MAC0110: EP EX.0 (QUALQUER MENSAGEM)

Por Vitor Santa Rosa Gomes, 10258862, para MAC0110 em 18/03/2017

1. ENUNCIADO

Ex. 0 (Qualquer mensagem)

Lembre o exemplo visto em sala:

Alice quer enviar a mensagem SENDMONEY para Bob.

Alice e Bob têm a chave comum yT25a5i/S.

Usando o algoritmo de one-time pad, Alice produz o texto cifrado gX76W3v7K, que ela envia para Bob.

Eve consegue acesso à mensagem cifrada gX76W3v7K. Ela tenta todas as chaves possíveis para decriptar esta mensagem cifrada. Entretanto, ao tentar a chave tTtpWk+1E, ela obtém o texto NEWTATTOO (veja página 19 da aula "Prologue: A Simple Machine"). Assim, ela poderia pensar (incorretamente) que a mensagem que Alice enviou a Bob é NEWTATTOO.

1. Mostre que há uma chave que, quando usada por Eve, gera o texto ILikeEve+.

2. Mostre também que existe uma chave que gera o texto IHateEve+.

3. Mais geralmente, prove que, para qualquer texto de 9 caracteres da tabela Base64 (página 9 das transparências), há uma chave que a gera.

4. Bônus: encontre as chaves que geram ILikeEve+ e IHateEve+.

2. PRINCÍPIOS

2.1. Correspondência base64:

000000 A	001000 I	010000 Q	011000 Y	100000 g	101000 o	110000 w	111000 4
000001 B	001001 J	010001 R	011001 Z	100001 h	101001 p	110001 x	111001 5
000010 C	001010 K	010010 S	011010 a	100010 i	101010 q	110010 y	111010 6
000011 D	001011 L	010011 T	011011 b	100011 j	101011 r	110011 z	111011 7
000100 E	001100 M	010100 U	011100 c	100100 k	101100 s	110100 0	111100 8
000101 F	001101 N	010101 V	011101 d	100101 l	101101 t	110101 1	111101 9
000110 G	001110 O	010110 W	011110 e	100110 m	101110 u	110110 2	111110 +
000111 H	001111 P	010111 X	011111 f	100111 n	101111 v	110111 3	111111 /

2.2. Notação *string*

String "" $= [\quad]$

Exemplo: *string* "abc" $= [a \quad b \quad c]$

2.3. Variáveis:

Nome	Definição	Descrição
Mes	$[mes_{1,1} \quad mes_{1,2} \quad \cdots \quad mes_{1,n}]$	Mensagem a ser enviada
$EnMes$	$[enMes_{1,1} \quad enMes_{1,2} \quad \cdots \quad enMes_{1,n}]$	Mensagem criptografada
$DeMes$	$[deMes_{1,1} \quad deMes_{1,2} \quad \cdots \quad deMes_{1,n}]$	Mensagem descriptografada
$EnKey$	$[\mathrm{enKey}_{1,1} \quad \mathrm{enKey}_{1,2} \quad \cdots \quad \mathrm{enKey}_{1,n}]$	Chave usada na criptografia
$DeKey$	$[\mathrm{deKey}_{1,1} \quad \mathrm{deKey}_{1,2} \quad \cdots \quad \mathrm{deKey}_{1,n}]$	Chave usada na descriptografia
$\mathbb{X}$	$\{0,1\}$	Conjunto de valores de *bits*
$\mathbb{Base64}$	$\{A, B, C, \cdots, +, /\}$	Conjunto dos caracteres *Base64*
$fromBase64(A_{1\times n})$	$B_{1\times(n/6)}$	Função que converte uma matriz de *bits* numa de caracteres *Base64*
$toBase64(A_{1\times n})$	$B_{1\times(n\cdot 6)}$	Função que converte uma matriz de caracteres *Base64* numa de *bits*
$xor(a,b); a,b \in \mathbb{X}$	$\begin{cases} 0, & a \neq b \\ 1, & a = b \end{cases}$	Função ou-exclusivo
$encript(A,B)$	$[xor(\mathrm{a}_{1,1}, \mathrm{b}_{1,1}) \quad \cdots \quad xor(\mathrm{a}_{1,n}, \mathrm{b}_{1,n})]$	Função de criptografia
$decript(A,B)$	$[xor(\mathrm{a}_{1,1}, \mathrm{b}_{1,1}) \quad \cdots \quad xor(\mathrm{a}_{1,n}, \mathrm{b}_{1,n})]$	Função de descriptografia

2.4. Propriedades da função ou-exclusivo:

2.4.1. $xor(a,b) = xor(b,a); \forall a,b \in \mathbb{X}$

2.4.2. Seja $c = xor(a,b)$, *então* $b = xor(a,c), a = xor(b,c); \forall a,b,c \in \mathbb{X}$

2.5. Caso fundamental:

2.5.1. $fromBase64\left(encript\left(toBase64(Mes), toBase64(EnKey)\right)\right) = fromBase64\left(\mathrm{de}cript\left(toBase64(EnMes), toBase64(DeKey)\right)\right) \Leftrightarrow Mes = DeMes$

2.6. Propriedade circunstancial:

2.6.1. $encript(A, B) =$

$[encript([a_{1,1}], [b_{1,1}]) \quad \cdots \quad encript([a_{1,n}], [b_{1,n}])], decript(A, B) =$

$[decript([a_{1,1}], [b_{1,1}]) \quad \cdots \quad decript([a_{1,n}], [b_{1,n}])] \Leftrightarrow encript(A, B) =$

$decript(A, B)$

3. RESOLUÇÃO

3.1. Mostre que há uma chave que, quando usada por Eve, gera o texto ILikeEve+.

Por 2.5.1:

$$fromBase64\left(encript\left(toBase64(\text{"SENDMONEY"}), toBase64(\text{"yT25a5i/S"})\right)\right)$$

$$= fromBase64\left(decript\left(toBase64(\text{"ILikeEve + "}), toBase64(De Key)\right)\right)$$

$$\Longrightarrow$$

$$\text{"gX76W3v7K"} = decript\left(toBase64(\text{"ILikeEve + "}), toBase64(De Key)\right)$$

Por 2.4.2 e por 2.6.1:

$$De Key = fromBase64\left(encript\left(toBase64(\text{"ILikeEve + "}), toBase64(\text{"gX76W3v7K"})\right)\right)$$

$$\Longrightarrow$$

$$De Key = fromBase64\left(encript\left(\begin{matrix}\text{"001000 001011 100010 100100 011110 000100 101111 011110 111110",}\\ \text{100000 010111 111011 111010 010110 110111 101111 111011 001010}\end{matrix}\right)\right)$$

$$\Longrightarrow$$

$$De Key = fromBase64(\text{"101000 011100 011001 011110 001000 110011 000000 100101 110100"})$$

$$\Longrightarrow$$

$$DeKey = \text{"}ocZeIzAl0\text{"}$$

3.2. Mostre também que existe uma chave que gera o texto IHateEve+.

Por 2.5.1:

$$fromBase64\left(encript\left(toBase64(\text{"SENDMONEY"}), toBase64(\text{"yT25a5i/S"})\right)\right) = fromBase64\left(\text{de}cript\left(toBase64(\text{"IHateEve} + \text{"}), toBase64(\text{De}Key)\right)\right) \Rightarrow$$

$$\text{"gX76W3v7K"} = \text{de}cript\left(toBase64(\text{"IHateEve} + \text{"}), toBase64(\text{De}Key)\right)$$

Por 2.4.2 e por 2.6.1:

$$\text{De}Key = fromBase64\left(encript\left(toBase64(\text{"IHateEve} + \text{"}), toBase64(\text{"gX76W3v7K"})\right)\right) \Rightarrow$$

$$\text{De}Key = fromBase64\left(encript\left(\begin{matrix}\text{"001000 000111 011010 101101 011110 000100 101111 011110 111110",}\\ \text{"100000 010111 111011 111010 010110 110111 101111 111011 001010"}\end{matrix}\right)\right) \Rightarrow$$

$$\text{De}Key = fromBase64(\text{"101000 010000 100001 010111 001000 110011 000000 100101 110100"}) \Rightarrow$$

$$DeKey = \text{"}oQhXIzAl0\text{"}$$

3.3. Mais geralmente, prove que, para qualquer texto de 9 caracteres da tabela Base64 (página 9 das transparências), há uma chave que a gera.

Admite-se a possível contradição do caso fundamental (2.5.1):

$$encript\left(toBase64(Mes), toBase64(EnKey)\right) \perp \text{de}cript\left(toBase64(EnMes), toBase64(DeKey)\right) \Leftrightarrow Mes \perp DeMes$$

* Símbolo ⊥ indica independência, ou seja, não necessariamente igual

Pelo caso fundamental (2.5.1):

Se $encript\left(toBase64(Mes), toBase64(EnKey)\right) = \text{de}cript\left(toBase64(EnMes), toBase64(DeKey)\right)$ e $EnKey \perp DeKey$, então $Mes \perp EnMes$, isto é, pode-se obter o mesmo texto cifrado a partir de combinações de textos de 9 caracteres diferentes.

Como $encript\big(toBase64(Mes), toBase64(EnKey)\big)$ é disponibilizada a Eve, aplica-se a propriedade 2.4.2:

$$DeMes = \mathrm{de}cript\left(toBase64\left(encript\big(toBase64(Mes), toBase64(EnKey)\big)\right), toBase64(DeKey)\right)$$

Por característica da propriedade 2.4.2, a função $decript$ relaciona uma dada chave à mensagem (supostamente) descriptografada:

$\forall DeKey = [\mathrm{deKey}_{1,1} \quad \mathrm{deKey}_{1,2} \quad \cdots \quad \mathrm{deKey}_{1,9}], \mathrm{deKey}_{1,n} \in \mathbb{Base64}$, há $DeMes = [\mathrm{deMes}_{1,1} \quad \mathrm{deMes}_{1,2} \quad \cdots \quad \mathrm{deMes}_{1,9}], \mathrm{deMes}_{1,n} \in \mathbb{Base64}$.

Para 9 caracteres, há $|\mathbb{Base64}|^9$ diferentes chaves, mensagens (supostamente) descriptografadas e relações entre ambas definidas pela função $decript$: $|\mathbb{Base64}|^9, |\mathbb{Base64}|^9 \longrightarrow |\mathbb{Base64}|^9$

3.4. Bônus: encontre as chaves que geram ILikeEve+ e IHateEve+.

"$ocZeIzAl0$" e "$oQhXIzAl0$", por 3.1 e 3.2, respectivamente.

4. COMENTÁRIOS

Não soube qual o nível de rigorosidade seguir durante a prova, espero que esteja inteligível.

4.1. Algoritmo de suporte

Script desenvolvido em JavaScript usado para verificação:

```
var encription = {
   encript: function (mes, key) {
           mes = this.encoding.fromBase64ToBinary(mes);
           key = this.encoding.fromBase64ToBinary(key);
           var enMes = "";
           for (var i = 0; i < mes.length; i++)
                   if (mes[i] == " ") enMes += " ";
                   else enMes += ((mes[i] == key[i]) ? "0" : "1");
           return this.encoding.toBase64FromBinary(enMes);
    },
   decript: this.encript,
   encoding: {
        base64Chars: [ "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M",
"N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "a", "b", "c", "d",
"e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u",
"v", "w", "x", "y", "z", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "+", "/" ],
        toBase64FromBinary: function (binary) {
            binary = binary.split(" ");
            var base64 = "";
            for    (var i = 0; i < binary.length; i++)
                if(binary[i] != " ") base64 += this.base64Chars[parseInt(binary[i], 2)];
            return base64;
        },
        fromBase64ToBinary: function (base64) {
```

```
            var binary = "";
            for    (var i = 0; i < base64.length; i++)
                if(base64[i]         !=         "         ")         binary         +=
this.formatBits(parseInt(this.base64Chars.indexOf(base64[i])).toString(2)) + " ";
            return binary.slice(0, -1);
        },
        formatBits: function (bits) {
          return bits.length >= 6 ? bits : new Array(6 - bits.length + 1).join(0) + bits;
        }
    }
}
```