Uma proposta para automação da administração de redes IPSec juntamente com PKIs, utilizando uma API própria.

Eduardo Fernandes Piva[1*], **Paulo Licio de Geus**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas
Caixa Postal 6176 – 13083-970 Campinas, SP

eduardo@las.ic.unicamp.br, paulo@las.ic.unicamp.br

Abstract. *This paper describes how to decrease the PKI related administrative tasks by removing the certificate renew and revocation processes, without losing the security features provided by the PKI. An API following the paper proposal was developed and is also described in this paper.*

Resumo. *Uma das dificuldades em se integrar uma PKI a um framework é a dificuldade em se encontrar APIs criptográficas que facilitem integração das aplicações deste framework a uma PKI, utilizando protocolos bem definidos. Este artigo descreve como diminuir as tarefas administrativas de uma infraestrutura de chaves públicas, removendo a necessidade de se administrar manualmente a renovação e revogação de certificados para ambientes com IPSec, utilizando a API desenvolvida neste projeto, com o objetivo de facilitar a integração e desenvolver soluções com PKIs, utilizando as especificações da IETF PKIX Workgroup.*

1. Introdução

As infraestruturas de chaves públicas[Loyd, 1999, Xenitellis, 2000] têm sido incorporadas em diversos sistemas onde a autenticação de entidades finais e a utilização dos serviços criptográficos de chaves assimétricas são necessárias, causando uma demanda crescente por ferramentas que auxiliem na administração e manutenção de tais certificados e chaves.

Um dos sistemas que têm tirado proveito dos certificados digitais é o IPSec[Frankel, 2001, Kent and Artkinson, 1998], fazendo o uso destes certificados ao estabelecer uma associação de segurança ISAKMP[Maughan et al., 1998]. Conforme será ilustrado posteriormente, a solução proposta para os sistemas IPSec atuais é a geração e manutenção destes certificados manualmente, ou seja, existe um administrador responsável por manter em segurança a chave privada da CA, e distribuir para cada máquina da rede o seu respectivo certificado, assim como o certificado da CA, utilizado para verificar a validade dos certificados recebidos pela rede, através de outras máquinas. Toda a administração de certificados (renovação e revogação principalmente) deve ser realizada manualmente por um administrador.

*Projeto financiado pelo CNPq.

Os aplicativos desenvolvidos utilizando a API criada fizeram com que estas tarefas administrativas fossem reduzidas, utilizando protocolos bem definidos para realizar as renovações de certificados e eliminar a necessidade de revogação de certificados para certos ambientes.

2. Caracterização do problema

A utilização de IPSec juntamente com certificados X509[Housley and Ford, 1999] é feita, nas implementações analisadas, registrando-se um certificado de uma CA e um certificado único por máquina, juntamente com sua chave privada, em todas as máquinas de uma rede que forem utilizar o IPSec. Quando duas máquinas necessitam estabelecer uma associação de segurança, elas irão primeiro negociar uma ISAKMPSA[1], autenticando e criando uma chave simétrica de sessão utilizando seus respectivos certificados digitais.

A adoção de certificados X509 aumenta a segurança do sistema, principalmente na autenticação das máquinas conectadas na rede IPSec, porém temos automaticamente mais duas tarefas administrativas que precisam ser realizadas, para manter a segurança e funcionalidade do ambiente:

- **Renovação de certificados:** Os certificados X509 têm um tempo de vida em que eles são válidos; após este tempo um novo certificado deve ser utilizado.
- **Revogação de certificados:** Eventualmente precisamos revogar um certificado, quando precisamos remover a confiança que todos os clientes que utilizam a PKI têm com o cliente que é dono do certificado a ser revogado. Isso é preciso em casos em que a chave privada referente a um certificado é roubada, durante uma invasão, por exemplo.

3. Soluções existentes

Tanto a renovação como a revogação de certificados são serviços oferecidos por uma PKI, portanto o primeiro passo tomado na resolução deste problema foi a análise de viabilidade de implantação das soluções existentes de PKI, e na tabela 1 relacionamos os softwares que oferecem algum serviço de PKI com os protocolos suportados e seu objetivo (API para desenvolvimento ou software para usuário final).

Abaixo segue uma listagem das implementações estudadas, com suas principais características e qualidades.

- **OpenCA:**[OpenCA, 2004]
  O projeto OPENCA tem como objetivo ser uma implementação completa e funcional de uma PKI, provendo os protocolos mais utilizados. Um dos objetivos de sua implementação é de ser de fácil configuração e manutenção, visando o usuário final.
- **Oscar:**[Oscar, 2003]
  O OSCAR é uma implementação de uma API com o intuito de poder ser utilizada como base para a implementação de uma PKI funcional. Ele utiliza os padrões PKIX[Workgroup, 2004] e PKCS[RSASecurity, 2004b]. É um projeto que implementa suas próprias funções criptográficas.

[1] Associaç ˜ao de seguranç a entre duas IKEs[Harkins and Carrel, 1998] utilizadas pelo IPSec.

Tabela 1: As colunas marcadas com X significam que o software contém a funcionalidade rotulada pela coluna, as marcadas com - significam que a informação não estava disponível e os espaços em branco são as funcionalidades ausentes.

	X509	PKCS#10	RFC 2510	API dispon´ıvel	Interface com usu´ario fi nal
OpenCA	X				X
Oscar	X			X	
OpenSSL	X	X		X	
pyCA	X	X			X
Jonah	X	-	-		X
CSP	X	X			

- **OpenSSL:**[OpenSSL, 2004]
  API criptográfica de uso geral, utilizada amplamente por outros projetos. É um dos projetos pioneiros de implementações criptográficas livres, e inicialmente tinha como objetivo ser uma biblioteca para SSL/TLS, mas hoje implementa diversas outras funções criptográficas, como criação e assinatura de certificados X509.
- **pyCA:**[pyCA, 2004]
  Foi criado com o objetivo de facilitar a criação e manutenção de uma CA. Utiliza os aplicativos da OPENSSL, fornecendo uma interface amigável para as funcionalidades presentes na OPENSSL. Atualmente a ferramenta foi descontinuada, sendo que o autor apenas se compromete a atualizar o pacote com correções enviadas por voluntários.
- **CSP:**[CSP, 2004]
  Tem o mesmo objetivo do PYCA, que é o de desenvolver uma interface amigável para manutenção de uma mini-CA, utilizando a OPENSSL como biblioteca criptográfica. Seu diferencial é a linguagem utilizada. Enquanto o PYCA é escrito em python, o CSP é escrito em Perl.
- **Jonah:**[Jonah, 2004b]
  Implementação desenvolvida pela IBM que tem como objetivo implementar os padrões PKIX. Apesar de sua importante contribuição para os padrões PKIX, a distribuição deste pacote é restrita pois utiliza bibliotecas proprietárias com restrições de distribuição e exportação, e portanto não tivemos acesso a essas bibliotecas.

Podemos distingüir duas classes de software: as APIs criptográficas e as implementações de PKIs propriamente ditas. Todas as APIs criptográficas listadas na tabela 1 implementam certificados digitais no formato X509, seguindo as RFCs definidas pelo PKIX Workgroup, mas implementam os pedidos de certificado conforme a PKCS#10, que não segue os padrões PKIX.

As PKIs com interface com o usuário final são baseadas nas APIs criptográficas livres existentes, e portanto utilizam os mesmos protocolos que estas bibliotecas. Elas são de fácil uso, visando sempre o usuário final e administrador de uma PKI, tendo sempre como objetivo principal a criação de certificados, emissão de pedido de certificado e revogação de certificados para entidades humanas (pessoas).

Dos softwares listados na tabela 1, dois deles apresentam alguns problemas. O

primeiro é o OSCAR, cuja informações fornecidas são antigas, visto que atualmente[2] sua página web está inacessível por motivo indeterminado. O segundo é o JONAH, uma implementação freewere[3] financiada pela IBM, e que por restrições de exportações de suas bibliotecas criptográficas, está disponível para download somente para residentes dos Estados Unidos da América, motivo pelo qual não foi possível identificar qual formato (PKCS#10[RSASecurity, 2004a] e/ou RFC2510[Addams, 1999]) é utilizado.

4. Motivação

Na seção anterior listamos as PKIs abertas mais utilizadas[Xenitellis, 2000, Jonah, 2004a], na sua maioria disponíveis para download. Uma característica importante de todas as implementações existentes é que todas utilizam, para pedido de certificado, o formato PKCS#10, enquanto que o PKIX Workgroup da IETF mantém todo um conjunto de normas para gerenciamento de uma PKI.

Uma solução para a eliminação das tarefas administrativas citadas na seção 2 é a utilização de algum mecanismo para renovação automática de certificados, em um curto intervalo de tempo (algumas poucas horas). Fazendo o uso desta técnica, podemos eliminar a necessidade da existência de uma lista de certificados revogados, visto que esta lista contém um intervalo de publicação (normalmente algumas horas), e podemos renovar os certificados em um período de tempo um pouco menor que o período que iríamos emitir listas de revogação de certificados.

Desta maneira, quando necessita-se revogar o certificado, tudo o que precisa-se fazer é reiniciar uma máquina na PKI, ou seja, cria-se um novo certificado para esta máquina. Como o período de validade dos certificados são curtos, o certificado que queremos revogar será invalidado automaticamente, após vencer.

A renovação automática de certificados deve ser feita preferencialmente sem intervenção humana. A RFC2510 do PKIX Workgroup define um protocolo completo para gerência de certificados, definindo inclusive formato de mensagens que possibilitam a atualização de certificados automaticamente por uma entidade final, apenas comprovando sua identidade e opcionalmente comprovando a posse da nova chave privada[4]. Porém, todas as implementações livres de PKI fornecem apenas o suporte às mensagens no formato PKCS#10, e este formato define apenas como uma mensagem deve chegar à uma PKI, não definindo nenhum formato e método de entrega da resposta deste pedido, juntamente com seu novo certificado.

Os sistemas livres atuais utilizam o formato PKCS#10, pois este, além de ser uma especificação mais simples[5], é adequado as principais necessidades de PKI atuais, já que são utilizados principalmente para emitir certificados para pessoas, e portanto a entrega da

[2]Em març o de 2004.

[3]Freeware ´e um termo utilizado pela pr´opria IBM, apesar de o c´odigo do Jonah estar dispon´ıvel para download.

[4]A prova de posse da nova chave privada ´e necess´aria apenas quando a abordagem utilizada ´e a de geraç ˜ao de chaves pelas entidades fi nais, e mesmo assim n˜ao e obrigat´oria.

[5]Enquanto o PKCS#10 defi ne um formato de pedido de certifi cado, a RFC2510 defi ne todo um protocolo para gerencia de uma PKI, incluindo renovaç ˜ao, revogaç ˜ao, prova de posse de chave privada, entre outras operaç ˜oes.

resposta do pedido de certificado e renovação de certificado pode ser realizada de maneira não programática (email e WEB, por exemplo).

Como não existia nenhuma implementação que seguisse à risca as normas do IETF para pedidos e renovações de certificados, pois a maioria se utiliza do formato PKCS#10 para pedidos de certificados, iniciamos um estudo da viabilidade de se implementar uma API para a criação de mensagens codificadas no formato DER que fossem compatíveis com as definições ASN1 das RFC 2510, do IETF PKIX Workgroup, possibilitando a renovação de certificados automaticamente.

Estudando a implementação das APIs criptográficas disponíveis, notamos que todas implementam funções ASN1 próprias, conforme as suas necessidades. Esta abordagem por parte destas APIs torna muito difícil a implementação de novos padrões e extensões pois, para cada novo tipo de mensagem necessária, precisa-se escrever uma nova função para a codificação e decodificação desta mensagem no formato DER. Além de ser trabalhosa a criação destas novas rotinas de codificação e decodificação de mensagens, há uma tendência em não se manter documentação sobre as funções ASN1 implementadas por estas bibliotecas, pois estas funções são de uso exclusivo dos desenvolvedores, e estes adotam como prioridade a documentação da API criptográfica.

Em vista dos problemas acima, a decisão tomada foi a de se implementar uma API para a criação de mensagens codificadas conforme define a RFC2510, com o objetivo de facilitar a integração de aplicações a uma infra-estrutura de chaves pública e também de se implementar uma PKI funcional, seguindo os mesmos padrões. Esta decisão foi tomada porque foi notado que não existe nenhuma API para comunicação com PKIs, por isso se torna tão difícil a integração de aplicativos já existentes com uma PKI, tornando inviável ou difícil a utilização desta tecnologia.

5. Implementação

Foi implementada uma API que possibilitasse criar uma PKI com suporte a renovação de certificados, um cliente que pudesse renovar certificados e um cliente para criar novos certificados. O objetivo inicial era o de se conseguir uma PKI e um cliente funcional, compatíveis com as RFCs, sem preocupação com relação ao desempenho ou à segurança desta PKI.

Devido a falta de bibliotecas que implementassem os formatos de mensagem ASN1/DER, a linguagem de programação utilizada foi C e para a codificação das mensagens no formato ASN1/DER foi utilizada a biblioteca LIBTASN1[Libtasn1, 2004], mantida em conjunto com o projeto GNUTLS[GNUTLS, 2004]. Para as funções criptográficas para assinatura e certificações digitais foi utilizada a OPENSSL.

A implementação da API para criação de pedido de certificado, respostas para um pedido de certificado e criação destes certificados ficou em um total de 1624 linhas de código. A implementação de um servidor concorrente para aceitar as requisições de renovação de certificado ficou em 913 linhas, e a criação do cliente para requisição de certificados ficou com um total de 309 linhas. Todo o projeto totalizou 2846 linhas de código, com comentários. Para a decisão de quais bibliotecas o sistema iria utilizar, diversos aplicativos para teste e validações de tais bibliotecas foram utilizados, totalizando

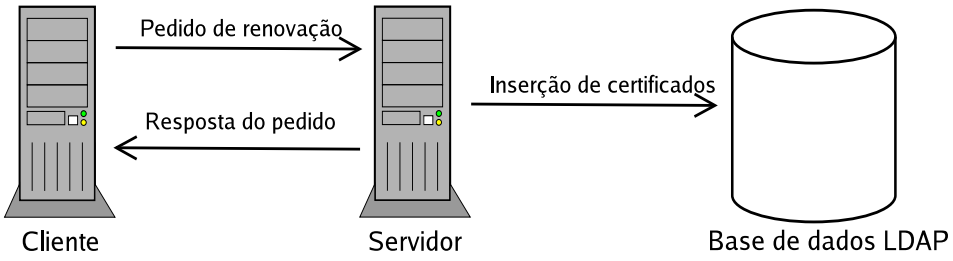

Figura 1: Funcionamento típico de uma renovação de certificado quando o pedido de certificado é válido.

2660 linhas de códigos para testes, com comentários.

A API para codificação de mensagens no formato PKIX foi dividida em dois módulos: um que implementa as funções criptográficas e outro que implementa a codificação de mensagens. Esta separação foi necessária não somente para se garantir uma melhor organização do projeto, mas também para se resolver conflitos entre as bibliotecas OPENSSL e LIBTASN1.

A RFC2510 foi utilizada como padrão para se implementar as APIs, e como o objetivo do projeto era de se obter uma API funcional, algumas decisões foram tomadas com relação a este padrão:

- **Geração de chaves:** A geração de chaves é realizada sempre no cliente. Apesar de a especificação permitir que a chave seja gerada no servidor, a abordagem de se gerar chaves no cliente é mais segura, pois a chave privada gerada pelo cliente não trafega na refe, e também torna o sistema mais escalável, pois a geração de chaves é uma operação pesada e será distribuída entre as diversas estações da rede, e não centralizada em um servidor.
- **Respostas aos pedidos:** Conforme a especificação, a resposta a um pedido enviado para uma PKI não é necessariamente instantânea. Uma mensagem de resposta pode conter uma referência para uma consulta futura, fazendo com que o cliente tenha que consultar futuramente a PKI para obter a resposta. Esta abordagem é opcional e não foi seguida, e as respostas são sempre instantâneas, pois não há a necessidade nem interesse de a operação de atualização ser off-line.
- **Posição da CA e RA:** As atualizações de certificados são sempre requisitadas para uma RA, e esta encaminha este pedido para a CA, após validar a mensagem. A abordagem utilizada foi a de se unir a CA e a RA, pois não há necessidade delas serem separadas neste cenário[6], e a especificação permite unir estas duas entidades.

A figura 1 ilustra como é o funcionamento geral de uma renovação de certificados, e detalhes de implementação do cliente e do servidor serão explicados nas subseções seguintes.

[6]A id´eia de se separar uma RA e uma CA ´e a de se adquirir uma seguranç a extra, fazendo que apenas pedidos j´a autenticados pela RA sejam encaminhados para a CA, e que apenas as RAs troquem mensagens com as CAs. No nosso cen´ario isso n˜ao ´e desejado, pois queremos respostas instantˆaneas.

5.1. Funções implementadas pela API

Algumas funções implementadas pela API serão abordadas nesta sub-seção. Nem todas as funções serão ilustradas neste artigo, pois este é o papel do manual de utilização da API, a ser distribuido junto com seu código fonte. Seguem as funções:

- `void asn1_init(const char *asn1file)`

  Inicializa a estrutura interna da biblioteca, dado um arquivo com definições ASN1/DER.
  - **asn1file**: Caminho do arquivo com as definições asn1 se encontra.

- ```
  dercode *asn1_pkix(void *pkey,
                const unsigned char *sender,
                const unsigned char *recipient,
                void *p_arg, void *e_arg,
                void p_func(void *, void *struc),
                void e_func(void *, void *struc))
  ```

  Cria uma estrutura codificada no formato DER/ASN1, segundo a especificaçõa do PKIX Workgroup.
  - **sender**: Remetente do pedido de renovação de certificado. Tipicamente o nome da máquina.
  - **recipient**: Destinatário do pedido de renovação de certificado. Tipicamente o nome da máquina onde o serviço de renovação de certificado ocorre.
  - **p_arg**: Argumento utilizado pela função de processamento do corpo da mensagem.
  - **e_arg**: Argumento utilizado pela função de processamento das informações extras da mensagem.
  - **p_func**: Função utilizada para efetuar o processamento do corpo da mensagem. Na implementação atual estas funções podem ser: asn1_certreqmessage e asn1_certrepmessage.
  - **e_func**: Função utilizada para efetuar o processamento das informações extras da mensagem. Na implementação atual estas funções podem ser: asn1_extra_null.

- ```
  void asn1_decode_pkix(dercode *der,
                        void *arg,
                        unsigned char *sender, int *slen,
                        unsigned char *recipient, int *rlen)
  ```

  Decodifica uma estrutura no formato DER/ASN.1, verificando sua assinatura e validade.

  - **der**: Mensagem no formato DER/ASN.1
  - **arg**: Estrutura onde será armazenada informações que foram decodificadas. Esta estrutura pode ser, na implementação atual, a estrutura x509_req_info ou x509_rep_info.
  - **sender**: Vetor contendo o remetente da mensagem.

  - **slen**: Inteiro contendo o tamanho do remetente da mensagem.
  - **recipient**: Vetor contendo o destinatário da mensagem.
  - **rlen**: Inteiro contendo o tamanho do destinatário da mensagem.
- `void asn1_extra_null(void *arg, void *stru)`

  Função utilizada para processar as informações extras de um pedido de renovação de certificado. Esta função inutiliza as informações extras de um pedido de renovação de certificados, porém mantendo o pedido no padrõa PKIX.

  - **arg**: Argumento utilizado internamente pela API.
  - **stru**: Argumento utilizado internamente pela API.
- `void asn1_certrepmessage(void *arg, void *stru)`

  Função utilizada para processar as informações do corpo do pedido de renovação de certificado. Esta função faz com que o corpo utilizado seja o corpo referente à resposta do pedido de renovação de certificado.

  - **arg**: Argumento utilizado internamente pela API.
  - **stru**: Argumento utilizado internamente pela API.
- `void asn1_certreqmessage(void *arg, void *stru)`

  Função utilizada para processar as informações do corpo do pedido de renovação de certificado. Esta função faz com que o corpo utilizado seja o corpo referente ao pedido de renovação de certificado.

  - **arg**: Argumento utilizado internamente pela API.
  - **stru**: Argumento utilizado internamente pela API.
- `void asn1_close()`

  Libera todos os recursos alocados durante a inicialização da biblioteca.

  A correta utilização desta API segue da seguinte maneira:

1. Inicializa-se as estruturas internas da API, utilizando a função asn1_init().
2. Cria-se um pedido de renovação de certificados, através da função asn1_pkix(). Como funções de montagem do corpo e extenções do pedido, são utilizadas as funções asn1_certreqmessage() e asn1_extra_null(), respectivamente.
3. Decodificação da mensagem pela entidade certificadora, utilizando a função asn1_decode_pkix(). Durante esta decodificação, a assinatura da mensagem é verificada.
4. Processamento dos dados extraídos da mensagem.
5. Criação de uma resposta ao pedido de renovação de certificados, pela entidade certificadora, através da função asn1_pkix(), utilizando as funções asn1_certreqmessage() e asn1_extra_null() para a criação do corpo e informações extras de uma mensagem, respectivamente.
6. Decodificação da mensagem pelo cliente, utilizando a função asn1_decode_pkix().
7. Atualização do certificado do cliente, após validação dos dados recebidos na resposta referente ao pedido de renovação de certificado.

5.2. Implementação do cliente

O papel do cliente é o de renovar o certificado de uma máquina, de maneira transparente. Um aplicativo de nome up-cert foi desenvolvido, que utiliza a API desenvolvida para gerar um pedido de renovação de certificado válido.

Uma renovação de certificados consiste basicamente de uma mensagem com informações do cliente, nome do servidor a ser contactado e sua nova chave pública, tudo isso assinado com a chave privada de seu certificado anterior, que ainda deve ser válido, mas que será substituído por um novo certificado.

Após o envio desta mensagem, uma mensagem de resposta da CA é esperada, e nesta resposta temos o status do pedido de renovação e, caso o status seja de aprovação, o novo certificado é retornado para o cliente.

Caso a operação seja concluída com sucesso, sua nova chave privada e certificado são colocados em um local pré-configurado, caso contrário eles são apagados.

5.3. Implementação do servidor

O papel do servidor neste contexto é o de validar os pedidos de certificados, retornando ao cliente certificados assinados ou então mensagens de erro, avisando o cliente caso um erro ocorra. A validação do certificado consiste em duas etapas:

1. Verificação de integridade da mensagem e da validade da assinatura da mensagem enviada pelo cliente.
2. Verificação da validade das informações fornecidas pelo cliente, conferindo com informações já armazenadas no servidor LDAP.

O servidor também precisa publicar os certificados emitidos em uma base de dados LDAP. Esta base contém informações de todos os usuários que dispõem de certificados, e pode ser consultada por qualquer entidade quando necessário.

O local para armazenagem destes certificados na base de dados LDAP é configurável; a única restrição quanto a isso é que em um dos objectClass dos nós onde residirão os certificados, o atributo *userCertificate* seja pelo menos opcional.

A única diferença desta mensagem com a de uma mensagem de erro é que em uma mensagem de erro não temos um certificado, e o código retornado é de erro e não de sucesso.

5.4. Ferramentas de apoio

Além da atualização dos certificados, é necessária uma maneira simples de inicializar uma estação com um certificado novo manualmente, para realizar a configuração inicial de uma máquina.

Um script de nome pki-add-cert.sh foi criado, e este recebe como parâmetro os locais onde o certificado e chave deverão residir. Este script consulta a base de dados da PKI, verifica o número serial que está em uso atualmente e cria um certificado com um número serial maior do que o atual. Após isso, o número serial da base é incrementado e o certificado é inserido na base.

Futuramente este script deve ser substituído pela operação de inicialização de certificados segundo o formato definido na RFC2510, para que esta operação seja compatível com a especificação do PKIX Working Group.

6. Resultados obtidos

Com a implementação realizada acima foi possível atualizar automaticamente certificados x509 de estações clientes que utilizavam comunicação via IPSec. O formato das mensagens para renovação de certificados foi implementado 100% conforme os padrões PKIX, utilizando a API NOME implementada especialmente para este fim.

Outro resultado importante, proveniente devido o desenvolvimento da API NOME, foram as correções na biblioteca LIBTASN1 utilizada para a codificação das mensagens no formato DER. Cerca de dez correções foram realizadas, proporcionando uma biblioteca de maior qualidade para o desenvolvimento de aplicações que necessitem utilizar o formato binário DER, já que esta biblioteca foi a melhor encontrada disponibilizada livremente.

Uma utilização completa das ferramentas desenvolvidas segue da seguinte forma: primeiro, iniciamos um certificado para uma máquina recém criada (os parâmetros adicionais como distinguished name, estão omitidos pois estamos utilizando os valores padrões, apenas para teste):

```
# ./pki-add-cert.sh pki.las.ic.unicamp.br
Criando chave privada.
Criando certificado X509.
Inserindo certificado na base de dados.
```

O certificado é criado e então transferido juntamente com sua chave privada para a máquina que utilizará este certificado, caso este seja criado fora dela. Após isto, podemos a qualquer momento atualizar o certificado desta máquina, sendo que esta operação deve ser preferencialmente realizada automaticamente (via crontab por exemplo):

```
# ./up-cert -H "127.0.0.1" -p 6543 -k chave.pem\
 -d "C=BR,CN=teste.las.ic.unicamp.br,L=Campinas,\
 OU=LAS,O=IC,E=eduardo@las.ic.unicamp.br,\
 ST=Sao Paulo" -c "CN=quasar.las.ic.unicamp.br,\
 L=Campinas,C=BR,OU=LAS,O=IC,\
 E=eduardo@las.ic.unicamp.br,ST=Sao Paulo"\
 -S teste.las.ic.unicamp.br -D\
 quasar.las.ic.unicamp.br\
 -a pkix.asn -X new-cert.pem

 .++++++
 ..............................++++++
```

```
 Resposta de pedido de certificação recebida.
 Certificado escrito em disco.
```

Se o servidor não estiver executando em modo “daemon”, ele emitirá uma confirmação para cada certificado renovado ou para cada pedido de renovação negado, na tela:

```
# ./pkid
 Certificado para C=BR,\
 CN=teste.las.ic.unicamp.br,\
 L=Campinas,OU=LAS,O=IC,\
 E=eduardo@las.ic.unicamp.br,\
 ST=Sao Paulo Renovado com sucesso.
```

7. Conclusões

Conseguimos obter uma API para a criação de mensagems no formato especificado na RFC2510 para gerência de uma PKI. Esta API pode ser estendida no futuro e facilitar a integração de outros serviços ao ambiente de uma PKI, já que a maior dificuldade encontrada pelos desenvolvedores para a integração de aplicações se deve ao fato de que todas as mensagens e protocolos precisam ser implementados por completo, já que não existe nenhuma biblioteca que facilite a integração.

Existem diversas extensões a serem implementadas nesta API, pois atualmente ela suporta apenas a renovação automática de certificados. A renovação automática de certificados foi implementada visando diminuir a manutenção de certos ambientes (em especial os que operam IPSec), sem perder qualidade e segurança nos serviços oferecidos. As futuras extensões como revogações de certificados e certificados cruzados, vão possibilitar que a integração funcional em ambientes com uma nova PKI seja realizada sem dificuldades, devido a sua fácil acoplagem.

Referências

Addams, C. e Farrel, S. (1999). Internet x.509 public key infrastructure certificate management protocols. Internet Engineering Task Force, RFC 2510.

CSP (2004). Csp - certificate service provider. Disponível em World Wide Web (Março de 2004): <http://devel.it.su.se/projects/CSP/>.

Frankel, S. (2001). Demystifying the ipsec puzzle.

GNUTLS (2004). The gnu transport layer security library. Disponível em World Wide Web (Fevereiro de 2003): <http://www.gnu.org/software/gnutls/documentation/libasn1/asn1.html>.

Harkins, D. and Carrel, D. (1998). The internet key exchange (ike). Internet Engineering Task Force, RFC 2409.

Housley, R. and Ford, W. (1999). Internet x.509 public key infrastructure certificate and crl profile. Internet Engineering Task Force, RFC 2459.

Jonah (2004a). Higher education pki technical activities group. Disponível em World Wide Web (Março de 2004): <http://middleware.internet2.edu/hepki-tag/opensrc.html>.

Jonah (2004b). Mit jonah pkix freeware distribution home page. Disponível em World Wide Web (Março de 2004): <http://web.mit.edu/pfl/>.

Kent, S. and Artkinson, R. (1998). Security architecture for the internet protocol. Internet Engineering Task Force, RFC 2401.

Libtasn1 (2004). Asn.1 structures parser. Disponível em World Wide Web (Fevereiro de 2003): <http://www.gnu.org/software/gnutls/documentation/libasn1/asn1.html>.

Loyd, A. . (1999). Understanding public-key infrastructure.

Maughan, D., Schertler, M., Schneider, M., and Turner, J. (1998). Internet security association and key management protocol (isakmp). Internet Engineering Task Force, RFC 2408.

OpenCA (2004). Disponível em World Wide Web (Março de 2004): <http://www.openca.org/>.

OpenSSL (2004). Openssl - the open source toolkit for ssl/tls. Disponível em World Wide Web (Março de 2004): <http://www.openssl.org>.

Oscar (2003). Disponível em World Wide Web (Fevereiro de 2003): <http://oscar.dstc.qut.edu.au>.

pyCA (2004). pyca - x.509 ca. Disponível em World Wide Web (Março de 2004): <http://www.pyca.de>.

RSASecurity (2004a). Pkcs#10 - certification request syntax standard. Disponível em World Wide Web (Março de 2004): <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-10>.

RSASecurity (2004b). Public-key cryptography standards. Disponível em World Wide Web (Março de 2004): <http://www.rsasecurity.com/rsalabs/pkcs/index.html>.

Workgroup, P. (2004). Public key infrastructure (x509) (pkix). Disponível em World Wide Web (Março de 2004): <http://www.ietf.org/html.charters/pkix-charter.html>.

Xenitellis, S. (2000). Open source pki book. Disponível em World Wide Web (Março de 2004): <http://ospkibook.sourceforge.net/>.