Criptografia - Prova 1 - maio 2020

Esta prova *não-presencial* é aplicada de acordo com a Resolução da Comissão de Graduação (CoG) da USP, número 7949, de 27 de abril de 2020.

- Horário da prova: das 16h às18h de 11-maio-2020. (Se fosse presencial seria também nesse intervalo.)
- Favor entregar a sua solução *preferencialmente* na forma de texto digitado. Mas ela pode ser manuscrita, contanto que seja bem *legível*; favor verificar ANTES da entrega.
- Entregar a sua solução *obrigatoriamente* no sistema Paca em forma digital, com todas as páginas comprimidas dentro de um único arquivo, entre 18h e 19h do mesmo dia 11, ou antes desse intervalo.
- Se for entregue entre 19h e 20h, a nota final terá 1 ponto a menos.

(Questão 1)-50%-

O algoritmo K64 de chave secreta, definida a seguir, criptografa um bloco B de 64 bits após aplicar *rounds* (iterações), como no AES e DES. O número de rounds é $R \geq 1$.

Operações sobre bytes de 8 bits

Três operações especiais são utilizadas sobre operandos x, y em bytes de 8 bits, resultando um byte; são definidas a seguir:

A primeira operação é: $y = 45^x \bmod 257$ ($y = 0$ se $x = 128$, pois $45^{128} \bmod 257 = 256$)

A segunda operação é: $x = \log_{45} y$ é a inversa da anterior, i.e., $\log_{45}(45^x \bmod 257) = x$. Convenciona-se que: $x = 128$ se $y = 0$, para ser consistente com a operação anterior. Um exemplo: $y = 45^3 \bmod 257 = 229$, e $x = \log_{45}(229) = 3$.

A terceira operação é: $x + y$ ´representa $(x + y) \bmod 257$ resultando um byte de 8 bits.

Descrição da geração das $2R + 1$ subchaves

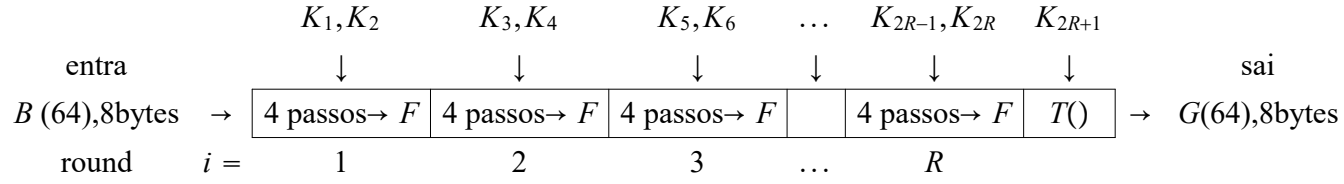
A chave principal K é de 64 bits. A seguir é descrita a geração das $2R + 1$ subchaves.

A primeira subchave, K_1, é a própria chave principal K de 64 bits.

Para $i = 1, 2, 3, \ldots, 2R$ as subchaves seguintes são geradas a partir de K_1 pela fórmula: $K_{i+1} = K_1 << 3i$, onde $v << t$ significa aplicar sobre v um deslocamento *circular para a esquerda* de t posições de bit.

Aplicação das $2R + 1$ subchaves

Uma mesma função é aplicada um certo número de rounds, chamado R. Seja um índice de round $i = 1, 2, 3, \ldots, R$.



Um bloco B de entrada, de 64 bits, é dividido em 8 sub-blocos $B_1, B_2, B_3, \ldots, B_8$. E então, R rounds são aplicados a estes sub-blocos, e depois uma transformação final $T()$ é aplicada, obtendo-se a saída final G de 64 bits, 8 bytes.

Descrição de um round (uma iteração)

Cada round de índice $i = 1, 2, \ldots R$ utiliza duas subchaves: K_{2i-1} e K_{2i}, e é constituída de 4 passos que são descritos a seguir, para calcular F (8 bytes). Esse resultado F (8 bytes) é a entrada B para o round $i + 1$.

- **Primeiro Passo**: aplica subchave K_{2i-1} sobre $B_1, B_2, B_3, \ldots, B_8$, resulta C (8 bytes)

Primeiramente cada sub-bloco B_j é submetido a *XOR* (ou-exclusivo) ou soma $\bmod 257$ com a subchave K_{2i-1}, resultando um byte, da seguinte forma (sendo $K^1_{2i-1}, K^2_{2i-1}, K^3_{2i-1}, K^4_{2i-1}, K^5_{2i-1}, K^6_{2i-1}, K^7_{2i-1}, K^8_{2i-1}$ os 8 bytes de K_{2i-1},):

$B_1 (XOR) K^1_{2i-1} = C_1, B_2 + K^2_{2i-1} = C_2, B_3 + K^3_{2i-1} = C_3, B_4 (XOR) K^4_{2i-1} = C_4,$

$B_5 (XOR) K^5_{2i-1} = C_5, B_6 + K^6_{2i-1} = C_6, B_7 + K^7_{2i-1} = C_7, B_8 (XOR) K^8_{2i-1} = C_8$

- **Segundo Passo**: resulta D (8 bytes)

Os 8 bytes C_j ($j = 1, 2, \ldots 8$) resultantes do passo anterior são submetidos às duas operações já definidas (45^x e $\log_{45}(y)$), da seguinte forma:

$(45)^{C_1} = D_1, \log_{45} C_2 = D_2, \log_{45} C_3 = D_3, (45)^{C_4} = D_4,$

$(45)^{C_5} = D_5, \log_{45} C_6 = D_6, \log_{45} C_7 = D_7, (45)^{C_8} = D_8$

- **Terceiro Passo**: aplica subchave K_{2i} sobre D, resulta E (8 bytes)

Os 8 bytes D_j obtidos no Segundo Passo são agora submetidos às operações de soma $\bmod 257$ e *XOR* com os bytes da subchave K_{2i} da seguinte forma (sendo $K^1_{2i}, K^2_{2i}, K^3_{2i}, K^4_{2i}, K^5_{2i}, K^6_{2i}, K^7_{2i}, K^8_{2i}$ os 8 bytes de K_{2i},):

$D_1 + K^1_{2i} = E_1, D_2 (XOR) K^2_{2i} = E_2, D_3 (XOR) K^3_{2i} = E_3, D_4 + K^4_{2i} = E_4,$

$D_5 + K_{2i}^5 = E_5, D_6(XOR)K_{2i}^6 = E_6, D_7(XOR)K_{2i}^7 = E_7, D_8 + K_{2i}^8 = E_8$

- **Quarto Passo**: resulta F (8 bytes)

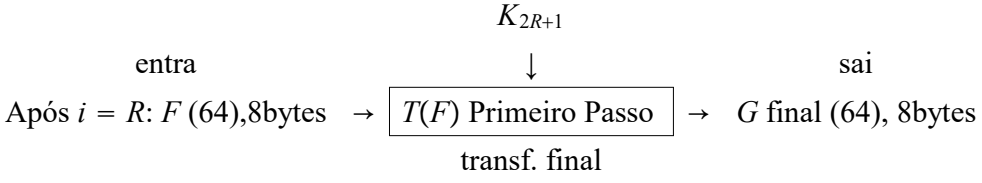
Finalmente, os 8 bytes E_j obtidos são sumetidos à operação $f: (a_1, a_2) \to (b_1, b_2)$ de 2 entradas a_1, a_2 (2 bytes) e 2 saídas b_1, b_2 (2 bytes) definida por:

$$b_1 = (2a_1 + a_2) \bmod 256$$
$$b_2 = (a_1 + a_2) \bmod 256$$

As operações são da seguinte forma: (Note que a seguir tem-se mod 256, propositalmente)

$F_1 = (2E_1 + E_2) \bmod 256, F_2 = (E_1 + E_2) \bmod 256$
$F_3 = (2E_3 + E_4) \bmod 256, F_4 = (E3 + E_4) \bmod 256$
$F_5 = (2E_5 + E_6) \bmod 256, F_6 = (E_5 + E_6) \bmod 256$
$F_7 = (2E_7 + E_8) \bmod 256, F_8 = (E_7 + E_8) \bmod 256$

Descrição da transformação final $T()$: aplica subchave K_{2R+1} sobre F e calcula G (64 bits)



Depois de calcular todos os R rounds ($i = 1, 2, 3, \ldots R$), os 8 sub-blocos de F final (após $i = R$) são submetidos a uma transformação final T que é exatamente *igual* ao **Primeiro Passo** descrito acima, só que a subchave utilizada deve ser a última: K_{2R+1}. E o resultado G é a saída final do algoritmo K64.

Esta questão consiste em:

1. Escrever o passo inverso de *cada* um dos quatro passos descritos e *demonstrar matematicamente* que cada um é de fato o inverso.
2. Escrever *apenas* a definição da inversa do round $i = 1$ do K64, e *demonstrar matematicamente* que B é recuperado quando a entrada é F.
3. Escrever apenas a função inversa da transformação final $T()$ do K64, e *demonstrar matematicamente* que F é recuperado quando a entrada é G.
4. Supondo que as inversas de cada um dos R rounds e de $T()$ estão disponíveis e corretas, escrever a função inversa global do K64 e *demonstrar matematicamente* que essa função recupera o bloco de entrada original B quando a entrada é G.

(Questão 2)-50%-

O algoritmo para Alice assinar um documento digital possui as seguintes premissas (hipóteses, por definição do esquema):

- Alice escolhe um primo p, público, e um primo q que divide $(p - 1)$, também público, longos.
- A seguir Alice calcula um gerador $g \in Z_p^*$ uma q-ésima raiz de 1 mod p, pública (*i.e.*, $g^q = 1 \bmod p$).
- Alice sorteia aleatoriamente a chave secreta $S \in Z_{q-1}$ tal que $T = g^S \bmod p$. Em resumo, o conjunto de chaves é $K = \{(p, q, g, S, T) : T = g^S \bmod p\}$, onde (T, p, q, g) são públicos e $S \in Z_{q-1}$ é a chave particular, secreta, conhecida apenas por Alice.

Algoritmo para Alice assinar $x \in Z_p^*$

1. Escolher um número aleatório secreto k (NONCE) tal que $1 \leq k \leq q - 1$
2. Calcular $C = (g^k \bmod p) \bmod q$
3. Calcular $D = (x + SC)k^{-1} \bmod q$
4. A assinatura é (C, D) que depende de x e k. Alice envia para Beto (x, C, D).

Algoritmo para Beto verificar a assinatura

Sendo (C, D) a assinatura de $x \in Z_p^*$, a verificação é como a seguir.

1. Se C e D *não* satisfizerem $0 < C < q$, $0 < D < q$, rejeitar a assinatura.
2. Beto calcula $e_1 = xD^{-1} \bmod q$, e $e_2 = CD^{-1} \bmod q$.
3. Se $(g^{e_1} \times T^{e_2} \bmod p) \bmod q = C$, a assinatura é correta.

Esta questão é constituída dos seguintes itens:

1. O NONCE k no passo (1) do algoritmo para assinar x aumenta ou diminui a segurança do esquema? Ou seja, qual é o objetivo de se usar esse k? Justifique a sua resposta.
2. Por quê a assinatura deve ser rejeitada se C e D não satisfizerem $0 < C < q$, $0 < D < q$ no passo (1) do algoritmo para verificar? Justifique a sua resposta.
3. Em *cada* passo dos dois algoritmos, escrever a **definição** do problema computacional que protege a chave particular S ou algum outro parâmetro crítico para a segurança contra ataques de um intruso que intercepte (x, C, D) verdadeiros e queira falsificar uma assinatura sobre um outro $u \in Z_p^*, u \neq x$. **Só** o nome do problema não basta.
4. Dado (x, C, D) demonstrar algebricamente que no **Algoritmo para verificar** acima, se $(g^{e_1} \times T^{e_2} \bmod p) \bmod q = C$, então a assinatura é da Alice verdadeira sobre x, e x não foi alterado.
5. E demonstrar a afirmação inversa, ou seja, se a assinatura for da Alice verdadeira sobre x e x não foi alterado, então deve-se ter $(g^{e_1} \times T^{e_2} \bmod p) \bmod q = C$.
6. O que ocorre se x for alterado para um outro valor diferente $u \in Z_p^*, u \neq x$? Justifique a sua resposta.
7. A assinatura de um documento qualquer, x, exige a presença do autor da assinatura? Por quê?
8. A verificação de uma assinatura exige a presença do autor da assinatura? Por quê?
9. A geração de uma assinatura demora mais ou menos que a verificação da mesma assinatura? Justifique a sua resposta em termos de número de operações básicas relativamente mais demoradas.
10. Como uma falsa Alice, sem conhecer o segredo S da Alice, poderia falsificar uma assinatura sobre um texto $u \in Z_p^*, u \neq x$? Qual é a probabilidade de sucesso de tal falsificação? A sua resposta deve levar em consideração o comprimento dos parâmetros envolvidos.

FIM-FIM