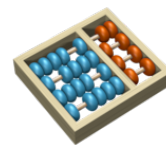




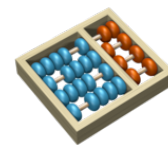
UNIVERSIDADE ESTADUAL DE CAMPINAS
Instituto de Computação – IC
MO421 – Introdução a Criptografia
Prof. Dr. Diego de Freitas Aranha



THALES EDUARDO NAZATTO

RELATÓRIO

Projeto 1 – Quebra de duas cifras encriptadas pela técnica “One-Time Pad”



1. Introdução

Desde a descoberta por Gilbert Vernan em 1917, cifras de fluxo são um método muito popular de encriptação. Quando a chave é perfeitamente randômica, essa cifra é conhecida como **“One-Time Pad”** (1). A técnica de **“One-Time Pad”** é uma técnica de encriptação que, quando é utilizada corretamente, garante perfeito sigilo da mensagem. Claude Shannon provou, na década de 1940, que a cifra é inquebrável (1) (2).

Entretanto, quando duas cifras são encriptadas pela mesma chave, essa cifra se torna perfeitamente quebrável através de uma simples manipulação. Dado textos planos $p1$ e $p2$, cifras $c1$ e $c2$ e uma chave k , e sendo $p1 \oplus k = c1$ e $p2 \oplus k = c2$ as operações de uma cifra de fluxo, é possível realizar um ataque obter os textos planos e a chave através das seguintes operações:

$$p1 \oplus p2 = c1 \oplus c2$$

$$p1 = c1 \oplus c2 \oplus p2$$

$$p2 = c1 \oplus c2 \oplus p1$$

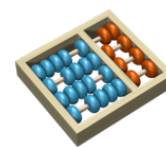
$$p1 \oplus c1 = p2 \oplus c2 = k$$

Desse modo, é possível utilizar uma técnica chamada *crib dragging*, que consiste em advinhar o posicionamento do texto munido com um dicionário de palavras, obtendo um suposto trecho do texto (*crib*) através da operação *XOR* entre as duas cifras e um trecho dentro de seu dicionário.

2. Desenvolvimento e resultados

Os textos foram obtidos **“challenge1.txt”** e **“challenge2.txt”** através da rede **“TeiaDoAranha”**, situado na sala do professor da disciplina. Como a rede é encriptada pelo padrão WEP, tornando-se fácil de ser invadida usando o programa **“aircrack”** do Linux. Para isso, é preciso monitorar a interface de rede para capturar pacotes, e com cerca de 80000 pacotes o programa é capaz de quebrar a senha.

Como a mensagem a ser decifrada estava em inglês, foram obtidos dicionários em (3), (4) e (5) com respectivamente 10000, 58000 e 194000 palavras. Para obter os textos planos resultantes, inicialmente foi feito um programa na linguagem C, mas devido a bugs e a necessidade de um



desenvolvimento mais acelerado o programa foi refeito na linguagem Java. Nele, a realização foi dividida em três fases: Manual, semi-automática e automática, detalhadas a seguir.

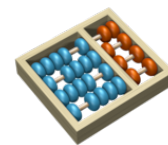
2.1 Fase manual

Nesta fase, os *cribs* foram realizados com base nos dicionários obtidos. Outros dicionários foram adicionados com a função de reproduzir expansões da frase obtida e validar se o trecho não é um falso positivo. Cada arquivo mostrava os *cribs* resultantes e as posições onde eles foram obtidos, conforme figura abaixo.

Um arquivo era gerado para cada palavra do dicionário. Como o alfabeto válido era o de letras e de pontuação, foi criada uma validação para evitar *cribs* desnecessários com base neste alfabeto. Isto poupou esforço de análise e evitou a criação de arquivos desnecessários, descartando palavras que não estariam no texto original. *Threads* foram utilizadas para que dicionários muito grandes (como os obtidos em (4) e (5)) gerassem seus *cribs* em tempo satisfatório. Após isto, era feito um trabalho manual de verificação dos *cribs*.

```
around.txt (~/MO421A/Projeto-1/cribs/A) - gedit
Abrir  Salvar
8: (u;aln
12: .xeldy
16: kodrt!
24: jroGlg
43: ban'tf
52: asllcl
58: srxxnd
59: aelung
62: arlnjb
63: aqtqhi
64: bikscb
78: fxuo(b
91: /q aje
93: .fktij
105: q;hs'n
115: wdlmn)
119: a?.sip
121: that
131: v nvs
133: .iwbnt
135: yeoeq
138: qtzqnd
139: ggkund
140: tvounh
145: m!,wbg
147: "pcvqh
149: mqpyld
150: bmcwnd
155: advnuh
156: wktnbc
157: xityii
160: mubmfe
162: ljgtc!
168: .y.sjh
170: tkycu
177: ztcoif
179: mhhwy-
183: v;jqno
184: (wkue!
193: -tmufe
Texto sem formatação  Largura da tabulação: 8  Lin 21, Col 12  INS
```

Figura 1. Detalhe de arquivo gerado pela fase manual da palavra “around” e de seu *crib* resultante “that”

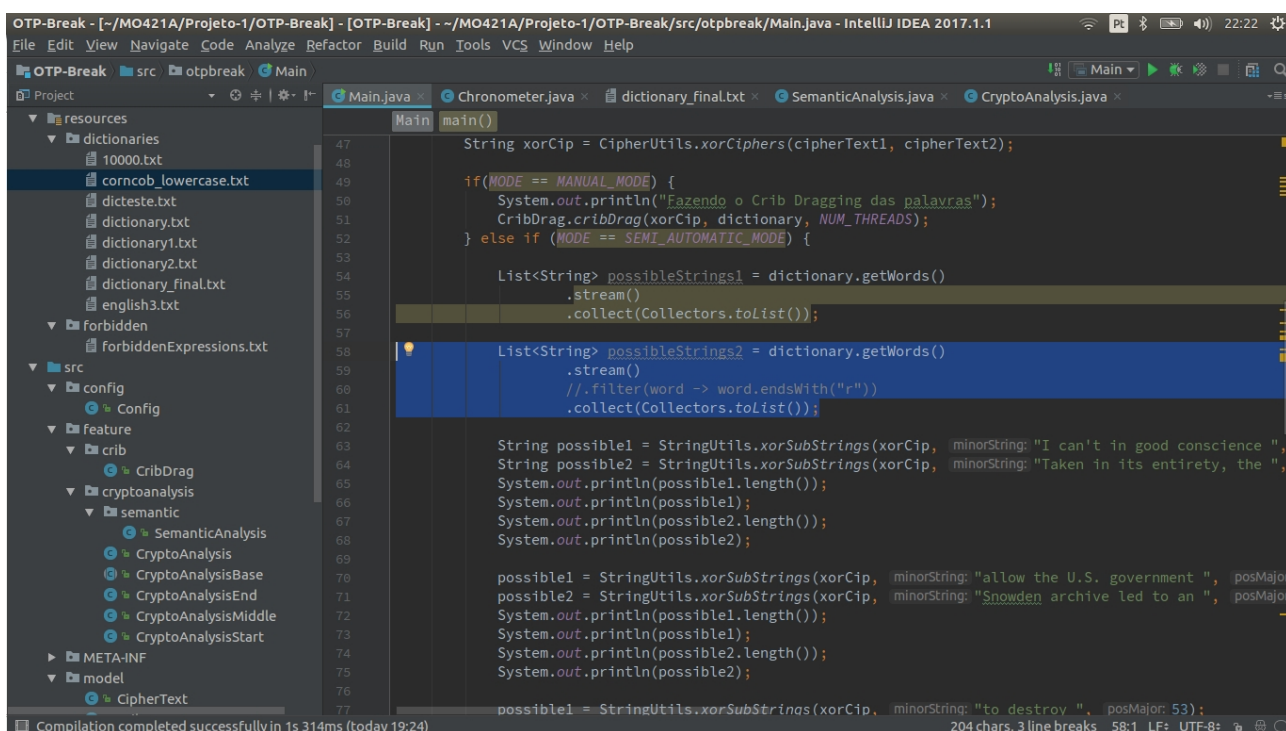


Nesta fase não foram obtidos resultados expressivos, mas os pares de palavras **“Taken in ” / “I can’t i”**, **“m that has a” / “ around the ”**, **“ with ” / “ goal ”** e **“ of electronic” / “e machine they”** serviram como guias para passar para a próxima fase.

2.2 Fase semi-automática

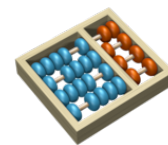
Como os resultados da fase manual demoravam a ser obtidos, foi decidido usar o processamento do computador nessa busca. Para isso, foi usada a API de *streams* do Java 8, que permitem que fosse feita a busca em uma lista guardada em memória sem a necessidade de implementação adicional.

Para isso, com os dicionários foram realizadas buscas conforme a necessidade das expansões. Assim, rapidamente os trechos foram expandidos: **“Taken in ” / “I can’t i”** expandiu para **“Taken in its entirety, the ” / “I can’t in good conscience ”**, **“m that has a” / “ around the ”** expandiu para **“ simple conclusion: the U.S. government had built a system that has as its ” / “ privacy, internet freedom and basic liberties for people around the world ”** e **“ of electronic” / “e machine they”** virou **“surveillance machine they’re secretly building.” / “elimination of electronic privacy worldwide.”**. Os filtros de palavras e as expansões eram feitas manualmente no código conforme a necessidade e o andamento da decifração da mensagem.



```
47 String xorCip = CipherUtils.xorCiphers(cipherText1, cipherText2);
48
49 if(MODE == MANUAL_MODE) {
50     System.out.println("Fazendo o Crib Draggng das palavras");
51     CribDrag.cribDrag(xorCip, dictionary, NUM_THREADS);
52 } else if (MODE == SEMI_AUTOMATIC_MODE) {
53     List<String> possibleStrings1 = dictionary.getWords()
54         .stream()
55         .collect(Collectors.toList());
56
57     List<String> possibleStrings2 = dictionary.getWords()
58         .stream()
59         .filter(word -> word.endsWith("r"))
60         .collect(Collectors.toList());
61
62     String possible1 = StringUtilities.xorSubStrings(xorCip, minorString: "I can't in good conscience ");
63     String possible2 = StringUtilities.xorSubStrings(xorCip, minorString: "Taken in its entirety, the ");
64     System.out.println(possible1.length());
65     System.out.println(possible1);
66     System.out.println(possible2.length());
67     System.out.println(possible2);
68
69     possible1 = StringUtilities.xorSubStrings(xorCip, minorString: "allow the U.S. government ", posMajor:
70     possible2 = StringUtilities.xorSubStrings(xorCip, minorString: "Snowden archive led to an ", posMajor:
71     System.out.println(possible1.length());
72     System.out.println(possible1);
73     System.out.println(possible2.length());
74     System.out.println(possible2);
75
76     possible1 = StringUtilities.xorSubStrings(xorCip, minorString: "to destroy ", posMajor: 53);
77
```

Figura 2. Exemplo de uso da API de *streams* do Java 8 na ferramenta IntelliJ IDEA



Depois, mais trechos foram descobertos e as mensagens foram decifradas com os seguintes conteúdos e tamanhos de 203 e 200 caracteres respectivamente, conforme suas respectivas cifras:

Mensagem 1 – “I can't in good conscience allow the U.S. government to destroy privacy, internet freedom and basic liberties for people around the world with this massive surveillance machine they're secretly building.”

Mensagem 2 – “Taken in its entirety, the Snowden archive led to an ultimately simple conclusion: the U.S. government had built a system that has as its goal the complete elimination of electronic privacy worldwide.”

Para garantir a veracidade da frase, foram realizadas buscas no Google ((6) e (7)), e ambas foram encontradas. A primeira é uma frase da autoria de Edward Snowden, e a segunda é uma frase encontrada no livro “No Place To Hide”, de Glenn Greenwald.

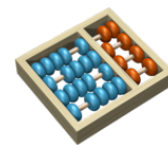
A grande dificuldade desta fase foi perceber que existiam palavras como “Snowden” e “U.S.” no meio do caminho, que se encaixam mais como nomes próprios e não são triviais para se obter em dicionários.

2.3 Fase automática

Após a mensagem ser decifrada, foi realizada a automatização do projeto. Ela foi iniciada a partir dos seguintes pressupostos:

- Toda frase começa com uma letra maiúscula e termina em ponto.
- Como a largura das cifras é diferente, e partindo do primeiro pressuposto, é possível adivinhar o primeiro caractere, e consequentemente um trecho da frase.
- Nas fases anteriores, foram necessárias mais de uma expansão para ter o todo da frase. A frase não virá logo na primeira vez, sendo necessários expansões em vários trechos.
- Há muitos falsos positivos. Muitos *cribs* levam a uma suposta frase, mas param no meio do caminho ou formam frases sem sentido nenhum.

Dessa forma, a automatização foi dividida em: Descoberta do fim, descoberta do início, descoberta do meio, junção das partes, eliminação dos falsos positivos e obtenção dos textos planos e da chave. A fase automática foi testada com os dicionários de 10000 e de 58000 palavras.



2.3.1 Descoberta do fim

Como uma cifra possui 203 caracteres e outra 200, e partindo que o caracter 200 é um ponto, é possível fazer a operação XOR desse caracter com a operação XOR resultante das duas cifras, obtendo o caracter ‘i’. Partindo que o caracter 203 também é um ponto, são buscadas as palavras cuja ante-penúltima letra seja o caracter ‘i’, e são obtidos os *cribs* válidos para a expansão do texto. Logo após a primeira expansão, um trecho novo da outra frase é revelado, e esta outra pode ser expandida revelando um trecho novo da primeira, repetindo o processo até que os dois textos parem em um espaço na mesma posição, impossibilitando o trecho de ser expandido.

Essa parte da automatização é a única que é feita em sentido inverso, pois começamos dos últimos caracteres. Para esta parte, foi necessária apenas a expansão do texto plano através de espaço. Após o final é feita uma validação para eliminar expansões que produziram palavras que não estavam no dicionário, ou que estavam no dicionário mas semanticamente soaram como falsos positivos (o funcionamento da parte de falsos positivos será explicada em 2.3.5).

2.3.2 Descoberta do início

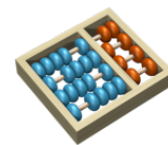
Para a primeira expansão, é possível partir que a primeira letra de uma frase é sempre maiúscula (De A a Z). Se a resultante da operação XOR de alguma delas não for uma letra correspondente, ela é eliminada. Logo após, são obtidos os *cribs* válidos para as palavras que começam com as letras válidas e as possibilidades de expansões.

Essas possibilidades são expandidas como na primeira parte, indo até que o trecho seja impossibilitado de se expandir, com a diferença de que ela é realizada no sentido tradicional e que são necessárias as expansões de texto por vírgula e por vírgula e por espaço.

2.3.3 Descoberta do meio

A descoberta do meio tem em sua essência a descoberta do início, mas há algumas diferenças entre as duas:

- A primeira expansão é feita com caracteres de A a Z, sem a necessidade de serem maiúsculas ou minúsculas.



- Ela exige que a descoberta do fim e a descoberta do início sejam feitas, para fins de obter o início e o fim da parte intermediária.
- Nela não é obtido pares de textos, e sim um objeto que guarda, além do par, a posição final e inicial, que serão importantes na hora da junção das partes.
- Como não há como saber em quantas partes o texto irá se “quebrar”, assim que uma expansão se finalizar será iniciada outra até chegar a largura da parte final.
- São necessárias as expansões por espaço, vírgula e espaço e dois pontos e espaço.
- Por questões de desempenho, nessa parte as validações de palavras presentes no dicionário são feitas após cada expansão e também após o final delas (quando o texto é impossibilitado de ser expandido). Dessa forma, são eliminadas expansões desnecessárias em certas partes.

2.3.4 Junção das partes

Nessa parte, são comparadas a posição final e a posição inicial de cada uma das partes intermediárias e, após a comparação, são feitas combinações das partes iniciais, partes intermediárias e partes finais de modo a formar um conjunto de textos planos possíveis.

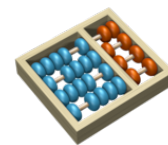
2.3.5 Eliminação dos falsos positivos

Após a junção, como não é formado apenas um par de textos planos após a expansão, a fase de eliminação dos falsos positivos é feita para corrigir o seguinte problema:

$$\text{“dois três”} \oplus \text{“três dois”} = \text{“dois dois”} \oplus \text{“três três”}$$

Em outras palavras, quando um texto cifrado pela **“One-Time Pad”** é dividido em partes em sua decifração, não há como saber exatamente qual é a parte a ser encaixada, gerando uma quantidade muito grande de falsos positivos. Para eliminá-los, foi necessária intervenção manual. Foi criado um arquivo texto de expressões proibidas que é alimentado manualmente conforme as execuções, e conforme os textos são unidos eles podem ser eliminados por essas expressões. Tal arquivo também participa das fases de descoberta para eliminar *cribs* desnecessários, mas é após a junção das partes de textos planos é que ele possui maior importância.

2.3.6 Obtenção dos textos planos e da chave



Com os textos planos obtidos, são gerados dois arquivos referentes aos textos planos correspondentes às suas cifras na pasta “**plaintexts**”. Para obtenção da chave, é feita uma operação *XOR* entre a maior cifra e o maior texto plano, e guardada em um *array* de *bytes*. Se colocados em um *array* de caracteres, a chave não decifrará os textos planos. Uma verificação da chave obtida é feita por garantia, decifrando as duas cifras.

Como os textos planos, também é gerado um arquivo da chave, na pasta “**key**”. Vale destacar que a chave original possuía 205 *bytes*, enquanto as cifras possuíam 203 e 200 *bytes*. Como as cifras possuem um tamanho menor que o da chave, é impossível resgatar a chave inteira: O máximo de *bytes* possível para ser resgatado é 203. Os 2 *bytes* restantes se perderam no momento em que as frases foram cifradas.

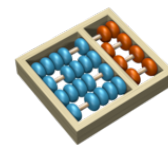
3. Conclusão

Podemos concluir que a técnica de “**One-Time Pad**” pode ser quebrada automaticamente se ela não for usada por uma única mensagem, desde que se tenha as cifras, um dicionário com todas as palavras destas mensagens e elimine todos os falsos positivos possíveis.

Como o programa foi testado apenas com as duas cifras obtidas, outras cifras podem apresentar *bugs* e acarretar um funcionamento imprevisível do mesmo. Como a cifra pode ser dividida em partes, *threads* podem ser adicionadas sem grandes impactos para decifrar os textos planos em menor tempo, desde que se faça o monitoramento correto delas. É possível utilizar análise semântica para obter os textos planos corretos, mas uma análise acurada exigiria muito tempo e provavelmente o prazo deste projeto acabaria estourando sem que a mesma estivesse estável.

4. Bibliografia

- (1) MASON, Joshua, WATKINS, Kathryn, EISNER, Jason, STUBBLEFIELD, Adam, “A Natural Language Approach to Automated Cryptanalysis of Two-time Pads”, págs 1-2, Novembro/2006, presente em: <<https://www.cs.jhu.edu/~jason/papers/mason+al.ccs06.pdf>>
- (2) One-Time Pad – Wikipedia, presente em: <https://en.wikipedia.org/wiki/One-time_pad> Acesso em: 15 Abr. 2017



- (3) JUST WORDS! Dictionaries and Word Lists, presente em:
<<http://www.gwicks.net/dictionaries.htm>> Acesso em: 03 Abr. 2017
- (4) The Corncob list of more than 58 000 English words, presente em:
<<http://www.mieliestronk.com/wordlist.html>> Acesso em: 03 Abr. 2017
- (5) GitHub – first20hours/google-10000-english, presente em:
<<https://github.com/first20hours/google-10000-english>> Acesso em: 06 Abr. 2017
- (6) Google Search, presente em: <<https://www.google.com.br/search?q=Taken+in+its+entirety+%2C+the+Snowden+archive+led+to+an+ultimately+simple+conclusion%3A+the+U.S.+government+had+built+a+system+that+has+as+its+goal+the+complete+elimination+of+electronic+privacy+worldwide.&oq=Taken+in+its+entirety+%2C+the+Snowden+archive+led+to+an+ultimately+simple+conclusion%3A+the+U.S.+government+had+built+a+system+that+has+as+its+goal+the+complete+elimination+of+electronic+privacy+worldwide.&aqs=chrome..69i57j0j4&sourceid=chrome&ie=UTF-8#q=I+can't+in+good+conscience+allow+the+U.S.+government+to+destroy+privacy,+internet+freedom+and+basic+liberties+for+people+around+the+world+with+this+massive+surveillance+machine+they're+secretly+building.>> Acesso em: 09 Abr. 2017
- (7) Google Search, presente em: <<https://www.google.com.br/search?q=Taken+in+its+entirety+%2C+the+Snowden+archive+led+to+an+ultimately+simple+conclusion%3A+the+U.S.+government+had+built+a+system+that+has+as+its+goal+the+complete+elimination+of+electronic+privacy+worldwide.&oq=Taken+in+its+entirety+%2C+the+Snowden+archive+led+to+an+ultimately+simple+conclusion%3A+the+U.S.+government+had+built+a+system+that+has+as+its+goal+the+complete+elimination+of+electronic+privacy+worldwide.&aqs=chrome..69i57j0j4&sourceid=chrome&ie=UTF-8>> Acesso em: 09 Abr. 2017