Criptografia por substituição

Dante Eleutério dos Santos GRR20206686
Richard Fernando Heise Ferreira GRR20191053

[1]Universidade Federal do Paraná
Curitiba – PR – Brasil

1. Introdução

O trabalho consiste na implementação de um algoritmo que utiliza substituição de caracteres a fim de criptografar um texto claro. Nosso algoritmo em específico recebe o nome de *bingchiling* e implementa o conhecido algoritmo One-Time Pad seguido de uma substituição alfabética simples, os detalhes estarão nas sessões abaixo.

2. O algoritmo

O algoritmo funciona da seguinte forma:

Para criptografia:
Primeiro, uma entrada é lida do teclado e armazenada em um buffer de chars, em seguida esse buffer é parseado para outro buffer codificado em UTF-32, para que possamos realizar operações matemáticas apropriadas com os caracteres a fim de substituir, especificamente, por caracteres do alfabeto japonês em geral.

Depois, realizamos a operação de um XOR com os bits dos primeiros 10 caracteres da entrada com nossa chave, que consiste na string do dia, mês e ano atual (no formato dd-mm-aaaa, dessa forma o tamanho dela é fixo sempre em 10); assim, garantimos que cada dia do ano terá uma chave diferente. Os demais caracteres da entrada são feitos XOR com a própria entrada, de forma que a extensão da chave seja sempre a própria entrada do usuário. Em seguida, somamos o valor de 0x3061 com o resultado do XOR para que atinjamos o alfabeto japonês em Unicode e, efetivamente, transformemos nossa saída em uma string com caracteres japoneses.

Para decriptografar:
A decriptografia é completamente análoga, já que o algoritmo é simétrico: primeiro, diminuímos o valor de 0x3061 da entrada e fazemos XOR dos primeiros 10 caracteres da entrada com o a chave, resultando na decriptografia desses caracteres que serão, então, usados para terminar de decriptografar o resto da string. Isso funciona pela seguinte propriedade do XOR:
a $\oplus b = c$
$a \oplus c = b$

3. Implementação

Utilizamos um código de conversão de UTF-8 para UTF-32 que encontramos no site rosettacode, visto que a conversão foi extremamente complexa quando tentamos realizá-la. O código está em C, possui Makefile e pode ser rodado seguindo o que explica o arquivo *usage.txt*.

O código está comentado e, no geral, seu funcionamento é simples o bastante para que não haja comentários nesta sessão que justifiquem-se.

4. Comentários gerais

A parte mais difícil do trabalho foi, de longe, trabalhar com UTF-8/UTF-32. Resolvemos utilizar UTF-8/UTF-32 por dois motivos: primeiro, gostaríamos de ter a saída em japonês para confundir quem estiver tentado decriptografar. Segundo, gostaríamos de preservar todos os caracteres especiais do português – e talvez demais línguas –, pois seria muito simples a remoção de acentos e 'ç' e, de certa forma, seria "chato" se só funcionasse com caracteres padrão ASCII, na nossa visão.

Devido à natureza das operações com XOR ser simétrica, achamos melhor fazer dessa forma a fazer uma soma e subtração modular, como propõe inicialmente o algoritmo de One-Time Pad. Também não aleatorizamos nosso alfabeto possível, portanto, nosso algoritmo não é essencialmente o One-Time Pad com substituição monoalfabética simples, mas uma espécie de junção das coisas e mais uma troca de chave diária, com a ideia do que era feito na Enigma (de forma extremamente simplificada, pelo menos).

Chamamos de *bingchiling* em homenagem à um vídeo famoso da internet.

Finalmente, alguns caracteres acabam não sendo representados corretamente – tivemos problemas em tentar criptografar ualguns emojis. Isso dependerá do terminal que o usuário está usando, visto que depende da interpretação do Unicode resultante da soma de 0x3061. Inclusive, usamos esse valor em específico devido a ter sido, empiricamente, o valor que menos resultou em caracteres não-representáveis no nosso terminal.