DI/FCT/UNL - Mestrado Integrado em Engenharia Informática
Segurança de Redes e Sistemas de Computadores 2º Sem. 2017/2018
Teste de frequência nº 1 (14/Abril/2018)
Parte I (parte sem consulta) - Duração: 1h15m

Questão 1

Tenha em conta o domínio da terminologia correta e a definições de serviços, mecanismos e propriedades de segurança, no seu estudo da *framework* de referência de segurança OSI X.800 para responder às seguintes questões:

a) Qual a diferença entre *Data-Integrity* e *Traffic-Flow Integrity* ?

b) Qual a diferença entre as noções de *Peer-Authentcity* (ou autenticidade de principal) e *Data-Origin-Authenticity* (ou autenticidade de origem de mensagem) ? Dê exemplos de uma e outra noção num contexto aplicacional específico.

Peer-Authenticity (e exemplo)

Data-Origin-Authenticity (e exemplo):

c) Comparativamente às definições das propriedades e serviços de segurança OXI X-800, a normalização NIST igualmente estudada apenas categoriza 3 requisitos raiz associados a propriedades de segurança, conhecidos como *CIA triad*: *Confidentaility, Integrity, Availability*. Isso significa que algumas das propriedades de *framework* X.800 não estão cobertas ?

d) Na definição de "mecanismos de segurança", são classificados dois tipos de mecanismos: mecanismos específicos e mecanismos permeados (ou *pervasivos*). Qual a diferença entre um mecanismo de segurança específico e um mecanismo de segurança permeado (ou *pervasivo*) ? Dê exemplos.

Um mecanismo é específico se ...

Um mecanismo é permeado (ou *pervasivo*) se ...

e) Na seguinte matriz que mapeia conceptualmente serviços de segurança em mecanismos concretos de segurança usados para a sua construção (escritos em língua inglesa) , estabeleça as diferenças entre "*Peer-Authentication*", "*Data Origin-Authentication*", "*Traffic-Flow-Confidentiality*", "*Data-Confidentiality*", preenchendo corretamente estas linhas com "Y" nas colunas corretas associadas aos mecanismos de proteção que podem ser usados ou combinados na base de suporte de cada um dos serviços indicadas (como nas que já se encontram corretamente preenchidas nos restantes casos).

Mechanism \ *Service*	AES	Public Key Signature	Access Control Table	Alg. HMAC ou CMAC	SHA-256	Password-Based Encryption	Traffic Padding	Routing Control	Notari-Zation and Authentication Exchange
Peer-Entity Authentication									
Data-Origin Authentication									
Access Control			Y						
Traffic Flow Confidentilaity									
Data Origin Integrity									
Non Repudiation		Y			Y				Y
Availability				Y	Y			Y	Y

Questão 2. Indique, para cada uma das seguintes afirmações, se é verdadeira (V) ou falsa (F). Não sendo obrigatório, se achar conveniente, pode justificar ou argumentar sobre as afirmações que considerar serem FALSAS, acrescentando a sua justificação em anexo, indicando a respetiva alínea**.**

Afirmação	**V ou F**
a) Uma construção criptográfica HMAC é um mecanismo específico de segurança que suporta garantias de segurança diferentes de uma construção CMAC	
b) Uma construção criptográfica HMAC que usa na sua parametrização a função SHA-512 é sempre mais rápida de computar do que uma construção CMAC que usa na sua parametrização o algoritmo AES com chave de 256 bits	
c) A utilização de MACs num protocolo de segurança defende de ataques à disponibilidade	
d) A utilização de um algoritmo criptográfico simétrico, como mecanismo específico, permite estabelecer prova de autenticação associada à noção de *Peer-Authenticity*	
e) As melhores garantias de robustez do ponto de vista de criptanálise em relação á segurança de um algoritmo criptográfico simétrico, é quando se prova a sua resistência a análises ou ataques do tipo "*chosen text*" , comparativamente ás tipologias "*ciphertext-only*", "*known-plaintext*", "*chosen ciphertext*" ou "*chosen plaintext*".	
f) O algoritmo Triple DES usando uma chave de 168 bits é constituído por uma sequência de processamento do algoritmo DES na forma *Encrypt-Decrypt-Encrypt*, porque essa sequência é mais segura do que se fosse usada a forma *Encrypt-Encrypt-Encrypt*	
g) Um algoritmo criptográfico simétrico ou uma função segura de síntese (*Secure Hash Function*), como por exemplo AES, ou SHA-256, pode ser usado como componente nuclear de uma função para geração pseudo-aleatória (ou PRF – *Pseudo Random Function*).	
h) No uso de métodos criptográficos simétricos, uma das vantagens de uso do modo CTR comparativamente a usar CBC operado num algoritmo criptográfico simétrico é que o processamento de cifar e decifra é paralelizável, eficiente, seguro e a operação de decifra pode ser feita usando apenas a função de cifra.	

i) No uso de métodos criptográficos simétricos, uma das vantagens de uso de modos como GCM ou CCM, comparativamente a CBC, é que dessa forma se pode ter uma prova implícita de autenticidade e integridade inerente á informação cifrada, o que pode evitar assim a necessidade de usar uma computação adicional do tipo HMAC ou CMAC para esse efeito	
j) Se estiver a processar um método de cifra de bloco usando um modo sem vetor de inicialização e *padding* PKCS7, se o tamanho do *plaintext* é múltiplo do tamanho do bloco base do algoritmo em causa, então o tamanho do *ciphertext* irá ser igual ao tamanho do *plaintext*	
k) Um algoritmo que implementa uma construção criptográfica do tipo PBE - *password-based encryption* constitui um mecanismo específico que permite assegurar confidencialidade	

Questão 3. Dadas as seguintes variantes V1 a V4 de construções criptográficas para proteger dados *plaintext* Mp (para garantias equivalentes à proteção de um *payload* Mp tal como na definição do protocolo SGCM-TLP na implementação do trabalho prático TP1), qual a que selecionaria , porquê e porque consideraria ser melhor face ao uso de cada uma das restantes variantes ?

V1: $E\ (K_S,\ [\ Mp\ ||\ MAC_{KM}\ (Mp)\ ]\ )\ ||\ MAC_{KA}\ (C)$

V2: $E\ (K_S,\ [\ Mp\ ||\ MAC_{Ks}\ (Mp)\ ]\ )\ ||\ MAC_{KA}\ (C)$

V3 $E\ (K_S,\ [\ Mp\ ||\ MAC_{Ks}\ (Mp)\ ]\ )\ ||\ SHA256\ (C)$

V4: $E\ (K_S,\ [\ Mp\ ||\ MAC_{Ks}\ (Mp)\ ]\ )\ ||\ MAC_{Ks}\ (C)$

NOTA: *Em todas as variantes, **C** refere o conteúdo ciphertex resultante da computação da parcela esquerda da concatenação em cada caso.*

Advogaria como melhor a variante: _____, e comparativamente com as restantes,

_____ é melhor do que a _____ porque:

_____ é melhor do que a _____ porque:

_____ é melhor do que a _____ porque:

DI/FCT/UNL - Mestrado Integrado em Engenharia Informática
Segurança de Redes e Sistemas de Computadores 2º Sem. 2017/2018
Teste de frequência nº 1 (14/Abril/2018)
Parte II (parte com consulta) – 1h 30 m
(Questões 4 e 5 serão contabilizadas como avaliação individual do Trabalho TP2)
Questão 6 será avaliada como componente do Teste 1

Questão 4

a) De acordo com as definições, terminologia e conceitos da framework X.800, considerando os serviços associados às propriedades de segurança que foram concretizados **na realização do trabalho prático nº1**, indique quais os ataques ativos e quais os ataques passivos no seu modelo de adversário para os quais estão implementadas as necessárias contra-medidas e em que consiste.

Ataques ativos protegidos:

Ataques passivos protegidos:

b) Considerando os serviços de segurança do protocolo de comunicação multiponto do trabalho prático nº1 (fase 1) indique quais os mecanismos específicos de segurança que foram utilizados e que propriedade de segurança implementam.

Lista de mecanismos específicos utilizados:	< ---------- >	Serviço de segurança

Questão 5

Considere a proteção estabelecida pelos mecanismos e serviços de segurança da sua implementação do trabalho prático nº 1 – fase 1 (protocolo SGCM-TLP).

Um atacante no canal de comunicação, capaz de descartar mensagens no canal executa as seguintes operações:

- Vai escolhendo seletivamente datagramas UDP trocados entre os *endpoints <IP porto>* envolvidos nas comunicações *multicast*, grava e impede que sejam recebidas pelos destinatários em causa.
- Mais tarde, decide arbitrariamente voltar a retransmitir datagramas que gravou previamente (sem qualquer alteração), sendo então estas recebidas pelos destinatários em causa

a) Considerando a sua implementação, a partir das referidas operações, o atacante conseguirá (assinale com X):

_____ realizar ataques de *message replaying*

_____ realizar desordenações ilícitas das mensagens no fluxo de tráfego de mensagens

_____ realizar um ataque que quebra as garantais de *traffic-flow integrity*

_____ realizar um ataque que quebra garantais de *traffic-flow confidentiality*, conseguindo identificar o protocolo aplicacional que se encontra protegido.

Justificação (nomeadamente no que consegue ter sucesso e no que não conseguirá e porquê).

b) Como se proporia iterar a concepção da sua implementação protocolo SGCM-TLP para inviabilizar o sucesso do atacante em relação a cada uma das ações que selecionou com X ?

Questão 6

Considere o seguinte código (ANEXO) que utiliza construções criptográficas (Java JCA/JCE) executadas nos dois troços do programa identificadas como emissor (que faz as operações de cifra) e o receptor (que faz as operações de decifra e demais verificações).

Este programa compila corretamente. Porém existem dois problemas.

- Problema A) impede que o programa processe corretamente, gerando exceções que o farão abortar.
- Problema B) não causará que o programa aborte no seu processamento mas corresponde a uma deficiente prática (ou prática criticável) na programação, do ponto de vista das propriedades de segurança que estão a ser implementadas.

a) Identifique o problema A) e diga como proporia ser corrigido.

b) Identifique o problema B) e diga como proporia ser corrigido.

ANEXO

```
import java.security.Key;
import java.security.MessageDigest;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.Mac;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class Test1
{
    public static void main(String[] args)
        throws Exception
    {
        byte[] keymaterial={0x71,0x32,0x1B,0x4C,0x33,0x12,0x69,0x08,
                            0x22,0x10,0x77,0x14,0x63,0x1C,0x5F,0x19};
        Key key = new SecretKeySpec(keymaterial, "AES");
        // Ciphersuite that will be used by Alice and Bob
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS7Padding");
        Mac mac = Mac.getInstance("DES");
        Key macKey = new SecretKeySpec(keymaterial,"DES");

        // Alice will send encrypted a string w/ a creditcard number to Bob
        String creditcard = "NR CREDIT CARD: 1234987612349876";
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] cipherText =
            new byte[cipher.getOutputSize(creditcard.length() + mac.getMacLength())];
        int ctLength =
            cipher.update(StringToByteArray(creditcard), 0, creditcard.length(),
               cipherText, 0);
        mac.init(macKey);
        mac.update(StringToByteArray(creditcard));
        ctLength += cipher.doFinal(mac.doFinal(), 0, mac.getMacLength(), cipherText,
               ctLength,0);

…… // Alice sends the cipherText to Bob ...
// .... here is the attacker in the channel
//
// Bob: received the cipherText from Alice possibly tampered by the attacker

        // Bob will retrieve in AliceCard credit card number sent from Alice
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte[] plainText = cipher.doFinal(cipherText, 0, ctLength);
        int    messageLength = plainText.length - mac.getMacLength();
        String AliceCard= ByteArrayToString(plainText, messageLength); // got it !
    }
public static byte[] StringToByteArray(String string)
    {
        byte[]  bytes = new byte[string.length()];
        char[]  chars = string.toCharArray();
        for (int i = 0; i != chars.length; i++)  bytes[i] = (byte)chars[i];
        return bytes;
    }
    public static String ByteArrayToString(byte[] bytes, int length)
    {
        char[]  chars = new char[length];
        for (int i = 0; i != chars.length; i++)  chars[i] = (char)(bytes[i] & 0xff);
        return new String(chars);
    }

}
```

```
import java.security.Key;
import java.security.MessageDigest;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.Mac;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class Test2
{
    public static void main(String[] args)
        throws Exception
    {
        // This is the key shared between Alice (the "sender" and
        // Bob (the "receiver")
        byte[] keymaterial={0x71,0x32,0x1B,0x4C,0x33,0x12,0x69,0x08,
                            0x22,0x10,0x77,0x14,0x63,0x1C,0x5F,0x19};
        Key key = new SecretKeySpec(keymaterial, "AES");
        // Ciphersuite that will be used by Alice and Bob
        Cipher cipher = Cipher.getInstance("AES/ECB/NoPadding");
        Mac mac = Mac.getInstance("RC5");
        Key macKey = new SecretKeySpec(keymaterial,"RC5");

        // Alice will send encrypted a string w/ a creditcard number to Bob
        String creditcard = "NR CREDIT CARD: 1234987612349876";
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] cipherText =
         new byte[cipher.getOutputSize(creditcard.length() + mac.getMacLength())];
        int ctLength =
         cipher.update(StringToByteArray(creditcard), 0, creditcard.length(),
          cipherText, 0);
        mac.init(macKey);
        mac.update(StringToByteArray(creditcard));
        ctLength += cipher.doFinal(mac.doFinal(), 0, mac.getMacLength(), cipherText,
          ctLength);

        // Alice sends the cipherText to Bob ...
        //                 |   .... here is the attacker
        //                 V
        // Bob: received the cipherText from Alice

        // Bob will retrieve in AliceCard credit card number sent from Alice
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte[] plainText = cipher.doFinal(cipherText, 0, ctLength);
        int    messageLength = plainText.length - mac.getMacLength();
        String AliceCard= ByteArrayToString(plainText, messageLength); // got it !
    }

    public static byte[] StringToByteArray(String string)
    {
        byte[] bytes = new byte[string.length()];
        char[]  chars = string.toCharArray();
        for (int i = 0; i != chars.length; i++)
            bytes[i] = (byte)chars[i];
        return bytes;
    }
    public static String ByteArrayToString(byte[] bytes, int length)
    {
        char[]  chars = new char[length];
        for (int i = 0; i != chars.length; i++)
            chars[i] = (char)(bytes[i] & 0xff);
        return new String(chars);
    }
```