	0.134865	15.241114	0.193271	34.873645	0.004249	0.003895
14.911917	10.583153	0.020536	0.01735	0.134865	16.145167	0.194686	34.875061	0.014516	0.017703
10.665275	11.648265	0.037886	0.013809	0.134865	19.189708	0.192209	38.625075	0.008497	0.006373
11.471631	10.450058	0.044613	0.008144	0.134511	15.088904	0.190793	34.254188	0.004957	0.004249
11.974630	11.467737	0.022661	0.006727	0.135573	16.174901	0.194332	34.609579	0.004249	0.006019
11.151991	10.770052	0.013809	0.00779	0.135573	16.153662	0.189377	38.195703	0.006727	0.004249
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
≈12.151734	≈11.516674	≈0.060309	≈0.058011	≈0.169816	≈16.100445	≈0.206789	≈35.693356	≈0.009659	≈0.019693

Table 2 IBE and RSA performance comparison results (on CPU Intel Core i7 2.9[GHz])

5.5.3 Sticky Policy

RMS IRM construct uses eXtensible rights Markup Language (XrML) the rights expression language that was designed for closed environments [152] where could implementation was not considered. SPIBE is fully based on XACML policy format. XrML [153] and XACML [97] are very similar. However, the semantics used to express access rights are different. Both define tuples where the subject is permitted to perform specific activity defined by predicate against access object. In XrML a condition is a functional equivalent of XACML obligations, although what gives XACML advantages are complex expressions and predicates with negative and deny assertions. Note that XrML supports the only positive assertion.

XrML defines basic rights and digital licensing where the issuer of the license is effectively an owner of the digital asset. Same as XACML, XrML supports basic cryptographic functions. While XACML leverages eXtensible Stylesheet Language Transformation (XSLT) to constrain the policy, XrML uses templates within a license document. Both standards support external references to internal policy elements. The flexibility of both regarding elements extensions is similar; these are terms that could be redefined to extend the schema. Despite XrML strengths fundamentally it is not suited for complex access policies and rules while XACML is intended to be suitable for a variety of application environments [97] what perfectly fits various cloud configurations.

XACML itself formulates attribute-based access control (ABAC) phrases. Its profiles can support different access control models as per [127], [65]. This policy-based access control model delivers time-constrained access functionality as well as capabilities to handle emergency access requests like in Break-Glass scenario. Cross-Enterprise

Security and Privacy Authorisation (XSPA) [112] is a XACML profile dedicated for large enterprise use but mostly for healthcare institutions that exchange information across various security boundaries. Technically the XACML is not a part of any particular cryptographic primitive therefore actual implementation can quickly adapt any new functional requirements. XACML offers various features like JavaScript Object Notation (JSON) [154] profile to format the policy using light in size attributes representation in comparison to the heavier XML predecessor. Furthermore, it could represent legacy access control objects structure via efficient serialisation as described in [113].

5.5.4 Document Integration – Supported Formats

Officially Microsoft RMS natively supports OOXML, Portable Document Format (PDF), plain text, CSV and image file formats. It provides integration for mailing client applications also allowing email protection. Protected file is identified by a changed extension. Original document file extension is prefixed with a ‘p’ letter, e.g. PDF document instead of *.pdf* has a *.ppdf* extension.

SPIBE concept integrates security part into original OOXML document keeping its internal structure unchanged. Hence data XML section is replaced with an encrypted data file and the empty document template. SPIBE does not change the document extension.

5.5.5 Security

It has been proven that IRM system with higher interoperability considering currently commonly used operating systems, cannot deliver secure, granular access control. It is not only one IRM product related vulnerability but a logical inference considering facts, e.g. that read-only protected document could be read by an unauthorised individual simply using shoulder surfing social attack. Microsoft Azure RMS offers a handful of deterrent safeguards discouraging an individual from illegitimate document changes or by restricting message recipients. Despite the obvious IRM vulnerabilities MS RMS suffers from simple security flow giving adversary full rights over document if the adversary has even minimal rights to view the content [148]. There is another, a more serious security issue with the RMS model allowing an adversary to re-encrypt the amended document. Although Microsoft Office online editor applications do not seem to suffer from the described vulnerability.

SPIBE aims to leverage Blockchain with document signatures and document versioning meta-data. The signature could be an incorporated part of XACML policy under Attributes. Encrypted document is bind to the policy, and even if tampered the change cannot be submitted to the blockchain as the original updated document. Central versioning chain despite IRM model selected is the only way to ensure document authenticity, non-repudiation and integrity.

5.5.6 Quantum Computing

Azure RMS due to used cryptographic primitive has relatively low potential to become a quantum ready product [74]. The current construct is complex considering in-deep integration with different RSA key pair certificates. With large RSA keys, it might be possible to securely share the public key together with encrypted data. However, the key size might be larger than the policy and the protected file together.

The SPIBE despite a public key encryption has been proved quantum computing ready [14], [89]. Preferred for SPIBE, the IBE-IPG construct based on pairing-based cryptography isogeny is a quantum safe encryption, although offers the same complexity as the evaluated here IBE BF scheme. Existing IBE BE based on Diffie-Hellman cryptographic primitive could be replaced with its quantum resistant supersingular isogeny equivalent [14]. IBE over NTRU Lattices [74], [89] is another quantum ready construct that could be potentially used under IBC scheme. Finally, a proposal to leverage IBE construct with quadratic residue gives SPIBE high chances to successfully compete with RMS if it comes to quantum computing.

5.6 IONIC vs SPIBE

5.6.1 Architecture

Most modern solutions here also IONIC Secure Files are built as a framework with publicly available application programming interfaces (API) and software development kit (SDK) [8] for simplified cross-platform integration. The product is built under service-oriented architecture (SOA) with various globally accepted open standards. Shortly it consists of a policy engine, key services including personal and technical identity on-boarding (i.e. enrolment service), client SDK suite, management API and auditing/analytics APIs.

5.6.2 Key Management – Trusted Architecture

IONIC uses a master root key to encrypt a key ring. Key ring contains tens of thousands key-encrypting keys. These keys are then referenced considering key residency to relevant key stores. Keystore stores up to one trillion data-encrypting keys. Official documentation is missing information about key management system (KMS) implementation details, and from the discussion, with IONIC developer community manager, it was hard to obtain such information including crypto modules in place or random key generation (RNG) hardware and software libraries.

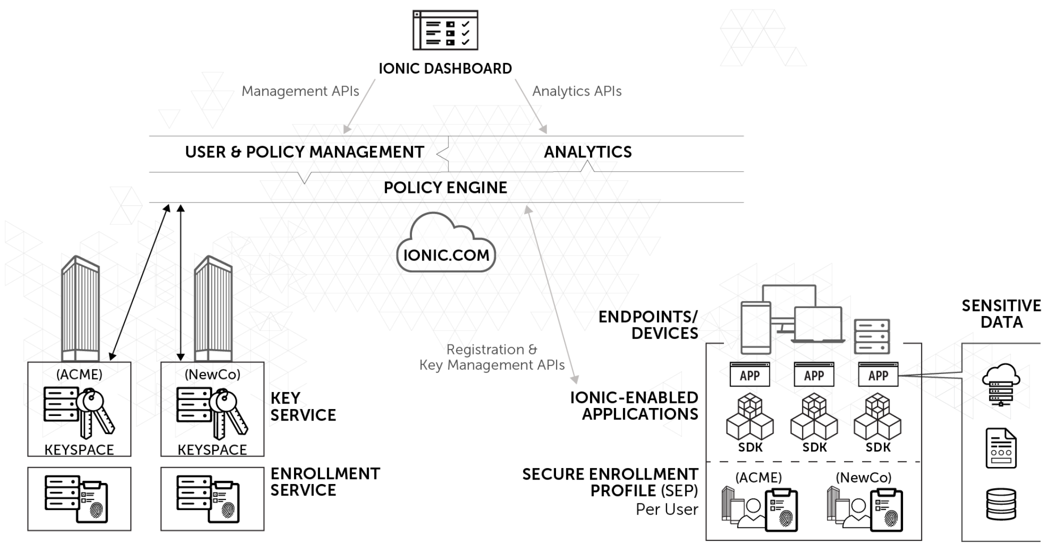


Figure 45 IONIC Solution architecture [8]

To manage data-encrypting symmetric keys IONIC Secure Files same as SPIBE uses Trusted Authority (TA). In Secure Files, symmetric data-encrypting keys are generated at the TA, i.e. Key Servers side [8] (see Figure 45). IONIC also encrypts immutable attributes using the same data-encrypting keys and authenticated AES256-GCM. The model enables a set of mutable attributes to be defined by the data owner, which could change during a document lifecycle without having an impact on the cryptographic verification. All IONIC Secure Files data-encrypting keys are stored in a dedicated key space located at a key server. Therefore the symmetric key escrow is possible at any time allowing immediate decryption of trillions of protected documents [8].

SPIBE does not support mutable and immutable attributes, but here all attributes are immutable. A key advantage of SPIBE is its ability to generate a symmetric data-encrypting key based on the policy itself. The initial protection of the document and the future re-encryption require the same operations including new key generation under IBE. The encryption key is generated every time the authorisation policy with its attributes changes. This is possible as keys could be generated dynamically under IBE algorithm without the need to upload and store them in the cloud. To decrypt the document upon a successful access request (see Figure 21) the decryption key is derived from the TA key and again from the access policy.

5.6.3 Sticky Policy

IONIC Secure Files adapted XACML version 3.0 [97] policy to represent complex access control statements. IONIC Policy Engine uses several JSON formatted XACML elements. It is strictly constrained with features that are applicable for files protection in shared environments (i.e. cloud computing). It supports limited Policy Sets with single *deny-overrides* condition across all enclosed policies. The IONIC implementation makes use of Obligation element. However, it excludes a little bit newer XACML elements called Advice. Next, the IONIC policy does not implement the full *Target* element format most probably due to a selected model where a single sticky policy is attached to a single document. Unlike IONIC Policy Engine the SPIBE could leverage *Target* [97] element to granularly control OOXML document access. Although the final SPIBE evaluation excludes granular access control functionality from the scope although with the *Target* element this part of evaluation would not be possible. Both IONIC Policy Engine and SPIBE do not support *AttributeSelector* and *Content* elements, and the XPath resolver. The main reason is the sticky policy model, where the policy protects a single document piece rather than multiple enclosed attributes or XML document sections. Despite these differences, IONIC Policy Engine and SPIBE use the same XACML elements, although the IONIC implementation is JSON formatted and SPIBE still evaluates only XML formatted policies. IONIC policies are already constrained with different templates. SPIBE requires each TA to deliver its own set of constrained policy templates. IONIC defined own set of attributes, it limited the data types and the functions. SPIBE, as well, aims to use the slightly limited XACML functionality to simplify the policy management from the end-user perspective.

5.6.4 Document Integration – Supported Formats

IONIC Secure Files natively supports OOXML, Portable Document Format (PDF), text, CSV and image files protection. Same as SPIBE, the IONIC embeds encrypted content using native OOXML functionality. IONIC adds unencrypted cover-page instructing the end-user about the encrypted content. SPIBE could potentially extend supportability for other file formats, however, for the evaluation purposes, OOXML was sufficient to show how potentially sticky policy could be integrated into existing file formats.

5.6.5 Quantum Computing

Both solutions, the IONIC Secure Files and SPIBE use AES symmetric encryption. IONIC used key servers to store all symmetric data-encrypting keys, while SPIBE derives ad hoc symmetric data-encrypting keys using identity-based encryption (IBE) primitive upon encryption or decryption request. Considering the fact that IBE construct could use different cryptographic primitives behind it has been proved that IBE as a construct could be quantum safe. With relatively large key space both constructs under AES encryption are post-quantum ready [74]. Since SPIBE could quickly respond to cryptographic primitive changes via policy obligations and advice the data is sufficiently protected upon re-encryption in case of new cryptographic vulnerabilities. Here security of the SPIBE model has to be considered as a comprehensive security solution rather than a simple cryptographic primitive. Only under this assumption, SPIBE security could be compared with IONIC Secure Files solution that uses only symmetric encryption.

5.7 Conclusions

SPIBE is an attractive construct for global cloud implementations. The IBE-IPG quantum ready schema in comparison to others used for IRM public key primitives offers a relatively long algorithm security lifecycle (ASL) providing reliable security for data at rest (see Table 2). The Microsoft RMS is more complicated than SPIBE. Therefore, it is hard to consider it as an adaptable framework. Overall key-encrypting-key operations performance evaluation (see Table 4) shows that IONIC Secure Files outmatch both RMS and SPIBE as it uses only AES symmetric encryption. RMS, despite of its double asynchronous encryption operation is faster than SPIBE, which performs only one actual bilinear mapping. The major operation cost will be the actual data encryption, which for all of the IRM solutions is nearly identical due to the same symmetric encryption algorithm. As a product the RMS has an obvious advantage, which is its market position. IONIC Secure Files, on the other hand, is a symmetric encryption-based product that despite all disadvantages of the symmetric key construct has a very high potential to become Azure RMS cloud-ready successor. SPIBE, unlike evaluated products, could become an open source IRM cloud-based ready framework. IBE does not constrain the solution to only one cryptographic primitive. Therefore, it could be following the technological changes. Well-defined XACML templates could make the solution very interoperable with different systems that could simply span across various cloud services. SPIBE, to become a fully valuable framework has to provide support not only for OOXML but also other file types and information formats, i.e. PDF, CSV, and emails.

	Characteristics	Azure RMS	IONIC	SPIBE
Specification	Policy Standard	XrML	XACML	XACML
	Information Type	PDF, OOXML, CSV, mail, image	PDF, OOXML, CSV, image	OOXML, *PDF, *CSV, *mail, *image
	Cryptographic construct	Asymmetric/Symmetric	Symmetric	Asymmetric/Symmetric
	Key-encrypting key	RSA 2048	AES256-CTR	IBE-IPG
	Data-encrypting key	AES128/AES256	AES256-GCM	AES256-CBC
	KMS	(custom) HSM Thales	(own Key Server) SSM	HSM
Property	ASL	Short	Long	Long
	Construct complexity	High	Mid	Mid
	Key management complexity	High	High	Low
	Quantum-ready	No	Yes	Yes
	Integrity assurance	N/A	Yes	Yes
	Authenticity assurance	No	No	Yes
	Key-encrypting key protection at rest	High	High	High
	Key-encrypting key protection in motion	Mid (custom HSM)	Low	Mid
	Data-encrypting key protection	Low	High	High

* - possible extension

Table 3. IRM solutions comparison

	Operation	Azure RMS	IONIC	SPIBE
Cryptographic Operations Cost Calculation Equation	Encryption $c_1 \leftarrow m_1$	$e_s + 2 \times e_p \leftarrow e_s(s_1, m_1) + e_p\left(p_{p1,}\left(e_p(p_{p1}, s_1), p_{p2}, POL\right)\right)$	$2 \times e_s \leftarrow e_s(s_1, s_2) + e_s(s_2, m)$	$e_s + \hat{e}_{p1} \leftarrow e_s(s_{\hat{e}}, m_1) + \left(\hat{e}\left(p_{p1}, POL\right)\right)$
	Decryption $m_1 \leftarrow c_1$	$d_s + d_p \leftarrow d_s\left(d_p(c_1)\right)$	$2 \times d_s \leftarrow d_s(s_1, s_2) + d_s(s_2, c_1)$	$d_s + \hat{e}_{s1} \leftarrow d_s(s_{\hat{e}}, c_1) + \left(\hat{e}(s_1, POL)\right)$
	Signing S_1	$S_p \leftarrow S_p(p_{s2}, c_1)$	N/A	$e_s + \hat{e}_{p1} \leftarrow e_s(s_{\hat{e}}, POL) + \left(\hat{e}\left(p_{p1}, m_1\right)\right)$
	Signing Verification S_1	N/A	N/A	$d_s + \hat{e}_{s1} \leftarrow d_s(s_{\hat{e}}, S_1) + \left(\hat{e}(s_1, m_1)\right)$
	Key Generation	$n \times s_1 + n \times \left(p_{p1}, p_{s1}\right) + \frac{1}{n} \times \left(p_{p2}, p_{s2}\right)$	$n \times s_2 + \frac{1}{n} \times s_1$	$n \times \left(p_{p1}, p_{s1}\right) + \frac{1}{n} \times s_1$
*Cryptographic Operations Cost [ms]	Encryption	$0.009659 + 2 \times 0.060309 = 0.130277$	$2 \times 0.009659 = 0.019318$	$0.009659 + 11.516674 = 11.526333$
	Decryption	$\underline{0.019693 + 0.058011}$ $\underline{= 0.077704}$	$2 \times 0.019693 = 0.039386$	$0.019693 + 12.151734 = 12.171427$

c_x – ciphertext
m_x – plaintext
S_x – signing product
n – number of client-side key pair generation requests
$\frac{1}{n}$ – denotes that the one-time key generation operation cost becomes negligible with higher n
e_s – symmetric encryption
e_p – asymmetric public key encryption
d_s – symmetric decryption
d_p – asymmetric public key decryption
S_p – asymmetric public key signing
s_x – symmetric key (x – key number)
p_{px} – asymmetric public key (x – key number)
p_{sx} – asymmetric private key (x – key number)
ê – bilinear mapping (s – on TA master key, p – on TA public key)
* – considering average operation cost from Table 2

Table 4. IRM operations comparison

6 Conclusions and Future Work

6.1 Achievement of Thesis Aim, Objectives and Research Questions

It is possible to deliver a secure cloud-based information sharing framework. However, the development should incorporate only standardised solutions apply only the latest security techniques. Since legacy IRM products entered into a single global data sharing security boundary, they struggle with delivering both security [148] and interoperability. Proposed SPIBE framework, however, could quickly adapt to technological changes and new security threats.

6.2 Recap of Contribution and Novelty

SPIBE offers a highly flexible model for secure information sharing in the cloud. The model, unlike other IRM solutions [7], [8], offers relatively long ASL [73] due to IBE scheme that could integrate various cryptographic primitives consequently addressing future threats starting from emerging quantum computing. The model successfully leveraged XACML formatted [97] sticky policy acting as an identity for the information to compute a key for symmetric data encryption. XACML as a part of a standard integrates with authorization systems as well as authentication systems like SAML [119], [125] to deliver one single compatible framework. While other models require either symmetric or both symmetric and asymmetric keys to be factorized prior data protection, the SPIBE under IBE paradigm [10] derives keys from the constructed sticky policy. The possible collisions caused by deriving two identical keys in case same server parameters are distributed to different end-users in compare to CP-ABE are mitigated in SPIBE by adding actual document global unique identifier and the document version. Blockchain solves the most common problem with IRM [148], the data authenticity and non-repudiation. Every information change made under SPIBE in order to pass into protected information lifecycle has to be versioned, authenticated and finally committed to a single global trusted blockchain of all legitimate changes.

6.3 Main Findings

Sticky policies as an access control technique satisfy systems where confidentiality and integrity of personal information are protected [49] based on policies set by the data owner. These secure policies follow the data and technically define possible scenarios in which data can be processed. Such a model is suitable for governments, financial institutions systems, electronic healthcare systems where patients' privacy policies would stick to a medical record [5]. Access policies can be expressed on top of different data structures. Modern computing powers made this relatively high level expressing language sufficient to compute complex rules. Cross-domain configurations are one of the XACML advantages over other policy languages [4]. The future work, especially in a cloud context, will aim to show the potential how XACML formatted sticky policy could securely span across various security domains and boundaries [155].

Like modern authentication methods, including OAuth, SAML, OpenID and other the XACML compliments suite with ready to use authorisation standard for large companies [97]. The future work needs to focus on showing that XACML could authorise subjects in global Business to Business (B2B) and Business to Consumer (B2C) cloud-based configurations. Furthermore, the proposed model consists of trust authority (TA) components, where authorisation and cryptographic modules are deployed as separate (sub) entities. For evaluation, both were implemented as two separate libraries called via a single application. However, the final solution would consist of two or more web services bound together using authentication where each service acts as an authenticated technical identity. Evaluation of the driver level architecture shown that this is a quite powerful approach with loads of potential for further development. Legacy file access control constructs [113] empowered with additional modern cryptographic and authorisation techniques give additional centralized, i.e. cloud-enabled control over the access management. For the evaluation purposes, Windows based driver has been developed and installed. The IBE encryption operations have been replaced with a symmetric XOR encoding against 0xFF value. The integration with XACML libraries was too complex for the scope of this evaluation. However, as a part of the research minifilter driver development has a high potential for a standard solution compatible with various editor applications. Due to difficulties to bound Windows Driver Kit (WDK) together with GMP crypto libraries, there was only a limited implementation completed towards the entire model evaluation.

In compare with other IRM solutions and actual market ready products (see Table 3) the SPIBE have couple major advantages. It leveraged quantum ready public key cryptographic protocols delivering both the secure public key exchange functionality together with quantum resistant cryptographic algorithms. By empowering IRM with blockchain technology the major problem of authenticating data changes is solved.

Simply by maintaining common chain of changes for every document the author and other contributors are sure they work on the main branch of legitimate changes.

6.4 Future Work

All developed and evaluated components in order to become a ready framework would require major architectural alignment, considering policy enforcement point as a starting point where all SPIBE elements such as access control, encryption and document management interact together. It was challenging to implement consistent components integrated as per Figure 25 due to different solution architectures. Prototype is sufficient for Proof of Concept and actual evaluation, but there is a different approach required to deliver a customer ready framework.

Online editor application could be the first prototyped artefact giving space for further platform/framework improvements. The final framework, however, has to consider different operating systems and protected file types. Therefore, architecture requires a modular approach where different core components could be referenced despite the underlying system architecture. Approach where only online editor application is considered simplifies interoperability challenges with PEP supporting different platforms.

It has been proven that currently accepted information rights management (IRM) solutions not excluding Azure MS RMS cannot efficiently authenticate changes as well as differentiate legitimate from illegitimate data amendments [148]. Considering IRM systems that are designed for interoperability with different operating systems and across different platforms, the secure solution based on legacy assumptions have to compromise with very limited data access control. From the moment a policy enforcement point releases a positive access decision, the editor application that hosts decrypted information is the only security boundary for the document. In other words, despite the rights assigned, the data processor has either full access rights or none. The secure solution would have to maintain a legitimate chain of all the document versions. This would guarantee that only authenticated and properly signed documents version could be superseded with its new version.

Considering available technologies for integrity and non-repudiation, a blockchain is a preferred option as unlike signature it verifies data in a historical context [156]. Furthermore, blockchain service together with Trust Authority (TA) may govern any illegitimate re-encryption attempt of the amended data. Changed document despite initially defined sticky policy rights giving only read rights, can be rejected by the TA therefore and will not be added to the blockchain.

Finally, a well-defined policy template could highly constrain a sticky policy in a given context as well as enforce the requirement to fill an authenticated originator attribute. Depends on the implementation each legitimate amendment made would require re-encryption with a different, new document version and it would require unique identifier of an authenticated change originator. XACML policy defines two safeguards, an obligation and advice (see Figure 10). Both could carry further instructions for policy enforcement point behind editor application defining how to handle the initial authorisation including basic requirements for data re-encryption under updated policy.

Regarding quantum tampering for SPIBE, the author would need another ten months to change used C libraries for IBE in the way it respects the other isogenous morphism as in IBE-IPG. Finally, there are newly developed cryptographic primitives [157] that could suit more platforms in terms of symmetric encryption efficiency especially considering low powered devices that do not provide native AES support. Chacha20 stream cypher seems to overrun AES when it comes to security. Cloud-based secure data sharing framework should be built on top of such cryptographic protocols.

Bibliography

[1] M. Baumgärtner *et al.*, “Cyber-Espionage Hits Berlin, The Breach from the East,” *Spiegel Online*, Berlin, 2018.

[2] T. S. Bernard, T. Hsu, N. Perlroth, and Ron Lieber, “Equifax Says Cyberattack May Have Affected 143 Million in the U.S.,” *The New York Times, Business Day*, New York, 07-Sep-2017.

[3] Swisscom, “Swisscom tightens security for customer information,” Bern, 2018.

[4] M. Mowbray, S. Pearson, and Y. Shen, “Enhancing privacy in cloud computing via policy-based obfuscation,” *J. Supercomput.*, vol. 61, no. 2, pp. 267–291, Mar. 2010.

[5] A. Abbas and S. Khan, “A Review on the State-of-the-Art Privacy Preserving Approaches in E-Health Clouds,” *IEEE J. Biomed. Heal. Informatics*, vol. 2194, no. c, pp. 1–1, 2014.

[6] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, “Scalable and Secure Sharing of Personal Health Records in Cloud Computing using Attribute-based Encryption,” *IEEE Trans. PARALLEL Distrib. Syst.*, vol. XX, no. Xx, pp. 1–14, 2012.

[7] Sergey Simakov, M. Sieber, and M. Norden, *Azure RMS Security Evaluation Guide*. Microsoft, 2015.

[8] Ionic Security Inc., “Documentation for Ionic .NET SDK.” Ionic Security Inc., 2017.

[9] Martin Lambert and A. Peet, “Oracle Information Rights Management 11g – Managing information everywhere it is stored and used,” *Management*, no. March. Oracle Corporation, Redwood Shores, CA 94065, USA, p. 23, 2010.

[10] D. Boneh and M. Franklin, “Identity-Based Encryption from the Weil Pairing,” *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[11] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” *Annu. Int. Conf. Theory …*, pp. 457–473, 2005.

[12] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-Policy Attribute-Based Encryption,” in *2007 IEEE Symposium on Security and Privacy (SP ’07)*, 2007.

[13] V. Goyal, A. Sahai, O. Pandey, and B. Waters, “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data,” pp. 1–16, 2006.

[14] T. Koshiba and K. Takashima, “Pairing Cryptography Meets Isogeny: A New Framework of Isogenous Pairing Groups.,” *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 1138, 2016.

[15] L. Hardesty, “Raising cryptography ’ s standards,” *MIT News Office*, p. 3, 2014.

[16] F. Pagano and D. Pagano, “Using in-memory encrypted databases on the cloud,” *2011 1st Int. Work. Secur. Serv. Cloud*, pp. 30–37, Sep. 2011.

[17] OECD, *Annex to the recommendation of the Council of 23 September 1980: Guidelines governing the protection of privacy and transborder flows of personal data*, no. September. European Union, 1980.

[18] European Parliament and T. Council, *Regulation 2016/679 of the European parliament and the Council of the European Union*, vol. 2014, no. March 2014. European Union: European Commission (EC), 2016, pp. 1–88.

[19] E. Chau and R. Hertzberg, *Assembly Bill No. 375*, no. 375. United States, California, 2018.

[20] D. Chappell, “Claims-based Identity for Windows; Technologies and Scenarios,” no. February. DavidChappell & Associates, 2011.

[21] P. A. Grassi, M. E. Garcia, and J. L. Fenton, “Digital Identity Guidelines. NIST Special Publication 800-63-3,” 2017.

[22] P. Resnick, “RFC5322 Internet Message Format.” IETF, pp. 1–57, 2008.
[23] International Telecommunication Union, *E.164 The International Public Telecommunication Numbering Plan*. Switzerland, 2011, pp. 1–32.
[24] E. Kursun, G. Fernandez, A. Berson, and B. Goodman, “Biometrics identification module and personal wearable electronics network based authentication and transaction processing,” US9892576B2, 2018.
[25] IEEE Standards Association, “Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID),” no. Cid. pp. 1–19, 2017.
[26] IEEE, “Guidelines for 64-bit global identifier (EUI-64),” *EUI-64 Guidelines*. IEEE Standards Association, 2007.
[27] G. E. Suh and S. Devadas, “Physical Unclonable Functions for Device Authentication and Secret Key Generation,” *2007 44th ACM/IEEE Des. Autom. Conf.*, pp. 9–14, 2007.
[28] X. Chen, “Identity Federation in Federated Trust Healthcare Network.” University of Virginia, Virginia, 2004.
[29] E. V. Wilson and D. M. Strong, “Editors ’ Introduction to the Special Section on Patient-centered e-Health : Research Opportunities and Challenges,” vol. 34, 2014.
[30] W. J. Buchanan *et al.*, “Who Would You Trust To Identify You In Accessing Your Health Record ? So who do we trust ?,” 2013.
[31] R. Zhang and L. Liu, “Security Models and Requirements for Healthcare Application Clouds,” *2010 IEEE 3rd Int. Conf. Cloud Comput.*, pp. 268–275, Jul. 2010.
[32] G. Zhao, Z. Li, W. Li, H. Zhang, and Y. Tang, “Privacy Enhancing Framework on PaaS,” *2012 Int. Conf. Cloud Serv. Comput.*, pp. 131–137, Nov. 2012.
[33] W. J. Buchanan, R. Lewis, D. L. Fan, and O. Uthmani, “Information Sharing Around Child Protection,” in *Information Sharing in the Public Sector*, 2012.
[34] A. Jain and C. Farkas, “Ontology-Based Authorization Model for XML Data in Distributed Systems,” in *Digital Rights Management*, IGI Global, 2013, pp. 210–236.
[35] X. H. Le, T. Doll, M. Barbosu, A. Luque, and D. Wang, “An enhancement of the role-based access control model to facilitate information access management in context of team collaboration and workflow.,” *J. Biomed. Inform.*, vol. 45, no. 6, pp. 1084–1107, Dec. 2012.
[36] R. W. P. Luk, H. V. Leong, T. S. Dillon, A. T. S. Chan, W. B. Croft, and J. Allan, “A survey in indexing and searching XML documents,” *J. Am. Soc. Inf. Sci. Technol.*, vol. 53(6), no. 6, pp. 415–437, 2002.
[37] R. Wyden, “Letter from Ron Wyden Ranking Member of Committee on Finance to Mark Zuckerberg CEO Facebook.” Ron Wyden United States Senator for Oregon, Washington DC, 2018.
[38] M. Zuckerberg, “Hearing Before the United States Senate Committee on the Judiciary and the United States Senate Committee on Commerce ,” 2018, pp. 1–7.
[39] W. Zeng, C.-Y. Lin, and H. Yu, Eds., *Multimedia Security Technologies for Digital Rights Management*, 1st Editio. Academic Press, 2006.
[40] K. LI, P. Hunt, B. Khasnabish, A. Nadalin, and Z. Zeltsan, “RFC7642 System for Cross-domain Identity Management: Definitions, Overview, Concepts, and Requirements.” IETF, pp. 1–19, 2015.
[41] W. Yeong, T. Howes, and S. Kille, “RFC 1487 X.500 Lightweight Directory Access Protocol Status.” IETF, pp. 1–21, 1993.
[42] G. Spyra, “Next Generation Authentication Infrastructures With Role Based

Security For Cloud Computing," Edinburgh Napier University, 2012.
[43] C. Collingham *et al.*, "OASIS Service Provisioning Markup Language (SPML) Version 2," 2006.
[44] P. Hunt, K. Grizzle, E. Wahlström, and C. Mortimore, "RFC7643 System for Cross-domain Identity Management: Core Schema." IETF, pp. 1–104, 2015.
[45] T. Lodderstedt, M. McGloin, and P. Hunt, "RFC6819 OAuth 2.0 Threat Model and Security Considerations." IETF, pp. 1–71, 2013.
[46] P. Hunt, K. Grizzle, M. Ansari, E. Wahlström, and C. Mortimore, "RFC7644 System for Cross-domain Identity Management: Protocol." IETF, pp. 1–89, 2015.
[47] P. Siriwardena, "Brief history of Identity Provisioning," Auckland, pp. 3–5, 2018.
[48] E. Mccallister and K. Scarfone, "Guide to Protecting the Confidentiality of Personally Identifiable Information (PII) Recommendations of the National Institute of Standards and Technology." U.S. Department of Commerce, 2010.
[49] S. Pearson and N. Wainwright, "An interdisciplinary approach to accountability for future internet service provision," *Int. J. Trust Manag. Comput. Commun.*, vol. 1, no. 1, p. 52, 2013.
[50] S. Salamatian, A. Zhang, P. Calmon, and S. Bhamidipati, "How to Hide the Elephant – or the Donkey – in the Room : Practical Privacy Against Statistical Inference for Large Data," in *1st IEEE Global Conference on Signal and Information Processing*, 2013.
[51] M. Chen, C. Yang, and M. Hwang, "Privacy Protection Data Access Control," *Int. J. Netw. Secur.*, vol. 15, no. 6, pp. 391–399, 2013.
[52] L. Sun and H. Wang, "A purpose-based access control in native XML databases," *Concurr. Comput. Pract. Exp.*, vol. 24, no. 10, pp. 1154–1166, 2012.
[53] European Parliament and The Council, *Directive 2016/680 of the European Parliament and the Council on the protection of natural persons with regard to the processing of personal data by competent authorities for the purposes of the prevention, investigation, detection or prosecution of crimi*, vol. 2014, no. April. European Union: European Commission (EC), 2016, p. L 119/89-L 119/131.
[54] G. Spyra, W. J. Prof Buchanan, P. Cruickshank, and D. E. Ekonomou, "Cloud-Based Identity and Identity Meta-Data: Secure and Control of Data in Globalization Era," *Int. J. Reliab. Qual. E-Healthcare*, vol. 3, no. 1, pp. 49–66, 2014.
[55] D. Ferraiolo and W. Jansen, "NISTIR 7987 Policy Machine : Features, Architecture, and Specification," Gaithersburg, 2014.
[56] D. Ferraiolo and S. Gavrila, "Policy Machine," *Computer Security Resource Center*, 2014. [Online]. Available: http://csrc.nist.gov/pm/index.html.
[57] M. A. Rahaman, Y. Roudier, P. Miseldine, and A. Schaad, "Ontology-Based Secure XML Content Distribution," *IFIP Adv. Inf. Commun. Technol.*, vol. 297, pp. 294–306, 2009.
[58] L. Zhou, V. Varadharajan, and M. Hitchens, "Cryptographic Role-Based Access Control for Secure Cloud Data Storage Systems," in *Security, Privacy and Trust in Cloud Systems*, S. Nepal and M. Pathan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 313–344.
[59] T. Finin, A. Joshi, J. Niu, R. Sandhu, and W. Winsborough, "ROWLBAC - Representing Role Based Access Control in OWL," in *ACM Symposium on Access Control Models and Technologies (SACMAT'08)*, 2008.
[60] J. F. Sequeda, Marcelo Arenas, and D. P. Miranker, "On Directly Mapping

Relational Databases to RDF and OWL (Extended Version),” in *Proceedings of the 21st international conference on World Wide Web*, 2012.
[61] G. Costa and R. Ortale, “On Effective XML Clustering by Path Commonality: An Efficient and Scalable Algorithm,” in *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, 2012, pp. 389–396.
[62] Apple *et al.*, “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.
[63] 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.
[64] A. 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.
[65] L. Gasparini, “XACML and Risk-Aware Access Control,” Aberdeen, 2013.
[66] E. Barker, “NIST 800-175B: Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms.” NIST, 2016.
[67] E. Barker and J. Kelsey, “NIST 800-90A: Recommendation for Random Number Generation Using Deterministic Random Bit Generators,” *NIST Special publication*, no. March. NIST, 2015.
[68] R. L. Rivest, A. Shamir, and L. M. Adleman, “Cryptographic communications system and method,” US05860586, 1977.
[69] D. Gabbasov, “Breaking the Enigma,” Tartu, Estonia, 2015.
[70] J. Daemen, V. Rijmen, and K. U. Leuven, “AES Proposal : Rijndael,” *Complexity*, pp. 1–45, 1999.
[71] V. S. Miller, “Uses of elliptic curves in cryptography,” *Adv. Cryptol. — CRYPTO ’85*, no. January, pp. 417–426, 1986.
[72] N. Koblitz, “Elliptic curve cryptosystems,” *Math. Comput.*, vol. 48, no. 177, pp. 203–203, 1987.
[73] E. Barker, “NIST 800-57: Recommendation for Key Management – Part 1: General,” *NIST Special Publication 800-57*. NIST, pp. 1–142, 2016.
[74] L. Chen *et al.*, “NISTIR 8105 Draft - Report on Post-Quantum Cryptography,” 2016.
[75] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. Cambridge University Press, 2011.
[76] P. W. Shor, “Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.
[77] J. Proos and C. Zalka, “Shor’s discrete logarithm quantum algorithm for elliptic curves.” Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Waterloo, 2008.
[78] Nigel P. Smart, V. Rijmen, B. Warinschi, and G. Watson, “Algorithms, Key Sizes and Parameters Report,” Heraklion, 2014.
[79] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC ’96*, 1996, pp. 212–219.
[80] E. B. Barker, M. Smid, D. Branstad, and S. Chokhani, “NIST 800-130: A Framework for Designing Cryptographic Key Management Systems,” *NIST Special Publication 800-130*. NIST, pp. 1–120, 2013.
[81] ID Quantique, “Random Number Generation using Quantum Physics.” ID Quantique, Geneva, 2010.

[82] G. Spyra, “MS AD DS Password Policies Issue Password Hash analysis,” *All Identities*, 2015. [Online]. Available: https://allidentities.wordpress.com/2015/02/02/privileged-account-password-policy-audit/. [Accessed: 21-May-2018].
[83] The Federal Assembly of the Swiss Confederation, *Federal Act on Data Protection*, no. 1 January 2014. Switzerland: Articles 95, 122 and 173 paragraph 2 of the Federal Constitution, 1992, pp. 1–24.
[84] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya, “ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption,” in *Proceedings of the 11th ACM conference on Computer and communications security - CCS ’04*, 2004, p. 354.
[85] S. Pearson, P. Bramhall, and HP Laboratories, “Towards Accountable Management of Identity and Privacy : Sticky Policies and Enforceable Tracing Services Marco Casassa Mont,” in *14th International Workshop on Database and Expert Systems Applications (DEXA’03)*, 2003.
[86] A. Shamir, “Identity-Based Cryptosystems and Signature Schemes,” in *Advances in Cryptology*, vol. 196, G. R. Blakley and D. Chaum, Eds. Springer Berlin Heidelberg, 1985, pp. 47–53.
[87] C. Cocks, “An identity based encryption scheme based on quadratic residues,” in *8th IMA International Conference on Cryptography and Coding*, 2001, pp. 360–364.
[88] M. Ajtai, “Generating Hard Instances of Lattice Problems,” in *STOC ’96 Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 99–108.
[89] L. Ducas, V. Lyubashevsky, and T. Prest, “Efficient Identity-Based Encryption over NTRU Lattices,” *ASIACRYPT 2014*, vol. 8874, pp. 22–41, 2014.
[90] B. Lynn, “Authenticated Identity-Based Encryption,” 2002.
[91] J. Sliwa and E. Benoist, “A Web Architecture Based on Physical Data Separation Supporting Privacy Protection in Medical Research,” *Int. J. Reliab. Qual. E-Healthcare*, vol. 1, no. 4, pp. 68–79, Jan. 2012.
[92] S. Kisilevich, L. Rokach, Y. Elovici, and B. Shapira, “Efficient Multidimensional Suppression for K-Anonymity,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 3, pp. 334–347, 2010.
[93] M. Ye, X. Wu, X. Hu, and D. Hu, “Anonymizing classification data using rough set theory,” *Knowledge-Based Syst.*, vol. 43, pp. 82–94, May 2013.
[94] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, “XML Signature Syntax and Processing (Second Edition),” 2008. [Online]. Available: http://www.w3.org/TR/xmldsig-core/.
[95] E. Kiltz, G. Neven, T. N. CWI Amsterdam, S. IBM Zürich Research laboratory, and B. Katholieke Universitet Leuven, “Identity-Based Signatures,” in *Cryptology and Information Security Series*, vol. 2, IOS Press, 2009, pp. 31–44.
[96] C. Gentry and A. Silverberg, “Hierarchical ID-Based Cryptography,” in *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2002)*, 2002, pp. 548–566.
[97] A. Saldhana *et al.*, “eXtensible Access Control Markup Language (XACML) Version 3.0,” 2013.
[98] N. Parab and A. Brown, “Cloud Storage Using Merkle Trees,” 20160110261, 2016.
[99] J. M. Stewart, E. Tittel, and M. Chapple, “Accountability and Access Control,” in *CISSP®: Certified Information Systems Security Professional Study Guide, Fifth Edition*, Fourth., 2011, pp. 1–45.
[100] V. Bertocci, *Programming Windows Identity Foundation*. Redmond,

Washington: Microsoft Press, 2011.
[101] B. Thigpen, "An Introduction to XACML," no. Security 401. 2003.
[102] D. F. Ferraiolo and D. R. Kuhn, "Role-Based Access Controls," *15th Natl. Comput. Secur. Conf. (1992), Balt.*, pp. 554–563, 1992.
[103] R. S. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST Model for Role-Based Access Control: Towards A Unified Standard," in *5th ACM Workshop on Role Based Access Control*, 2012, pp. 47–63.
[104] J. López, A. Maña, and M. I. Yagüe, "XML-based Distributed Access Control System," *E-Commerce Web Technol. Lect. Notes Comput. Sci.*, vol. 2455, no. i, pp. 203–213, 2002.
[105] K. Yang and X. Jia, "ABAC: Attribute-Based Access Control," in *Security for Cloud Storage Systems*, New York, NY: Springer New York, 2014, pp. 39–58.
[106] J. Hur and D. K. Noh, "Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.
[107] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," *Proc. 15th ACM Conf. Comput. Commun. Secur. - CCS '08*, p. 417, 2008.
[108] C. I. Fan, V. S. M. Huang, and H. M. Ruan, "Arbitrary-state attribute-based encryption with dynamic membership," *IEEE Trans. Comput.*, vol. 63, no. 8, pp. 1951–1961, 2014.
[109] Y. Demchenko, "Security Languages for Access Control and Authorisation: SAML and XACML Languages Overview," 2010.
[110] F. Li *et al.*, "Cyberspace-Oriented Access Control: A Cyberspace Characteristics based Model and its Policies," *IEEE Internet Things J.*, vol. X, no. X, p. 13, 2018.
[111] Y. Demchenko, O. Koeroo, C. de Laat, and H. Sagehaug, "Extending XACML Authorisation Model to Support Policy Obligations Handling in Distributed Application," in *Proceedings of the 6th international workshop on Middleware for grid computing - MGC '08*, 2008, pp. 1–6.
[112] 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.
[113] G. Karjoth and A. Schade, "Serialization of XACML policies," US8458764 B2, 2013.
[114] ContentGuard, "eXtensible rights Markup Language (XrML) 2.0 Specification Part I: Primer 20," no. November. Eduworks, pp. 1–46, 2001.
[115] OECD, *Pursuant to Directive 95/46/EC of the European Parliament and of the Council on the adequacy of the protection provided by the EU-U.S. Privacy Shield*. Brussels: United States Federal Trades Commission, 2016.
[116] OECD, "OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data." Organisation for Economic Co-operation and Development, 2013.
[117] Y. Yang, X. Liu, and R. H. Deng, "Lightweight Break-glass Access Control System for Healthcare Internet-of-Things," *IEEE Trans. Ind. Informatics*, vol. 3203, no. c, pp. 1–8, 2017.
[118] P. E. Sevinc, "Securing Information by Controlling Access to Data in Documents," Eidgenössische Technische Hochschule Zürich, 2007.
[119] S. Cantor *et al.*, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0," 2005.
[120] S. Petride, A. Tarachandani, N. Agarwal, and S. Idicula, "Managing and Processing Office Documents in Oracle XML Database," in *DBKDA 2011, The*

Third International Conference on Advances in Databases, Knowledge, and Data Applications., 2011, no. c, pp. 89–95.

[121] S. Pearson, M. C. Mont, and G. Kounga, "Enhancing Accountability in the Cloud via Sticky Policies," *Secur. Trust Comput. Data Manag. Appl. Commun. Comput. Inf. Sci.*, vol. 187, pp. 146–155, 2011.

[122] J. Lai, R. H. Deng, Y. Yang, and J. Weng, "Adaptable ciphertext-policy attribute-based encryption," *2007 IEEE Symp. Secur. Privacy(SP'07)*, vol. 8365 LNCS, pp. 199–214, 2007.

[123] G. Spyra, W. J. Buchanan, and E. Ekonomou, "Sticky policies approach within cloud computing," *Comput. Secur.*, pp. 1–9, 2017.

[124] J. C. Cha and J. H. Cheon, "An Identity-Based Signature from Gap Diffie-Hellman Groups," *Int. Assoc. Cryptologic Res.*, pp. 18–30, 2002.

[125] S. Track and W. Product, "XACML v3.0 XML Digital Signature Profile Version 1.0 Specification URIs," no. May. The Organization for the Advancement of Structured Information Standards (OASIS), pp. 1–11, 2014.

[126] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[127] A. Anderson, "XACML Profile for Role Based Access Control (RBAC), Version 2.0," 2004.

[128] V. Bertocci, *Modern authentication with Azure Active Directory for web applications*. Redmond, Washington: Microsoft Press, 2016.

[129] L. Chen and C. Kudla, "Identity Based Authenticated Key Agreement Protocols from Pairings," *16th IEEE Computer Security Foundations Workshop, 2003. Proceedings.* IEEE, pp. 219–233, 2003.

[130] G. Hsieh and R.-J. Chen, "Design for a secure interoperable cloud-based Personal Health Record service," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings (2012)*, 2012, pp. 472–479.

[131] F. Rezaeibagha and Y. Mu, "Distributed clinical data sharing via dynamic access-control policy transformation," *Int. J. Med. Inform.*, vol. 89, pp. 25–31, 2016.

[132] S. Alshehri, S. P. Radziszowski, and R. K. Raj, "Secure access for healthcare data in the cloud using Ciphertext-Policy Attribute-Based Encryption," *Proc. - 2012 IEEE 28th Int. Conf. Data Eng. Work. ICDEW 2012*, pp. 143–146, 2012.

[133] A. Mohandas and S. S, "Privacy Preserving Content Disclosure for Enabling Sharing of Electronic Health Records in Cloud Computing," *Proc. 7th ACM India Comput. Conf.*, p. 7:1--7:7, 2014.

[134] K. Seol, Y. G. Kim, E. Lee, Y. D. Seo, and D. K. Baik, "Privacy-preserving attribute-based access control model for XML-based electronic health record system," *IEEE Access*, vol. 6, pp. 9114–9128, 2018.

[135] A. Bahga and V. K. Madisetti, "A cloud-based approach for interoperable electronic health records (EHRs)," *IEEE J. Biomed. Heal. Informatics*, vol. 17, no. 5, pp. 894–906, 2013.

[136] E. P. M. O. Veterans Health Administration Office of Information & Technology and Office of Information & Analytics, *VistA (VA) Monograph*, January 13. Department of Veterans Affairs, United States of America, 2017.

[137] U. Premarathne *et al.*, "Hybrid Cryptographic Access Control for Cloud-Based EHR Systems," *IEEE Cloud Comput.*, vol. 3, no. 4, pp. 58–64, 2016.

[138] M. Peleg, D. Beimel, D. Dori, and Y. Denekamp, "Situation-Based Access Control: Privacy management via modeling of patient data access scenarios," *J. Biomed. Inform.*, vol. 41, no. 6, pp. 1028–1040, 2008.

[139] R. Gajanayake, R. Iannella, and T. Sahama, "Privacy oriented access control

for electronic health records," *Electron. J. Heal. Informatics*, vol. 8, no. 2, 2014.
[140] A. Lunardelli, I. Matteucci, P. Mori, and M. Petrocchi, "A prototype for solving conflicts in XACML-based e-Health policies," *Proc. CBMS 2013 - 26th IEEE Int. Symp. Comput. Med. Syst.*, pp. 449–452, 2013.
[141] J. Calvillo-Arbizu, I. Roman-Martinez, and L. M. Roa-Romero, "Standardized access control mechanisms for protecting ISO 13606-based electronic health record systems," *2014 IEEE-EMBS Int. Conf. Biomed. Heal. Informatics, BHI 2014*, pp. 539–542, 2014.
[142] P. Gope and R. Amin, "A Novel Reference Security Model with the Situation Based Access Policy for Accessing EPHR Data," *J. Med. Syst.*, vol. 40, no. 11, 2016.
[143] K. Yang, Z. Liu, X. Jia, and X. S. Shen, "Time-Domain Attribute-Based Access Control for Cloud-Based Video Content Sharing: A Cryptographic Approach," *IEEE Trans. Multimed.*, vol. 18, no. 5, pp. 940–950, 2016.
[144] T. Neubauer and J. Heurix, "A methodology for the pseudonymization of medical data.," *Int. J. Med. Inform.*, vol. 80, no. 3, pp. 190–204, Mar. 2011.
[145] M. T. SandIkkaya, B. De Decker, and V. Naessens, "Privacy in commercial medical storage systems," *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng.*, vol. 69 LNICST, pp. 247–258, 2011.
[146] S. Sharma and V. Balasubramanian, "A biometric based authentication and encryption Framework for Sensor Health Data in Cloud," *Conf. Proc. - 6th Int. Conf. Inf. Technol. Multimed. UNITEN Cultiv. Creat. Enabling Technol. Through Internet Things, ICIMU 2014*, pp. 49–54, 2015.
[147] R. Au and P. Croll, "Consumer-Centric and Privacy-Preserving Identity Management for Distributed E-Health Systems," in *Proceedings of the 41st Hawaii International Conference on System Sciences - 2008*, 2008.
[148] M. Grothe, C. Mainka, P. Rösler, and J. Schwenk, "How to Break Microsoft Rights Management Services," *10th USENIX Work. Offensive Technol. (WOOT 16)*, pp. 1–14, 2016.
[149] G. Gulati, "Announcing new Microsoft Azure Information Protection policy decision point capabilities with Ionic Security," 2018. [Online]. Available: https://cloudblogs.microsoft.com/enterprisemobility/2018/04/17/announcing-new-microsoft-azure-information-protection-policy-decision-point-capabilities-with-ionic-security/.
[150] G. Spyra, W. J. Buchanan, and E. Ekonomou, "Sticky policy enabled authenticated OOXML," in *SAI Computing Conference 2016*, 2016.
[151] P. DiToro, "Hardware Key Management in the Azure Cloud." Thales e-security, 2016.
[152] Microsoft Corporation, "XrML," vol. 747717, pp. 2–3, 2008.
[153] ContentGuard, "eXtensible rights Markup Language (XrML) 2 . 0 Specification Part II : Core Schema," no. November. Eduworks, pp. 1–46, 2001.
[154] H. Lockhart and B. Parducci, "JSON Profile of XACML 3.0 Version 1.0," no. December 2014. OASICS, pp. 1–34, 2014.
[155] S. Pearson and M. C. Mont, "Sticky Policies : An Approach for Managing Privacy across Multiple Parties," *CS Digital Library*, IEEE Computer Society, pp. 60–68, Sep-2011.
[156] K. Okupski, "Bitcoin Developer Reference," Eindhoven, The Netherlands, 2015.
[157] Y. Nir and A. Langley, "RFC7539: ChaCha20 and Poly1305 for IETF Protocols." Internet Research Task Force (IRTF), pp. 1–45, 2015.

APPENDIXES

APPENDIX A XACML VERSION 3.0 POLICIES AND RULE-COMBINING ALGORITHMS

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I{D}**	**I{P}**	**I{DP}**
Permit	Permit	Deny	Permit	I{D}	Permit	I{DP}
Deny	Deny	Deny	Deny	Deny	Deny	Deny
N/A	Permit	Deny	N/A	I{D}	I{P}	I{DP}
I{D}	I{D}	Deny	I{D}	I{D}	I{P}	I{DP}
I{P}	Permit	Deny	I{P}	I{DP}	I{P}	I{DP}
I{DP}	I{DP}	Deny	I{DP}	I{DP}	I{DP}	I{DP}

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 5 Deny-overrides

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I{D}**	**I{P}**	**I{DP}**
Permit	Permit	Deny	Permit	I{D}	Permit	
Deny	Deny	Deny	Deny	Deny	Deny	Deny
N/A	Permit	Deny	N/A	I{D}	I{P}	I{DP}
I{D}	I{D}	Deny	I{D}	I{D}	I{D}	I{DP}
I{P}	Permit	Deny	I{P}	I{D}	I{P}	I{DP}
I{DP}	I{DP}	Deny	I{DP}	I{D}	I{D}	I{DP}

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 6. Ordered-deny-overrides

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I{D}**	**I{P}**	**I{DP}**
Permit	Permit	Permit	Permit	Permit	Permit	Permit
Deny	Permit	Deny	Deny	Deny	I{P}	I{DP}
N/A	Permit	Deny	N/A	I{D}	I{P}	I{DP}
I(D}	Permit	Deny	I{D}	I{D}	I{DP}	I{DP}
I{P}	Permit	I{P}	I{P}	I{DP}	I{P}	I{DP}
I{DP}	Permit	I{DP}	I{DP}	I{DP}	I{DP}	I{DP}

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 7. Permit-overrides

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I{D}**	**I{P}**	**I{DP}**
Permit	Permit	Permit	Permit	Permit	Permit	Permit
Deny	Permit	Deny	Deny	Deny	I{P}	I{DP}
N/A	Permit	Deny	N/A	I{D}	I{P}	I{DP}
I(D}	Permit	Deny	I{D}	I{D}	I{DP}	I{DP}
I{P}	Permit	I{P}	I{P}	I{DP}	I{P}	I{DP}
I{DP}	Permit	I{DP}	I{DP}	I{DP}	I{DP}	I{DP}

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 8. Ordered-permit-overrides

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I{D}**	**I{P}**	**I{DP}**
Permit	Permit	Permit	Permit	Permit	Permit	Permit
Deny	Permit	Deny	Deny	Deny	Deny	Deny
N/A	Permit	Deny	Deny	Deny	Deny	Deny
I{D}	Permit	Deny	Deny	Deny	Deny	Deny
I{P}	Permit	Deny	Deny	Deny	Deny	Deny
I{DP}	Permit	Deny	Deny	Deny	Deny	Deny

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 9. Deny-unless-permit

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I{D}**	**I{P}**	**I{DP}**
Permit	Permit	Deny	Permit	Permit	Permit	Permit
Deny	Deny	Deny	Deny	Deny	Deny	Deny
N/A	Permit	Deny	Permit	Permit	Permit	Permit
I{D}	Permit	Deny	Permit	Permit	Permit	Permit
I{P}	Permit	Deny	Permit	Permit	Permit	Permit
I{DP}	Permit	Deny	Permit	Permit	Permit	Permit

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 10. Permit-unless-deny

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I{D}**	**I{P}**	**I{DP}**
Permit	Permit	Deny	Permit	I{D}	I{P}	I{DP}
Deny	Permit	Deny	Deny	I{D}	I{P}	I{DP}
N/A	Permit	Deny	N/A	I{D}	I{P}	I{DP}
I{D}	Permit	Deny	I{D}	I{D}	I{P}	I{DP}
I{P}	Permit	Deny	I{P}	I{D}	I{P}	I{DP}
I{DP}	Permit	Deny	I{DP}	I{D}	I{P}	I{DP}

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 11. First-applicable

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I{D}**	**I{P}**	**I{DP}**
Permit	I	I	Permit	I{D}	I{P}	I{DP}
Deny	I	I	Deny	I{D}	I{P}	I{DP}
N/A	Permit	Deny	N/A	I{D}	I{P}	I{DP}
I{D}	I{D}	I{D}	I{D}	I{D}	I{DP}	I{DP}
I{P}	I{P}	I{P}	I{P}	I{DP}	I{P}	I{DP}
I{DP}	I{DP}	I{DP}	I{DP}	I{DP}	I{DP}	I{DP}

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 12. Only-one-applicable – only for Policy Set;

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I**	**I{D}**	**I{P}**
Permit	Permit	Deny	Permit	Deny	I	I
Deny	Deny	Deny	Deny	Deny	Deny	Deny
N/A	Permit	Deny	N/A	Deny	I	I
I	Deny	Deny	Deny	Deny	I	I
I{D}	I	Deny	I	I	I	I
I{P}	I	Deny	I	I	I	I

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 13. Legacy Deny-overrides

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I**	**I{D}**	**I{P}**
Permit	Permit	Deny	Permit	Deny	I	I
Deny	Deny	Deny	Deny	Deny	Deny	Deny
N/A	Permit	Deny	N/A	Deny	I	I
I	Deny	Deny	Deny	Deny	I	I
I{D}	I	Deny	I	I	I	I
I{P}	I	Deny	I	I	I	I

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 14. Legacy Ordered-deny-overrides

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I**	**I{D}**	**I{P}**
Permit	Permit	Permit	Permit	Permit	Permit	Permit
Deny	Permit	Deny	Deny	I	I	I
N/A	Permit	Deny	N/A	I	I	I
I	Permit	Deny	I	I	I	I
I{D}	Permit	I	I	I	I	I
I{P}	Permit	I	I	I	I	I

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

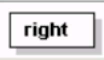
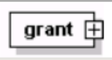
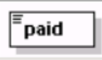
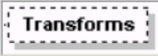
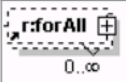
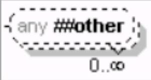
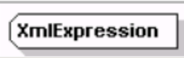
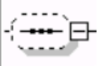
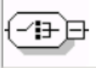
Table 15. Legacy Permit-overrides

2nd ↓ 1st →	**Permit**	**Deny**	**N/A**	**I**	**I{D}**	**I{P}**
Permit	Permit	Permit	Permit	Permit	Permit	Permit
Deny	Permit	Deny	Deny	I	I	I
N/A	Permit	Deny	N/A	I	I	I
I	Permit	Deny	I	I	I	I
I{D}	Permit	I	I	I	I	I
I{P}	Permit	I	I	I	I	I

I	Indeterminate
{D}	Deny
{P}	Permit
N/A	NotApplicable

Table 16. Legacy Ordered-permit-overrides

APPENDIX B XRML DIAGRAM CONVENTIONS [114]

	A single mandatory element.
	A single mandatory element with child elements.
	A single mandatory element containing Parsed Character Data (#PC-Data). The content may be simple content or mixed complex content.
	A single optional element.
	A multiple optional element (in this case, zero to infinity) with child elements.
	A placeholder for any element from any namespace.
	A complex type.
	A sequence. Solid lines connect this symbol to required elements within the sequence. Dotted lines connect this symbol to optional elements in the sequence.
	A choice.

APPENDIX C EXTERNAL LIBRARIES / PACKAGES

Library Name	Rel.	Comments	Licensing	Url
The Pairing-Based Cryptography Library	*0.5.14*		*GNU Lesser General Public License*	*https://crypto.stanford.edu/pbc/files/pbc-0.5.14.tar.gz*
OpenSSL	*1.0.2*		*OpenSSL SSLeay*	*https://github.com/openssl/openssl*
GMP	*6.0.0*		*GNU LGPL v3 GNU GPL v2*	*https://gmplib.org/download/gmp/gmp-6.1.0.tar.bz2*
XACML.Core	*0.0.0.0*	*Unknown Source; Found .Net executable before research started; Extracted libraries and modified;*	*N/A*	https://github.com/GregSpyra/xacml-core
Microsoft Windows Driver Kit	*10.0.15063.0*	*Required for spibedrv MiniFilter*	*Microsoft*	https://go.microsoft.com/fwlink/p/?LinkID=845980
Microsoft Software Development Kit	*10.0.15063.468*	*Required for spibedrv MiniFilter*	*Microsoft*	*https://go.microsoft.com/fwlink/p/?LinkID=845298*

APPENDIX D IBE WITH ECC AND RSA EVALUATION – ENVIRONMENT SETUP

```
@SET PATH=%PATH%;C:\Projects\IBE\gmp-6.0.0;
@SET PATH=%PATH%;C:\Projects\IBE\pbc-0.5.14;
@SET GMP_Dir=C:\Projects\IBE\gmp-6.0.0
@SET GMP_InstallDir=/c/Projects/IBE/gmp
@SET PBC_Dir=C:\Projects\IBE\pbc-0.5.14
@SET PBC_InstallDir=/c/Projects/IBE/pbc_install
@SET OSL_Install=/c/Projects/IBE/openssl
@SET OSL_Dir=C:\Projects\IBE\openssl-1.0.2d
```

APPENDIX E IBE WITH ECC AND RSA EVALUATION – COMPILE CRYPTOGRAPHIC LIBRARIES

```
C:\MinGW\msys\1.0\msys.bat
cd $GMP_DIR

export CXXFLAGS="$CXXFLAGS --output-def"

#configure --prefix=$GMP_INSTALLDIR --host=i686-pc-mingw32
configure --prefix=$GMP_INSTALL --host=coreisbr-pc-mingw32 --disable-static --
enable-shared
make & make check & make install

cd $PBC_DIR
#configure --prefix=$PBC_INSTALLDIR --host=coreisbr-pc-mingw32
configure --prefix=$PBC_INSTALL --host=i686-pc-mingw32 --disable-static --
enable-shared ABI=64
make & make check & make install

#openssl-1.0.1q
#Under Visual Studio CMD
cd $OSL_DIR
perl Configure VC-WIN32 --prefix=C:\Projects\IBE\openssl
ms\do_ms
nmake -f ms\nt.mak
nmake -f ms\nt.mak install


#NTSHELL - don’t generate libs under Visual Studio folder!
SET PATH=%PATH%;C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin
lib /machine:x86 /def:libgmp.def

lib /machine:x86 /def:libpbc.def
```

APPENDIX F IBE WITH ECC AND RSA EVALUATION – SBE\VC\SBE.SLN (VISUAL STUDIO SOLUTION)

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 2013
VisualStudioVersion = 12.0.30501.0
MinimumVisualStudioVersion = 10.0.40219.1
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "ConsoleApplication",
"ConsoleApplication\ConsoleApplication.vcxproj", "{02419819-AF2A-4E28-A2D9-
086A7AC4A955}"
EndProject
Global
     GlobalSection(SubversionScc) = preSolution
       Svn-Managed = True
       Manager = AnkhSVN - Subversion Support for Visual Studio
     EndGlobalSection
     GlobalSection(SolutionConfigurationPlatforms) = preSolution
       Debug|Win32 = Debug|Win32
       Debug|x64 = Debug|x64
       Release|Win32 = Release|Win32
       Release|x64 = Release|x64
     EndGlobalSection
     GlobalSection(ProjectConfigurationPlatforms) = postSolution
       {02419819-AF2A-4E28-A2D9-086A7AC4A955}.Debug|Win32.ActiveCfg =
Debug|Win32
       {02419819-AF2A-4E28-A2D9-086A7AC4A955}.Debug|Win32.Build.0 =
Debug|Win32
       {02419819-AF2A-4E28-A2D9-086A7AC4A955}.Debug|x64.ActiveCfg = Debug|x64
       {02419819-AF2A-4E28-A2D9-086A7AC4A955}.Debug|x64.Build.0 = Debug|x64
       {02419819-AF2A-4E28-A2D9-086A7AC4A955}.Release|Win32.ActiveCfg =
Release|Win32
       {02419819-AF2A-4E28-A2D9-086A7AC4A955}.Release|Win32.Build.0 =
Release|Win32
       {02419819-AF2A-4E28-A2D9-086A7AC4A955}.Release|x64.ActiveCfg =
Release|x64
       {02419819-AF2A-4E28-A2D9-086A7AC4A955}.Release|x64.Build.0 =
Release|x64
     EndGlobalSection
     GlobalSection(SolutionProperties) = preSolution
       HideSolutionNode = FALSE
     EndGlobalSection
EndGlobal
```

APPENDIX G IBE WITH ECC AND RSA EVALUATION – SBE\VC\CONSOLEAPPLICATION\CONSOLEAPPLICATION.VCXPROJ (VISUAL STUDIO PROJECT)

```xml
<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="14.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{02419819-AF2A-4E28-A2D9-086A7AC4A955}</ProjectGuid>
    <Keyword>Win32Proj</Keyword>
    <RootNamespace>ConsoleApplication</RootNamespace>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
Label="Configuration">
    <ConfigurationType>Application</ConfigurationType>
    <UseDebugLibraries>true</UseDebugLibraries>
    <PlatformToolset>v140</PlatformToolset>
    <CharacterSet>Unicode</CharacterSet>
    <UseOfMfc>Dynamic</UseOfMfc>
    <CLRSupport>true</CLRSupport>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
Label="Configuration">
    <ConfigurationType>Application</ConfigurationType>
    <UseDebugLibraries>true</UseDebugLibraries>
    <PlatformToolset>v140</PlatformToolset>
    <CharacterSet>Unicode</CharacterSet>
    <UseOfMfc>Dynamic</UseOfMfc>
    <CLRSupport>true</CLRSupport>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
Label="Configuration">
    <ConfigurationType>Application</ConfigurationType>
    <UseDebugLibraries>false</UseDebugLibraries>
    <PlatformToolset>v140</PlatformToolset>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <CharacterSet>Unicode</CharacterSet>
    <CLRSupport>false</CLRSupport>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
Label="Configuration">
    <ConfigurationType>Application</ConfigurationType>
    <UseDebugLibraries>false</UseDebugLibraries>
    <PlatformToolset>v140</PlatformToolset>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <CharacterSet>Unicode</CharacterSet>
    <CLRSupport>false</CLRSupport>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
Label="PropertySheets">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
  </ImportGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
Label="PropertySheets">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <LinkIncremental>false</LinkIncremental>
    <IgnoreImportLibrary>false</IgnoreImportLibrary>

<ExtensionsToDeleteOnClean>*.a;*.dll;$(ExtensionsToDeleteOnClean)</ExtensionsT
oDeleteOnClean>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <LinkIncremental>true</LinkIncremental>
    <IgnoreImportLibrary>false</IgnoreImportLibrary>

<ExtensionsToDeleteOnClean>*.a;*.dll;$(ExtensionsToDeleteOnClean)</ExtensionsT
oDeleteOnClean>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <LinkIncremental>false</LinkIncremental>

<ExtensionsToDeleteOnClean>*.a;*.dll;$(ExtensionsToDeleteOnClean)</ExtensionsT
oDeleteOnClean>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <LinkIncremental>true</LinkIncremental>

<ExtensionsToDeleteOnClean>*.a;*.dll;$(ExtensionsToDeleteOnClean)</ExtensionsT
oDeleteOnClean>
  </PropertyGroup>
  <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <ClCompile>
      <PrecompiledHeader>Use</PrecompiledHeader>
      <WarningLevel>Level3</WarningLevel>
      <Optimization>Disabled</Optimization>

<PreprocessorDefinitions>WIN32;_DEBUG;_CONSOLE;_LIB;_CRT_SECURE_NO_WARNINGS;%(
PreprocessorDefinitions)</PreprocessorDefinitions>
      <SDLCheck>true</SDLCheck>

<AdditionalIncludeDirectories>C:\Projects\SBE\gmp\include;C:\Projects\SBE\open
ssl\include;C:\Projects\SBE\pbc\include;C:\Projects\IBE\openssl\lib;%(Addition
alIncludeDirectories)</AdditionalIncludeDirectories>
      <CompileAsManaged>true</CompileAsManaged>
    </ClCompile>
    <Link>
      <SubSystem>Console</SubSystem>
      <GenerateDebugInformation>true</GenerateDebugInformation>

<AdditionalDependencies>libgcc.a;libmingwex.a;libpbc.lib;libgmp.lib;libeay32.l
ib;ssleay32.lib;%(AdditionalDependencies)</AdditionalDependencies>

<AdditionalLibraryDirectories>$(SolutionDir)fodder;$(SolutionDir)lib;%(Additio
nalLibraryDirectories)</AdditionalLibraryDirectories>
      <AdditionalOptions>/SAFESEH:NO %(AdditionalOptions)</AdditionalOptions>

<LinkTimeCodeGeneration>UseLinkTimeCodeGeneration</LinkTimeCodeGeneration>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
    </Link>
    <PostBuildEvent>
      <Command>COPY /Y "$(SolutionDir)"\lib\*.DLL* "$(TargetDir)"</Command>
    </PostBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <ClCompile>
      <PrecompiledHeader>Use</PrecompiledHeader>
      <WarningLevel>Level3</WarningLevel>
      <Optimization>Disabled</Optimization>

<PreprocessorDefinitions>WIN32;_DEBUG;_CONSOLE;_LIB;%(PreprocessorDefinitions)
</PreprocessorDefinitions>
      <SDLCheck>true</SDLCheck>

<AdditionalIncludeDirectories>C:\Projects\SBE\pbc\include\pbc;%GMP%\include;%(
AdditionalIncludeDirectories)</AdditionalIncludeDirectories>
      <CompileAsManaged>true</CompileAsManaged>
    </ClCompile>
    <Link>
      <SubSystem>Console</SubSystem>
      <GenerateDebugInformation>true</GenerateDebugInformation>

<AdditionalDependencies>libgcc.a;libmingwex.a;libpbc.lib;libgmp.lib;%(Addition
alDependencies)</AdditionalDependencies>

<AdditionalLibraryDirectories>$(SolutionDir)fodder;$(SolutionDir)lib;%(Additio
nalLibraryDirectories)</AdditionalLibraryDirectories>
    </Link>
    <PostBuildEvent>
      <Command>COPY /Y "$(SolutionDir)"\lib\*.DLL* "$(TargetDir)"</Command>
    </PostBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <ClCompile>
      <WarningLevel>Level3</WarningLevel>
      <PrecompiledHeader>Use</PrecompiledHeader>
      <Optimization>MaxSpeed</Optimization>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <IntrinsicFunctions>true</IntrinsicFunctions>

<PreprocessorDefinitions>WIN32;NDEBUG;_CONSOLE;_LIB;_CRT_SECURE_NO_WARNINGS;%(
PreprocessorDefinitions)</PreprocessorDefinitions>
      <SDLCheck>true</SDLCheck>

<AdditionalIncludeDirectories>C:\Projects\SBE\gmp\include;C:\Projects\SBE\open
ssl\include;C:\Projects\SBE\pbc\include;C:\Projects\IBE\openssl\lib;%(Addition
alIncludeDirectories)</AdditionalIncludeDirectories>
      <CompileAsManaged>false</CompileAsManaged>
    </ClCompile>
    <Link>
      <SubSystem>Console</SubSystem>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <OptimizeReferences>true</OptimizeReferences>

<AdditionalDependencies>libgcc.a;libmingwex.a;libpbc.lib;libgmp.lib;ssleay32.l
ib;libeay32.lib;%(AdditionalDependencies)</AdditionalDependencies>

<AdditionalLibraryDirectories>$(SolutionDir)fodder;$(SolutionDir)lib;%(Additio
nalLibraryDirectories)</AdditionalLibraryDirectories>
      <AdditionalOptions>/SAFESEH:NO %(AdditionalOptions)</AdditionalOptions>
    </Link>
    <PostBuildEvent>
      <Command>COPY /Y "$(SolutionDir)"\lib\*.DLL "$(TargetDir)"</Command>
    </PostBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <ClCompile>
      <WarningLevel>Level3</WarningLevel>
      <PrecompiledHeader>Use</PrecompiledHeader>
      <Optimization>MaxSpeed</Optimization>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <IntrinsicFunctions>true</IntrinsicFunctions>

<PreprocessorDefinitions>WIN32;NDEBUG;_CONSOLE;_LIB;%(PreprocessorDefinitions)
</PreprocessorDefinitions>
      <SDLCheck>true</SDLCheck>

<AdditionalIncludeDirectories>C:\Projects\SBE\gmp\include;C:\Projects\SBE\pbc\
include;%(AdditionalIncludeDirectories)</AdditionalIncludeDirectories>
    </ClCompile>
    <Link>
      <SubSystem>Console</SubSystem>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <OptimizeReferences>true</OptimizeReferences>

<AdditionalDependencies>libgcc.a;libmingwex.a;libpbc.lib;libgmp.lib;%(Addition
alDependencies)</AdditionalDependencies>

<AdditionalLibraryDirectories>$(SolutionDir)fodder;$(SolutionDir)lib;%(Additio
nalLibraryDirectories)</AdditionalLibraryDirectories>
    </Link>
    <PostBuildEvent>
      <Command>COPY /Y "$(SolutionDir)"\lib\*.DLL "$(TargetDir)"</Command>
    </PostBuildEvent>
  </ItemDefinitionGroup>
  <ItemGroup>
    <Text Include="ReadMe.txt" />
  </ItemGroup>
  <ItemGroup>
    <ClInclude Include="stdafx.h" />
    <ClInclude Include="targetver.h" />
  </ItemGroup>
  <ItemGroup>
    <ClCompile Include="ConsoleApplication.cpp" />
    <ClCompile Include="stdafx.cpp">
      <PrecompiledHeader
Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">Create</PrecompiledH
eader>
      <PrecompiledHeader
Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">Create</PrecompiledHea
der>
      <PrecompiledHeader
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">Create</Precompile
dHeader>
      <PrecompiledHeader
Condition="'$(Configuration)|$(Platform)'=='Release|x64'">Create</PrecompiledH
eader>
    </ClCompile>
  </ItemGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
  <ImportGroup Label="ExtensionTargets">
  </ImportGroup>
</Project>
```

APPENDIX H IBE WITH ECC AND RSA EVALUATION – SBE\VC\CONSOLEAPPLICATION\CONSOLEAPPLICATION.CPP

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <assert.h>
#include <fcntl.h>
#include <math.h>
#include <string.h>
#include <openssl/aes.h>
#include <openssl/err.h>
#include <openssl/sha.h>
#include <openssl/ssl.h>
#include <pbc.h>
#include <windows.h>

static char *gen_file_date_time(char *file_text, char *file_extension)
{
      const int TIME_SIZE = 16;
      const char FORMATTER_DATETIME[] = "%Y%m%d-%H%M%S";
      const char FORMATTER_FILENAME[] = "%s%s.%s";
      time_t raw_time;
      struct tm *info;
      char staged_result[TIME_SIZE];
      char *file_name = (char*)malloc(TIME_SIZE + strlen(file_text) +
strlen(file_extension) + 1);

      time(&raw_time);
      info = localtime(&raw_time);

      strftime(staged_result, TIME_SIZE, FORMATTER_DATETIME, info);

      sprintf(file_name, FORMATTER_FILENAME, file_text, staged_result,
file_extension);

      return file_name;
}

static void gen_random(int size, char *arr)
{
      srand(time(0));
      arr[size + 1] = { 0 };
      for (int i = 0; i < size; i++)
      {
        arr[i] = (char)(rand() % 255 + 0);
      }
      arr[size] = '\0';
}

static int to_base64(const unsigned char *data, int data_sz, char
**base64encoded)
{
      BIO *bio, *b64;
      BUF_MEM *ptr_buffer;

      b64 = BIO_new(BIO_f_base64());
      bio = BIO_new(BIO_s_mem());
      bio = BIO_push(b64, bio);
      BIO_set_flags(bio, BIO_FLAGS_BASE64_NO_NL);
      BIO_write(b64, data, data_sz);

      BIO_flush(b64);
      BIO_get_mem_ptr(bio, &ptr_buffer);
      BIO_set_close(bio, BIO_NOCLOSE);
```

```
    *base64encoded = (char*)malloc((ptr_buffer->length + 1) * sizeof(char));
    memcpy(*base64encoded, ptr_buffer->data, ptr_buffer->length);
    (*base64encoded)[ptr_buffer->length] = '\0';

    BIO_free_all(bio);

    return (0);
}

size_t calcDecodeLength(const char* base64encoded)
{
    size_t len = strlen(base64encoded),
      padding = 0;

    if (base64encoded[len - 1] == '=' && base64encoded[len - 2] == '=')
      padding = 2;
    else if (base64encoded[len - 1] == '=')
      padding = 1;

    return (len * 3) / 4 - padding;
}

int from_base64(const char* base64encoded, unsigned char** data, size_t*
length)
{
    BIO *bio, *b64;

    int decodeLen = calcDecodeLength(base64encoded);
    *data = (uint8_t*)malloc(decodeLen);

    bio = BIO_new_mem_buf((char *)base64encoded, -1);
    b64 = BIO_new(BIO_f_base64());
    bio = BIO_push(b64, bio);

    BIO_set_flags(bio, BIO_FLAGS_BASE64_NO_NL);
    *length = BIO_read(bio, *data, strlen(base64encoded));

    BIO_free_all(bio);

    return (0);
}

static void pbc_pairing_init(pairing_t pairing, int argc, char **argv)
{
    char s[16384];
    FILE *fp = stdin;

    if (argc > 1) {
      fp = fopen(argv[1], "r");
      if (!fp) pbc_die("error opening %s", argv[1]);
    }
    size_t count = fread(s, 1, 16384, fp);
    if (!count) pbc_die("input error");
    fclose(fp);

    if (pairing_init_set_buf(pairing, s, count)) pbc_die("pairing init
failed");
}

static unsigned char *get_text_from_file(char *file_path, long *bytes_read)
{
    FILE *fp;
    unsigned char *txt;
    long file_sz;

    fp = stdin;
```

```
     fp = fopen(file_path, "rb");
     if (!fp) pbc_die("error opening %s", file_path);
     fseek(fp, 0L, SEEK_END);

     file_sz = ftell(fp);
     rewind(fp);

     txt = (unsigned char*)calloc(1, file_sz + 1);
     if (!txt)
     {
       fclose(fp);
       pbc_die("memory alloc fails for new ID!");
     }
     if (1 != fread(txt, file_sz, 1, fp))
     {
       fclose(fp);
       free(txt);
       pbc_die("ID read fails!");
     }
     fclose(fp);
     *bytes_read = file_sz;

     return txt;
}

static unsigned char *get_data_from_file(char *file_path, long *bytes_read)
{
     FILE *fp;
     unsigned char *data;
     long file_sz;

     fp = stdin;
     fp = fopen(file_path, "rb");
     if (!fp) pbc_die("error opening %s", file_path);
     fseek(fp, 0L, SEEK_END);

     file_sz = ftell(fp);
     rewind(fp);

     data = (unsigned char*)calloc(1, file_sz + 1);
     if (!data)
     {
       fclose(fp);
       pbc_die("memory alloc fails for new ID!");
     }
     if (1 != fread(data, file_sz, 1, fp))
     {
       fclose(fp);
       free(data);
       pbc_die("ID read fails!");
     }
     fclose(fp);
     *bytes_read = file_sz;

     return data;
}

static int ibe_encrypt(element_t r, element_t U, element_t P, element_t gid,
element_t mapped_id_hash_Qid, element_t Ppub, unsigned char *data, int
data_len, unsigned char *cipher, char *err)
{
     const int HASH_LEN = 32;
     char hash[HASH_LEN] = { 0 };
     unsigned char *gs;

     element_random(r);
     element_mul_zn(U, P, r);
     element_pairing(gid, mapped_id_hash_Qid, Ppub);
```

```
	element_pow_zn(gid, gid, r);
	gs = (unsigned char*)malloc(element_length_in_bytes(gid));

	element_to_bytes(gs, gid);

	if (SHA256((unsigned char*)gs, element_length_in_bytes(gid), (unsigned
char *)hash) == NULL)
	{
	  ERR_error_string(ERR_get_error(), err);
	  printf("%s\n", err);
	}

	//if (SHA1(gs, HASH_LEN, (unsigned char *)hash) == NULL)
	//{
	//	ERR_error_string(ERR_get_error(), err);
	//	printf("%s\n", err);
	//}
	for (int i = 0; i < data_len; i++)
	{
	  cipher[i] = data[i] ^ hash[i % HASH_LEN];
	}
	free(gs);

	return (0);
}

void aes256_encrypt(const unsigned char *data, const unsigned char *key,
unsigned char **cipher)
{
	AES_KEY enc_key;
	AES_set_encrypt_key(key, 256, &enc_key);
	AES_encrypt((unsigned char*)data, (unsigned char*)*cipher, &enc_key);
}

void aes256_decrypt(const unsigned char *cipher, const unsigned char *key,
unsigned char **data)
{
	AES_KEY dec_key;
	AES_set_decrypt_key(key, 256, &dec_key);
	AES_decrypt((unsigned char*)cipher, (unsigned char*)*data, &dec_key);
}

int aes_evp256_encrypt(unsigned char *plaintext, int plaintext_len, unsigned
char *key, unsigned char *iv, unsigned char *ciphertext)
{
	EVP_CIPHER_CTX *ctx;

	int len;

	int ciphertext_len;

	/* Create and initialise the context */
	if (!(ctx = EVP_CIPHER_CTX_new()))
	{
	  ;
	}

	if (1 != EVP_EncryptInit_ex(ctx, EVP_aes_256_cbc(), NULL, key, iv))
	{
	  ;
	}

	if (1 != EVP_EncryptUpdate(ctx, ciphertext, &len, plaintext,
plaintext_len))
	{
	  ;
	}
	ciphertext_len = len;
```

```
     if (1 != EVP_EncryptFinal_ex(ctx, ciphertext + len, &len))
     {
       ;
     }

     ciphertext_len += len;

     EVP_CIPHER_CTX_free(ctx);

     return ciphertext_len;
}

int aes_evp256_decrypt(unsigned char *ciphertext, int ciphertext_len, unsigned
char *key, unsigned char *iv, unsigned char *plaintext)
{
     EVP_CIPHER_CTX *ctx;

     int len;

     int plaintext_len;

     if (!(ctx = EVP_CIPHER_CTX_new()))
     {
       ;
     }

     if (1 != EVP_DecryptInit_ex(ctx, EVP_aes_256_cbc(), NULL, key, iv))
     {
       ;
     }

     if (1 != EVP_DecryptUpdate(ctx, plaintext, &len, ciphertext,
ciphertext_len))
     {
       ;
     }
     plaintext_len = len;

     if (1 != EVP_DecryptFinal_ex(ctx, plaintext + len, &len))
     {
       ;
     }

     plaintext_len += len;

     EVP_CIPHER_CTX_free(ctx);

     return plaintext_len;
}

static int ibe_bf_aes256_encrypt(element_t r, element_t U, element_t P,
element_t gid, element_t mapped_id_hash_Qid, element_t Ppub, unsigned char
*data, long data_len, unsigned char *cipher, char *err)
{
     const int HASH_LEN = 32;
     char hash[HASH_LEN] = { 0 };
     unsigned char *gs;

     element_random(r);
     element_mul_zn(U, P, r);

     /********
     ++++INIT_TIMING
     *********/
     //LARGE_INTEGER frequency;        // ticks per second
     //LARGE_INTEGER t1, t2;           // ticks
```

```
	//double time_spent;
	//QueryPerformanceFrequency(&frequency);
	/********
	----INIT_TIMING
	*********/
	//FILE *file_key_gen;
	//file_key_gen = fopen(gen_file_date_time("C:\\Projects\\Evaluation\\IBE-
Pairing_Policy-Public-", "csv"), "w");
	//for (int i = 0; i < 100; i++)
	{
	  /********
	  ++++BEG_TIMING
	  *********/
	  //QueryPerformanceCounter(&t1);
	  /********
	  ----BEG_TIMING
	  *********/
	  element_pairing(gid, mapped_id_hash_Qid, Ppub);

	  /********
	  ++++END_TIMING
	  *********/
	  //QueryPerformanceCounter(&t2);

	  //time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC /
frequency.QuadPart;
	  //printf("%f\n", time_spent);
	  //fprintf(file_key_gen, "%f;[ms]\n", time_spent);
	  /********
	  ----END_TIMING
	  *********/
	}

	element_pow_zn(gid, gid, r);
	gs = (unsigned char*)malloc(element_length_in_bytes(gid));

	element_to_bytes(gs, gid);

	if (SHA256((unsigned char*)gs, element_length_in_bytes(gid), (unsigned
char *)hash) == NULL)
	{
	  ERR_error_string(ERR_get_error(), err);
	  printf("%s\n", err);
	}
	unsigned char iv[128] = { 0 };
	int cipher_len = aes_evp256_encrypt((unsigned char*)data, data_len,
(unsigned char*)hash, iv, cipher);

	free(gs);

	return cipher_len;
}

static int ibe_bf_decrypt(element_t xt, element_t private_key_Did, element_t
U, unsigned char *data, unsigned char *cipher, int cipher_len, char *err)
{
	unsigned char *gs;
	const int HASH_LEN = 32;
	char hash[HASH_LEN] = { 0 };

	element_pairing(xt, private_key_Did, U);
	gs = (unsigned char*)malloc(element_length_in_bytes(xt));
	element_to_bytes(gs, xt);

	if (SHA256((unsigned char*)gs, element_length_in_bytes(xt), (unsigned
char *)hash) == NULL)
	{
	  ERR_error_string(ERR_get_error(), err);
```

```
        printf("%s\n", err);
      }
/*
      if (SHA1((unsigned char*)gs, HASH_LEN, (unsigned char *)hash) == NULL)
      {
        ERR_error_string(ERR_get_error(), err);
        printf("%s\n", err);
      }*/

      for (int i = 0; i < cipher_len; i++)
      {
        data[i] = cipher[i] ^ hash[i % HASH_LEN];
      }

      free(gs);
      return (0);
}

static int ibe_bf_aes256_decrypt(element_t xt, element_t private_key_Did,
element_t U, unsigned char *data, unsigned char *cipher, long cipher_len, char
*err)
{
      unsigned char *gs;
      const int HASH_LEN = 32;
      char hash[HASH_LEN] = { 0 };

      /********
      ++++INIT_TIMING
      *********/
      //LARGE_INTEGER frequency;        // ticks per second
      //LARGE_INTEGER t1, t2;           // ticks
      //double time_spent;
      //QueryPerformanceFrequency(&frequency);
      /********
      ----INIT_TIMING
      *********/
      //FILE *file_key_gen;
      //file_key_gen = fopen(gen_file_date_time("C:\\Projects\\Evaluation\\IBE-
Pairing_Policy-Private-", "csv"), "w");
      //for (int i = 0; i < 100; i++)
      {
        /********
        ++++BEG_TIMING
        *********/
        //QueryPerformanceCounter(&t1);
        /********
        ----BEG_TIMING
        *********/
        element_pairing(xt, private_key_Did, U);
        /********
        ++++END_TIMING
        *********/
        //QueryPerformanceCounter(&t2);

        //time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC /
frequency.QuadPart;
        //printf("%f\n", time_spent);
        //fprintf(file_key_gen, "%f;[ms]\n", time_spent);
        /********
        ----END_TIMING
        *********/
      }


      gs = (unsigned char*)malloc(element_length_in_bytes(xt));
      element_to_bytes(gs, xt);

      if (SHA256((unsigned char*)gs, element_length_in_bytes(xt), (unsigned
char *)hash) == NULL)
```

```
     {
       ERR_error_string(ERR_get_error(), err);
       printf("%s\n", err);
     }

     unsigned char iv[128] = { 0 };
     int data_len = aes_evp256_decrypt((unsigned char*)cipher, cipher_len,
(unsigned char*)hash, iv, data);

     free(gs);

     return data_len;
}

static int ibe_bf_set_public_key(const unsigned char *id, long id_size,
unsigned char *key, const int key_size, char *err)
{
     const int HASH_LEN = 32;
     unsigned char hash[HASH_LEN] = { 0 };

     key = (unsigned char *)malloc(key_size+1);

     if (SHA256(id, id_size, hash) == NULL)
     {
       ERR_error_string(ERR_get_error(), err);
       printf("%s\n", err);

       return -1;
     }
     for (int i = 0; i < key_size; i++)
     {
       key[i] = hash[i % HASH_LEN];
     }
     key[key_size] = '\0';

     return strlen((char *)key);
}

static int ibe_bf_pkg_gen(pairing_t pairing, element_t master_key, element_t
private_key)
{

}

static int element_serialize(element_t element, char **serialized_element)
{
     int element_sz = element_length_in_bytes(element);
     unsigned char *data = (byte*)malloc(element_sz);

     element_to_bytes(data, element);
     to_base64(data, element_sz, serialized_element);
     free(data);

     return (0);
}

static int element_deserialize(const char *serialized_element, element_t
element)
{
     unsigned char *data;
     size_t data_size;

     from_base64(serialized_element, &data, &data_size);
     element_from_bytes(element, (unsigned char *)data);

     free(data);
     return (0);
```

```
}

static inline void element_write(element_t elem, FILE *myfile)
{
     int sz = element_length_in_bytes(elem);
     fwrite(&sz, 4, 1, myfile);
     unsigned char* data = (unsigned char*)pbc_malloc(sz);
     if (!data) printf("DATA IS NULL\n");
     element_to_bytes(data, elem);
     fwrite(data, sz, 1, myfile);
     pbc_free(data);
}

static inline void element_read(element_t elem, FILE *myfile) {
     int sz;
     fread(&sz, 4, 1, myfile);
     unsigned char* data = (unsigned char*)pbc_malloc(sz);
     fread(data, sz, 1, myfile);
     element_from_bytes(elem, data);
     pbc_free(data);
}

static RSA *rsa_create_key_pair(unsigned char **public_key, int
*public_key_size, unsigned char **private_key, int *private_key_size)
{
     const int KEY_SIZE = 1024;
     const int PUB_EXP = 3;
     RSA *key_pair;

     key_pair = RSA_generate_key(KEY_SIZE, PUB_EXP, NULL, NULL);

     BIO *bio_private_key = BIO_new(BIO_s_mem());
     BIO *bio_public_key = BIO_new(BIO_s_mem());

     PEM_write_bio_RSAPrivateKey(bio_private_key, key_pair, NULL, NULL, 0,
NULL, NULL);
     PEM_write_bio_RSAPublicKey(bio_public_key, key_pair);

     *private_key_size = BIO_pending(bio_private_key);
     *public_key_size = BIO_pending(bio_public_key);

     *private_key = (unsigned char *)malloc(*private_key_size);
     *public_key = (unsigned char *)malloc(*public_key_size);

     BIO_read(bio_private_key, *private_key, *private_key_size);
     BIO_read(bio_public_key, *public_key, *public_key_size);

     return key_pair;
}

static void ibe_eval(int argc, char **argv)
{
     const int KEY_SZ = 32; //Bytes
     const int KEY_SPACE_LENGTH = 256;
     pairing_t pairing;
     element_t gen_P, Ppub, private_key_Did, mapped_id_hash_Qid, U, r, xt,
gid;
     element_t master_key_s;

     unsigned char *key = (unsigned char *)malloc(KEY_SZ);
     char err[80] = { 0 };

     long data_sz;
     long id_sz;

     unsigned char *gs = NULL;
```

```
	unsigned char *id =
get_text_from_file("C:\\Projects\\SBE\\vc\\ConsoleApplication\\IIA007Policy.xm
l", &id_sz);
	unsigned char *data =
get_data_from_file("C:\\Projects\\SBE\\vc\\ConsoleApplication\\ProgressReportT
emplate.docx", &data_sz);

	unsigned char *cipher = (unsigned char *)malloc(data_sz);
	int cipher_sz;
	unsigned char *mv = (byte*)malloc(data_sz);

	LARGE_INTEGER frequency;        // ticks per second
	LARGE_INTEGER t1, t2;           // ticks
	double time_spent;
	QueryPerformanceFrequency(&frequency);

	/***
	errors strings initialization for SHA1 & clock initialization for times
computation
	***/
	ERR_load_crypto_strings();
	SSL_load_error_strings();

	printf("IBE\n\n");

	/***
	pairing function initalization from the input file which contains the
pairing parameters
	***/
	pbc_pairing_init(pairing, argc, argv);
	if (!pairing_is_symmetric(pairing)) pbc_die("pairing must be symmetric");

	//G1
	element_init_G1(gen_P, pairing);
	element_init_G1(Ppub, pairing);
	element_init_G1(mapped_id_hash_Qid, pairing);
	element_init_G1(private_key_Did, pairing);
	element_init_G1(U, pairing);

	//Zr
	element_init_Zr(master_key_s, pairing);
	element_init_Zr(r, pairing);

	//GT
	element_init_GT(gid, pairing);
	element_init_GT(xt, pairing);

	//PKG generation of P, master_key_s and Ppub
	element_random(gen_P);
	element_random(master_key_s);

	//FILE *file_key_gen;
	//file_key_gen = fopen(gen_file_date_time("C:\\Projects\\Evaluation\\IBE-
Key_Gen_Policy-Public-", "csv"), "w");
	//for (int i = 0; i < 100; i++)
	{
	  /********
	  ++++BEG_TIMING
	  *********/
//	  QueryPerformanceCounter(&t1);
	  /********
	  ----BEG_TIMING
	  *********/
	  element_mul_zn(Ppub, gen_P, master_key_s);
	  //element_printf("++s: %B\n", master_key_s);
	  //element_printf("++P:  %B\n", gen_P);
	  //element_printf("++Ppub: %B\n", Ppub);
```

```
        /********
        ++++END_TIMING
        *********/
        //QueryPerformanceCounter(&t2);

        //time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC /
frequency.QuadPart;
        //printf("%f\n", time_spent);
        //fprintf(file_key_gen, "%f;[ms]\n", time_spent);
        /********
        ----END_TIMING
        *********/
      }

      //generate 256 long key
      int pkey_sz = ibe_bf_set_public_key(id, id_sz, key, KEY_SPACE_LENGTH,
err);

/*    char *b64IdHash;
      to_base64((unsigned char *)key, KEY_SZ, &b64IdHash);
      printf("++idHash: %s\n", b64IdHash);
*/
      //FILE *file_key_gen;
      //file_key_gen = fopen(gen_file_date_time("C:\\Projects\\Evaluation\\IBE-
Key_Gen_Policy-Private-", "csv"), "w");
      //for (int i = 0; i < 100; i++)
      {
        /********
        ++++BEG_TIMING
        *********/
        //QueryPerformanceCounter(&t1);
        /********
        ----BEG_TIMING
        *********/
        element_from_hash(mapped_id_hash_Qid, key, pkey_sz);
        /********
        ++++END_TIMING
        *********/
        //QueryPerformanceCounter(&t2);

        //time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC /
frequency.QuadPart;
        //printf("%f\n", time_spent);
        //fprintf(file_key_gen, "%f;[ms]\n", time_spent);
        /********
        ----END_TIMING
        *********/
      }

      /*      char *serialized;
      element_serialize(master_key_s, &serialized);
      printf("+++serialized: %s", serialized);
      free(serialized);

      element_deserialize(serialized, master_key_s);
      */


      //FILE *file_key_gen;
      //file_key_gen = fopen(gen_file_date_time("C:\\Projects\\Evaluation\\IBE-
AES256-Enc_Policy-", "csv"), "w");
      //for (int i = 0; i < 100; i++)
      {
        /********
        ++++BEG_TIMING
        *********/
      //      QueryPerformanceCounter(&t1);
        /********
        ----BEG_TIMING
```

```
	*********/

	//ENCRYPTION
	//ibe_encrypt(r, U, P, gid, mapped_id_hash_Qid, Ppub, data, data_sz, 
cipher, err);
	cipher_sz = ibe_bf_aes256_encrypt(r, U, gen_P, gid, mapped_id_hash_Qid, 
Ppub, data, data_sz, cipher, err);
	//char *b64MsgHash;
	//to_base64(cipher, data_sz, &b64MsgHash);
	//element_printf("++m: %s\n", b64MsgHash);

	/********
	++++END_TIMING
	*********/
	//	QueryPerformanceCounter(&t2);
	///********
	//----END_TIMING
	//*********/

	//	time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC 
/ frequency.QuadPart;
	//	printf("%f\n", time_spent);
	//	fprintf(file_key_gen, "%f;[ms]\n", time_spent);
	}

	element_mul_zn(private_key_Did, mapped_id_hash_Qid, master_key_s);
	//element_printf("++Qid: %B\n", mapped_id_hash_Qid);
	//element_printf("++Did: %B\n", private_key_Did);

	//FILE *file_key_gen;
	//file_key_gen = fopen(gen_file_date_time("C:\\Projects\\Evaluation\\IBE-
Dec_Policy-", "csv"), "w");
	//for (int i = 0; i < 100; i++)
	//{
	/********
	++++BEG_TIMING
	*********/
	//QueryPerformanceCounter(&t1);
	/********
	----BEG_TIMING
	*********/
	int data_len = ibe_bf_aes256_decrypt(xt, private_key_Did, U, mv, cipher, 
cipher_sz, err);
	/********
	++++END_TIMING
	*********/
	//QueryPerformanceCounter(&t2);
	/********
	----END_TIMING
	*********/

	//	time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC 
/ frequency.QuadPart;
	//	printf("%f\n", time_spent);
	//	fprintf(file_key_gen, "%f;[ms]\n", time_spent);
	//}
	//printf("\n%d\n", memcmp(data, mv, data_len));

	/***free mem***/
	element_clear(gen_P);
	element_clear(Ppub);
	element_clear(mapped_id_hash_Qid);
	element_clear(private_key_Did);
	element_clear(U);
	element_clear(gid);
	element_clear(r);
	element_clear(xt);
	element_clear(master_key_s);
	pairing_clear(pairing);
	free(gs);
	free(id);
	free(key);
	//free(cipher);
	free(data);
}

static void rsa_eval()
{
	const int KEY_SPACE_LENGTH = 256;
	const int AES_KEY_SZ = 256;
	const int RSA_KEY_SZ = 4096;
	const int PUB_EXP = 3;

	unsigned char aes_key_seed[AES_KEY_SZ] = { 1 };
	unsigned char aes_key[AES_KEY_SZ];
	unsigned char aes_key_v[AES_KEY_SZ];

	printf("RSA\n\n");


	LARGE_INTEGER frequency;        // ticks per second
	LARGE_INTEGER t1, t2;           // ticks
	double time_spent;
	QueryPerformanceFrequency(&frequency);

	//Key_Gen
	int aes_key_bytes = ibe_bf_set_public_key(aes_key_seed, AES_KEY_SZ,
aes_key, KEY_SPACE_LENGTH, NULL);

	//FILE *file_key_gen;
	//file_key_gen = fopen(gen_file_date_time("C:\\Projects\\Evaluation\\RSA-
Key_Gen-4096-", "csv"), "w");
	//for (int i = 0; i < 100; i++)
	//{
	  /********
	  ++++BEG_TIMING
	  *********/
	//	QueryPerformanceCounter(&t1);
	  /********
	  ----BEG_TIMING
	  *********/

	//	RSA *key_pair = RSA_generate_key(RSA_KEY_SZ, PUB_EXP, NULL,
NULL);

	  /********
	  ++++END_TIMING
	  *********/
	//	QueryPerformanceCounter(&t2);
	  ///********
	  //----END_TIMING
	  //*********/

	//	RSA_free(key_pair);
	//	time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC
/ frequency.QuadPart;
	//	printf("%f\n", time_spent);
	//	fprintf(file_key_gen, "%f;[ms]\n", time_spent);
	//}

	RSA *key_pair = RSA_generate_key(RSA_KEY_SZ, PUB_EXP, NULL, NULL);
	unsigned char *cipher = (unsigned char*)malloc(RSA_size(key_pair));

	//FILE *file_key_gen;
	//file_key_gen = fopen(gen_file_date_time("C:\\Projects\\Evaluation\\RSA-
Enc_AES256Key-4096-", "csv"), "w");
	//for (int i = 0; i < 100; i++)
```

```
     //{
     /********
     ++++BEG_TIMING
     *********/
//   QueryPerformanceCounter(&t1);
     /********
     ----BEG_TIMING
     *********/

     int size_enc = RSA_public_encrypt(AES_KEY_SZ, aes_key, cipher, key_pair,
RSA_PKCS1_PADDING);
     /********
     ++++END_TIMING
     *********/
//   QueryPerformanceCounter(&t2);
     /********
     ----END_TIMING
     *********/

     //      time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC
/ frequency.QuadPart;
     //      printf("%f\n", time_spent);
     //      fprintf(file_key_gen, "%f;[ms]\n", time_spent);
     //}
     //fclose(file_key_gen);

     //printf("%d\n", size_enc);

     //FILE *file_key_gen;
     //file_key_gen = fopen(gen_file_date_time("C:\\Projects\\Evaluation\\RSA-
Dec_AES256Key-4096-", "csv"), "w");
     //for (int i = 0; i < 100; i++)
     //{
       /********
       ++++BEG_TIMING
       *********/
       //QueryPerformanceCounter(&t1);
       /********
       ----BEG_TIMING
       *********/
       int size_dec = RSA_private_decrypt(size_enc, cipher, aes_key_v,
key_pair, RSA_PKCS1_PADDING);

       /********
       ++++END_TIMING
       *********/
       //QueryPerformanceCounter(&t2);
       /********
       ----END_TIMING
       *********/

     //      time_spent = (t2.QuadPart - t1.QuadPart) * (double)CLOCKS_PER_SEC
/ frequency.QuadPart;
     //      printf("%f\n", time_spent);
     //      fprintf(file_key_gen, "%f;[ms]\n", time_spent);
     //}
     //fclose(file_key_gen);

     //      printf("\n%d\n", memcmp(aes_key, aes_key_v, size_dec));

     /*for (int i = 0; i < size_enc; i++)
     {
       printf("\n%d  -  %d::%d", i, aes_key[i], aes_key_v[i]);
     }*/

     RSA_free(key_pair);
     //free(aes_key);
     //free(aes_key_v);
     //free(cipher);
     //free(rsa_private_key);
     //free(rsa_public_key);
}

int main(int argc, char **argv)
{
     ibe_eval(argc, argv);
     rsa_eval();

     return 0;
}
```

APPENDIX I XACML XML POLICY (1 OF 400)

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
      xmlns="urn:oasis:names:tc:xacml:1.0:policy"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
        cs-xacml-schema-policy-01.xsd"
      PolicyId="urn:oasis:names:tc:xacml:1.0:conformance-test:IIA002:policy"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
    <Description>
        Policy for Conformance Test IIA002.
    </Description>
    <Target>
        <Subjects>
            <AnySubject/>
        </Subjects>
        <Resources>
            <AnyResource/>
        </Resources>
        <Actions>
            <AnyAction/>
        </Actions>
    </Target>
    <Rule
          RuleId="urn:oasis:names:tc:xacml:1.0:conformance-test:IIA002:rule"
          Effect="Permit">
        <Description>
            A subject with a role attribute of "Physician" can read or
            write Bart Simpson's medical record.
        </Description>
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#string">Physician</AttributeValue>
                        <SubjectAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:example:attribute:role"

DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </SubjectMatch>
                </Subject>
            </Subjects>
            <Resources>
                <Resource>
                    <ResourceMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
                        <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#anyURI">http://medico.com/record/pa
tient/BartSimpson</AttributeValue>
                        <ResourceAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"

DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
                    </ResourceMatch>
                </Resource>
            </Resources>
            <Actions>
                <Action>
```

```
                    <ActionMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
                        <ActionAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </ActionMatch>
                </Action>
                <Action>
                    <ActionMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#string">write</AttributeValue>
                        <ActionAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </ActionMatch>
                </Action>
            </Actions>
        </Target>
    </Rule>
</Policy>
```

APPENDIX J XACML XML POLICY REQUEST (1 OF 400)

```
<?xml version="1.0" encoding="UTF-8"?>
<Request
      xmlns="urn:oasis:names:tc:xacml:1.0:context"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
        cs-xacml-schema-context-01.xsd">
    <Subject>
        <Attribute
              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
              DataType="http://www.w3.org/2001/XMLSchema#string">
            <AttributeValue>B65172AD-4D9E-4440-8745-
2AC3C1B9FC49</AttributeValue>
        </Attribute>
    </Subject>
    <Resource>
        <Attribute
              AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
              DataType="http://www.w3.org/2001/XMLSchema#anyURI">
            <AttributeValue>http://nhs.uk/services/API/FE5A064C-0C55-449F-
9EB4-45596370AE96</AttributeValue>
        </Attribute>
    </Resource>
    <Action>
        <Attribute
              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
              DataType="http://www.w3.org/2001/XMLSchema#string">
            <AttributeValue>read</AttributeValue>
        </Attribute>
    </Action>
</Request>
```

APPENDIX K XACML XML POLICY RESPONSE (1 OF 400)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Response
      xmlns="urn:oasis:names:tc:xacml:1.0:context"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
        cs-xacml-schema-context-01.xsd">
    <Result>
        <Decision>Permit</Decision>
        <Status>
            <StatusCode
                  Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
```

APPENDIX L XACML JSON-FORMATTED POLICY (1 OF 400)

```
{
    "?xml": {
        "@version": "1.0",
        "@encoding": "UTF-8"
    },
    "Policy": {
        "@xmlns": "urn:oasis:names:tc:xacml:1.0:policy",
        "@xmlns:xacml-context": "urn:oasis:names:tc:xacml:1.0:context",
        "@xmlns:md": "http://www.medico.com/schemas/record",
        "@xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
        "@xsi:schemaLocation": "urn:oasis:names:tc:xacml:1.0:policy\r\n
cs-xacml-schema-policy-01.xsd",
        "@PolicyId": "urn:oasis:names:tc:xacml:1.0:conformance-
test:IIIG006:policy",
        "@RuleCombiningAlgId": "urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides",
        "Description": "Evaluation Policy 398",
        "PolicyDefaults": {
            "XPathVersion": "http://www.w3.org/TR/1999/Rec-xpath-19991116"
        },
        "Target": {
            "Subjects": {
                "AnySubject": null
            },
            "Resources": {
                "AnyResource": null
            },
            "Actions": {
                "AnyAction": null
            }
        },
        "Rule": {
            "@RuleId": "urn:oasis:names:tc:xacml:1.0:conformance-
test:IIIG006:rule",
            "@Effect": "Permit",
            "Condition": {
                "@FunctionId": "urn:oasis:names:tc:xacml:1.0:function:and",
                "Apply": [
                          {
                          "@FunctionId":
"urn:oasis:names:tc:xacml:1.0:function:integer-equal",
                          "Apply": {
                          "@FunctionId":
"urn:oasis:names:tc:xacml:1.0:function:xpath-node-count",
                          "AttributeValue": {
                          "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                          "#text": "./xacml-context:Resource/xacml-
context:ResourceContent/md:record//md:name"
                          }
                          },
                          "AttributeValue": {
                          "@DataType":
"http://www.w3.org/2001/XMLSchema#integer",
                          "#text": "2"
                          }
                          },
                          {
                          "@FunctionId":
"urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal",
                          "AttributeValue": [
                                                   {
                                                   "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                                                   "#text": "./xacml-
context:Resource/xacml-context:ResourceContent/md:record"
```

```
                                                                    },
                                                                    {
                                                                    "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                                                                    "#text": "//md:record"
                                                                    }
                                                                    ]
                                    },
                                    {
                                    "@FunctionId":
"urn:oasis:names:tc:xacml:1.0:function:xpath-node-match",
                                    "AttributeValue": [
                                                                    {
                                                                    "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                                                                    "#text": "."
                                                                    },
                                                                    {
                                                                    "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                                                                    "#text": "./xacml-
context:Resource/xacml-context:ResourceContent/md:record"
                                                                    }
                                                                    ]
                                    }
                                    ]
                        }
                    }
                }
            }
```

APPENDIX M XACML JSON-FORMATTED POLICY REQUEST (1 OF 400)

```
{
    "?xml": {
        "@version": "1.0",
        "@encoding": "UTF-8"
    },
    "Request": {
        "@xmlns": "urn:oasis:names:tc:xacml:1.0:context",
        "@xmlns:md": "http://www.medico.com/schemas/record",
        "@xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
        "@xsi:schemaLocation": "urn:oasis:names:tc:xacml:1.0:context\r\n
cs-xacml-schema-context-01.xsd",
        "Subject": {
            "Attribute": [
                          {
                          "@AttributeId":
"urn:oasis:names:tc:xacml:1.0:subject:subject-id",
                          "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                          "AttributeValue": "B65172AD-4D9E-4440-8745-
2AC3C1B9FC49"
                          },
                          {
                          "@AttributeId":
"urn:oasis:names:tc:xacml:1.0:conformance-eva:test-attr",
                          "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                          "AttributeValue": {
                          "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                          "#text": "Evaluation 398"
                          }
                          },
                          {
                          "@AttributeId":
"urn:oasis:names:tc:xacml:1.0:conformance-test:test-attr",
                          "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                          "AttributeValue": {
                          "@DataType":
"http://www.w3.org/2001/XMLSchema#string",
                          "#text": "Evaluation 398"
                          }
                          }
                          ]
        },
        "Resource": {
            "ResourceContent": {
                "md:record": {
                    "md:hospital_info": {
                        "md:name": "Balmoral Hospital",
                        "md:department": "Urology"
                    },
                    "md:patient_info": {
                        "md:name": "James Lovelock",
                        "md:age": "80",
                        "md:sex": "male",
                        "md:health_insurance": "201510120000"
                    },
                    "md:diagnosis_info": {
                        "md:diagnosis": {
                            "md:item": [
                                         {
                                         "@type": "primary",
                                         "#text": " Acute Bacterial
Prostatitis"
                                         },
                                        {
                                        "@type": "secondary",
                                        "#text": "Asymptomatic Inflammatory
Prostatitis"

                                        }
                                        ]
                        },
                        "md:pathological_diagnosis": {
                            "md:diagnosis": {
                                "md:item": {
                                    "@type": "primary",
                                    "#text": "Acute Bacterial Prostatitis
(Prostatitis (Inflammation of the Prostate Gland))"
                                }
                            },
                            "md:date": "2015-10-05",
                            "md:malignancy": {
                                "@type": "yes"
                            }
                        }
                    }
                }
            },
            "Attribute": {
                "@AttributeId":
"urn:oasis:names:tc:xacml:1.0:resource:resource-id",
                "@DataType": "http://www.w3.org/2001/XMLSchema#anyURI",
                "AttributeValue": "http://nhs.uk/services/API/FE5A064C-0C55-
449F-9EB4-45596370AE96"
            }
        },
        "Action": {
            "Attribute": {
                "@AttributeId": "urn:oasis:names:tc:xacml:1.0:action:action-
id",
                "@DataType": "http://www.w3.org/2001/XMLSchema#string",
                "AttributeValue": "read"
            }
        }
    }
}
```

```
{
    "?xml": {
        "@version": "1.0",
        "@encoding": "UTF-8"
    },
    "Response": {
        "@xmlns": "urn:oasis:names:tc:xacml:1.0:context",
        "@xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
        "@xsi:schemaLocation": "urn:oasis:names:tc:xacml:1.0:context\r\n
cs-xacml-schema-context-01.xsd",
        "Result": {
            "Decision": "Permit",
            "Status": {
                "StatusCode": {
                    "@Value": "urn:oasis:names:tc:xacml:1.0:status:ok"
                }
            }
        }
    }
}
```

APPENDIX O   OOXML HELPER

```
#***********************************************************************
*
# Script Name: ModelEvaluation.py
# Language: Python
# Author: Greg Spyra      G$
#
# Project: Sticky-Policies Eval400
# Date Written:
# Reason:Sticky-Policies Model Workflows
# Notes:
# Version: 1.0
# Version History:
# 1.0G$      25 Jan 2016
#***********************************************************************
*
#!/usr/bin/python


#**********************************Variables*******************************
*
CONST_WORD_DOC_PATH = 'word/document.xml'
CONST_DOC_PATH = 'Z:\\University\\Publications\\Dissertation_PhD-
2013\\Thesis\\Evaluation\\400\\XUnitTest.zip'

#**********************************Functions*******************************
*
#************************************
#extract_word_doc_path extracts doc section from OOXML file
#************************************
def extract_word_doc_path(full_doc_path):
     file_handle = open(full_doc_path, 'rb')
     with zipfile.ZipFile(file_handle) as zip_file:
       for arch_elem in zip_file.namelist():
              if arch_elem.startswith(CONST_WORD_DOC_PATH):
                     zip_file.extract(arch_elem)

#************************************
#zip_word_doc_path encapsulates doc section into OOXML file
#************************************
def zip_word_doc_path(full_doc_path, full_word_doc_path, word_doc_path):
     with zipfile.ZipFile(full_doc_path, 'w') as zip_file:
       zip_file.write(full_word_doc_path, word_doc_path, zipfile.ZIP_STORED)

#************************************Main**********************************
*
import zipfile

extract_word_doc_path(CONST_DOC_PATH)
#zip_word_doc_path(CONST_DOC_PATH, CONST_WORD_DOC_PATH, CONST_WORD_DOC_PATH)




#***********************************************************************
*
#***********************************************************************
*
#***********************************************************************
*
#***********************************************************************
*
```

APPENDIX P XACML POLICY TESTER – XACML-EVALUATION\MAINCLASS.CS

```
using Xacml.Core;
using Xacml.Core.Runtime;
using System;
using System.Xml;

namespace XacmlTest
{
  internal class MainClass
  {
    private static void Main(string[] args)
    {
      string policyDocument = string.Empty;
      string contextDocument = string.Empty;
      bool verbose = false;
      foreach (string str in args)
      {
        if ((int) str[0] == 47 || (int) str[0] == 45)
        {
          if ((int) str[1] == 112 || (int) str[1] == 80)
            policyDocument = str.Substring(3);
          if ((int) str[1] == 114 || (int) str[1] == 82)
            contextDocument = str.Substring(3);
          if ((int) str[1] == 118 || (int) str[1] == 86)
            verbose = true;
        }
      }
      try
      {
        if (contextDocument.Length == 0 || policyDocument.Length == 0)
          throw new Exception("Request or policy file not specified.");
        new EvaluationEngine(verbose).Evaluate(policyDocument,
contextDocument, XacmlVersion.Version11).WriteDocument((XmlWriter) new
XmlTextWriter(Console.Out)
        {
          Formatting = Formatting.Indented
        });
      }
      catch (Exception ex)
      {
        Console.WriteLine(ex.Message);
        Console.WriteLine();
        Console.WriteLine("Usage:");
        Console.WriteLine("\t-p:[policyFilePath]  - The path to the policy
file");
        Console.WriteLine("\t-r:[requestFilePath] - The path to the request
file");
        Console.WriteLine("\t-v                   - Makes the execution
verbose");
      }
    }
  }
}
```

APPENDIX Q OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\ISTICKY.SLN (VISUAL STUDIO SOLUTION)

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 2013
VisualStudioVersion = 12.0.30501.0
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "ConsoleApplication",
"ConsoleApplication\ConsoleApplication.csproj", "{EC1FD466-EB9A-46EC-8B0C-
AF7DF9AC866E}"
EndProject
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "AppHandler",
"AppHandler\AppHandler.csproj", "{7F9EEEB1-3698-4AB5-BE66-8B82F7161A8A}"
EndProject
Project("{2150E333-8FDC-42A3-9474-1A3956D46DE8}") = "pep", "pep", "{496C2933-
A2CA-4E11-9E44-10753B15E0BD}"
EndProject
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "UnitTest",
"UnitTest\UnitTest.csproj", "{8E832E59-61E2-4C07-B80B-72182C913427}"
EndProject
Project("{E24C65DC-7377-472B-9ABA-BC803B73C61A}") = "Web",
"http://localhost:1607", "{131694CE-5D98-41D6-BA98-918E4923A288}"
     ProjectSection(WebsiteProperties) = preProject
       UseIISExpress = "true"
       TargetFrameworkMoniker = ".NETFramework,Version%3Dv3.5"
       Debug.AspNetCompiler.VirtualPath = "/localhost_1607"
       Debug.AspNetCompiler.PhysicalPath = "Web\"
       Debug.AspNetCompiler.TargetPath = "PrecompiledWeb\localhost_1607\"
       Debug.AspNetCompiler.Updateable = "true"
       Debug.AspNetCompiler.ForceOverwrite = "true"
       Debug.AspNetCompiler.FixedNames = "false"
       Debug.AspNetCompiler.Debug = "True"
       Release.AspNetCompiler.VirtualPath = "/localhost_1607"
       Release.AspNetCompiler.PhysicalPath = "Web\"
       Release.AspNetCompiler.TargetPath = "PrecompiledWeb\localhost_1607\"
       Release.AspNetCompiler.Updateable = "true"
       Release.AspNetCompiler.ForceOverwrite = "true"
       Release.AspNetCompiler.FixedNames = "false"
       Release.AspNetCompiler.Debug = "False"
       SlnRelativePath = "Web\"
     EndProjectSection
EndProject
Global
     GlobalSection(SubversionScc) = preSolution
       Svn-Managed = True
       Manager = AnkhSVN - Subversion Support for Visual Studio
     EndGlobalSection
     GlobalSection(SolutionConfigurationPlatforms) = preSolution
       Debug|Any CPU = Debug|Any CPU
       Release|Any CPU = Release|Any CPU
     EndGlobalSection
     GlobalSection(ProjectConfigurationPlatforms) = postSolution
       {EC1FD466-EB9A-46EC-8B0C-AF7DF9AC866E}.Debug|Any CPU.ActiveCfg =
Debug|Any CPU
       {EC1FD466-EB9A-46EC-8B0C-AF7DF9AC866E}.Debug|Any CPU.Build.0 =
Debug|Any CPU
       {EC1FD466-EB9A-46EC-8B0C-AF7DF9AC866E}.Release|Any CPU.ActiveCfg =
Release|Any CPU
       {EC1FD466-EB9A-46EC-8B0C-AF7DF9AC866E}.Release|Any CPU.Build.0 =
Release|Any CPU
       {7F9EEEB1-3698-4AB5-BE66-8B82F7161A8A}.Debug|Any CPU.ActiveCfg =
Debug|Any CPU
       {7F9EEEB1-3698-4AB5-BE66-8B82F7161A8A}.Debug|Any CPU.Build.0 =
Debug|Any CPU
       {7F9EEEB1-3698-4AB5-BE66-8B82F7161A8A}.Release|Any CPU.ActiveCfg =
Release|Any CPU
       {7F9EEEB1-3698-4AB5-BE66-8B82F7161A8A}.Release|Any CPU.Build.0 =
Release|Any CPU
```

```
		{8E832E59-61E2-4C07-B80B-72182C913427}.Debug|Any CPU.ActiveCfg =
Debug|Any CPU
		{8E832E59-61E2-4C07-B80B-72182C913427}.Debug|Any CPU.Build.0 =
Debug|Any CPU
		{8E832E59-61E2-4C07-B80B-72182C913427}.Release|Any CPU.ActiveCfg =
Release|Any CPU
		{8E832E59-61E2-4C07-B80B-72182C913427}.Release|Any CPU.Build.0 =
Release|Any CPU
		{131694CE-5D98-41D6-BA98-918E4923A288}.Debug|Any CPU.ActiveCfg =
Debug|Any CPU
		{131694CE-5D98-41D6-BA98-918E4923A288}.Debug|Any CPU.Build.0 =
Debug|Any CPU
		{131694CE-5D98-41D6-BA98-918E4923A288}.Release|Any CPU.ActiveCfg =
Debug|Any CPU
		{131694CE-5D98-41D6-BA98-918E4923A288}.Release|Any CPU.Build.0 =
Debug|Any CPU
	EndGlobalSection
	GlobalSection(SolutionProperties) = preSolution
		HideSolutionNode = FALSE
	EndGlobalSection
	GlobalSection(NestedProjects) = preSolution
		{EC1FD466-EB9A-46EC-8B0C-AF7DF9AC866E} = {496C2933-A2CA-4E11-9E44-
10753B15E0BD}
		{7F9EEEB1-3698-4AB5-BE66-8B82F7161A8A} = {496C2933-A2CA-4E11-9E44-
10753B15E0BD}
		{131694CE-5D98-41D6-BA98-918E4923A288} = {496C2933-A2CA-4E11-9E44-
10753B15E0BD}
	EndGlobalSection
EndGlobal
```

APPENDIX R OOXML Sticky Policy Handler evaluation – iSticky\AppHandler\AppHandler.csproj (Visual Studio Project)

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="12.0" DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import
Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.prop
s"
Condition="Exists('$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.C
ommon.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == ''
">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{7F9EEEB1-3698-4AB5-BE66-8B82F7161A8A}</ProjectGuid>
    <OutputType>Library</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>AppHandler</RootNamespace>
    <AssemblyName>AppHandler</AssemblyName>
    <TargetFrameworkVersion>v4.5</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU'
">
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' ==
'Release|AnyCPU' ">
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup>
    <SignAssembly>true</SignAssembly>
  </PropertyGroup>
  <PropertyGroup>
    <AssemblyOriginatorKeyFile>keyPair.snk</AssemblyOriginatorKeyFile>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="DocumentFormat.OpenXml, Version=2.5.5631.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL"
/>
    <Reference Include="System" />
    <Reference Include="System.Core" />
    <Reference Include="System.Transactions" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data.DataSetExtensions" />
    <Reference Include="Microsoft.CSharp" />
    <Reference Include="System.Data" />
    <Reference Include="System.Xml" />
    <Reference Include="WindowsBase" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="CandyDelivery\FileStreamer.cs" />
    <Compile Include="CandyDelivery\SingleStream.cs" />
    <Compile Include="CandyStore\Candy.cs" />
    <Compile Include="CandyStore\PDF\Document.cs" />
    <Compile Include="CandyStore\PDF\HPDF.cs" />
    <Compile Include="CandyStore\ICandy.cs" />
    <Compile Include="CandyStore\IDocument.cs" />
    <Compile Include="CandyStore\OOXML\Document.cs" />
    <Compile Include="CandyStore\OOXML\Master.cs" />
    <Compile Include="Properties\AssemblyInfo.cs" />
  </ItemGroup>
  <ItemGroup>
    <None Include="keyPair.snk" />
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
</Project>
```

APPENDIX S OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\APPHANDLER\CANDYSTORE\IDOCUMENT.CS (PEP.APPHANDLER.CANDYSTORE.IDOCUMENT INTERFACE)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace pep.AppHandler.CandyStore
{
     interface IDocument
     {

     }
}
```

APPENDIX T OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\APPHANDLER\CANDYSTORE\ICANDY.CS (PEP.APPHANDLER.CANDYSTORE.ICANDY INTERFACE)

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace pep.AppHandler.CandyStore
{
     /// <summary>
     /// Interface for wrapped data content
     /// Delivers secure execution space for sticky policy data
     /// </summary>
     public interface ICandy : IDisposable
     {

       /// <summary>
       /// Builds interpreted XACML master document wrapper
       /// </summary>
       /// <param name="documents">Documents file streams</param>
       void Wrap(List<FileStream> documents);

//     void PlantFile(byte[] FileContent);
     }
}
```

APPENDIX U OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\APPHANDLER\CANDYSTORE\CANDY.CS (PEP.APPHANDLER.CANDYSTORE.CANDY CLASS)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace pep.AppHandler.CandyStore
{
     /// <summary>
     /// Class provides information about different documents (candies)
supported
     /// </summary>
     public class Candy
     {
       #region Enums
       /// <summary>
       /// Enumerator for supported document types
       /// </summary>
       enum Type
       {
              OOXML = 0x01,
              PDF = 0x02
       };
       #endregion
     }
}
```

APPENDIX V OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\APPHANDLER\CANDYSTORE\PDF\DOCUMENT.CS (PEP.APPHANDLER.CANDYSTORE.PDF.DOCUMENT CLASS)

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace pep.AppHandler.CandyStore.PDF
{
     class Document
     {
       #region Enums
       public enum FileTypeExtension
       {
              Unsupported = 0,
              PDF = 1
       }
       #endregion

       #region Variables
       private static readonly byte[] FILE_SIGNATURE_PDF = new byte[] { 0x25,
0x50, 0x44, 0x46 };
       #endregion

       #region Public Static Methods
       public static FileTypeExtension GetFileTypeExtensionFromSignature(ref
byte[] FileContent)
       {
              byte[] temp =
FileContent.Take(FILE_SIGNATURE_PDF.Length).ToArray();
              if(
StructuralComparisons.StructuralEqualityComparer.Equals(FILE_SIGNATURE_PDF,
FileContent.Take(FILE_SIGNATURE_PDF.Length).ToArray()) )
              {
                     return FileTypeExtension.PDF;
              }
              return FileTypeExtension.Unsupported;
       }
       #endregion
     }
}
```

APPENDIX W OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\APPHANDLER\CANDYSTORE\PDF\HPDF.CS (PEP.APPHANDLER.CANDYSTORE.PDF.HPDF CLASS)

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.IO.MemoryMappedFiles;
using System.Linq;
using System.Security.AccessControl;
using System.Text;
using System.Threading.Tasks;

namespace pep.AppHandler.CandyStore.PDF
{
    public class HPDF : IDisposable
     {
      #region Variables
      private bool _bDisposed;
      private FileStream _fileStream;
      private string _filePath;

      private FileSystemRights _fileSystemRights;
      private FileSecurity _fileSecurity;
      private FileOptions _fileOptions;
      #endregion

      #region Constructors
      public HPDF()
      {

      }

      public HPDF(byte[] FileContent)
      {
             this.PlantFile(FileContent, out this._filePath);
      }
      #endregion

      #region Public Methods

      /// <summary>
      /// Method opens data using default assigned Windows appliction
      /// Data with only NoCoat enabled policy can be opened.
      /// Data with disabled NoCoat policy can be opened only if data type
      /// is supported for secure file management
      /// </summary>
      public void OpenDefault()
      {
             try
             {
                    using(Process processHandle = new Process {StartInfo =
new ProcessStartInfo(this._filePath)})
                    {
                           processHandle.Start();
                           processHandle.WaitForExit();
                    }
             }
             catch (Exception eX)
             {
                    throw new Exception(string.Format("{0}::{1}", new
StackFrame(0, true).GetMethod().Name, eX.Message));
             }
      }
```

```
        /// <summary>
        /// Creates MemoryMappedFile for unwrapped policy data
        /// </summary>
        /// <param name="FileContent">Data file content</param>
        public void PlantFile(byte[] FileContent)
        {
              this.PlantFile(FileContent, out this._filePath);
        }

        /// <summary>
        /// Creates MemoryMappedFile for unwrapped policy data
        /// </summary>
        /// <param name="FileContent">Data file content</param>
        /// <param name="FilePath">Path for newly generated file</param>
        private void PlantFile(byte[] FileContent, out string FilePath)
        {
              Document.FileTypeExtension fileExtension =
Document.GetFileTypeExtensionFromSignature(ref FileContent);
              FilePath = String.Format(@"{0}\{1}.{2}", Path.GetTempPath(),
Guid.NewGuid(), fileExtension);

              this._fileSystemRights = FileSystemRights.Read |
FileSystemRights.CreateFiles;
              this._fileOptions = FileOptions.Encrypted;

              //Define complete access control rights
              //this._fileSecurity.

              this._fileStream = new FileStream(FilePath, FileMode.CreateNew,
this._fileSystemRights, FileShare.None, 8, this._fileOptions,
this._fileSecurity);

              using (       BinaryWriter binWriter = new
BinaryWriter(this._fileStream))
              {
                    binWriter.Write(FileContent);
              }
        }
        #endregion

        #region IDisposable Members
        public void Dispose()
        {
              Dispose(true);
              GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool bDisposing)
        {
              if (!this._bDisposed)
              {
                    if (bDisposing)
                    {
                          this._fileStream.Dispose();
                          File.Delete(this._filePath);
                    }

                    this._bDisposed = true;
              }
        }
        #endregion
      }
}
```

APPENDIX X OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\APPHANDLER\CANDYSTORE\OOXML\DOCUMENT.CS (PEP.APPHANDLER.CANDYSTORE.OOXML.XDOCUMENT CLASS)

```
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Wordprocessing;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;

using pep.AppHandler.CandyDelivery;

namespace pep.AppHandler.CandyStore.OOXML
{
     /// <summary>
     /// Class handing low level OOXML document operations
     /// </summary>
     public class XDocument : IDisposable
     {
       #region Variables
       private bool disposed = false;
       private WordprocessingDocument data;
       private SingleStream singleStream;
       private byte[] policy;
       #endregion

       #region Properties
       /// <summary>
       /// OOXML document property
       /// </summary>
       public WordprocessingDocument Data
       {
              get
              {
                     return this.data;
              }
              set
              {
                     this.data = value;
              }
       }
       #endregion

       #region Constructors
       /// <summary>
       /// Creates an instance of a single OOXML document
       /// </summary>
       /// <param name="singleStream">Document file stream</param>
       public XDocument(SingleStream singleStream)
       {
              this.singleStream = singleStream;
              this.WrapDocument(FileAccess.Read);
       }

       /// <summary>
       /// Creates an instance of a single OOXML document supporting basic
access control functionality
       /// </summary>
       /// <param name="singleStream">Document file stream</param>
       /// <param name="fileAccess">Access type</param>
       public XDocument(SingleStream singleStream, FileAccess fileAccess)
       {
```

```
                this.singleStream = singleStream;
                this.WrapDocument(fileAccess);
        }
        #endregion

        #region Public Methods
        /// <summary>
        /// Returns single OOXML document
        /// </summary>
        /// <returns>Single OOXML document</returns>
        public WordprocessingDocument GetDocument()
        {
                return this.data;
        }

        public void AddPolicy(XmlDocument xmlDocument)
        {
                CustomFilePropertiesPart customProperty;
                if (this.data.CustomFilePropertiesPart == null)
                {
                        customProperty = this.data.AddCustomFilePropertiesPart();
                }
                else
                {
                        customProperty = this.data.CustomFilePropertiesPart;
                }
                xmlDocument.Save(customProperty.GetStream());
        }
        #endregion

        #region Private Methods
        /// <summary>
        /// Method wraps document stream into OOXML document
        /// </summary>
        private void WrapDocument(FileAccess fileAccess)
        {
                OpenSettings settings = new OpenSettings();
                settings.AutoSave = false;
                this.data = WordprocessingDocument.Open(singleStream.Stream,
(fileAccess == FileAccess.ReadWrite), settings);
        }

        /// <summary>
        /// Method tries openning OOXML document from the stream
        /// </summary>
        /// <param name="singleStream">Document stream</param>
        /// <param name="document">Opened out OOXML document</param>
        /// <returns></returns>
        private bool TryOpenFromStream(SingleStream singleStream, out
WordprocessingDocument document)
        {
                OpenSettings settings = new OpenSettings();
                settings.AutoSave = false;
                try
                {
                        document =
WordprocessingDocument.Open(singleStream.Stream, false, settings);
                }
                catch
                {
                        document = null;
                        return false;
                }
                return true;
        }
        #endregion

        #region IDisposable Members
        public void Dispose()
        {
                Dispose(true);
                GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool disposing)
        {
                if (!this.disposed)
                {
                        if (disposing)
                        {
                                if (this.data != null)
                                        this.data.Close();
                                        this.data.Dispose();
                                if (this.singleStream != null)
                                {
                                        singleStream.Dispose();
                                }
                        }

                        this.disposed = true;
                }
        }
        #endregion
      }
}
```

APPENDIX Y OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\APPHANDLER\CANDYSTORE\OOXML\MASTER.CS (PEP.APPHANDLER.CANDYSTORE.OOXML.XMASTER CLASS)

```
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Wordprocessing;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using pep.AppHandler.CandyDelivery;

namespace pep.AppHandler.CandyStore.OOXML
{
     /// <summary>
     /// Class responsible for handling OOXML master document generation
     /// </summary>
     public class XMaster : IDisposable
     {
       #region Variables
       private bool disposed = false;
       private string name;
       private WordprocessingDocument data;
         private MemoryStream memoryStream;
       #endregion

       #region Constructors
       public XMaster()
       {
               this.name = Guid.NewGuid().ToString();
              this.memoryStream = new MemoryStream();
               this.data = WordprocessingDocument.Create(this.memoryStream,
DocumentFormat.OpenXml.WordprocessingDocumentType.Document, true);
       }
       #endregion

       #region Public Methods
       /// <summary>
       /// Builds interpreted XACML master document wrapper
       /// </summary>
       /// <param name="documents">Documents file streams</param>
       public void Wrap(List<SingleStream> documents)
       {
               string altChunkId = "AltChunkId1";
              MainDocumentPart mainPart = this.data.AddMainDocumentPart();
               AlternativeFormatImportPart chunk =
mainPart.AddAlternativeFormatImportPart(
                       AlternativeFormatImportPartType.WordprocessingML,
altChunkId);
               foreach(SingleStream singleStream in documents)
               {
                  OpenSettings settings = new OpenSettings();
                  settings.AutoSave = false;
                  WordprocessingDocument doc =
WordprocessingDocument.Open(singleStream.Stream, false, settings);

                  //FileMode.Open
                       chunk.FeedData((Stream)singleStream.Stream);
                       AltChunk altChunk = new AltChunk();
                       altChunk.Id = altChunkId;
                       mainPart.Document
                               .Body
```

```
                          .InsertAfter(altChunk,
mainPart.Document.Body.Elements<Paragraph>().Last());
             }
             mainPart.Document.Save();
      }
      public void PlantFile() { }
      //public
      #endregion

      #region IDisposable Members
      public void Dispose()
      {
             Dispose(true);
             GC.SuppressFinalize(this);
      }

      protected virtual void Dispose(bool disposing)
      {
             if (!this.disposed)
             {
                    if (disposing)
                    {
                           if (this.data != null)
                                  this.data.Dispose();
                    }

                    this.disposed = true;
             }
      }
      #endregion
     }
}
```

APPENDIX Z OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\APPHANDLER\ CANDYDELIVERY\FILESTREAMER.CS (PEP.APPHANDLER.CANDYDELIVERY.FILESTREAMER CLASS)

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace pep.AppHandler.CandyDelivery
{
     public class FileStreamer : IDisposable
     {
       #region Variables
       private bool disposed = false;
       private string sqlConnectionString;
       private SqlConnection sqlConnection;
       #endregion

       #region Properties
       public SqlConnection SqlConnection
       {
              get
              {
                     return this.sqlConnection;
              }
       }
       #endregion

       #region Constructors
       public FileStreamer(string sqlConnectionString)
       {
              this.sqlConnectionString = sqlConnectionString;
              this.sqlConnection = new SqlConnection(sqlConnectionString);
       }
       #endregion

       #region IDisposable Members
       public void Dispose()
       {
              Dispose(true);
              GC.SuppressFinalize(this);
       }

       protected virtual void Dispose(bool disposing)
       {
              if (!this.disposed)
              {
                     if (disposing)
                     {
                            if (this.sqlConnection != null)
                                   this.sqlConnection.Dispose();
                     }

                     this.disposed = true;
              }
       }

       #endregion
     }
}
```

APPENDIX AA OOXML STICKY POLICY HANDLER EVALUATION – ISTICKY\APPHANDLER\ CANDYDELIVERY\SINGLESTREAM.CS (PEP.APPHANDLER.CANDYDELIVERY.SINGLESTREAM CLASS)

```
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Wordprocessing;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.IO;
using System.Linq;
using System.Security;
using System.Security.AccessControl;
using System.Security.Permissions;
using System.Security.Principal;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Transactions;

namespace pep.AppHandler.CandyDelivery
{
     /// <summary>
     /// Class handing SqlFileStreams for documents
     /// </summary>
     public class SingleStream : IDisposable
    {
        #region Constants
        private const int packetSize = 10 * 1000 * 1024; //[MB]
        #endregion

        #region Variables
        private bool disposed = false;
       private string metaDataID;
       private FileStreamer fileStreamer;
       private Stream stream;
       SqlTransaction sqlTransaction;
       private FileAccess fileAccess;
       private byte[] streamHandle;
       #endregion

       #region Properties
       public Stream Stream
       {
              get
              {
                     return this.stream;
              }
       }
       #endregion

       #region Constructors

       public SingleStream(FileStreamer fileStreamer, Stream stream)
       {
              this.fileStreamer = fileStreamer;
              this.fileAccess = FileAccess.ReadWrite;
              this.UploadData(stream);
              this.stream = GetData();
       }

       public SingleStream(FileStreamer fileStreamer, string metaDataID)
       {
              this.fileAccess = FileAccess.ReadWrite;
              this.fileStreamer = fileStreamer;
```

```
             this.metaDataID = metaDataID;
             this.stream = GetData();
       }
       #endregion

       #region Public Methods

       #endregion

       #region Private Methods
       public void UploadData(Stream stream)
       {
             const string sqlTransactionQuery = @"
INSERT INTO
     MetaData
     (MetaDataFile)
OUTPUT
     INSERTED.MetaDataFile.PathName(),
     INSERTED.MetaDataID,
     GET_FILESTREAM_TRANSACTION_CONTEXT()
VALUES (0x)";
             try
             {
                    if (this.fileStreamer.SqlConnection.State ==
System.Data.ConnectionState.Closed)
                    {
                          this.fileStreamer.SqlConnection.Open();
                    }
                    using (SqlCommand sqlCommand = new
SqlCommand(sqlTransactionQuery, this.fileStreamer.SqlConnection))
                    {
                          this.sqlTransaction =
this.fileStreamer.SqlConnection.BeginTransaction();
                          sqlCommand.Transaction = this.sqlTransaction;
                          sqlCommand.CommandText = sqlTransactionQuery;

                          string sqlStreamFullPath;
                          byte[] data;

                          using (SqlDataReader sqlReader =
sqlCommand.ExecuteReader())
                          {
                                 sqlReader.Read();
                                 sqlStreamFullPath = sqlReader.GetString(0);
                                 this.metaDataID =
sqlReader.GetGuid(1).ToString();
                                 data = sqlReader.GetSqlBytes(2).Buffer;
                          }

                          using (SqlFileStream sqlFileStream = new
SqlFileStream(sqlStreamFullPath, data, FileAccess.Write))
                          {
                                 stream.CopyTo(sqlFileStream);
                          }

                          sqlCommand.Transaction.Commit();
                    }
             }
             catch
             {
             }
       }

       private SqlFileStream GetData()
       {
             const string SQL_TRANS_QUERY = @"SELECT
GET_FILESTREAM_TRANSACTION_CONTEXT()";
            //byte[] buffer;
            //UInt32 position = 0;
```

```
			string sqlQuery = String.Format(@"
SELECT TOP 1
	[MetaDataFile].PathName()
FROM
	[NEHST].[dbo].[MetaData]
WHERE
	[MetaDataID] = '{0}'", this.metaDataID);

			if( this.fileStreamer.SqlConnection.State ==
System.Data.ConnectionState.Closed)
			{
				this.fileStreamer.SqlConnection.Open();
			}
			using (SqlCommand sqlCommand = new SqlCommand(sqlQuery,
this.fileStreamer.SqlConnection))
			{
				//using (SqlTransaction sqlTransaction
				this.sqlTransaction =
this.fileStreamer.SqlConnection.BeginTransaction(this.metaDataID.Replace("-",
String.Empty));
				sqlCommand.Transaction = this.sqlTransaction;

				string filePath = (string)sqlCommand.ExecuteScalar();
				//SetRemoteSecurityContext(filePath);

				sqlCommand.CommandText = SQL_TRANS_QUERY;

				this.streamHandle = (byte[])sqlCommand.ExecuteScalar();
				return new SqlFileStream(filePath, this.streamHandle,
this.fileAccess);
			}
		}
		#endregion

		private void SetRemoteSecurityContext(string filePath)
		{
			string securityContext = Thread.CurrentPrincipal.Identity.Name;
			FileSystemAccessRule rule = new
FileSystemAccessRule(securityContext, FileSystemRights.Write,
AccessControlType.Allow);

			PermissionSet permissionSet = new
PermissionSet(PermissionState.Unrestricted);
			permissionSet.AddPermission(new
FileIOPermission(FileIOPermissionAccess.Read, new string[] { filePath }));
			permissionSet.AddPermission(new
FileIOPermission(FileIOPermissionAccess.Write |
FileIOPermissionAccess.PathDiscovery, new string[] { filePath }));
			permissionSet.Assert();

			DirectoryInfo dirInfo = new
DirectoryInfo(Path.GetDirectoryName(filePath));


			bool what = false;
			DirectorySecurity security = dirInfo.GetAccessControl();

			security.ModifyAccessRule(AccessControlModification.Add, rule,
out what);
			dirInfo.SetAccessControl(security);
		}
		#region IDisposable Members
		public void Dispose()
		{
			Dispose(true);
			GC.SuppressFinalize(this);
		}
```

```
            protected virtual void Dispose(bool disposing)
            {
                if (!this.disposed)
                {
                    if (disposing)
                    {
                        if (this.stream != null)
                            this.stream.Dispose();
                        if (this.sqlTransaction != null)
                            this.sqlTransaction.Dispose();
                    }

                    this.disposed = true;
                }
            }

            #endregion
        }
    }
```

APPENDIX BB OOXML STICKY POLICY HANDLER EVALUATION – UNITTEST\UNITTEST.CSPROJ (VISUAL STUDIO PROJECT)

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="12.0" DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == ''
">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{8E832E59-61E2-4C07-B80B-72182C913427}</ProjectGuid>
    <OutputType>Library</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>UnitTest</RootNamespace>
    <AssemblyName>UnitTest</AssemblyName>
    <TargetFrameworkVersion>v4.5</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>
    <ProjectTypeGuids>{3AC096D0-A1C2-E12C-1390-A8335801FDAB};{FAE04EC0-301F-
11D3-BF4B-00C04F79EFBC}</ProjectTypeGuids>
    <VisualStudioVersion Condition="'$(VisualStudioVersion)' ==
''">10.0</VisualStudioVersion>
    <VSToolsPath Condition="'$(VSToolsPath)' ==
''">$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v$(VisualStudioVersion)<
/VSToolsPath>
    <ReferencePath>$(ProgramFiles)\Common Files\microsoft
shared\VSTT\$(VisualStudioVersion)\UITestExtensionPackages</ReferencePath>
    <IsCodedUITest>False</IsCodedUITest>
    <TestProjectType>UnitTest</TestProjectType>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU'
">
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' ==
'Release|AnyCPU' ">
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="DocumentFormat.OpenXml, Version=2.5.5631.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL"
/>
    <Reference Include="System" />
    <Reference Include="System.XML" />
    <Reference Include="System.Xml.Linq" />
  </ItemGroup>
  <Choose>
    <When Condition="('$(VisualStudioVersion)' == '10.0' or
'$(VisualStudioVersion)' == '') and '$(TargetFrameworkVersion)' == 'v3.5'">
      <ItemGroup>
        <Reference
Include="Microsoft.VisualStudio.QualityTools.UnitTestFramework,
Version=10.1.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a,
processorArchitecture=MSIL" />
      </ItemGroup>
    </When>
    <Otherwise>
      <ItemGroup>
```

```
        <Reference
Include="Microsoft.VisualStudio.QualityTools.UnitTestFramework" />
      </ItemGroup>
    </Otherwise>
  </Choose>
  <ItemGroup>
    <Compile Include="UnitTest1.cs" />
    <Compile Include="Properties\AssemblyInfo.cs" />
  </ItemGroup>
  <ItemGroup>
    <ProjectReference Include="..\AppHandler\AppHandler.csproj">
      <Project>{7f9eeeb1-3698-4ab5-be66-8b82f7161a8a}</Project>
      <Name>AppHandler</Name>
    </ProjectReference>
    <ProjectReference
Include="..\ConsoleApplication\ConsoleApplication.csproj">
      <Project>{ec1fd466-eb9a-46ec-8b0c-af7df9ac866e}</Project>
      <Name>ConsoleApplication</Name>
    </ProjectReference>
  </ItemGroup>
  <Choose>
    <When Condition="'$(VisualStudioVersion)' == '10.0' And '$(IsCodedUITest)'
== 'True'">
      <ItemGroup>
        <Reference
Include="Microsoft.VisualStudio.QualityTools.CodedUITestFramework,
Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a,
processorArchitecture=MSIL">
          <Private>False</Private>
        </Reference>
        <Reference Include="Microsoft.VisualStudio.TestTools.UITest.Common,
Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a,
processorArchitecture=MSIL">
          <Private>False</Private>
        </Reference>
        <Reference Include="Microsoft.VisualStudio.TestTools.UITest.Extension,
Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a,
processorArchitecture=MSIL">
          <Private>False</Private>
        </Reference>
        <Reference Include="Microsoft.VisualStudio.TestTools.UITesting,
Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a,
processorArchitecture=MSIL">
          <Private>False</Private>
        </Reference>
      </ItemGroup>
    </When>
  </Choose>
  <Import Project="$(VSToolsPath)\TeamTest\Microsoft.TestTools.targets"
Condition="Exists('$(VSToolsPath)\TeamTest\Microsoft.TestTools.targets')" />
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
  <!-- To modify your build process, add your task inside one of the targets
below and uncomment it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```

APPENDIX CC OOXML Sticky Policy Handler evaluation – UnitTest\UnitTest1.cs (Test Class)

```
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Wordprocessing;
using System;
using System.Collections.Generic;
using System.IO;
using System.Xml;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using pep.AppHandler.CandyStore.OOXML;
using pep.AppHandler.CandyDelivery;

namespace UnitTest
{
     [TestClass]
     public class UnitTest1
     {
       [TestMethod]
       public void TestMethod1()
       {

               string connectionString = @"Persist Security
Info=False;Integrated Security=SSPI;database=NEHST;server=WIN-
TK7VVQF2IT3;Connect Timeout=120";
               //string connectionString = @"Persist Security
Info=False;Trusted_Connection=False;database=NEHST;server=WIN-
TK7VVQF2IT3;Connect Timeout=120;User ID=developer;password=Secure123";


               using( SingleStream singleStream = new SingleStream (new
FileStreamer(connectionString), @"FE8F3116-99A4-E411-9403-000C29E5BED1" ) )
               {
                       using (XDocument candy = new XDocument(singleStream,
FileAccess.ReadWrite))
                       {
                               XmlDocument xmlDocument = new XmlDocument();

     xmlDocument.Load(@"Z:\Projects\Hamster2.5\policy.xml");
                               candy.AddPolicy(xmlDocument);
                       }
               }
       }

       [TestMethod]
       public void TestMethod2()
       {
               string connectionString = @"Persist Security
Info=False;Integrated Security=SSPI;database=NEHST;server=WIN-
TK7VVQF2IT3;Connect Timeout=120";
               //string connectionString = @"Persist Security
Info=False;Trusted_Connection=False;database=NEHST;server=WIN-
TK7VVQF2IT3;Connect Timeout=120;User ID=developer;password=Secure123";

               using (FileStream fileStream = new
FileStream(@"Z:\Projects\Hamster2.5\Hamster2_Ewa.docx", FileMode.Open))
               {
                       using (SingleStream singleStream = new SingleStream(new
FileStreamer(connectionString),  (Stream)fileStream))
                       {
                               using (XDocument candy = new
XDocument(singleStream, FileAccess.ReadWrite))
                               {
                                       XmlDocument xmlDocument = new XmlDocument();

     xmlDocument.Load(@"Z:\Projects\Hamster2.5\policy.xml");
                                       candy.AddPolicy(xmlDocument);
```

```
                                        }
                                    }
                                }
                            }
                        }
                    }
```

APPENDIX DD OOXML STICKY POLICY HANDLER EVALUATION – WEB\DOWNLOADER.ASHX (DOWNLOADER CLASS)

```
<%@ WebHandler Language="C#" Class="downloader" %>

using System;
using System.Web;
using System.IO;
using System.Text;
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Wordprocessing;
using DocumentFormat.OpenXml;

public class downloader : IHttpHandler {

    public void ProcessRequest (HttpContext context)
    {
        try
        {
            string fileName = Guid.NewGuid().ToString() + ".docx";
            string path = context.Server.MapPath("~/docs/") + fileName;
            if (!(string.IsNullOrEmpty(context.Request["what"])) &&
(context.Request["what"].ToLower() == "saveasword")
                && !(string.IsNullOrEmpty(context.Request.Form[0])))
            {
                SaveAsWord(context.Request.Form[0], path, fileName);
            }

        }
        catch (Exception ex)
        {
            context.Response.ContentType = "text/plain";
            context.Response.Write(ex.ToString());
            System.Diagnostics.Trace.WriteLine(ex.ToString());
        }

    }

    public bool IsReusable {
        get {
            return false;
        }
    }


    /// <summary>
    /// Wrapper function to save the richtext values as word
    /// </summary>
    /// <param name="input">rich text</param>
    private void SaveAsWord(string input, string fullFilePath, string
fileNameOnly)
    {
        CreateDocument(fullFilePath);
        input = ReplaceMailMerge(input);
        generateWordDocument(input, fullFilePath, fileNameOnly);

    }


    /// <summary>
    /// Replace the merge field with some custom text
    /// In real world this may come from database
    /// </summary>
    /// <param name="input">input text</param>
    /// <returns>Mail merged text</returns>
    private string ReplaceMailMerge(string input)
    {
        input = input.Replace("{^FirstName^}", "Billy");
        input = input.Replace("{^LastName^}", "Bob");
        input = input.Replace("{^Title^}", "Dr.");
        return input;
    }

    /// <summary>
    /// Create a new docx
    /// </summary>
    /// <param name="path">Path where the doc file is to be created</param>
    private void CreateDocument(string path)
    {

        // Create a Wordprocessing document.
        using (WordprocessingDocument myDoc =
WordprocessingDocument.Create(path, WordprocessingDocumentType.Document))
        {
            // Add a new main document part.
            MainDocumentPart mainPart = myDoc.AddMainDocumentPart();
            //Create DOM tree for simple document.
            mainPart.Document = new Document();
            Body body = new Body();
            Paragraph p = new Paragraph();
            Run r = new Run();
            Text t = new Text("");
            //Append elements appropriately.
            r.Append(t);
            p.Append(r);
            body.Append(p);
            mainPart.Document.Append(body);
            // Save changes to the main document part.
            mainPart.Document.Save();
        }
    }

    /// <summary>
    /// Add the HTML markup in the file
    /// Save the document
    /// Send the path of this document to the caller
    /// </summary>
    /// <param name="htmlMarkup">Rich text</param>
    /// <param name="fullFilePath">Path of the file</param>
    /// <param name="fileNameOnly">Name of the file</param>
    public void generateWordDocument(string htmlMarkup, string fullFilePath,
string fileNameOnly)
    {
        try
        {
            /*----------- Generate the Document -----------------------*/
            //put some title
            string pageTitle = Guid.NewGuid().ToString();
            //open the document
            using (WordprocessingDocument wordDoc =
WordprocessingDocument.Open(fullFilePath, true))
            {
                //get the document
                MainDocumentPart mainPart = wordDoc.MainDocumentPart;
                int altChunkIdCounter = 1;
                int blockLevelCounter = 1;

                string mainhtml = "<html><head><style
type='text/css'>.catalogGeneralTable{border-collapse: collapse;text-align:
left;} .catalogGeneralTable td, th{ padding: 5px; border: 1px solid #999999;
}</style></head><body style='font-family:Trebuchet MS;font-size:.9em;'>" +
htmlMarkup + "</body></html>";
                string altChunkId = String.Format("AltChunkId{0}",
altChunkIdCounter++);
```

```
                //Import data as html content using Altchunk
                AlternativeFormatImportPart chunk =
mainPart.AddAlternativeFormatImportPart(AlternativeFormatImportPartType.Html,
altChunkId);

                //add the chunk to the doc
                using (Stream chunkStream = chunk.GetStream(FileMode.Create,
FileAccess.Write))
                {
                    using (StreamWriter stringWriter = new
StreamWriter(chunkStream, Encoding.UTF8)) //Encoding.UTF8 is important to
remove special characters
                    {
                        stringWriter.Write(mainhtml);
                    }
                }

                AltChunk altChunk = new AltChunk();
                altChunk.Id = altChunkId;
                //insert the text in the doc
                mainPart.Document.Body.InsertAt(altChunk,
blockLevelCounter++);
                //save the document
                mainPart.Document.Save();
            }
            /*----------- End Generate the Document -----------------------*/

            /* ------- Send the response -----------*/
            //clear the response object
            HttpContext.Current.Response.ClearContent();
            //add the demilited string to the response object and write it.
            string url = HttpContext.Current.Request.ApplicationPath +
"/docs/" + fileNameOnly;
            HttpContext.Current.Response.Write(url);
            HttpContext.Current.Response.End();

            /* -------End Send the response -----------*/
        }
        catch (Exception ex)
        {
            HttpContext.Current.Response.Write(ex.Message.ToString());
        }
    }


    /// <summary>
    /// Function to download the newly generated doc file
    /// </summary>
    /// <param name="completeFilePath">File path</param>
    /// <param name="contentType">Content type</param>
    private void DownloadFile(string completeFilePath, string fileNameOnly,
string contentType)
    {
        Stream iStream = null;

        // Buffer to read 10K bytes in chunk:
        byte[] buffer = new Byte[10000];

        // Length of the file:
        int length;

        // Total bytes to read:
        long dataToRead;

        try
        {
            // Open the file.
            iStream = new FileStream(completeFilePath, FileMode.Open,
            FileAccess.Read, FileShare.Read);

            // Total bytes to read:
            dataToRead = iStream.Length;

            HttpContext.Current.Response.ContentType = contentType;
            HttpContext.Current.Response.AddHeader("Content-Disposition",
"attachment; filename=" + fileNameOnly);

            // Read the bytes.
            while (dataToRead > 0)
            {
                // Verify that the client is connected.
                if (HttpContext.Current.Response.IsClientConnected)
                {
                    // Read the data in buffer.
                    length = iStream.Read(buffer, 0, 10000);

                    // Write the data to the current output stream.
                    HttpContext.Current.Response.OutputStream.Write(buffer, 0,
length);

                    // Flush the data to the HTML output.
                    HttpContext.Current.Response.Flush();

                    buffer = new Byte[10000];
                    dataToRead = dataToRead - length;
                }
                else
                {
                    //prevent infinite loop if user disconnects
                    dataToRead = -1;
                }
            }
        }
        catch (Exception ex)
        {
            // Trap the error, if any.
            HttpContext.Current.Response.Write("Error : " + ex.Message);
        }
        finally
        {
            if (iStream != null)
            {
                //Close the file.
                iStream.Close();
            }
            HttpContext.Current.Response.Close();
        }
    }
}
```

APPENDIX EE OOXML STICKY POLICY HANDLER EVALUATION – WEB\DEFAULT.ASPX (DEFAULT PAGE)

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" ValidateRequest="false"  %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Online Editor with Mail Merge</title>
    <script src="scripts/ckeditor/ckeditor.js"
type="text/javascript"></script>
    <script src="scripts/ckeditor/adapters/jquery.js"
type="text/javascript"></script>
    <script type="text/javascript" src="scripts/jquery-1.3.2.min.js"></script>
    <script src="scripts/myAjax.js" type="text/javascript"></script>
    <script language="javascript" type="text/javascript">
        //select the merge fields dropdown and implement the onchange event
        //to insert the selected value in the text area
        $(document).ready(function() {
            $("#MergeFields").val('0');
            $("#MergeFields").change(onSelectChange);
        });

        function onSelectChange() {
            var selected = $("#MergeFields option:selected");
            var oEditor = CKEDITOR.instances.editor1;

            if (selected.val() != 0) {
                var valueToInsert = selected.text();
                // Check the active editing mode.
                if (oEditor.mode == 'wysiwyg') {
                    // Insert the desired HTML.
                    oEditor.insertHtml(valueToInsert);
                }
                else {
                    alert('You must be on WYSIWYG mode!');
                }
            }
            $("#MergeFields").val('0');
        }
    </script>

</head>
<body>
    <form id="form1" runat="server">
    <div>
        <h2>
           Online Richtext editor with Mail Merge
         </h2>
         <!-- This <div> holds alert messages to be display in the sample
page. -->
         <div id="alerts">
           <noscript>
                  <p>
                         <strong>Online Richtext editor requires JavaScript to
run</strong>.
                         In a browser with no JavaScript support, like yours,
you should still see
                         the contents (HTML data) and you should be able to
edit it normally,
                         without a rich editor interface.
                  </p>
           </noscript>
         </div>
```

```
            <p>
                This is online rich text editor sample with mail merge.<br />
                Use the text area below to type in your document. <br />
                Use the dropdowns below that have mail merge fields.
            </p>
             <textarea cols="50" id="editor1" name="editor1" rows="10"></textarea>
            <!-- instantiate a new instance of CKEDITOR -->
        <script type="text/javascript">
            //<![CDATA[
            // Replace the <textarea id="editor1"> with an CKEditor instance.
            var editor = CKEDITOR.replace('editor1',
             {
                 toolbar: 'myToolBar', skin: 'office2003', width: '60%'
             });
            //]]>
        </script>

      <div id="eMessage">
        </div>
        <div id="eButtons">
            <span><b>Insert Merge Fields from here</b></span>
            <select id="MergeFields" >
                <option value="0" selected="selected">Select Merge
Fields</option>
                <option value="1">{^Title^}</option>
                <option value="2">{^FirstName^}</option>
                <option value="3">{^LastName^}</option>
            </select>

            <input type="button" name="saveAsWord" id="saveAsWord"
                 onclick="javascript:ajaxDownloadDoc()" value="Download as
word" />
        </div>

    </div>
    </form>
</body>
</html>
```

APPENDIX FF OOXML STICKY POLICY HANDLER EVALUATION – SPIBEDRV\SPIBEDRV.SLN

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25123.0
MinimumVisualStudioVersion = 12.0
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "spibedrv",
"spibedrv.vcxproj", "{18A7AC9C-B3DA-4935-9A35-56FD35FF2705}"
EndProject
Global
     GlobalSection(SolutionConfigurationPlatforms) = preSolution
       Debug|Win32 = Debug|Win32
       Debug|x64 = Debug|x64
       Release|Win32 = Release|Win32
       Release|x64 = Release|x64
     EndGlobalSection
     GlobalSection(ProjectConfigurationPlatforms) = postSolution
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Debug|Win32.ActiveCfg =
Debug|Win32
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Debug|Win32.Build.0 =
Debug|Win32
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Debug|Win32.Deploy.0 =
Debug|Win32
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Debug|x64.ActiveCfg = Debug|x64
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Debug|x64.Build.0 = Debug|x64
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Debug|x64.Deploy.0 = Debug|x64
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Release|Win32.ActiveCfg =
Release|Win32
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Release|Win32.Build.0 =
Release|Win32
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Release|Win32.Deploy.0 =
Release|Win32
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Release|x64.ActiveCfg =
Release|x64
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Release|x64.Build.0 =
Release|x64
       {18A7AC9C-B3DA-4935-9A35-56FD35FF2705}.Release|x64.Deploy.0 =
Release|x64
     EndGlobalSection
     GlobalSection(SolutionProperties) = preSolution
       HideSolutionNode = FALSE
     EndGlobalSection
EndGlobal
```

APPENDIX GG OOXML STICKY POLICY HANDLER EVALUATION – SPIBEDRV\SPIBEDRV.VCXPROJ

```
<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="12.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{18A7AC9C-B3DA-4935-9A35-56FD35FF2705}</ProjectGuid>
    <RootNamespace>$(MSBuildProjectName)</RootNamespace>
    <Configuration Condition="'$(Configuration)' == ''">Debug</Configuration>
    <Platform Condition="'$(Platform)' == ''">Win32</Platform>
    <SampleGuid>{740DFB1D-45E5-406F-BBB6-4AF0E6C30DBA}</SampleGuid>
    <ProjectName>spibedrv</ProjectName>

<WindowsTargetPlatformVersion>$(LatestTargetPlatformVersion)</WindowsTargetPla
tformVersion>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Label="Configuration"
Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <TargetVersion>Windows10</TargetVersion>
    <UseDebugLibraries>False</UseDebugLibraries>
    <DriverTargetPlatform>Universal</DriverTargetPlatform>
    <DriverType>WDM</DriverType>
    <PlatformToolset>WindowsKernelModeDriver10.0</PlatformToolset>
    <ConfigurationType>Driver</ConfigurationType>
  </PropertyGroup>
  <PropertyGroup Label="Configuration"
Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <TargetVersion>Windows10</TargetVersion>
    <UseDebugLibraries>True</UseDebugLibraries>
    <DriverTargetPlatform>Universal</DriverTargetPlatform>
    <DriverType>WDM</DriverType>
    <PlatformToolset>WindowsKernelModeDriver10.0</PlatformToolset>
    <ConfigurationType>Driver</ConfigurationType>
  </PropertyGroup>
  <PropertyGroup Label="Configuration"
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <TargetVersion>Windows10</TargetVersion>
    <UseDebugLibraries>False</UseDebugLibraries>
    <DriverTargetPlatform>Universal</DriverTargetPlatform>
    <DriverType>WDM</DriverType>
    <PlatformToolset>WindowsKernelModeDriver10.0</PlatformToolset>
    <ConfigurationType>Driver</ConfigurationType>
  </PropertyGroup>
  <PropertyGroup Label="Configuration"
Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <TargetVersion>Windows10</TargetVersion>
    <UseDebugLibraries>True</UseDebugLibraries>
    <DriverTargetPlatform>Universal</DriverTargetPlatform>
    <DriverType>WDM</DriverType>
    <PlatformToolset>WindowsKernelModeDriver10.0</PlatformToolset>
    <ConfigurationType>Driver</ConfigurationType>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <PropertyGroup>
    <OutDir>$(IntDir)</OutDir>
  </PropertyGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')" />
  </ImportGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')" />
  </ImportGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')" />
  </ImportGroup>
  <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')" />
  </ImportGroup>
  <ItemGroup Label="WrappedTaskItems" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <TargetName>spibedrv</TargetName>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <TargetName>spibedrv</TargetName>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <TargetName>spibedrv</TargetName>
    <EnableInf2cat>false</EnableInf2cat>
    <Inf2CatUseLocalTime>true</Inf2CatUseLocalTime>
    <Inf2CatNoCatalog>false</Inf2CatNoCatalog>
    <IntDir>$(ConfigurationName)\</IntDir>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <TargetName>spibedrv</TargetName>
    <EnableInf2cat>false</EnableInf2cat>
  </PropertyGroup>
  <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <ClCompile>
      <TreatWarningAsError>true</TreatWarningAsError>
      <WarningLevel>Level4</WarningLevel>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
      <ExceptionHandling>
      </ExceptionHandling>
    </ClCompile>
    <Link>

<AdditionalDependencies>%(AdditionalDependencies);$(DDK_LIB_PATH)\fltMgr.lib</
AdditionalDependencies>
    </Link>
    <Midl>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
    </Midl>
    <ResourceCompile>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
```

```
POOL_NX_OPTIN=1</PreprocessorDefinitions>
    </ResourceCompile>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <ClCompile>
      <TreatWarningAsError>true</TreatWarningAsError>
      <WarningLevel>Level4</WarningLevel>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
      <ExceptionHandling>
      </ExceptionHandling>
    </ClCompile>
    <Link>

<AdditionalDependencies>%(AdditionalDependencies);$(DDK_LIB_PATH)\fltMgr.lib</
AdditionalDependencies>
    </Link>
    <Midl>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
    </Midl>
    <ResourceCompile>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
    </ResourceCompile>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
    <ClCompile>
      <TreatWarningAsError>true</TreatWarningAsError>
      <WarningLevel>Level4</WarningLevel>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
      <ExceptionHandling>
      </ExceptionHandling>
      <Inf2CatAdditionalOptions>
      </Inf2CatAdditionalOptions>
    </ClCompile>
    <Link>

<AdditionalDependencies>%(AdditionalDependencies);$(DDK_LIB_PATH)\fltMgr.lib</
AdditionalDependencies>
    </Link>
    <Midl>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
    </Midl>
    <ResourceCompile>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
    </ResourceCompile>
    <Inf>
      <DateStamp>04/08/2018</DateStamp>
      <DriverVersionSectionName>1.0.0.0</DriverVersionSectionName>
      <CatalogFileName>spibedrv.cat</CatalogFileName>
    </Inf>
    <PostBuildEvent>
      <Command>"$(WindowsSdkDir)bin\$(DDKPlatform)\inf2cat.exe"
/os:10_$(DDKPlatform)
/driver:"$(ProjectDir)$(IntDir)$(MSBuildProjectName)"</Command>
    </PostBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup
```

```
Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <ClCompile>
      <TreatWarningAsError>true</TreatWarningAsError>
      <WarningLevel>Level4</WarningLevel>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
      <ExceptionHandling>
      </ExceptionHandling>
    </ClCompile>
    <Link>

<AdditionalDependencies>%(AdditionalDependencies);$(DDK_LIB_PATH)\fltMgr.lib</
AdditionalDependencies>
    </Link>
    <Midl>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
    </Midl>
    <ResourceCompile>

<PreprocessorDefinitions>%(PreprocessorDefinitions);_WIN2K_COMPAT_SLIST_USAGE;
POOL_NX_OPTIN=1</PreprocessorDefinitions>
    </ResourceCompile>
  </ItemDefinitionGroup>
  <ItemGroup>
    <ClCompile Include="spibedrv.c" />
    <ClCompile Include="spibelib.c" />
    <ResourceCompile Include="spibedrv.rc" />
  </ItemGroup>
  <ItemGroup>
    <Inf Exclude="@(Inf)" Include="*.inf" />
    <FilesToPackage Include="$(TargetPath)"
Condition="'$(ConfigurationType)'=='Driver' or
'$(ConfigurationType)'=='DynamicLibrary'" />
  </ItemGroup>
  <ItemGroup>
    <None Exclude="@(None)" Include="*.txt;*.htm;*.html" />
    <None Exclude="@(None)"
Include="*.ico;*.cur;*.bmp;*.dlg;*.rct;*.gif;*.jpg;*.jpeg;*.wav;*.jpe;*.tiff;*
.tif;*.png;*.rc2" />
    <None Exclude="@(None)" Include="*.def;*.bat;*.hpj;*.asmx" />
  </ItemGroup>
  <ItemGroup>
    <ClInclude Include="resource.h" />
    <ClInclude Include="spibelib.h" />
  </ItemGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
</Project>
```

APPENDIX HH OOXML STICKY POLICY HANDLER EVALUATION – SPIBEDRV\SPIBELIB.C

```
#include <fltKernel.h>

#define OOXML_EXTENSION_SIZE 4

long XOrBuffer(PUCHAR buffer, const long BUFFER_SIZE)
{
      if (buffer)
      {
        for (long i = 0; i < BUFFER_SIZE; i++)
        {
                buffer[i] ^= 0xFF;
        }

        return BUFFER_SIZE;
      }
      else
      {
        return (long)-1;
      }
}

BOOLEAN IsOOXML(PFLT_CALLBACK_DATA Data)
{
      WCHAR string[OOXML_EXTENSION_SIZE];
      UNICODE_STRING oOxmlExtension;
      oOxmlExtension.Buffer = string;
      oOxmlExtension.MaximumLength = OOXML_EXTENSION_SIZE;
      oOxmlExtension.Length = OOXML_EXTENSION_SIZE;

      RtlInitUnicodeString(&oOxmlExtension, L"docx");

      PFLT_FILE_NAME_INFORMATION FileNameInformation = NULL;
      NTSTATUS status;

      status =
        FltGetFileNameInformation(Data, FLT_FILE_NAME_NORMALIZED,
&FileNameInformation);

      if (NT_SUCCESS(status))
      {
        status = FltParseFileNameInformation(FileNameInformation);

        if (NT_SUCCESS(status))
        {
                //KdPrint(("Parent Dir is %S\n", FileNameInformation-
>ParentDir.Buffer));
                if (RtlCompareUnicodeString(&FileNameInformation->Extension,
&oOxmlExtension, TRUE) == 0)
                {
                        FltReleaseFileNameInformation(FileNameInformation);
                        return TRUE;
                }

                FltReleaseFileNameInformation(FileNameInformation);
        }
        else
        {
                KdPrint(("Error FltParseFileNameInformation"));
        }
      }
      else if (status == STATUS_FLT_INVALID_NAME_REQUEST)
      {
        KdPrint(("Error STATUS_FLT_INVALID_NAME_REQUEST"));
      }
      else if (status == STATUS_INSUFFICIENT_RESOURCES)
      {
        KdPrint(("Error STATUS_INSUFFICIENT_RESOURCES"));
      }
      else if (status == STATUS_INVALID_PARAMETER)
      {
        KdPrint(("Error STATUS_INVALID_PARAMETER"));
      }
      return FALSE;
}
```

APPENDIX II OOXML STICKY POLICY HANDLER EVALUATION – SPIBEDRV\SPIBELIB.C

```
#ifndef SPIBE_LIB_
#define SPIBE_LIB_

long  XOrBuffer(PUCHAR buffer, const long BUFFER_SIZE);
//long  XOrBuffer(unsigned char **buffer, const long BUFFER_SIZE);

#endif //SPIBE_LIB_
```

APPENDIX JJ OOXML STICKY POLICY HANDLER EVALUATION – SPIBEDRV\SPIBEDRV.C

```
#include <fltKernel.h>
#include "spibelib.h"
#include <suppress.h>

/*
Based on Microsoft SwapBuffers template
*/

#pragma prefast(disable:__WARNING_ENCODE_MEMBER_FUNCTION_POINTER, "Not valid
for kernel mode drivers")

PFLT_FILTER gFilterHandle;

#define BUFFER_SWAP_TAG     'bdBS'
#define CONTEXT_TAG         'xcBS'
#define NAME_TAG            'mnBS'
#define PRE_2_POST_TAG      'ppBS'

typedef struct _VOLUME_CONTEXT
{
     UNICODE_STRING Name;

     ULONG SectorSize;

} VOLUME_CONTEXT, *PVOLUME_CONTEXT;

#define MIN_SECTOR_SIZE 0x200

typedef struct _PRE_2_POST_CONTEXT
{
     PVOLUME_CONTEXT VolCtx;

     PVOID UnpackedCandy;

} PRE_2_POST_CONTEXT, *PPRE_2_POST_CONTEXT;

NPAGED_LOOKASIDE_LIST Pre2PostContextList;

NTSTATUS
InstanceSetup(
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _In_ FLT_INSTANCE_SETUP_FLAGS Flags,
     _In_ DEVICE_TYPE VolumeDeviceType,
     _In_ FLT_FILESYSTEM_TYPE VolumeFilesystemType
);

VOID
CleanupVolumeContext(
     _In_ PFLT_CONTEXT Context,
     _In_ FLT_CONTEXT_TYPE ContextType
);

NTSTATUS
InstanceQueryTeardown(
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _In_ FLT_INSTANCE_QUERY_TEARDOWN_FLAGS Flags
);

DRIVER_INITIALIZE DriverEntry;
NTSTATUS
DriverEntry(
     _In_ PDRIVER_OBJECT DriverObject,
     _In_ PUNICODE_STRING RegistryPath
);

NTSTATUS
FilterUnload(
     _In_ FLT_FILTER_UNLOAD_FLAGS Flags
);

FLT_PREOP_CALLBACK_STATUS
SwapPreReadBuffers(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _Flt_CompletionContext_Outptr_ PVOID *CompletionContext
);

FLT_POSTOP_CALLBACK_STATUS
SwapPostReadBuffers(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _In_ PVOID CompletionContext,
     _In_ FLT_POST_OPERATION_FLAGS Flags
);

FLT_POSTOP_CALLBACK_STATUS
SwapPostReadBuffersWhenSafe(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _In_ PVOID CompletionContext,
     _In_ FLT_POST_OPERATION_FLAGS Flags
);

FLT_PREOP_CALLBACK_STATUS
SwapPreDirCtrlBuffers(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _Flt_CompletionContext_Outptr_ PVOID *CompletionContext
);

FLT_POSTOP_CALLBACK_STATUS
SwapPostDirCtrlBuffers(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _In_ PVOID CompletionContext,
     _In_ FLT_POST_OPERATION_FLAGS Flags
);

FLT_POSTOP_CALLBACK_STATUS
SwapPostDirCtrlBuffersWhenSafe(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _In_ PVOID CompletionContext,
     _In_ FLT_POST_OPERATION_FLAGS Flags
);

FLT_PREOP_CALLBACK_STATUS
SwapPreWriteBuffers(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _Flt_CompletionContext_Outptr_ PVOID *CompletionContext
);

FLT_POSTOP_CALLBACK_STATUS
SwapPostWriteBuffers(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _In_ PVOID CompletionContext,
     _In_ FLT_POST_OPERATION_FLAGS Flags
);
```

```
VOID
ReadDriverParameters(
     _In_ PUNICODE_STRING RegistryPath
);

#ifdef ALLOC_PRAGMA
#pragma alloc_text(PAGE, InstanceSetup)
#pragma alloc_text(PAGE, CleanupVolumeContext)
#pragma alloc_text(PAGE, InstanceQueryTeardown)
#pragma alloc_text(INIT, DriverEntry)
#pragma alloc_text(INIT, ReadDriverParameters)
#pragma alloc_text(PAGE, FilterUnload)
#endif

CONST FLT_OPERATION_REGISTRATION Callbacks[] =
{

{ IRP_MJ_READ,
  0,
  SwapPreReadBuffers,
  SwapPostReadBuffers },


{ IRP_MJ_WRITE,
  0,
  SwapPreWriteBuffers,
  SwapPostWriteBuffers },


{ IRP_MJ_DIRECTORY_CONTROL,
  0,
  SwapPreDirCtrlBuffers,
  SwapPostDirCtrlBuffers },


{ IRP_MJ_OPERATION_END }
};


CONST FLT_CONTEXT_REGISTRATION ContextNotifications[] =
{


{ FLT_VOLUME_CONTEXT,
   0,
   CleanupVolumeContext,
   sizeof(VOLUME_CONTEXT),
   CONTEXT_TAG },

{ FLT_CONTEXT_END }
};


CONST FLT_REGISTRATION FilterRegistration =
{
sizeof(FLT_REGISTRATION),         //  Size
FLT_REGISTRATION_VERSION,           //  Version
0,                                  //  Flags

ContextNotifications,               //  Context
Callbacks,                          //  Operation callbacks

FilterUnload,                       //  MiniFilterUnload

InstanceSetup,                      //  InstanceSetup
InstanceQueryTeardown,              //  InstanceQueryTeardown
```

```
NULL,                               //  InstanceTeardownStart
NULL,                               //  InstanceTeardownComplete

NULL,                               //  GenerateFileName
NULL,                               //  GenerateDestinationFileName
NULL                                //  NormalizeNameComponent

};

#define LOGFL_ERRORS    0x00000001  // if set, display error messages
#define LOGFL_READ      0x00000002  // if set, display READ operation info
#define LOGFL_WRITE     0x00000004  // if set, display WRITE operation info
#define LOGFL_DIRCTRL   0x00000008  // if set, display DIRCTRL operation info
#define LOGFL_VOLCTX    0x00000010  // if set, display VOLCTX operation info

ULONG LoggingFlags = 0;             // all disabled by default

#define LOG_PRINT( _logFlag, _string )                              \
    (FlagOn(LoggingFlags,(_logFlag)) ?                              \
        DbgPrint _string  :                                         \
        ((int)0))

NTSTATUS
InstanceSetup(
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _In_ FLT_INSTANCE_SETUP_FLAGS Flags,
     _In_ DEVICE_TYPE VolumeDeviceType,
     _In_ FLT_FILESYSTEM_TYPE VolumeFilesystemType
)
{
     PDEVICE_OBJECT devObj = NULL;
     PVOLUME_CONTEXT ctx = NULL;
     NTSTATUS status = STATUS_SUCCESS;
     ULONG retLen;
     PUNICODE_STRING workingName;
     USHORT size;
     UCHAR volPropBuffer[sizeof(FLT_VOLUME_PROPERTIES) + 512];
     PFLT_VOLUME_PROPERTIES volProp = (PFLT_VOLUME_PROPERTIES)volPropBuffer;

     PAGED_CODE();

     UNREFERENCED_PARAMETER(Flags);
     UNREFERENCED_PARAMETER(VolumeDeviceType);
     UNREFERENCED_PARAMETER(VolumeFilesystemType);

     try
     {
       status = FltAllocateContext(FltObjects->Filter,
              FLT_VOLUME_CONTEXT,
              sizeof(VOLUME_CONTEXT),
              NonPagedPool,
              &ctx);

       if (!NT_SUCCESS(status))
       {
              leave;
       }

       status = FltGetVolumeProperties(FltObjects->Volume,
              volProp,
              sizeof(volPropBuffer),
              &retLen);

       if (!NT_SUCCESS(status))
       {
              leave;
       }
```

```
        FLT_ASSERT((volProp->SectorSize == 0) || (volProp->SectorSize >=
MIN_SECTOR_SIZE));

        ctx->SectorSize = max(volProp->SectorSize, MIN_SECTOR_SIZE);

        ctx->Name.Buffer = NULL;

        status = FltGetDiskDeviceObject(FltObjects->Volume, &devObj);

        if (NT_SUCCESS(status))
        {
                status = IoVolumeDeviceToDosName(devObj, &ctx->Name);
        }

        if (!NT_SUCCESS(status))
        {
                FLT_ASSERT(ctx->Name.Buffer == NULL);

                if (volProp->RealDeviceName.Length > 0)
                {
                        workingName = &volProp->RealDeviceName;
                }
                else if (volProp->FileSystemDeviceName.Length > 0)
                {
                        workingName = &volProp->FileSystemDeviceName;
                }
                else
                {
                        status = STATUS_FLT_DO_NOT_ATTACH;
                        leave;
                }

                size = workingName->Length + sizeof(WCHAR);

#pragma prefast(suppress:__WARNING_MEMORY_LEAK, "ctx->Name.Buffer will not be
leaked because it is freed in CleanupVolumeContext")
                ctx->Name.Buffer = ExAllocatePoolWithTag(NonPagedPool,
                        size,
                        NAME_TAG);
                if (ctx->Name.Buffer == NULL)
                {
                        status = STATUS_INSUFFICIENT_RESOURCES;
                        leave;
                }

                ctx->Name.Length = 0;
                ctx->Name.MaximumLength = size;

                RtlCopyUnicodeString(&ctx->Name,
                        workingName);

                RtlAppendUnicodeToString(&ctx->Name,
                        L":");
        }

        status = FltSetVolumeContext(FltObjects->Volume,
                FLT_SET_CONTEXT_KEEP_IF_EXISTS,
                ctx,
                NULL);


        LOG_PRINT(LOGFL_VOLCTX,
                ("spibedrv!InstanceSetup:                Real SectSize=0x%04x,
Used SectSize=0x%04x, Name=\"%wZ\"\n",
                        volProp->SectorSize,
                        ctx->SectorSize,
                        &ctx->Name));
        if (status == STATUS_FLT_CONTEXT_ALREADY_DEFINED)
        {
                status = STATUS_SUCCESS;
        }
      }
      finally
      {
        if (ctx)
        {
                FltReleaseContext(ctx);
        }


        if (devObj)
        {
                ObDereferenceObject(devObj);
        }
      }
      return status;
}


VOID
CleanupVolumeContext(
      _In_ PFLT_CONTEXT Context,
      _In_ FLT_CONTEXT_TYPE ContextType
)
{
      PVOLUME_CONTEXT ctx = Context;

      PAGED_CODE();

      UNREFERENCED_PARAMETER(ContextType);

      FLT_ASSERT(ContextType == FLT_VOLUME_CONTEXT);

      if (ctx->Name.Buffer != NULL)
      {
        ExFreePool(ctx->Name.Buffer);
        ctx->Name.Buffer = NULL;
      }
}

NTSTATUS
InstanceQueryTeardown(
      _In_ PCFLT_RELATED_OBJECTS FltObjects,
      _In_ FLT_INSTANCE_QUERY_TEARDOWN_FLAGS Flags
)
{
      PAGED_CODE();

      UNREFERENCED_PARAMETER(FltObjects);
      UNREFERENCED_PARAMETER(Flags);

      return STATUS_SUCCESS;
}

NTSTATUS
DriverEntry(
      _In_ PDRIVER_OBJECT DriverObject,
      _In_ PUNICODE_STRING RegistryPath
)
{
      NTSTATUS status;

      ExInitializeDriverRuntime(DrvRtPoolNxOptIn);

      ReadDriverParameters(RegistryPath);
```

```
     ExInitializeNPagedLookasideList(&Pre2PostContextList,
       NULL,
       NULL,
       0,
       sizeof(PRE_2_POST_CONTEXT),
       PRE_2_POST_TAG,
       0);

     status = FltRegisterFilter(DriverObject,
       &FilterRegistration,
       &gFilterHandle);

     if (!NT_SUCCESS(status))
     {

       goto SwapDriverEntryExit;
     }

     status = FltStartFiltering(gFilterHandle);

     if (!NT_SUCCESS(status))
     {
       FltUnregisterFilter(gFilterHandle);
       goto SwapDriverEntryExit;
     }

SwapDriverEntryExit:

     if (!NT_SUCCESS(status))
     {
       ExDeleteNPagedLookasideList(&Pre2PostContextList);
     }

     return status;
}

NTSTATUS
FilterUnload(
     _In_ FLT_FILTER_UNLOAD_FLAGS Flags
)
{
     PAGED_CODE();

     UNREFERENCED_PARAMETER(Flags);

     FltUnregisterFilter(gFilterHandle);

     ExDeleteNPagedLookasideList(&Pre2PostContextList);

     return STATUS_SUCCESS;
}

FLT_PREOP_CALLBACK_STATUS
SwapPreReadBuffers(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _Flt_CompletionContext_Outptr_ PVOID *CompletionContext
)
{
     PFLT_IO_PARAMETER_BLOCK iopb = Data->Iopb;
     FLT_PREOP_CALLBACK_STATUS retValue = FLT_PREOP_SUCCESS_NO_CALLBACK;
     PVOID newBuf = NULL;
     PMDL newMdl = NULL;
     PVOLUME_CONTEXT volCtx = NULL;
     PPRE_2_POST_CONTEXT p2pCtx;
     NTSTATUS status;
     ULONG readLen = iopb->Parameters.Read.Length;
```

```
    try
    {
      if (readLen == 0)
      {
            leave;
      }

      status = FltGetVolumeContext(FltObjects->Filter,
            FltObjects->Volume,
            &volCtx);

      if (!NT_SUCCESS(status))
      {

            LOG_PRINT(LOGFL_ERRORS,
                  ("spibedrv!SwapPreReadBuffers:             Error getting
volume context, status=%x\n",
                        status));

            leave;
      }

      if (FlagOn(IRP_NOCACHE, iopb->IrpFlags))
      {
            readLen = (ULONG)ROUND_TO_SIZE(readLen, volCtx->SectorSize);
      }

      newBuf = FltAllocatePoolAlignedWithTag(FltObjects->Instance,
            NonPagedPool,
            (SIZE_T)readLen,
            BUFFER_SWAP_TAG);
      if (newBuf == NULL)
      {

            LOG_PRINT(LOGFL_ERRORS,
                  ("spibedrv!SwapPreReadBuffers:             %wZ Failed to
allocate %d bytes of memory\n",
                        &volCtx->Name,
                        readLen));

            leave;
      }

      if (FlagOn(Data->Flags, FLTFL_CALLBACK_DATA_IRP_OPERATION))
      {
            newMdl = IoAllocateMdl(newBuf,
                  readLen,
                  FALSE,
                  FALSE,
                  NULL);

            if (newMdl == NULL)
            {

                  LOG_PRINT(LOGFL_ERRORS,
                        ("spibedrv!SwapPreReadBuffers:             %wZ
Failed to allocate MDL\n",
                              &volCtx->Name));

                  leave;
            }

            MmBuildMdlForNonPagedPool(newMdl);
      }

      p2pCtx = ExAllocateFromNPagedLookasideList(&Pre2PostContextList);
```

```
        if (p2pCtx == NULL)
        {

                LOG_PRINT(LOGFL_ERRORS,
                        ("spibedrv!SwapPreReadBuffers:                %wZ Failed to
allocate pre2Post context structure\n",
                                &volCtx->Name));

                leave;
        }

        LOG_PRINT(LOGFL_READ,
                ("spibedrv!SwapPreReadBuffers:                %wZ newB=%p newMdl=%p
oldB=%p oldMdl=%p len=%d\n",
                        &volCtx->Name,
                        newBuf,
                        newMdl,
                        iopb->Parameters.Read.ReadBuffer,
                        iopb->Parameters.Read.MdlAddress,
                        readLen));

        iopb->Parameters.Read.ReadBuffer = newBuf;
        iopb->Parameters.Read.MdlAddress = newMdl;
        FltSetCallbackDataDirty(Data);

        p2pCtx->UnpackedCandy = newBuf;
        p2pCtx->VolCtx = volCtx;

        *CompletionContext = p2pCtx;

        retValue = FLT_PREOP_SUCCESS_WITH_CALLBACK;
      }
      finally
      {
        if (retValue != FLT_PREOP_SUCCESS_WITH_CALLBACK)
        {

                if (newBuf != NULL)
                {

                        FltFreePoolAlignedWithTag(FltObjects->Instance,
                                newBuf,
                                BUFFER_SWAP_TAG);
                }

                if (newMdl != NULL)
                {

                        IoFreeMdl(newMdl);
                }

                if (volCtx != NULL)
                {

                        FltReleaseContext(volCtx);
                }
        }
      }
      return retValue;
}

FLT_POSTOP_CALLBACK_STATUS
SwapPostReadBuffers(
      _Inout_ PFLT_CALLBACK_DATA Data,
      _In_ PCFLT_RELATED_OBJECTS FltObjects,
      _In_ PVOID CompletionContext,
      _In_ FLT_POST_OPERATION_FLAGS Flags
```

```
)
{
     PVOID origBuf;
     PFLT_IO_PARAMETER_BLOCK iopb = Data->Iopb;
     FLT_POSTOP_CALLBACK_STATUS retValue = FLT_POSTOP_FINISHED_PROCESSING;
     PPRE_2_POST_CONTEXT p2pCtx = CompletionContext;
     BOOLEAN cleanupAllocatedBuffer = TRUE;

     FLT_ASSERT(!FlagOn(Flags, FLTFL_POST_OPERATION_DRAINING));

     try
     {
       if (!NT_SUCCESS(Data->IoStatus.Status) ||
              (Data->IoStatus.Information == 0))
       {

              LOG_PRINT(LOGFL_READ,
                     ("spibedrv!SwapPostReadBuffers:            %wZ newB=%p No
data read, status=%x, info=%Iu\n",
                            &p2pCtx->VolCtx->Name,
                            p2pCtx->UnpackedCandy,
                            Data->IoStatus.Status,
                            Data->IoStatus.Information));

              leave;
       }

       if (iopb->Parameters.Read.MdlAddress != NULL)
       {
              FLT_ASSERT(((PMDL)iopb->Parameters.Read.MdlAddress)->Next ==
NULL);

              origBuf = MmGetSystemAddressForMdlSafe(iopb-
>Parameters.Read.MdlAddress,
                     NormalPagePriority | MdlMappingNoExecute);

              if (origBuf == NULL)
              {
                     LOG_PRINT(LOGFL_ERRORS,
                            ("spibedrv!SwapPostReadBuffers:            %wZ
Failed to get system address for MDL: %p\n",
                                   &p2pCtx->VolCtx->Name,
                                   iopb->Parameters.Read.MdlAddress));

                     Data->IoStatus.Status = STATUS_INSUFFICIENT_RESOURCES;
                     Data->IoStatus.Information = 0;
                     leave;
              }

       }
       else if (FlagOn(Data->Flags, FLTFL_CALLBACK_DATA_SYSTEM_BUFFER) ||
              FlagOn(Data->Flags, FLTFL_CALLBACK_DATA_FAST_IO_OPERATION))
       {
              origBuf = iopb->Parameters.Read.ReadBuffer;
       }
       else
       {
              if (FltDoCompletionProcessingWhenSafe(Data,
                     FltObjects,
                     CompletionContext,
                     Flags,
                     SwapPostReadBuffersWhenSafe,
                     &retValue))
              {
                     cleanupAllocatedBuffer = FALSE;
              }
              else
              {
                LOG_PRINT(LOGFL_ERRORS,
                      ("spibedrv!SwapPostReadBuffers:            %wZ
Unable to post to a safe IRQL\n",
                            &p2pCtx->VolCtx->Name));

                Data->IoStatus.Status = STATUS_UNSUCCESSFUL;
                Data->IoStatus.Information = 0;
            }

            leave;
        }

        try
        {
            XOrBuffer((p2pCtx->UnpackedCandy),
                  Data->IoStatus.Information);

            RtlCopyMemory(origBuf,
                  p2pCtx->UnpackedCandy,
                  Data->IoStatus.Information);
        }
        except(EXCEPTION_EXECUTE_HANDLER)
        {
            Data->IoStatus.Status = GetExceptionCode();
            Data->IoStatus.Information = 0;

            LOG_PRINT(LOGFL_ERRORS,
                  ("spibedrv!SwapPostReadBuffers:            %wZ Invalid
user buffer, oldB=%p, status=%x\n",
                        &p2pCtx->VolCtx->Name,
                        origBuf,
                        Data->IoStatus.Status));
        }

      }
      finally
      {
        if (cleanupAllocatedBuffer)
        {
            LOG_PRINT(LOGFL_READ,
                  ("spibedrv!SwapPostReadBuffers:            %wZ newB=%p
info=%Iu Freeing\n",
                        &p2pCtx->VolCtx->Name,
                        p2pCtx->UnpackedCandy,
                        Data->IoStatus.Information));

            FltFreePoolAlignedWithTag(FltObjects->Instance,
                  p2pCtx->UnpackedCandy,
                  BUFFER_SWAP_TAG);

            FltReleaseContext(p2pCtx->VolCtx);

            ExFreeToNPagedLookasideList(&Pre2PostContextList,
                  p2pCtx);
        }
      }
      return retValue;
}

FLT_POSTOP_CALLBACK_STATUS
SwapPostReadBuffersWhenSafe(
      _Inout_ PFLT_CALLBACK_DATA Data,
      _In_ PCFLT_RELATED_OBJECTS FltObjects,
      _In_ PVOID CompletionContext,
      _In_ FLT_POST_OPERATION_FLAGS Flags
)
{
      PFLT_IO_PARAMETER_BLOCK iopb = Data->Iopb;
```

```
     PPRE_2_POST_CONTEXT p2pCtx = CompletionContext;
     PVOID origBuf;
     NTSTATUS status;

     UNREFERENCED_PARAMETER(FltObjects);
     UNREFERENCED_PARAMETER(Flags);
     FLT_ASSERT(Data->IoStatus.Information != 0);

     status = FltLockUserBuffer(Data);

     if (!NT_SUCCESS(status))
     {
       LOG_PRINT(LOGFL_ERRORS,
             ("spibedrv!SwapPostReadBuffersWhenSafe:    %wZ Could not lock
user buffer, oldB=%p, status=%x\n",
                   &p2pCtx->VolCtx->Name,
                   iopb->Parameters.Read.ReadBuffer,
                   status));

       Data->IoStatus.Status = status;
       Data->IoStatus.Information = 0;
     }
     else
     {
       origBuf = MmGetSystemAddressForMdlSafe(iopb-
>Parameters.Read.MdlAddress,
             NormalPagePriority | MdlMappingNoExecute);

       if (origBuf == NULL)
       {
             LOG_PRINT(LOGFL_ERRORS,
                   ("spibedrv!SwapPostReadBuffersWhenSafe:    %wZ Failed to
get system address for MDL: %p\n",
                         &p2pCtx->VolCtx->Name,
                         iopb->Parameters.Read.MdlAddress));

             Data->IoStatus.Status = STATUS_INSUFFICIENT_RESOURCES;
             Data->IoStatus.Information = 0;
       }
       else
       {
             XOrBuffer(p2pCtx->UnpackedCandy,
                   Data->IoStatus.Information);

             RtlCopyMemory(origBuf,
                   p2pCtx->UnpackedCandy,
                   Data->IoStatus.Information);
       }
     }

     LOG_PRINT(LOGFL_READ,
       ("spibedrv!SwapPostReadBuffersWhenSafe:    %wZ newB=%p info=%Iu
Freeing\n",
             &p2pCtx->VolCtx->Name,
             p2pCtx->UnpackedCandy,
             Data->IoStatus.Information));

     FltFreePoolAlignedWithTag(FltObjects->Instance,
       p2pCtx->UnpackedCandy,
       BUFFER_SWAP_TAG);

     FltReleaseContext(p2pCtx->VolCtx);

     ExFreeToNPagedLookasideList(&Pre2PostContextList,
       p2pCtx);

     return FLT_POSTOP_FINISHED_PROCESSING;
}
```

```
FLT_PREOP_CALLBACK_STATUS
SwapPreDirCtrlBuffers(
     _Inout_ PFLT_CALLBACK_DATA Data,
     _In_ PCFLT_RELATED_OBJECTS FltObjects,
     _Flt_CompletionContext_Outptr_ PVOID *CompletionContext
)
{
     PFLT_IO_PARAMETER_BLOCK iopb = Data->Iopb;
     FLT_PREOP_CALLBACK_STATUS retValue = FLT_PREOP_SUCCESS_NO_CALLBACK;
     PVOID newBuf = NULL;
     PMDL newMdl = NULL;
     PVOLUME_CONTEXT volCtx = NULL;
     PPRE_2_POST_CONTEXT p2pCtx;
     NTSTATUS status;

     try
     {
       if (iopb->Parameters.DirectoryControl.QueryDirectory.Length == 0)
       {

              leave;
       }

       status = FltGetVolumeContext(FltObjects->Filter,
              FltObjects->Volume,
              &volCtx);

       if (!NT_SUCCESS(status))
       {

              LOG_PRINT(LOGFL_ERRORS,
                     ("spibedrv!SwapPreDirCtrlBuffers:          Error getting
volume context, status=%x\n",
                            status));

              leave;
       }

       newBuf = ExAllocatePoolWithTag(NonPagedPool,
              iopb->Parameters.DirectoryControl.QueryDirectory.Length,
              BUFFER_SWAP_TAG);

       if (newBuf == NULL)
       {
              LOG_PRINT(LOGFL_ERRORS,
                     ("spibedrv!SwapPreDirCtrlBuffers:          %wZ Failed to
allocate %d bytes of memory.\n",
                            &volCtx->Name,
                            iopb-
>Parameters.DirectoryControl.QueryDirectory.Length));

              leave;
       }

       RtlZeroMemory(newBuf, iopb-
>Parameters.DirectoryControl.QueryDirectory.Length);

       newMdl = IoAllocateMdl(newBuf,
              iopb->Parameters.DirectoryControl.QueryDirectory.Length,
              FALSE,
              FALSE,
              NULL);

       if (newMdl == NULL)
       {
              LOG_PRINT(LOGFL_ERRORS,
                    ("spibedrv!SwapPreDirCtrlBuffers:          %wZ Failed to
allocate MDL.\n",
                          &volCtx->Name));

                leave;
        }

        MmBuildMdlForNonPagedPool(newMdl);

        p2pCtx = ExAllocateFromNPagedLookasideList(&Pre2PostContextList);

        if (p2pCtx == NULL)
        {

                LOG_PRINT(LOGFL_ERRORS,
                    ("spibedrv!SwapPreDirCtrlBuffers:          %wZ Failed to
allocate pre2Post context structure\n",
                          &volCtx->Name));

                leave;
        }

        LOG_PRINT(LOGFL_DIRCTRL,
                ("spibedrv!SwapPreDirCtrlBuffers:          %wZ newB=%p newMdl=%p
oldB=%p oldMdl=%p len=%d\n",
                    &volCtx->Name,
                    newBuf,
                    newMdl,
                    iopb-
>Parameters.DirectoryControl.QueryDirectory.DirectoryBuffer,
                    iopb-
>Parameters.DirectoryControl.QueryDirectory.MdlAddress,
                    iopb->Parameters.DirectoryControl.QueryDirectory.Length));

        iopb->Parameters.DirectoryControl.QueryDirectory.DirectoryBuffer =
newBuf;
        iopb->Parameters.DirectoryControl.QueryDirectory.MdlAddress = newMdl;
        FltSetCallbackDataDirty(Data);

        p2pCtx->UnpackedCandy = newBuf;
        p2pCtx->VolCtx = volCtx;

        *CompletionContext = p2pCtx;

        retValue = FLT_PREOP_SUCCESS_WITH_CALLBACK;

      }
      finally
      {
        if (retValue != FLT_PREOP_SUCCESS_WITH_CALLBACK)
        {
                if (newBuf != NULL)
                {

                    ExFreePool(newBuf);
                }

                if (newMdl != NULL)
                {

                    IoFreeMdl(newMdl);
                }

                if (volCtx != NULL)
                {

                    FltReleaseContext(volCtx);
                }
            }
        }

        return retValue;
}


FLT_POSTOP_CALLBACK_STATUS
SwapPostDirCtrlBuffers(
        _Inout_ PFLT_CALLBACK_DATA Data,
        _In_ PCFLT_RELATED_OBJECTS FltObjects,
        _In_ PVOID CompletionContext,
        _In_ FLT_POST_OPERATION_FLAGS Flags
)

{
        PVOID origBuf;
        PFLT_IO_PARAMETER_BLOCK iopb = Data->Iopb;
        FLT_POSTOP_CALLBACK_STATUS retValue = FLT_POSTOP_FINISHED_PROCESSING;
        PPRE_2_POST_CONTEXT p2pCtx = CompletionContext;
        BOOLEAN cleanupAllocatedBuffer = TRUE;

        FLT_ASSERT(!FlagOn(Flags, FLTFL_POST_OPERATION_DRAINING));

        try
        {
          if (!NT_SUCCESS(Data->IoStatus.Status) ||
                (Data->IoStatus.Information == 0))
          {

                LOG_PRINT(LOGFL_DIRCTRL,
                        ("spibedrv!SwapPostDirCtrlBuffers:         %wZ newB=%p No
data read, status=%x, info=%Ix\n",
                                &p2pCtx->VolCtx->Name,
                                p2pCtx->UnpackedCandy,
                                Data->IoStatus.Status,
                                Data->IoStatus.Information));

                leave;
          }

          if (iopb->Parameters.DirectoryControl.QueryDirectory.MdlAddress !=
NULL)
          {
                origBuf = MmGetSystemAddressForMdlSafe(iopb-
>Parameters.DirectoryControl.QueryDirectory.MdlAddress,
                        NormalPagePriority | MdlMappingNoExecute);

                if (origBuf == NULL)
                {
                        LOG_PRINT(LOGFL_ERRORS,
                                ("spibedrv!SwapPostDirCtrlBuffers:         %wZ
Failed to get system address for MDL: %p\n",
                                        &p2pCtx->VolCtx->Name,
                                        iopb-
>Parameters.DirectoryControl.QueryDirectory.MdlAddress));

                        Data->IoStatus.Status = STATUS_INSUFFICIENT_RESOURCES;
                        Data->IoStatus.Information = 0;
                        leave;
                }

          }
          else if (FlagOn(Data->Flags, FLTFL_CALLBACK_DATA_SYSTEM_BUFFER) ||
                FlagOn(Data->Flags, FLTFL_CALLBACK_DATA_FAST_IO_OPERATION))
          {
                origBuf = iopb-
```

```
>Parameters.DirectoryControl.QueryDirectory.DirectoryBuffer;
        }
        else
        {
                if (FltDoCompletionProcessingWhenSafe(Data,
                        FltObjects,
                        CompletionContext,
                        Flags,
                        SwapPostDirCtrlBuffersWhenSafe,
                        &retValue))
                {
                        cleanupAllocatedBuffer = FALSE;
                }
                else
                {
                        LOG_PRINT(LOGFL_ERRORS,
                                ("spibedrv!SwapPostDirCtrlBuffers:          %wZ
Unable to post to a safe IRQL\n",
                                        &p2pCtx->VolCtx->Name));

                        Data->IoStatus.Status = STATUS_UNSUCCESSFUL;
                        Data->IoStatus.Information = 0;
                }

                leave;
        }

        try
        {

                RtlCopyMemory(origBuf,
                        p2pCtx->UnpackedCandy,
                        /*Data->IoStatus.Information*/
                        iopb->Parameters.DirectoryControl.QueryDirectory.Length);

        }
        except(EXCEPTION_EXECUTE_HANDLER)
        {
                Data->IoStatus.Status = GetExceptionCode();
                Data->IoStatus.Information = 0;

                LOG_PRINT(LOGFL_ERRORS,
                        ("spibedrv!SwapPostDirCtrlBuffers:          %wZ Invalid
user buffer, oldB=%p, status=%x, info=%Iu\n",
                                &p2pCtx->VolCtx->Name,
                                origBuf,
                                Data->IoStatus.Status,
                                Data->IoStatus.Information));
        }
      }
      finally
      {
        if (cleanupAllocatedBuffer)
        {
                LOG_PRINT(LOGFL_DIRCTRL,
                        ("spibedrv!SwapPostDirCtrlBuffers:          %wZ newB=%p
info=%Iu Freeing\n",
                                &p2pCtx->VolCtx->Name,
                                p2pCtx->UnpackedCandy,
                                Data->IoStatus.Information));

                ExFreePool(p2pCtx->UnpackedCandy);
                FltReleaseContext(p2pCtx->VolCtx);

                ExFreeToNPagedLookasideList(&Pre2PostContextList,
                        p2pCtx);
        }
      }
```

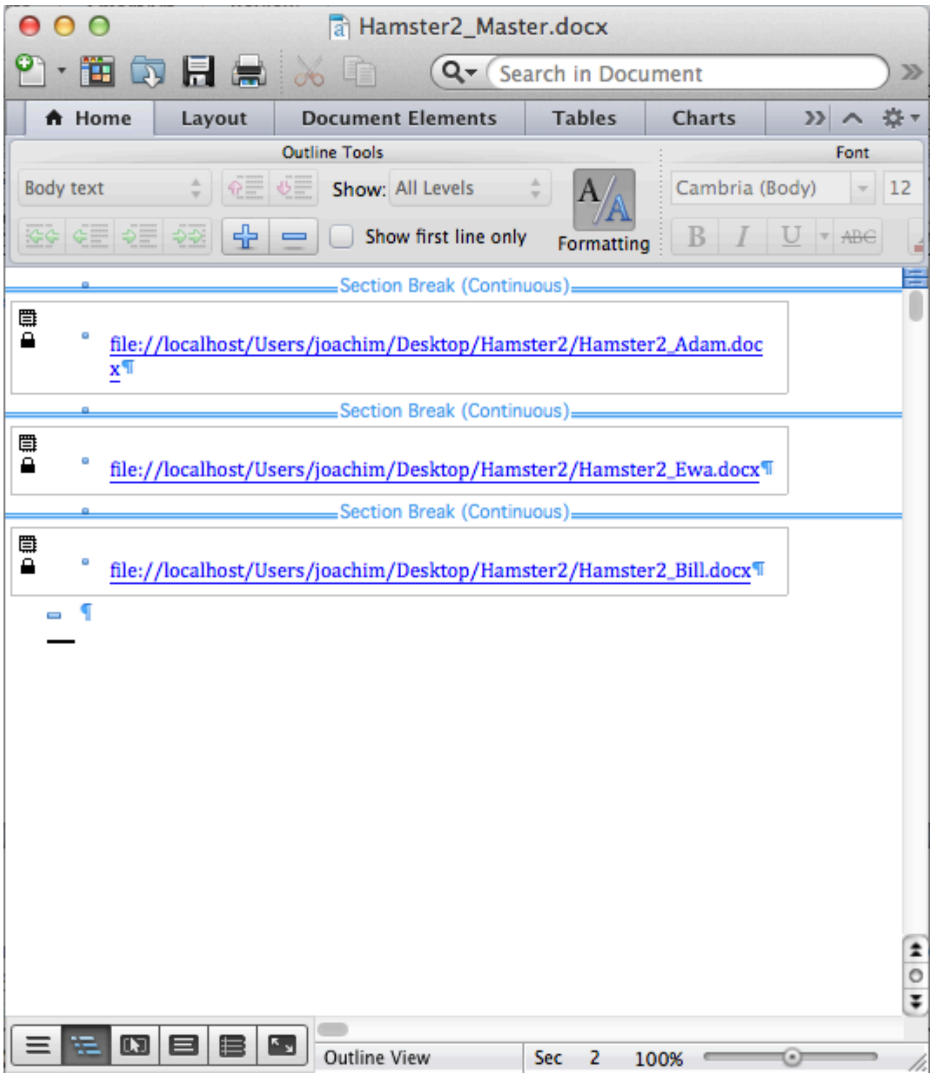
```
    return retValue;
}

FLT_POSTOP_CALLBACK_STATUS
SwapPostDirCtrlBuffersWhenSafe(
    _Inout_ PFLT_CALLBACK_DATA Data,
    _In_ PCFLT_RELATED_OBJECTS FltObjects,
    _In_ PVOID CompletionContext,
    _In_ FLT_POST_OPERATION_FLAGS Flags
)
{
    PFLT_IO_PARAMETER_BLOCK iopb = Data->Iopb;
    PPRE_2_POST_CONTEXT p2pCtx = CompletionContext;
    PVOID origBuf;
    NTSTATUS status;

    UNREFERENCED_PARAMETER(FltObjects);
    UNREFERENCED_PARAMETER(Flags);
    FLT_ASSERT(Data->IoStatus.Information != 0);

    status = FltLockUserBuffer(Data);

    if (!NT_SUCCESS(status))
    {
      LOG_PRINT(LOGFL_ERRORS,
            ("spibedrv!SwapPostDirCtrlBuffersWhenSafe: %wZ Could not lock
user buffer, oldB=%p, status=%x\n",
                    &p2pCtx->VolCtx->Name,
                    iopb-
>Parameters.DirectoryControl.QueryDirectory.DirectoryBuffer,
                    status));

      Data->IoStatus.Status = status;
      Data->IoStatus.Information = 0;
    }
    else
    {
      origBuf = MmGetSystemAddressForMdlSafe(iopb-
>Parameters.DirectoryControl.QueryDirectory.MdlAddress,
            NormalPagePriority | MdlMappingNoExecute);

      if (origBuf == NULL)
      {
            LOG_PRINT(LOGFL_ERRORS,
                    ("spibedrv!SwapPostDirCtrlBuffersWhenSafe: %wZ Failed to
get System address for MDL: %p\n",
                            &p2pCtx->VolCtx->Name,
                            iopb-
>Parameters.DirectoryControl.QueryDirectory.MdlAddress));

            Data->IoStatus.Status = STATUS_INSUFFICIENT_RESOURCES;
            Data->IoStatus.Information = 0;
      }
      else
      {
            RtlCopyMemory(origBuf,
                    p2pCtx->UnpackedCandy,
                    /*Data->IoStatus.Information*/
                    iopb->Parameters.DirectoryControl.QueryDirectory.Length);
      }
    }

    LOG_PRINT(LOGFL_DIRCTRL,
      ("spibedrv!SwapPostDirCtrlBuffersWhenSafe: %wZ newB=%p info=%Iu
Freeing\n",
            &p2pCtx->VolCtx->Name,
            p2pCtx->UnpackedCandy,
            Data->IoStatus.Information));

    ExFreePool(p2pCtx->UnpackedCandy);
    FltReleaseContext(p2pCtx->VolCtx);

    ExFreeToNPagedLookasideList(&Pre2PostContextList,
      p2pCtx);

    return FLT_POSTOP_FINISHED_PROCESSING;
}

FLT_PREOP_CALLBACK_STATUS
SwapPreWriteBuffers(
    _Inout_ PFLT_CALLBACK_DATA Data,
    _In_ PCFLT_RELATED_OBJECTS FltObjects,
    _Flt_CompletionContext_Outptr_ PVOID *CompletionContext
)
{
    PFLT_IO_PARAMETER_BLOCK iopb = Data->Iopb;
    FLT_PREOP_CALLBACK_STATUS retValue = FLT_PREOP_SUCCESS_NO_CALLBACK;
    PVOID newBuf = NULL;
    PMDL newMdl = NULL;
    PVOLUME_CONTEXT volCtx = NULL;
    PPRE_2_POST_CONTEXT p2pCtx;
    PVOID origBuf;
    NTSTATUS status;
    ULONG writeLen = iopb->Parameters.Write.Length;

    try
    {
      if (writeLen == 0)
      {
            leave;
      }

      status = FltGetVolumeContext(FltObjects->Filter,
            FltObjects->Volume,
            &volCtx);

      if (!NT_SUCCESS(status))
      {

            LOG_PRINT(LOGFL_ERRORS,
                    ("spibedrv!SwapPreWriteBuffers:            Error getting
volume context, status=%x\n",
                            status));

            leave;
      }

      if (FlagOn(IRP_NOCACHE, iopb->IrpFlags))
      {
            writeLen = (ULONG)ROUND_TO_SIZE(writeLen, volCtx->SectorSize);
      }

      newBuf = FltAllocatePoolAlignedWithTag(FltObjects->Instance,
            NonPagedPool,
            (SIZE_T)writeLen,
            BUFFER_SWAP_TAG);

      if (newBuf == NULL)
      {

            LOG_PRINT(LOGFL_ERRORS,
                    ("spibedrv!SwapPreWriteBuffers:            %wZ Failed to
allocate %d bytes of memory.\n",
                            &volCtx->Name,
                            writeLen));
            leave;
      }

      if (FlagOn(Data->Flags, FLTFL_CALLBACK_DATA_IRP_OPERATION))
      {
            newMdl = IoAllocateMdl(newBuf,
                    writeLen,
                    FALSE,
                    FALSE,
                    NULL);

            if (newMdl == NULL)
            {

                    LOG_PRINT(LOGFL_ERRORS,
                            ("spibedrv!SwapPreWriteBuffers:              %wZ
Failed to allocate MDL.\n",
                                        &volCtx->Name));

                    leave;
            }
            MmBuildMdlForNonPagedPool(newMdl);
      }

      if (iopb->Parameters.Write.MdlAddress != NULL)
      {
            FLT_ASSERT(((PMDL)iopb->Parameters.Write.MdlAddress)->Next ==
NULL);

            origBuf = MmGetSystemAddressForMdlSafe(iopb-
>Parameters.Write.MdlAddress,
                    NormalPagePriority | MdlMappingNoExecute);

            if (origBuf == NULL)
            {
                    LOG_PRINT(LOGFL_ERRORS,
                            ("spibedrv!SwapPreWriteBuffers:              %wZ
Failed to get system address for MDL: %p\n",
                                        &volCtx->Name,
                                        iopb->Parameters.Write.MdlAddress));

                    Data->IoStatus.Status = STATUS_INSUFFICIENT_RESOURCES;
                    Data->IoStatus.Information = 0;
                    retValue = FLT_PREOP_COMPLETE;
                    leave;
            }
      }
      else
      {
            origBuf = iopb->Parameters.Write.WriteBuffer;
      }

      try
      {
            RtlCopyMemory(newBuf,
                    origBuf,
                    writeLen);
            XOrBuffer(newBuf, writeLen);
      }
      except(EXCEPTION_EXECUTE_HANDLER)
      {
            Data->IoStatus.Status = GetExceptionCode();
            Data->IoStatus.Information = 0;
            retValue = FLT_PREOP_COMPLETE;

            LOG_PRINT(LOGFL_ERRORS,
                    ("spibedrv!SwapPreWriteBuffers:              %wZ Invalid
user buffer, oldB=%p, status=%x\n",
                            &volCtx->Name,
                            origBuf,
                            Data->IoStatus.Status));

                leave;
        }

        p2pCtx = ExAllocateFromNPagedLookasideList(&Pre2PostContextList);

        if (p2pCtx == NULL)
        {
                LOG_PRINT(LOGFL_ERRORS,
                        ("spibedrv!SwapPreWriteBuffers:            %wZ Failed to
allocate pre2Post context structure\n",
                            &volCtx->Name));

                leave;
        }

        LOG_PRINT(LOGFL_WRITE,
                ("spibedrv!SwapPreWriteBuffers:            %wZ newB=%p newMdl=%p
oldB=%p oldMdl=%p len=%d\n",
                        &volCtx->Name,
                        newBuf,
                        newMdl,
                        iopb->Parameters.Write.WriteBuffer,
                        iopb->Parameters.Write.MdlAddress,
                        writeLen));

        iopb->Parameters.Write.WriteBuffer = newBuf;
        iopb->Parameters.Write.MdlAddress = newMdl;
        FltSetCallbackDataDirty(Data);

        p2pCtx->UnpackedCandy = newBuf;
        p2pCtx->VolCtx = volCtx;

        *CompletionContext = p2pCtx;

        retValue = FLT_PREOP_SUCCESS_WITH_CALLBACK;
      }
      finally
      {
        if (retValue != FLT_PREOP_SUCCESS_WITH_CALLBACK)
        {

                if (newBuf != NULL)
                {

                        FltFreePoolAlignedWithTag(FltObjects->Instance,
                                newBuf,
                                BUFFER_SWAP_TAG);

                }

                if (newMdl != NULL)
                {

                        IoFreeMdl(newMdl);
                }

                if (volCtx != NULL)
                {

                        FltReleaseContext(volCtx);
                }
        }
```

```
    }

    return retValue;
}


FLT_POSTOP_CALLBACK_STATUS
SwapPostWriteBuffers(
    _Inout_ PFLT_CALLBACK_DATA Data,
    _In_ PCFLT_RELATED_OBJECTS FltObjects,
    _In_ PVOID CompletionContext,
    _In_ FLT_POST_OPERATION_FLAGS Flags
)
{
    PPRE_2_POST_CONTEXT p2pCtx = CompletionContext;

    UNREFERENCED_PARAMETER(FltObjects);
    UNREFERENCED_PARAMETER(Flags);

    LOG_PRINT(LOGFL_WRITE,
      ("spibedrv!SwapPostWriteBuffers:           %wZ newB=%p info=%Iu
Freeing\n",
            &p2pCtx->VolCtx->Name,
            p2pCtx->UnpackedCandy,
            Data->IoStatus.Information));


    FltFreePoolAlignedWithTag(FltObjects->Instance,
      p2pCtx->UnpackedCandy,
      BUFFER_SWAP_TAG);

    FltReleaseContext(p2pCtx->VolCtx);

    ExFreeToNPagedLookasideList(&Pre2PostContextList,
      p2pCtx);

    return FLT_POSTOP_FINISHED_PROCESSING;
}


VOID
ReadDriverParameters(
    _In_ PUNICODE_STRING RegistryPath
)
{
    OBJECT_ATTRIBUTES attributes;
    HANDLE driverRegKey;
    NTSTATUS status;
    ULONG resultLength;
    UNICODE_STRING valueName;
    UCHAR buffer[sizeof(KEY_VALUE_PARTIAL_INFORMATION) + sizeof(LONG)];

    if (0 == LoggingFlags)
    {
      InitializeObjectAttributes(&attributes,
            RegistryPath,
            OBJ_CASE_INSENSITIVE | OBJ_KERNEL_HANDLE,
            NULL,
            NULL);

      status = ZwOpenKey(&driverRegKey,
            KEY_READ,
            &attributes);

      if (!NT_SUCCESS(status))
      {
            return;
        }

        RtlInitUnicodeString(&valueName, L"DebugFlags");

        status = ZwQueryValueKey(driverRegKey,
            &valueName,
            KeyValuePartialInformation,
            buffer,
            sizeof(buffer),
            &resultLength);

        if (NT_SUCCESS(status))
        {

            LoggingFlags = *((PULONG)
&(((PKEY_VALUE_PARTIAL_INFORMATION)buffer)->Data));
        }

        ZwClose(driverRegKey);
      }
}
```

APPENDIX KK OOXML STICKY POLICY HANDLER EVALUATION – SPIBEDRV\SPIBEDRV.INF

```
[Version]
signature   = "$Windows NT$"
Class       = "Encryption"
ClassGuid   = {46b682d4-a96f-409b-b450-f56d3a6f703b}
Provider    = %ProviderString%
DriverVer   = 04/08/2018,1.0.0.0
CatalogFile = spibedrv.cat

[DestinationDirs]
DefaultDestDir          = 12
MiniFilter.DriverFiles  = 12            ;%windir%\system32\drivers

[DefaultInstall]
OptionDesc          = %ServiceDescription%
CopyFiles           = MiniFilter.DriverFiles

[DefaultInstall.Services]
AddService          = %ServiceName%,,MiniFilter.Service

[DefaultUninstall]
DelFiles     = MiniFilter.DriverFiles
DelReg       = MiniFilter.DelRegistry

[DefaultUninstall.Services]
DelService = spibedrv,0x200

[MiniFilter.Service]
DisplayName      = %ServiceName%
Description      = %ServiceDescription%
ServiceBinary    = %12%\%DriverName%.sys        ;%windir%\system32\drivers\
Dependencies     = "FltMgr"
ServiceType      = 2                            ;SERVICE_FILE_SYSTEM_DRIVER
StartType        = 0                           ;SERVICE_BOOT_START
StartType        = 3                            ;SERVICE_DEMAND_START
ErrorControl     = 1                            ;SERVICE_ERROR_NORMAL
LoadOrderGroup   = "FSFilter Encryption"
AddReg           = MiniFilter.AddRegistry

[MiniFilter.AddRegistry]
HKR,,"SupportedFeatures",0x00010001,0x3
HKR,"Instances","DefaultInstance",0x00000000,%Instance1.Name%
HKR,"Instances\"%Instance1.Name%,"Altitude",0x00000000,%Instance1.Altitude%
HKR,"Instances\"%Instance1.Name%,"Flags",0x00010001,%Instance1.Flags%

[MiniFilter.DelRegistry]
HKR,,"SupportedFeatures",0x00010001,0x3
HKR,"Instances","DefaultInstance",0x00000000,%Instance1.Name%
HKR,"Instances\"%Instance1.Name%,"Altitude",0x00000000,%Instance1.Altitude%
HKR,"Instances\"%Instance1.Name%,"Flags",0x00010001,%Instance1.Flags%

[MiniFilter.DriverFiles]
%DriverName%.sys

[SourceDisksFiles]
spibedrv.sys = 1,,

[SourceDisksNames]
1 = %DiskId1%,,,

[Strings]
ProviderString          = "Napier"
ServiceDescription      = "Sticky Policy Identity-based Encryption Mini-Filter
Driver"
```

```
ServiceName             = "spibedrv"
DriverName              = "spibedrv"
DiskId1                 = "spibedrv Device Installation Disk"

Instance1.Name          = "spibedrv Instance"
Instance1.Altitude      = "141000"
Instance1.Flags         = 0x0          ; allow automatic attachments
```



Section Break (Continuous)

file://localhost/Users/joachim/Desktop/Hamster2/Hamster2_Adam.docx¶

Section Break (Continuous)

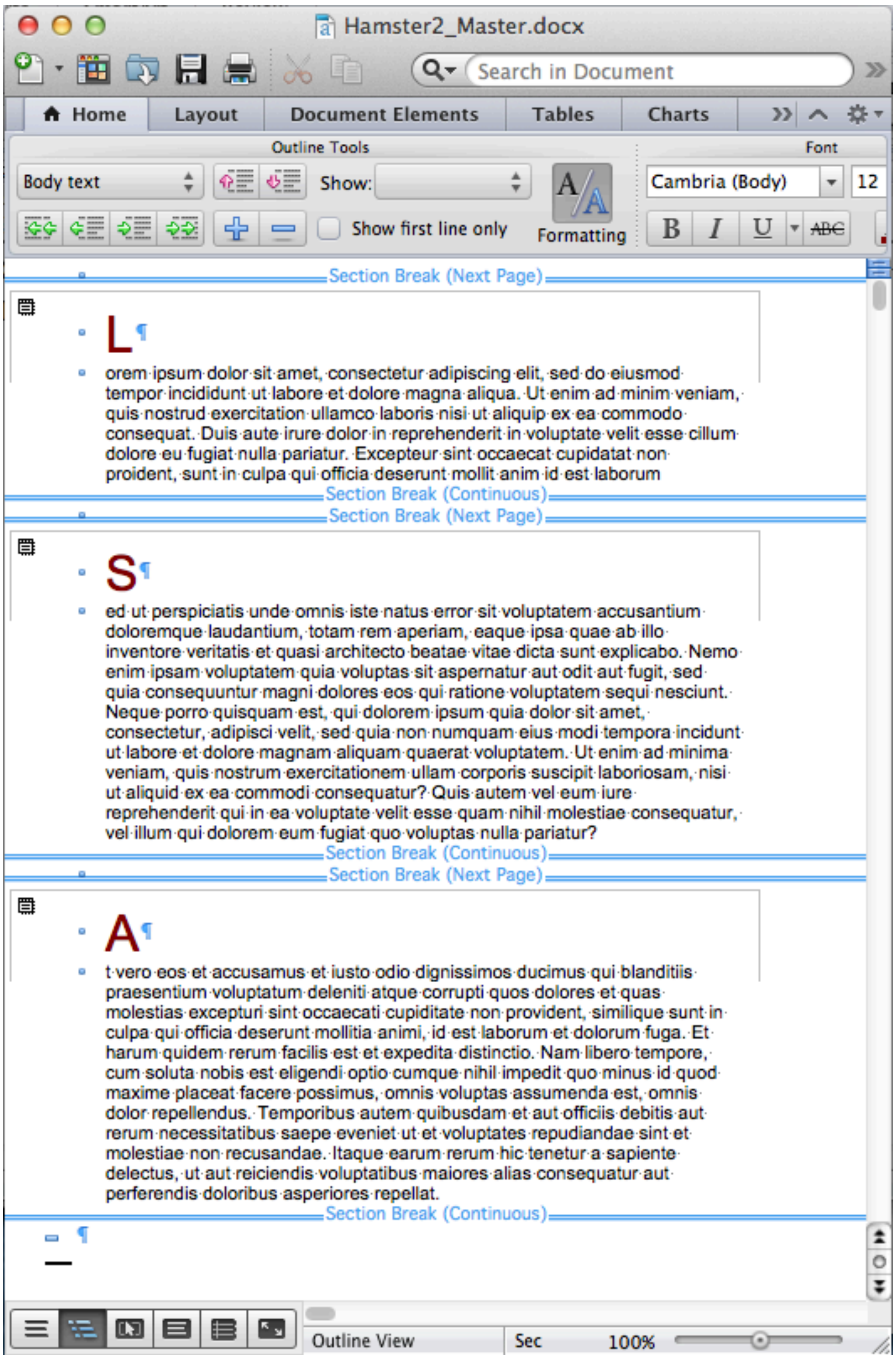
file://localhost/Users/joachim/Desktop/Hamster2/Hamster2_Ewa.docx¶

Section Break (Continuous)

file://localhost/Users/joachim/Desktop/Hamster2/Hamster2_Bill.docx¶

¶

APPENDIX MM   OOXML MASTER DOCUMENT ACCESS CONTROL – UNLOCKED



Section Break (Next Page)

L¶

orem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Section Break (Continuous)

Section Break (Next Page)

S¶

ed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Section Break (Continuous)

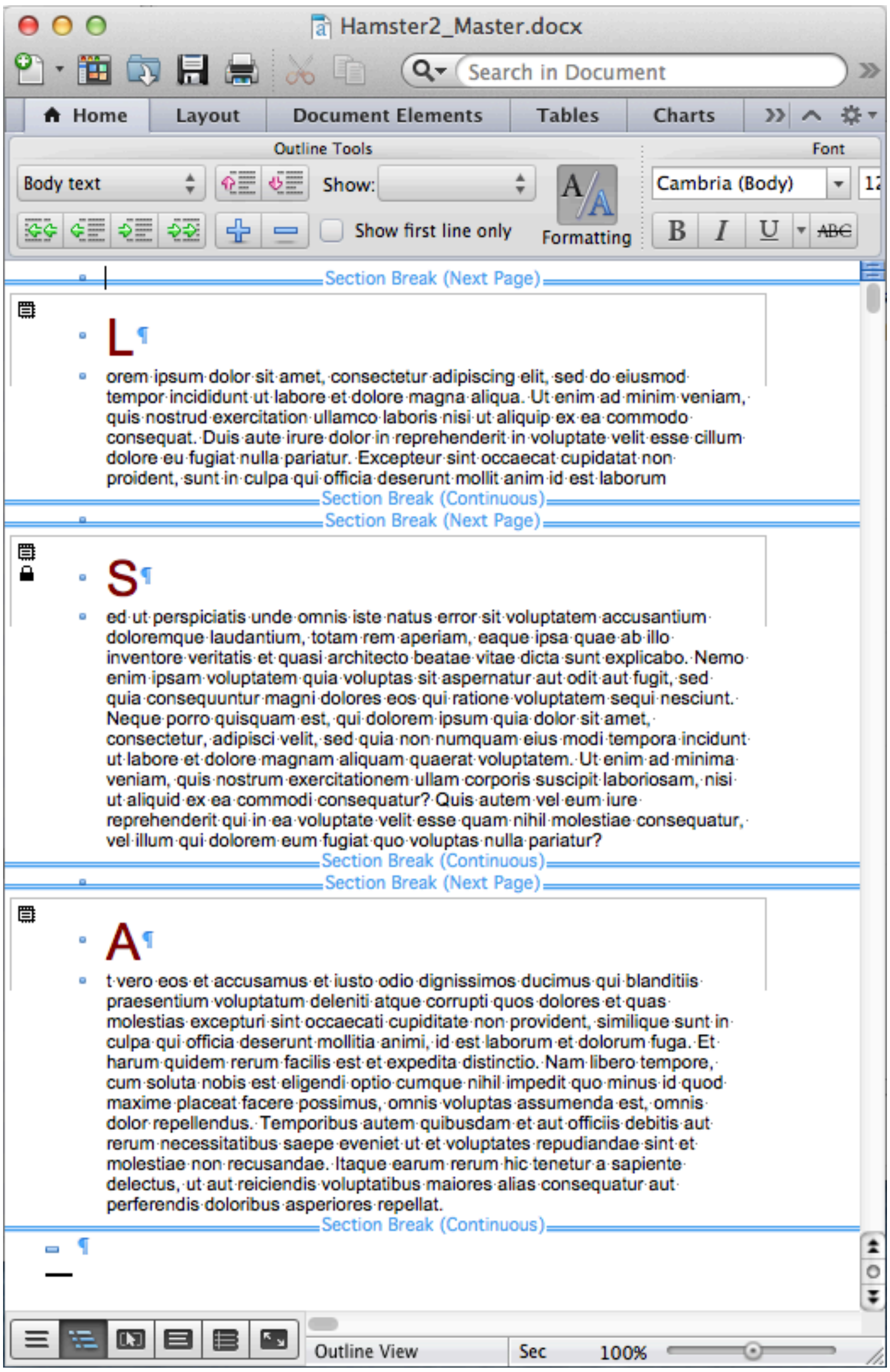
Section Break (Next Page)

A¶

t vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat.

Section Break (Continuous)

¶



L¶

orem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

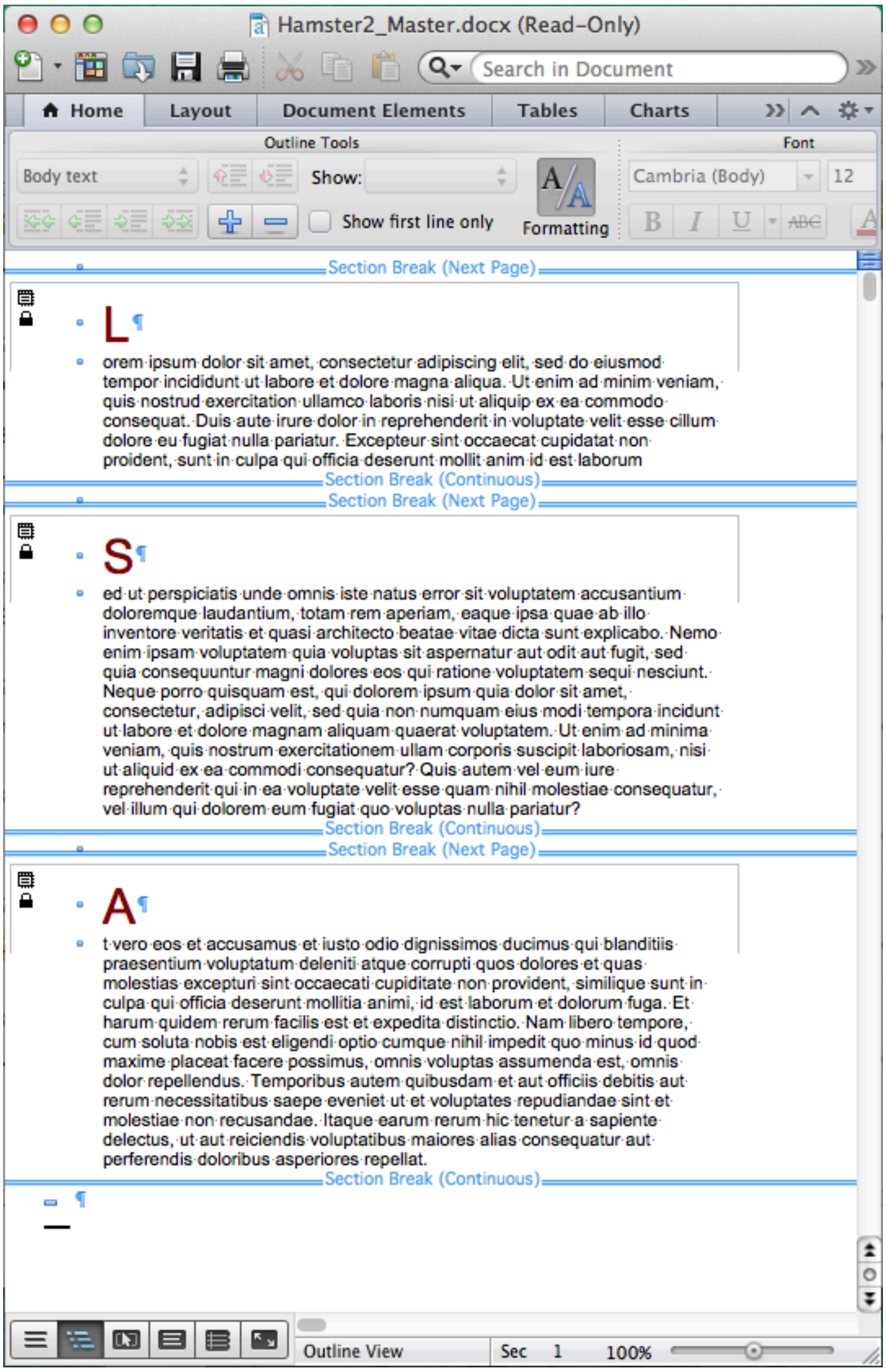
S¶

ed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

A¶

t vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat.

APPENDIX OO OOXML MASTER DOCUMENT ACCESS CONTROL – CURRENT DOCUMENT PROCESSOR WAS GRANTED READONLY RIGHTS OVER ENTIRE CONTENT



L¶

orem·ipsum·dolor·sit·amet,·consectetur·adipiscing·elit,·sed·do·eiusmod·tempor·incididunt·ut·labore·et·dolore·magna·aliqua.·Ut·enim·ad·minim·veniam,·quis·nostrud·exercitation·ullamco·laboris·nisi·ut·aliquip·ex·ea·commodo·consequat.·Duis·aute·irure·dolor·in·reprehenderit·in·voluptate·velit·esse·cillum·dolore·eu·fugiat·nulla·pariatur.·Excepteur·sint·occaecat·cupidatat·non·proident,·sunt·in·culpa·qui·officia·deserunt·mollit·anim·id·est·laborum

S¶

ed·ut·perspiciatis·unde·omnis·iste·natus·error·sit·voluptatem·accusantium·doloremque·laudantium,·totam·rem·aperiam,·eaque·ipsa·quae·ab·illo·inventore·veritatis·et·quasi·architecto·beatae·vitae·dicta·sunt·explicabo.·Nemo·enim·ipsam·voluptatem·quia·voluptas·sit·aspernatur·aut·odit·aut·fugit,·sed·quia·consequuntur·magni·dolores·eos·qui·ratione·voluptatem·sequi·nesciunt.·Neque·porro·quisquam·est,·qui·dolorem·ipsum·quia·dolor·sit·amet,·consectetur,·adipisci·velit,·sed·quia·non·numquam·eius·modi·tempora·incidunt·ut·labore·et·dolore·magnam·aliquam·quaerat·voluptatem.·Ut·enim·ad·minima·veniam,·quis·nostrum·exercitationem·ullam·corporis·suscipit·laboriosam,·nisi·ut·aliquid·ex·ea·commodi·consequatur?·Quis·autem·vel·eum·iure·reprehenderit·qui·in·ea·voluptate·velit·esse·quam·nihil·molestiae·consequatur,·vel·illum·qui·dolorem·eum·fugiat·quo·voluptas·nulla·pariatur?

A¶

t·vero·eos·et·accusamus·et·iusto·odio·dignissimos·ducimus·qui·blanditiis·praesentium·voluptatum·deleniti·atque·corrupti·quos·dolores·et·quas·molestias·excepturi·sint·occaecati·cupiditate·non·provident,·similique·sunt·in·culpa·qui·officia·deserunt·mollitia·animi,·id·est·laborum·et·dolorum·fuga.·Et·harum·quidem·rerum·facilis·est·et·expedita·distinctio.·Nam·libero·tempore,·cum·soluta·nobis·est·eligendi·optio·cumque·nihil·impedit·quo·minus·id·quod·maxime·placeat·facere·possimus,·omnis·voluptas·assumenda·est,·omnis·dolor·repellendus.·Temporibus·autem·quibusdam·et·aut·officiis·debitis·aut·rerum·necessitatibus·saepe·eveniet·ut·et·voluptates·repudiandae·sint·et·molestiae·non·recusandae.·Itaque·earum·rerum·hic·tenetur·a·sapiente·delectus,·ut·aut·reiciendis·voluptatibus·maiores·alias·consequatur·aut·perferendis·doloribus·asperiores·repellat.