Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Segurança de Sistemas Informáticos
Guia para Aula Laboratorial 2
2º Ciclo em Engenharia Informática
2º Ciclo em Eng. Eletrotécnica e de Computadores
2º Ciclo em Matemática e Aplicações

Computer Systems Security
Guide for Laboratory Class 2
M.Sc. in Computer Science and Engineering
M.Sc. in Electrical and Computer Engineering
M.Sc. in Mathematics and Applications

Sumário
Utilização de funções de *hash* para construção de provas de trabalho. Exercícios de revisão de conceitos de criptografia de chave pública e assinaturas digitais.

Summary
Usage of hash functions to construct proofs of work. Exercises for revision of concepts concerning public key criptography and digital signatures.

Pré-requisitos:
A maior parte das tarefas enunciadas em baixo requerem a utilização da ferramenta `OpenSSL` (`http://www.openssl.org/`). Um ambiente linha de comandos robusto também facilita a execução de algumas dessas tarefas. Sugere-se, assim, o uso de uma distribuição comum de Linux, onde todas estas condições estarão provavelmente preenchidas ou pressupõe-se o acesso a um sistema com a possibilidade de instalar o *software* necessário.

1 Prova de Trabalho – *Mining Bitcoins*

Proof of Work – Mining Bitcoins

O guia laboratorial anterior contém tarefas para revisão rápida de conceitos relativos a funções resumo criptográficas, nomeadamente a *Secure Hash Algorithm 1* (SHA1). A função SHA256, pertencente a uma família de funções de *hash* criptográficas mais recente e é, de resto, usada no processo de mineração de Bitcoins. Nesta moeda criptográfica, a mineração não é mais do que o processo de procura de uma sequência de bytes com um valor de *hash* que é menor do que um determinado valor conhecido e ajustado a cada 2016 blocos. Este valor, pré-determinado, é designado por *target*.

Para efeitos desta aula, vamos considerar uma simplificação do procedimento utilizado pelo Bitcoin. Neste caso, considere que estamos apenas interessados em encontrar sequências de bytes cujo valor de *hash* começa por um determinado número de bits iguais a 0. Por exemplo, imagine que lhe era pedido para gerar vários ficheiros, até encontrar um cujos 10 primeiros bits do respetivo valor de *hash* eram iguais a 0, oferecendo-lhe 25 cêntimos por cada ficheiro que encontrasse com esta propriedade. A primeira parte deste guia laboratorial tem como objetivos desmistificar e ilustrar este processo.

Q1.: Quantos bits devolve o SHA256?
☐ 128 bits ☐ 160 bits ☐ 256 bits ☐ 512 bits
☐ Depende do tamanho do ficheiro de entrada.

Tarefa 1 *Task 1*

Considere analisar, compilar e executar o código incluído a seguir.

```c
#include <openssl/sha.h>
#include <stdio.h>

int main(int argc, char *argv[]){
   SHA_CTX sha1_ctx;
   SHA1_Init(&sha1_ctx);
   unsigned char caMD[20];
   unsigned char caByte[4];
   unsigned long lRand = atoi(argv[1]);

   caByte[3]= lRand & 0xff;
   caByte[2]= (lRand>>8) & 0xff;
   caByte[1]= (lRand>>16) & 0xff;
   caByte[0]= (lRand>>24) & 0xff;
   SHA1_Update(&sha1_ctx,&caByte,4);

   SHA1_Final(caMD,&sha1_ctx);
   int i;
   for(i = 0; i < 20; i++)
     printf("%02x",caMD[i] );
```

```
    printf("\n");
}
```

Q2.: Como se compila o código incluido antes?

- ☐ `> cc sha1sum.c -lssl`
- ☒ `> cc sha1sum.c -lcrypto`
- ☐ `> rc sha1sum.c`
- ☐ `> sudo -lssl`
- ☐ `> cc sha1sum.c -lssl`

Q3.: O que faz o código anterior?

- ☐ Não percebo o que faz, nomeadamente aquela parte que tem os símbolos >>.
- ☐ Debita aqueles símbolos mostrados nos monitores dos filmes matrix, todos bacanas.
- ☐ Não percebo o que faz, principalmente porque não percebo o que significa `0xff`.
- ☒ Calcula o valor de *hash* SHA1 para uma sequência pseudo aleatória com 4 bytes.
- ☐ Calcula o valor de *hash* SHA1 do valor que é passado como parâmetro aquando da invocação do programa.

Tarefa 2 *Task 2*

Insira os dois blocos de código seguintes no local certo certo do programa acima mencionado de modo a que passe a tentar iterativamente novos números inteiros até encontrar um cujo valor de *hash* comece com um determinado número de bits igual a zero.

```
    int iFoundIt = 0;
    do{
```

```
      lRand++;
      if( ( caMD[0] == 0x00 )
        & ( caMD[1] == 0x00 ) )
        iFoundIt = 1;
    }while( !iFoundIt );
```

Q4.: Observando o segundo bloco de código incluído antes, de quantos bits iguais a 0 é que o procedimento vai andar à procura?

- ☐ 1 bit ☐ 2 bits ☐ 4 bits ☐ 8 bits
- ☒ 16 bits ☐ 32 bits ☐ 64 bits

Tarefa 3 *Task 3*

Corra o programa resultante (e.g., considere que o executável respetivo tinha o nome `a.out`) através da ferramenta `time` do Linux:
`> time ./a.out`.
Tente correr o programa desta forma várias vezes.

Q5.: Quanto tempo demora o programa a executar?

0,007s

Tarefa 4 *Task 4*

Altere o programa de modo a que, em vez de 2 bytes, o valor de *hash* tenha agora 3 bytes iguais a 0. No final, corra de novo o programa através de `> time ./a.out` e indique o tempo que agora demora na linha seguinte:

0,806s

Q6.: Em comparação com o anterior, quanto tempo mais demora este programa a executar?

- ☐ 2 vezes mais. ☐ 3 vezes mais.
- ☐ 4 vezes mais. ☐ 8 vezes mais.
- ☐ 16 vezes mais. ☐ 32 vezes mais.
- ☐ 128 vezes mais. ☒ 256 vezes mais.

Q7.: Sempre que se adiciona um novo bit igual a 0, qual é o decremento da performance?

- ☐ A performance passa para o dobro.
- ☒ A performance passa para metade.
- ☐ A performance passa para 1.5.
- ☐ A performance passa para o triplo.

Tarefa 5 *Task 5*

A dificuldade atual para a Bitcoin, anunciada em `https://bitcoinwisdom.com/bitcoin/difficulty`, é o equivalente a cerca de 70 bits. I.e., é preciso encontrar valores de *hash* com os primeiros 70 bits iguais a 0 para se minerarem 25BTC.
Q8.: Partindo do princípio de que precisava de aproximadamente 3 segundos para encontrar *hashes* com os primeiros 24 bits iguais a 0, de quantos dias precisaria para minerar 25BTC?

Precisaria de cerca de 25418658283 dias

Q9.: Nas condições da questão anterior, de quantos computadores iguais ao seu precisaria para calcular um valor de *hash* que valesse 25BTC em 8,6 minutos (que é o tempo médio atual para se minerar 25BTC)?

Cerca de 4.26 trillion computers computadores.

Nesta altura, deve estar a começar a perceber para que serve a prova de trabalho, mas enverede pelas seguintes questões para obter uma ideia mais nítida. **Q10.: Um mineiro de Bitcoins gasta di-**

nheiro verdadeiro no processo de mineração?
☒ Sim, gasta. ☐ Não, não gasta.

Se respondeu que não na questão anterior, atente na seguinte. **Q11.: Quem é que paga a eletricidade do mineiro, enquanto o computador deste procura valores de *hash* intensivamente?**
☐ Partindo do princípio de que não fez uma *"puxada"* ilegal, é ele próprio.
☐ Isto não tem nada a ver...
☐ São os contribuintes...

Para saber mais sobre o Bitcoin, recomenda-se a leitura do artigo *Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto*, disponível em `https://Bitcoin.org/Bitcoin.pdf`.

2 Cifras de Chave Pública

Public-key Criptography

Tarefa 6 *Task 6*

É sempre possível aplicar o modelo de ataque texto limpo escolhido adaptativamente (*adaptatively chosen plaintext attack*) ao modo livro de escola da função alçapão RSA. Esta tarefa demonstra o problema que isso pode acarretar em termos práticos.

Junto com este guia laboratorial é disponibilizado um ficheiro contendo uma chave pública, chamado `pk1.pem`. Construa e emita o comando *OpenSSL* que lhe permite formatar a chave para análise humana, transcrevendo-o para aqui:

openssl rsa -in pk1.pem -pubin -text -noout

Q12.: Qual o tamanho do módulo usado nesta chave pública?
☐ 128. ☐ 256. ☐ 512. ☒ 1024. ☐ 2048.

Q13.: O tamanho da chave parece-lhe seguro para utilizações atuais?
☐ Está quase a sair de moda, mas sim.
☐ Não, é muito grande.
☒ Não, é muito pequena.

Q14.: O que significa o acrónimo RSA?
☐ *Resource Symmetric Algorithm*.
☐ *Resource Sypher Assymetric*.
☐ Rui, Santori and Adelino.
☒ Nenhuma das anteriores. Rivest Shamir Adleman

Q15.: Num sistema criptográfico de chave pública, que chave utilizaria para cifrar uma mensagem para um(a) colega seu(ua)?
☐ A minha chave pública.
☐ A minha chave privada.
☒ A chave pública do(a) colega.
☐ A chave privada do(a) colega.

Tarefa 7 *Task 7*

O Professor usou esta chave para cifrar um número de 0 a 9, guardando o criptograma respetivo no ficheiro com o nome `numero.rsa`. O ficheiro original é constituído por 127 0s e o número. E.g.:

```
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000X
```

Q16.: Consegue descobrir o número que foi cifrado?
☒ Consigo. Ingiro doses destas todos os dias ao pequeno almoço.
☐ Não estou a ver como, se não tenho a chave privada.
☐ Não consigo.

Q17.: Qual foi o número cifrado?
Foi o ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☒ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 0 .

Sugestão: use uma combinação de comandos parecidos com:

```
> echo -n "00000...000001" > numero2.txt
> openssl rsautl -encrypt -inkey ...
-pubin -in ...  -out ...  -raw
> diff numero.rsa numero2.rsa
```

Repare na opção `-raw`, que determina o uso da cifra sem *padding*.

Q18.: É seguro usar o RSA em modo livro de escola para cifrar valores pequenos?
☐ Sim, é seguro.
☒ Não, não é seguro porque a função alçapão define um algoritmo determinístico e um atacante é sempre capaz de cifrar quaisquer mensagens. Se o espaço de mensagens for pequeno (como era o caso), o atacante pode sempre tentar cifrar todas as mensagens até obter o criptograma correspondente...

Q19.: Qual é o *padding* que é aplicado por definição quando se usa o *OpenSSL* para se cifrar algo com RSA?

☐ *Optimal Assymetric Encryption Padding (OAEP).*
☒ *Public Key Cryptography Standards (PKCS)* 1.5.
☐ Nenhum.

3 Assinatura Digital

Digital Signature

As próximas tarefas convergem para a elaboração e verificação de assinaturas digitais que usam o esquema que combina funções de *hash* criptográficas com o RSA. A primeira tarefa será a de elaboração de um ficheiro cobaia e a segunda prender-se-á, portanto, com a criação de um par de chaves para o efeito de assinatura digital.

Tarefa 8 *Task 8*

Crie o ficheiro `es-feio.txt` com um segredo cabeludo acerca do(a) seu(ua) colega do lado. Não mostre o conteúdo a ninguém.

Tarefa 9 *Task 9*

Crie um par de chaves RSA usando o comando seguinte:

```
> openssl genrsa -out pk-and-sk.pem 1024
```

Q20.: Quantos bits tem o módulo das chaves públicas e privadas geradas com o comando anterior?
☐ 512. ☐ 845. ☐ 2^{1024}. ☒ 1024. ☐ $2^{1024^{1024}}$

Q21.: Se quiser assinar o ficheiro `es-feio.txt` que chave deve utilizar?
☒ A minha chave privada.
☐ A minha chave pública.
☐ A chave pública do(a) colega a quem vou enviar o ficheiro.
☐ A chave privada do(a) colega a quem vou enviar o ficheiro.
☐ Uma chave simétrica aleatória.

Tarefa 10 *Task 10*

Escreva o comando que permite extrair a chave pública, contida no ficheiro `pk-and-sk.pem` para o ficheiro `aNum-Aluno.pem`.

```
openssl rsa -in pk-and-sk.pem -pubout -out aNum-Aluno.pem
```

Tarefa 11 *Task 11*

De forma a garantir todas as as propriedades de uma assinatura digital, o que é normalmente assinado é o valor de *hash* do documento eletrónico, e não o próprio documento. Desta forma, a assinatura também já se aplica a todo o tipo de ficheiros. Tendo isto em consideração e para que fique bem patente que foi você quem escreveu o segredo cabeludo acerca do(a) seu(ua) colega, escreva o comando `OpenSSL` que permite criar a assinatura digital do ficheiro `es-feio.txt` com a combinação MD5 e RSA.

```
openssl dgst -md5 -sign pk-and-sk.pem -out es-feio.md5-with-rsa es-feio.txt
```

Sugestão: use o comando `openssl dgst` com algumas das suas opções para levar esta empreitada a bom porto com um único comando. Chame ao ficheiro contendo a assinatura `es-feio.md5-with-rsa`.

Depois de ter construido a assinatura, verifique o conteúdo do ficheiro `es-feio.md5-with-rsa` com:

```
> hexdump es-feio.md5-with-rsa
```

Tarefa 12 *Task 12*

Considere que tinha conseguido produzir a assinatura para o ficheiro `es-feio.md5-with-rsa` com sucesso. A questão é: **Q22.: o que é que falta no comando seguinte para que a verificação seja feita com sucesso?**

```
> openssl dgst -md5 -verify pk.pem
-signature es-feio.md5-with-rsa ...
```

☐ Não falta nada.
☐ Falta o nome do ficheiro do qual queremos provar a assinatura para o qual a assinatura foi calculada.

Q23.: Depois da emissão correta do comando anterior, o que é lhe diz se uma assinatura digital está ou não está válida?
☒ Depois de emitir o comando, o `OpenSSL` escreve no ecrã que a assinatura está `OK`.
☐ O facto de conseguir ler o que aparece no ecrã depois de emitir o comando.
☐ O facto de não conseguir ler o que aparece no ecrã depois de emitir o comando.

Q24.: Quais são os *inputs* do algoritmo de verificação de uma assinatura digital?

☒ A chave pública do(a) colega.
☐ A chave privada do(a) colega.
☒ O ficheiro original.
☒ A assinatura.
☐ A sua chave privada.
☐ A sua chave pública.

Q25.: Quais são os *inputs* do algoritmo de elaboração de uma assinatura digital?

☐ A chave pública do(a) colega.
☐ A chave privada do(a) colega.
☒ O ficheiro original.
☐ A assinatura.
☒ A sua chave privada.
☐ A sua chave pública.

Tarefa 13 *Task 13*

Adultere (i.e., altere) o documento `es-feio.txt` para o qual está a verificar a assinatura, e volte a verificá-la, para obter a ideia concreta do resultado.

Q26.: O que é que o *OpenSSL* devolve desta vez?

Verification Failure

Tarefa 14 *Task 14*

Para mentes inquietas:
Verifique, usando 2 comandos `OpenSSL`, que a assinatura contida em `es-feio.md5-with-rsa` é, de facto, a cifra do valor devolvido pelo MD5 para o ficheiro `es-feio.txt` com a chave privada.

Sugestão: precisa decifrar a assinatura com o comando `> openssl rsautl -verify ...`.

Outra sugestão: é possível que o valor do *hash* que é cifrado no `OpenSSL` esteja num formato diferente de hexadecimal.

```
openssl rsautl -verify -pubin -inkey public_key.pem -in es-feio.md5-with-rsa -out decrypted.md5
```

4 Trabalho de Casa

Home Work

A próxima tarefa é para fazer em grupos de 2, como trabalho de casa ou no final de uma aula, caso tenha tempo.

Tarefa 15 *Task 15*

Investigue ou recorde a forma de cifrar e assinar mensagens/ficheiros definida pelo *Pretty Good Privacy* (PGP). Junte-se com um(a) colega e execute os seguintes passos (cada aluno(a) executa todos os passos seguintes), escrevendo os comandos nos espaços respetivos:

1. Crie 2 (dois) pares de chaves RSA (um para cifrar e decifrar e outro para fazer e verificar assinaturas digitais).
2. Troque as chaves públicas com o(a) seu(ua) colega.
3. Um dos dois cria um ficheiro com um poema de Fernando Pessoa, o outro coloca lá 2 oitavas dos Lusíadas.
4. Assine o ficheiro com a chave respetiva, usando a combinação de algoritmos `SHA1withRSA`.
5. Cifre o ficheiro à moda do PGP usando RSA e AES em modo CTR.
6. Envie o ficheiro e a assinatura ao(à) colega.
7. Decifre e verifique a assinatura do ficheiro que receber.


```
openssl dgst -md5 es-feio.txt | openssl enc -base64 -d > original.md5
```