Relatório de Trabalho de Implementação - Segurança Computacional

Eduardo de Azevedo dos Santos
14/0136967 Universidade de Brasília (UnB)
Dep. Ciência da Computação
Brasília, Brasil

Dezembro de 2020

1 Introdução

O sistema criptográfico RSA é um dos primeiros dos sistemas que utilizam chave pública, sendo amplamente utilizado para transmissão segura de dados. Um usuário do RSA cria e publica uma chave (a chave pública) baseada em dois números primos grandes, junto com um valor auxiliar. Os números primos devem permanecer secretos. Isso se reflete no conceito de chave pública e chave privada, onde a chave pública é o par (n, e) e a chave privada a tripla (p, q, d), onde p e q são números primos grandes, $n = pq$ e $d = (1/e)(mod\phi(n))$.

A cifragem RSA ocorre através de potenciação modular, como na seguinte equação, onde m é uma mensagem tal que $1 < m < n - 1$.A mensagem é então cifrada em uma mensagem cifrada c.

$$m^e \equiv cmodn$$

2 Objetivos

Almejou-se construir um programa o qual seria capaz de gerar e verificar assinaturas RSA. Deveria possuir uma geração de chaves de tamanho mínimo de 1024 bits, com a assinatura da mensagem utilizando de cálculo de hashes através da função de hash SHA-3. O programa também deveria ser capaz de ler o documento assinado, decifrar a assinatura e realizar a verificação quanto ao hash do arquivo.

3 Implementação

Para realizar a implementação do trabalho, foi utilizada a linguagem C++ e a biblioteca OpenSSL. Foi implementado tratamento de erros utilizando o sistema de mensagens de erro da OpenSSl e almejou-se realizar a programação utilizando passagem de parâmetros e da maneira mais modular possível. O projeto pode ser encontrado em um repositório do GitHub do autor (link: `https://github.com/gerudoking/ProjetoSegcomp`), de onde pode ser clonado. Para realizar-se a compilação, foi utilizado o compilador GCC(no sistema operacional Windows, com o WSL Ubuntu 20.04 LTS) com a seguinte linha de comando pelo terminal(estando na pasta do repositório):

```
g++ Assinaturas.cpp -o Assinaturas -lssl -lcrypto
```

Foi utilizado a senha **42** para autorizar a cifragem a ser realizada, e o tamanho escolhido para a chave foi de 2048 bytes. O programa possui a seguinte lógica de funcionamento:

- É realizada a chamada à função *main* do programa.
- Inicializa-se as variáveis a serem utilizadas durante a execução.
- O programa pede uma entrada do usuário, a qual será cifrada futuramente.
- É gerada a chave RSA privada que será utilizada na assinatura.
- A chave é salva em formato de certificado *.pem* em disco.
- O programa lê a chave e então começa a cifrar a mensagem recebida pelo usuário, fazendo uso da chave lida.
- Salva a mensagem cifrada em um arquivo *.txt* no disco.
- Executa a limpeza da memória das variáveis dinamicamente alocadas.

A função *main*, por sua vez, pode detalhar melhor o fluxo de funcionamento(o código a seguir exclui as chamadas a função printf presentes no programa para melhor leitura):

Listing 1: Função *main*

```
int main()
{
    int ret = EXIT_FAILURE;
    char *str = NULL;
    unsigned char *sig = NULL;
    size_t slen = 0;
    unsigned char msg[BUFFSIZE];
    size_t mlen = 0;
    RSA* rsa = RSA_new();
```

```
    scanf("%s", str);

    GenerateKeys(rsa);

    EVP_PKEY *key = ReadPKeyFromFile(KEYFILE);

    mlen = strlen((const char*)msg);

    SignMessage(key, msg, mlen, &sig, &slen);

    WriteSignedMessage(sig);

    OPENSSL_free(sig);
    EVP_PKEY_free(key);
    sig = NULL;

    return 0;
}
```

3.1 Métodos

Os seguintes métodos foram desenvolvidos para melhor modularizar e organizar o código do programa.

Listing 2: Métodos implementados no programa

```
EVP_PKEY *ReadPKeyFromFile(const char * fname);

void GenerateKeys(RSA* rsa);

int SignMessage(EVP_PKEY *key, const unsigned char *msg,
    const size_t mlen, unsigned char **sig, size_t *slen);

void WriteSignedMessage(unsigned char* message);

std::string getOpenSSLError(EVP_MD_CTX* mdctx,
    unsigned char **sig);
```

- O método *ReadPKeyFromFile* lê uma chave privada de um arquivo de nome *fname*, retornando um ponteiro para a chave privada.
- O método *GenerateKeys* recebe um ponteiro para um objeto RSA e possui retorno vazio.
- O método *SignMessage* recebe o ponteiro para a chave privada, uma mensagem *msg* a ser cifrada, o tamanho *mlen* da mensagem, um ponteiro de ponteiro para *sig*, a qual irá guardar a mensagem cifrada, e o tamanho *slen* desta mensagem cifrada.

- O método *WriteSignedMessage* escreve em um arquivo a mensagem *message*, possuindo retorno vazio.
- O método *getOpenSSLError* recebe informações necessárias para obter possíveis erros através da biblioteca OpenSSL, retornando uma cadeia de caracteres com o erro ocorrido.

4 Conclusão

O algoritmo RSA foi descrito em 1978, mas permanece sendo amplamente utilizado devido a sua segurança. É uma parte da vida cotidiana do mundo globalizado, e o conhecimento deste e outros algoritmo é essencial para qualquer profissional da área de computação. Com a experiência adquirida através da implementação deste projeto, o autor passa a possuir uma nova perspectiva quanto a práticas de programação no geral.