Incorporação de Certificados SPKI/SDSI ao Protocolo SSL

Cristian Ferreira de Souza
Universidade de Brasília – UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Campus Universitário – Asa Norte
Brasília, DF – 70910-900
cristian@cic.unb.br

Luiz Antônio da Frota Mattos
Universidade de Brasília – UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Campus Universitário – Asa Norte
Brasília, DF – 70910-900
frota@cic.unb.br

RESUMO

Desde 1978 quando Kohnfelder introduziu o conceito de certificados digitais criou-se a falsa idéia de que a única função dos certificados digitais é a de associar um nome a uma chave pública, como ocorre nos certificados X.509. Rivest, Lampson e Carl Ellisson propõem uma nova infra-estrutura de chave pública – SPKI/SDSI (Simple Public Key Infrastructure/Simple Distributed Security Infrastructure), onde é introduzido o conceito de certificados de autorização, espaço local de nomes e abolido o papel da Autoridade Certificadora (CA). Discutiremos a respeito da teoria presente nos certificados SPKI/SDSI e analisaremos as possíveis modificações a serem feitas no protocolo SSL a fim de que suporte a incorporação destes certificados.

ABSTRACT

Since 1978, when Kohnfelder introduced the concept of digitals certificates, one originated the false idea that the only function of the digitals certificates is to associate a name to a public key, as it occurs on the X.509 certificates. Rivest, Lampson and Carl Ellisson propose a new infrastructure of public key – SPKI/SDSI (Simple Public Key Infrastructure/Simple Distributed Security Infrastructure), where it is introduced the concept of authorization certificates, local space of names and it is banished the role of Certification Authority. We will discuss about the present theory on the SPKI/SDSI certificates and analyse the possible modification to be made in the SSL protocol in order to support these certificates

1 INTRODUÇÃO

Diffie e Hellman em seu artigo *New Directions in Cryptography* [1] sugeriram que um modo mais fácil de se disponibilizar a chave pública de outros usuários seria a criação de um diretório on-line, semelhante a uma lista telefônica, onde ao invés de se ter nomes, endereços, e telefones, teria-se uma associação direta entre os nomes e suas chaves públicas. Kohnfelder[2] por sua vez notou que a criação de um diretório on-line como este sofreria graves problemas de performance, propondo então que cada entrada deste diretório passasse a ser um objeto assinado digitalmente, o qual chamaria de certificado. Desde então o termo certificado vem sendo assumido apenas como uma associação entre um nome e uma chave.

Na verdade podemos dizer que os certificados podem ser divididos em duas grandes classes: Os certificados de identidade – que procuram atestar algum conhecimento que o emissor do certificado (*Issuer)*, têm sobre a identidade do proprietário do certificado (*Subject),* e os certificados de autorização – que expressam autorização, delegação e cujo foco principal é o controle de acesso. O interesse pelos certificados de autorização surgiram a partir de duas iniciativas relacionadas mas que vinham sendo desenvolvidas de forma independente; o SPKI [3] pela IETF, Carl Ellison e outros membros e o SDSI [4] por Ronald Rivest e Butler Lampson. Devido estas duas propostas possuírem vários pontos em comum elas convergiram para um só especificação SPKI/SDSI 2.0 .

Abordando a infra-estrutura SPKI/SDSI e os certificados de autorização acima citados, o artigo encontra-se organizado da seguinte maneira: A seção 2 apresenta o espaço local de nomes e os problemas existentes com as Autoridades Certificadoras. A seção 3 trata sobre a estrutura dos certificados de autorização e a redução 5-upla. A seção 4 discute o novo processo de revogação de certificados com suas diferenças em relação aos certificados X.509. A seção 5 sugere as possíveis modificações no protocolo SSL para a incorporação dos certificados SPKI/SDSI. A seção 6 informa sobre a implementação, e a seção 7 apresenta as conclusões do artigo.

2 ESPAÇO LOCAL DE NOMES

O modelo hierárquico de confiança em que se baseia os certificados X.509 e a tentativa de utilizar um nome que seja único e que identifique globalmente o proprietário do certificado, na prática infelizmente

não funcionam como inicialmente proposto.

1. As Autoridades Certificadoras (CAs) trabalham de forma individualizada, não havendo nenhuma relação de confiança entre elas.
2. Como cada Autoridade Certificadora trabalha de forma individual, cada uma com seus próprios critérios, não há como estabelecer um *Distinguished Name* que seja realmente único e identifique de maneira global o proprietário do certificado.

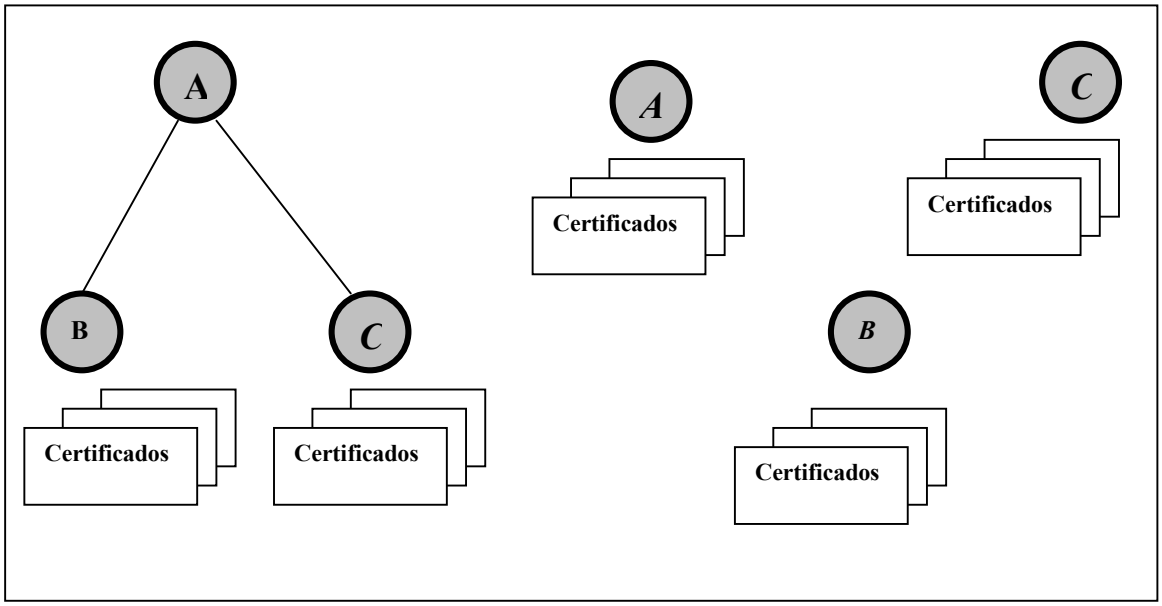

Figura 1: Modelo hierárquico de confiança X.509 como proposto e seu funcionamento atual.

No SPKI/SDSI não existe mais o *Distinguished Name;* os certificados passam a ser identificados pela chave pública e o sujeito é a própria chave pública. O mapeamento entre a chave pública e o proprietário do certificado *(keyholder)* passa a existir no *espaço local de nomes*. Os nomes não precisam mais ser globalmente únicos, entretanto eles precisam ser únicos e significativos para a pessoa que os mantém, semelhantes aos nomes que cadastramos em nossas agendas telefônicas ou aos apelidos em nossa lista de contato de e-mails. Neste caso, cada usuário é livre para que de acordo com o seu grau de confiança aceite ou emita certificados para quem desejar, dispensando o papel da Autoridade Certificadora.

Os certificados contidos nos diversos espaços locais de nomes podem ser vinculados através de nomes SDSI. Um nome SDSI é uma seqüência de tamanho arbitrário seguida por um ou mais identificadores, i. e. $\mathbf{K_{Carol}}$ **BOB ADVOGADO** como exemplificado abaixo:

```
(public-key
        (rsa-pkcs1-md5
          (e #010001#)
          (n
            |AKd6zpHoNSq53rFFZ6XZV5m+YZKr1Ym9fUMuE8yHMOZx9AsrWqkZyMxZ9Cn
            5mIPaTLqYEakoJ7lir25a/Nq1IpvFC8mlalpI2LINDRnIt3HbmiFEj97Eaqk
            2ffhwXotFHIbPGyrgrMobOtWIpG56Ru5AK230lG68+ZyntCqB53v5|)))
         BOB ADVOGADO))
```

O símbolo $\mathbf{K_{Carol}}$ representa a chave pública que define o espaço local de nomes de Carol. O identificador **BOB** que vem em seguida pode ser interpretado como outro nome SDSI ($\mathbf{K_{BOB}}$) presente no espaço local de nomes de K_{Carol}. O identificador **ADVOGADO** indica que este por sua vez está definido no espaço local de nomes definido pela chave pública associada a **BOB** .

3 CERTIFICADOS DE AUTORIZAÇÃO e REDUÇÃO 5-UPLA

Os certificados de autorização podem ser representados como sendo uma 5-upla <I, S, D, A, V> que será utilizada para a redução de uma cadeia de certificados. Uma 5-upla é dita uma representação do certificado

justamente porque a diferença existente é a ausência de assinatura na 5-upla, já que ela existe apenas em memória.

Quando um determinado sujeito ou chave pública deseja obter acesso a algum recurso, ele apresenta uma cadeia de certificados [5] ao verificador que irá realizar a redução dos certificados e verificar as autorizações permitidas.

Os elementos de uma 5-upla são:

ISSUER – I : O emissor pode ser uma chave pública ou o hash de uma chave pública. A esta chave pública o emissor possuirá uma chave privada correspondente ao qual somente tem acesso.

SUBJECT – S: O sujeito é aquele que irá receber a(s) autorização(ções) do Issuer. Pode ser uma chave pública, o hash de uma chave pública ou um nome SDSI. Suponhamos que S possua o valor **Bob**, neste caso as autorizações estão sendo passadas para a chave pública cujo certificado de identidade com o nome **Bob** está no espaço local de nomes de I.

DELEGATION – D: O campo delegation é um campo lógico e indica se o sujeito S pode delegar as autorizações recebidas de I para outros sujeitos.

AUTHORIZATION – A: Contém as autorizações recebidas do Issuer. As autorizações são definidas em um formato livre, variando de acordo com a aplicação. Um exemplo de autorização é o seguinte: `(ftp (host ftp.clark.net) (dir /pub/cme))` , onde é permitido acesso completo (read,write,...) mas somente ao diretório `/pub/cme.`

VALIDITY – V: Define o período de validade do certificado, que pode tanto ser um intervalo entre "not-before" e "not-after" como também alguma forma de validação on-line.

Vejamos o seguinte exemplo referente a redução: Suponhamos que <I1, S1, D1, A1, V1> seja um certificado emitido para o sujeito S1, pelo emissor I1, com D1 indicando *Delegation* = true , Autorizações A1, e validade V1.

Como S1 possui D1= true, S1 agora emite um certificado para S2 que pode ser representado por <I2, S2, D2, A2, V2>

O resultado final do processo de redução pode ser expresso como uma 5-upla:

<I1, S2, D2, AIntersect(A1,A2), VIntersect(V1,V2)>

Um dos principais objetivos da redução além de simplificar o processo de autorização, é assegurar que dentro da cadeia de certificados, mesmo que o emissor no topo da cadeia conceda o direito de delegar autorizações para outros sujeitos, estes não poderão obter autorizações maiores que as do próprio emissor.

Os campos de um certificado SPKI/SDSI são: *version, cert-display, issuer, issuer-loc, subject, subject-loc, deleg, authorization, validity, comment, signature.* Todos estes, exceto issuer, subject, authorization e signature, são campos opcionais.

Campo	Descrição
version	Indica a versão do certificado; caso não seja especificado, por definição subentende-se que a versão é 0.
cert-display	Instrui a aplicação que irá manipular o certificado a como realizar a interpretação de determinados itens que estejam presentes nele. Pode ser utilizado por exemplo em um auto-certificado onde o emissor emite um certificado para si próprio e adiciona um objeto específico. `( Photo: [image/gif] ...)`
issuer-loc	Também chamado issuer-info. Se presente, ele provê a (s) URL (s) da localização dos certificados pelo qual o emissor deriva a autoridade passada através do campo *authorization* no presente certificado. É esperado que no processo de redução o proprietário do certificado reuna estes certificados e apresente-os ao verificador. Neste caso o campo issuer-loc tem o propósito de facilitar esta tarefa.
subject-loc	Semelhante ao campo issuer-loc, ele provê informações a respeito do sujeito. Se o sujeito for por exemplo o hash de uma chave pública, ele pode fornecer a localização desta

	chave. Se o sujeito for um nome SDSI, então ele poderá dar a localização do servidor de certificados deste nome SDSI.
comment	Comentário livre sobre o certificado que só tem significado para um outro *keyholder*, sendo ignorado pelo software que manipulará o certificado.

4 REVOGAÇÃO DE CERTIFICADOS

Nas infra-estruturas de chave pública a revogação ocorre quando o certificado emitido deve tornar-se sem efeito antes de sua data de expiração; seja pelo mal uso da autoridade delegada, comprometimento da chave privada do proprietário ou quando não existe mais nenhuma relação entre e o sujeito e as informações contidas no certificado, i.e. quando um funcionário é desligado de uma empresa. No caso dos certificados X.509 a revogação é feita com base no uso de *Listas de Revogação de Certificados* (CRLs) que deverão ser emitidas periodicamente pelas Autoridades Certificadoras; a CRL é uma lista assinada digitalmente que contém sua data de emissão, os certificados revogados, e a data em que será emitida uma próxima CRL.

O principal problema existente é justamente o intervalo de tempo entre uma emissão e outra, que poderá tanto ser daqui a uma hora, um dia, uma semana ou um mês. Para uma determinada Autoridade Certificadora o intervalo de uma semana pode ser considerado um tempo bom, mas na verdade quem está correndo o risco de aceitar um certificado que não seja mais válido neste período é o verificador e não a CA. Outro problema é que cabe ao verificador a tarefa de localizar todas as CRLs emitidas pelas diversas Autoridades Certificadoras existentes.

Na infra-estrutura SPKI/SDSI assume-se que os usuários que emitem certificados possam prover um serviço on-line, ou designar um servidor específico, que tenha as funções de armazenar os certificados a fim de que sejam obtidos por outros usuários e responder a requisições de reconfirmação de assinatura. Ao invés de se ter as listas de revogação de certificados como forma de revogar a assinatura em um objeto previamente assinado, o emissor do certificado pode ao realizar as assinaturas especificar um período de reconfirmação apropriado para cada uma delas [4]. Cabe ao proprietário do certificado buscar evidências junto ao emissor que seu certificado é válido e apresentá-las ao verificador[6].

Um requisição de reconfirmação de assinatura têm o seguinte formato:

```
( Reconfirm.Query:
      ( To: ( Principal: ... ) )
      ( Signed-Object:
         ( Signed:
            ( Object-Hash: ( SHA1: ... ) )
            ( Date: 1999-12-25-08:00.000-0500 )
            ( Signature: ... ) ) )
```

O campo `Signed-Object` contém o hash do objeto assinado, cuja assinatura está sendo solicitada a reconfirmação. A resposta da reconfirmação de assinatura poderá ser a assinatura do hash do objeto juntamente com a data da reconfirmação, ou um novo certificado.

5 INCORPORAÇÃO SPKI/SDSI/SSL

A proposta do SPKI/SDSI é servir como uma nova infra-estrutura de chave pública para autenticação e principalmente autorização, entretanto em sua especificação não está definido nenhum mecanismo que assegure a segurança do canal, o qual dentre os principais riscos existentes encontram-se os ataques ativos. O protocolo SSL [14] , padrão de *facto* para construção de canais seguros na Internet, foi concebido tendo em vista apenas os certificados X.509 que são certificados de identidade emitidos único e exclusivamente por uma Autoridade Certificadora.

Sendo parte do protocolo SSL, o protocolo *Handshake* é responsável por uma série de mensagens trocadas entre o cliente e o servidor. Dentre estas, 4 estão relacionadas diretamente aos certificados:

- **certificate** **S→C:** `cadeia de certificados X.509v3`
- **certificate_request** **S→C:** `tipo, autoridades_certificadoras`
- **certificate** **C→S:** `cadeia de certificados X.509v3`
- **certificate_verify** **C→S:** `assinatura`

Vejamos então o significado atual das mensagens e as possíveis modificações propostas:

certificate S→C: `cadeia de certificados X.509v3`

O servidor para ser autenticado envia um ou mais certificados X.509v3 para o cliente, que deverá possuir o certificado raiz (root) da Autoridade Certificadora para a verificação da assinatura.

Os Browsers que utilizamos atualmente contam com um arquivo de mais de 90 chaves públicas de autoridades certificadoras para que seja realizada a verificação da assinatura. No caso do SPKI/SDSI é esperado que o próprio cliente seja responsável por estabelecer sua relação de confiança com os outros usuários, e portanto não haveria previamente um arquivo com todas estas chaves públicas. Neste caso podemos ter a seguinte abordagem:

Suponhamos que um banco chamado BANCOX deseje que os correntistas movimentem suas contas através de certificados. O BANCOX emite um certificado para seu gerente, GERENTEX, que será encarregado de estabelecer uma relação de confiança com o CORRENTISTAX. O correntista passaria então a ter em sua posse a seguinte cadeia de certificados, mostrados na forma <Emissor, Sujeito> :

$<K_{BANCOX}, K_{GERENTEX}>$

$<K_{GERENTEX}, K_{CORRENTISTAX}>$

O correntista agora quer movimentar sua conta e ter a certeza que está se comunicando com o servidor correto.

O servidor enviaria o certificado $<K_{BANCOX}, K_{BANCOX}>$ para o cliente, e como o cliente acredita que seu certificado $<K_{GERENTEX}, K_{CORRENTISTAX}>$ é válido pois confia em $K_{GERENTEX}$ e portanto a cadeia de certificados de onde descende as autorizações também é valida; utiliza a chave pública do emissor do certificado que está no topo da cadeia e verifica a assinatura de $<K_{BANCOX}, K_{BANCOX}>$.

Ao invés de um arquivo com chaves públicas de Autoridades Certificadoras, a verificação das assinaturas deverá ser feita pelo cliente utilizando as chaves públicas dos Emissores (*Issuers)* que se encontram nos topos das cadeias.

certificate_request S→C: `tipo, autoridades_certificadoras`

O servidor requisita o certificado do cliente informando o tipo do certificado (algoritmo de chave pública) e uma lista de distinguished names de autoridades certificadoras confiáveis.

```
enum {
        rsa_sign(1),dss_sign(2),rsa_fixed_dh(3),dss_fixed_dh(4),
        rsa_ephemeral_dh(5),dss_ephemeral_dh(6),fortezza_dms(20),(255)
} ClientCertificateType;

struct {
        ClientCertificateType certificate_types<1..28-1>;
        DistinguishedName certificate_authorities<3..216-1>;
} CertificateRequest;
```

O campo enumerado `ClientCertificateType` define os vários tipos de certificados permitidos, entretanto como os certificados SPKI/SDSI utilizam basicamente RSA o campo `certificate_types` pode apenas informar o tipo `rsa_sign`.

O campo `certificate_authorities` pode ser excluído já que sua função era de informar ao cliente quais autoridades certificadoras o servidor confiava como emissoras de certificado. Agora o cliente deverá obrigatoriamente enviar uma cadeia de certificados em que o emissor no topo desta cadeia é o mesmo emissor do certificado apresentado pelo servidor, isto porque como o servidor é o proprietário dos recursos que estão sendo protegidos, cabe a ele aceitar somente certificados em que o emissor no topo da cadeia é o próprio.

certificate C→S: `cadeia de certificados X.509v3`

O cliente para ser autenticado envia uma cadeia de certificados X.509v3 de acordo com os parâmetros definidos na mensagem certificate_request.

Como dito anteriormente, o cliente só poderá obter acesso aos recursos protegidos pelo servidor desde que ele envie uma cadeia de certificados em que o emissor no topo desta cadeia é o próprio servidor.

certificate_verify C→S: `assinatura`

O cliente deverá provar explicitamente que é o verdadeiro proprietário do certificado utilizando a chave privada correspondente.

```
CertificateVerify.signature.md5_hash
        MD5(master_secret + pad2 + MD5(handshake_messages +
```

```
                    master_secret + pad1));

    Certificate.signature.sha_hash
            SHA(master_secret + pad2 + SHA(handshake_messages +
                    master_secret + pad1));
```

A mensagem permanecerá a mesma. O cliente deverá provar através da sua chave privada que ele é o *Subject* que consta no certificado da base da cadeia.

6 SOBRE A IMPLEMENTAÇÃO

Esta implementação do SSL foi desenvolvida a partir de alterações realizadas em duas bibliotecas: uma responsável pela geração dos certificados, desenvolvida por Jonna Partanen [14], e a outra JCSI (Java Crypto and Security Implementation) [15] responsável por implementar as APIs `javax.net.ssl`. Na primeira biblioteca as alterações fizeram-se necessárias para que se pudesse extrair de forma correta a partir do certificado a chave pública RSA do sujeito e do emissor – corrigindo o método *getIssuer* que construía somente chaves DSA a partir do formato canônico e adicionando o método *getSubjectPublicKeyRSA*. Na segunda biblioteca foram modificadas todas as referências à classe `X509Certificate` para a classe abstrata `SPKICertificate`, como também outros métodos específicos como o *verifyCertChain* que recebe um *array* de certificados como parâmetro e verifica a validade da cadeia.

Através das classes de exemplo `br.unb.cic.SSLClient` e `br.unb.cic.SSLServer`, que correspondem ao cliente e ao servidor respectivamente, é possível visualizar na Figura 2 a troca de certificados e os respectivos sujeitos.

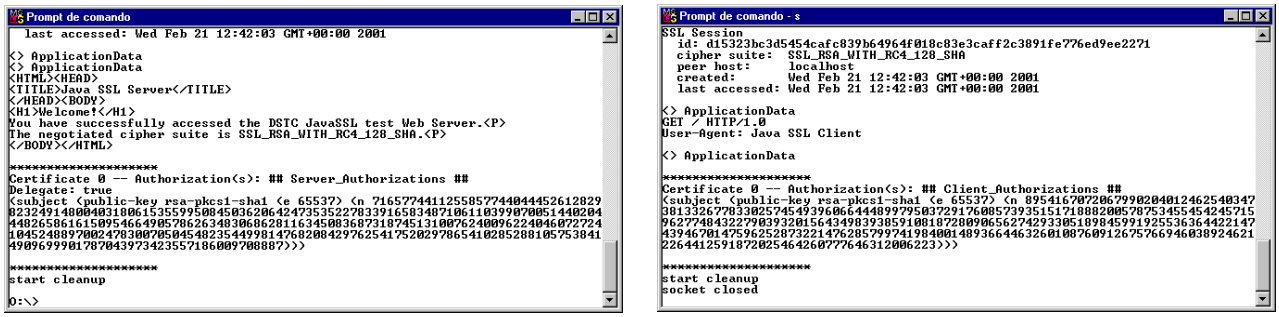

Figura 2: Execução do cliente e do servidor SSL

7 CONCLUSÃO

No trabalho de Maywah [16], parte do projeto SDSI [4], utiliza-se uma conexão SSL mas não é suportada a autenticação do servidor através de certificados SPKI/SDSI. Com a incorporação destes certificados de autorização ao protocolo SSL, esperamos contribuir para o aprimoramento da infra-estrutura de chave pública SPKI/SDSI e trazer novas perspectivas de segurança para aplicações e áreas como o comércio eletrônico.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Diffie, W., Hellman, M., *New Directions in Criptography.* IEEE Transactions on Information Theory IT-22, Novembro 1976.

[2] Konhfelder, L. , *Towards a Practical public-key cryptosystem, M.I.T. Thesis,* Maio 1978.

[3] Ellison, C. M., *SPKI Requirements, RFC2692* Fevereiro 1997.

[4] Rivest, R., Lampson, L., *SDSI – A Simple Distributed Security Infrastructure,* http://theory.lcs.mit.edu/~rivest/sdsi.html, 1996.

[5] Elien, J. , *Certificate Discovery Using SPKI/SDSI 2.0 Certificates, M.I.T. Thesis,* Maio 1998.

[6] Rivest, R. , *Can We Eliminate Certificate Revocation Lists ?,* M.I.T 1998.

[7] Ellison, C. M., *Establishing identity without certification authorities, 6th USENIX Security Symposium,* Julho 1996.

[8] Blaze M., Feigenbaum J., Lacy J., *Decentralized Trust Management, IEEE Symposium on Security and Privacy,* Maio 1996.

[9] Lampson, B., Abadi M., Burrows M., Wobber E., *Authentication in Distributed Systems: Theory and Practice,* Novembro 1992.

[10] NCSA, *A guide to understanding Discretionary Access Control is Trusted Systems,* Novembro 1987.

[11] Ellison, C. M., Rivest, R., Lampson B., Frantz, B., Thomas, B. M., Ylonen T., *SPKI Certificate Theory ,RFC2693* Setembro 1996.

[12] Netscape Corporation, *SSL 3.0 Specification,* http://home.netscape.com/eng/ssl3, Novembro 1996.

[13] Schneier B., Elisson C. , *Ten Risks of PKI,* http://www.counterpane.com/pki-risks.pdf, Janeiro 2000.

[14] Partanen J., http://www.tml.hut.fi/Research/TeSSA/Old_pages/SPKI/TeSSA-SPKI-alpha1.zip

[15] Universidade de Queensland , *Java Crypto and Security Implementation* http://security.dstc.edu.au/projects/java/jcsi.html

[16] Maywah, A. J., *An Implementation of a Secure Web Client Using SPKI/SDSI Certificates, M.I.T. Thesis,* Junho 2000.