Universidade do Minho

Escola de Engenharia

Departamento de Informática

João Miguel Barbosa Gomes

Telmo Rafael Rodrigues Remondes

Relatório Trabalho Prático

18 Junho 2013

Índice

Grupo I

1.1 Problema

O problema consiste a criptoanálise de quatro criptogramas obtidos recorrendo a cifras de substituição, shift e Vigenere. O objectivo é descobrir com qual a cifra utilizada para encriptar os criptogramas e obter o texto limpo dos mesmos.

1.2 Abordagem

Visto se saber quais os tipos de cifras usadas para a obtenção dos diferentes criptogramas o primeiro passo passou pela elaboração de programas que fossem capazes de auxiliar à decifração das respectivas cifras. Assim sendo, as capacidades dos diferentes módulos passam pela análise da frequência de caracteres, um algoritmo capaz de dado um valor efetuar o shift em todos os elementos da cifra, um algoritmos que dada uma cifra retornava o espaço entre palavras repetidas na mesma, para além de um programa capaz de fazer a substituição letra a letra de forma muito simples e maleável.

Tendo os algoritmos necessários à decifragem o passo seguinte passou por testar cada um deles nos diferentes criptogramas analisando o resultado que ia sendo obtido de forma a provar que era a cifra que cifrou o criptograma.

Cada para a detecção de cada uma das cifras foram efetuados os seguintes passos:

Shift: Primeiramente foi feita uma análise da frequência de caracteres. Tendo estes valores e sabendo-se que se trataria de um texto limpo em inglês ou francês foram comparadas as letras mais observadas nas cifras com as letras mais utilizadas em cada língua. Após a análise feita foi calculado o shift da letra respectiva, tendo o valor do shift foi executado o algoritmo de forma a fazer o shift inverso no criptograma de forma a analisar se o resultado obtido era um texto limpo. Visto ser um criptograma pequeno e visto que a frequência de caracteres podia não bater certo com a frequência de aparecimento no inglês ou francês, foram utilizadas as três letras mais usadas nas línguas e as três letras que mais apareciam no criptograma e calculado o shift para cada uma das possibilidades, juntando todas com todas, de forma a diminuir o risco de erro devido ao tamanho do criptograma.

Substituição: Para testar se o criptograma tinha sido cifrado com esta cifra foi feita uma análise da frequência dos caracteres do mesmo. Tendo a frequência de aparecimento dos caracteres do criptograma e a frequência destes nas respectivas línguas, foi feita uma aproximação entre ambos e feita a substituição na cifra. Visto a frequência não coincidir para todos a primeira abordagem foi apenas para um preenchimento inicial da tabela, que foi posteriormente sendo mudada de forma a construir palavras com sentido até obter um texto limpo.

Vigenere: De forma a tentar resolver esta cifra foi inicialmente feita uma análise do espaçamento entre os conjuntos de caracteres repetidos e construída uma tabela com os divisores da distancia entre as palavras repetidas. Tendo a tabela preenchida é observado qual o divisor que mais vezes aparece nas palavras repetidas, sendo que este poderá ser o tamanho da chave. Tendo um candidato para tamanho da chave, o criptograma é partido em strings desse mesmo tamanho, tornando o criptograma em pequenas cifras de césar. É feita então a analise de frequência das letras em cada coluna e comparadas com a frequência das letras em cada língua, sendo depois calculada a diferença entre a letra encontrada na coluna e a mais usada na língua, ou seja, o valor do shift. Neste momento é então aplicado o shift encontrado a todas as letras da coluna. Tem de se repetir o processo para cada coluna. Sendo o shift encontrado para cada coluna cada letra da chave, por exemplo se o shift for "1" a letra da chave será "A".

1.3 Resultados

Os resultados obtidos foram os seguintes:

Criptograma 1

Chave:

Cifra:

Resultado:

Criptograma 2

Chave: crypto

Cifra: Vigenere

Resultado:

ilearnedhowtocalculatetheamountofpaperneededforaroomwheniwasatschooly

oumultiplythesquarefootageofthewallsbythecubiccontentsofthefloorandceili

ngcombinedanddoubleityouthenallowhalfthetotalforopeningssuchaswindowsand

doorsthenyouallowtheotherhalfformatchingthepatternthenyoudoublethewholet

hingagaintogiveamarginoferrorandthenyouorderthepaper

Criptograma 3

Chave:

Cifra:

Resultado:

Criptograma 4

Chave: theory

Cifra: Vigenere

Resultado:

igrewupamongslowtalkersmeninparticularwhodroppedwordsafewatatimelikeb
eansinahillandwhenigottominneapoliswherepeopletookalakewobegoncommatomea
ntheendofastoryicouldntspeakawholesentenceincompanyandwasconsiderednotto
obrightsoienrolledinaspeechcoursetaughtbyorvillesandthefounderofreflexiv
erelaxologyaselfhypnotictechniquethatenabledapersontospeakuptothreehundr
edwordsperminute

Grupo II

1.4 Problema

Foram cifrados vinte textos recorrendo à cifra One-Time-Pad, com aritmética módulo vinte seis. Esta cifra é segura apenas quando usada uma vez, mas nestes vinte criptogramas foi usada a mesma chave para cifrar dois deles. O objectivo é descobrir quais os criptogramas que usam a mesma chave.

1.5 Abordagem

Para tentar resolver o problema foi feito um algoritmo que dados dois criptograma fazia a operação de xor entre eles e devolvia o resultado. O resultado obtido por este algoritmo é o xor das duas mensagens caso tivessem sido cifrados pela mesma chave.

1.6 Resultados

O resultado foi que não foi possível diferenciar quais as mensagens cifradas com a mesma chave. Embora a ideia da forma como se procedia o ataque e das limitações de cifrar duas mensagens com a mesma chave, não foi possível detectar qual a cifra resultante que era constituída pelo xor das duas mensagens.

Grupo III

1.7 Problema

O problema prende-se com o facto de mostrar que a cifra por blocos Electronic Code Book não consegue esconder os padrões do texto limpo. Para mostrar que isto é verdade, e para observar num caso real essa propriedade do ECB é pedido que seja escolhida uma imagem, retirada apenas a parte que corresponde à imagem propriamente dita, e cifrar esse bloco de informação usando o ECB, e em seguida guarda a imagem novamente. A ideia será verificar que os padrões continuam visíveis mesmo após ser cifrada.

1.8 Abordagem

Para resolver o problema teve em primeiro lugar de se estudar a forma como a imagem era construída. Visto isto, foi ler a área correspondente à imagem, converter para um array de bytes e cifrar, tendo sido cifrado foi apenas necessário voltar a converter para um array de inteiros e reconstruir a imagem.

1.9 Resultados

Como expectável os padrões continuaram visíveis na imagem quando usado o modo ECB. A titulo de experiencia usou se o modo CBC também.

Figura 1 - Normal

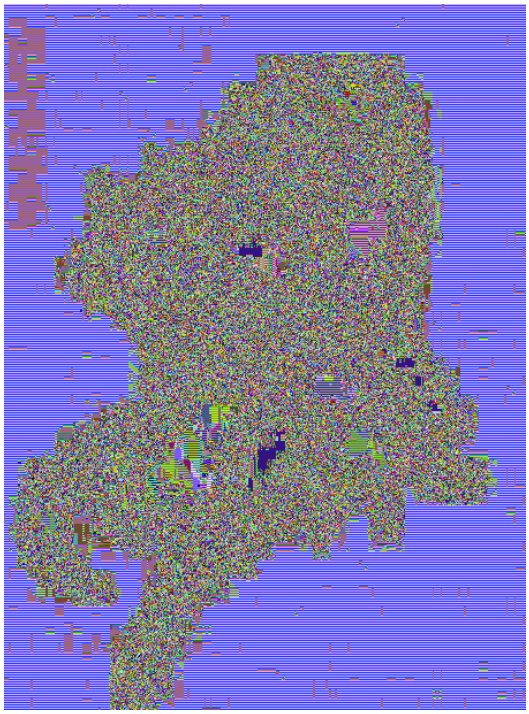


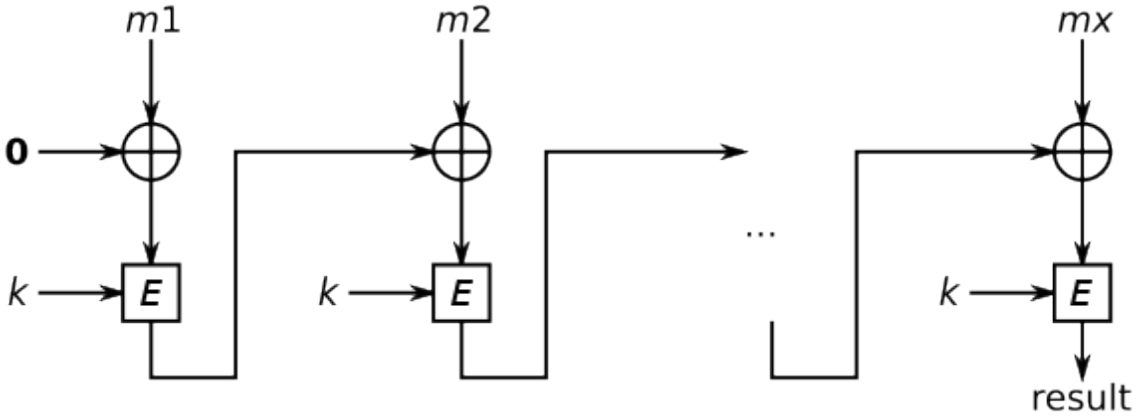
Figura 2 - Modo ECB

Figura 3 - Modo CBC

Grupo IV

1.10 Problema

O problema pedido neste guião prático é explorar as vulnerabilidades do *CBC-Mac* quando todas as tags são retornadas ou quando o IV(Initialization Vector) é aleatório em vez de fixo como acontece na generalidade dos casos. Em baixo segue a descrição do algoritmo através de um esquema, para uma mensagem *m* separada em blocos *m1 m2 m3 ... ml(n):*



1.11 Abordagem

O Objectivo é conseguir construir uma mensagem *m'* com uma tag *t*, de forma a que a função *Ver(m,t) = true*. As funções de gerar a chave, verificar a validade da mensagem e de gerar um *cbc-mac* são dadas por:

Algorithm $\mathsf{Gen}(1^n)$
$\mathsf{k} \leftarrow\!\!\$\ \{0,1\}^n$
Return k

Algorithm $\mathsf{Ver_k}(\mathsf{m},\mathsf{t})$
If $|\mathsf{m}| \neq \ell(n)\cdot n$ Return F
Return $\mathsf{Mac_k}(\mathsf{m}) = t$

Algorithm $\mathsf{Mac_k}(\mathsf{m})$
If $|\mathsf{m}| \neq \ell(n)\cdot n$ Return $\perp$
$(\mathsf{m}'_1,\ldots,\mathsf{m}'_{\ell(n)}) \leftarrow \mathsf{partition}(\mathsf{m}',n)$
$\mathsf{t}_0 \leftarrow 0^n$
For $i \in [1\ldots\ell(n)]$
  $\mathsf{t}_i \leftarrow F_{\mathsf{k}}(\mathsf{t}_{i-1} \oplus \mathsf{m}'_i)$
Return $\mathsf{t}_{\ell(n)}$

Como primeiro passo foi implementado uma função que para uma dada mensagem, devolve-se a tag correspondente à mensagem. Posteriormente a função foi modificada de forma a conter as seguintes variantes: gerar uma tag com um *IV* aleatório e outra variante que devolve-se todas as tags e não apenas a última que serve para autenticar a mensagem.

Na versão que devolve todos blocos é possível construir uma mensagem *m'* a partir da mensagem original tal que *Ver(m',t) = true.* Para isso é construído a mensagem *m'* da seguinte maneira: *m' = xor(t0,m1) ||m2||m3||...||xor(t(l(n)),m0)*. Com este ataque aparentemente trivial é fácil garantir que *Ver(m',t) = true.*

Com o IV aleatório torna-se ainda mais fácil falsificar uma mensagem e uma tag. Para isso basta construir a mensagem *m'=xor(IV,m0)||m2||m3||...||m(l(n)).*

1.12 Resultados

O resultado do programa escrito em java deverá retornar o resultado da função que verifica a validade das tags, para os dois casos temos o seguinte *output:*

```
run:
Real cbc-mac: true
Forged cbc-mac:true
Forged iv, cbc-mac:true
BUILD SUCCESSFUL (total time: 1 second)
```

Com esta experiência foi possível comprovar que com as vulnerabilidades pedidas para serem testadas, o cbc-mac torna-se altamente inseguro e muito vulnerável a ataques triviais

Grupo V

1.13 Problema

Este guião foi dividido em 3 problemas.

1.13.1 Sistema de Congruências

Na primeira parte era pedido para resolver um sistema de congruências, ou seja, um sistema com este aspecto:

$$\begin{cases} x \equiv 12 \pmod{25} \\ x \equiv 9 \pmod{26} \\ x \equiv 23 \pmod{27} \end{cases}$$

1.13.2 RSA

Na segunda parte era pedido para atacar o RSA, considerando que o n é um número de pequena ordem(31313, 18923). Para além do n, era dado também b= 4913 para o primeiro caso e b= 1261.

1.13.3 Símbolos de Jacobi

Na terceira parte era pedido para calcular símbolos de Jacobi numa primeira fase, e depois recorrer aos mesmos símbolos de Jacobi para encontrar as bases b, para as quais n é um pseudo-primo de Euler.

1.14 Abordagem

1.14.1 Sistema de Congruências

Para resolver o Sistema de congruências foi utilizado o CRT(Teorema Chinês dos Restos). Tendo um conjunto *n1,...,nk* de inteiros mutualmente coprimos e dada uma sequência de inteiros *a1,...,ak,* existe um *x* que resolve o sistema de congruências.
Para além do CRT, foi utilizado o algoritmo extendido de Euclides para auxílio aos CRT. Com o algoritmo de Euclides é possível encontrar inteiros *r* e *s* tal que *r*n+s(N=n)=1*.

1.14.2 RSA

O primeiro passo foi encontrar *p,q* tal que *n=(p*q)* em que ambos são primos. Como *n* é de pequena ordem, foi possível encontrar *p,q* em tempo útil. Em caso de *n* ser de grande ordem seria impossível realizar este passo em tempo razoável. Depois disto foi apenas aplicar o algoritmo normal RSA, calcular a função totiente de Euler e computar o expoente privado *d* tal que *d =e mod totient(n)*. Com o expoente privado, foi apenas aplicar o algoritmo de decifração normal para obter a mensagem original.

1.14.3 Símbolos de Jacobi

Para os símbolos de Jacobi foi implementada uma função de forma recursiva de forma a pode corresponder a todas as propriedades:

(a) Se n é um inteiro impar, e $m_1 \equiv m_2 \pmod{n}$, então
$\left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right)$.

(b) Se n é um inteiro impar, então
$$\left(\frac{2}{n}\right) = \begin{cases} 1 & se\ n \equiv \pm 1 \pmod 8 \\ -1 & se\ n \equiv \pm 3 \pmod 8 \end{cases}.$$

(c) Se n é um inteiro impar, então
$\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right)\left(\frac{m_2}{n}\right)$
e, em particular, se $m = 2^k * t$
$\left(\frac{m}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{t}{n}\right)$.

(d) Se m e n são inteiros impares, então
$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right) & se\ m \equiv n \equiv 3 \pmod 4 \\ \left(\frac{n}{m}\right) & nos\ restantes\ casos \end{cases}.$$

Depois foi implementar o algoritmo de Euler para calcular b, para as quais n é um pseudo-primo de Euler.

```
public static BigInteger gcd(BigInteger a, BigInteger b){
  BigInteger x = a;
  while(x.compareTo(BigInteger.ZERO) != 0){
    if(x.remainder(a).compareTo(BigInteger.ZERO)==0 && x.remainder(b).compareTo(BigInteger.ZERO)==0 ) {
    return x;
    }
    x = x.subtract(BigInteger.ONE);
  }
  return null;
}

public static boolean eulerPrime(BigInteger b, BigInteger n){
  BigInteger j = jacobi(b,n).remainder(n);
  BigInteger p = b.pow(n.intValue()-1).remainder(n);
  if(j.compareTo(p) == 0 && gcd(b,n).compareTo(BigInteger.ONE) == 0){
    return true;
  }
  else {
    return false;
  }
}

public static void eulerBase(BigInteger n){
    BigInteger i = BigInteger.valueOf(2);
    BigInteger ct = BigInteger.valueOf(0);
while (ct.compareTo(BigInteger.valueOf(5))<0){
  if(eulerPrime(i,n)){
    primes.add(i);
    ct =ct.add(BigInteger.ONE);
    i = i.add(BigInteger.ONE);
  }
}
}
```

1.15 Resultados

Para o sistema de congruências foram obtidos os seguintes resultados:

```
Val: -183213 modulus: 17550
9837
BUILD SUCCESSFUL (total time: 0 seconds)
```

No RSA foi obtido os textos limpos

"lake wobegon is mostly poor sandy soil and every spring the earth heaves up a new crop of rockspiles of rocks ten feet high in the corners of fields picked by generations of us monuments to ourindustry our ancestors chose the place tired from their long journey sad for having left the motherland behind and this place reminded them of there so they settled here forgetting that they hadleft there because the land wasnt so good so the new life turned out to be a lot like the

old exceptthe winters are worsez”

”i became involved in an argument about modern painting a subject upon which i am spectacularlyill informed however many of my friends can become heated and even violent on the subject andi enjoy their wrangles in a modest way i am an artist myself and i have some sympathy with theabstractionists although i have gone beyond them in my own approach to art i am a lumpist twoor three decades ago it was quite fashionable to be a cubist and to draw every thing in cubes thenthere was a revolt by the vorticists who drew every thing in whirls we now have the abstractionistswho paint every thing in a very abstracted manner but my own small works done on my telephonepad are composed of carefully shaded strangely shaped lumps with traces of cubism vorticism andabstractionism in them for those who possess the seeing eye as a lumpist i stand alone”

Grupo VI

1.16 Problema

Neste o grupo o problema era quebrar a crifra *El-Gamal.* Recordando o *El-Gamal*

Algorithm $\mathsf{Gen}(1^n)$
$(\mathbb{G}, q, g) \leftarrow\$\ \mathcal{G}(1^n)$
$x \leftarrow\$\ \mathbb{Z}_q$
$h \leftarrow g^x$
$\mathsf{sk} \leftarrow (x, \mathbb{G}, q, g)$
$\mathsf{pk} \leftarrow (h, \mathbb{G}, q, g)$
Return $(\mathsf{sk}, \mathsf{pk})$

Algorithm $\mathsf{Enc}(\mathsf{m}, \mathsf{pk})$
$(h, \mathbb{G}, q, g) \leftarrow \mathsf{pk}$
$y \leftarrow\$\ \mathbb{Z}_q$
Return $(g^y, \mathsf{m} \cdot h^y)$

Algorithm $\mathsf{Dec}((\mathsf{c}_1, \mathsf{c}_2), \mathsf{sk})$
$(x, \mathbb{G}, q, g) \leftarrow \mathsf{sk}$
$\mathsf{m} \leftarrow \mathsf{c}_2 / \mathsf{c}_1^x$
Return m

1.17 Abordagem

Neste guião foi necessário apenas implementar o mecanismo de decifragem do *El-Gamal,* sendo os principais parâmetros já dados.

```
public static BigInteger[] dec(Pair[] crypto) {
  BigInteger[] clearText = new BigInteger[crypto.length];
  int i = 0;
  while(i < crypto.length){
    BigInteger c1 = (BigInteger) crypto[i].getLeft();
    BigInteger c2 = (BigInteger) crypto[i].getRight();
    BigInteger s = c1.modPow(x.negate(), p);
    BigInteger m = s.multiply(c2).mod(p);
    clearText[i] = m;
    i++;
  }
  return clearText;
}
```

1.18 Resultados

O criptograma obtido

”she stands up in the garden where she has been working and looks into the distance she has-sensed a change in the weather there is another gust of wind a buckle of noise in the air and thetall cypresses sway she turns and moves up hill towards the house climbing over a low wall feelingthe first drops of rain on her bare arms she crosses the loggia and quickly enters the house”