# Criptografia Assimétrica

O chefe da polícia do Rio supervisiona várias patrulhas na cidade. Elas estão espalhadas e vigiam os pontos onde ocorre o tráfico de drogas. Regularmente, estas patrulhas têm que enviar um relatório sobre a movimentação das quadrilhas, sem que seja possível aos traficantes interceptar estas mensagens. Optou-se por entregar a cada patrulha um aparelho codificador de mensagens de texto, que é enviado por rádio para uma central que analisa estas informações. Para que não se corresse o risco de alguém enviar uma mensagem com a chave criptográfica errada, ficou decidido que cada aparelho teria a chave criptográfica escrita em baixo do aparelho. Isto assegura que se o policial esquecesse a "*chave*", ele sempre poderia olhar em baixo do aparelho e usar a string correta. Para facilitar ainda mais o processo, ficou decidido que todos os aparelhos usariam a mesma chave criptográfica. Você foi contratado para escrever o software desta máquina, de tal forma que mesmo que uma dessas máquinas fosse interceptada pelos bandidos, eles não poderiam decifrar as mensagens enviadas para o chefe de polícia.

Quando ouviu sua incumbência, você ficou estupefato!

- *Mas o que me pedes, ó amado chefe, se afigura deveras difícil, quiçá impossível! Na primeira vez que os meliantes capturarem uma máquina dessas saberão todos nossos segredos!*

Seu chefe, apiedado da sua ignorância de principiante, mas sabedor do grande potencial que você esconde, apesar das suas lamúrias, apenas disse, enigmaticamente:

- Estude a criptografia assimétrica, e suas dúvidas se desvanecerão como gotas de orvalho ao sol da manhã.

Depois de muito pensar, você pediu ajuda a um mestre de ciências ocultas que, por acaso, também era matemático. Este mestre consultando seu oráculo, deu a seguinte ideia:

- Escolha **dois números primos**. Eles não devem ser muito pequenos, mas para nossa demonstração, o **13** e o **17** servem. Pegue o número que seja o produto desses dois números primos, no caso, o número **221**. Tire um de cada um dos primos iniciais, o que dá $12 \times 16 = 192$.

- *Deveras intrigante, amado mestre.*

- Tenha fé, jovem **incréu**!, gritou o mestre. Continue me acompanhando. Vamos representar este número **192** pela letra *Y*. Este número será muito importante na codificação. Pegue agora um número que não tenha divisor comum com *Y*, ou seja, este novo número e o Y serão **primos entre si**.

- *Finalmente aquilo que aprendi na escola primária vai ter uma utilidade!* Você exclamou exultante de alegria.

- Outro conhecimento útil da escola será a fatoração. Completou o mestre. Vamos fatorar 192. Isto dá $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 3$. Basta agora escolher um número que não seja divisível nem por 2 nem por 3 (os fatores de 192). Escolhemos o **5**. As suas **chaves públicas** serão **5** e **221**. Este número 5 e o número inicial 221 serão aqueles que você vai escrever em baixo de cada aparelho. Jogue fora os números primos que você escolheu inicialmente. Agora vejamos como deverá ser seu algoritmo de criptografia. Pegue cada letra, ou seja, seu código ASCII, e eleve a 5 e guarde o resto da divisão deste resultado por 221.

Aqui está um exemplo:

**Celacanto**

**67 101 108 97 99 97 110 116 111**

agora convertemos:

$$(67^5)\%221 = \mathbf{84}$$
$$(101^5)\%221 = \mathbf{186}$$
$$(108^5)\%221 = \mathbf{75}$$

$$(97^5)\%221 = \mathbf{54}$$
$$(99^5)\%221 = \mathbf{216}$$
$$(97^5)\%221 = \mathbf{54}$$
$$(110^5)\%221 = \mathbf{145}$$
$$(116^5)\%221 = \mathbf{12}$$
$$(111^5)\%221 = \mathbf{76}$$

**84 186 75 54 216 54 145 12 76**

Esta mensagem é enviada.

Para decodificar, vamos olhar qual seria a chave para descriptografar. Repartimos do número Y. A chave de decodificação é o número cujo produto pelo primeiro número publicado (5) tem como resto 1 na divisão por Y (192). No nosso caso, este número é o 77 ($5 \times 77 = 385 = 2 \times 192 + 1$).

Vamos usar este número para decodificar.

$(84^{77})\%221 = \mathbf{67} = C$
$(186^{77})\%221 = \mathbf{101} = e$
$(75^{77})\%221 = \mathbf{108} = l$
$(54^{77})\%221 = \mathbf{97} = a$
$(216^{77})\%221 = \mathbf{99} = c$
$(54^{77})\%221 = \mathbf{97} = a$
$(145^{77})\%221 = \mathbf{110} = n$
$(12^{77})\%221 = \mathbf{116} = t$
$(76^{77})\%221 = \mathbf{111} = o$

- *Voilá! A coisa funciona!* Você exclama admirado, depois de ver que conseguimos com simples contas recuperar a mensagem original.

Depois de pensar um pouco, você vê uma pequena dificuldade. Como usuário de sistemas operacionais avançados, você usa Linux e na linha de comando com a supercalculadora **bc** você descobre que **216 elevado a 77** (basta chamar **bc** na linha de comando e digitar $216 \wedge 77$) é igual a

566.159.551.582.825.161.743.621.404.474.404.760.031.142.448.407.527.580.407.064.
693.864.409.775.046.008.707.686.153.584.777.361.889.405.985.989.370.510.938.768.
427.806.052.676.828.062.427.197.896.352.669.811.974.404.740.004.068.150.214.656,

ou seja, uns 566 "*zilhões*".

Seu pobre compilador C só consegue fazer contas com inteiros até uns **4 bilhões**...

- *Amado mestre, como poderei eu, com um reles compilador C, calcular números tão assombrosamente grandes?*

- A resposta, jovem ignaro, está na aritmética de relógio, na fatoração e em um pouco de imaginação. Com estes recursos, toda essa conta se resolve em poucas linhas de código C.

Para resolver este nó górdio, vamos usar a aritmética modular. A aritmética modular tem a propriedade distributiva na multiplicação. Assim, se 77 for decomposto em seus bits: $77 = 1001101 = 2^6 + 2^3 + 2^2 + 2^0 = 64 + 8 + 4 + 1$.

Ou seja,

$$(216^{77})\%221 = (216^{64} \times 216^8 \times 216^4 \times 216^1)\%221)$$

usando a propriedade distributiva, temos:

$$(216^{77})\%221 = [((216^{64})\%221)) \times ((216^8)\%221)) \times ((216^4)\%221)) \times ((216^1)\%221))]\%221)$$

ora, agora tudo ficou mais fácil:

$$(216^4)\%221 = [((216^2)\%221)) \times ((216^2)\%221))]\%221) = (25x25)\%221 = \mathbf{183}$$

$$(216^8)\%221 = [((216^4)\%221)^2]\%221 = (183^2)\%221 = \mathbf{118}$$

Não é necessário aqui, mas para mostrar que a definição usa os resultados anteriores, vamos calcular $(216^{16})\%221$ e $(216^{32})\%221$

$$(216^{16})\%221 = [((216^{8})\%221)^2]\%221 = (118^2)\%221 = \mathbf{1}$$

$$((216^{32}))\%221 = [((216^{16})\%221)^2]\%221 = (1^2)\%221 = \mathbf{1}$$

$$(216^{64})\%221 = [((216^{32})\%221)^2]\%221 = (1^2)\%221 = \mathbf{1}$$

ou seja,

$$(216^{77})\%221 = (\mathbf{1} \times \mathbf{118} \times \mathbf{183} \times \mathbf{216})\%221 = \mathbf{4664304}\%221 = \mathbf{99}$$

De posse desses conhecimentos inestimáveis, faça dois programas: o primeiro (geranum.c) recebe dois números primos e gera dois arquivos, um com os números de criptografia (numcripto.txt) e o outro com os números de descriptografia (numdescripto.txt) e um programa (cripto.c )que receba um texto ASCII de várias linhas e converta-o usando a criptografia descrita acima. Note que as operações sobre os números inteiros podem ser simplificadas com um simples *for* ou função que calcule a módulo de determinada conta sem mesmo saber o resultado final da exponenciação.

A nota do trabalho é individual e levará em conta também a clareza do código.

Os seus programas devem ser genéricos, isto é, os parâmetros iniciais de geranum.c são apenas os dois números primos quaisquer, e os de cripto.c, as opções -c ou -d para codificar ou decodificar. Teste para ver se os números entrados são realmente primos. Depois, gere os números necessários para a codificação e decodificação, no exemplo acima, foram os números **221**, **5** e **77**. A saída codificada deve ser em inteiros pois alguns caracteres ASCII não serão imprimíveis e você deve pegar os números dos arquivos gerados. A sua decodificação deve pegar esta saída e gerar o arquivo original. Um bom teste é usar um arquivo texto como entrada do seu programa, por redirecionamento de entrada. Se, por exemplo, você usar um arquivo "teste.txt", você pode gerar a saída criptografada com o comando

*cripto -c < teste.txt > saida.crp,*

levando em conta que seu programa foi compilado com o nome de "cripto", e a opção foi de codificar. Para recuperar o arquivo original, você deve fazer

*cripto -d < saida.crp > decode.txt.*

Você deve criar uma conta em um serviço de controle de versão, como o *github*, e usá-lo para guardar as versões de seu programa.

Seu código também deve ser formatado usando o programa *astyle*.

Seu código deve ser compilado com os flags *-std=c99 -Wall -Werror -pedantic*.

Boa Sorte, mas cada variável global desconta um ponto...

Não esqueça de colocar seu nome em cada arquivo fonte.

Os programas serão corrigidos usando o compilador gcc em ambiente Linux.

Obs.: Qualquer tentativa de desonestidade intelectual, conhecida popularmente como cola, será considerada falta grave e punida com nota zero.

Seu programa deve estar bem indentado, documentado e organizado. A indentação deve deixar clara a estrutura de subordinação dos comandos. Os comentários devem ser esclarecedores.

---