

NOVO ALGORÍTMO DE ASSINATURA DIGITAL BASEADO NO
RSA (HOSec)

NEW DIGITAL SIGNATURE ALGORITHM BASED ON RSA (HOSec)

HUGO L.S PAULO

OSVALDO B.S DOMINGOS

Luanda 2023

RESUMO

Assinatura digital é um mecanismo de autenticação que permite ao criador de uma mensagem anexar um código que actue como uma assinatura. A assinatura é formada tomando o hash da mensagem e criptografando-a com a chave privada do criador. A assinatura garante a origem e a integridade da mensagem.

Palavras Chaves: assinatura digital, RSA, novo, hash, texto claro, texto cifrado

Abstract

Digital signature is a mechanism of authentication that allow the maker of a message to attach a code as a signature. The signature is formed taking the hash of the message and encrypt with the private key of the maker. The signature gives the maker the authentication of the message.

Keywords: digital signature, RSA, new, hash, plain text, coded text

INTRODUÇÃO

Assinatura digital é o desenvolvimento mais importante do trabalho em criptografia de chave pública. A assinatura digital oferece um conjunto de recursos de segurança que seria difícil de implementar de qualquer outra maneira.

Criar nunca é fácil, mas nos empenhamos para oferecer um artigo digno e que possa ser facilmente entendido e se possível integrado...

ASSINATURA DIGITAL HOsec

Depois de uma análise metódica e cuidadosa, conseguimos identificar como é que a assinatura digital do RSA funciona. O seu funcionamento é descrito a seguir:

1. A assinatura digital no RSA, a mensagem a ser assinada é inserida em uma função de hash que produz um código de hash seguro, de tamanho fixo.
2. Esse código de hash, é então, criptografado usando uma chave privada do emissor para formar a assinatura.
3. Tanto a mensagem quanto a assinatura são então transmitidas.
4. O destinatário também decriptografa a assinatura, usando a chave pública do emissor. Se o código de hash calculado combinar adequadamente com a assinatura decriptografada, a assinatura é aceite com válida.
5. Como somente o emissor conhece a chave privada, somente ele poderia ter produzido uma assinatura válida.

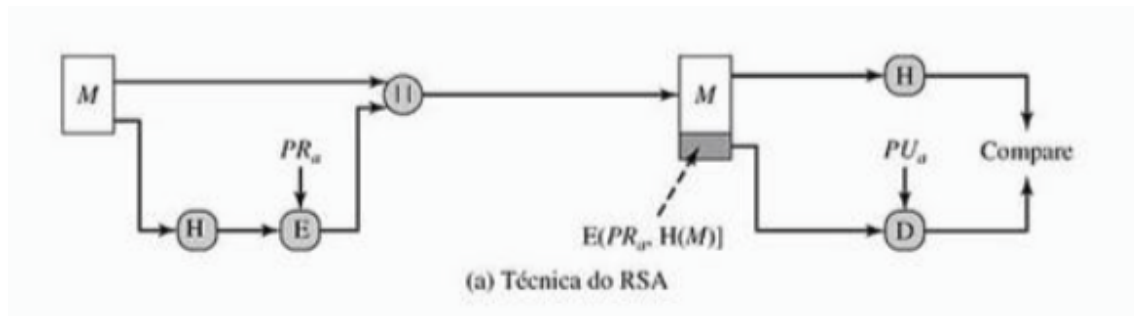
Depois desta análise do seu funcionamento devemos nos concentrar nas seguintes partes:

1. A parte da mensagem que é concatenada com o código de hash criptografado com a chave privada do emissor, notamos que ela é deixada como está (clara) e então pensamos no seguinte:
 - E se tivesse um criptoanalista no lado do emissor e conseguisse recuperar essa mensagem clara?. Então, pensamos em tornar todas as partes da mensagem ilegíveis, tornar a mensagem toda numa junção do código de hash de tamanho fixo, dispersa (sem ordem lógica)
2. Já que o código de hash tem tamanho fixo (128) não importa a mensagem que é inserida então pensamos no seguinte:
 - Dividimos os 128 caracteres em blocos de 32 posições e em cada bloco tem 4 caracteres logo:

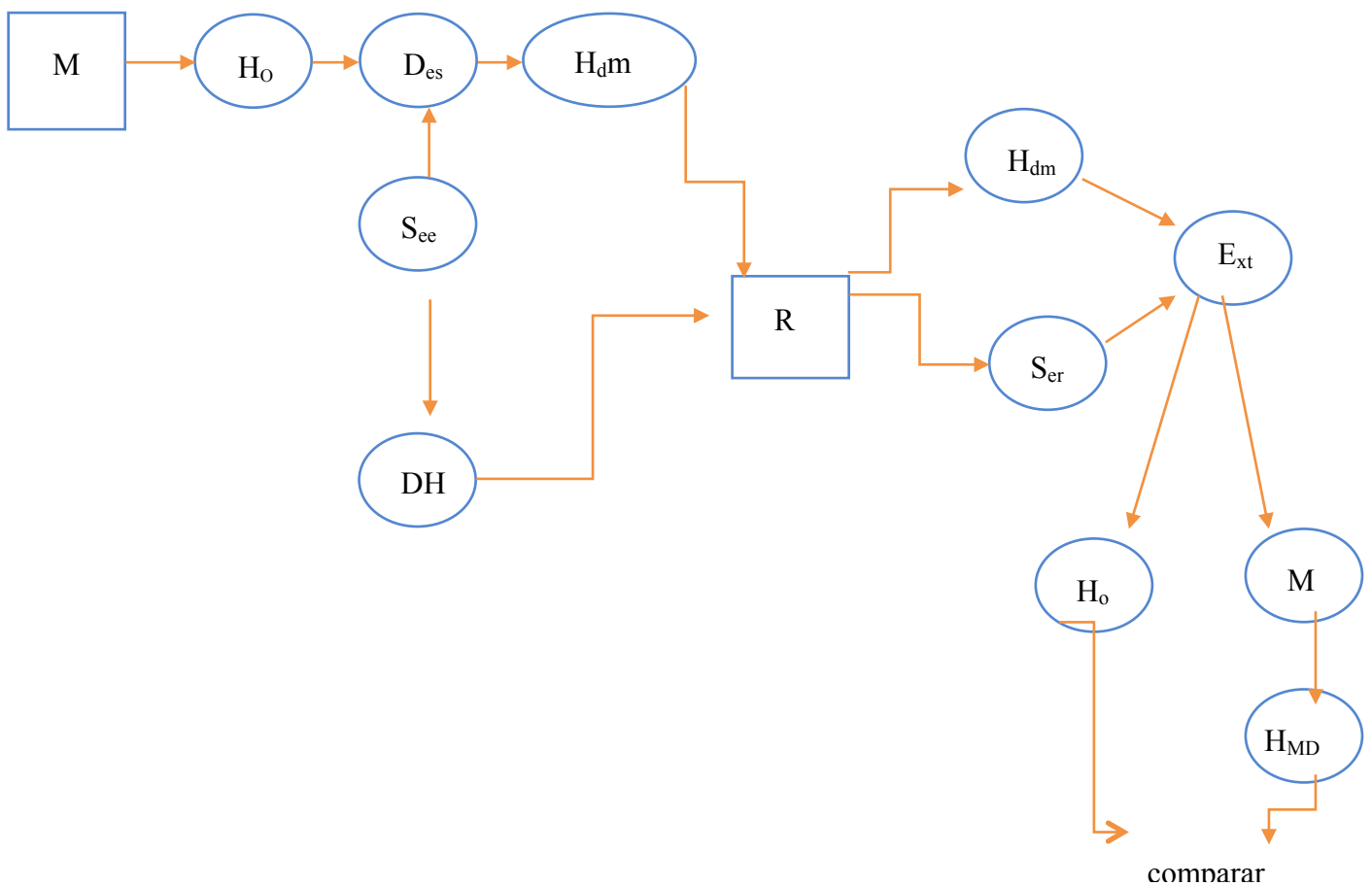
$$tf = 32p * 4c$$

3. E estas 32 posições com os 4 caracteres estão dispersos. Ou seja, os mesmos não apresentam a mesma sequência formada no código de hash
4. É fornecida uma semente, que a mesma gera números que nos possibilitam dispersar tanto o código hash junto com a mensagem
5. Esta semente é transmitida para o receptor para poder desmistificar a mensagem e assim poder lê-la
6. Este processo de envio de semente é feito por intermédio do algoritmo de Diffie Hellman

DESCRIÇÃO GRÁFICA DO SEU FUNCIONAMENTO



TÉCNICA DO HOsec



Descrição:

1. M : corresponde a mensagem a ser assinada
2. H_o : corresponde ao hash original formado pela mensagem
3. D_{es} : é a função que dispersa o hash original, a partir da semente (S_{ee}). Que é assinada na função Des
4. H_{dm} : é o hash disperso, formado pela introdução da Des no H_o
5. DH: é o algoritmo de Diffie Hellman, que foi usado para enviar a S_e para o receptor
6. R: corresponde ao receptor do hash disperso (H_{dm}), e que aplicará as seguintes transformações:
 - a. E_{xt} : é a função que permite decifrar ou faz o processo inverso para achar o H_o (contém a M imbutida) através da S_{er} (que é a semente do receptor)
 - b. Depois, extrai o H_o e a M (que estava entre o hash)
 - c. Com esta M o receptor cria seu código hash H_{MD} e compara com o H_o

Onde:

M : mensagem;

H_o : hash original;

D_{es} : dispersar;

S_{ee} : semente do emissor;

H_{dm} : hash disperso com M ;

DH: Diffie Hellman;

R: receptor;

E_{xt} : extrair;

S_{er} : semente do receptor;

H_{MD} : hash M do destino.

HOSec

DEMONSTRAÇÕES

Criar sequência:

Descrição

```
functionCreatSequence($root,&$randSequence,&$hSequence,$mess,&$NonConfusehash){
    $index=0;
    $second_index=0;
    $tam=32;
    $hReal="";

    $controlOperation=0;
    setSequenceRand($randSequence,$root);
    while($index<$tam){
        if($controlOperation==0){
            $controlOperation=1; $tam=128;
            $hReal=hash("sha512",$mess);
            $NonConfuse=$hReal;
            echo $NonConfusehash;
        }
        $index!=0 && $controlOperation!=1
?checkRepeatedElem($randSequence,$index):0;
        $hSequence[$randSequence[$second_index]]= substr($hReal, $index,
4);

        $second_index++;
        $index+=4;
    }
}
```

➤ Este algoritmo permite o seguinte:

1. Cria uma sequência
 2. Faz a divisão do hash
-

Verificar elementos repetição

Descrição

```
function checkRepeatedElem($middleArray,&$currentPostion){  
  
    $ind=-1;  
  
    while (++$ind<$currentPostion){  
  
        if($middleArray[$ind]==$middleArray[$currentPostion]){  
  
            $currentPostion--;  
            break;  
        }  
    }  
}
```

➤ Este algoritmo permite o seguinte:

1. Verifica se um determinado número na sequência já foi usado, ou seja, se não há repetição dos números usados para dispersão
-

Extrair o hash

Descrição

```
function extractHash(&$bHsh_Txt=null,$Droot=0){  
  
    srand($Droot);  
    $realLeng=32;  
    $cont=-1;  
  
    while (++$cont<$realLeng){  
  
        $bHsh_Txt[$cont]=rand()%50;  
        checkRepeatedElem($bHsh_Txt,$cont);  
  
    }  
}
```

➤ Este algoritmo permite o seguinte:

A partir do hash disperso com a mensagem esta função permite retirar a dispersão e ter o hash original

Preencher as mensagens

Descrição

```
function preencSMS(&$me__sager,&$k){

    $auxLe= strlen($me__sager);
    $control=-1;

    $pos="";
    $indexBlock=0;
    $indexf=0;
    if(strlen($me__sager)%4!=0){

        if(($auxLe+3)%4==0){$control=0; $pos='b11'; }
        if(($auxLe+2)%4==0 && $control==1){$control=0; $pos='10';}
        if(($auxLe+1)%4==0 && $control==1){$control=0; $pos='1';}
    }

    while ($indexBlock<strlen($me__sager.$pos)){

        $aux[$indexf]= substr($me__sager.$pos, $indexBlock, 4);
        $indexf++;
        $indexBlock+=4;
        $k++;
    }

    finalMind($GLOBALS["HashSequence"],$GLOBALS["forSendBlock"],$aux,$GLOBAL
ALS["randSequence"]);

}
```

➤ Este algoritmo permite o seguinte:

Permite preencher a mensagem no código de hash em variadas posições

Final mind

Descrição

```
function finalMind($bHash=null,&$bTotal=null,$plai=null,$sRan=null,$tAl=32){

    $i=-1;
    $iSend=0;

    initalBlock($bTotal,$GLOBALS["tamBlockPlain"]+32);
    while (++$i<($tAl)){
        $bTotal[$sRan[$i]]=$bHash[$sRan[$i]];
    }
}
```

```

$J=-1;

while(++$J<($tAl+$GLOBALS["tamBlockPlain"])){

    if(key_exists($J,$bTotal) && $iSend<$GLOBALS["tamBlockPlain"]){
        strcmp($bTotal[$J],")==0 ?$bTotal[$J]=$plai[$iSend++]:0;
    }
    if(strcmp($bTotal[$J],")==0 && $iSend==$GLOBALS["tamBlockPlain"]){
        unset($bTotal[$J]);
    }
}

}

```

➤ Este algoritmo permite o seguinte:


Permite verificar o tamanho dos blocos, e verifica-los até 32 que é o tamanho desejado por nós

Resumo (hash normal)

Descrição

este é o hash formado pela mensagem =>

am 79a4 34d0be63680e9a5a8b0beadb252b55c8b26232831dfed99a142dd7630039a6400a10b4f07e8a0eafca2337f6f077a288be4a91be9bf4b54f92f4522c49e

 posição antes da desorganização

Resumo (hash disperso, dividido)

Descrição

array(33) { [0]=> string(4) "07e8" [1]=> string(4) "8be4" [2]=> string(4) "00a1" [3]=> string(4) "337f" [4]=> string(4) "634d" [5]=> string(4) "e9a5" [6]=> string(4) "6f07" [7]=> string(4) "8b26" [8]=> string(4) "am10" [9]=> string(4) "3680" [10]=> string(4) "a91b" [11]=> string(4) "7a28" [12]=> string(4) "e9bf" [13]=> string(4) "79a4" [14]=> string(4) "31df" [17]=> string(4) "4b54" [23]=> string(4) "fca2" [26]=> string(4) "c49e" [27]=> string(4) "b252" [29]=> string(4) "9a64" [31]=> string(4) "0beb" [32]=> string(4) "4522" [47]=> string(4) "a8b0" [35]=> string(4) "bead" [33]=> string(4) "b55c" [38]=> string(4) "2328" [44]=> string(4) "ed99" [40]=> string(4) "a142" [48]=> string(4) "dd76" [46]=> string(4) "3003" [49]=> string(4) "0b4f" [43]=> string(4) "a0ea" [45]=> string(4) "f92f" } array(32) { [0]=> int(13) [1]=> int(4) [2]=> int(31) [3]=> int(9) [4]=> int(5) [5]=> int(47) [6]=> int(35) [7]=> int(27) [8]=> int(33) [9]=> int(7) [10]=> int(38) [11]=> int(14) [12]=> int(44) [13]=> int(40) [14]=> int(48) [15]=> int(46) [16]=> int(29) [17]=> int(2) [18]=> int(49) [19]=> int(0) [20]=> int(43) [21]=> int(23) [22]=> int(3) [23]=> int(6) [24]=> int(11) [25]=> int(1) [26]=> int(10) [27]=> int(12) [28]=> int(17) [29]=> int(45) [30]=> int(32) [31]=> int(26) }

CONCLUSÃO

Chegamos a conclusão que a assinatura digital é mesmo uma das mais importantes técnicas dos algoritmos de chave pública, e que foi bom este desafio visto que conseguimos conciliar a teoria com a prática.

A dificuldade na integração do algoritmo de Diffie Hellman condicionou-nos durante algo tempo, mas demos o nosso jeito.

Esperamos que este artigo possa ser analisado e melhorado por outros profissionais que tenham interesse em segurança da informação e não só.

BIBLIOGRAFIA

Criptografia e segurança de redes 4ª edição

Serious Cryptography. Copyright 2018 by Jean-Philippe Aumasson. ISDN-10:1-59327-826-8; ISDN-13: 978-1-59327-826-7

[S._C._Coutinho]_Numeros_inteiros_e_criptografia_R(z-lib.org)

Cryptanalytic attacks on RSA; Song Y.Yan; University of Bedfordshire, UK; Massachusetts Institute of Technology, USA

RSA security's official guide to Cryptography; Steve Burnett and Stephen Paine; Osborne/McGraw-Hill, New York...