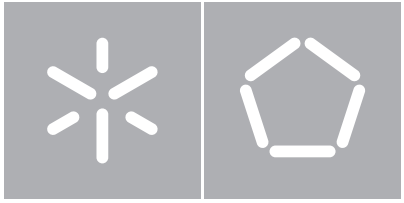
Universidade do Minho
Escola de Engenharia

Ricardo Joaquim Pereira de Macedo

Reforço da Privacidade Através do Controlo da Pegada Digital

Junho de 2013



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Ricardo Joaquim Pereira de Macedo

Reforço da Privacidade Através do Controlo da Pegada Digital

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de

Professor José Carlos Bacelar

Junho de 2013

Agradecimentos

Quero agradecer ao meu orientador académico, o Professor José Carlos Bacelar, pela colaboração, orientação e disponibilidade durante todo o projeto.

Ao meu orientador na PT Inovação, o Engenheiro Ricardo Azevedo, por me ter encaminhado na elaboração deste projeto e também à restante equipa da PT inovação pela ajuda prestada.

Aos meus colegas e amigos, Henrique Martins, Vasco Amaral e Samuel Rodrigues, pela companhia e apoio prestado.

Aos meus pais, Joaquim e Augusta Macedo, por me terem ajudado sempre que precisei e por me terem dado todas as condições e motivação para atingir as minhas metas. E ao meu irmão, Rafael Macedo, pelo companheirismo e amizade.

Por fim, quero agradecer especialmente à Marta Fernandes, por ter sido o pilar, ajudando-me na elaboração do projeto e apoiando-me em todos os momentos.

A todos o meu muito obrigado.

Resumo

Atualmente existe ainda uma relação assimétrica entre os utilizadores e os fornecedores de serviços disponibilizados pela internet. É prática comum, aquando da apresentação de um serviço, que o utilizador seja questionado sobre a aceitação, ou não, de um conjunto de políticas referentes ao uso de informação privada facultada ao fornecedor (por exemplo, a morada, o número de telefone, preferências, etc...). Geralmente os utilizadores aceitam a política com base na confiança que têm no fornecedor e/ou no contrato formal que lhes é apresentado. Os casos de violação de privacidade por parte de alguns fornecedores de serviços, vendendo ou facultando informação privada sobre os seus clientes a outros, são amplamente conhecidos e resultam em grande medida da falta de controlo que os utilizadores finais têm sobre a informação que entregam aos fornecedores.

Este problema também tem grande impacto no ambiente empresarial. Quase toda a informação de uma organização é guardada em claro. Mesmo que esta seja guardada num local seguro, aqueles que conhecerem bem o sistema poderão ter indevidamente acesso a informação privada da organização. Além disto, se a organização for alvo de um ataque informático e o atacante conseguir aceder aos dados poderá consulta-los livremente.

Neste trabalho propomos a implementação de um mecanismo que possibilite o envio de informações sem que o utilizador tenha necessidade de confiar no local onde as mesmas serão armazenadas, através da utilização do conceito de "sticky policies". Através da utilização de técnicas criptográficas, é estabelecido um vínculo entre a informação cifrada e as políticas de acesso à informação. O sistema desenvolvido garante que, para um terceiro aceder às informações pessoais de um utilizador, terá que cumprir o conjunto de regras definidas pelo dono da informação.

Visto que um utilizador autorizado a aceder às informações pode ter um comportamento incorreto, partilhando indevidamente as informações, propomos também adicionar mecanismos de auditoria dos acessos à informação gerida pelo sistema.

Palavras-chave: "Sticky policies", sistema criptográfico, PKE, IBE, ABE, PRE, controlo de acessos, DAC, MAC, ABAC, RBAC, linguagem de políticas, XACML, XrML, EPAL.

Abstract

Nowadays there is an asymmetrical relationship between users and service providers available over the internet. A common practice during service subscription is to ask users to accept a set of policies regarding use of private information (for example, address, telephone number, preferences, etc...). Generally users agree to the policy based on the confidence they have in the supplier and/or the formal contract that is presented to them. Cases of violation of privacy by some service providers, selling or providing private information about their customers to others, are widely known and result in large part from the lack of control that end users have over the information they deliver to suppliers.

This issue also has great impact on business environment. Almost all the information of an organization is stored in clear. Even though it is stored in a safe place, those who know the system may have improper access to private information.

In this work we propose the implementation of a mechanism for sending information without the user ever need to trust where they will be stored, using the concept of sticky policies. Through the use of cryptographic techniques, a link is established between the encrypted information and their access control policies. The system ensures that when a third party tries to access the information, has to fulfill the set of rules defined by the owner of the information.

Since a user authorized to access the information may have an incorrect behavior, by improperly sharing information, we also propose to add auditing mechanisms to the information managed by the system.

Keywords: Sticky policies, cryptographic system, PKE, IBE, ABE, PRE, access control, DAC, MAC, ABAC, RBAC, policy language, XACML, XrML, EPAL.

Conteúdo

Lista de Figuras

Lista de Tabelas

1. Introdução

1.1 Motivação

Nestes últimos anos tem-se testemunhado um grande crescimento no número de utilizadores da internet assim como na quantidade de informação disponível online. Segundo o site Internet World Stats [56] haviam, em 30 de junho de 2012, mais de 2.4 mil milhões de utilizadores, enquanto que a 31 de dezembro de 2000 apenas se registavam perto de 361 milhões de utilizadores. Tal facto representa um crescimento de 566% em doze anos.

Este crescimento de utilizadores na internet fez com que se fossem criando comunidades de utilizadores e para ajudar a comunicação entre utilizadores e criação de comunidades surgiu o conceito de redes sociais.

A utilização das redes sociais tem vindo igualmente a aumentar nos últimos anos, tomando como exemplo o facebook [20] (a rede social com mais utilizadores atualmente), que segundo o site Internet World Stats [55], a 31 de março de 2012 continha mais de 835 milhões de utilizadores. Comparando com o número de utilizadores registados a 31 de março de 2011 (664 milhões), vê-se que apenas num ano o número de utilizadores do facebook aumentou quase 26%.

Este crescimento e com as notícias diárias sobre ataques informáticos, originou uma insegurança na utilização da internet por um grande número de utilizadores.

Para além deste facto, existe ainda uma relação assimétrica entre os utilizadores e os fornecedores de serviços. É prática comum, aquando da apresentação de um serviço, que o utilizador seja questionado sobre a aceitação, ou não, de um conjunto de políticas referentes ao uso da sua informação facultada ao fornecedor (essa informação, parte da identidade do utilizador, é por exemplo a morada, o número de telefone, etc...). Geralmente os utilizadores aceitam a política, com base na confiança que têm do fornecedor e no contrato formal que lhes é apresentado. Os casos de violação de privacidade por parte de alguns fornecedores de serviços, vendendo ou facultando informação privada sobre os seus clientes a outros, são amplamente conhecidos e resultam da falta de controlo que os utilizadores finais têm sobre a informação que entregam aos fornecedores.

Por sua vez, o grande crescimento da internet e da informatização dos serviços promoveu enumeras vantagens para a população, mas com elas têm vindo a surgir alguns problemas.

Um dos grandes problemas da utilização da internet é o risco de desconhecidos terem acesso a informações pessoais. Isto pode acontecer através de ciber ataques (hacking) ou através da divulgação indevida por parte de pessoas autorizadas. Enviando informações pessoais para um terceiro corremos o risco de este as utilizar de forma inadequada, podendo divulgar as informações com outros, pondo a privacidade da pessoa em risco.

Este problema faz com que um utilizador, ao enviar informações pessoais, necessite de ter confiança no destinatário. Esta necessidade dos utilizadores confiarem nos serviços que utilizam fez surgir dois tipos de utilizadores, os que confiam em tudo, partilhando constantemente as suas

informações pessoais, e por outro lado, aqueles mais desconfiados que se privam da utilização de diversos serviços para terem um maior controlo sobre as suas informações pessoais.

No que diz respeito ao ambiente empresarial, este problema também tem um grande impacto. Quase toda a informação de uma organização é guardada em claro. Mesmo que esta seja guardada num local seguro, aqueles que conhecerem bem o sistema poderão ter indevidamente acesso a informação privada da organização. Além disto se a organização for alvo de um ataque informático e o atacante conseguir aceder aos dados poderá consulta-los livremente.

Em janeiro de 2003, Alan Giang Tran de 28 anos, efetuou um ataque aos computadores das empresas Airline Coach Service e Sky Limousine (pertencentes à mesma sociedade), onde trabalhava como administrador de redes. Neste ataque Tran alterou todas as passwords do sistema, o que causou grandes prejuízos à empresa [43]. Neste caso o atacante não acedeu às informações internas, mas poderia ter acedido e vendido essas informações a outras empresas. Este é um exemplo da importância das informações estarem protegidas de colaboradores não autorizados da empresa.

Por outro lado, uma das grandes vantagens da utilização da internet é a facilidade com que se pode partilhar informação, porém para um utilizador enviar as suas informações necessita de acreditar que estarão seguras. Como referido anteriormente, existem muitos fatores que tornam os utilizadores inseguros. Deste modo, surge a necessidade de uma ferramenta que permita aos utilizadores poderem controlar a sua informação.

O presente projeto centra-se no problema de não existirem mecanismos de controlo das informações partilhadas com terceiros, fazendo com que uma vez enviadas as informações, o utilizador deixa de ter qualquer controlo sobre as mesmas. Assim sendo, o projeto propõe a criação de uma framework que permita aos utilizadores controlar os acessos às suas informações, por parte de terceiros.

1.2 O Conceito de "Sticky Policies"

O conceito de "sticky policies" foi introduzido pelos artigos [30] e [29] no âmbito do projeto de linguagem de políticas P3P [63]. Nestes artigos foi proposto a utilização de uma linguagem que permitisse associar um conjunto de políticas a informações, introduzido o paradigma de "sticky policies" e respetivos mecanismos para o cumprimento da privacidade empresarial.

A necessidade de criar um mecanismo para permitir a associação de políticas a informações surge do facto de que até ao momento as políticas de controlo de acesso serem apenas baseadas num conjunto de regras definidas pelo administrador de segurança do sistema. Este conjunto de regras consiste na especificação do modo (leitura, escrita, execução) em que objetos (ficheiros, aplicações) podem ser acedidos e por quais utilizadores ou conjunto de utilizadores.

Através da associação de um conjunto de políticas a informações é possível que o controlo de acessos seja realizado segundo políticas mais variadas e permitindo que estas sejam definidas pelos donos das informações.

Contudo, o conceito proposto pelos artigos pressupõe que se está inserido num sistema confiável em que existe um monitor de recurso que apenas autoriza o acesso aos recursos se forem respeitadas as políticas. Isto faz com que as políticas só necessitam de ser respeitadas dentro de sistemas semelhantes, devido a este problema foi introduzida criptografia no conceito das "sticky policies".

Um exemplo da utilização de criptografia para implementar o conceito de "sticky policies" pode ser visto no artigo [39]. Neste artigo é proposto uma arquitetura onde as informações são cifradas e as políticas associadas são mapeadas na chave que permite decifrar a informação. Apenas autoridades de confiança conseguem gerar a chave. Assim ninguém, não autorizado consegue aceder às informações ou alterar as políticas.

Cifrando os dados e ligando as políticas à chave ou aos dados cifrados é possível enviar um conjunto de informações para entidades desconhecidas, pois é certo que apenas terão acesso às informações caso respeitem as políticas associadas.

1.3 Arquiteturas de "Sticky Policies"

São agora analisadas três diferentes arquiteturas de implementação de "sticky policies", estas arquiteturas são descritas nos artigos [39], [59] e [46].

O foco do artigo [59] não está na arquitetura utilizada, mas é um bom exemplo a considerar pois é bastante diferente das arquiteturas descritas nos artigos [39] e [46].

No artigo [59] é utilizada uma arquitetura composta por três elementos (figura 1.1), o dono da informação, uma entidade que deseja aceder à informação e outra entidade de confiança. Esta arquitetura é utilizada para explicar mecanismos criptográficos para garantir a segurança dos dados.

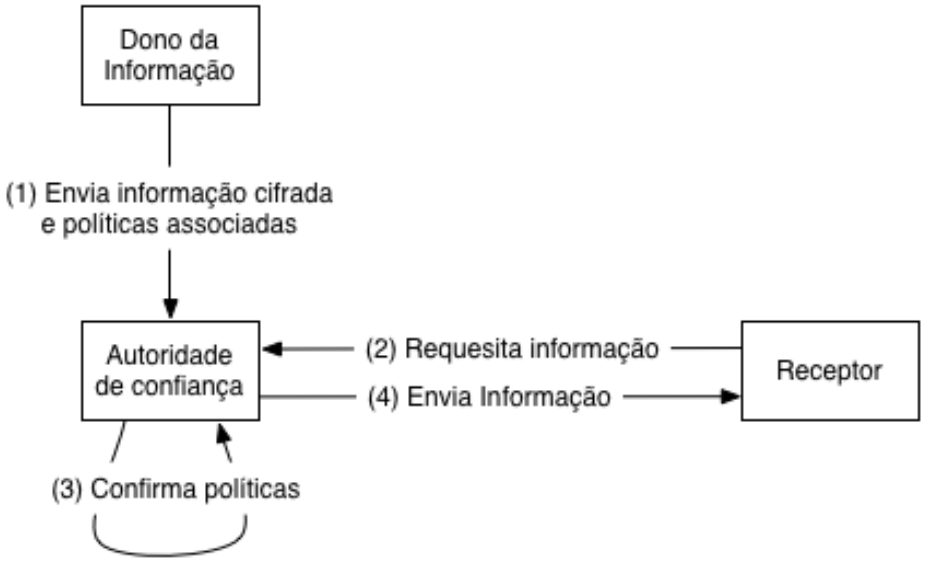


Figura 1.1 Exemplo de arquitetura de "sticky policies".

Nesta arquitetura o dono da informação envia para a entidade de confiança as informações e define as políticas que pretendem que lhes sejam associadas. Enquanto que, a entidade que deseja ter acesso às informações terá que fazer o pedido à entidade de confiança, esta enviar-lhe-á o pedido caso sejam respeitadas as políticas associadas.

A arquitetura está centralizada na entidade de confiança, pois é responsável por guardar, garantir a segurança dos dados enviados pelo dono e fazer com que as políticas sejam respeitadas.

Devido a esta centralização, as políticas não necessitam de estar fortemente ligadas às informações correspondentes, como a entidade é de confiança e os dados são guardados por si, a entidade conhece e não altera a associação políticas-informações. No caso da entidade de confiança ser totalmente segura os dados não necessitariam de estar cifrados, pois apenas teria acesso quem fosse autorizado e assim não existiria a carga computacional de cifrar e decifrar dados.

Por um lado a centralização da arquitetura na entidade de confiança traz algumas vantagens, mas por outro lado ela constituí uma grande desvantagem, isto é, uma arquitetura centralizada apenas é viável para ser utilizada como exemplo ou para sistemas pequenos. O facto de ter de guardar todos os dados e executar todo o trabalho computacional faz com que não seja escalável.

A arquitetura apresentada no artigo [39] também é descrita através de três entidades (figura 1.2), o dono da informação, o receptor da informação e uma autoridade de confiança. Mas ao contrário da arquitetura apresentada anteriormente, esta não está centrada na autoridade de confiança.

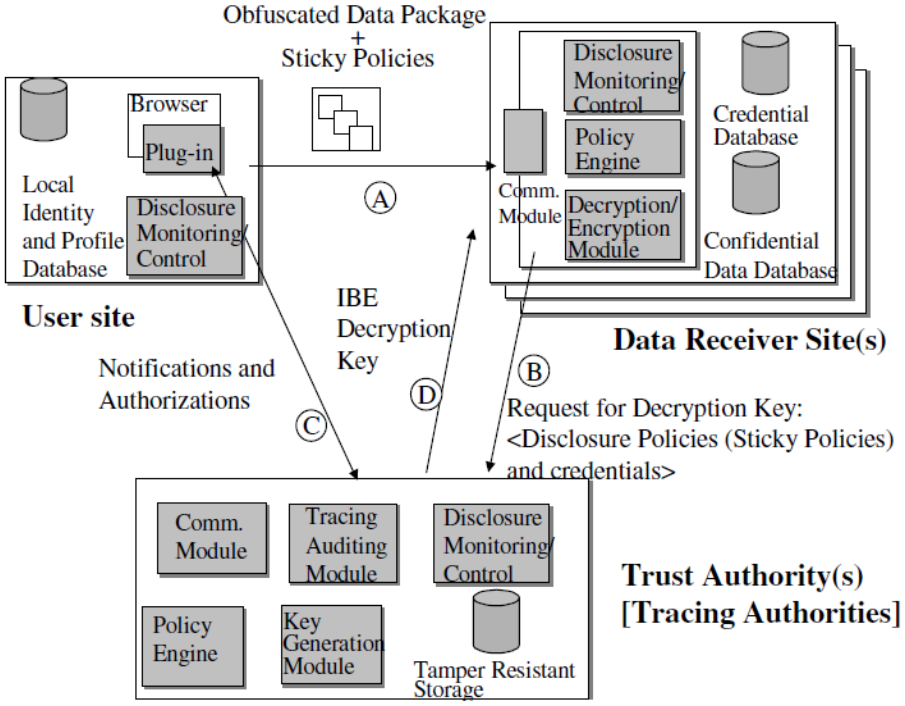


Figura 1.2 Exemplo de arquitetura de "sticky policies" presente em [39].

Nesta arquitetura o dono da informação vai enviar a informação diretamente para os receptores e não para a autoridade de confiança. Como as informações são enviadas diretamente para entidades desconhecidas elas têm que ir protegidas, assim o trabalho de cifrar as informações e escolher e mapear as políticas nas informações cifradas será realizado pelo dono das informações. Neste caso é muito importante que os mecanismos de criptografia utilizados permitam ligar as políticas aos dados cifrados para que as entidades desconhecidas não sejam capazes de alterar as políticas.

Na arquitetura anterior o trabalho de cifrar e decifrar as informações era realizado pela autoridade de confiança, o que centralizava o processamento. Nesta arquitetura o trabalho de cifrar é realizado pelo dono e o trabalho de decifrar é realizado pelos receptores. Os receptores para acederem às informações que receberam do dono terão que comunicar com a autoridade

de confiança para que esta lhes disponibilize a chave para decifrar as informações.

O trabalho da autoridade de confiança consiste em garantir que as políticas associadas às informações são cumpridas, para isso, ao receber o pedido da chave para decifrar, a autoridade vai avaliar se as políticas estão a ser respeitadas e dependendo das políticas associadas poderá ter que comunicar com o dono da informação para o notificar ou para ele dar autorização. Só depois deste trabalho ser efetuado é que a autoridade irá enviar a chave ao receptor.

No artigo [9] é proposta uma arquitetura para o cenário do projeto EnCoRe [27], neste cenário pretende-se que o dono das informações seja capaz de enviar as suas informações para uma organização em que não confie e essa organização possa guardar e enviar as informações para outras organizações.

Esta arquitetura (figura 1.3), tal como na apresentada em [39], quando o dono enviar as suas informações, estas já necessitam de ir protegidas e com as políticas associadas. Também existe uma autoridade de confiança que será responsável por verificar se as políticas associadas são cumpridas, e nesse caso enviará a chave para decifrar as informações.

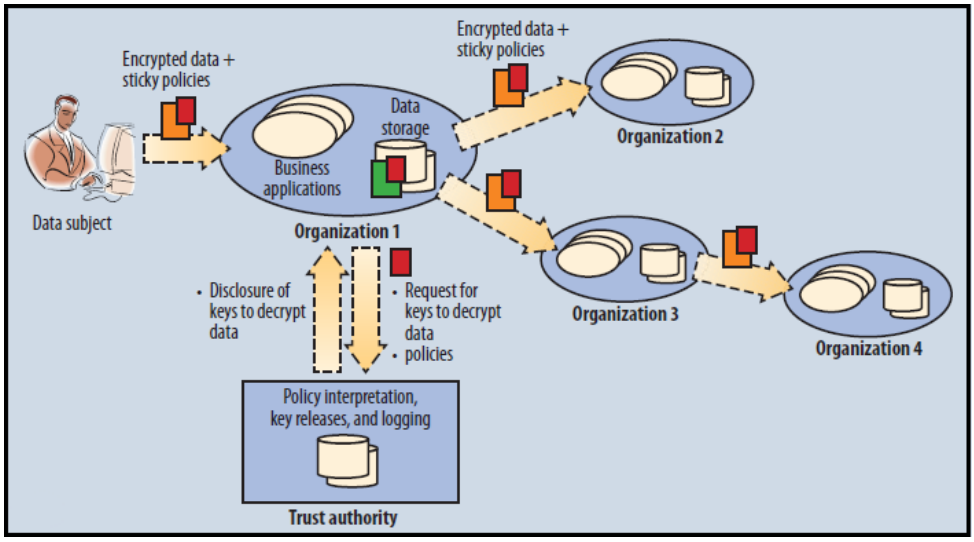


Figura 1.3 Esquema de "sticky policies". [46]

O utilizador envia as suas informações diretamente para uma organização. As organizações guardam as informações dos utilizadores e enviam as informações para outras organizações.

1.4 Objetivos

O presente trabalho tem como objetivo principal estudar o conceito de "sticky policies" e os algoritmos criptográficos associados, aplicando este conhecimento a um cenário prático.

Porém, existem ainda um conjunto de objetivos igualmente importantes, tais como:

- Estudar diferentes modelos de autorização e diferentes linguagens de especificação de políticas;
- Desenhar e implementar uma arquitetura de autorização com base nos conceitos estudados, demonstrando a sua aplicabilidade num cenário real;
- Abordar a temática de auditoria de acessos a ficheiros, procurando discutir a sua relevância em sistemas de partilha de ficheiros, comparando e descrevendo diferentes implementações de mecanismos de auditoria;
- Implementar no sistema de partilha de ficheiros criado neste projeto, mecanismos de auditoria de acessos a ficheiros.

1.5 Estrutura do Documento

No capítulo seguinte é apresentado o estado da arte. Nas diferentes secções são abordadas as temáticas de sistemas criptográficos para implementar "sticky policies", políticas de controlo de acessos, linguagens de políticas, e por fim, registos de auditoria.

Segue-se o capítulo onde é descrito o cenário de demonstração escolhido para o projeto, bem como o seu modelo de segurança e as principais opções tomadas relativamente à arquitetura de "sticky policies", política de controlo de acessos, sistema criptográfico e linguagem de políticas.

No capítulo de implementação é abordado o sistema implementado neste projeto, para tal é descrita a prova de conceito implementada e a arquitetura do sistema, de seguida são apresentados e discutidos os resultados obtidos.

O capítulo seguinte refere-se aos registos de auditoria do acesso a ficheiros, onde é descrito o mecanismo de registo de auditoria utilizado e a sua implementação. Por fim são apresentados os resultados do mesmo.

Por último, são enumerados alguns aspetos que podem dar continuidade a este projeto de modo a melhorar o sistema desenvolvido e tecem-se as considerações finais.

2. Estado da Arte

2.1 Sistemas Criptográficos Para Implementar "Sticky Policies"

Neste capítulo serão apresentados diferentes sistems criptográficos para a implementação de "sticky policies".

A grande diferença entre os diferentes sistemas será a técnica criptográfica utilizada. Serão abordadas quatro técnicas criptográficas, criptografia de chave pública, PKE (Public-Key Encryption), criptografia baseada na identidade, IBE (Identity-Based Encryption), criptografia baseada em atributos, ABE (Attribute-Based Encryption), e criptografia utilizando esquemas de proxy, PRE (Proxy Re-Encryption).

Os sistemas que serão apresentados de seguida são retirados do artigo [59], e será também utilizado o cenário proposto no mesmo artigo. O cenário consiste no seguinte:

Um utilizador deseja enviar informações para um outro utilizador, garantindo que o segundo cumpre um conjunto de políticas definidas pelo primeiro. Para isto o utilizador enviará as informações e políticas para autoridade de confiança que apenas enviará as informações para o segundo utilizar se este cumprir as políticas. Mesmo que um atacante consiga aceder ao conteúdo armazenado na autoridade de confiança, ele não será capaz de retirar informação desse conteúdo.

2.1.1 Criptografia de Chave Pública, PKE

Criptografia de chave pública, PKE (Public-Key Encryption), é criptografia assimétrica que utiliza duas chaves diferentes, uma para cifrar e outra para decifrar.

Das duas chaves utilizadas em PKE, uma delas será privada e a outra pública. A chave pública será utilizada para cifrar os dados, enquanto que a chave privada será utilizada para decifrar. Uma vez os dados cifrados com a chave pública, apenas utilizando a chave privada é que será possível voltar a aceder às informações, devido a esta propriedade a chave pública pode ser partilha livremente [41].

Como pode ser visto nos esquemas apresentados em [8], na criptografia PKE existem quatro algoritmos principais, "Setup", "KeyGen", "Encrypt" e "Decrypt".

"Setup" recebe um parâmetro de segurança e retorna um conjunto de parâmetros.

"KeyGen" recebe o conjunto de parâmetros e retorna um par de chaves (chave pública, chave privada).

"Encrypt" recebe a mensagem a ser cifrada, o conjunto de parâmetros e a chave pública e retorna a mensagem cifrada.

"Decrypt" recebe o conjunto de parâmetros, a chave privada e a mensagem cifrada, retorna a mensagem original.

No artigo [59] é apresentado um esquema de "sticky policies" utilizando PKE. O esquema proposto segue o cenário anteriormente apresentado. Neste esquema a autoridade de confiança possui um certificado de cada utilizador do sistema, para poder partilhar as chaves públicas associadas à chave privada de cada utilizador.

Na figura 2.1 podemos ver como tradicionalmente funciona o esquema PKE, onde a "Alice" para enviar a informação para o "Bob", pede à autoridade de confiança o seu certificado com o qual cifra a informação e envia o criptograma para o "Bob". Visto que o "Bob" possui a chave secreta associada ao seu certificado consegue decifrar a informação.

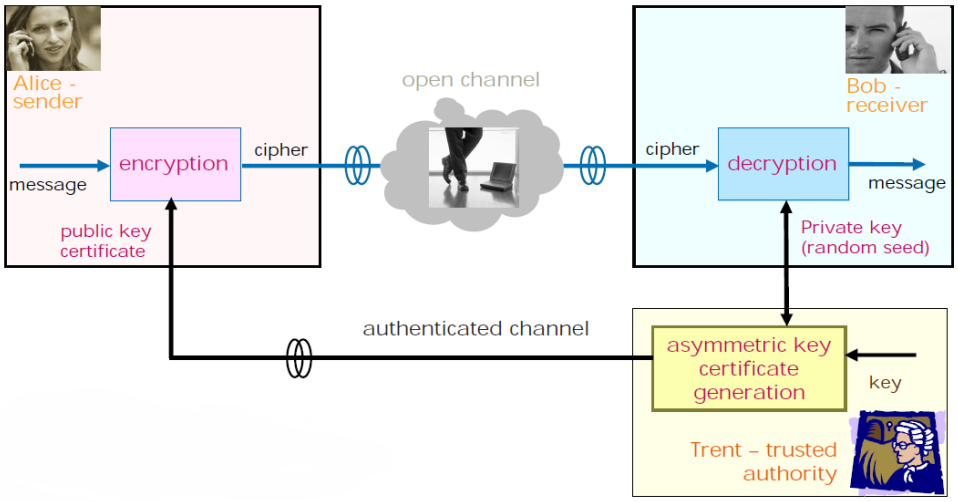


Figura 2.1 Esquema PKE. [15]

Para implementar o esquema de "sticky policies" utilizando PKE tem que se fazer algumas alterações ao esquema PKE. Para o dono enviar as suas informações para o utilizador "u" terá que pedir o certificado de "u" à autoridade de confiança. Assim que o certificado for obtido, o dono enviará as informações cifradas e o conjunto de políticas desejado para a autoridade de confiança. Depois para "u" receber as informações terá que fazer um pedido à autoridade de confiança que irá verificar se "u" cumpre as políticas associadas. Caso seja autorizado, a autoridade de confiança enviará as informações cifradas para "u". Por fim, como as informações foram cifradas com a chave pública presente no certificado de "u", este possui a chave privada para as decifrar.

2.1.2 Criptografia Baseada na Identidade, IBE

A criptografia baseada na identidade, IBE (Identity-Based Encryption), foi introduzida em 1984 por Adi Shamir [54]. O grande objetivo deste novo esquema criptográfico era tornar possível utilização da própria identidade como chave pública. Shamir, com este esquema, pretendia simplificar a gestão de certificados no sistema de email.

Por exemplo, com este esquema, figura 2.2, se a Alice pretendesse enviar um email cifrado para o Bob, usaria uma string, contendo a identidade do Bob como chave para cifrar e enviaria o

resultado para o Bob. Para decifrar a mensagem o Bob teria que contactar uma terceira entidade, o gerador de chaves privadas. Depois de o Bob se autenticar perante o gerador, este enviar-lhe-ia a chave privada. Com este esquema a Alice não necessita de receber o certificado do Bob para cifrar, e a chave privada pode ser gerada apenas no momento que se pretende decifrar a informação.

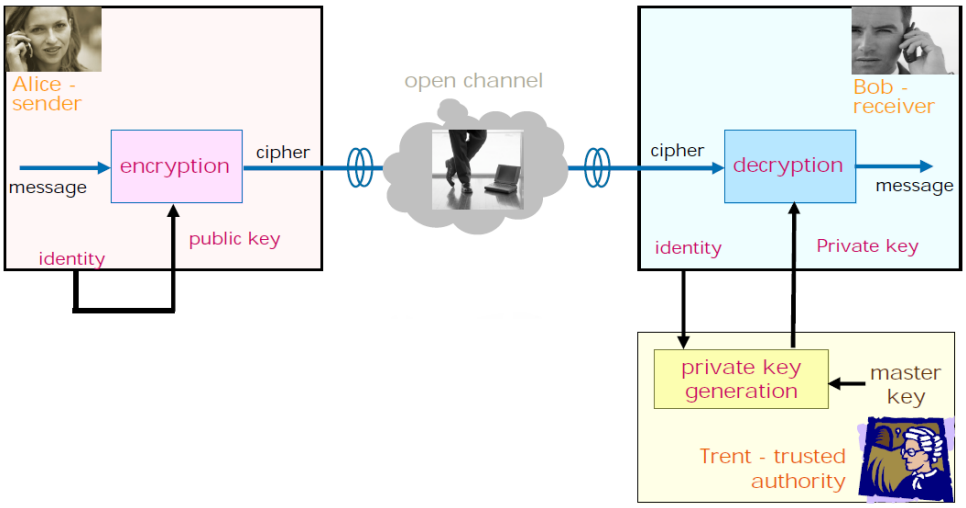


Figura 2.2 Esquema IBE. [15]

Uma das aplicações da utilização do esquema IBE é a revocação de chaves [13]. Num esquema de PKE é necessário emitir certificados com uma data de expiração, quando essa data expirar é necessário criar e difundir um novo certificado. Porém, num esquema de IBE, se for utilizada uma chave do tipo "identidade || ano", em que "ano" corresponde ao ano atual, terá que ser gerada uma nova chave para cada ano não sendo necessário a divulgação da chave pública, realizando-se apenas uma alteração no campo ano.

Num esquema de IBE existem quatro algoritmos, "Setup", "Extract", "Encrypt" e "Decrypt" [13].

"Setup" recebe um parâmetro de segurança e retorna um conjunto de parâmetros e uma chave-mestra.

"Extract" recebe o conjunto de parâmetros, a chave mestra e uma chave pública (uma string com a identificação do receptor) e retorna a chave privada.

"Encrypt" recebe a mensagem a ser cifrada, o conjunto de parâmetros e a chave pública e retorna a mensagem cifrada.

"Decrypt" recebe o conjunto de parâmetros, a chave pública, a chave privada e a mensagem cifrada e retorna a mensagem original.

O esquema criptográfico para implementar "sticky policies" utilizando o esquema IBE apresentado em [59] tem mais uma entidade que o esquema utilizado para PKE. Neste novo esquema existe o dono das informações, o receptor, uma autoridade de confiança que armazena as informações e verifica se as políticas são cumpridas, PEP, e uma outra autoridade de confiança que irá disponibilizar a chave privada, TTP.

Neste esquema o dono da informação cifra a informação com a identidade do receptor e envia a informação cifrada, as políticas definidas e a identidade do receptor para o PEP. De seguida, o receptor enviará um pedido ao PEP, este verificará se o receptor respeita as políticas, em caso afirmativo, envia-lhe as informações cifradas e a identidade utilizada para cifrar as informações. Obtendo isto, o receptor irá demonstrar ao TTP que a identidade é sua, depois de o TTP confirmar a autenticação enviará a chave privada ao receptor, com a qual decifrará as informações.

2.1.3 Criptografia Baseada em Atributos, ABE

Attribute-Based Encryption, ABE, ou criptografia baseada em atributos foi proposto por Sahai e Waters em [49], neste artigo foi proposto um esquema baseado no esquema IBE, FIBE (Fuzzy Identity-Based Encryption). Como foi referido anteriormente, num esquema IBE uma identidade é vista como um conjunto de atributos que formam uma string, para alguém conseguir decifrar a informação necessita de obter uma chave associada à mesma string. O objetivo deste novo esquema é permitir que uma identidade seja vista como um conjunto de atributos descritivos com isto, torna possível que a informação seja decifrada por alguém que tenha alguns dos atributos e não necessariamente todos.

Além de apresentar este novo esquema, FIBE, o artigo propõe duas aplicações práticas, um esquema IBE que utilize dados biométricos, e um esquema ABE. Neste esquema ABE proposto, quando alguém desejar cifrar informação funcionará de forma semelhante ao esquema IBE, ou seja, é cifrada utilizando um conjunto de atributos, mas além disto também é definido um nível mínimo de semelhança. Este nível vai ser utilizado no processo de decifrar a informação, pois ao contrário do IBE, a chave utilizada para decifrar não precisa de ter exatamente os mesmos atributos que os que foram utilizados para cifrar, apenas é necessário que a diferença não seja maior do que o nível mínimo definido.

Devido às limitações do ABE, foi introduzido um enriquecimento do ABE [24], o KP-ABE (Key-Policy Attibute-Based Encryption). No esquema ABE não é possível definir uma estrutura de acesso, o único meio de controlo para permitir decifrar a informação é o nível mínimo de semelhança entre os atributos associados à chave para decifrar e a os ligados à chave para decifrar. Deste modo, o KP-ABE surgiu para permitir definir estruturas de acesso à informação.

No KP-ABE a informação cifrada está associada a um conjunto de atributos, enquanto que as chaves privadas estão associadas a políticas, estas políticas correspondem a uma estrutura em árvore de acesso, onde as folhas correspondem a atributos e os nodos consistem nas operações lógicas "e" ou "ou". Uma chave privada apenas consegue decifrar a informação se os atributos associados a esta informação satisfizerem a política associada à chave.

No final do artigo [24] é proposto um esquema inverso ao KP-ABE, o CP-ABE (Ciphertext-Policy Attribute-Based Encryption). Este esquema funciona de forma inversa ao KP-ABE, ou seja, as chaves privadas estão associadas a um conjunto de atributos e a informação cifrada está associada a uma política.

A primeira construção de um esquema CP-ABE é fornecida no artigo [9]. Neste esquema existem quatro algoritmos fundamentais, "Setup", "Encrypt", "Key generation" e "Decrypt".

"Setup" assim como no esquema IBE, recebe um parâmetro de segurança e retorna um conjunto de parâmetros e uma chave-mestra.

"Encrypt" recebe a mensagem a ser cifrada, o conjunto de parâmetros e a estrutura de acesso (política), retorna a mensagem cifrada.

"Key generation" recebe a chave mestra e um conjunto de atributos, retorna a chave privada.

"Decrypt" recebe o conjunto de parâmetros, a chave privada e a mensagem cifrada, retorna a mensagem original.

Além deste quatros algoritmos principais é ainda disponibilizado um quinto algoritmo, o "Delegate".

"Delegate" recebe uma chave privada e um conjunto de atributos que está contido no conjunto de atributos da chave privada, retorna uma nova chave privada para o conjunto de atributos fornecido.

Ao contrário da utilização do esquema IBE, em que uma identidade corresponde a uma string que pode ser escolhida livremente pelos utilizadores, na utilização do esquema ABE, é necessário definir à priori o conjunto de atributos. É necessário que seja conhecido todo o universo de atributos para ser possível construir as estruturas de acesso com as propriedades impostas [9].

É proposto no artigo [59] um esquema criptográfico para implementar o conceito de "sticky policies" utilizando um esquema CP-ABE. Tal como na implementação utilizando o esquema IBE, esta também utiliza as mesmas entidades, o dono da informação, o receptor, o PEP e o TTP.

Neste esquema o dono cifra a informação com a estrutura de acesso desejada e envia para o PEP a informação cifrada e as políticas escolhidas. De seguida, o receptor enviará um pedido ao PEP, este verificará se o receptor respeita as políticas, em caso afirmativo, envia-lhe as informações cifradas. O TTP não necessita de estar online desde que tenha já disponibilizado as chaves privadas de cada utilizador. O receptor, com a sua chave privada, será capaz de decifrar a informação caso a estrutura de acesso seja satisfeita com os atributos da sua chave privada. Caso as políticas utilizadas fossem simples, do género de que quem tivessem certos papéis poderia aceder à informação, então não seria necessário utilizar o PEP avaliar as políticas, visto que esse controlo seria realizado no processo de decifragem, em que apenas quem estivesse autorizado seria capaz de decifrar.

2.1.4 Criptografia de Reencriptação por Proxy, PRE

O conceito de criptografia de reencriptação por proxy, PRE (Proxy Re-Encryption), foi introduzido por Mambo e Okamoto [37]. Este esquema tem como objetivo, permitir transformar um texto cifrado para uma chave privada num texto cifrado para outra chave privada. Por exemplo, se a Alice, o delegante, desejar encaminhar para o Bob, o delegado, emails que foram cifrados para si, irá utilizará uma proxy que irá converter o texto cifrado que apenas a Alice pode decifrar, para o texto cifrado que apenas o Bob pode decifrar.

Este conceito foi bem aceite, dando origem a vários esquemas e aplicações práticas, como é exemplo os esquemas de PRE propostos para aplicações de armazenamento de dados [5].

Nos esquemas PRE o delegante, com a sua chave privada e a chave pública do delegado, gera uma chave de delegação que envia para a proxy, com esta chave a proxy é capaz de converter um texto cifrado para a chave privada do delegante num texto cifrado para a chave privada do delegado. Com a chave de delegação a proxy é capaz de converter todos os textos cifrados com a chave pública do delegante. Tal acontecimento pode ser um problema se o delegante não desejar que a proxy seja capaz de converter tudo o que for cifrado com a sua chave pública. Para ultrapassar este constrangimento, Tang propôs o conceito de reencriptação por proxy baseada em tipos, TPRE (Type-based Proxy Re-Encryption) [60].

No artigo [60], Tang apresenta a sua implementação de um esquema TPRE. O objetivo deste esquema é introduzir tipos, ou seja, o delegante poderá dividir os seus textos cifrados em categorias. Com isto o delegante, ao gerar a chave de delegação, que está dependente também do tipo, sendo diferente para cada tipo, assim a proxy depois de obter a chave de delegação apenas vai ser capaz de converter textos cifrados para um tipo de textos cifrados. O tipo da mensagem corresponde a uma string escolhida pelo delegante.

Neste esquema a criptografia de chave pública, PKE, utilizada sofre algumas alterações. Relativamente aos algoritmos de PKE utilizados, o "Setup" e o "KeyGen" não sofrem modificações, funcionando como foi explicado anteriormente na secção do PKE, enquanto que o "Encrypt" e o "Decrypt" sofrem uma pequena alteração.

"Encrypt" recebe a mensagem a ser cifrada, o conjunto de parâmetros, a chave pública e o tipo da mensagem, retorna a mensagem cifrada.

"Decrypt" recebe o conjunto de parâmetros, a chave privada, a mensagem cifrada e o tipo da mensagem, retorna a mensagem original.

Além destes algoritmos também temos os algoritmos característicos do PRE, "Pextract" e "Preenc", em que "Pextract" é executado pelo delegante e o "Preenc" pela proxy.

"Pextract" recebe a chave pública e a privada do delegante, o tipo da mensagem e a chave pública do delegado, retorna a chave de delegação.

"Preenc" recebe o texto cifrado para o delegante, o tipo da mensagem e a chave de delegação, retorna o texto cifrado para o delegado.

Posteriormente foi apresentado um esquema alternativo ao TPRE [26], o esquema de re-encriptação por proxy baseado em tipos e identidades, TIPRE (Type-and-Identity-based Proxy Re-Encryption). Este em vez de utilizar o esquema PKE utiliza um esquema IBE, devido às vantagens que esta alteração traz pois, utilizando um esquema IBE deixa de ser necessário a partilha de certificados com as chaves públicas.

No esquema de TIPRE são utilizados os algoritmos do esquema IBE, "Setup", "Extract", "Encrypt" e "Decrypt", com exceção do "Encrypt", os algoritmos não sofrem alterações e são definidos como foram anteriormente no esquema IBE. Relativamente ao algoritmo "Encrypt" é adicionado o campo do tipo da mensagem.

"Encrypt" recebe a mensagem a ser cifrada, o conjunto de parâmetros e a chave pública e retorna a mensagem cifrada.

Os algoritmos "Pextract" e "Preenc" do TIPRE também sofrem alterações comparativamente com os do TPRE.

"Pextract" recebe a identidade e a chave privada do delegante, o tipo da mensagem e a identidade do delegado, retorna a chave de delegação.

"Preenc" recebe o texto cifrado para o delegante e a chave de delegação, retorna o texto cifrado para o delegado.

Tanto o "Decrypt" como o "Preenc" não necessitam de receber o tipo da mensagem pois o "Encrypt" coloca o tipo da mensagem no criptograma da mensagem cifrada.

É definido em [59], um esquema de implementação de "sticky policies" utilizando um esquema TPRE. Tal como na implementação utilizando o esquema IBE, esta também utiliza as mesmas entidades, o dono da informação, o receptor e o PEP.

Para o dono da informação enviar a informação para o receptor, começa por cifrar a informação com o tipo da mensagem e a sua chave pública. De seguida, o dono valida a chave pública do receptor, e gera a chave de delegação com a sua chave pública e a privada, o tipo da mensagem e a chave pública do receptor. Por fim, o receptor envia a informação cifrada, o conjunto de políticas, o tipo da mensagem e a chave de delegação para o PEP. A geração e o envio da chave de delegação para o PEP só é realizado um vez por cada par dono-receptor.

Para o receptor receber a informação tem que fazer um pedido ao PEP, este ao receber o pedido verificará se o receptor cumpre as políticas associadas. Em caso positivo reencripta a informação para que o receptor a possa decifrar e envia-lhe o novo texto cifrado.

Se em vez de utilizarmos o esquema TPRE, desejarmos utilizar o esquema TIPRE o dono da informação não necessita de validar a chave pública do receptor, utilizando a identidade do receptor. Por sua vez, o receptor ao receber a informação se ainda não possuir a chave privada contactará o TTP, que caso o receptor comprove ser quem diz ser, ser-lhe-á enviada a chave privada para essa identidade.

2.2 Políticas de controlo de acessos

Num sistema informático, controlar o acesso aos seus objetos é uma das questões fundamentais. Este controlo é definido como controlo de acessos. Para desenvolver um sistema de controlo de acessos é utilizada uma abordagem de múltiplas fases baseadas nos conceitos de política de segurança, modelo de segurança e mecanismo de segurança, que são definidos da seguinte forma [50]:

Política de segurança define as regras (alto nível) segundo as quais o controlo de acessos deve ser regulado.

Modelo de segurança fornece uma representação formal da política de segurança do controlo de acessos. Esta formalização permite que seja realizada a prova das propriedades na segurança fornecida pelo sistema de controlo de acessos.

Mecanismo de segurança define as funções baixo nível que implementam os controlos impostos pela política de segurança.

Neste capítulo serão analisadas quatro das várias políticas utilizadas no controlo de acessos, MAC (Mandatory Access Control), DAC (Discretionary Access Control), ABAC (Attribute-Based Access Control) e RBAC (Role-Based Access Control).

Tal como foi abordado no artigo [52], não existem políticas melhores que outras. Todas possuem diferentes características. Assim, deve-se escolher a mais adequada para cada cenário. Porém não são exclusivas, ou seja, em certos cenários pode ser adequado utilizar uma combinação de diferentes políticas.

2.2.1 Controlo de Acessos Discricionário, DAC

No Controlo de Acessos Discricionário, DAC (Discretionary Access Control), um utilizador individual, ou um programa a operar em seu nome, é capaz de especificar explicitamente os tipos de acesso que outros utilizadores podem realizar nas informações que estão sobre o seu controlo [19].

As primeiras formulações do DAC apareceram em [31] e [25], e a definição standard foi publicada pelo Departamento de Defesa dos Estados Unidos da América em [19]. Na definição apresentada, não é referido o conceito dono, assim, é possível haver implementações do DAC que não utilizem este conceito. A ideia central do DAC é de que um utilizador, normalmente o criador, tem autoridade discricionária sobre quem pode aceder ao objeto [45].

A grande maioria dos sistemas operativos utiliza DAC como o seu principal mecanismo de controlo de acessos. A flexibilidade das políticas discricionárias faz com que sejam adequadas para uma grande variedade de sistemas e aplicações, mas o DAC não fornece uma garantia real sobre o fluxo da informação num sistema [52]. Isto deve-se ao facto de que se um utilizador tem acesso a um ficheiro, mas não está autorizado a permitir que outros utilizadores tenham

acesso a esse mesmo ficheiro, ele pode simplesmente copiar as informações para outro ficheiro e partilha-lo livremente. Para resolver este problema pode-se fazer uma combinação com outras políticas de controlo de acessos como por exemplo, com o MAC.

Tal como foi referido em cima, o DAC tem a possibilidade de ser combinado com outras políticas de controlo de acessos. Mesmo que sendo combinado com outros mecanismos, pode ter diferentes implementações. Por exemplo, duas variações do DAC no que diz respeito a conceder acessos são descritos em [45]. No modo mais restrito apenas o dono pode conceder acesso a outros utilizadores, enquanto que no modo liberal podemos ter camadas em que utilizadores, que não o dono, também possam conceder acesso a outros utilizadores.

2.2.2 Controlo de Acessos Obrigatório, MAC

O Controlo de Acessos Obrigatório, MAC (Mandatory Access Control), é definido utilizando rótulos ligados aos sujeitos e aos objetos. Deste modo, será dada, ou não, autorização para o sujeito realizar uma certa operação no objeto, através da comparação entre o rótulo do sujeito e o do objeto. Os rótulos ligados aos utilizadores são denominados certificados de segurança (security clearance), enquanto que o rótulos ligados aos objetos são denominados por classificação de segurança (security classification).

Assim como o DAC, também o MAC foi definido pelo Departamento de Defesa dos Estados Unidos da América em [19], como sendo um meio de restringir o acesso a objetos baseado na sensibilidade (representada pelo rótulo) das informações contidas nos objetos e da autorização formal (certificado) dos sujeitos para aceder às informações de tal sensibilidade.

Estas políticas de controlo de acessos foram apresentados por Bell-LaPadula no artigo [7], onde o MAC é definido de forma semelhante ao que foi dito em cima, mas também são referidos três conceitos - utilizador, sujeito e objeto. Os objetos contêm ou recebem informação e estão associados a um nível de segurança (classificação). Enquanto que, como é referido no artigo de LaPadula e chamado à atenção no artigo [50], ao contrário do que acontece no DAC, no MAC existe uma clara distinção entre utilizador e sujeito. Utilizadores são humanos que podem aceder ao sistema, enquanto que os sujeitos são processos a operar em nome do utilizador. Cada utilizador tem associado a si um nível de segurança (certificado), mas pode fazer login no sistema com qualquer nível de segurança, desde que seja dominado pelo nível de segurança do seu certificado. Os sujeitos possuem o nível de segurança com o qual o utilizador efetuou login.

Ainda no artigo [7], são descritas duas propriedades que definem as políticas para os sujeitos poderem ler ou escrever em objectos, sendo estas a "propriedade simples de segurança" (simple security property) e a "propriedade-*" (lê-se "propriedade estrela") restrita (retricted *-property).

Propriedade simples de segurança significa que um sujeito S é autorizado a ler o objeto O apenas de o nível de segurança do sujeito L(S), dominar o nível de segurança do objeto L(O), isto é, $L(O) \preceq L(S)$.

Propriedade-* restrita significa que um sujeito S é autorizado a escrever num objeto O se o

nível de segurança do objeto L(O) for igual ao nível de segurança do sujeito L(S), isto é, L(O) = L(S).

A propriedade-* também pode ser definida como propriedade-* liberal. Neste caso, em vez de se impor que os níveis de segurança sejam iguais, apenas se impõe que o nível de segurança do objeto O domine o nível de segurança do sujeito, isto é, L(O) $\succeq$ L(S).

Para uma melhor compreensão do funcionamento dos níveis de segurança, será utilizado o exemplo apresentado no artigo [25]. Como se pode ver na Figura 2.3 neste sistema existem quatro níveis de segurança (TS, S, C e U). A Alice, que está certificada com nível S, deseja enviar informações para utilizadores do nível C e U porém, está impossibilitada pois os objetos escritos por si própria necessitam de estar classificados como S ou TS, assim sujeitos de nível C e U não podem ler os seus objetos. Mas a Alice como está certificada com nível S pode fazer login com o nível S, C ou U, assim é capaz de escrever objetos que possam ser acedidos pelos sujeitos com menor nível.

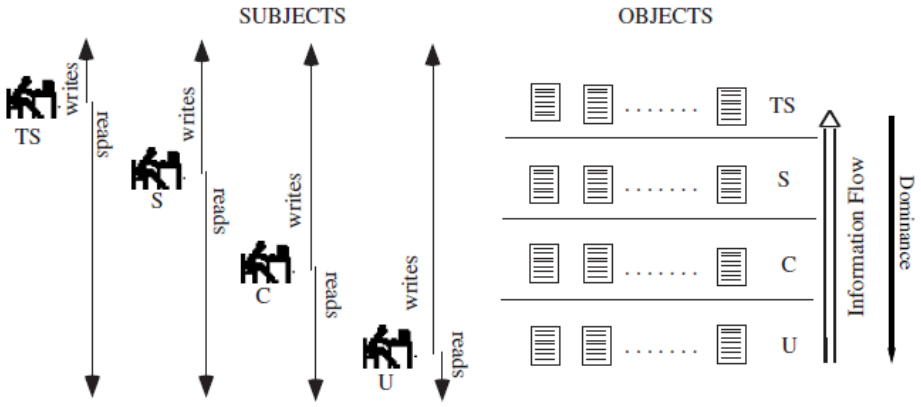


Figura 2.3 Fluxo de informações. [50]

A razão pela qual um sujeito de nível S não poder escrever um objeto para os de níveis C e U, mas um utilizador poder fazer login com níveis mais baixos, é que um utilizador ao ser certificado com nível S quer dizer que existe confiança no ser humano para que este não revele informações privadas para os níveis mais baixos. Mas quando o utilizador faz login com o nível S é o sujeito (o programa a executar em nome do utilizador) que realiza as operações sobre os objetos, e neste não é depositada confiança para não revelar informações ou níveis mais baixos. Com isto é impedido que um vírus no sistema consiga revelar informações privadas.

2.2.3 Controlo de Acessos Baseado em Papéis, RBAC

No controlo de acessos baseado em papéis, RBAC (Role-Based Access Control), para controlar os utilizadores que podem aceder aos ficheiros são utilizados papéis, isto é, os utilizadores estão associados a papéis e é sobre esses papéis que é realizado o controlo de acessos.

O RBAC é baseado em três entidades denominadas utilizador, papel e permissão. Um utilizador corresponde a um ser humano que utiliza o sistema, um papel está associada a um papel ou cargo de um trabalho dentro de uma organização e por sua vez, uma permissão é a aprovação da realização de uma operação sobre um ou mais ficheiros [45]. Ao contrário do que acontece no MAC e no DAC, as operações definidas nas permissões podem ser de alto nível e não correspondem apenas às operações de leitura, escrita e execução de ficheiros.

A grande motivação para a construção do RBAC é a necessidade de especificar e aplicar as políticas de segurança especificas das empresas mapeando de forma natural a estrutura de uma organização [23]. Para tal, são utilizados papéis como nas empresas, por exemplo, no sistema de um hospital poderão existir os papéis de médico, enfermeiro, clínico e farmacêutico [22].

No modelo geral RBAC, figura 2.4, apresentado em [51] são visíveis as três principais entidades do RBAC (utilizadores U, papéis R e permissões P), pode-se também ver que as relações entre utilizadores e papéis, papéis e permissões, papéis e papéis, são designadas por UA (user assignment ou atribuição de utilizadores), PA (permission assignment ou atribuição de permissões) e RH (role hierarchy ou hierarquia de papéis), respetivamente. UA, PA e RH são relações de muitos para muitos (representado pela dupla seta), isto significa que:

UA cada utilizador pode ter vários papéis e cada papel pode ser atribuído a vários utilizadores.

PA cada papel pode ter várias permissões e cada permissão pode ser atribuída a várias papéis.

RH cada papel pode herdar as permissões de vários papéis, esta relação é utilizada para se implementar uma hierarquia de papéis onde estes herdam as permissões de outros papéis.

Existem ainda as relações entre utilizadores, sessões e papéis. As sessões são comparáveis ao processo de login referido no MAC. Ao ser dito que um utilizador faz login com um nível de segurança, isto significa que o utilizador iniciou uma nova sessão com esse nível de segurança. No RBAC, numa sessão, existe um utilizador que, ao contrário do que acontece no MAC, pode ter várias papéis.

A abordagem do controlo de acessos baseado em papéis trás consigo vantagens como a gestão de autorização, a hierarquia de papéis, o mínimo privilégio, a separação de responsabilidades e a classe de objetos [52].

- Gestão de autorização - A utilização de RBAC permite ter uma melhor gestão de autorizações, pois esta gestão é dividida em duas tarefas separadas. Uma tarefa consiste em atribuir papéis a utilizadores e a outra tarefa consiste em atribuir permissões a papéis. Esta divisão permite que numa situação em que existe um funcionário de uma empresa que é promovido, fará com que as permissões no sistema sejam alteradas, em vez de ser necessário verificar todas as permissões desse utilizador ou seja, apenas é necessário alterar as suas papéis.

- Hierarquia de papéis - Como referido anteriormente, o RBAC permite que seja construída um hierarquia de papéis, assim os papéis poderão herdar permissões de outros papéis. Por

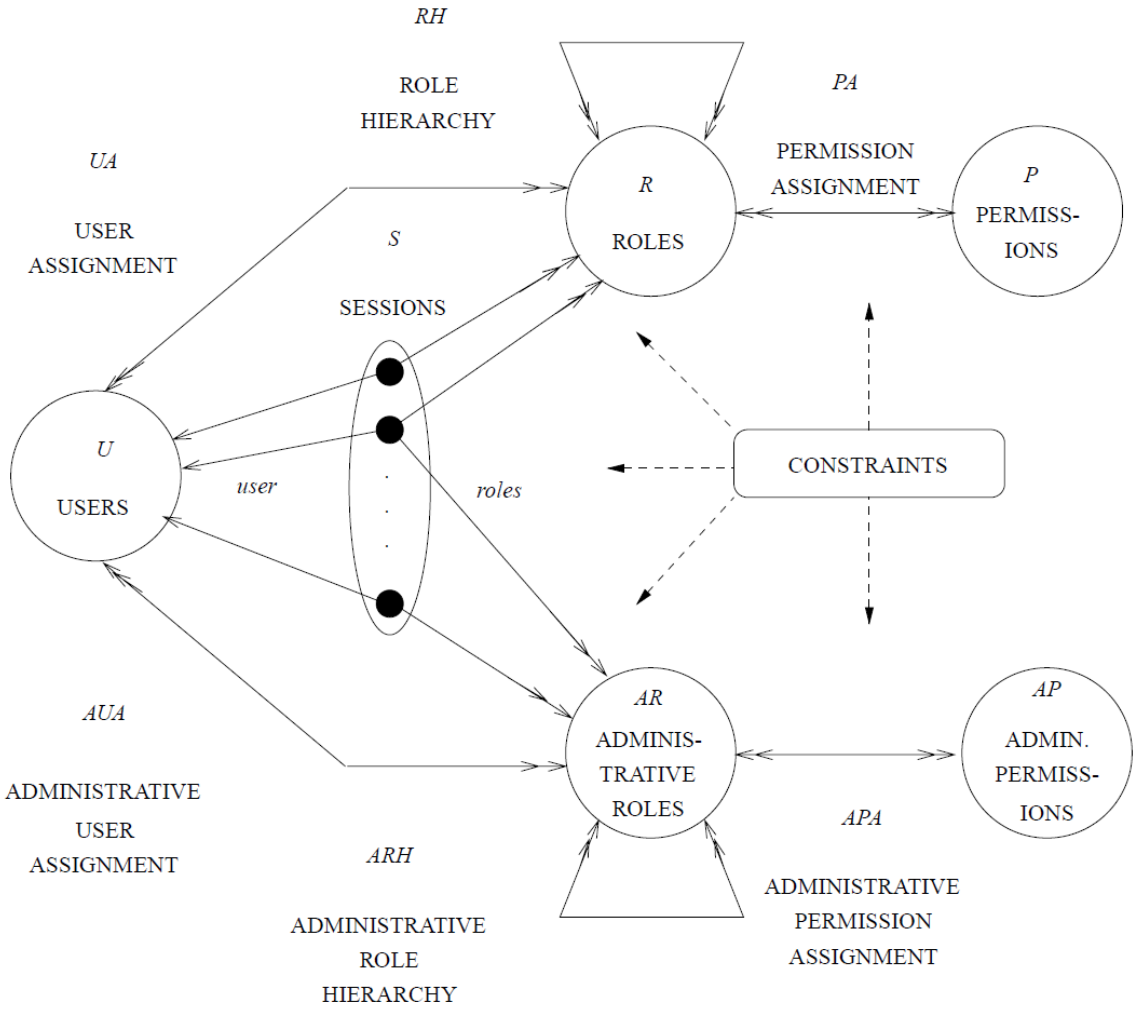


Figura 2.4 Modelo RBAC. [51]

exemplo, num sistema onde existissem quatro papéis (empregado, vendedor, mecânico e supervisor), onde todos os empregados tivessem um conjunto de permissões semelhantes, os vendedores tivessem um conjunto de permissões próprias, os mecânicos também tivessem um conjunto de permissões próprias e os supervisores tivessem todas as permissões de todos utilizadores dos sistema. Para representar o cenário do exemplo anterior, podemos utilizar a hierarquia de papéis presente na figura 2.5, onde as papéis de vendedor e de mecânico herdam as permissões de empregado, enquanto que supervisor herda as permissões de vendedor e de mecânico (supervisor ao herdar as permissões de vendedor e mecânico, também herda as permissões de empregado). A hierarquização de papéis simplifica a atribuição de permissões, pois no caso da papel de vendedor, apenas é necessário atribuir as permissões específicas do papel. Por sua vez, as permissões semelhantes a todos empregados são herdadas do papel empregado.

- Mínimo privilégio - A atribuição de permissões a papéis permite que os utilizadores efetuem login no sistema com o mínimo de permissões necessárias para poderem realizar o seu trabalho. Com isto, é reduzido o risco de o sistema ser danificado por atacantes que se façam passar por utilizadores legítimos.

- Separação de responsabilidades - O RBAC permite a separação de responsabilidades, isto é, para atividades críticas, as permissões para as concluir podem ser distribuídas por vários papéis. Assim um utilizador com um certo papel não poderá concluir a atividade sozinho. Isto faz com que no caso de existir um atacante fazendo-se passar por um utilizador legítimo não seja capaz de concluir uma atividade que pode por em risco a integridade do sistema.

- Classe de objetos - No RBAC é fornecida a classificação de utilizadores, mas também de objetos. Os objetos podem ser classificados segundo o seu tipo (ex.: cartas, manuais) ou segundo a sua área de aplicação (ex.: cartas comerciais, cartas de marketing). Assim, em vez de se especificar individualmente quais os objetos que um papel pode aceder, é especificado o tipo dos objetos. Por exemplo, para especificar quais os objetos que o papel de diretor comercial pode aceder é dada a permissão para aceder aos ficheiros com tipo comercial, em vez de os ter que especificar individualmente. Esta abordagem faz com que a administração de autorizações seja mais simples e melhor controlada.

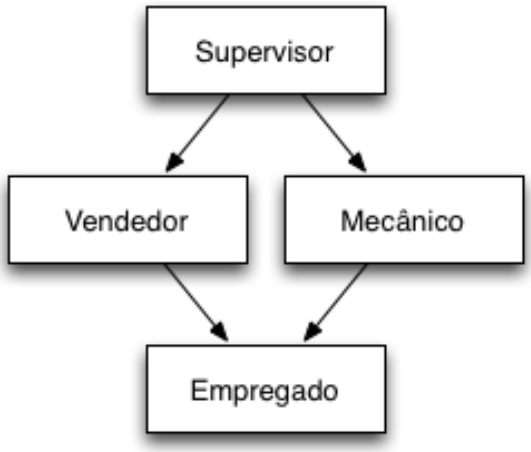


Figura 2.5 Exemplo de um hierarquia de papéis.

2.2.4 Controlo de Acessos Baseado em Atributos, ABAC

O controlo de acessos baseado em atributos, ABAC (Attribute-Based Access Control), ao contrário do que acontece nas outras políticas de controlo de acessos, não necessita de saber a

identidade do utilizador (no DAC existe uma clara ligação entre utilizador e objetos, no MAC são definidos níveis de segurança para os utilizadores e no RBAC os utilizadores estão ligados a funções), apenas é necessário saber alguns dos seus atributos. No ABAC as decisões de autorização são baseadas nos valores dos atributos necessários, estes atributos podem ser do utilizador, do objeto (ou recuros) ou do ambiente [64][32][34][11].

A facto de o ABAC não necessitar de saber a identidade é bastante útil em sistemas distribuídos como a internet, em que um utilizador pode ter acesso a um sistema mesmo sem estar registado nesse sistema. Por exemplo, numa loja de venda de livros online que oferece descontos a estudantes, para um utilizador ter acesso ao desconto, a loja apenas necessita de uma prova em como é estudante mas não precisando de saber quem é. [16].

No exemplo anterior é referido que um utilizador necessita de dar uma prova do atributo seu (ser estudante) sem precisar de dizer quem é, para isto foram desenvolvidos certificados de atributos [21]. Estes certificados podem ser utilizados para suportar sistemas ABAC e, tal como os certificados digitais utilizados em PKE (secção 2.1.1), também necessitam de ser assinados por uma autoridade de confiança para que tenham valor. Geralmente certificados de atributos costumam ter tempos de vida menor que os utilizados em PKE.

Como referido anteriormente, as decisões de autorização são realizadas através da comparação entre atributos. Num sistema ABAC, o responsável de segurança para criar as regras de autorização em vez de associar utilizadores a objetos (ou recurso), associa atributos de utilizadores a atributos de objetos e pode associar também atributos de ambiente. Estes diferentes tipos de atributos podem ser descritos da seguinte forma [66]:

Atributos de utilizador definem as características do utilizador, aquele faz o pedido para realizar uma ação sobre o objeto. Estes atributos podem ser diversos, como o nome, o seu cargo, os seus papéis, a idade.

Atributos de objeto , assim como os utilizadores, também os objetos requeridos por eles possuem atributos que os caracterizam. Estes atributos podem passar pelo seu título, assunto, data, autor, muitas vezes os atributos de um objeto podem ser retirados dos seus metadados.

Atributos de ambiente têm sido ignorados pelas restantes políticas de controlo de acessos, eles caraterizam o contexto ou o ambiente operacional ou técnico em que o pedido de acesso é realizado. Eles podem ser o tempo ou data atual, as atuais atividades de vírus, o nível de segurança da rede.

A primeira tentativa de fornecer uma “framework” uniforme de especificação e aplicação de um sistema ABAC foi apresentado em [12]. Onde propuseram uma “framework” para controlar o acesso a serviços e divulgação de informações num sistema de rede, aberto e distribuído como a internet.

Num sistema ABAC , como pode ser visto na figura 2.6, os utilizadores e os objetos são então representados por conjuntos de atributos. As permissões consistem em combinações entre

descritores de objetos (“object descriptor”) e uma operação que será realizada sobre os objetos descritos pelo descritor de objetos. Autorizações são definidas através da ligação entre descritores de utilizadores (“subject descriptors”) e permissões [47].

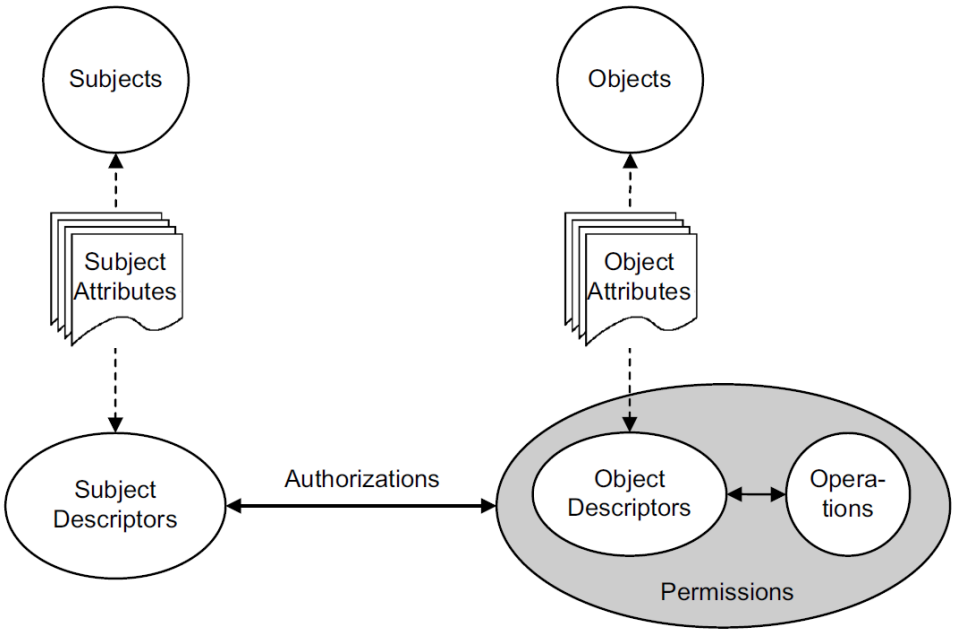


Figura 2.6 Modelo ABAC básico. [47]

Um descritor de objetos é um conjunto de atributos de objetos enquanto que um descritor de utilizadores corresponde a um conjunto de atributos de utilizadores.

Será agora apresentado um modelo ABAC estendido (figura 2.7). Neste modelo, tal como no apresentado anteriormente (figura 2.6), podemos verificar que os utilizadores (“subject”) e os objetos (“object”) podem ter vários atributos e que o descritor de utilizadores e o descritor de objetos estão ligados a diversos atributos de utilizador e de objeto, respetivamente. Além disto, neste modelo, ao contrário do anterior onde apresentava o conceito de permissões, as autorizações são definidas como ligações entre descritores de utilizadores e descritores de objetos com um tipo de acesso.

Neste modelo (figura 2.7) é introduzido o conceito de sessão e o conceito de condição. O conceito de sessão é o mesmo apresentado na secção 2.2.3 (RBAC), assim para cada sessão o utilizador pode utilizar o conjunto mínimo necessário de atributos para poder realizar as suas tarefas, utilizando assim o conceito de mínimo privilégio referido na secção 2.2.3 (RBAC). O conceito de condição surge neste modelo para permitir adicionar novas condições às autorizações, assim poderão ser utilizados outros atributos de utilizador ou de objeto que não estejam presentes nos descritores de utilizadores ou de objetos. Estas condições permitem também a utilização dos atributos de ambiente descritos anteriormente [48].

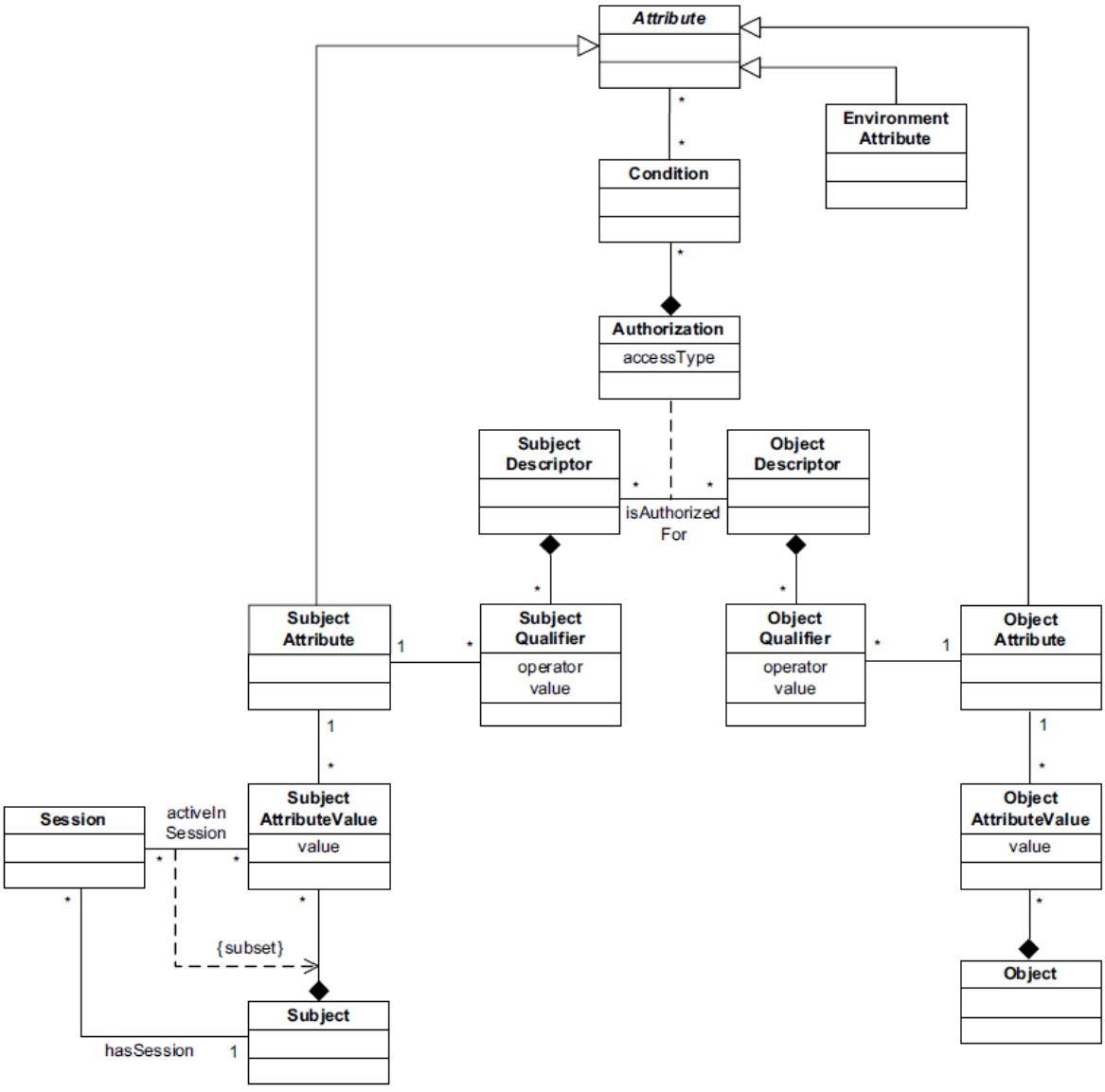


Figura 2.7 Modelo ABAC estendido. [48]

2.3 Linguagens de Políticas

As linguagens de políticas servem, como o nome indica, para ajudar o utilizador a definir políticas. No caso das "sticky policies", as políticas serão associadas às informações para fazer o seu controlo de acessos. Para um utilizador ter acesso às informações terá que respeitar as políticas associadas. Deste modo, é importante que a escolha de uma linguagem de políticas seja completa, mas simultaneamente compreensível para os utilizadores.

Uma das linguagens de políticas mais antiga é P3P [4], proposta pela IBM para definir políticas de segurança em empresas. Contudo esta linguagem tem dois grandes defeitos, um consiste na dificuldade de compreensão das suas políticas por parte de humanos, o segundo reside na não imposição das políticas, ou seja, as políticas P3P não necessitam de ser cumpridas, funcionam apenas como uma guia de boas práticas [6].

O trabalho realizado pela IBM em "sticky policies" levou-os a apresentar uma nova linguagem de políticas, a EPAL (Enterprise Privacy Authorization Language) [3].

Com o aumento do número de aplicações que utilizam XML como seu modelo de dados ou exportam os seus dados relacionais para formato XML, começaram a aparecer várias linguagens de políticas baseadas em XML [14]. Entre estas linguagens estão a XACML e a XrML, que serão descritas nas próximas secções.

2.3.1 XACML

A linguagem de políticas XACML (eXtensible Access Control Markup Language) permite especificar políticas de controlo de acessos em XML. O XACML define uma linguagem de políticas e a semântica para determinar o processamento dessas políticas [35][36][53][6].

O XACML foi proposto pelo comité da OASIS [42] com o objetivo de servir como uma interface única para políticas de várias aplicações e ambientes. A especificação standard mais recente (versão 2) do XACML foi publicada em [61]. Porém, a OASIS já lançou uma nova candidata (versão 3) a standard [62].

A especificação das políticas de controlo de acessos no XACML é baseada essencialmente em cinco componentes, atributos, funções, regras, políticas e conjuntos de políticas [38].

Atríbutos são caraterísticas de utilizadores, recursos, ações ou do ambiente que podem ser usados para definir restrições. No XACML não existe uma lista de atributos pré-definida, existe apenas um conjunto de tipos de dados que podem ser usados para criar atributos, alguns destes tipos de dados são "String", "Time", "Boolean" e "Double".

Funções são possíveis operadores que podem ser utilizados sobre os tipos de dados dos atributos, algumas das funções definidas no XACML são "String-Equal", "Greater-Then", "Date-Equal" e "String-Is-In".

Regras são um elemento básica de uma política. Uma regra define uma condição de autorização que existe isoladamente da política em que foi criada. Regras são compostas por um

alvo que identifica o conjunto de objetos/recursos sobre o qual a regra é criada, um **efeito** que é "permitir" ou "negar" e um **conjunto de condições** que representa as condições que a regra tem que cumprir para o resultado ser o efeito definido.

Políticas são a combinação de uma ou mais regras. Elas são compostas por um **alvo** (o mesmo definido nas regras), um **conjunto de regras** e um **algoritmo de combinação** que específica como deve ser computada o resultado da política no caso de haver regras com diferentes resultados, esses algoritmos são os que estão presentes na tabela 2.1.

Conjunto de Políticas representa as condições que devem ser cumpridas no caso da decisão de autorização considerar os requisitos de controlo de acessos de várias partes. Um conjunto de políticas é composto por um **alvo**, um **conjunto de políticas** e um **algoritmo de combinação** (table 2.1).

Tabela 2.1 Algoritmos de combinação de regras e políticas. [38]

Algoritmo de Combinação	Comportamento Esperado
Sobrepor Negação	O resultado combinado é negação se o resultado de pelo menos uma regra/política for de negação.
Sobrepor Permissão	O resultado combinado é permissão se o resultado de pelo menos uma regra/política for de permissão.
Primeira Aplicável	O resultado combinado é igual ao resultado da primeira regra/política.
Apenas um Aplicável	O resultado combinado corresponde ao resultado da única regra/política que se aplica ao pedido.

O XACML fornece também um método para avaliar e aplicar as políticas de controlo de acessos, além de quem faz o pedido de acesso ("Access requester") este método utiliza quatro entidades (figura 2.8), o ponto de informação de políticas (PIP, "Policy Information Point"), o ponto de decisão da política (PDP, "Policy Decision Point"), o ponto de aplicação da política (PEP, "Policy Enforcement Point") e o manipulador de contexto ("context handler") [36][67][38].

PIP é a entidade onde são armazenadas as políticas em XACML.

PDP é a entidade responsável por avaliar e tomar a decisão de autorização.

PEP é a entidade responsável por aplicar a decisão retornada pelo PDP.

Manipulador de contexto é a entidade responsável por converter os pedidos para o formato nativo do XACML e por converter a decisão em XACML para o formato nativo utilizado no PEP.

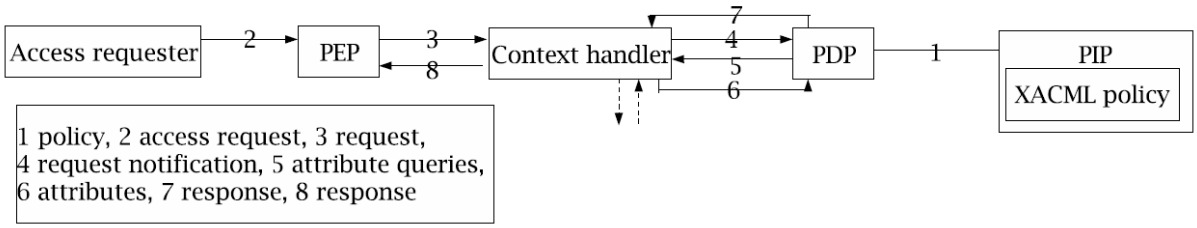


Figura 2.8 Método de avaliação e aplicação das políticas em XACML. [67]

Como se pode ver na imagem 2.8, o primeiro passo neste processo é o de definir as políticas em XACML que serão armazenadas no PIP, que se encontra em contacto com o PDP. Os pedidos de acessos são enviados ao PEP que os reencaminha para o manipulado de contexto, aqui os pedidos são convertidos para o formato XACML para que possa ser enviado para o PDP. Quando o pedido em formato XACML chega ao PDP, este comunica com o PIP e com o manipulador de contexto para obter todas as informações necessárias (políticas e atributos de que fez o pedido e dos recursos para os quais é realizado o pedido), com a informação disponível o PDP toma a decisão de "permitir" ou de "negar" e comunica-a ao manipulador de contexto. Por sua vez, o manipulador de contexto ao receber a decisão, converte-a para o formato utilizado no PEP e envia-lhe para que este aplique a decisão tomada.

2.3.2 XrML

XrML (eXtensible Rights Markup Languagem) é uma linguagem baseada em XML para gestão de direitos digitais (DRM) e condições, como a data de validade, associados a recursos digitais e serviços. A linguagem XrML fornece um método universal de especificar direitos e condições associados à utilização de conteúdo digital e de serviços [6][40][65].

O XrML foi inicialmente desenvolvido na Xerox PARC (Palo Alto Research Center). Na sua primeira versão (1.0) facilitava a criação de arquiteturas de gestão de direitos digitais de conteúdos e recurso, mas a versão 2.0 veio expandir as capacidades da linguagem permitindo estabelecer direitos e condições para o acesso a serviços web [65].

Na linguagem XrML existem três componentes principais, o identificador de regras, RIML (Rule Identification Markup Language), a estrutura das regras, RSML (Rule Structure Markup Language) e desencadeador de regras, RTML (Rule Triggering Markup Language), descritos de seguida [33][28].

RIML identifica as regras implicitamente expressas.

RSML representa a estrutura formal das regras.

RTML define as condições que desencadeiam determinadas regras.

Na figura 2.9 podemos ver o processo de aquisição de regras de páginas web na abordagem XrML.

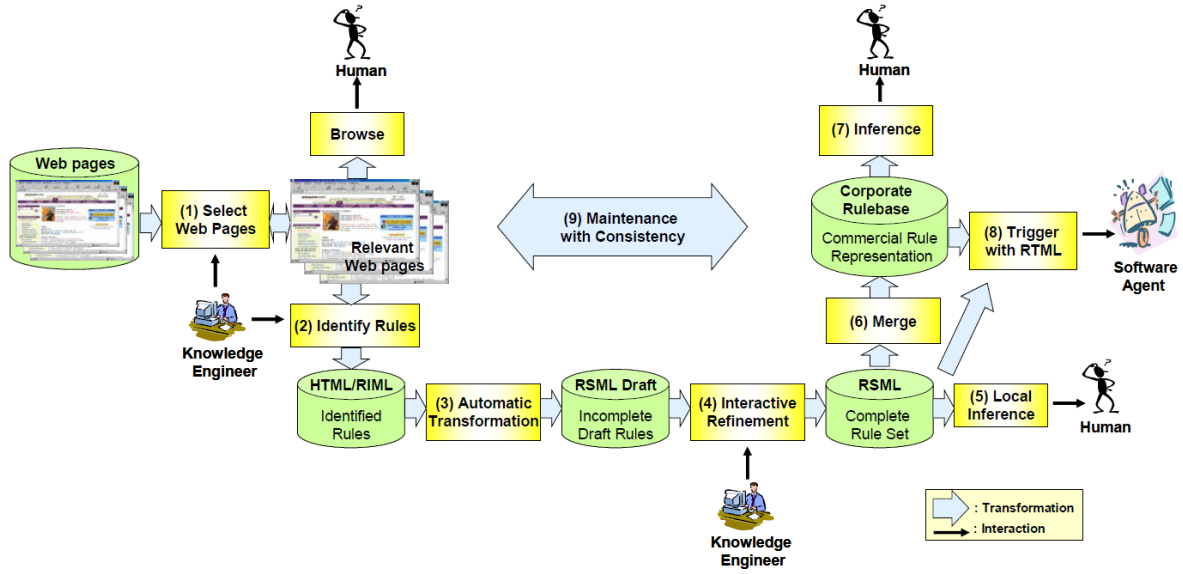


Figura 2.9 Processo de aquisição de regras em XrML. [28]

Os passos identificados com números na figura 2.9, são agora especificados [28]:

1. Selecionar páginas web relevantes – O responsável (“knowledge engineer”) por construir e manter a base das regras, seleciona as páginas web necessárias e relevantes para a base das regras.

2. Identificar as regras – O responsável identifica as regras das páginas web que, com isto o editor XrML gera declarações RIML embebidas em ficheiros HTML.

3. Transformação automática – As regras identificadas em HTML/RIML são transformadas em sintaxe RSML.

4. Refinamento interativo – As regras podem precisar de ser refinadas para ficarem completas.

5. Inferências locais – O conjunto de regras completas RSML podem ser utilizadas para se fazer inferências locais.

6. Junção – As regras RSML podem ser convertidas na sintaxe do sistema e englobadas às restantes regras da empresa.

7. Inferência – A base das regras pode ser utilizada para se realizar inferências.

8. Desencadear com RTML – Uma inferência realizada que vá contra as regras pode desencadear ações nos agentes do software.

9. Gestão com consistência – Qualquer alteração realizada nas páginas web ou na base das regras geradas pelo XrML, é detetada e são tomadas as devidas ações para manter o sistema consistente.

Um dos problemas do XrML é que os direitos têm de ser especificados antes de haver qualquer troca de informação. Assim, não é possível especificar inferências durante o tempo de execução [6]. Além disto, comparativamente com o XACML, o XrML é mais difícil de compreender sendo menos flexível, o XrML não é adequado para políticas de controlo de acessos complexas [40].

2.3.3 EPAL

Como foi referido anteriormente, a linguagem de políticas EPAL apareceu devido ao trabalho realizado pela IBM de "sticky policies", o objetivo do desenvolvimento da EPAL foi preencher a lacuna que existia entre a especificação de políticas e a sua aplicação encontrada no P3P [6].

A EPAL é utilizada para garantir que as informações estão seguras e que a sua utilização vai de acordo com as políticas de segurança da empresa. Com a EPAL a IBM introduziu uma forma automática de aplicar políticas de segurança em aplicações e sistemas [57].

O modelo utilizado pela EPAL é o mesmo utilizado pelo XACML (figura 2.10), assim como no XACML, também na EPAL temos as entidades PEP e PDP. Como explicado anteriormente, o PEP é responsável por receber os pedidos de acesso e aplicar as decisões tomadas, enquanto que o PDP toma as decisões com base nas políticas e nos atributos necessários.

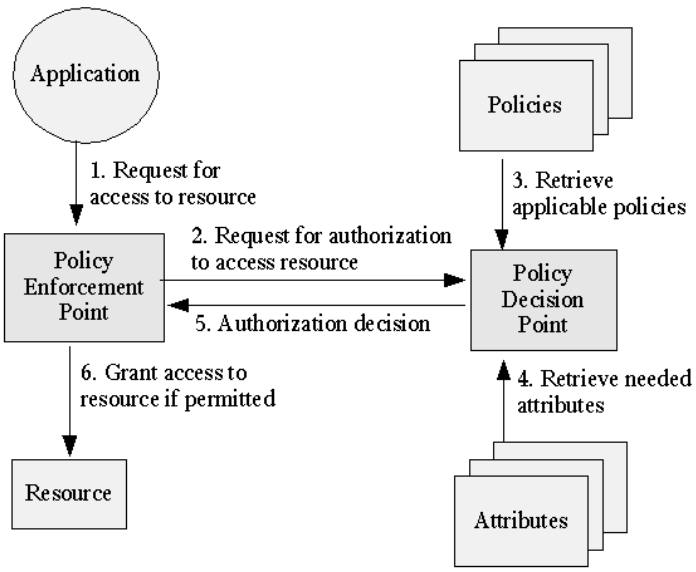


Figura 2.10 Modelo básico utilizado na EPAL e XACML. [2]

Ainda que partilhem o modelo de autorização (figura 2.10), a EPAL e o XACML são muito diferentes. Na comparação entre as duas a EPAL fica a perder, pois as diferenças entre as duas traduzem-se, na maioria dos casos, em menos funcionalidades na EPAL, e além disso estas diferenças fazem com escalabilidade das políticas de segurança seja mais complicada.

Algumas das funcionalidades presentes no XACML que a EPAL não possui são [1]:

- A capacidade de combinar resultados de várias políticas desenvolvidas de forma potencialmente independente.
- A capacidade de referenciar outras políticas, como parte de uma política.
- A capacidade de especificar as condições em vários utilizadores que podem ser envolvidos num pedido.
- A capacidade de separar os resultados para cada nó quando o acesso a um recurso hierárquico é solicitado.
- Políticas direcionadas para a manipulação de condições de erro e faltas atributos.
- Suporte para valores de atributos que são instâncias de elementos XML.
- Suporte para novos tipos de dados (incluindo X.500 Nomes Distintos, RFC822 nomes e endereços IP).

2.4 Registos de Auditoria

Atualmente, é cada vez mais acentuado o problema de que tudo o que é reproduzível é copiável, isto é, se uma pessoa consegue visualizar um ficheiro, então facilmente o consegue transmitir a qualquer outra pessoa, por exemplo, fotografando a informação que pretende transmitir.

Este problema põe em causa os sistemas de controlo de acessos, pois é possível que alguém não autorizado consiga aceder à informação através de alguém autorizado. Visto que nenhum sistema informático consegue controlar o comportamento dos utilizadores humanos, existem mecanismos para tentar combater estes comportamentos não autorizados.

Estes mecanismos são denominados registos de auditoria ou trilha de auditoria (em inglês "audit trail" ou "audit log"), sendo responsáveis pela monitorização do sistema, uma vez que, na existência de problemas torna-se possível a descoberta da sua causa.

No caso de um sistema de controlo de acessos a ficheiros, o registo de auditoria é responsável por fazer o registo de todos os acessos aos ficheiros. Se for detetado o acesso a um ficheiro por parte de um utilizador não autorizado, são utilizadas técnicas de auditoria sobre os registos dos acessos numa tentativa de descoberta do responsável.

Como é referido no artigo [10], no registo são armazenados os eventos e estatísticas que fornecem informação sobre o sistema e o seu desempenho, enquanto que a auditoria analisa estes registos para retirar informação sobre o sistema de forma clara e compreensível. No artigo [18] é apresentado um sistema de deteção de intrusões através da monitorização dos registos. Assim, os registos de auditoria num sistema de controlo de acessos são utilizados para compreender acontecimentos no sistema através de técnicas de auditoria.

Os registos de auditoria de um sistema podem ser feitos de forma centralizada ou distribuída. Ambas possuem vantagens e desvantagens, devendo ser analisado o contexto em que será aplicado para se poder aferir qual a que se aplica melhor ao contexto em questão.

Se o objetivo do registo de auditoria for o de aplicar técnicas de auditoria para controlar o comportamento de todo o sistema, a melhor escolha é a de ser centralizado visto que desta forma a informação se encontra toda no mesmo local.

Porém, se por outro lado, o objetivo for de controlar ou monitorizar partes destintas de um sistema, é mais adequado realizar os registos de auditoria de forma distribuída. Fazendo os registos de forma distribuída, conseguimos distribuir melhor a capacidade computacional para registrar e analisar os registos. Por exemplo, no caso de um sistema distribuído em três subsistemas independentes entre si, onde é pretendido controlar de forma independente, se utilizarmos registos de auditoria distribuídos, cada sistema terá apenas que analisar os seus registos, enquanto que, se fosse centralizado o sistema de análise, iria analisar todos os registos dos três subsistemas.

Visto que o registo de auditoria é utilizado para, posteriormente, ser possível utilizar técnicas de auditoria de modo a compreender o comportamento do sistema, torna-se imprescindível que um atacante não seja capaz de o alterar. A alteração do registo de auditoria significa que, ao serem aplicadas técnicas de auditoria, a informação que será retirada não será verdadeira,

induzindo em erro quem a consultar. Por exemplo, se um atacante realizar ações suspeitas mas de seguida conseguir eliminar estas ações do registo de auditoria, quando se tentar descobrir o causador ou não será possível ou até se poderá concluir que alguém que não o atacante.

Assim sendo, num mecanismo de registo de auditoria, independentemente de ser distribuído ou centralizado, é vital que seja garantida a integridade dos registos realizados para que o resultado da aplicação de técnicas de auditoria seja verdadeiro.

3 . Desenvolvimento do Projeto

3.1 Cenário de Demonstração

Pelo facto da presente dissertação de mestrado se encontrar integrada num estágio profissional na PT Inovação, o cenário de demonstração vai igualmente ao encontro das necessidades sentidas na empresa. Deste modo, o cenário de demonstração consta de um sistema de partilha de ficheiros num ambiente empresarial, em que o controlo de acessos é baseado nos papéis dos utilizadores.

O cenário apresentado consiste num ecossistema empresarial em que os utilizadores (trabalhadores da empresa) possuem um conjunto de papéis. O sistema de partilha de ficheiros entre os diversos utilizadores realiza o controlo de acessos baseado nos papéis dos mesmos.

Assim sendo, quando um utilizador desejar enviar ficheiros para o sistema terá que descriminar que papéis podem aceder a essas informações (definir a política de acessos). Quando um qualquer utilizador tentar aceder aos ficheiros, apenas terá acesso se o seu papel/papéis estiverem autorizados a aceder ao mesmo.

A informação será armazenada num sistema externo designado por serviço de armazenamento, pretende-se que seja possível utilizar qualquer fornecedor de computação na nuvem como serviço de armazenamento, isto é, tem que se considerar que o sistema de armazenamento não é seguro, podendo estar sujeito a ataques, ou alterações unilaterais de políticas do serviço.

Visto que não é garantida a segurança da informação no serviço de armazenamento a informação terá que ser protegida antes de ser armazenada, para isto será utilizada criptografia através da utilização do serviço criptográfico. Este deve ser transparente ao sistema, ou seja, o sistema não necessita de ter qualquer informação acerca do funcionamento do serviço criptográfico. Deste modo, será um serviço "REST" que poderá ser utilizado por vários sistemas em simultâneo.

Além disto, também é importante que seja viável a utilização do sistema em qualquer dispositivo, independentemente da sua capacidade computacional e autonomia.

Como se vê na figura 3.1, o utilizador "User 1" envia para o sistema um ficheiro e especifica que apenas pode ser acedido pelos utilizadores com papel "admin". O sistema ao receber o ficheiro em claro irá enviá-lo para o serviço criptográfico, de modo que este seja cifrado criando uma ligação entre o criptograma e as políticas definidas. De seguida o sistema envia o ficheiro cifrado e as políticas associadas para o serviço de armazenamento. Quando um utilizador desejar aceder ao ficheiro o sistema vai ao serviço de armazenamento e envia o ficheiro e a identificação do utilizador para o serviço criptográfico. Este apenas decifrará caso o utilizador esteja autorizado, como é representado na figura 3.1, em que o "User 2" que tem papel "eps" não é autorizado e o "User 3" como tomando o papel "admin" recebe o ficheiro decifrado. Visto que os ficheiros são enviados em claro, o canal de comunicação entre os utilizadores e o serviço terá de ser seguro.

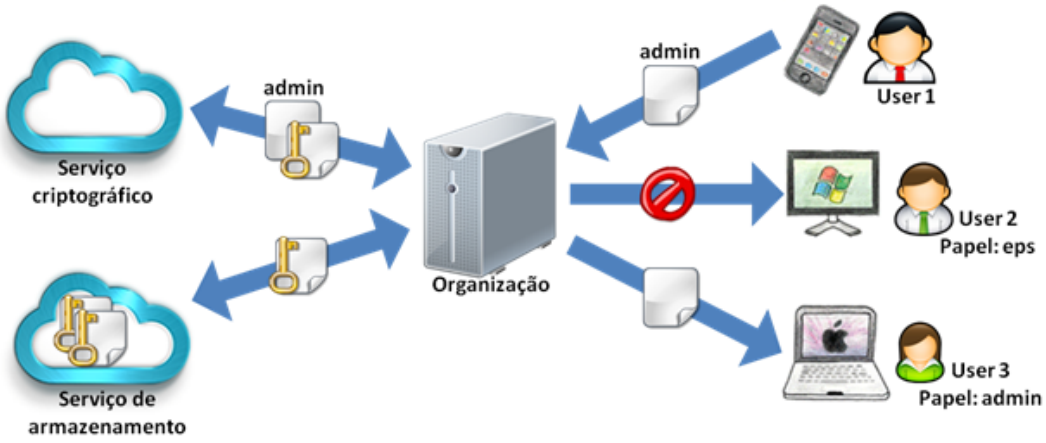


Figura 3.1 Esquema do cenário de demonstração.

3.2 Modelo de Segurança

Nesta secção será descrito o modelo de segurança do sistema, ou seja, o grau de confiança que é necessário possuir em cada um dos componentes para ser possível confiar no sistema.

Visto que o serviço criptográfico responde a todos os pedidos realizados pela organização, acredita-se que esta autentica os utilizadores, para que o serviço criptográfico seja capaz de realizar corretamente o controlo de acessos aos ficheiros.

No envio dos ficheiros dos utilizadores para a organização e da organização para o serviço criptográfico e utilizadores, os ficheiros são enviados em claro, assim acredita-se que os canais de comunicação entre a organização e os utilizadores e entre a organização e o serviço criptográfico são seguros.

Além dos canais de comunicação, também terá que se confiar na organização e no serviço criptográfico, visto que estes terão acesso à informação presente nos ficheiros.

Relativamente ao serviço de armazenamento, apenas se acredita que disponibiliza algumas funcionalidades caraterísticas do armazenamento na nuvem, nomeadamente a disponibilidade constante dos ficheiros e a capacidade de repor o estado dos ficheiros para versões antigas. Assim não é necessário confiar no serviço de armazenamento para garantir a integridade dos ficheiros.

3.3 Opções Tomadas

São agora abordadas as opções tomadas para a implementação do sistema adequado ao cenário de demonstração.

Assim sendo, descrever-se-á a opção tomada e os motivos da escolha relativamente à arquitetura de "sticky polices"; ao sistema criptográfico para implementar a arquitetura de "sticky policies"; qual a política de controlo de acessos que será utilizada pelo sistema e qual a linguagem de políticas que utilizada.

3.3.1 Arquitetura de "Sticky Polices"

Como é descrito no cenário de demonstração, os ficheiros e as políticas de controlo de acessos associadas estarão armazenados num local inseguro, podendo existir acessos indevidos aos ficheiros. Por esta razão, a informação necessita de ser cifrada antes de armazenada, tal como acontece nos exemplos da secção 1.3. Ao contrário do que acontece no primeiro exemplo e tal como acontece nos seguintes, as políticas necessitam de estar fortemente ligadas ao ficheiro cifrado.

Para garantir esta dependência da informação cifrada às políticas, a informação será cifrada utilizando as políticas definidas pelo dono como chave pública. Como acontece nos exemplos da secção 1.3, para um utilizador aceder às informações cifradas terá que comunicar com um agente de confiança, que aplicará as políticas associadas e de seguida gerará a chave privada, caso as políticas autorizem.

Relativamente ao envio dos ficheiros para o sistema, em todos os exemplos da secção 1.3 o utilizador cifra os ficheiros antes de os enviar mas, como se pretende no cenário de demonstração, que o sistema possa ser utilizado por aparelhos com muito ou pouco poder computacional, tal não pode acontecer. Se o utilizador estiver responsável por cifrar a informação e pretender utilizar o seu "smartphone" terá problemas de desempenho e duração da bateria. Por este motivo, os dados serão enviados para um agente de confiança através de um canal seguro, para que sejam cifrados e posteriormente armazenados.

3.3.2 Política de Controlo de Acessos

Na secção 2.2 são analisadas quatro das mais utilizadas políticas de controlo de acessos. Para a implementação deste projeto foi selecionada uma política de controlo de acessos baseado em papéis (RBAC, secção 2.2.3). Esta escolha deve-se ao facto de ser um pressuposto do cenário de demonstração.

Apesar de o cenário de demonstração especificar o tipo de políticas de controlo de acessos que devem ser utilizados, está é uma escolha adequada, visto que o cenário representa um ambiente empresarial e as políticas RBAC são adequadas para este tipo de ambiente, como é explicado na secção 2.2.3.

3.3.3 Sistema Criptográfico

Na secção 2.1 são descritas um conjunto de técnicas criptográficas, PKE (secção 2.1.1), IBE (secção 2.1.2), ABE (secção 2.1.3), PRE (secção 2.1.4), que podem ser utilizadas para implementar uma arquitetura de "sticky policies", sendo que algumas têm mais vantagens que outras.

Além das técnicas de criptografia assimétrica pode também ser utilizada criptografia simétrica, visto que confiamos que o serviço criptográfico é seguro poderia ser utilizada criptografia simétrica, sendo que o serviço iria guardar as chaves. Apesar da criptografia simétrica ter tempos de execução muito inferiores aos tempos de execução da criptografia assimétrica, a sua utilização torna o serviço criptográfico centralizado nas chaves simétricas utilizadas. Visto que um ponto essencial deste projeto é desenvolver um sistema que possa ser visto como uma nuvem com grande escalabilidade, serão aproveitados os baixos tempos de execução da criptografia simétrica para cifrar/decifrar a grande quantidade de informação, enquanto que a criptografia assimétrica será apenas utilizada para cifrar/decifrar chave simétrica, o que não tem grande influência no desempenho do sistema.

Como foi referido anteriormente, pretende-se utilizar uma destas técnicas criptográficas para ligar fortemente as políticas ao ficheiro associado, para isso iremos utilizar as políticas para cifrar o ficheiro. Desta forma, estamos a criar uma grande ligação entre políticas e ficheiro, visto que as políticas são a chave pública do criptograma. Sendo assim, a escolha terá que recair entre utilizar IBE ou ABE. Para utilizarmos IBE de forma a ligar as políticas aos ficheiros, pode ser usado um "hash" das políticas, isto é, visto que no esquema IBE pode ser utilizada qualquer "string" como chave pública, podemos gerar um "hash" do ficheiro que contem as políticas e utilizar esse "hash" como chave pública para o esquema IBE. Desta forma as políticas ficam ligadas ao criptograma, visto que para gerar a chave privada de modo a decifrar, é necessário o "hash" utilizado que só é possível gerar com as políticas originalmente utilizadas. Se estas forem alteradas por um atacante, a chave privada gerada para o "hash" destas novas políticas não conseguirão decifrar o ficheiro.

Visto que no cenário de demonstração apenas é referido que se pretende realizar o controlo de acessos com base nos papéis, neste caso também é possível utilizar ABE. Para tal, seria necessário usar os papéis como sendo os atributos do esquema ABE. Neste esquema, as políticas de controlo de acessos também estão ligadas ao criptograma mas, contrariamente ao IBE, no esquema ABE as políticas são verificadas e aplicadas diretamente no processo de decifrar assim, não seria necessário criar um ponto para verificar e aplicar as políticas, tal como acontece no IBE.

No esquema IBE é necessário utilizar uma linguagem de políticas para criar e verificar as políticas, ao contrário do que acontece no ABE que tem este processo embebido no processo de decifrar. Porém, o ABE tem algumas desvantagens relativamente ao IBE. Visto que o universo de atributos utilizados pelo ABE necessita de ser definido no início, os papéis utilizados por um sistema não podem ser alterados, nem adicionados novos papéis. Para se alterar ou adicionar papéis, teria que se voltar a gerar todos os parâmetros do ABE, decifrar toda a informação e tonar a cifrar tudo com os novos parâmetros.

Ainda assim, como o IBE está totalmente separado das políticas, o sistema para cifrar e decifrar pode ser utilizado em qualquer sistema, independentemente das políticas de controlo de acessos que utilize. O que não acontece com o ABE, que para diferentes políticas de controlo de acessos, teria de se adaptar aos seus mecanismos sendo possível implementá-los através da utilização dos atributos do ABE.

Outro ponto em que a utilização do esquema IBE, neste contexto, pode ser vantajosa relativamente ao esquema ABE, é a geração das chaves privadas. No esquema IBE, é gerada uma chave privada para cada chave pública utilizada, isto é, através da chave pública que foi utilizada para cifrar é possível gerar a chave privada que apenas é capaz de decifrar criptogramas cifrados com essa chave pública. No caso de introduzirmos um elemento de identificação do ficheiro na política (por exemplo o nome do ficheiro), para diferentes ficheiros, iremos ter diferentes políticas, fazendo com que o "hash" gerado a partir da política também seja diferente. Deste modo, se utilizarmos este "hash" como chave pública, a chave privada que permitirá decifrar um ficheiro será diferente para diferentes ficheiros. Enquanto que no esquema ABE a informação é cifrada utilizando uma política de acesso, depois é gerada uma chave privada para cada utilizador. Essa chave privada irá conter os atributos do utilizador e este será capaz de decifrar um documento se os seus atributos cumprirem a política de acessos utilizada no momento em que foi cifrado. Por outro lado, utilizando o esquema ABE numa na arquitetura de "sticky policies", se uma chave privada for comprometida, o atacante irá conseguir decifrar todos os documentos em que os atributos dessa chave estejam autorizados. Por outro lado, se utilizarmos o esquema IBE e se uma chave privada for comprometida, o atacante apenas conseguirá decifrar o ficheiro para a qual foi gerada.

Por estes factores, será utilizado um esquema IBE neste projeto, para que as políticas de controlo de acessos e o processo de cifragem sejam transparentes entre si, permitindo uma melhor separação de papéis e uma melhor distribuição da capacidade de processamento. Outro fator relevante para esta escolha é a flexibilidade com que se podem adicionar novos papéis e a capacidade de a mesma estruturar de cifragem de um sistema poder ser utilizada por diferentes sistemas, em que nos diferentes sistemas podem ser utilizadas diferentes políticas de controlo de acessos, existindo assim uma transparência entre as políticas de controlo de acessos e o processo de cifragem.

3.3.4 Linguagem de Políticas

Relativamente à especificação das políticas de controlo de acessos é possível utilizar mecanismos que não necessitem de uma linguagem de especificação de políticas, por exemplo, a utilização do esquema ABE, tal como foi referido anteriormente.

Como foi analisado na secção 2.3, a linguagem XrML (secção 2.3.2) adequa-se a cenários de gestão de direitos digitais, o que não é o caso, por essa razão não é considerada uma boa escolha para o cenário de demonstração do projeto. Relativamente à EPAL (secção 2.3.3) e ao XACML (secção 2.3.1), as duas poderiam ser escolhidas para especificar as políticas RBAC utilizadas no projeto, mas tal como é referido na secção 2.3.3, na comparação entre as duas

linguagens a EPAL fica em desvantagem, deste modo, optou-se pela utilização de XACML para especificação das políticas de controlo de acessos.

Sabe-se que a utilização de XACML, para especificar políticas simples trás consigo um "overhead" desnecessário, apesar desta desvantagem acredita-se que esta é a escolha mais adequada para que o projeto seja mais escalável e transparente às políticas utilizadas. Isto é, o XACML pode ser utilizado para especificar diferentes políticas de controlo de acessos, assim os mecanismos criados para avaliar e aplicar as políticas são os mesmos independentemente das políticas que esteja a ser utilizado no sistema.

4. Implementação

4.1 Prova de Conceito

Como prova de conceito será implementado um sistema de partilha de ficheiros que respeite o cenário de demonstração (secção 3.1) seguido.

Como o foco deste projeto é a arquitetura de "sticky polices", onde se pretende cifrar a informação e realizar o controlo de acesso, não será implementado o serviço de armazenamento (figura 3.1), sendo que os ficheiros serão guardados localmente no serviço. Apesar de não se implementar o serviço de armazenamento, onde é utilizado um fornecedor de computação na nuvem para armazenar a informação, o sistema deve estar preparado para isso, tal como é indicado no cenário de demonstração, em que é imposto que a informação possa ser armazenada num local inseguro.

Visto que a autenticação dos utilizadores também não é um dos pontos abordados neste projeto, esta será realizada por um ponto de autenticação de utilizadores de um produto da PT Inovação denominado IAM (Identity and Access Manager). O IAM é um produto que atua no domínio da gestão de identidades, autenticação e autorização, proporcionando um conjunto de funcionalidades que permite responder a cenários de autenticação centralizada, "single sign-on", autorização granular e gestão de identidades (contas, passwords, atributos) dos utilizadores. O IAM utiliza protocolos standard para comunicar com os serviços que lhe delegam o processo de autenticação. Funcionalmente, todos os serviços do IAM estão expostos por serviços oferecendo, dessa forma, a possibilidade de sistemas externos poderem interagir de forma simples com o IAM.

Assim sendo, será implementado o serviço criptográfico (figura 3.1) que através da utilização de servidores "REST", será capaz de cifrar, decifrar e controlar os acessos aos ficheiros. Para representar o sistema da empresa, será implementada uma aplicação web que utilizará o IAM para autenticar os utilizadores e comunicará com o serviço criptográfico, como pode ser visto na figura 4.1.

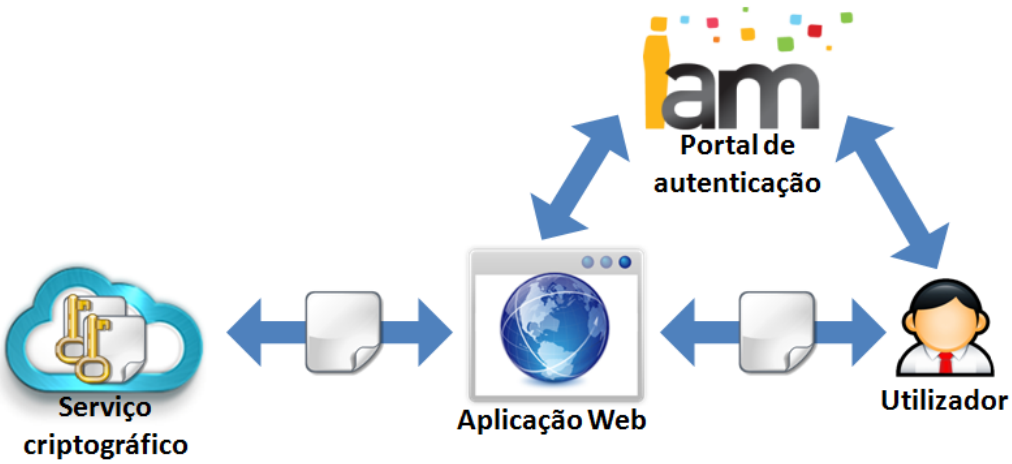


Figura 4.1 Esquema da prova de conceito.

4.2 Descrição da Arquitetura

Nesta secção será descrita a arquitetura utilizada para implementar o sistema narrado na secção 4.1. As principais opções tomadas para a implementação foram descritas anteriormente na secção 3.3, seguindo-se agora a descrição da arquitetura implementada neste projeto.

Com o objetivo de melhorar a performance deste serviço criptográfico, o trabalho foi distribuído por quatro servidores, tal como está representado na figura 4.2. Assim, este serviço é composto por quatro serviços web, o "Service", o "Keyman", o "Encryptor" e o "Decryptor". O "Service" é o responsável pelas políticas de controlo de acesso, o "Keyman" é o responsável por partilhar os parâmetros públicos e gerar as chaves privadas do esquema IBE, o "Encryptor" e o "Decryptor" são responsáveis por cifrar e decifrar a informação, respetivamente.

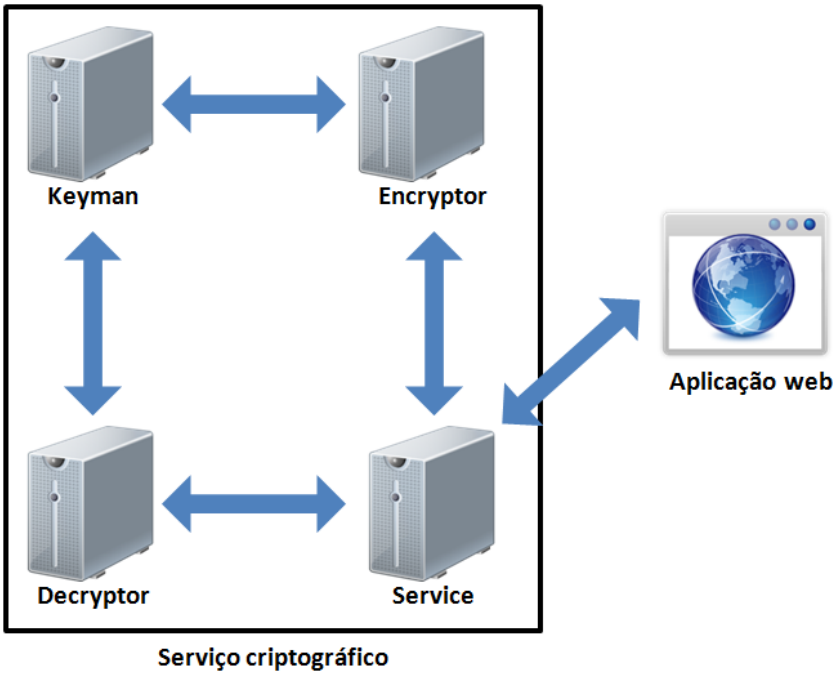


Figura 4.2 Esquema da arquitetura do serviço criptográfico.

Na figura estão representadas as ligações que existem entre os diferentes serviços. O "Service" comunica com a aplicação web, com o "Encryptor" e com o "Decryptor". Por sua vez, o "Encryptor" e o "Decryptor" comunicam também com o "Keyman".

Para implementar estes serviços REST em Java foi utilizada a biblioteca Jersey [44].

Nesta secção serão então apresentados diagramas de sequência simples das iterações que a aplicação web pode ter como o serviço criptográfico, de seguida, serão descritos mais pormenorizadamente o funcionamento dos diferentes serviços que compõem o serviço criptográfico ("Service", "Encryptor", "Decryptor" e "Keyman").

4.2.1 Diagramas de Sequência

A aplicação web pode fazer dois tipos de pedidos ao serviço criptográfico, upload ou download de um ficheiro. Para se perceber melhor como são realizadas as interações entre os diferentes serviços que compõe o serviço criptográfico serão agora apresentados dois diagramas de sequência (o de upload e o de download de um ficheiro).

No diagrama de sequência de upload de um ficheiro (figura 4.3) podemos ver quais são as interações necessárias entre os diferentes serviço para se realizar a tarefa.

A aplicação web ("Web App") começa por enviar para o "Service" o ficheiro ("file"), a identidade do utilizador ("id") que está a fazer upload do ficheiro e os papéis ("roles") que estão autorizados a aceder ao ficheiro. Com estas informações é criada a política de controlo de acessos ("pols") e é enviada juntamente com o ficheiro para o "Encryptor". De seguida o "Encryptor" comunica com o "KeyMan" para que este lhe envie os parâmetros públicos ("params") utilizados no esquema IBE. O ficheiro é cifrado através do esquema IBE utilizando os parâmetros fornecidos pelo "KeyMan" e o "hash" das políticas ("hashPols") gerado como chave pública, obtendo assim o criptograma ("encFile") que é enviado e guardado no "Service".

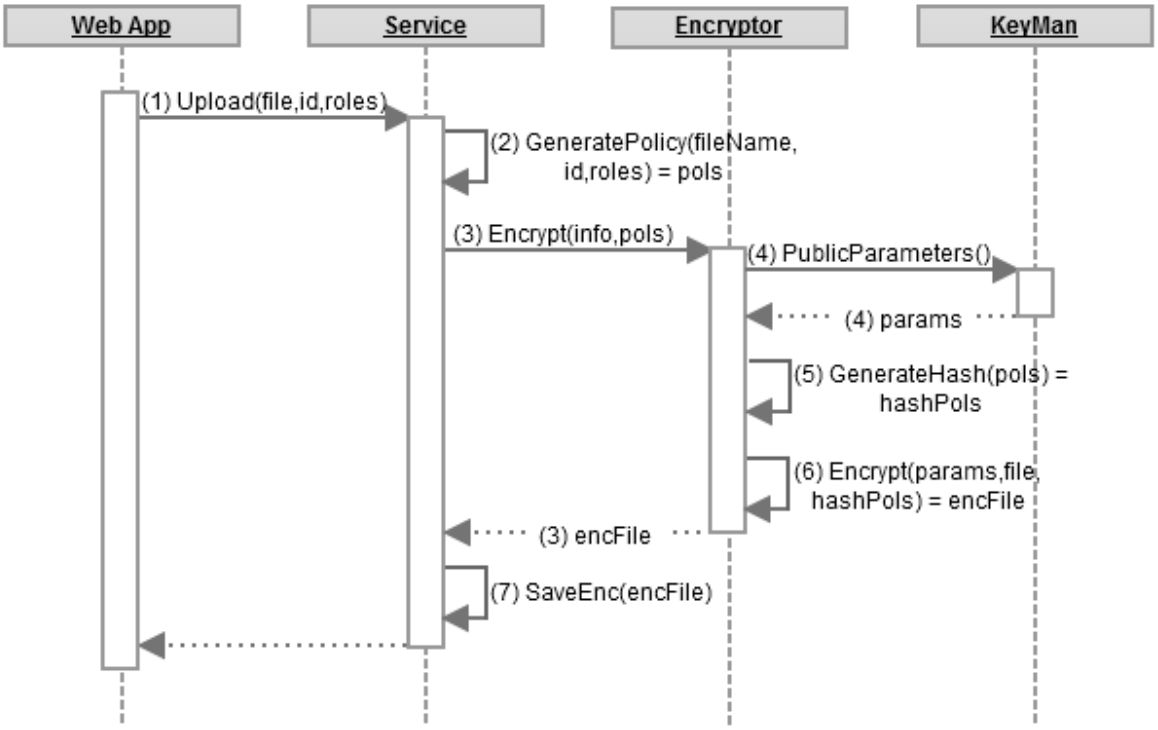


Figura 4.3 Diagrama de sequência do upload de um ficheiro.

Na figura 4.4 está presente o diagrama de sequência do download de um ficheiro em que o utilizador que está autorizado a aceder e os ficheiros guardados não foram corropidos.

Assim a aplicação web (“Web App”) começa por enviar o pedido de download com o nome do ficheiro (“fileName”) pretendido, a identidade do utilizador (“id”) que está a requerer o ficheiro e os seus papéis (“roles”). O “Service” ao receber este pedido vai ao local onde estão armazenados os ficheiros para carregar o ficheiro cifrado (“encFile”) e a suas políticas associadas (“pols”) e de seguida gera um pedido XACML (req) com as informações recebidas no pedido da aplicação web. De seguida o PDP presente no “Service” avalia as políticas (“pols”) para o pedido (“req”). Caso o utilizador não estivesse autorizado o “Service” responderia logo à aplicação web com essa informação. Estando o utilizador autorizado, o “Service” envia para o “Decrypt” o ficheiro cifrado e as políticas para que este o decifre. O “Decrypt” começa por gerar o “hash” das políticas “hashPols” que envia para o “KeyMan”, para que este lhe responda com a chave privada (“privKey”) associada a esse “hash”. Com a chave privada o “Decrypt” decifra o ficheiro e envia o ficheiro decifrado (“file”) para o “Service” que por sua vez envia para a aplicação web. Caso o ficheiro cifrado ou das políticas tivessem sido alterados o “Decryptor” iria detetar e em vez de enviar o ficheiro para o “Service” enviaria a informação do sucedido. Por fim o “Service” enviaria essa informação para a aplicação web.

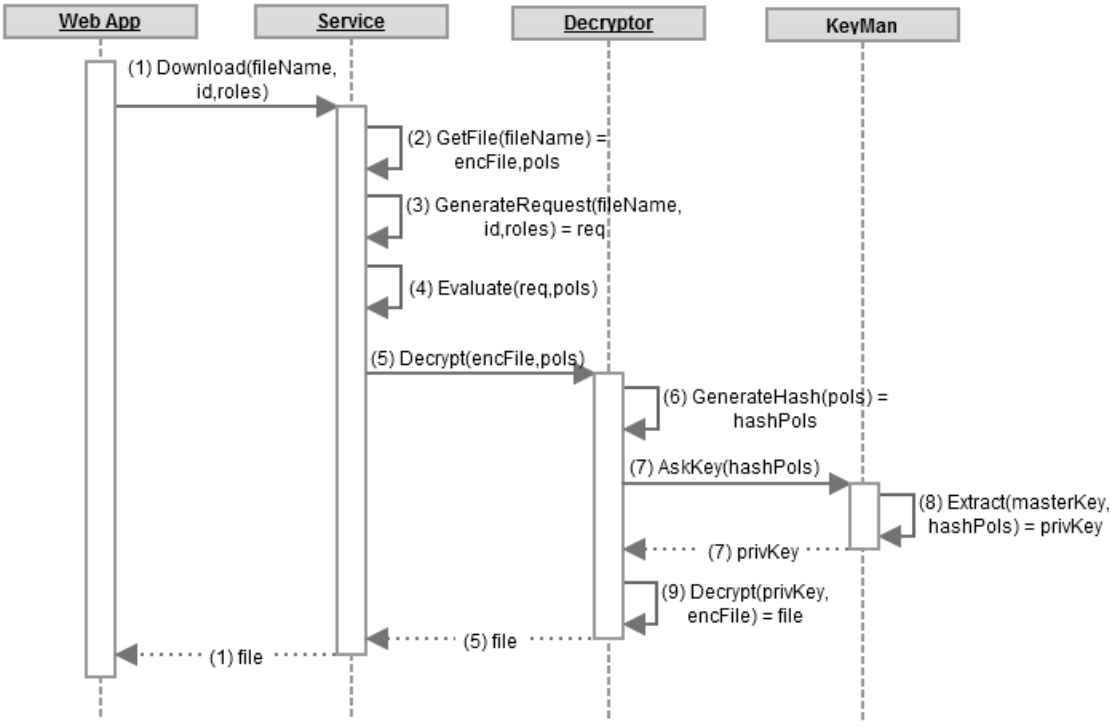


Figura 4.4 Diagrama de sequência do download de um ficheiro.

4.2.2 Service

O “Service” é o ponto do serviço criptográfico que comunica com o exterior, neste caso comunica com a aplicação web. Como pode ser visto nos diagramas de sequência, a aplicação web pode enviar dois tipos de pedidos para o “Service” fazer upload ou download de um ficheiro.

Para se fazer upload de um ficheiro, a aplicação web tem que enviar o ficheiro, o conjunto de papéis que estão autorizados a aceder e a identidade do utilizador. Com estas informações este serviço cria uma política XACML, utilizando a implementação XACML da Sun [58].

A política XACML gerada para cada ficheiro é composta por um conjunto de regras. Existem duas regras que são sempre adicionadas na política, a regra que dá permissão ao dono do ficheiro (o utilizador que faz o upload do ficheiro) e a regra final que nega o acesso ao ficheiro. Como o algoritmo de combinação que está a ser utilizado é o de sobrepor permissão (tabela 2.1), caso não se aplique nenhuma das regras que dão permissão, é utilizada a regra final que nega o acesso. Desta forma, se um utilizador não é abrangido pelas regras de permissão significa que não está autorizado a aceder ao ficheiro. Além destas duas regras é gerada uma regra para cada papel selecionado pelo dono ao fazer o upload. Por exemplo, se forem escolhidos os papéis “admin” e “eps”, é gerada uma regra para cada um deles para descrever aquele quem tem pelo menos um destes papéis, podendo aceder ao ficheiro. Na política XACML também é adicionado o nome do ficheiro ao qual esta corresponde, assim para diferentes ficheiros temos diferentes políticas.

Na figura 4.5 podemos ver parte de uma política XACML gerada pelo “Service”, nesta imagem podemos ver a regra que permite o acesso aos utilizadores que possuem o papel de “admin”.

```
<Rule RuleId="Role_admin" Effect="Permit">
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <AnyResource/>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">download</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>
  <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
      <SubjectAttributeDesignator AttributeId="role" DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="admin@users.example.com"/>
    </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
  </Condition>
</Rule>
```

Figura 4.5 Exemplo de uma regra de uma política em XACML.

Depois de gerada a política de acesso ao ficheiro, é enviado o ficheiro e as políticas associadas para o serviço responsável por cifrar a informação, o “Encryptor”. No final, o “Service” receberá o criptograma que será guardado juntamente com as suas políticas associadas.

Na situação da aplicação web enviar um pedido de download de um dos ficheiros, o "Service" segue o esquema representado na figura 2.8. Ou seja, o serviço recebe, da aplicação web, o pedido de acesso e neste pedido é enviado o nome do ficheiro pretendido, a identidade do utilizador que está a requerer o acesso bem como os seus papéis.

Com o pedido no serviço é manipulado o contexto para o transformar num pedido XACML, neste pedido XACML é especificado o ficheiro que se pretende aceder, a identidade do utilizador que fez o pedido e os seus papéis. Neste caso, como não existe um repositório central de políticas (as políticas estão com os ficheiros correspondentes), não temos um PIP assim, o PEP terá de ir buscar a política armazenada e enviá-la ao PDP juntamente com o pedido XACML. O PDP avaliará a política relativamente ao pedido e enviará a resposta ao PEP. Se o PDP responder positivamente ao pedido de autorização, o criptograma e as políticas associadas são enviadas para o serviço responsável por decifrar, o "Decryptor". Caso tenha sido autorizado e não tenham existido problemas a decifrar o ficheiro, é enviada para a aplicação web o ficheiro decifrado, caso contrário é enviada uma mensagem onde é explicada a razão para não ter sido enviado o ficheiro.

4.2.3 Keyman

O serviço "Keyman" é responsável por gerar e armazenar os parâmetros públicos e privados do esquema IBE e ainda por gerar as chaves privadas utilizadas no esquema IBE (subsecção 2.1.2).

Na implementação do esquema IBE, foi utilizado neste projeto o esquema IBE descrito no artigo [13]. Para implementar em Java os algoritmos do esquema IBE descritos no artigo, foi utilizada a biblioteca jPBC [17].

No caso do "Keyman", foram implementados os algoritmos "setup" e "extract". O "setup" é utilizado no arranque do sistema e este gera os parâmetros públicos e privados utilizados no esquema IBE. Enquanto que o "extract" é utilizado para gerar chaves privadas para uma certa chave pública, isto é, o algoritmo recebe a chave pública que foi utilizada para cifrar e retorna a chave privada que permite decifrar essa mesma informação.

Como é descrito no artigo [13], o algoritmo "extract" tem dois passos, no primeiro a chave pública utilizada é mapeada num ponto de uma curva elíptica $Q_{ID} \in E/F_p$ de ordem q, no segundo é gerada a chave privada (d_{ID}) em que $d_{ID} = sQ_{ID}$, onde s é a chave mestra do esquema IBE. Na figura 4.6 podemos ver a implementação em Java do algoritmo "extract".

Assim sendo, o serviço "Keyman" responde a dois tipos de pedidos – dos parâmetros públicos que pode ser realizado por qualquer serviço, e da chave privada associada a uma chave pública. Este segundo tipo de pedidos apenas pode ser realizado pelo serviço "Decryptor".

4.2.4 Encryptor

Tal como foi referido anteriormente, o serviço "Encryptor" recebe do "Service" o ficheiro para ser cifrado e a política de controlo de acesso.

O ficheiro é cifrado utilizando um cifra simétrica por blocos servindo-se de um vetor de

```java
/**
 * Function to generate the private key to some identity.
 *
 * @param id is a byte array with the identity used to encrypt.
 * @return a byte array with the IBE private key.
 */
public byte[] extract(byte[] id) {

    logger.info(conf.getString("BEGIN_EXTRACT_IBE"));

    //Convert id into an element
    Element Qid = params.getPairing().getG1().newElement();
    Qid.setFromHash(id, 0, id.length);

    //Calculate did (the private key)
    Element did = Qid.mulZn(params.getS());

    logger.info(conf.getString("END_EXTRACT_IBE"));

    return did.toBytes();
}
```

Figura 4.6 Implementação do algoritmo “extract” em Java.

inicialização aleatório. Para isso é gerada uma chave simétrica e um vetor de inicialização.

De seguida o serviço pede ao “Keyman” os parâmetros públicos do esquema IBE e gera o “hash” da política associada ao ficheiro. Assim, a chave simétrica é cifrada através do algoritmo “encrypt” do esquema IBE, usando o “hash” gerado como chave pública (funcionamento do esquema IBE na subsecção 2.1.2).

De modo a garantir a integridade do criptograma e dos restantes elementos são concatenados à chave simétrica cifrada com esquema IBE, o “hash” da política utilizado para cifrar a chave simétrica, o vetor de inicialização da cifra simétrica e o criptograma, sendo que é gerado o MAC do bloco com a chave simétrica utilizada para gerar o criptograma.

No final, o ficheiro enviado para o serviço “Service” é o resultado de concatenar o bloco com o seu MAC.

4.2.5 Decryptor

O “Decryptor”, tal como o “Encryptor”, recebe pedidos do “Service”, mas neste caso são pedidos para decifrar um certo ficheiro. Neste pedido é enviado o ficheiro para ser decifrado e a política de acesso associada.

Quando este serviço recebe um pedido começa por dividir o criptograma nos seus diversos componentes (chave simétrica cifrada, informação cifrada, “hash” da política e MAC do criptograma), se o criptograma não possuir todos estes componentes, o serviço responde que o criptograma foi alterado.

Depois de dividido o criptograma começa-se por gerar o "hash" da política, que será comparado com o "hash" presente no criptograma, caso estes sejam diferentes significa que o criptograma ou o ficheiro com a política de controlo de acessos foram alterados, se isto acontecer o serviço responde que um ou outro foram alterados, visto que não conseguimos saber qual realmente foi alterado.

Caso os "hash's" sejam iguais, o "hash" é enviado para o serviço responsável por gerar as chaves privadas do esquema IBE, o "Keyman". Este responde com a chave privada que o "Decryptor" utiliza no algoritmo "decrypt" do esquema IBE (subsecção 2.1.2) para decifrar a chave simétrica.

Com a chave simétrica, o serviço verificará a integridade do ficheiro, gerando o MAC e comparando-o com o MAC presente no ficheiro. Caso não sejam iguais, o serviço não decifra o criptograma e responde informando que o criptograma ou o ficheiro de políticas foi alterado. Mais uma vez não podemos ter a certeza de qual foi alterado visto que, se o criptograma foi alterado a sua nova assinatura será diferente, mas se o ficheiro da política foi alterado dará origem a uma chave que gerará uma assinatura diferente da verdadeira.

Garantida também a integridade da política de acesso, o serviço decifra o criptograma utilizando a chave simétrica e o vetor de inicialização. Assim que é decifrado, o ficheiro é enviado para o "Service".

4.3 Resultados

Nesta secção serão apresentados os resultados do sistema implementado. Primeiramente surgem os resultados da utilização do sistema de forma legítima bem como a simulação de ataques ao sistema. De seguida surgem os tempos de execução dos algoritmos criptográficos.

4.3.1 Utilização do Sistema

Quando um utilizador acede à aplicação web é-lhe apresentado a página presente na figura 4.7.

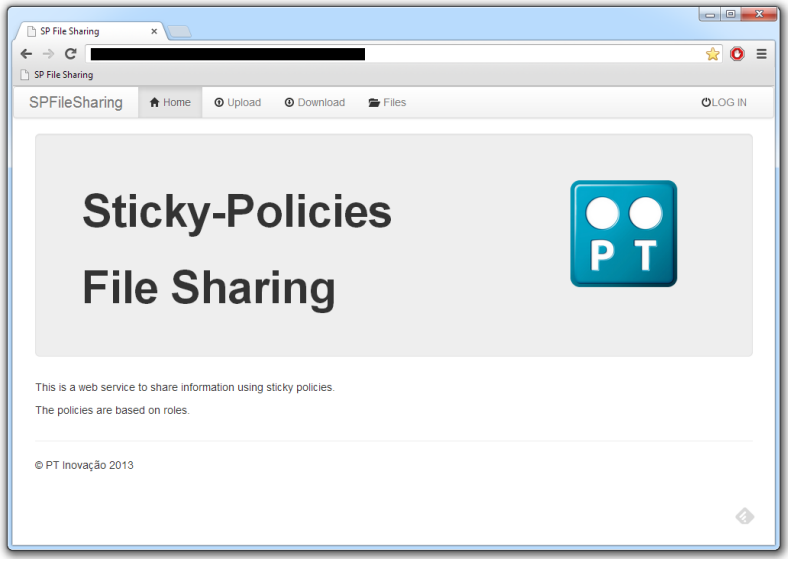


Figura 4.7 Página inicial da aplicação web.

Neste momento o utilizador ainda não está autenticado, para isso terá que carregar em "LOG IN" para que seja redirecionado para a página do IAM (figura 4.8). Nesta página o utilizador terá que inserir as suas credenciais para se autenticar no sistema.

Caso o utilizador introduza corretamente as suas credenciais será redirecionado novamente para a página inicial da aplicação web, mas desta vez, podemos ver (figura 4.9) que o seu nome de utilizador já aparece no canto superior direito pois já se encontra autenticado.

Na página onde é possível fazer o upload de ficheiros o utilizador terá que escolher os papéis que estarão autorizados a aceder ao ficheiro, bem como o ficheiro que deseja fazer upload. De seguida terá que carregar no botão "Upload". Na figura 4.10 podemos ver o resultado da realização do upload do ficheiro "diagrama.png" escolhendo os papéis "eps" e "admin".

Relativamente à página de download de ficheiros, é apresentada uma tabela com os vários ficheiros no sistema. Para cada ficheiro é exposto o seu nome, o seu tamanho, o seu dono e os papéis autorizados a aceder. Na figura 4.11, temos o exemplo da realização do download de um

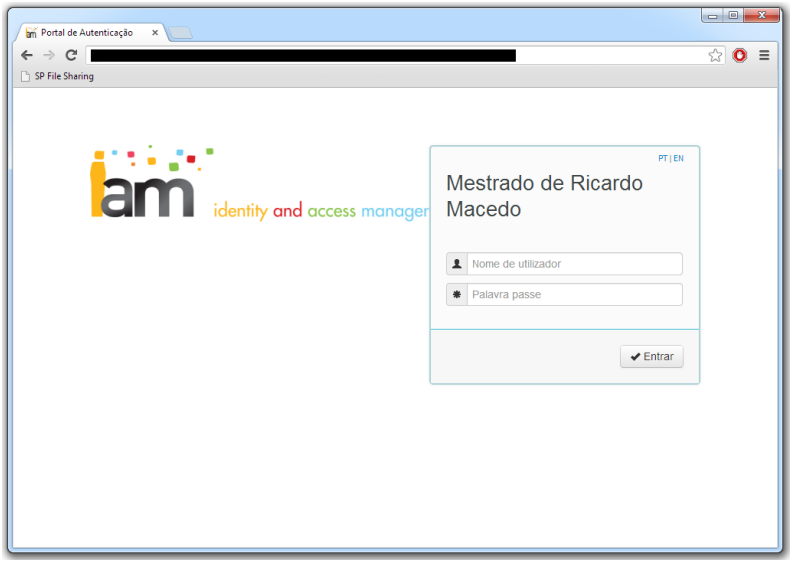


Figura 4.8 Página de autenticação do IAM.

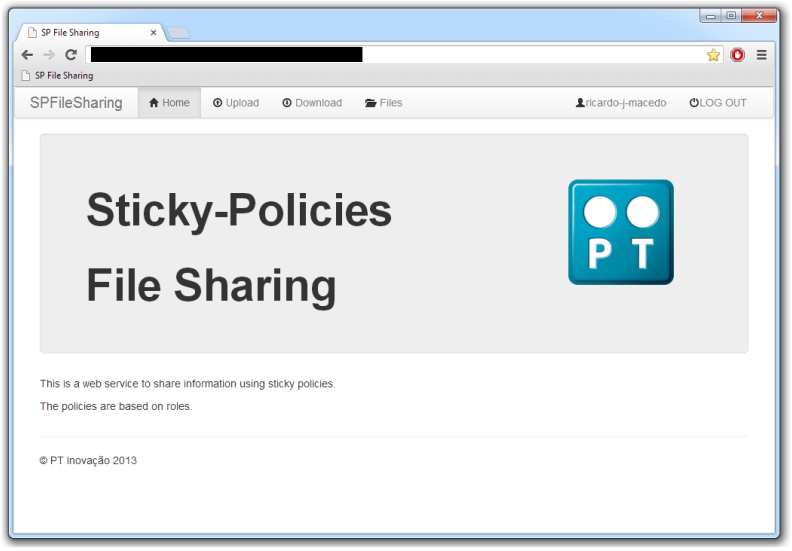


Figura 4.9 Página inicial da aplicação web com o utilizador autenticado.

ficheiro com sucesso. Neste exemplo, a aplicação está a ser utilizada pelo utilizador "ricardo-j-macedo" que tem os papéis "intern" e "user", como o ficheiro escolhido "texto.docx" pode ser acedido pelos papéis "admin", "intern" e "user", o ficheiro é descarregado corretamente. Dos ficheiros presentes no sistema (figura 4.11) o utilizador "ricardo-j-macedo" não poderia realizar o download do ficheiro "documento.txt" pois apenas permite o acesso dos papéis "eps" e "admin", mas poderia fazer o download do ficheiro "diagrama.png" por ser o dono do mesmo.

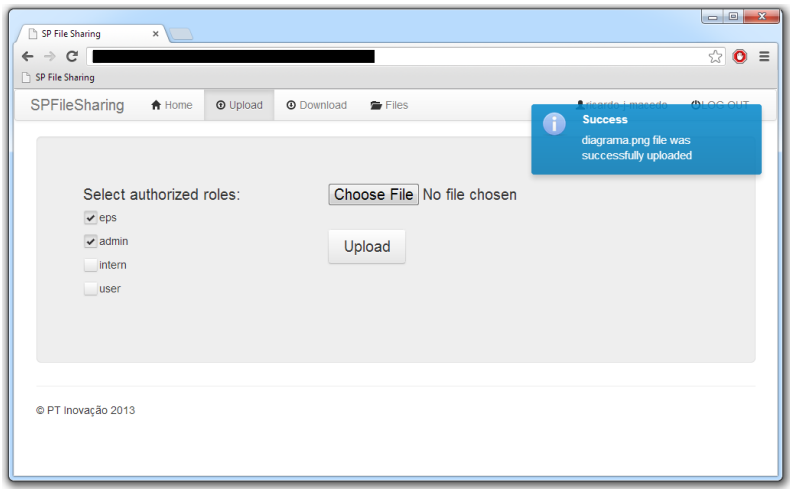


Figura 4.10 Página de upload da aplicação web.

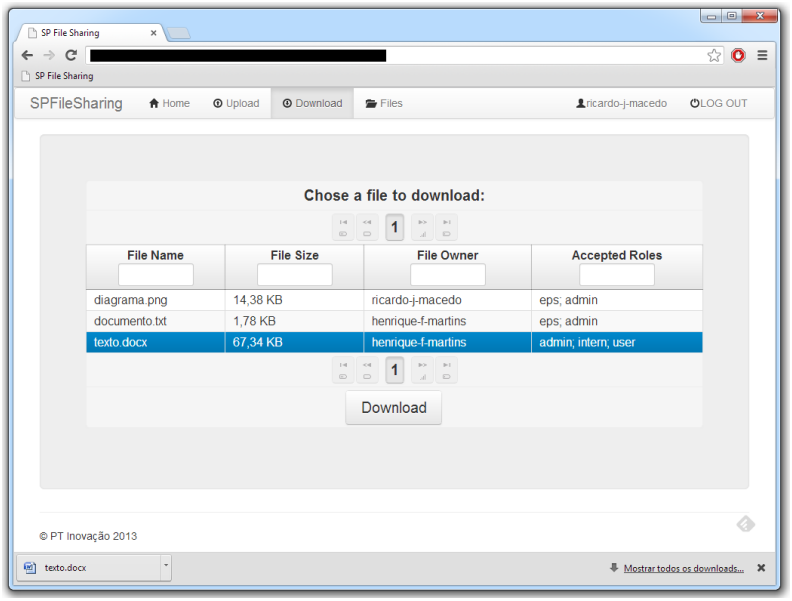


Figura 4.11 Página de download da aplicação web.

4.3.2 Utilização Incorreta do Sistema

Serão agora apresentados os resultados de utilizar incorretamente o sistema por parte de um utilizador, ou seja, os resultados de tentativas de não respeitar as regras do sistema.

O utilizador, para utilizar o sistema terá que estar autenticado, caso tente aceder ao sistema sem se autenticar é apresentada a mensagem presente na figura 4.12. Além de aparecer a mensagem o sistema não permite que o utilizador faça upload ou download de ficheiros.

Depois de autenticado, o utilizar pode tentar fazer o download de um ficheiro ao qual não está autorizado a aceder, neste caso surgirá a mensagem da figura 4.13. No exemplo presente na

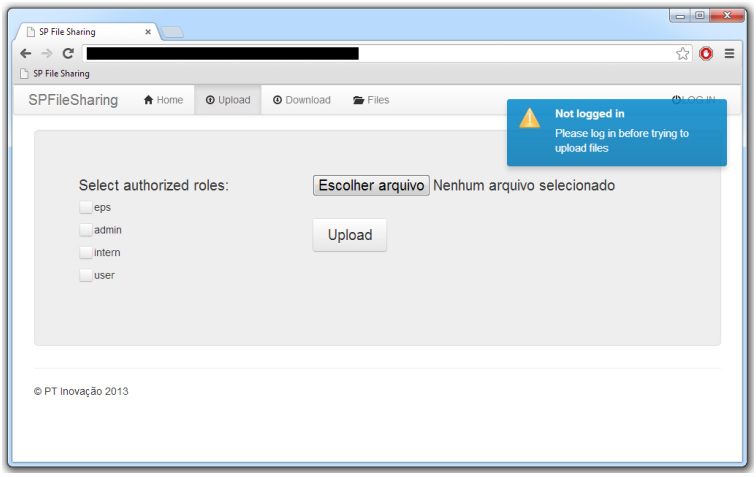


Figura 4.12 Erro de utilização do sistema sem efetuar a autenticação.

figura 4.13, o utilizador "ricardo-j-macedo" com os papéis "intern" e "user" está a tentar aceder ao ficheiro "documento.txt" que só pode ser acedido pelos utilizadores com o papel "eps" e/ou "admin" e pelo seu dono, o utilizador "henrique-f-martins".

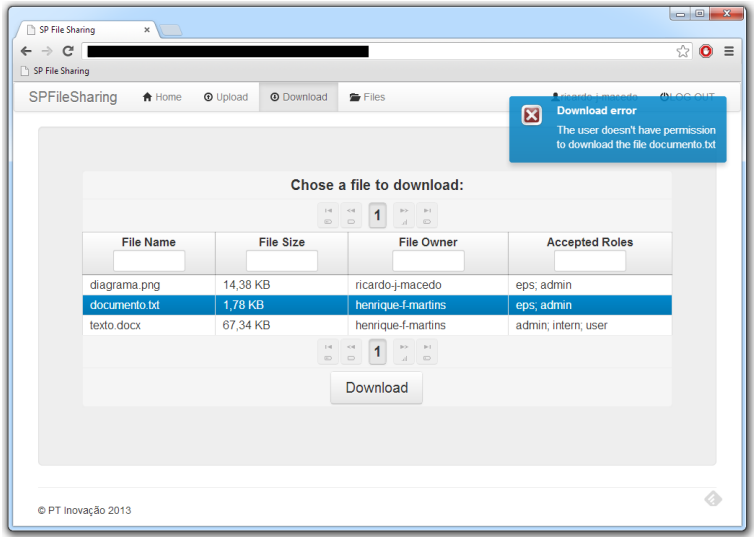


Figura 4.13 Download de um ficheiro sem ser permitido.

Além destas utilizações incorretas da aplicação web, podem ser realizados ataques aos ficheiros guardados, visto que estes podem ser armazenados num local inseguro.

Um utilizador pode tentar alterar a política associada para que assim tenha autorização para aceder ao ficheiro, isto é, no ficheiro com a política de controlo de acessos onde são descritos os papéis autorizados, pode alterar os papéis autorizados. Por exemplo, a política autoriza o papel "admin" e o utilizador alterar para "user" (um papel seu para poder ter acesso). Além da política, o utilizador pode também tentar alterar o criptograma, neste dois casos quando

depois se tenta fazer o "download" do ficheiro é apresentada a mensagem presente na figura 4.14. Nesta Mensagem é referido que um ou outro ficheiro estão corrompidos visto que não é possível saber realmente qual foi, como foi explicado na subsecção 4.2.5.

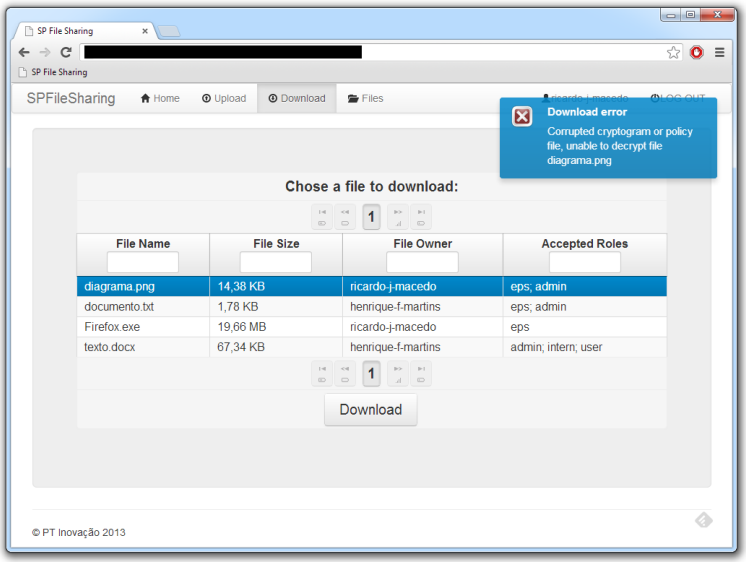


Figura 4.14 Download de um ficheiro depois de corromper o seu ficheiro das políticas.

Porém, existe um situação em que temos a certeza que o criptograma foi alterado – quando o serviço "Decryptor" não é capaz de dividir o criptograma nos seu diversos componentes, neste caso a mensagem a presentada é de que o criptograma foi alterado (figura 4.15).

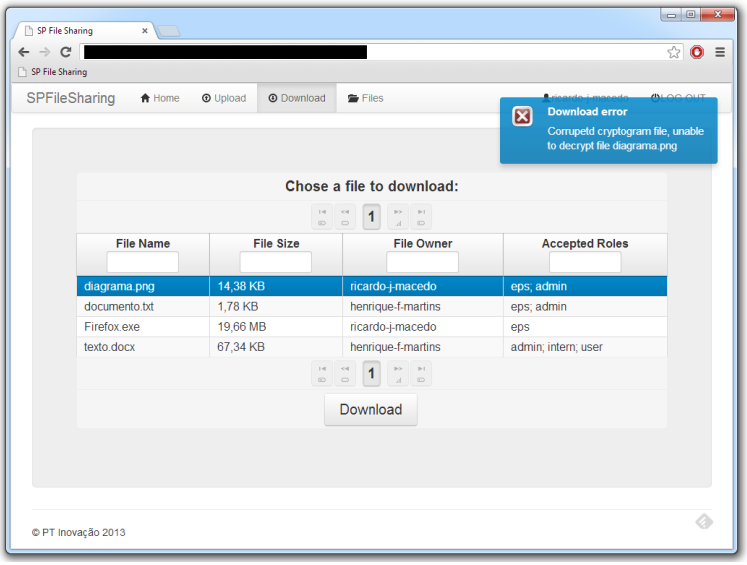


Figura 4.15 Download de um ficheiro depois de corromper o seu criptograma.

4.3.3 Tempos de Execução

Serão agora apresentados os tempos de execução dos principais algoritmos criptográficos utilizados no serviço criptográfico. Para isso serão apresentadas três tabelas com os tempos de execução para ficheiros de diferentes tamanhos. Será apresentada uma tabela para os algoritmos utilizados no upload de um ficheiro, uma para os algoritmos utilizados no download e outra para o algoritmo utilizado pelo serviço que gera as chaves privadas ("KeyMan").

Na tabela 4.1 podemos ver os tempos de execução dos principais algoritmos criptográficos utilizados para fazer upload de um ficheiro. Nesta tabela cada linha representa o upload de um ficheiro, em que na primeira coluna é descrito o tamanho do ficheiro, na segunda o tempo que levou a cifrar o ficheiro através da cifra simétrica e a última coluna contém o tempo que levou a cifrar a chave simétrica utilizando o esquema IBE.

Tabela 4.1 Tempos de execução no upload de um ficheiro.

Tamanho do Ficheiro	Tempo de Cifrar o Ficheiro	Tempo de Cifrar a Chave
14,38 KB	7 ms	139 ms
67,34 KB	13 ms	151 ms
272 KB	27 ms	144 ms
696,51 KB	30 ms	138 ms
1,66 MB	36 ms	142 ms
18,66 MB	795 ms	139 ms

Relativamente à tabela 4.2, tal como na tabela 4.1, são apresentados os tempos de execução dos principais algoritmos criptográfico utilizados, mas nesta, para realizar o download de um ficheiro. Assim, cada linha representa o download de um ficheiro, em que na primeira coluna contém o tamanho do ficheiro, na segunda o tempo de execução do algoritmo que decifrar a chave simétrica através do esquema IBE e na terceira o tempo que levou a decifrar o ficheiro através da cifra simétrica.

Tabela 4.2 Tempos de execução no download de um ficheiro.

Tamanho do Ficheiro	Tempo de Decifrar a Chave	Tempo de Decifrar o Ficheiro
14,38 KB	70 ms	4 ms
67,34 KB	71 ms	17 ms
272 KB	71 ms	18 ms
696,51 KB	67 ms	15 ms
1,66 MB	69 ms	39 ms
18,66 MB	68 ms	869 ms

Por fim, na tabela 4.3 são apresentados os tempos de execução do algoritmo de geração das chaves privadas do esquema IBE, utilizados pelo serviço "KeyMan", em que cada linha

representa uma chave gerada. Como este algoritmo apenas depende do ficheiro da política de controlo de acessos e não do ficheiro a decifrar, na primeira coluna são especificados o número de papéis que estão autorizados pela política e na segunda o tempo que levou a ser gerada a chave privada.

Tabela 4.3 Tempos de execução de gerar chaves privadas no esquema IBE.

Número de Papéis	**Tempo de Gerar a Chave Privada**
1	95 ms
2	94 ms
3	93 ms
4	94 ms

4.4 Comentário dos Resultados

Como pode ser visto na subsecção 4.3.1 e na subsecção 4.3.2 o sistema desenvolvido no decorrer deste projeto tem as funcionalidades pretendidas a funcionar correctamente, sendo estas a capacidade de fazer upload e download de ficheiros, armazenar a informação cifrada e conseguir detetar um conjunto de possíveis ataques causados pela falta de segurança no armazenamento dos ficheiros.

Relativamente aos tempos de execução (subsecção 4.3.3), podemos ver que o tempo de execução dos algoritmos do esquema IBE são desprezáveis, isto é, no contexto de fazer o upload ou o download de um ficheiro ter um acréscimo 150 milissegundos no upload e um acréscimo de 165 milissegundos no download (tempo de gerar a chave privada mais o de decifrar a chave simétrica) são desprezáveis.

Os tempos de execução dos algoritmos do esquema IBE mantêm-se constantes com o crescimento dos ficheiros e com o crescimento das políticas. No que respeita ao tempo de cifrar e decifrar a chave simétrica mantém-se constante pois o tamanho da chave simétrica é independente do tamanho do ficheiro, permanecendo constante mesmo para ficheiros de tamanhos diferentes. O tempo de gerar a chave privada do esquema IBE também se mantém constante com o crescimento das políticas pois o que é utilizado como chave pública não é a política em si, mas um "hash" gerado através dessa política, assim o tamanho do "hash" também é constante.

Através da análise dos tempos é fácil perceber o porquê de a criptografia assimétrica ser apenas utilizada para cifrar/decifrar a chave utilizada pela criptografia simétrica para cifrar/decifrar a informação. Como podemos ver na tabela 4.1 o tempo de cifrar um ficheiro de 1,66 MB é de apenas 36 milissegundos enquanto que cifrar uma chave de 128 bits demorou 142 milissegundos, o mesmo acontece no download (tabela 4.2) que para decifrar esse mesmo ficheiro demorou 39 milissegundos, enquanto que decifrar a chave de 128 bits demorou 69 milissegundos, sem se contabilizar o tempo que foi gasto para gerar a chave privada do esquema IBE.

5 . Registo de Auditoria do Acesso a Ficheiros

Com o sistema que foi desenvolvido neste projeto conseguimos controlar quem consegue aceder aos ficheiros armazenados, mas ainda assim existe a possibilidade de pessoas acederem a ficheiros para os quais não estão autorizados. Isto deve-se ao problema de não ser possível controlar o comportamento dos utilizadores autorizados, ou seja, alguém que seja autorizado a fazer o download de um ficheiro pode-o passar a um outro não autorizado.

Para combater este problema podem ser utilizados registos de auditoria (secção 2.4), como forma de registar os acessos realizados aos ficheiros, para que em caso de ocorrer um difusão indevida de algum ficheiro, se tentar descobrir o responsável.

Nesta secção será então descrito o mecanismo de registo de auditoria utilizado, como se implementou no sistema e os resultados obtidos.

5.1 Mecanismo de Registo de Auditoria

Como foi descrito na secção 2.4, o registo de auditoria consiste em registar os eventos que ocorrem no sistema para se poder analisar o seu comportamento. No contexto deste projeto, é pretendido controlar o acesso aos ficheiros, para isso será realizado o registo dos acessos a cada ficheiro.

A realização deste registo de auditoria pode ser realizado de forma centralizada ou distribuída, visto que no contexto do nosso sistema não estamos interessados em monitorizar o comportamento dos utilizadores no sistema, mas sim o seu comportamento relativamente a cada ficheiro individualmente optou-se por realizar o registo de forma distribuída.

Fazendo o registo de auditoria individualmente para cada ficheiro simplificamos o processo de análise dos acessos a um ficheiro, visto que, quando for detetado a difusão indevida de um ficheiro apenas é necessário analisar os acessos a esse mesmo ficheiro. Também, ao termos um registo de auditoria junto com o ficheiro, como acontece com a política, facilmente o nosso sistema criptográfico pode ser utilizado por diferentes serviços simultaneamente, visto que o registo de auditoria é transparente.

Desta forma, o mecanismo que iremos implementar no nosso sistema registará os acessos a cada ficheiro, guardando essa informação num ficheiro que ficará junto àquele que está a monitorizar. Será importante garantir a segurança do ficheiro de registo de auditoria tal como acontece com o ficheiro da política de controlo de acesso.

5.2 Implementação de Registo de Auditoria no Sistema

Para implementar um mecanismo de registo de auditoria de forma distribuída, como foi explicado anteriormente, é necessário criar uma ligação entre o ficheiro com os registos e o criptograma ao qual diz respeito.

Para criar esta ligação entre criptograma e registo de auditoria, poderíamos utilizar o mesmo mecanismo utilizado com a política de controlo de acessos, isto é, seria gerado o "hash" do registo para que fosse adicionado ao criptograma, assim assinar-se-ia juntamente com os restantes componentes do criptograma.

Contudo, esta escolha iria trazer uma sobrecarga para a rede. Como é adicionado ao criptograma o "hash" do ficheiro de registo de auditoria, sempre que este ficheiro fosse alterado teria que se gerar um novo "hash" e alterar o criptograma. Isto faria com que um utilizador, sempre que acedesse ao ficheiro cifrado, o serviço criptográfico teria que retornar o ficheiro decifrado e o novo criptograma, visto que cada acesso é registado no ficheiro de registo, o que implica que para cada acesso seja gerado um novo criptograma.

Depois de gerado o novo criptograma seria necessário garantir que o sistema apenas aceitava o criptograma mais recente, pois caso contrário, se um atacante desejasse aceder ao ficheiro sem que fosse realizado o registo desta ação, guardava o criptograma atual, acedia ao ficheiro e substituía o criptograma e o ficheiro de registo gerados pelos antigos. Assim, o sistema não iria detetar a alteração e o ficheiro de registo não teria o acesso do atacante. Tal é possível uma vez que não é garantida segurança no local de armazenamento.

Para que o sistema conseguisse detetar este tipo de ataques, teria que guardar a informação do criptograma mais recente assim, se o atacante alterasse o criptograma mais recente por um antigo o sistema já iria detetar.

Mas se temos que guardar o registo do criptograma mais recente no sistema, podemos em vez disso, guardar o registo do ficheiro de registo de auditoria atualizado. Em vez de criarmos a ligação entre o criptograma e o ficheiro de registo de auditoria através do criptograma, poderíamos criar esta ligação no serviço criptográfico.

Guardando um registo da última versão do ficheiro de registo de auditoria e a qual o criptograma que diz respeito, criamos uma ligação entre os dois e assim, sempre que um utilizador aceder ao ficheiro, apenas é necessário retornar o novo ficheiro de registo de auditoria em vez de retornar todo o criptograma.

Para implementar este esquema teremos no "Service" do serviço criptográfico uma base de dados em que associa o "hash" do criptograma à assinatura do ficheiro de registo de auditoria. Sempre que o "Service" receber um pedido de download de um ficheiro irá gerar o novo registo que enviará para o "Decryptor" juntamente com o registo antigo e a assinatura do registo antigo.

Caso o "Decryptor" decifre o ficheiro, responderá com o ficheiro decifrado e também com a assinatura do novo registo que será atualizada na base de dados do "Service".

Se a base de dados do "Service" onde seriam guardados as assinaturas do ficheiro de registo de auditoria, fosse segura, então nem seria necessário assinar o ficheiro. Sempre que o registo de auditoria fosse atualizado o "Service" geraria e guardaria o seu "hash". Utilizando este

método, seria necessário que a base de dados estivesse segura porém, haveria uma diminuição da troca de informação entre o "Service" e o "Decryptor" e reduziria o trabalho realizado pelo "Decryptor", visto que seria desnecessário assinar o ficheiro de registo de auditoria.

Com este mecanismo, o serviço criptográfico responderá aos pedidos de decifrar com o ficheiro decifrado e o ficheiro de registo de auditoria recente que deverá ser atualizado no local onde está armazenado.

5.3 Resultados Obtidos

Depois de implementado o mecanismo de registo de auditoria descrito na secção anterior, o sistema desenvolvido neste projeto faz agora o registo dos acessos aos ficheiros.

Na aplicação desenvolvida é possível escolher um dos ficheiros e visualizar os seus registos de auditoria. Como se pode ver na figura 5.1, é apresentada uma tabela com os acessos ao ficheiro (neste caso escolheu-se o ficheiro "texto.docx") em que para cada acesso é apresentado a data e o tempo em que foi realizado, o utilizador que realizou o pedido e a ação foi executada.

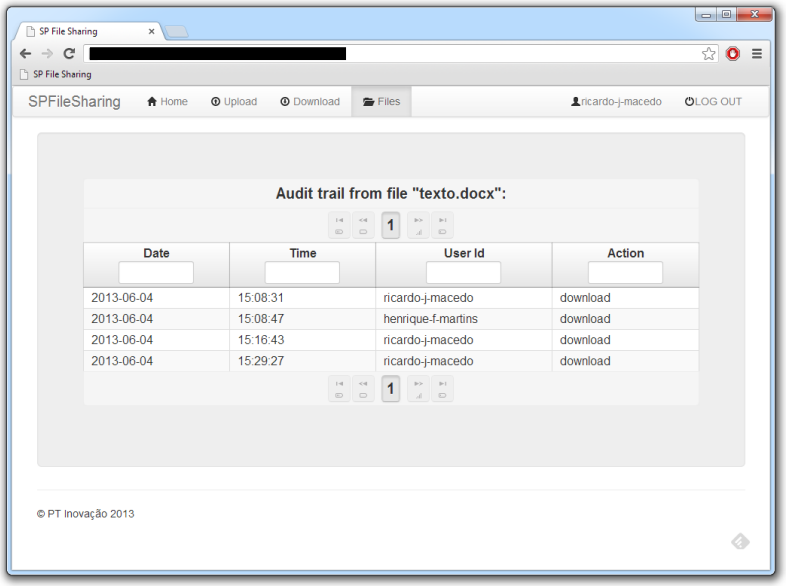


Figura 5.1 Visualização do registo de auditoria.

Como foi descrito na secção anterior, o sistema está preparado para que atacantes tentem manipular ou alterar o ficheiro de registo de auditoria. Caso tal aconteça, é apresentada a mensagem presente na figura 5.2. Nesta imagem alterou-se o ficheiro de registo de auditoria e de seguida tentou-se fazer o download do ficheiro correspondente.

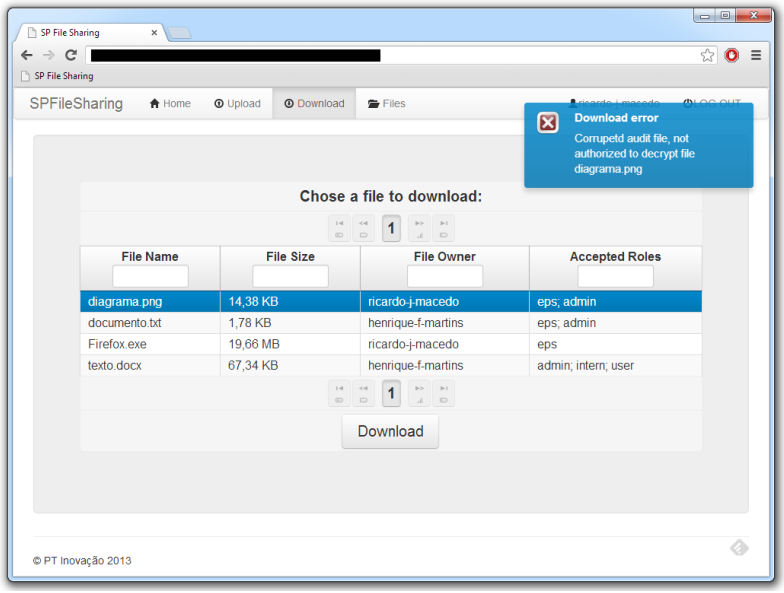


Figura 5.2 Dowload de um ficheiro depois de alterar o registo de auditoria.

6 . Considerações Finais

Com o objetivo de solucionar o problema presente no ambiente empresarial – a informação que é partilhada poder ser indevidamente acedida devido a ser armazenada em claro – foi desenvolvido o sistema apresentado no projeto. O sistema cifra a informação com base num conjunto de políticas, este conjunto de políticas é definido pelo dono da informação e define quem poderá aceder às informações.

As políticas de controlo de acessos são em XACML, permitindo ao sistema lidar com um enorme conjunto de políticas de controlo de acesso. Neste projeto foi utilizado um controlo de acessos baseado em papéis, o qual pode ser alterado e sempre utilizando a mesma linguagem de políticas.

O serviço criptográfico implementado cria uma forte ligação entre o criptograma de um ficheiro e a política de controlo de acesso associada. Assim, poderão ser armazenados em qualquer lugar, visto que um atacante não conseguirá retirar informação do criptograma e qualquer alteração realizada será detetada pelo serviço criptográfico. Esta ligação é criada através da utilização do esquema IBE, em que é gerado um "hash" da política de controlo de acesso que é utilizada como chave de cifragem (o "hash" da política é utilizada como a identidade do destinatário).

Este serviço criptográfico, que é distribuído e facilmente escalável, pode funcionar como um serviço da nuvem em que disponibiliza primitivas de cifragem, controlo de acesso e decifragem. Este controlo de acessos é realizado através das políticas associadas aos ficheiros.

Neste projeto, foi também estudado e implementado um mecanismo distribuído de registo de auditoria, com o objetivo de ser controlado o comportamento dos utilizadores face aos ficheiros. Como foi explicado anteriormente, não foi conseguido obter um mecanismo de registo de auditoria totalmente distribuído. Para implementar este mecanismo foi necessário recorrer ao armazenamento de alguma informação de forma centralizada.

Apesar do sistema com o mecanismo de registo de auditoria necessitar de um ponto central para garantir a integridade dos registos efetuados, se não for utilizado registos de auditoria, o sistema é distribuído e escalável. Visto que os servidores não guardam estado, podem ser replicados para melhorar a performance do sistema face a uma grande quantidade de pedidos.

Como foi referido na secção 4.4, o sistema funciona corretamente avisando o utilizador quando existe alguma tentativa de ludibriar o sistema. Por exemplo, alerta se um atacante tiver alterado o criptograma ou a política de controlo de acesso. Relativamente aos tempos de execução dos algoritmos do esquema IBE, estes são reduzidos e constantes, e por isso desprezáveis. O que tem mais influência no tempo de resposta do sistema são os tempos de execução dos algoritmos criptográficos simétricos utilizados para cifrar e decifrar a informação. Estes aumentam com o aumento do tamanho dos ficheiros porém, são ainda assim tempos razoáveis para um bom funcionamento do sistema.

Como trabalho futuro propõe-se que o sistema implementado como prova de conceito seja alterado de forma a implementar a ligação com um serviço de armazenamento na nuvem para

armazenar os criptogramas devolvidos pelo serviço criptográfico.

Além desta alteração, propõe-se também que sejam estudadas e implementadas novas funcionalidades no sistema, como a possibilidade de o dono do ficheiro alterar a política de controlo de acesso ao mesmo. No sistema atual, uma alteração na política de controlo de acesso significa uma alteração do criptograma, assim sendo, se ao alterar a política, o criptograma também for alterado, então o serviço criptográfico terá que devolver o novo criptograma e um mecanismo que garanta que os antigos criptogramas não serão aceites. Este é um problema idêntico ao que foi descrito no capítulo 5, relativamente ao registo de auditoria do acesso a ficheiros.

Este projeto motivou a elaboração de dois artigos, um aceite para ser publicado na revista técnica da PT Inovação, "Saber & Fazer Telecomunicações", e outro que foi submetido e aguarda notificação para o simpósio de informática, INForum.

Bibliografia

[1] Anne H. Anderson. A comparison of two privacy policy languages: Epal and xacml. In *Proceedings of the 3rd ACM workshop on Secure web services*, SWS '06, pages 53–60, New York, NY, USA, 2006. ACM.

[2] Anne H. Anderson. A comparison of two privacy policy languages: Epal and xacml (online), 2006.

[3] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise privacy authorization language (epal 1.2). Technical report, IBM, 2003.

[4] Paul Ashley, Satoshi Hada, Günter Karjoth, and Matthias Schunter. E-p3p privacy policies and privacy authorization. In *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, WPES '02, pages 103–109, New York, NY, USA, 2002. ACM.

[5] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, February 2006.

[6] Sruthi Bandhakavi, Charles C. Zhang, and Marianne Winslett. Super-sticky and declassifiable release policies for flexible information dissemination control. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, WPES '06, pages 51–58, New York, NY, USA, 2006. ACM.

[7] D. Bell and L. LaPadula. Secure computer system unified exposition and multics interpretation. Technical Report MTR-2997, MITRE Corp., Bedford, MA, July 1975.

[8] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer Berlin Heidelberg, 1998.

[9] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 321 –334, may 2007.

[10] Matt Bishop, Christopher Wee, and Jeremy Frank. Goal-oriented auditing and logging, 1996.

[11] Hai bo Shen and Fan Hong. An attribute-based access control model for web services. In *Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT '06. Seventh International Conference on*, pages 74 –79, dec. 2006.

[12] Piero A. Bonatti and Pierangela Samarati. A uniform framework for regulating service access and information release on the web. *J. Comput. Secur.*, 10(3):241–271, September 2002.

[13] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, 2001.

[14] Sabrina Capitani di Vimercati, Pierangela Samarati, and Sushil Jajodia. Policies, models, and languages for access control. In Subhash Bhalla, editor, *Databases in Networked Information Systems*, volume 3433 of *Lecture Notes in Computer Science*, pages 225–237. Springer Berlin Heidelberg, 2005.

[15] Liqun Chen. Identity-based cryptography. In *FOSAD 2006*, 2006.

[16] E. Damiani, S.D.C. di Vimercati, and P. Samarati. New paradigms for access control in open environments. In *Signal Processing and Information Technology, 2005. Proceedings of the Fifth IEEE International Symposium on*, pages 540 –545, dec. 2005.

[17] Angelo De Caro. The java pairing based cryptography library (jpbc) - http://gas.dia.unisa.it/projects/jpbc/, 2012.

[18] Dorothy E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232, February 1987.

[19] Dod, Donald C. Latham, and An Unknown. Department of defense standard department of defense trusted computer system evaluation criteria, 1985.

[20] Facebook. www.facebook.com, 2012.

[21] S. Farrell and R. Housley. An internet attribute certificate profile for authorization, 2002.

[22] David F. Ferraiolo and D. Richard Kuhn. Role-based access controls. *CoRR*, abs/0903.2171, 2009.

[23] D.Richard Kuhn David F. Ferraiolo. Role-based access control (rbac): Features and motivations. 1995.

[24] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

[25] G. Scott Graham and Peter J. Denning. Protection: principles and practice. In *Proceedings of the May 16-18, 1972, spring joint computer conference*, AFIPS '72 (Spring), pages 417–429, New York, NY, USA, 1972. ACM.

[26] Luan Ibraimi, Qiang Tang, Pieter Hartel, and Willem Jonker. A type-and-identity-based proxy re-encryption scheme and its application in healthcare. In Willem Jonker and Milan Petković, editors, *Secure Data Management*, volume 5159 of *Lecture Notes in Computer Science*, pages 185–198. Springer Berlin Heidelberg, 2008.

[27] UK industry & academia. Encore project - www.encore-project.info, 2008.

[28] Juyoung Kang and Jae Kyu Lee. Rule identification from web pages by the xrml approach. *Decision Support Systems*, 41(1):205 – 227, 2005.

[29] G. Karjoth and M. Schunter. A privacy policy model for enterprises. In *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE*, pages 271 – 281, 2002.

[30] Günter Karjoth, Matthias Schunter, and Michael Waidner. Platform for enterprise privacy practices: Privacy-enabled management of customer data. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 69–84. Springer Berlin Heidelberg, 2003.

[31] Butler W. Lampson. Protection. *SIGOPS Oper. Syst. Rev.*, 8(1):18–24, January 1974.

[32] Bo Lang, Ian Foster, Frank Siebenlist, Rachana Ananthakrishnan, and Tim Freeman. Attribute based access control for grid computing.

[33] Jae Kyu Lee and Mye M. Sohn. The extensible rule markup language. *Commun. ACM*, 46(5):59–64, May 2003.

[34] Ninghui Li, J.C. Mitchell, and W.H. Winsborough. Design of a role-based trust-management framework. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 114 – 130, 2002.

[35] Markus Lorch, Dennis Kafura, and Sumit Shah. An xacml-based policy management and authorization service for globus resources. In *Proceedings of the 4th International Workshop on Grid Computing*, GRID '03, pages 208–, Washington, DC, USA, 2003. IEEE Computer Society.

[36] Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafura, and Sumit Shah. First experiences using xacml for access control in distributed systems. In *Proceedings of the 2003 ACM workshop on XML security*, XMLSEC '03, pages 25–37, New York, NY, USA, 2003. ACM.

[37] Masahiro MAMBO and Eiji OKAMOTO. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts (special section on cryptography and information security). *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 80(1):54–63, jan 1997.

[38] Pietro Mazzoleni, Bruno Crispo, Swaminathan Sivasubramanian, and Elisa Bertino. Xacml policy integration algorithms. *ACM Trans. Inf. Syst. Secur.*, 11(1):4:1–4:29, February 2008.

[39] M.C. Mont, S. Pearson, and P. Bramhall. Towards accountable management of identity and privacy: sticky policies and enforceable tracing services. In *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, pages 377 – 382, sept. 2003.

[40] M. Naedele. Standards for xml and web services security. *Computer*, 36(4):96 – 98, april 2003.

[41] James Nechvatal. Public-key cryptography. Technical report, Security Technology Group, 1991.

[42] OASIS. Advancing open standards for the information society - www.oasis-open.org, 2012.

[43] U. S. Departement of Justice. Ex-employee of airport transportation company arrested for allegedly hacking into computer, destroying data - www.justice.gov/criminal/cybercrime/press-releases/2003/tranarrest.htm, February 2003.

[44] Oracle. Project jersey - http://jersey.java.net/, 2013.

[45] Sylvia Osborn, Ravi Sandhu, and Qamar Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Inf. Syst. Secur.*, 3(2):85–106, May 2000.

[46] S. Pearson and M.C. Mont. Sticky policies: An approach for managing privacy across multiple parties. *Computer*, 44(9):60 –68, sept. 2011.

[47] T. Priebe, W. Dobmeier, and N. Kamprath. Supporting attribute-based access control with ontologies. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, page 8 pp., april 2006.

[48] Torsten Priebe, Wolfgang Dobmeier, Björn Muschall, and Günther Pernul. Bac - ein referenzmodell für attributbasierte zugriffskontrolle. In Hannes Federrath, editor, *Sicherheit*, volume 62 of *LNI*, pages 285–296. GI, 2005.

[49] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer Berlin Heidelberg, 2005.

[50] Pierangela Samarati and SabrinaCapitani Vimercati. Access control: Policies, models, and mechanisms. In Riccardo Focardi and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, pages 137–196. Springer Berlin Heidelberg, 2001.

[51] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38 –47, feb 1996.

[52] R.S. Sandhu and P. Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40 –48, sept. 1994.

[53] L. Seitz, E. Rissanen, T. Sandholm, B. S. Firozabadi, and O. Mulmo. Policy administration control and delegation using xacml and delegent. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, GRID '05, pages 49–54, Washington, DC, USA, 2005. IEEE Computer Society.

[54] Adi Shamir. Identity-based cryptosystems and signature schemes. In GeorgeRobert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin Heidelberg, 1985.

[55] Internet World Stats. Facebook users in the world - www.internetworldstats.com/facebook.htm, 2012.

[56] Internet World Stats. Internet usage statistics - www.internetworldstats.com/stats.htm, 2012.

[57] William H. Stufflebeam, Annie I. Antón, Qingfeng He, and Neha Jain. Specifying privacy policies with p3p and epal: lessons learned. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, WPES '04, pages 35–35, New York, NY, USA, 2004. ACM.

[58] Sun. Sun's xacml implementation - http://sunxacml.sourceforge.net/, 2006.

[59] Qiang Tang. On using encryption techniques to enhance sticky policies enforcement. Technical Report TR-CTIT-08-64, Centre for Telematics and Information Technology University of Twente, Enschede, 2008.

[60] Qiang Tang. Type-based proxy re-encryption and its construction. In *Proceedings of the 9th International Conference on Cryptology in India: Progress in Cryptology*, INDOCRYPT '08, pages 130–144, Berlin, Heidelberg, 2008. Springer-Verlag.

[61] The OASIS technical commitee. *eXtensible Access Control Markup Language (XACML) Version 2.0*. February 2005.

[62] The OASIS technical commitee. *eXtensible Access Control Markup Language (XACML) Version 3.0*. September 2012.

[63] W3C. Platform for privacy preferences (p3p) project - www.w3.org/p3p/, 2007.

[64] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. A logic-based framework for attribute based access control. In *Proceedings of the 2004 ACM workshop on Formal methods in security engineering*, FMSE '04, pages 45–55, New York, NY, USA, 2004. ACM.

[65] Xin Wang, Guillermo Lao, Thomas DeMartini, Hari Reddy, Mai Nguyen, and Edgar Valenzuela. Xrml – extensible rights markup language. In *Proceedings of the 2002 ACM workshop on XML security*, XMLSEC '02, pages 71–79, New York, NY, USA, 2002. ACM.

[66] E. Yuan and J. Tong. Attributed based access control (abac) for web services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pages 2 vol. (xxxiii+856), july 2005.

[67] Nan Zhang, Mark Ryan, and Dimitar P. Guelev. Synthesising verified access control systems in xacml. In *Proceedings of the 2004 ACM workshop on Formal methods in security engineering*, FMSE '04, pages 56–65, New York, NY, USA, 2004. ACM.