	Title : CS4614 Study
	Student Name : Brian O Regan
	Student Number : 110707163
	Module : CS4614
	Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**Summer 2013**

<p><b>Alice sends message M to Bob over a untrusted network. Assuming they share a secret, sketch how message secrecy, integrity and authentication should be provided. (6 marks)</b></p> <p><b>A</b> <i>If Alice and Bob share a strong cryptographic secret key <math>K_{AB}</math> then</i></p> $A \rightarrow B : E_{CBC}^{IV}(K_{AB}, M, h(M))$ <p><i>where <math>h()</math> a one-way cryptographic hash function and <math>E_{CBC}^{IV}()</math> is a block cipher configured for encryption in CBC mode with fresh initialization vector IV.</i> <span style="float: right;"><u>A</u></span></p>
<p><b>Explain how salt defends against a password pre-computation dictionary attack. (6 marks)</b></p> <p><b>A</b></p> <p>Looking for explanation of dictionary attack using hash-table of dictionary (pass)words and how salt can significantly increase its size. A</p>
<p><b>In the movie Skyfall, James Bond’s Walther PPK handgun has a biometric reader designed to recognise his palm print, so that only he can fire it. Explain whether the designers of this authentication mechanism need to worry about the Birthday Paradox. (6 marks)</b></p> <p><b>A</b></p> <p>Intuitively, the Birthday Paradox tells us that the probability that k agents will all have distinct palm prints is less than 0.5 if <math>k &gt; \sqrt{1/FAR}</math>, where FAR is the false accept rate for the biometric system. The designers do not have to worry about the birthday paradox since the handgun only has to recognize James Bond’s palm and nobody else. It does not store biometrics for a group of agents, and therefore the birthday paradox does not apply. For the given gun, the probability of someone else being false recognized as James Bond is given by FAR.</p>
<p><b>d) Alice receives a document signed by Bob and a certificate for his public key. Sketch the operations carried out by Alice to confirm the document’s authenticity. (6 marks)</b></p>



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**A** Bob has public, private key pair  $(K_B, K_B^{-1})$ . Let  $\{Bob, expiryDate, \dots, K_B\}_{aK_T}$  and  $\{Doc\}_{aK_B}$  denote Bob's certificate and signed document, respectively. We assume that Alice knows of and trusts the signing CA  $K_T$ . The signed document corresponds to the document and the hash of the document encrypted using Bob's private key  $K_B^{-1}$ .

Alice checks the the signature on the document by re-computing the hash of the document and comparing it against the encrypted hash provided (decrypted using Bob's public key). If valid then Alice checks the validity of the certificate for  $K_B$  by confirming that its properly signed by a CA she trusts and that it has not expired. She may also consult a CRL or OCSP server (if specified in the certificate) to check whether the certificate is revoked. If the document signature and certificate is valid then Alice can believe that the document was signed by Bob.  $\overline{A}$

e) If Alice and Bob know each other's public keys ( $K_A$  and  $K_B$ , respectively) and  $K_{AB}$  is a session key, then explain which of the following provide a digital signature for message  $M$ .

$$A \rightarrow B : \{M\}_{K_{AB}}, \{\{h(M), K_{AB}, A, B, \dots\}_{K_A^{-1}}\}_{K_B} \quad (1)$$

$$A \rightarrow B : \{M, h(M)\}_{K_{AB}}, \{\{K_{AB}, A, B, \dots\}_{K_A^{-1}}\}_{K_B} \quad (2)$$

(6 marks)

**A** In Protocol (1), Alice signs the hash of the message  $M$  and thus, its not possible for Bob to for example modify  $M$  and claim it came from Alice. In Protocol (2), only the shared key is signed, and thus Bob can modify the message, recompute the hash, and claim that  $\{M', h(M')\}_{K_{AB}}$  came from Alice.  $\overline{A}$

2. Alice  $A$  and Bob  $B$  share secret keys  $K_{AT}$  and  $K_{BT}$ , respectively, with trusted authentication server  $T$ . Alice wishes to communicate securely with Bob and initiates the following protocol.

Msg1 :  $A \rightarrow B : A$

Msg2 :  $B \rightarrow T : A, B$

Msg3 :  $T \rightarrow B : \{K_{AB}\}_{K_{AT}}, \{B\}_{K_{AT}}, \{K_{AB}, A\}_{K_{BT}}$

Msg4 :  $B \rightarrow A : \{K_{AB}\}_{K_{AT}}, \{B\}_{K_{AT}},$

a) Describe how this protocol should be used to provide authenticated secure access to network resources. Highlight how it is different to a Kerberos-style protocol. (15 marks)

**A** Looking for straightforward description of how a protocol establishes secure and authentic key between Alice and Bob. This is not unlike relationships in a Kerberos style protocol, however the description needs to explain the slightly different setup between initiator, respondent and authentication service.

*In Kerberos, the assumption is that a connection exists between the authentication server and the initiator/Alice, while it is not necessary to have a direct connection between the the authentication server and the respondent/Bob. In the above protocol, the implicit assumption is that the connection is between the respondent and the authentication server while a connection between the initiator and authentication server is not necessary. In Kerberos the initiator/Alice directly contacts the authentication server for a ticket for the service/respondent/Bob. In the above protocol, the respondent Bob must contact the authentication server for a ticket for the initiator/Alice. For example, in a conventional workstation setup under Kerberos, the workstation contacts the Kerberos server. The above protocol could be used, for example, when a smart card (initiator) does not have a connection to an authentication server, but the smart card reader (respondent) does have a connection.*

**A**

b) Describe an attack on the protocol whereby a malicious user Mike can trick Alice into believing that she is initiating a secure connection with Bob (but it is actually Mike).

(10 marks)

**A** Mike eavesdrops on past run  $\alpha$  of the protocol between Alice and Bob and extracts attribute  $\{B\}_{K_{AT}}$  from

Msg $\alpha$ 3 :  $T \rightarrow B : \{K_{AB}\}_{K_{AT}}, \{B\}_{K_{AT}}, \{K_{AB}, A\}_{K_{BT}}$

Mike, a participant, shares a key  $K_{MT}$  with  $T$  and initiates another run  $\beta$  with  $T$ :

Msg $\beta$ 2 :  $M \rightarrow T : A, M$

Msg $\beta$ 3 :  $T \rightarrow M : \{K_{AM}\}_{K_{AT}}, \{M\}_{K_{AT}}, \{K_{AM}, A\}_{K_{MT}}$

Mike keeps a copy of  $\{K_{AM}\}_{K_{AT}}$  and obtains key  $K_{AM}$  by decrypting  $\{K_{AM}, A\}_{K_{MT}}$ .

Mike sets himself up on the network, pretending to be Bob and waits for Alice to connect in run  $\delta$ .

Msg $\delta$ 1 :  $A \rightarrow B[M] : A$

Mike responds to Alice with the attributes collected from runs  $\alpha$  and  $\beta$ :

Msg $\delta$ 4 :  $B[M] \rightarrow A : \{K_{AM}\}_{K_{AT}}, \{B\}_{K_{AT}},$

on receipt of message  $\delta$ 4, Alice believes she is sharing the key  $K_{AM}$  with Bob.

**A**



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

3. UCC lecturer Alice securely submits exam results to a network-based Exams-Office service using the Java code fragment below. Results `rsIts` are sent over a socket-based connection (encapsulated as `DataOutputStream out`). Alice's Java KeyStore `keystore` stores her public DSA key, alias "alicePK".

```
Random rangen = new Random(0);
byte[] keySession = new byte[2];
rangen.nextBytes(keySession);
SecretKeyFactory desF = SecretKeyFactory.getInstance("DES");
KeySpec ks = new DESKeySpec(keySession);
SecretKey key = desF.generateSecret(ks);
Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, key);
byte[] cBytes= cipher.doFinal(rsIts);
out.write(cBytes);

String alice= "alicePassword".toCharArray();
PrivateKey priv = (PrivateKey) keystore.getKey("alicePK",alice);
Signature signature = Signature.getInstance("DSA");
signature.initSign(priv);
byte[] sig = signature.sign(keySession);
out.write(sig);
```

- a) Identify and explain the security vulnerabilities in this implementation. (15 marks)

**A** Code implements the protocol  $A \rightarrow B : \{msg\}_K, \{K\}_{sK_A}$

*Protocol problems: Secrecy is not provided by the protocol since it does not protect the secrecy of  $K$ : any principal can read the key in  $\{K\}_{sK_A}$ , since it is only signed. Once in possession of the key  $K$ , the principal can decrypt  $\{M\}_K$  and read the message  $M$ . Integrity is not provided since, given that the key  $K$  can be determined by any principal, then any principal (Eve) can change the message to  $M'$  and re-encrypt with  $K$  and  $B$  cannot detect the modification. Authentication is not provided since, given that the key  $K$  can be determined by any principal, then any principal (Eve) who sees a past exchange can generate a message and pretend that it came from  $A$ .*

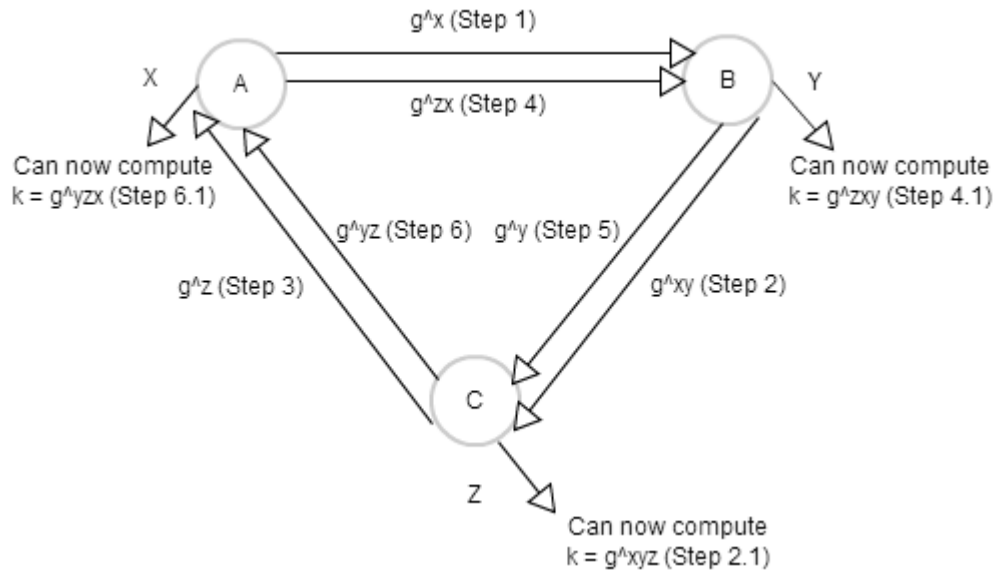
*Code Problems: Should not have a password to the keystore hard coded within a program. `java.util.Random` is not a secure random number generator: an attacker can predict in advance future random numbers/session keys. The same value is used to seed the random number generator and thus every key will be the same. The effective session key size is only 16 bits, which is vulnerable to a brute force attack. The message is encrypted using DES-ECB mode, which is vulnerable to cut-paste attack and traffic analysis. **A***

- b) It has been suggested that it would be better to use Java SSL to secure the connection between Alice and Bob. Outline how Java SSL should be used in this case and include an explanation of how the use of public key certificates in the protocol can help Bob to discover Alice's public key. (10 marks)

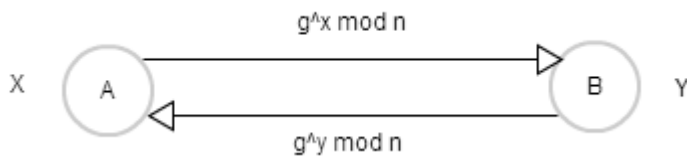
**A** Looking for a general discussion of Java SSL and in particular the configuration of the key-manager and trust manager for the above problem. This should also include discussion of the certificates, certificate-chains and the trusted third party. I dont expect students to generate complete syntax/working code with all the details. But what I am looking for is an explanation of the components and how they t together. **A**

**Notes**

**Diffie-Hellman – 3-Way**



**Diffie-Hellman – Basic**





Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

## Exercise Sheet 1 - Answers

Note: Number 4 in previous years exercise is left out in this years. Also number 16 in this years is not in last years.

### Important Topics

- Meet in the Middle Attack: 3.33
- MACs: 3.47
- Rainbow Table: 4.14
- Birthday Paradox: 4.19
- Keyed One Way Hash Function: 4.25

KSG = Key Stream Generator

MAC = Message Authentication Code

FAR = False Accept Rate

ECB = Electronic Code Book

CBC = Chain Block Cipher

A 256 bit message is made up of four 64-bit blocks  $b_3, b_2, b_1, b_0$ . It is encrypted using DES-CBC to cipher blocks  $c_3, c_2, c_1, c_0$ , where

$$\begin{aligned}c_0 &= E_{DES}(k, b_0) \\ C_i &= E_{DES}(k, b_i \oplus c_{i-1}) \quad [i > 0]\end{aligned}$$

Q1 How should DES-CBC be used to decrypt  $c_3, c_2, c_1, c_0$ , given key  $k$ ?

If the first block has index 1, the formula for CBC encryption is  $C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV$

while the formula for CBC decryption is  $P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV$ .

so to answer the above..

$$P_1 = D_K(C_1) \oplus C_{0-1}, C_0 = IV$$

$$P_2 = D_K(C_2) \oplus C_{2-1}$$

$$P_3 = D_K(C_3) \oplus C_{3-1}$$

(Alan R)

I think he left out a bit of it. from where he says  $C_i = E.Des(K, b_i \text{ XOR } C_{(i-1)})$  means that there is chaining from the previous cipher block, he says  $C_0 = E.Des(K, b_0)$  but i think it should be  $C_0 = E.Des(K, b_0 \text{ XOR } IV)$  where IV is the initialisation vector

(Alba)

$$(A \text{ xor } B) \text{ xor } B = A$$

$$\text{So } D_{des}(k, c_i) \text{ XOR } c_{i-1} = b_i \text{ XOR } c_{i-1} \text{ XOR } c_{i-1}$$

$$\text{Finally } D_{des}(k, c_i) \text{ XOR } c_{i-1} = b_i$$



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

There is also no IV in this question.

---

## Q 2

**Explain the desirable properties of a one-way hash function and discuss the implications of the birthday attack when designing a hash function.**

A hash function  $h$  maps arbitrary length value  $x$  to fixed length value  $y$  such that:

- **Hard to reverse.** Given value  $y$  not feasible to find  $x$  with  $y = h(x)$ .
- **Collision freeness.** Hard to find values  $x, x'$  such that  $h(x) = h(x')$ .
- **Unpredictability.** The hash value  $h(x)$  does not give any information about any part of its operand  $x$ .

The birthday attack / paradox states that: With **23 people in room there's more than 50% chance 2 share the same birthday.** The output of a collision-resistant hash function needs to be at least  $2^n$  bits large if collision search is to be infeasible.

The reason this is an issue is finding  $(x, x')$  such that  $h(x) = h(x')$ ; Probability of no match after  $K$  tests is:  $(2^n)! / [(K!).(2^{n-k})]$  => is less than .5 when  $k$  is roughly  $\sqrt{2^n}$  if  $2^n$  is considered sufficiently infeasible (for a cpu) then the output of a collision-resistant hash function needs to be at least  $2n$  bits large if collision search is to be infeasible.

---

## Q 3

**Recommend which DES mode (ECB or **CBC**) should be used to hash passwords in a password file? Explain your answer.**

**ECB = Electronic Code Book**

**CBC = Cipher Block Chaining**

ECB has been used and has been hacked in the past:

Example, Windows LAN Manager

- Turn password into 14-character string, either by truncating longer passwords or padding shorter passwords with nulls.
- Using each 7-char string as a DES key, encrypt a fixed constant with each key, yielding two 8-byte encrypted strings.
- Concatenate the two strings together to create 16-byte hash value.

Dictionary attacks are easy against this scheme.

- Most people pick easily guessable passwords.
- No salt values used. Easy to build a dictionary of hash values.
- The two 7-char 'halves' of password are hashed independently. Brute force halves independently; Complexity of two halves same as complexity of one half. Easy to recognize passwords less than 7 characters (second half of key is all nulls).



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

ECB Open to:

- A brute force dictionary attack (assuming users pick easy password, such as only lowercase letters)
- A pre-computation dictionary attack. Main cost is storage of table. Cheap cost to lookup table. Attacker stores hashed passwords as
  - word : password, where password = h(word)
- Patterns in the plaintext show through in the ciphertext.
- Is subject to a cut and paste attack where ciphertext blocks can be replaced without effecting any of the surrounding ciphertext blocks.

In general ECB mode encryption should not be used.

CBC in comparison overcomes many of the problems with ECB listed above and as a result I believe it should be used to hash passwords in a password file. CBC used with MACs not only provides integrity of data, but also provides secrecy if extra precautions are taken. (ie. a second cryptographic pass through the message)

---

**Q 4**

**Could a keyed one-way hash function be used to protect the secrecy of data? Explain your answer.**

A keyed hash function  $h_K(M)$  provides a hash-based implementation of a message authentication code.

(Vlad's Solution)

A **HMAC is not about secrecy, but about integrity**. (Making sure the text hasn't been altered on the way to destination).

So, if we both want to communicate, and we share a key, I can send you a message (not encrypted, anyone can read it), but in order to make sure it hasn't been altered, I also provide the hmac signature of that message. When you get it, you compute the HMAC, and you compare it against the one I've sent.

If they're different, the message has been tampered with, and you shouldn't trust it.

So, HMAC cannot be used to provide secrecy of information, but integrity. **If you want to achieve secrecy, use a symmetric key based algorithm (DES or AES) that allows you to encrypt, and decrypt. HMAC, its one way!**

---

**Q 5**

**Why is it preferable not to store secrets such as passwords in a system?**

Should the system become compromised it may be feasible for an attacker to discover these secrets.





Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

For example, an attacker will know that an uneducated (in security protection) user will use easily guessable passwords. Such as all lower case, initials, common words or phrases. The attacker may then build a dictionary / several dictionaries of recomputed passwords (to carry out a pre-computation attack) and there relevant hashes (should the attacker know that they system passwords are stored with hashes). Should there be a match between a system password and a password in the recomputed table of hashed passwords it will be found will relatively easy effort.

Better Answer (Need to explain better - 4.09)

If passwords are stored in a system, it may become an issue when the systems security is upgraded. For example when the system is upgraded, it may use different security protocols to authenticate users, however the old passwords may be stored in such a way that they cannot be properly used with the systems new level of security. In this case both versions of system authentication will be used. An attacker may then concentrate their attack on the weaker form of authentication, and then use this to find a password for the stronger form of authentication. E.g. Windows NTLM Hash function.

---

#### Q 6

**Some car alarms uses a (wireless) remote control to arm/disarm the system. When the button on the remote-control is pushed, it broadcasts a unique code known only to the remote control and the car. A car-alarm that receives the correct code, toggles between arm/disarm.**

**(a) Outline a replay attack that could be used to disarm the car.**

If the authentication-interaction is always the same then a replay attack is possible. An attacker may be listening on the frequency that the remote-control and the car is communicating on. The attacker would only need to copy this 'unique code' and then send this code again, masquerading as the remote-control, at a later date.

**(b) Suggest an improved mechanism that is resilient against replay attacks. Discuss any disadvantages of your scheme.**

A challenge should be used between the remote-control(R) and the car(C) such that:

Msg1 : R -> C : remoteid  
Msg2 : C -> R : challenge  
Msg3 : R -> C : {challenge}KRC  
-> Car may now be armed/disarmed by the remote-control.

An appropriate challenge may be the current time, where the issued challenge becomes unusable after 10(10 seconds because sending a signal communication from the remote to the car should not take any longer than 10 seconds) seconds.

This means that an attacker may not use an issued challenge to authenticate after 10 seconds has passed, however a good attacker will be able to utilise this challenge within the 10 seconds.

(Richard's Solution)



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

To simplify it, I think a **timestamp along with the remote id** would suffice encrypted under KRC.

Due to the nature of the system it would only take a few seconds (max) for the broadcast to reach the car meaning the time to live of the timestamp would only be maybe 5 seconds. This means the replay attack is pointless and infeasible as the owner will be in or near the car.

Adding a challenge to this introduces state and a two-way communication method on both devices which I'd say would over complicate it.

To make it safer you could add a constant value(e.g. one signal unlocks car, another locks car) for unlocking/locking and hash this also, That means if the driver is walking away from the car and locks it a replay attack would have no effect as the car is already locked and/or the time to live on the timestamp is up

**{timestamp, action, remoteid}KRC**

*Own Note: Timestamp may also be pointless, if the replay attacker can modify the signal to include the current timestamp (e.g. set the timestamp for the time they are going to steal the car, my solution would be the inclusion of the timestamp and a NFID (near frequency ID) tag that has a radius of only a few inches which unlocks the doors. The replay attacker will then need to get within a very close proximity to the owners car keys to pick up the NFID signal before also requiring the frequency to disarm the car alarm and perform a reply attack. This also plays into the what you know and what you have and will also serve as a double function of locking the doors when the keys are away from the car !*

#### Q 7

A Bank's ATM cards have a magnetic strip on one side. This strip holds details about the account number and PIN (Personal Identification Number) of the customer. The Bank's IT department has decided that the fields

**( {PIN}KB, {AccountID}KB )**

should be stored on this magnetic strip. {PIN}KB gives the PIN encrypted under a symmetric key KB, where KB is a key known only to the Bank (and its ATM machines). An ATM uses key KB to validate the PIN, entered by the customer, against that on the ATM card before allowing any activity on the account.

Outline a simple attack on this scheme, whereby a criminal can gain access to another customer's account and does not need to know the customer's PIN. Propose an improved scheme for ATM cards and briefly explain why your proposal is secure.

The ATM simply checks to see if the pin entered by the user is the same as the pin stored on the ATM card, if they match the user is granted access to the accountID stored on the card.

The card is open to a **cut and paste attack**. Here the attacker simply makes another card, pastes in the victims AccountID encrypted under key KB(Possibly obtained from the victims own ATM card), and then pastes in a different PIN to the card. This would be a pin that the attacker knows. The



Title : CS4614 Study  
Student Name : Brian O Regan  
Student Number : 110707163  
Module : CS4614  
Exam Date: Wednesday 17<sup>th</sup> December 2014 @ 16.30

attacker now has a card which strip contains the victims accountID but another PIN. The attacker can login to the victims account.

Short-hand -> "An attacker could simply make a copy of the users ATM card, keeping the victims accountID but replacing the victims pin with a new pin of the attackers choosing." For example, the attacker has an account at that bank, with their own pin and atm card. They simply paste in their own pin encrypted from their atm card to the bogus card.

To avoid this, store  $[E(K_B, (acctid : pin))]$  on card. In this case, even if the attacker is able to copy the victim's ATM card, they will not be able to separate the pin from the accountID and so cannot carry out the attack.

### Q 8

**How should DES-CBC be used as a keyed one-way hash scheme? Why should DES-ECB not be used?**

A keyed hash function  $hK(M)$  provides a hash-based implementation of a message authentication code.

Message Authentication Code (MAC) is the last ciphertext block returned when encrypting (CBC) a message  $M$  under a secret key.

For example, Alice and Bob share a secret key  $K_{AB}$ . Alice wants to ensure integrity and sends message  $M$  to Bob

Alice -> Bob :  $M, hK_{AB}(M)$

Bob recalculates  $hK_{AB}(M)$  and checks it against hash provided.

The standard HMAC provides keyed hash calculations for MD5, SHA, etc.

Given hash function  $h(M)$ , it is approximately implemented as

$$hK(M) = h(K \hat{h}(K \hat{M}))$$

Which is not unreasonable given our requirement of Unpredictability. The hash value  $h(x)$  does not give any information about any part of its operand  $x$ .

Should we wish to achieve secrecy, we would make a second cryptographic pass of the message before sending it to Bob.

**ECB Open to:**

- A brute force dictionary attack (assuming users pick easy password such as only lowercase letters)
- A pre-computation dictionary attack. Main cost is storage of table. Cheap cost to lookup table.
- Integrity attack. ECB is open to cut and paste attacks.
- Patterns in the plain text show through in the ciphertext.



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

In general ECB mode encryption should not be used.

### Q 9

A vendor uses Message Authentication Codes (MACs) as signatures for orders from customers. A customer generating purchase order P O emails message (P O,  $hk(P O)$ ) to the vendor, where  $hk$  is a keyed one way-hash function and secret key  $k$  is known only to the vendor and the customer. To verify the signature the vendor simply re-computes the hash of the purchase order and compares it with the MAC provided. What is wrong with this digital signature scheme?

Message Authentication Code (MAC) is the last ciphertext block returned when encrypting (CBC) a message  $M$  under a secret key.

$A \rightarrow B : M, MAC$

Receiver can check integrity of plaintext  $M$  by recomputing MAC and comparing it with received MAC.

In general, a MAC is a cryptographic checksum that allows one to check the integrity of a message but does not provide secrecy. In the case above, a customer can take a PO generated by a different customer and email it to the vendor, and get those products ordered to themselves (unless the vendor checks for PO's that have been processed already).

We need a second cryptographic pass through the message to provide secrecy.

For example, Alice and Bob share secrets  $K_{sAB}$  and  $K_{iAB}$ . Alice computes a MAC of the message using key  $K_{iAB}$  and encrypts the message plus MAC:

Alice  $\rightarrow$  Bob :  $E(K_{sAB}, [M, MAC])$

A hash function provides a more effective way of achieving secrecy and integrity.

(Vlad's Solution)

The only answer that comes into my mind is this: the products that are attached to the PO can be modified without affecting the current process, so the integrity check is somehow useless anyway?

Alternatively, I can see that a customer can take a PO generated by a different customer and email it to the vendor, and get those products ordered to themselves (unless the vendor checks for PO's that have been processed already).

### Q 10

A programmer modifies the Unix login program to support 12 character passwords. The hash of the password is computed as follows. The password is first padded to 12 characters with blank spaces; the first 6 characters of the padded password provide key  $K_1$  and the second 6 characters provide key  $K_2$ . If  $nulls$  is an eight byte block of null values, then DES encryption  $[E(K_1, nulls),$



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**E(K2, nulls)] gives the corresponding hash value of the password and is stored in the normal way in /etc/passwd.**

**Discuss any vulnerabilities that this implementation may have.**

The two 6-char ‘halves’ of the password are encrypted independently. Complexity of two halves same as complexity of one half, thus a brute force takes half the time. Easy to recognize passwords less than 6 characters (second half of key is all nulls).

This implementation also used DES mode encryption. Note that DES uses a 56-bit key which is moderately weak and given resources it is feasible for an attacker to carry out a brute force attack on the 256 bit key space.

However, if we limit ourselves to a 6-character password then in practice it is likely that the effective key space is smaller (meaning less work for the attacker) assuming that the user limits their passwords to the more usual (printable) ASCII characters

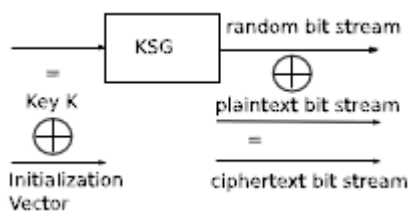
**Q 11**

**A secure web server uses the a standard C library random number generator rand(), seeded with a passphrase, as a stream cipher in order to provide simple group-based web-page security. Each group of users share a common passphrase k which is used to create and view shared web-pages, encrypted as  $C = P \oplus \text{rand}(k)$ . Comment on the effectiveness of this mechanism and discuss how a stream cipher might be properly used in this case.**

If an attacker was to gain access to two encrypted web pages(both used by users of the same group), they would be able to spot the differences in the two encrypted web pages and reverse engineer the encrypted web pages to get the web pages themselves. ie.  $P1 \text{ XOR } P2 = C1 \text{ XOR } C2$ . The attacker would simply XOR the two ciphertexts to get the two plaintexts.

*“Suppose Alice saves her document P1 (plaintext), protected using key K. Ciphertext  $C1 = P1 \oplus RC4(K)$  is saved to disk. Suppose that Alice edits her document, creating P2, which is saved as  $C2 = P2 \oplus RC4(K)$ , using the same key. Its easy to spot differences between the encrypted documents C1 and C2.”*

The same passphrase should not be used more than once to provide access. **This can be done by adding a different initialization vector to the key, each time it is used.**





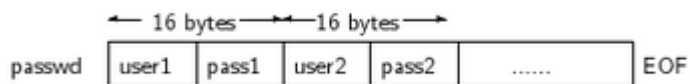
Title : CS4614 Study  
Student Name : Brian O Regan  
Student Number : 110707163  
Module : CS4614  
Exam Date: Wednesday 17<sup>th</sup> December 2014 @ 16.30

For example, given plaintext P and secret key K then a random initialization vector IV is generated and  $C = RC4(K \oplus IV, P)$ .

Note that the IV must also be sent/stored with C so that the recipient, knowing K, can generate the key-stream for decryption.

### Q 12

A server maintains details of users and their passwords in a file composed of sequences of 16-byte records, each one giving an 8 character user-id and 8 character password. The file is encrypted under key K using DES-ECB.



Client systems (who share K) maintain copies of passwd for local user authentication. The server periodically broadcasts copies of passwd to clients over a public network.

Msg1 : Server → Client : {passwd}K

(a) Describe how a client should use K to authenticate the server.

Msg1 : Client → Server: Challenge

Msg2 : Server → Client : {Challenge}K

Client sends challenge to server. Server returns challenge encrypted under key k. Client computes challenge encrypted under key k. If both challenges encrypted under key k match, the client can trust the server and thus authenticates the server.

(b) Outline an attack on the password file that would enable a user to gain unauthorized access to somebody else's login account on a client.

The password file is encrypted using DES-ECB, making it insecure. ECB encryption mode is open to both brute-force and precomputation attacks. Apart from that DES uses a 56-bit key space which is relatively small and makes it feasible for an attacker to brute-force this key space. Once the attacker has found the appropriate key, they can simply decrypt the passwd file, and it will be relatively easy to discover that each user's details (by analysing the patterns in the plaintext) is stored in 16 byte blocks, where the first half is their userid and the second half is their password.

(c) What advice about designing security protocols would you give to the designer of this system?

I would advise that a different type of encryption is used. For example, an AES-CBC encryption is a much stronger way to encrypt the passwd file in the sense that it uses a much larger key space ( $2^{128}$  upwards) and CBC also has many advantages over ECB.



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**Q 13**

**A secure USB memory stick uses onboard AES hardware to encrypt data files to be stored. The AES key is computed as an MD5 hash of a password provided by the user. The device requires the user to change their password at least once every three months and, in order to prevent the user re-using old passwords, the device stores a list of the hash of every previous password. A proactive password checker also requires that passwords be comprised of at least 4 uppercase alphabetic and 3 numeric characters. Discuss any vulnerabilities that this implementation may have.**

This implementation of AES uses MD5 encryption to has the user password users current password. However MD5 is not a good encryption to use as the users password can be easily gained using a brute force attack on the MD5 hash of the users password.

All previous passwords are stored on the device, again as hash values using MD5 encryption. An attacker could again brute force the users old list of password and perhaps gain passwords that the user uses elsewhere.

The implementation above also uses requires at least 4 uppercase alphabetic and 3 numeric characters for passwords. This vastly reduces the possible keyspace that an attacker has to search through in order to find a matching password, using either a brute force attack or a brute force dictionary attack.

**Q 14**

**One-time password key-fobs are issued to all employees for access to the company systems. Each key-fob generates fresh time-based passcodes at 30 second intervals, is tamper-resistant and stores a master secret key K (known only to the authentication server). A key-fob calculates the passcode as  $\{time\}K$  ,  $\{userid\}K$  ), for its owner userid. Outline an attack on this scheme that would allow an attacker gain access to another user account (without having to steal the victim's key-fob).**

Replay attack: The attacker could listen in on the network that is being used between the employer and the company system. Once the attacker overhears/oversees a message being transmitted, they make a copy of it. Then(within the 30 seconds) the attacker computes the secret key k(assume the attacker previously found the employees userid elsewhere) by decrypting the employees userid. The attacker then encrypts another users id with key k, inserts it into the key-fob and relays the message to the company system. The attacker now has access to another users account.

(Richards Solution)

If the attacker screens the network traffic for employees logging into systems he'll soon be able to see the makeup of the pin due to similarities in pins of users.

If user1 and user2 log in at the same time their pins may be:

1238283



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

1239084

This means that the first 3 digits of the pin could be the (and most likely are) the timestamp and the last 4 digits are the hashed userid. He can then build up a table of hashed user IDs. When he wants to authenticate as a user he just waits for another pin to be used, splits the hashed user ID and the hashed timestamp and adds another hashed user id to the end and authenticates himself using this.

**Q 15**

**Explain the properties of a one-way cryptographic hash function. When using a one-way hash function to protect passwords, describe how the use of salt can make dictionary / pre-computation attacks difficult.**

A hash function  $h$  maps arbitrary length value  $x$  to fixed length value  $y$  such that:

- Hard to reverse. Given value  $y$  not feasible to find  $x$  with  $y = h(x)$ .
- Collision freeness. Hard to find values  $x, x'$  such that  $h(x) = h(x')$ .
- Unpredictability. The hash value  $h(x)$  does not give any information about any part of its operand  $x$ .

A salt, when hashed with passwords, makes it impractical to build a dictionary table.

When a password is chosen by the user, a random salt value  $s$  is generated and hashed with the password. The password file stores

$$s : h(s^{\wedge}\text{password})$$

where  $\wedge$  denotes concatenation. If the salt is large then building the dictionary table becomes costly.

word	salt	$h(\text{salt}^{\wedge}\text{word})$
aardvark	0	\$1\$29b43ef4c7e4b84ff9f25ea158f46818
aardvark	1	\$1\$263818db1dc48169633a51e04fa0bf98
...	...	...
boy	0	\$1\$fc0f90e9b32b460b569c6d27291bc3ba
boy	1	\$1\$ef90e3e32b460b569c6d2723234aeba
...	...	...

**Q 16**

**The Department of Computer Science is evaluating a Biometric identification system for tracking lecture attendance by its 1,000 students. One vendor proposes a system that uses a Biometric device that achieves 99.9% correct rejection performance (that is, one false accept in 1,000 trials). Advise the department on this system.**

- Hard to achieve high accuracy when using a Biometric identification system alone as a means of identification.
- Biometrics can be useful to help authenticate a claimed identity.
- Avoid using a biometric system to identify a person.

Calculate chances? Not sure of formula, yet!





Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

There is always a chance that a student may be identified as another student. More than one form of biometric identification should be used to increase the authentication accuracy e.g. thumbprints and several fingerprints (all used to identify student).

## Exercise Sheet 2 – Answers

**Q 1.**

**A programmer reads the S/KEY one-time-password scheme and (incorrectly) implements it as:**

- For a given  $i$ ,  $h_i(s)$  denotes  $h(h(\dots h(s)))$ , representing  $i$  applications of one-way hash function  $h$  to value  $s$ , where  $s$  represents an initial password (seed) chosen by the user.
- When a user picks her (initial) password  $s$ , the system stores  $(1, h(s))$  in the password file.
- If a user has logged-in  $i - 1$  times since choosing initial password  $s$ , then the system stores  $(i, h_i(s))$  in the password file.
- When a user attempts to log-in for the  $i$ th time the system presents  $i$  as a challenge. The user (knowing  $s$ ) provides response  $r = h_{i-1}(s)$ . The system compares  $h(r)$  with the hash value stored in the password file and, if equal, updates this password entry to  $(i + 1, h(h(r)))$ .

**Outline an attack on this scheme and describe how it should be fixed.**

This scheme is open to a man in the middle attack where the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker.

A man in the middle attack is possible where we assume an attacker manages to get hold of a password that was used for a successful authentication. The attacker's goal would be to find out password  $i-1$ , because this password is the one that will be used for the next authentication.

I would suggest using a keyed one way hash function. So if the attacker "piggybacks" on the connection, they will not be able to use any information transmitted (between the client and the server) to find out how to authenticate itself for future connections.

This vulnerability can also be avoided using ssh/SSL connections.

---

**Q 2.**



Title : CS4614 Study  
Student Name : Brian O Regan  
Student Number : 110707163  
Module : CS4614  
Exam Date: Wednesday 17<sup>th</sup> December 2014 @ 16.30

**A bank provides one-time password key-fobs to customers who wish to do their banking over the Internet. Each customer is given a unique key-fob which generates fresh time-based passcodes at 30 second intervals.**

**Each key-fob is tamper-resistant and stores a master secret key  $K$  (known only to the bank) and its owner's 8 byte userid. A key-fob calculates the passcode as  $(\{userid, time\}K)$  using DES-ECB encryption. When a customer attempts to login, she provides (userid, passcode); the remote bank system decrypts the pass-code fields, matches the userid and checks that the time is current.**

**(a) Outline an attack on this scheme that would allow an eavesdropper gain access to a another customer's account (without having to steal the victim's key-fob).**

Cut and paste attack - attacker simply cuts out usrid from above, pastes in different userid. (ECB)

A key fob is a type of security token: a small hardware device with built-in authentication mechanisms.

An attacker listening in on the connection could make a copy of a login from a customer. The attacker could then perform a brute force attack. As DES uses a 56 bit key it is moderately weak and is feasible for an attacker to scan the entire 256 bit key space. While carrying out the brute force attack, once a result is found in which there exists the customers userid(the customer he originally copied the login from) he can save the key knowing it is the master key.

Once another customer attempts to connect, the attacker can quickly decrypt the login sent by the customer over the network, make a copy of the time-based passcode and then encrypt his own login protocol with another customer's userid and the time-based passcode using the key  $k$  to gain access to another customer's account.

**(b) Suggest a fix to the scheme above and argue why you think your proposal is resistant to attack.**

USE CBC

Mutual authentication between the customer and the bank.

Msg1  $A \rightarrow B$  : I'm Alice,  $R_2$   
Msg2  $B \rightarrow A$  :  $R_1, \{R_2\}_{KAB}$   
Msg3  $A \rightarrow B$  :  $\{R_1\}_{KAB}$

Here both parties will have authenticated each other and so can ensure that they are speaking with whom they originally intended.

I would suggest using an AES CBC when encrypting the messages between Alice and Bob, where the AES cipher uses a 256 bit key. This would be a much better protocol as it is not feasible for an attacker to brute force a key space of 2256.

---

**Q 3.**



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**A programmer wants to use DES Cipher Block Chaining to support both integrity and confidentiality. He implements the following scheme. He appends a block of nulls at the end of the plaintext message prior to encryption. If the block of nulls is not present after decryption then message has been corrupted. Outline an attack on this scheme, whereby an attacker can corrupt the ciphertext blocks without being detected. Describe how message integrity and confidentiality should be implemented.**

An attacker may corrupt the initial ciphertext blocks. In this case, the ciphertext blocks will “recover” over time. As a result the block of nulls will still be present at the end of the plaintext message even after the beginning of the plaintext message was corrupted.

Message Authentication Code (MAC) is the last ciphertext block returned when encrypting (CBC) a message M under a secret key.

$$A \rightarrow B : M, MAC$$

Receiver Bob can check integrity of plaintext M by recomputing MAC and comparing it with received MAC.

In general, a MAC is a cryptographic checksum that allows one to check the integrity of a message but does not provide secrecy. We need a second cryptographic pass through the message to provide secrecy.

For example, Alice and Bob share secrets  $K_{AB}^s$  and  $K_{AB}^i$ . Alice computes a MAC of the message using key  $K_{AB}^i$  and encrypts the message plus MAC:

$$Alice \rightarrow Bob : E(K_{AB}^s, [M, MAC])$$

A hash function provides a more effective way of achieving secrecy and integrity.

---

#### Q 4.

**At one point, Microsoft’s WindowsCE operating system stores user passwords xored with the string sausageP . Outline how an attacker could determine a user’s password and suggest a better way to store passwords.**

For an attacker it would be relatively easy to find the users password. The password is simply xored with the word “sausageP”, and once the attacker knows this piece of information, they could simply XOR sausageP with the users encrypted password, to get the plaintext password.

(Richard’s Solution)

If a malicious user gains access to the system they can find the passwords stored by the OS and XOR them with the string 'sausageP' meaning they now have the plain text passwords for all the users of the system.

There are a vast number of ways this attack can be mitigated, for example, a much better, more secure method of storing user passwords would be through the use of salts. ie. When a password



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

is chosen by the user, a random salt value  $s$  is generated and hashed with the password. The password file stores

$$s : h(s^{\wedge}\text{password})$$

where  $\wedge$  denotes concatenation.

If the salt is large then building the dictionary table becomes costly. Other methods would involve using ECD or CBC encryption etc. Also do not use the same word ie. 'sausageP' when encrypting the password as it is easy for the attacker to reverse.

---

**Q 5.**

The following protocol is used to authenticate a client C to a server S. Both principles share secret pass, R is a random challenge, and  $h()$  is a one-way hash function.

$$\text{Msg1} : S \rightarrow C : R$$
$$\text{Msg2} : C \rightarrow S : h(R, \text{pass})$$

The following Java code fragment from the server-side of this protocol reflects a number of (poor) implementation decisions. You may assume that the client-side uses similar implementation decisions.

```
MessageDigest md = MessageDigest.getInstance("MD5");
DataOutputStream out = ... // stream to connecting client
DataInputStream in = ... // stream from connecting client
byte[] passwd = ... // shared password
```

```
Random random = new Random(0); //java.util.Random generator
byte[] R = new byte[1]; //-random seed used is 0
random.nextBytes(R); // generate 1 byte random value
out.write(R); // send to client
```

```
byte[] hashR = new byte[16];
in.readFully(hashR);
byte[] hashpass = new byte[16];
in.readFully(hashpass);
```

```
if
(MessageDigest.isEqual(hashR,md.digest(R))
MessageDigest.isEqual(hashpass,md.digest(pass)))
... // client authenticated
```

**Identify and explain the security vulnerabilities in this implementation.**

1. The random number is sent in plaintext from the server to the client. An eavesdropping attacker can easily make a copy of this value and store for future (very near) attacks.



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

2. An MD5 MessageDigest is partially implemented. MD5 is relatively easy to brute force by attackers, and in the case that the client (or the server before sending the random value) had actually used the message digest, the attacker could also carry out a pre-computational dictionary attack.
3. The server does not hash any values before sending and because we “may assume that the client-side uses similar implementation decisions”, I would say that the client does hash their random number or their pass (even tho the variables that the client takes in are called hashR & hashpass!)
4. Msg2 : C -> S : h(R, pass) <- Here the client (apparently) sends the hash of the random value and a password. While this isn't a terrible protocol, it can be improved.
5. The server (and client) does not make use of symmetric block ciphers.
6. The connection is not closed. If the connection is left open for prolonged amounts of time, an attacker may have an opportunity to perform an attack.
7. No secret key with mac(message authentication code) used.

There's at least two more so if anyone spots them please let me know! 2-3 from the above would be plenty.

#### Outline how the code should be repaired.

1. Hash the random number with a key known only by the client and the server, before sending it to the client.
2. I would suggest initializing a SHA1 message digest, and then incorporating that message digest in the code. For example (client file),

```
MessageDigest md= MessageDigest.getInstance("SHA1");  
byte[] rslt = md.digest("password, random value"); //generate hash value  
out.write( rslt ); // send to server
```

3. Hash all pieces of information before it is sent to make it more difficult for attacker to perform an attack.
4. A timestamp could also be hashed with the random value and the pass. To maintain an appropriate level of security, this timestamp should be updated every 30 seconds (client side). When the server receives h(R, pass, timestamp), it can check if the timestamp is within the last 30 seconds, and if so authenticate the user.
5. Ciphers can be easily implemented in the code. eg.

```
Cipher cipher = Cipher.getInstance ("DES/ECB/PKCS5Padding");  
cipher.init (Cipher.ENCRYPT_MODE, key); (assuming we have a DES key)  
byte[] cBytes = cipher.doFinal(data); // data to be encrypted
```

This data can then be sent, and decrypted as follows

```
....  
cipher . init (Cipher .DECRYPT MODE, key);  
byte[] pBytes= cipher.doFinal(cBytes);
```

6. Simply close the connection.. out.close(); & in.close();
7. Implement the mofucka



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**Q 6.**

A programmer develops a 64-bit key extension of DES. A 64-bit key  $K_x$  is split into two 32-bit parts and each part is padded out with nulls to 56 bits to give (standard DES) keys  $K_1$  and  $K_2$ . A (64-bit) block of plaintext  $P$  is encrypted by  $K_x$  to produce ciphertext  $C$  as:

$$C = E(K_1, E(K_2, P))$$

$E(k, B)$  represents the DES encryption of a 64-bit block  $B$  using (56-bit) key  $k$ .

Describe an attack on this cipher.

Note symmetry:  $E(k_2, P) = X = D(k_1, C)$

A meet in the middle attack is applicable to any double encryption cipher.

Known plaintext attack given  $P$  and  $C$ :

1. Encrypt  $P$  for all 256 values of  $k_2$  and store in a table indexed by  $X = E(k_2, P)$ .
2. Decrypt  $C$  for all 256 values of  $k_1$ ; as decryption is calculated, check result against table for a match; if match occurs, then test resulting pairs against the  $P, C$  pair.

---

**Q 7.**

A document editor provides an option to store documents in encrypted form based on a user provided passphrase.

(a) Suppose that the standard C library pseudo-random number generator  $\text{rand}()$  is used as a stream cipher to encrypt a document  $P$  as  $C = P \oplus \text{rand}(k)$ , where the seed  $k$  is a one-way hash of the passphrase. Comment on the effectiveness of this mechanism and discuss how a stream cipher might be properly used.

Suppose a document  $P_1$  (plaintext) is saved, protected using key  $K$ .

Ciphertext  $C_1 = P_1 \oplus \text{rand}(k)$  is saved to disk.

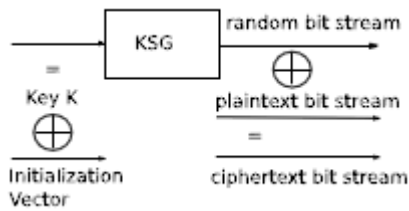
Suppose the document is edited and saved again, creating  $P_2$ , which is saved as

$C_2 = P_2 \oplus \text{rand}(k)$ , using the same key.

Its easy to spot differences between the encrypted documents  $C_1$  and  $C_2$ , which in turn will allow an attacker to find the plaintext of both files.

$$C_1 \oplus C_2 = P_1 \oplus P_2$$

The same keystream should not be used to encrypt more than one message. This can be done by adding a different initialization vector to the key, each time it is used.



For example, given plaintext  $P$  and secret key  $K$  then a random initialization vector  $IV$  is generated and  $C = \text{rand}(k \oplus IV, P)$ .

Note that the  $IV$  must also be sent/stored with  $C$  so that the recipient, knowing  $k$ , can generate the key-stream for decryption.

**(b) A programmer modifies the document editor and uses DES in ECB mode for encryption. For added security, each block of plaintext is encrypted twice using an eight character user password as the key. Prior to encryption, a block of null values is appended to the end of the plaintext document. When the document is loaded/decrypted, the block is used to confirm the integrity of the document. Comment on the effectiveness of this design and suggest how it might be improved.**

Using nulls as a recognizer at the end of a message does not act as a check for integrity.

DES-ECB is not a suitable cipher mode for encrypting a stream of plaintext blocks since it encrypts each block independently. As a consequence

- patterns in the plaintext show through in the ciphertext
- it is subject to a cut and paste attack whereby an attacker can replace any ciphertext block by the contents of another ciphertext block without interfering with the content of the surrounding blocks.

Note that DES uses a 56-bit key which is moderately weak and given resources it is feasible for an attacker to carry out a brute force attack on the 256 bit key space.

However, if we limit ourselves to an 8-character password then in practice it is likely that the effective key space is smaller (meaning less work for the attacker) assuming that the user limits their passwords to the more usual (printable) ASCII characters

This design may be improved by using AES CBC mode to encrypt the document. CBC does not encrypt each block independently. Each block is encrypted with the ciphertext of the previous block, completely eliminating any patterns that may have been shown otherwise. The key space is also increased to 128 bits.

**Q 8.**

**A server maintains details of users and their passwords in a file that is composed of a sequence of 16-byte records. Each record contains an eight byte user-id and an eight byte password, and is**



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

sorted by user-id. Client systems maintain their own copy of this file for local user authentication. Clients periodically obtain an encrypted copy of the password file across a public network. The file is encrypted with a secret key (known to server and all clients) using the following Java code.

```
Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE,key);
FileInputStream fin = new FileInputStream(passwdplain);
FileOutputStream fout = new FileOutputStream(passwdenc);
CipherOutputStream out = new CipherOutputStream(fout, cipher);
```

```
byte[] buffin new byte[1024]; int length;
while ((length = fin.read(buffin))!=-1)
out.write(buffer,0,length);
fin.close; fout.close;
```

(a) Describe how a passphrase could be used to generate the key key. What is a dictionary attack on a passphrase? What defences should be used to make it harder to carry out a dictionary attack on pass phrases? Explain your answer.

```
byte[] keyBytes = "passphrase";
SecretKey key = new SecretKeySpec(keyBytes, "HmacSHA1");
Mac mac = Mac.getInstance("HmacSHA1"); // generate MAC
mac.init (key); // initialize MAC with this key
```

A dictionary attack occurs where an attacker has a dictionary of words (may also be phrases, sayings, mixtures etc. -> whatever grants the attacker a better chance of succeeding in attack. Keep in mind more words = more time to compute) which he thinks the victim may be using for a secret passphrase. The attacker computes the hash of each word in their dictionary and compares with the hash of the passphrase. If a match is found the attacker will know that the current word in their dictionary is the one the victim is using as their passphrase.

Strategy: make it impractical to build a dictionary table. Salts should be used.

When a password is chosen by the user, a random salt value  $s$  is generated and hashed with the password. The password file stores

$$s : h(s^{\wedge}password)$$

where  $\wedge$  denotes concatenation.

If the salt is large then building the dictionary table becomes costly.

(b) Provide Java code that a client system could use to extract a plaintext copy of the encrypted password file. Explain its operation.

```
// source file & pass taken in
FileInputStream file = new FileInputStream( srcFile );
FileInputStream pass = new FileInputStream( pass ); // bad security - just eg,

// generate message digest
MessageDigest md = MessageDigest.getInstance("DES/CBC/PKCS5Padding");
```





Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

```
// generate hash value
byte[] keyBytes = md.digest(pass);

// turn hash value into an HmacSHA1 key
SecretKey key = new SecretKeySpec(keyBytes, "HmacSHA1");

// cipher generated
Cipher cipher = Cipher.getInstance("DES/CBC/PKCS5Padding");

// decrypt operation
cipher.init (Cipher.DECRYPT_MODE, key);

// srcFile decrypted with key under DES/CBC/PKCS5Padding mode
byte[] decryptedFile = cipher.doFinal(file);

// FileInputStream closed
file.close();
pass.close();
```

**(c) Describe an attack on the above scheme whereby a user of a client system, who controls the network, can log in as another user. How can this attack be avoided?**

Clients are sent a copy of the password file, which contains all userids and passwords and each record contains an eight byte user-id and an eight byte password. While this password is encrypted, all clients know the key and so can decrypt the file. There is no mention of the clients passwords being under any sort of encryption. A client can log in as another user simply by using another users id and password from the password file once it has been decrypted.

This can be avoided by encrypting each clients userid and password separately in the password file. For this to work the server would need to share a secret key with each client, and each client would need to share a secret key with the server. This would ensure that any client who views the password file will not be able to simply read the data contained.

### **Exercise Sheet 3 – Answers**

#### **Important Attacks & Protocols**

- Mutual Auth - Reflection attack - 7 : 7
- Wide Mouth Frog Protocol - 7 : 13
- Long / short term // session keys - 7 : 15
- Wide Mouth Frog Replay - 7 : 16
- Wide Mouth Frog: Key Revocation - 7 : 17
- Wide Mouth Frog Replay 2 - 7 : 19
- Needham-Schroeder Protocol - 7 : 22
- Kerberos - 7 : 25
- Diffie Hellman Key Exchange - 9 : 7



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

- Public Key Cryptography - 9 : 11
- Diffie Hellman Mutual Auth - 9 : 30
- Null prefix certificates and X509 Implementation - 10 : 23
- Secure Email using Pretty Good Privacy - 10 : 27

### Notes (From copy too)

- Simon just wants general structure of protocols. It is ok if you miss some parameters in the protocols.
- Always clearly identify runs of your protocol to make sure you are correct and to reduce mistakes. Also makes it easier to understand, and for Simon to award marks.
- Always clearly state the assumption being made. (mostly related to protocols)
- Diffie Hellman Key Exchange
  - Cannot be used for authentication (and because there are no authentication procedures put in place), because of problem with Eve in the middle. Instead used for exchanging public keys.
  - One problem with DH, Alice does not know who she is making key exchange with.
  - For the attacker to get X, they will need to know y, in order to compute  $X^y \text{ mod } n = K_{AB}$ . Maybe get y from Y i.e. compute the discrete log of Y, however that is a hard / unfeasible problem to solve.
  - Big X public, little x private.
  - $g^y \Rightarrow$  only Bob knows little y
- Diffie Hellman Key Exchange open to man in the middle attack. A person in the middle may establish two distinct Diffie–Hellman key exchanges, one with Alice and the other with Bob, effectively masquerading as Alice to Bob, and vice versa, allowing the attacker to decrypt (and read or store) then re-encrypt the messages passed between them. A method to authenticate the communicating parties to each other is generally needed to prevent this type of attack.
  - For this attack Eve would create her own secret  $Z = g^z \text{ mod } n$

Basic DH Protocol (in any order):

Msg1 A  $\rightarrow$  B :  $g^x \text{ mod } n$

Msg2 B  $\rightarrow$  A :  $g^y \text{ mod } n$

Neither party really knows with whom it shares secret  $k = g^{xy} \text{ mod } n$ .

Eve routes messages between Alice and Bob

alpha : Msg1 : A - B[E] :  $g^x \text{ mod } n$

alpha : Msg2 : B[E] - A :  $g^z \text{ mod } n$

beta : Msg1 : A[E] - B :  $g^z \text{ mod } n$

beta : Msg2 : B - A[E] :  $g^y \text{ mod } n$

Alice uses  $k_a = g^{xz} \text{ mod } n$  to 'speak' with Bob, Bob uses  $k_b = g^{zy} \text{ mod } n$  to 'speak' with Alice, and Eve knows both keys, while carrying out a man-in-the-middle attack.



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

- The Needham–Schroeder Symmetric Key Protocol is based on a symmetric encryption algorithm. It forms the basis for the Kerberos protocol.
- Man in the middle attack also known as a bucket brigade attack. Macs are not signatures, they do not tie a msg to a specific person / principle. They can be used by anyone who knows the key. MAC provides integrity check, not authentication check.
- Streisand effect (key subsequently published in a number of other places) - 9 : 26
- Digital Signature Scheme: A third party can resolve disputes about the validity of a digital signature without having to know the signer's key. Digital signature schemes should ideally support non-repudiation.
- $\{ TA, KAB, A \} KA-1 \Rightarrow$  Authentication
- $\{ \{ TA, KAB, A \} KA-1 \} KB \Rightarrow$  Secrecy
- KB decrypted with KB-1  $\Rightarrow$  only Bob knows KB-1, so only Bob can decrypt a message encrypted with KB
- Everyone can encrypt message etc. with public key KA, but only A can decrypt this encrypted information with KA-1
- ^ ^ In above, Eve can masquerade as Alice (replay attack)
  - Should include intended recipient in message;
  - $\{ \{ TA, KAB, B, A \} KA-1 \} KB$
- A  $\rightarrow$  B : { Message }SKA-1
  - s = signed by
  - Anyone can decrypt (as all know KA)
  - If not interfered with, must be from A
- A  $\rightarrow$  B : RSA( KA-1, ( M, h( M ) ) )
  - Better than above protocol
  - Bob decrypts with KA  $\Rightarrow$  gives ( M, h( M ) )
  - Then recomputes h( M ) and checks against the message value provided  $\Rightarrow$  integrity check
- A  $\rightarrow$  B : M, RSA( KA-1, h( M ) )
  - KA-1, h( M )  $\Rightarrow$  encrypt hash of message using private key
  - RSA( KA-1, h( M ) )  $\Rightarrow$  digital signature
  - Assuming that Bob knows KA is owned by Alice, then Bob can check the signature by decrypting using KA and checking that the crypto checksum == the checksum of the message provided in plaintext
- KB (public key) decrypted with KB-1 (private key)
- A  $\rightarrow$  B : { grade = 95, hg }KAB, {KAB, A, B,...}KSA } KB



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

- { grade = 95, hg }KAB => judge does not know who generated this as both A and B know KAB
  - {KAB, A, B,...}KSA } KB => encrypted under A, so must be A that generated it
  - Proper digital signature : judge does not need to know any secrets to confirm validity of message.
- If one principle is given a challenge, the response challenge must be different to avoid reflection / replay attacks. eg. Nonce Na & Na + 1
  - To make protocol stronger: should use timestamps, nonces, add extra info to protocol;
    - Extra info eg.
      - Bad = { KAB }
      - Good = { KAB, A, B, Na }

### Q 1

Alice (A) wishes to communicate securely with Bob (B) and proposes a symmetric key KAB , a copy of which she intends to give to Bob. Trent is a trusted third party who shares secret (symmetric) key KAT with Alice and secret (symmetric) key KBT with Bob. The following protocol is used to pass the key KAB to Bob.

Msg 1:  $A \rightarrow T : (\{B\}_{K_{AT}}, \{K_{AB}\}_{K_{AT}})$

Msg 2:  $T \rightarrow B : (\{A\}_{K_{BT}}, \{K_{AB}\}_{K_{BT}})$

(a) Discuss any disadvantages in the operation of the above protocol. In particular, compare it with the operation of a Kerberos/Needham Schroeder style protocol.

- The above protocol has disadvantages over Needham-Schroeder does not make use of a nonce or a timestamp -> open to a replay attack / reflection attack.
- This protocol relies on Alice to be competent at generating reliable keys. If Alice uses an ordinary random number generator she is being unreliable. In the improved protocol below Trent is reliable.
- The protocol is stateful. This is usually undesired because it requires more functionality and capability from the server. For example, T must be able to deal with situations in which B is unavailable. In Needham-Schroeder, Alice can get a key and a ticket only if Bob is online (ie. if Trent was able to reach Bob). If Bob is unreachable then Alice decides what to do, not Trent.

Msg1 :  $A \rightarrow T : \{ A, B, Na, KAB \}_{K_{AT}}$

Msg2 :  $T \rightarrow A : \{ A, B, Na + 1, KAB, \{ A, B, KAB \}_{K_{BT}} \}_{K_{AT}}$

Msg3 :  $A \rightarrow B : \{ A, B, KAB \}_{K_{BT}} \quad (\text{ticket})$

^ ie. Alice gets a key & a ticket from Trent, and then sends them to Bob.



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**(b) Suppose that principle B above is a Ticket Granting Server that controls access to, and shares a secret key with, the file server C. Propose and explain a protocol exchange between A and B that might result in A obtaining a ticket enabling it to authenticate/connect securely with C. Discuss any problems that your protocol design might have.**

**Question outline: Do you understand how Kerberos works?**

A sends to Bob who she wants to communicate with and the key she wants to use. Bob sends key to proposed party( C ). A & C can now communicate.

- Trent is trusted third party (Kerberos : Auth Service)
- Bob is the Ticket Grant Service (TGS)

Initially A authenticates with T in order to get access to TGS B.

Msg1 : A -> T : { B }KAT, { KAB }KAT  
Msg2 : T -> B : { A }KBT, { KAB }KBT

Then, Alice can rerun this protocol to connect any service, for example C.

Msg3 : A -> B : { C }KAB, { KAC }KAB  
Msg4 : B -> C : { A }KBC, { KAC }KBC

Last two steps can be repeated for different services.

Problems this protocol has:

- Does not use timestamps or nonces
- Same problems as part (a), ie. bottleneck problem - A has to go through B to connect to C.

**(c) Illustrate how third user, Eve (who shares a secret key KET with Trent) can subvert the protocol and get a copy of a key KAB that Alice gives to Bob using this protocol.**

(Tip: Find out what Eve wants and work backwards )

Basically Eve wants to trick T into thinking this KAB was intended for B, while T is actually sending key KAB to Eve. Eve wants:

Msg1 : A[ E ] -> T : { E }KAT, { KAB }KAT  
Msg2 : T -> E : { A }KET, { KAB }KET

At some point in the past A & B runs protocol;

alpha : Msg1 : A -> T : { B }KAT, { KAB }KAT  
alpha : Msg2 : T -> B : { A }KBT, { KAB }KBT

Eve requests to share key KEA with Alice...

beta : Msg1 : E -> T : { A }KET, { KEA }KET  
beta : Msg2 : T -> A : { E }KAT, { KEA }KAT

**(d) Illustrate how Eve can subvert the protocol and masquerade as Alice to Bob, even when Alice does not initiate a key exchange with Bob. (same as q 2b in sample xmas 2011)**



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

Eve generates key KEB that she wants to share with Bob, but makes Bob think she is Alice. Eve wants;

Msg1 : A [ E ] -> T : { B }KAT, { KEB }KAT  
Msg2 : T -> B : { A }KBT, { KEB }KBT

alpha : Msg1 : B -> T : { A }KBT, { KXX }KBT  
alpha : Msg2 : T -> A : { B }KAT, { KXX }KAT

beta : Msg1 : E -> T : { A }KET, { KEB }KET  
beta : Msg2 : T -> A : { E }KAT, { KEB }KAT

However, this attack could not be used if Bob never initiated a connection with A (alpha msgs). A better, easier attack for Eve would be;

alpha: Msg1 : E -> T : { A }KET, { B }KET  
alpha : Msg2 : T -> A : { E }KAT, { B }KAT

beta : Msg1 : E -> T : { A }KET, { KEB }KET  
beta : Msg2 : T -> A : { E }KAT, { KEB }KAT

Here, Eve does all the work, and does not rely on any other principle to (help her)carry out the attack.

Another Attack;

At some point in past..

Msg1 : A -> T : { B }KAT, { KAB }KAT

Msg1 : E -> T : { A }KET, { KEB }KET  
Msg2 : T -> A : { E }KAT, { KEB }KAT

## Q 2

The following mutual authentication protocol has been designed to be resilient against reflection attacks. This is done by ensuring that the challenge from the initiator looks different from the

$$\begin{aligned} \text{Msg1 } A \rightarrow B &: \text{ I'm Alice, } R_2 \\ \text{Msg2 } B \rightarrow A &: R_1, \{\text{Bob, } R_2\}_{K_{AB}} \\ \text{Msg3 } A \rightarrow B &: \{\text{Alice, } R_1\}_{K_{AB}} \end{aligned}$$

challenge from the responder.

Suppose that a programmer implements the above protocol across a unix network, where principal names are eight (8) characters long and triple DES-ECB is used for encryption. Outline a possible (reflection) attack on this protocol.

How far can Eve get (Msg 2);

alpha : Msg1 : A[E] -> B : I'm Alice, Na  
alpha : Msg2 : B -> A[E] : { Na }KAB, Nb



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

Eve needs  $N_b$  to be encrypted using the key  $K_{AB}$  -> Eve thinks to herself, how can I trick either A or B into doing this?

beta: Msg1 : A[E] -> B : I'm Alice,  $N_b$

beta : Msg2 : B -> A[E] : {  $N_b$  } $K_{AB}$

Finally, Eve has what she wants and sends message;

alpha : Msg3 : A[E] -> B : {  $N_b$  } $K_{AB}$

Here, information is encrypted using DES-ECB. Each item is encrypted in blocks. eg. { Alice } $K_{AB}$ , {  $N_a$  } $K_{AB}$ . This makes it easy for different blocks of encrypted data to be 'cut and paste', by an attacker.

### Q 3

A programmer simplifies the Needham Schroeder protocol as follows.

Msg1  $A \rightarrow T : A, B$

Msg2  $T \rightarrow A : \{B, K_{AB}, \{K_{AB}, A\}_{K_{BT}}\}_{K_{AT}}$

Msg3  $A \rightarrow B : \{K_{AB}, A\}_{K_{BT}}, \{N'_A\}_{K_{AB}}$

Msg4  $B \rightarrow A : \{N'_A - 1, N_B\}_{K_{AB}}$

Msg5  $A \rightarrow B : \{N_B - 1\}_{K_{AB}}$

Suppose that Eve manages to steal Bob's key  $K_{BT}$ . Can the protocol be compromised even when Bob and Trent re-key ( use a new, uncompromised key  $K_{BT}'$  )? Explain your answer.

At some point Eve steals key  $K_{BT}$ . Eve keeps a copy of msg2 & msg3 from the above protocol. From msg3, Eve can discover  $K_{AB}$ . Then Bob rekeys to  $K_{BT}'$ .

Suppose A initiates another connection to B, where Eve is pretending to be Trent;

Msg1 : A -> T[E] : A, B

Msg2 : T[E] -> A : { B,  $K_{AB}$  {  $K_{AB}, A$  } $K_{BT}$  } $K_{AT}$

Msg3 : A -> B[E] : {  $K_{AB}, A$  } $K_{BT}$

Alice thinks she is speaking with Bob, but really it is Eve. Next Alice will authenticate Eve as Bob in msgs 4 & 5.

Here, Alice makes a request, but the response belongs to a previous run of the protocol. Could put timestamp in msg3 ( however attacker may make successful attack within time( window of opportunity ) ) but there is a better solution. The use of nonces;

Msg1 : A -> T : A, B,  $N_a$

Msg2 : T -> A : { B,  $K_{AB}$ ,  $N_a$  {  $K_{AB}, A$  } $K_{BT}$  } $K_{AT}$



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

#### Q 4

A programmer (foolishly) decides to simplify the Kerberos protocol specification. The Kerberos server and ticket-granting servers are implemented by a single (trusted key) forwarding service T . The revised protocol is specified as:

Message 1 :  $A \rightarrow T : B, \{K_{AB}\}_{K_{AT}}$

Message 2 :  $T \rightarrow B : A, \{K_{AB}\}_{K_{BT}}$

Assuming Alice shares secret key KAT with T then she uses this protocol to propose a session key KAB to T so that a (temporary) secure channel based on KAB may be established between A and the service B (a file system, for example).

(a) It is the duty of T to authenticate requests from Alice. How might Alice's password be used for login and initial authentication in the protocol above?

Yes it is the duty of T to authenticate requests from Alice.

Msg1 : A - T : A

Msg2 : T - A : Nonce

Msg3 : A - T : { Nonce }KAT

(b) It is the duty of T to mediate access requests, that is, to decide whether Alice may access certain services. How might this be achieved using the protocol above?

Yes it is the duty of T to mediate access requests. When Alice asks to access a certain service, T will check (behind the scenes check outside of the protocol itself) if Alice is allowed to access that particular service. If Alice is allowed to access B for example, the key she proposed (KAB) will be sent to B.

(c) Illustrate how a legitimate user Eve can subvert the protocol and masquerade as another principle.

beta : Msg1 : A[E] - T : E, { KAB }KAT

beta : Msg2 : T - A : E, { KAB }KAT

^^ above Eve tricks T to send her the key KAB.

Basically Eve wants to trick T into thinking she is another user. Eve wants:

alpha : Msg1 : A[E] - T : B, { KAB }KAT (Eve needs the bit in Red(ish))

alpha : Msg2 : T - B : A, { KAB }KBT

Then Eve opens another connection to Alice, via Tent.

beta : Msg1 : E - T : A, { KAB }KET

beta : Msg2 : T - A : E, { KAB }KAT

Now Eve can run the alpha protocol, and pretend to be A while making a connection to B.





Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

( Richard's Answer )

Basically Eve wants to trick T into thinking she is another user. Eve wants:

alpha : Msg1 : A[E] - T : A  
alpha : Msg2 : T - A[E] : Nonce  
alpha : Msg3 : A[E] - T : { Nonce }KAT

Then Even open another connection to Alice, via Trent.

beta : Msg1 : E - T : A, { Nonce }KET (nonce from alpha msg 2)  
beta: Msg2 : T - A : E, { Nonce }KAT

Above, Eve has correctly authenticated herself as 'Alice' to Trent.

---

#### Q 5

**A programmer wants to use DES-CBC to support both integrity and confidentiality. He implements the following scheme. He computes a message authentication code MAC based on the last cipher block generated from encrypting (DES-CBC) plaintext blocks  $b_0, \dots, p_{n-1}$ . He then encrypts the stream of blocks  $b_0, \dots, p_{n-1}$ , MAC using DES-CBC. When decrypting, the MAC block can be used to check for integrity. Outline an attack on this scheme, whereby an attacker can corrupt the ciphertext blocks without being detected.**

In this design, the flaws can be found within CBC (does not provide integrity) :

- Each block is XORed with the ciphertext of the previous block which means that the message will recover over time. This means that blocks further on, in the message will not be affected.
- As the MAC is generated from the last block, an attacker can potentially modify the initial ciphertext blocks and due to CBC's design, later ciphertext blocks will recover to their original value.
- When the MAC is generated this potentially means that the integrity has been compromised without the user being aware of it.

---

#### Q 6

**Why are authentication protocols such as Kerberos and Needham-Schroeder more practical than the wide-mouth frog protocol?**

Wide-mouth frog is a stateful protocol. This is usually undesired because it requires more functionality and capability from the server. For example,  $S$  must be able to deal with situations in which  $B$  is unavailable. Whereas in the Kerberos / Needham-Schroeder protocol,  $A$  would request a connection to  $B$ , and if  $B$  is available Kerberos would reply with a ticket. If no ticket is supplied,  $A$  will know  $B$  is not available and can then decide on an appropriate course an action.

In the Wide Mouth Frog protocol, every message has to go through  $S$ . In Kerberos / Needham Schroeder,  $S$  is used for granting tickets, after which users can then communicate directly without the use of the  $S$ .



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**Q 7 (Ignore question as Simon says(lol) it's too difficult and won't be on end of year exam either!)**

The following protocol is used by principle B to authenticate principle A.

Msg1:  $A \rightarrow B \quad A$

Msg2:  $B \rightarrow A \quad N_B$

Msg3:  $A \rightarrow B \quad \{N_B\}_{K_{AS}}$

Msg4:  $B \rightarrow S \quad \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}}$

Msg5:  $S \rightarrow B \quad \{N_B\}_{K_{BS}}$

Symmetric keys KAS and KBS are shared between principals A and S, and between B and S, respectively. NA and NB represent nonces.

(a) How well designed is this protocol? Explain your answer.

(b) Outline a possible attack on this protocol.

8. Consider the following fragment from a Kerberos-like authentication protocol, whereby initiator  $A$  requests, and is granted, a ticket from Authentication Server  $S$  to be used with service  $B$ .

Msg 1 :  $A \rightarrow S : A, B, N_a$

Msg 2 :  $S \rightarrow A : T_{ab}, \{B, L, N_a, K_{ab}\}_{K_a}$

Principals  $A$  and  $B$  share long-term secret keys  $K_a$  and  $K_b$  with server  $S$ , respectively;  $N_a$  is a nonce;  $\{\dots\}_K$  represents symmetric key encryption with secret key  $K$ . The server issues a ticket  $T_{ab} = \{A, L, K_{ab}\}_{K_b}$  for the session key  $K_{ab}$ , valid for time period specified by  $L$ .

(a) Suppose that  $B$  above is a ticket granting service and, thus,  $T_{ab}$  is a ticket granting ticket. Propose and explain a suitable protocol exchange between  $A$  and  $B$  that will result in  $A$  obtaining a ticket for a file server  $C$ . Your solution should use (time based) authenticators to defend against replay attacks.

**Q 8**

- Answered in End of Term 2012.



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**(b) With reference to your answer in (8a) above, discuss some of the advantages and disadvantages of using authenticators (versus challenge-response) as strategies for avoiding replay attacks.**

Authenticators = passwords, secret keys, etc.

(Richard's Answer)

Reduce the network traffic/time to be authenticated. The challenge response would need 3 packets just to be sure the user is who they say they are. Authenticators can be done in one it seems.

A -> C : I'm A

C -> A : Nonce

A -> C : {Nonce}Kca

Issues with Authenticators would be that it's quite easy to acquire someones password as they're usually quite short, and if they don't notice you can have a dangerous user on the network, A challenge response using public key crypto(or something like that) would reduce the chance of this happening as it's much tougher to pass than it is to get than a password, however a challenge-response may also be intercepted and / or corrupted by an attacker.

ie.

Authenticators may speed up protocols eg. by using certificates, but only if the cert can be trusted. Authenticators do not ask for an authenticated reply ie. no use of nonces.

<http://www.kerberos.org/software/tutorial.html>

[http://docstore.mik.ua/oreilly/networking\\_2ndEd/ssh/ch11\\_04.htm](http://docstore.mik.ua/oreilly/networking_2ndEd/ssh/ch11_04.htm)

**(c) Suppose that T is a secure time service sharing secret key Kt with Authentication Server S. How might server C use T to set its (C's) clock at boot-up time?**

**(send back timestamp)**

**(Simon said we didn't cover this)**

**(d) Suppose that A prefers not to synchronise her clock with T , yet wants to obtain tickets from B. Suggest a strategy that A can use to manage her skewed clock.**

**(Simon said we didn't cover this too)**

## **Exercise Sheet 4 – Answers**

**Notes**



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

- The RSA algorithm is based on the fact that there is no efficient way to factor very large numbers. Deducing an RSA key, therefore, requires an extraordinary amount of computer processing power and time.
- CRL - Certificate Revocation List
  - eg. When certificate from Amazon is received, check against CRL on main certificate authority ( above Amazon in hierarchy ) to see if Amazons cert has expired. If Amazons cert has expired, Alice rejects Amazons cert as it cannot be guaranteed that the key Amazon provides has really come from them.
- Attacks on Cert Signing
  - Suppose Mike (attacker) pretends to be amazon.com by corrupting / attacking Bobs (browser) DNS server or being a man in the middle. When Bob establishes a connection to the bogus amazon.com, he shared a session key  $K_{AB'}$  (prime) with the owner of  $K_m$  (Mike). However, the presumption is that Mike cannot get a certificate signed by a recognized CA that states the owner of  $K_m$  is amazon.com. Therefore, Bobs browser will ultimately reject the connection.
  - However there could also be a malicious CA, who does say that  $K_m$  is amazon.com  
->  $Cert_m = \{ K_m \text{ is amazon.com} \}_s K_x$
- Dealing with expired certs (from CA)
  - CRL
  - OCSP - Every time a person connects to a certified website, the certs expiration date will be checked.
- Kerberos
  - Works on the basis of "tickets" to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. Its designers aimed primarily at a client-server model, and it provides mutual authentication— both the user and the server verify each other's identity.
  - After authentication with Kerberos, a client can 'throw away' their password as they have now proven their identity and can use the ticket granted to them to prove who they are from now on. Using this method, attackers who compromise the server will not be able to discover clients passwords.
  - Kerberos protocol messages are protected against eavesdropping and replay attacks.
  - Builds on symmetric key cryptography and requires a trusted third party, and optionally may use public-key cryptography during certain phases of authentication.
  - Drawbacks;
    - Single point of failure: It requires continuous availability of a central server. When the Kerberos server is down, no one can log in.
    - Kerberos has strict time requirements, the tickets have a time availability period and if the host clock is not synchronized with the Kerberos server clock, the authentication will fail.
    - Since all authentication is controlled by a centralized KDC, compromise of this authentication infrastructure will allow an attacker to impersonate any user.



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

- Each network service which requires a different host name will need its own set of Kerberos keys. This complicates virtual hosting and clusters.
- Authentication Server: replies to the initial authentication request from the client, when the user, not yet authenticated, must enter the password. In response to an authentication request, the AS issues a special ticket known as the Ticket Granting Ticket(TGT). Now users can obtain other services without having to re-enter their password.
- Ticket Granting Server: distributes service tickets to clients with a valid TGT, guaranteeing the authenticity of the identity for obtaining the requested resource on the application servers. The TGS can be considered as an application server (given that to access it, it is necessary to present the TGT) which provides the issuing of service tickets as a service.
- The Wide-Mouth Frog protocol is a computer network authentication protocol designed for use on insecure networks (the Internet for example). It allows individuals communicating over a network to prove their identity to each other while also preventing eavesdropping or replay attacks. This protocol provides both authentication and key exchange. The protocol can be specified as follows in security protocol notation:

A, B, and S are identities of Alice, Bob, and the trusted server respectively  $T_A$  and  $T_S$  are timestamps generated by A and S respectively  $K_{AS}$  is a symmetric key known only to A and S  $K_{AB}$  is a generated symmetric key, which will be the session key of the session between A & B  $K_{BS}$  is a symmetric key known only to B and S

$$A \rightarrow S : A, \{T_A, B, K_{AB}\}_{K_{AS}} \quad S \rightarrow B : \{T_S, A, K_{AB}\}_{K_{BS}}$$

Note that to prevent active attacks, some form of authenticated encryption (or message authentication) must be used.

The protocol has several problems:

- a global clock is required.
- the server S has access to all keys.
- the value of the session key  $K_{AB}$  is completely determined by A, who must be competent enough to generate good keys.
- can replay messages within period when timestamp is valid.
- A is not assured that B exists.
- The protocol is stateful. This is usually undesired because it requires more functionality and capability from the server. For example, S must be able to deal with situations in which B is unavailable.
- Needham-Schroeder: The goal of the protocol is to establish mutual authentication between two parties A and B in the presence of an attacker, who can;
  - Intercept messages
  - Delay messages
  - Read and copy messages
  - Generate messages



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

- Digital Certificates: We would like to be able to believe that when we confirm that  $\{M\}_{sKA}$  was signed by the owner of KA then the principal cannot easily repudiate their signature. This is subject to them not having declared their private key to be compromised.

#### Reflection Attack

- A reflection attack is a method of attacking a challenge-response authentication system that uses the same protocol in both directions. That is, the same challenge-response protocol is used by each side to authenticate the other side. The essential idea of the attack is to trick the target into providing the answer to its own challenge.

#### Attack

The general attack outline is as follows:

1. The attacker initiates a connection to a target.
2. The target attempts to authenticate the attacker by sending it a challenge.
3. The attacker opens another connection to the target, and sends the target this challenge as its own.
4. The target responds to the challenge.
5. The attacker sends that response back to the target on the original connection.

If the authentication protocol is not carefully designed, the target will accept that response as valid, thereby leaving the attacker with one fully authenticated channel connection (the other one is simply abandoned).

#### Solution

Some of the most common solutions to this attack are described below:

- The responder sends its identifier within the response so, if it receives a response that has its identifier in it, it can reject it.
  1. Alice initiates a connection to Bob
  2. Bob challenges Alice by sending a nonce.  $B \rightarrow A: N$
  3. Alice responds by sending back her identifier and the nonce encrypted using the shared key  $K_{ab}$ .  $A \rightarrow B: \{A, N\}_{K_{ab}}$
  4. Bob decrypts the message, makes sure its from Alice and not a message he had sent in the past by finding A in it and not B and if the nonce is the same as the one he sent in his challenge then he accepts the message.
- Require the initiating party to first respond to challenges before the target party responds to its challenges.
- Require the key or protocol to be different between the two directions.
  
- Man in the Middle Attack
  - [http://en.wikipedia.org/wiki/Man-in-the-middle\\_attack](http://en.wikipedia.org/wiki/Man-in-the-middle_attack)
- sKa - owner of public key Ka has signed this key. Non-repudiation. Owner is declaring his key has not been compromised and anything signed by this key is from him.
- SSL - can be used anywhere, where remote machines want to exchange keys and achieve integrity, secrecy and authentication.

### Certs from Certificate Authorities

- How does one come to trust that a public key  $K_A$  is owned by principal Alice?

Have Certification Authorities issue public key certificates that associate a principal's name with their public key.

- KT : widely known public key owned by trusted third party Trent.
- Assume that we trust statements that KT signs.
- Alice (securely) presents a public key  $K_A$  that she owns to Trent and asks for a signed certificate:

$$\text{cert} = \{ \text{Alice}, K_A, \text{validityPeriod} \}_s \text{KT}$$

- Alice presents certificate with her signed message  
 $A \rightarrow B : \text{cert}, \{ \text{a message from Alice} \}_s K_A$
- If Bob knows/trusts KT he can confirm that message is from Alice.

### Q1

1. An early version of the SSL protocol was specified as

$$\begin{aligned} \text{Msg1 } A \rightarrow B &: \{K_{AB}\}_{K_B} \\ \text{Msg2 } B \rightarrow A &: \{N_B\}_{K_{AB}} \\ \text{Msg3 } A \rightarrow B &: \{Cert_A, \{N_B\}_{K_A^{-1}}\}_{K_{AB}} \end{aligned}$$

where  $K_B$  is  $B$ 's public key,  $Cert_A$  a certificate for  $A$ 's public key  $K_A$  (private key  $K_A^{-1}$ ), and  $N_B$  a random nonce.  $K_{AB}$  is the proposed session key.

Outline an attack on this protocol.

Same as Q10

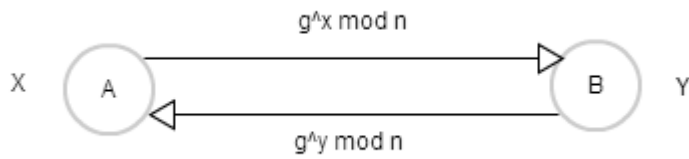
### Q 2

In lectures we saw how the Diffie-Hellman Key Exchange protocol can be used to establish the sharing secret between two principles. Suggest how you might adapt it to work for three principles and argue why your protocol works. For the sake of simplicity it is not necessary for the parties to authenticate each other.

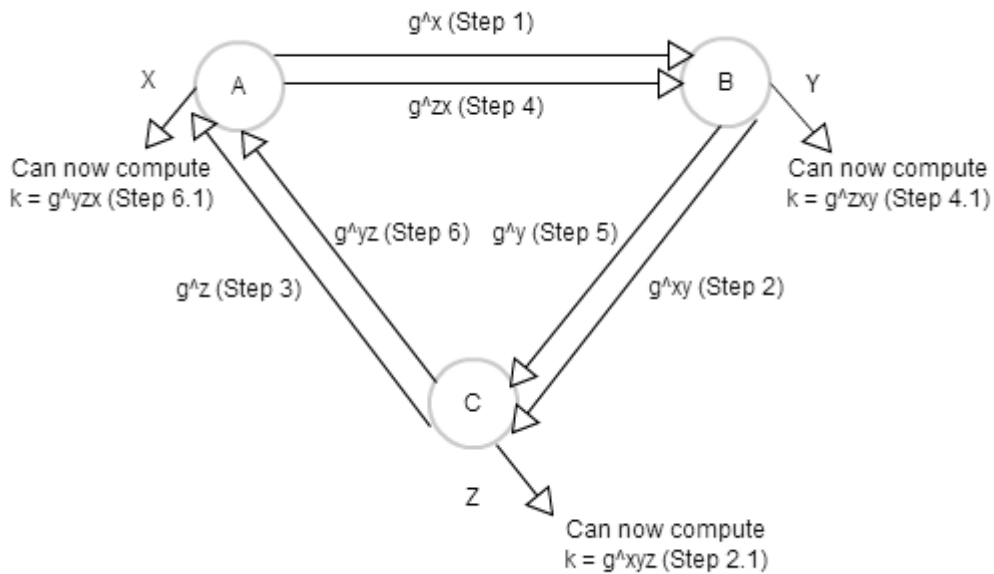
Basic DH Protocol (in any order):

$$\text{Msg1 } A \rightarrow B : g^x \text{ mod } n$$

$$\text{Msg2 } B \rightarrow A : g^y \text{ mod } n$$



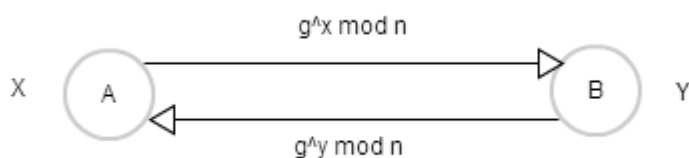
3 Principle DH



An attacker who finds any of these values cannot compute the key, as it involves solving the discrete logarithm problem, which is computationally infeasible. eg. given  $g^x = P$  find, little  $x$ . If  $x$  is large, this is a computationally difficult problem to solve.

Q 3

Suggest how one might use RSA digital signatures to provide authentication for principles using the Diffie-Hellman Key Exchange protocol. Explain how your approach solves the man-in-the-middle attack.



B needs to know  $g^x \text{ mod } n$  comes from A. One solution is to use digital signatures.

A  $\rightarrow$  B : {  $g^x \text{ mod } n$ , ... }<sub>sKA</sub>  
 B  $\rightarrow$  A : {  $g^y \text{ mod } n$ , ... }<sub>sKB</sub>





Title : CS4614 Study  
Student Name : Brian O Regan  
Student Number : 110707163  
Module : CS4614  
Exam Date: Wednesday 17<sup>th</sup> December 2014 @ 16.30

Alice knows that KB is Bob.  
Bob knows that KA is Alice.  
ie. A and B know they share  $KAB = g^{xy} \text{ mod } n$  with each other.

This approach solves the man in the middle attack because while an attacker can intercept messages sent between A and B, neither A or B will be fooled by any messages the attacker sends to them as the messages will not be digitally signed by the correct person. (A or B)

#### Q4

4. After a simple Diffie-Hellman key exchange neither party knows who they are talking to and must authenticate each other in some way. What is wrong with the following protocol that uses a DH-exchange followed by a mutual authentication. Propose a scheme to fix this weakness. Explain your answer.

Msg1  $A \rightarrow B : g^x \text{ mod } n, N_A$   
Msg2  $B \rightarrow A : g^y \text{ mod } n, N_B$   
Msg3  $A \rightarrow B : \{Alice, N_A + 1\}_K$   
Msg3  $B \rightarrow A : \{Bob, N_B + 1\}_K$

where  $N_A$  and  $N_B$  are challenges and  $K = g^{xy} \text{ mod } n$ .

Here Eve will already have had a chance to carry out a man in the middle attack from the first two messages, meaning the last two messages used for mutual authentication are useless.

Also A is not authorized and could actually be Eve masquerading as Alice.

Msg1  $A \rightarrow B : \{g^x \text{ mod } n, N_A\}_{sKA}$   
Msg2  $B \rightarrow A : \{g^y \text{ mod } n, N_A + 1, N_B\}_{sKB}$   
Msg3  $A \rightarrow B : \{Alice, N_B + 1\}_K$

Here, Alice signs message, Bob can be sure the message came from Alice. Bob also signs his.

Nonces are also added to add extra security to the protocol and to avoid other attacks such as reflection & replay attacks.

#### Q5

**Outline how the Kerberos protocol might be extended to incorporate public keys for authentication.**

(Alba's Answer)

If we suppose that all the principles have a public key, and kerberos knows that's public key so it works like an authenticator, so when a client wants to talk with Kerberos, it starts:

Msg1 :  $C - K : \{ \{ C, TGS, N_s \}_{sKc} \}_{Kk}$

Msg2 : K - C : { { TGS, Ktgs, Tc, Nc }sKk }Kc

- Ktgs is the public key of TGS
- Tc is {C, Kc} where kerberos authenticates C
- Kc, Ktgs are public keys
- sKc, sKk are session keys

**Q 6**

Suppose that we devise a very simple form of public-key certificate as follows. A certificate denoted  $\text{cert}(A, \text{keyA}, \text{keyB})$  states that the public key  $\text{keyA}$  is owned by A and has been signed by (the private key corresponding to public key)  $\text{keyB}$ .

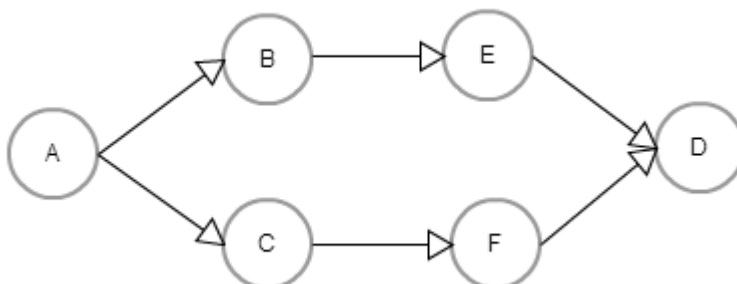
(a) Suppose that Alice's (A) public-key is  $\text{keyA}$  (she owns private  $\text{keyA-1}$ ). Alice holds :  $\text{cert}(B, \text{keyB}, \text{keyA})$ ,  $\text{cert}(C, \text{keyC}, \text{keyA})$ ,  $\text{cert}(D, \text{keyD}, \text{keyE})$ ,  $\text{cert}(E, \text{keyE}, \text{keyB})$  and  $\text{cert}(D, \text{keyD}, \text{keyF})$ . Can Alice trust  $\text{keyD}$ ? Explain your answer.

$\text{cert}(B, \text{keyB}, \text{keyA}) \Rightarrow B$  is trusted by A  
 $\text{cert}(C, \text{keyC}, \text{keyA}) \Rightarrow C$  is trusted by A  
 $\text{cert}(D, \text{keyD}, \text{keyE}) \Rightarrow D$  is trusted by E  
 $\text{cert}(E, \text{keyE}, \text{keyB}) \Rightarrow E$  is trusted by B  
 $\text{cert}(D, \text{keyD}, \text{keyF}) \Rightarrow D$  is trusted by F

Yes Alice can trust  $\text{keyD}$ . Alice trusts  $\text{keyB}$  &  $\text{keyC}$ . Bob trusts  $\text{keyE}$ . Eve trusts  $\text{keyD}$ .  $\text{keyD}$  is in Alices chain of trust, thus, Alice can trust  $\text{keyD}$ .

(b) Suppose Alice also holds  $\text{cert}(F, \text{keyF}, \text{keyC})$ , in addition to the certificates above, but she only marginally trusts (in a PGP-sense)  $\text{cert}(B, \text{keyB}, \text{keyA})$  and  $\text{cert}(C, \text{keyC}, \text{keyA})$ . Can she still trust  $\text{keyD}$ ? Explain your answer.

$\text{cert}(B, \text{keyB}, \text{keyA}) \Rightarrow B$  is trusted by A .. marginal trust  
 $\text{cert}(C, \text{keyC}, \text{keyA}) \Rightarrow C$  is trusted by A .. marginal trust  
 $\text{cert}(D, \text{keyD}, \text{keyE}) \Rightarrow D$  is trusted by E  
 $\text{cert}(E, \text{keyE}, \text{keyB}) \Rightarrow E$  is trusted by B  
 $\text{cert}(D, \text{keyD}, \text{keyF}) \Rightarrow D$  is trusted by F  
 $\text{cert}(F, \text{keyF}, \text{keyC}) \Rightarrow F$  is trusted by C





Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

Alice cannot trust keyD. Alice marginally trusts keyB & keyC. keyB is not trusted by anyone else. keyC is not trusted by anyone else. Alice needs to find at least two principals that trust either keyB or keyC in order to trust those certs herself (she currently only marginally trusts keyB and keyC). Alice cannot continue on her chain of (cert)trust, as she cannot fully trust anyone in her immediate chain of trust, and thus cannot get a (trusted)path to keyD. Thus, Alice cannot trust keyD.

PGP => Pretty Good Privacy

Levels of Cert Trust;  
Completely Trusted

- If any other key is signed by this key then add the new key to key-ring. Alice is saying that she trusts Bob to vouch for the validity of any key.

Marginally Trusted

- Certificate (key) must be signed for by two or more other keys before added to key-ring. Alice does not trust Bob very much and needs to have claims about keys corroborated by others.

Untrusted

- Do not use this key in determining whether other keys can be added to key-ring. Alice does not trust Bob to vouch for any key at all!

Unknown

- A level of trust can not be determined for this key.

---

**Q 7**

**Alice and Bob know each others public keys (KA and KB , respectively) and wish to determine whether they are speaking with each other. They use the following mutual authentication protocol.**

Msg1  $A \rightarrow B : \{N_a, A\}_{K_B}$

Msg2  $B \rightarrow A : \{N_A, N_B\}_{K_A}$

Msg3  $A \rightarrow B : \{N_B\}_{K_B}$

**Where NA and NB are nonces. Does this protocol work? Explain your answer.**

This protocol does not work. It is open to a man in the middle attack. If Eve can persuade A to initiate a session key with her, she can relay messages to B and convince B that he is communicating with A. The attack is proposed in the following protocol;

alpha : Msg1 : A - E :  $\{N_a, A\}_{K_E}$

beta : Msg1 : A[E] - B :  $\{N_a, A\}_{K_B}$  (Eve relays message to Bob, pretending to be Alice)

beta : Msg2 : B - A[E] :  $\{N_a, N_b\}_{K_A}$



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

alpha :  $Msg2 : E - A : \{ Na, Nb \}KA$  (Attacker relays message  $Msg3$  to A who will decrypt it)  
alpha :  $Msg3 : A - E : \{ Nb \}KE$  ( Alice sends decrypted  $Nb$  to Eve )  
beta :  $Msg3 : E - B : \{ Nb \}KB$  ( Eve re-encrypts  $Nb$  and sends it to Bob )

Here, Eve convinces B that she has decrypted  $Nb$ , making B falsely believe that A is communicating with him, and that  $Na$  and  $Nb$  are known only to A and B.

A better protocol (9 : 30)

beta :  $Msg2 : B \rightarrow A[ E ] : \{ Na, Nb, B \}KA$

Here B is saying that the message is from him, and thus if Eve is communicating with A, she cannot relay this message pretending it is from her as A will see in the msg (when A decrypts) that the msg is really from B.

---

#### Q 8

**My Netscape browser establishes an 'SSL' connection to a secure web site. How much confidence can I have in the security of any transaction that I engage in with this web site? Explain your answer.**

Assuming you have a proper implementation using 128-bit keys, it should take vastly longer than the age of the universe to brute force a private key.

However, vulnerabilities like man in the middle attacks, malicious SSL cert authorities, physical attacks to the host, etc. are still possible.

Unfortunately, there are easier ways to attack SSL sites than actually breaking the encryption, including using similar names to legitimate sites (with foreign alphabets, they may even be visually indistinguishable from the legitimate name), using JavaScript to fake the SSL lock, and even putting a lock icon into the page content, where many people will not realize it's a design artifact rather than a security guarantee.

Should the CA use (the old)MD5 encryption, rather than a SHA-1 cryptographic algorithm, the certs encryption can be broken by colliding two MD5 hashes.

Source;

<http://stackoverflow.com/questions/951386/how-secure-is-ssl>

<http://www.darkreading.com/security/news/212700234>

---

#### Q 9

**The fact that collisions can be generated in the MD5 one-way hash function has been known for some time. In January 2009 a group showed how to generate different X509 certificates that had the same signature when the signature was based on an MD5 hash of the certificate. Discuss the implications of this vulnerability.**



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

With this vulnerability considered, it is not unreasonable to assume that two sites may be 'verified' using the same certificate. In this case one site would be the original site, while the other would be the malicious fake site. However, both sites will possess the same certificate, virtually making them both 'appear/look' just as trustworthy.

### Q 10

10. A server(A) uses the following protocol to secure client (B) connections.

Msg1  $B \rightarrow A : Cert_B$   
Msg2  $A \rightarrow B : \{K_{AB}\}_{K_B}$   
Msg3  $B \rightarrow A : \{N_B\}_{K_{AB}}$   
Msg4  $A \rightarrow B : \{Cert_A, \{N_B\}_{K_A^{-1}}\}_{K_{AB}}$

where  $Cert_A$  is the X509 certificate (issued by a well known Certification Authority) for public key  $K_A$  (private key  $K_A^{-1}$ ) owned by A, and similarly,  $Cert_B$  is the certificate for B's public key.  $N_B$  a nonce,  $K_{AB}$  is the proposed session key. Outline an attack on this protocol. Suggest how the protocol might be repaired.

Attack 2 (wild guess (first time) )

How far can Eve get;

alpha : Msg1 : B [ E ] - A : CertB

alpha : Msg2 : A - B [ E ] : { KAB }KB

Eve cannot continue as she cannot decrypt msg2 to get the key...

And so, Eve then tricks B into authenticating with her, and sends him { KAB }KB from alpha msg2;

beta : Msg1 : B - A [ E ] : CertB

beta : Msg2 : A [ E ] - B : { KAB }KB

beta : Msg3 : B - A [ E ] : { Nb }KAB

Above Bob has encrypted the nonce for her. She then continues with her protocol run...

alpha : Msg3 : B [ E ] - A : { Nb }KAB

Now, Eve is authenticated as Bob.

Alternatively, Eve could just eavesdrop.

This is a variation of the SSL protocol looked at in class. Its goal is to provide mutual authentication, that is, that A knows she's talking to B, and vice-versa. Looking at the protocol, its clear that B is the only person who can learn of the session key K<sub>AB</sub>, since B is the only person with the private key  $K_B^{-1}$  (and can decrypt the message in Msg3).

B issues a challenge  $N_B$  to A, which he expects A to sign as proof that she is present. However, there is nothing that ties B to this nonce: an attacker, pretending to be A (to B), could ask A (in another run of the protocol) to sign the nonce  $N_B$  and then present that signed  $N_B$  back to B. Thus, the protocol does not provide authentication of A (to B).



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

This attack should be formally presented in terms of the protocol run-notation used in lectures (and I'll leave that to you as exercise).

### Attack 1

Here, CertB could be coming from Eve, where Eve is pretending to be Bob, but is actually sending her own CertE. As the Cert is not signed, A cannot be sure that it actually came from B. Once Eve has sent the bogus cert, A will inevitably send her back the key  $K_B$  (as she is pretending to be Bob), encrypted under  $K_E$  (as CertE is for public key  $K_E$ ). This allows Eve to extract the key, and continue the protocol and future protocols under the guise of Bob.

How to Repair protocol - through the use of signatures, timestamps and nonces;

Msg1 : B - A : { CertB } $s_{K_B}$

Msg2 : A - B : { Na, Ta,  $K_B$  } $s_{K_A}$

Msg3 : B - A : { Na + 1, Tb, Nb } $s_{K_B}$

Here each message is signed by the sender i.e. Eve will not be able to send messages of her own pretending they are from other principals.

If B takes a time longer than usual to reply, A will know that the message could have been corrupted in this 'extra time' and can reject the connecting principal.

---

### Q11

11. (a) Let  $cert_{K_A}^{K_T}$  be a public key certificate issued by Trent, the owner of public key  $K_T$ , and concerning the public key  $K_A$  owned by Alice. Describe the typical contents of this certificate, how it is implemented (signed) and how it is used in practice.

In the case of an email address;

$cert_{K_A}^{K_T} = \{ \text{alice@cs.ucc.ie}, K_T, K_A \}$

Another;

$cert_{K_A}^{K_T} = \{ \text{Alice}, K_A, \text{validityPeriod} \} s_{K_T}$

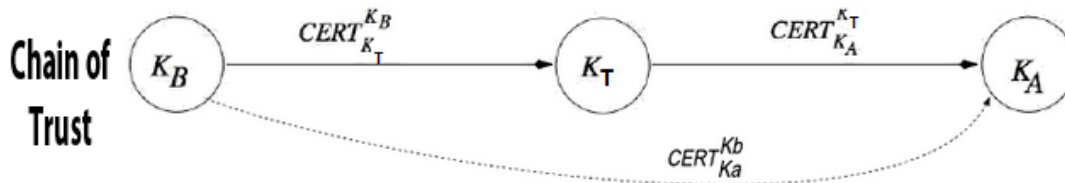
Here Trent is happy with the validity of the public-key  $K_A$ , and so, signs the  $cert_{K_A}^{K_T}$ . This cert signifies that Trent is happy with trusting any key that A signs. As a result, anybody that trusts T can trust certs he signs and so, can also trust that the key  $K_A$  is owned by Alice in this case.

How the cert is used;

Alice presents certificate with her signed message

$A \rightarrow B : cert, \{ \text{a message from Alice} \} s_{K_A}$

If Bob knows / trusts  $K_T$  he can trust that this message is from Alice.



Here, Bob trusts Trent, and Trent trusts Alice, so Bob in turn can also trust Alice.

(b) A Certification Authority owns public key  $K_T$  and issues certificate  $cert_{K_A}^{K_T}$  for Alice's public key  $K_A$ . Alice  $A$  sends a message  $M$  to Bob  $B$  using the following protocol:

$$A \rightarrow B : cert_{K_A}^{K_T}, \{M, h(M)\}_K, \{A, B, K\}_{sK_A}$$

where,  $K$  is a secret session-key,  $h()$  is a cryptographic one-way hash function and  $\{. . .\}_{sK_A}$  denotes signing by the owner of public key  $K_A$ . The goal of the protocol is to securely send a digitally signed message to  $B$ . Identify and discuss weaknesses in the protocol and suggest an improved protocol.

- Answered in End of Term 2012 Q1

(c) Given suitable public generator  $g$  and modulus  $n$ , principals  $A$  and  $B$  generate suitable secrets  $x$  and  $y$ , respectively, and engage in the Diffie-Hellman (DH) Key exchange.

$$\text{Msg1: } A \rightarrow B \quad g^x \text{ mod } n$$

$$\text{Msg2: } B \rightarrow A \quad g^y \text{ mod } n$$

- How do  $A$  and  $B$  determine their shared key  $K$ ?
- Explain why  $K$  cannot be determined by a third party observing the exchange.
- Suppose that  $A$  has RSA public key  $K_A$ . Revise the DH Key exchange in order to provide authentication of  $A$ .

i)

- Alice and Bob agree on a good  $g$  and  $n$ . Can be done in public.
- Alice picks a large random integer  $x$  and computes  $X = g^x \text{ mod } n$ . She keeps  $x$  secret, but it doesn't matter who knows  $X$  (discrete log).
- Bob behaves in the same way, picking  $y$  and computing  $Y = g^y \text{ mod } n$ .
- Alice sends  $X$  to Bob, and Bob sends  $Y$  to Alice.
- Alice computes  $k = Y^x \text{ mod } n$  and Bob computes  $k' = X^y \text{ mod } n$ .
- By modular arithmetic  $k = g^{xy} \text{ mod } n = k'$ .  $k$  is secret key between Alice and Bob.

ii)

$K$  cannot be determined by a third party observing the key exchange as they do not know the appropriate pieces of information necessary to compute the key  $K$ . As both Alice and Bob pick large random integers  $x$  and  $y$  and compute  $X = g^x \text{ mod } n$  and  $Y = g^y \text{ mod } n$  respectively, the third party



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

would have to solve the logarithm problem to obtain K. This is currently considered difficult, and no one listening on the channel can compute key  $g^{xy}$  in a reasonable amount of time.

iii)

Msg1  $A \rightarrow B \{ A, Na \}_sKa$

Msg2  $B \rightarrow A \{ A, Na + 1, Nb, g^y \text{ mod } n \}_sKb$

Msg3  $A \rightarrow B \{ Nb + 1, g^x \text{ mod } n \}_sKa$

Above provided authentication of A & B.

### Mid-Term 2012 Sample – Answers

Simon Foley,  
Department of Computer Science,  
University College, Cork.

CS4614 Computer Security  
End of semester test, 2012  
Time 45 min

STUDENT NAME	
STUDENT NUMBER	

TOTAL MARKS = 10/2 = 5%

1. Alice  $A$  (owner of public key  $K_A$ ) wishes to securely send a signed copy of a message  $M$  to Bob  $B$  using message exchange:

$$A \rightarrow B : \{M, h(M)\}_{K_{ab}}, \{A, B, K_{ab}\}_{K_{sA}}$$

where,  $h()$  is a one-way hash function,  $\{\dots\}_{sK_A}$  represents digital signing by the owner of  $K_A$  and  $K_{ab}$  is a session key generated by  $A$ . Comment on the effectiveness of this protocol. (2.5 marks)

**A** The goals of the protocol are as follows.

**Secrecy** is not provided by the protocol since it does not protect the secrecy of  $K_{ab}$ : any principal can read the key in  $\{A, B, K_{ab}\}_{K_{sA}}$ , since it is only signed. Once in possession of the key  $K_{ab}$ , the principal can decrypt  $\{M, h(M)\}_{K_{ab}}$  and read the message  $M$ . [Note: in order to protect the secrecy of  $K_{ab}$  it would be necessary to encrypt the key with, for example, the public key of  $B$ ,  $\{\{A, B, K_{ab}\}_{K_{sA}}\}_{K_B}$ . In this case  $A$  can be sure that the only principal that can decrypt the message is the owner of  $K_b$ , that is, the principal that holds the corresponding private key  $K_b^{-1}$ .]

**Integrity** is not provided since, given that the key  $K_{ab}$  can be determined by any principal, then any principal (Eve) can change the message to  $M'$ , recompute  $h(M)$  and re-encrypt with  $K_{ab}$  and  $B$  cannot detect the modification.





Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**Authentication** is not provided since, given that the key  $K_{ab}$  can be determined by any principal, then any principal (Eve) who sees a past exchange can generate a message and pretend that it came from A.

**Digital Signature** is not provided for the document since it is not possible for a third party (eg a judge) to prove without doubt that the message  $M$  originated from A.

A protocol that provides secrecy, integrity, authentication and digital signature is, for example,

$$A \rightarrow B : \{M\}_{K_{ab}}, \{\{A, B, K_{ab}, h(M)\}_{K_{sA}}\}_{K_B}$$

where  $K_B$  is the public key owned by B.

**A**

2. Rewrite the protocol in Question 1 whereby the session key is derived from a Diffie Hellman key exchange. (2.5 marks)

**A** Rather than having A generate the key  $K_{ab}$  Diffie Hellman is used between A and B. Given suitable generator  $g$  and modulus  $n$  known to A and B, then A generates secret  $x$  and B generates secret  $y$ , and

$$\text{Msg 1} : A \rightarrow B : \{g^x \text{ mod } n\}_{sK_A}$$

$$\text{Msg 2} : B \rightarrow A : g^y \text{ mod } n$$

$$\text{Msg 3} : A \rightarrow B : \{M\}_{K_{ab}}, \{A, B, h(M)\}_{sK_A}$$

where  $K_{ab} = g^{xy} \text{ mod } n$ . In this case, the signing of A's public DH component means that B is sure that he shares the key  $K_{ab}$  with A. However, A does not know who she speaks with (this could be rectified by having B sign his public DH component with his public key, if available).

**A**



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

3. Consider the following fragment from a Kerberos-style authentication protocol, whereby initiator  $A$  requests, and is granted, a ticket from Authentication Server  $S$  for use with service  $B$ .

$$\begin{aligned} \text{Msg 1 : } & A \rightarrow S : A, B, N_a \\ \text{Msg 2 : } & S \rightarrow A : T_{ab}, \{N_a, K_{ab}\}_{K_{as}} \end{aligned}$$

Principals  $A$  and  $B$  share long-term secret keys  $K_{as}$  and  $K_{bs}$  with server  $S$ , respectively;  $N_a$  is a nonce;  $\{\dots\}_K$  represents symmetric key encryption with secret key  $K$ . The server issues a ticket  $T_{ab} = \{L, K_{ab}\}_{K_{bs}}$  for the session key  $K_{ab}$ , valid for time period specified by  $L$ .

(a) Suppose that  $B$  above is a ticket granting service and, thus,  $T_{ab}$  is a ticket granting ticket. Propose a suitable protocol exchange between  $A$  and  $B$  that will result in  $A$  obtaining a ticket for a file server  $C$ . (2.5 marks)

**A** *Looking for any reasonable protocol fragment/description whereby  $B$  uses an authenticator  $\{A, C, t_a\}_{K_{ab}}$ ; timestamp  $t_a$ ) to test for freshness of request from  $A$  for ticket  $T_{ac}$  for  $C$ .*

$$\begin{aligned} \text{Msg 3 : } & A \rightarrow B : A, C, T_{ab}, \{A, C, t_a\}_{K_{ab}} \\ \text{Msg 4 : } & B \rightarrow A : T_{ac}, \{A, C, L', K_{ac}\}_{K_{ab}} \end{aligned}$$

$$\text{and } T_{ac} = \{A, L, K_{ac}\}_{K_{bc}} \quad \overline{\mathbf{A}}$$

(b) Why is it preferable for  $A$  to use a ticket granting service instead of asking  $B$  directly for a ticket for  $C$ ? (0 marks)

**A** *(There was a typo in this question, it should have read “Why is it preferable for  $A$  to use a ticket granting service instead of asking  $S$  directly for a ticket for  $C$ ?”. As a result, the question will be ignored and the other questions are graded out of 2.5 marks each. ) Answering the intended question, the design ensures that long-term secrets need not be retained on the user workstations. Alice’s long term key is a one-way hash of her password, which she must keep secret. It should not be stored on the workstation and the authentication server  $S$  is assumed to be hardened bastion host which stores all the user passwords. At login, Alice’s password is needed to determine the session key for the ticket-granting server, and once this key is determined the login daemon should completely delete her password/ $K_{as}$ : if the system is subsequently compromised by an attacker during her session then no long term secret is revealed.*

*The design can also help eliminate a possible bottleneck at  $S$  by spreading the load (requests for service tickets) across multiple TGS, if desired.*  $\overline{\mathbf{A}}$



Title : CS4614 Study  
 Student Name : Brian O Regan  
 Student Number : 110707163  
 Module : CS4614  
 Exam Date: Wednesday 17<sup>th</sup> December 2014 @ 16.30

(c) Describe an attack on the protocol whereby Eve can masquerade as A. (2.5 marks)

**A** The problem here is that the tickets do not bind a principal to the session key. For example,  $T_{ab}$  makes no mention of A in the ticket and so B cannot be sure that  $K_{ab}$  is a key that is good for talking with A.

The attack is as follows. Eve runs the protocol to request to connect to B.

$$\text{Msg}\alpha 1 : E \rightarrow S : E, B, N_e$$

$$\text{Msg}\alpha 2 : S \rightarrow E : T_{eb}, \{N_b, K_{eb}\}_{K_{es}}$$

Eve has a copy of the ticket  $T_{eb} = \{L, K_{eb}\}_{K_{bs}}$  intended for Bob. She simply pretends to be A and offers the ticket to Bob. For example, Eve pretending to be A requests a key to talk with C:

$$\text{Msg}\alpha 3 : A[E] \rightarrow B : A, C, T_{eb}, \{A, C, t_a\}_{K_{eb}}$$

B thinks its A and issues a session key for the service C.

$$\text{Msg}\alpha 4 : B \rightarrow A[E] : T_{ac}, \{A, C, L', K_{ac}\}_{K_{eb}}$$

### End-Term 2011 – Answers

Simon Foley,  
 Department of Computer Science,  
 University College, Cork.

CS4614 Computer Security  
 End of semester test, 2011  
 Time 45 min

STUDENT NAME	
STUDENT NUMBER	

TOTAL MARKS = 10/2 = 5%

1. Given suitable public generator  $g$  and modulus  $n$ , principals A and B generate suitable secrets  $x$  and  $y$ , respectively, and engage in the Diffie-Hellman Key exchange.

$$\text{Msg1: } A \rightarrow B \quad g^x \text{ mod } n$$

$$\text{Msg2: } B \rightarrow A \quad g^y \text{ mod } n$$

- (a) How do A and B compute their shared key? (2 marks)

**A** B calculates

$$K = (g^x \text{ mod } n)^y \text{ mod } n = (g^{xy} \text{ mod } n)$$

and A calculates

$$K = (g^y \text{ mod } n)^x \text{ mod } n = (g^{xy} \text{ mod } n)$$

Note: a complete answer will show how both parties compute their key (and that their key is the same value). **A**



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

*Question 1. continued*

- (b) Suppose that  $K_A$  is a public (RSA) key owned by  $A$ . Modify the protocol above so that it provides authentication of  $A$ . (2 marks)

**A** *DH does not provide authentication:  $A$  and  $B$  cannot be sure of who they share the key with.  $A$  should sign their public DH component:*

$$\text{Msg1: } A \rightarrow B \quad \{g^x \bmod n\}_{sK_A}$$

$$\text{Msg2: } B \rightarrow A \quad g^y \bmod n$$

*where  $\{\dots\}_{sK_A}$  denotes signing (using  $K_A^{-1}$ ) by the owner of  $K_A$ . Assuming that  $B$  knows that  $A$  is the owner of the public key  $K_A$  then  $B$  can use  $K_A$  to check the signature on message 1 and confirm that message 1 originated from  $A$ .*

*Note: a complete answer will give the revised (signed) message, state that  $B$  checks the signature on this message on receipt and that its assumed that  $B$  knows that  $A$ 's public key is  $K_A$ . A*

2. Alice ( $A$ ) wishes to communicate securely with Bob ( $B$ ) and proposes a symmetric session key  $K_{AB}$ , a copy of which she intends to give to Bob. Trent is a trusted third party who provides a message translation service. Trent shares symmetric  $K_{AT}$  with Alice, and symmetric key  $K_{BT}$  with Bob. All keys are 128-bit AES keys. The following protocol is used to pass the key  $K_{AB}$  to Bob.

$$\text{Msg1: } A \rightarrow T : B, \{A, K_{AB}\}_{K_{AT}}$$

$$\text{Msg2: } T \rightarrow A : \{A, K_{AB}\}_{K_{BT}}$$

$$\text{Msg3: } A \rightarrow B : \{A, K_{AB}\}_{K_{BT}}$$

- (a) What is the difference between long term and session keys? (2 marks)

**A** *A long term key is a secret that is shared between principals over a long period of time and is typically used to authenticate the principal and to securely exchange short-term/session keys. For example, a key based on a user password and used by an authentication server to authenticate a user, or a public-private key pair used to authenticate a web-site.*

*A session-term key is a secret shared between principals for a short period of time, for the duration of a session or for encrypting a piece of data. For example, a short-term key issued by a Kerberos TGS that is to be used to encrypt bulk data between a client workstation and a file-server for a period of 4 hours; a random symmetric key that is used to encrypt an email message (and the key is securely transmitted with the email message using a long-term key, such as a public key); a short-term random symmetric key that is used to encrypt HTTP traffic between a web-browser and a web-server. A*



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

*Question 2. continued*

- (b) Illustrate how Eve can subvert the protocol above and masquerade as Alice to Bob, *even when Alice does not initiate a key exchange with Bob.* (4 marks)

**A** *At some point in the past Alice initiates an exchange with Eve.*

$$\alpha.1 : A \rightarrow T : E, \{A, K_{AE}\}_{K_{AT}}$$

$$\alpha.2 : T \rightarrow A : \{A, K_{AE}\}_{K_{ET}}$$

$$\alpha.3 : A \rightarrow E : \{A, K_{AE}\}_{K_{ET}}$$

*Having eavesdropped on message  $\alpha.1$ , Eve, pretending to be Alice initiates a connection with Bob:*

$$\beta.1 : A[E] \rightarrow T : B, \{A, K_{AE}\}_{K_{AT}}$$

$$\beta.2 : T \rightarrow A[E] : \{A, K_{AE}\}_{K_{BT}}$$


$$\beta.2 : A[E] \rightarrow B : \{A, K_{AE}\}_{K_{BT}}$$

*Eve can now masquerade as Alice to Bob.*

*Note that there are variations to this attack. Another unconventional attack is*

$$\alpha.1 : E \rightarrow T : B, \{A, K_{AE}\}_{K_{ET}}$$

*and makes the assumption that the implementation of T is flawed in that on receipt of the message principal T that the encrypted ticket actually originated from the declared sender (E) but neglects to confirm that the principal mentioned in the 'ticket' corresponds to E. This is a bit of a stretch, but it does illustrate the weakness of the notation we use to describe protocols. For the test its OK as an answer so long as the explanation is clear and shows that you really understand the assumption that is being made, etc. **A***

	Title : CS4614 Study
	Student Name : Brian O Regan
	Student Number : 110707163
	Module : CS4614
	Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

**Mid-Term Sample 2011 – Answers**

Simon Foley,  
 Department of Computer Science,  
 University College, Cork.

**CS4614 Network Security**  
**2011 Mid-Term test**  
 Time: 45 mins

STUDENT NAME	
STUDENT NUMBER	

TOTAL MARKS = 10/2 = 5%

1. Explain the desirable properties of a one-way hash function (2 marks)

**A** A hash function  $h$  maps arbitrary length value  $x$  to fixed length value  $y$  such that:

- Hard to reverse. Given value  $y$  not feasible to find  $x$  with  $y = h(x)$ .
- Collision freeness. Hard to find values  $x, x'$  such that  $h(x) = h(x')$ .
- Unpredictability. The hash value  $h(x)$  does not give any information about any part of its operand  $x$ .

**A**

2. A document editor provides an option to store documents in encrypted form using DES in ECB mode and based on a user-provided eight character passphrase as the key. Prior to encryption, a block of null values is appended to the end of the plaintext document. When the document is loaded/decrypted, this last block is checked (for nulls) and used to confirm the integrity of the document.

Comment on the effectiveness of this design and suggest how it might be improved. (4 marks)

**A** Using nulls as a recognizer at the end of a message does not act as a check for integrity. An attacker can corrupt earlier ciphertext blocks without interfering with the encrypted block that contains the nulls.

*DES-ECB is not a suitable cipher mode for encrypting a stream of plaintext blocks since it encrypts each block independently. As a consequence*

- patterns in the plaintext show through in the ciphertext
- it is subject to a cut and paste attack whereby an attacker can replace any ciphertext block by the contents of another ciphertext block without interfering with the content of the surrounding blocks.

*Note that DES uses a 56-bit key which is moderately weak and given resources it is feasible for an attacker to carry out a brute force attack on the  $2^{56}$  bit key space. However, if we limit ourselves to an 8-character password then in practice it is likely that the effective key space is smaller (meaning less work for the attacker) assuming that the user limits their passwords to the more usual (printable) ASCII characters*

**A**



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

3. Alice logs in to Bob's remote server  $B$  over the Internet. Alice ( $A$ ) uses the following challenge-response user authentication protocol to prove her identity to  $B$ :

$$\begin{aligned} \text{msg1} &: A \rightarrow B : \text{userid\_of\_}A \\ \text{msg2} &: B \rightarrow A : \text{challenge} \\ \text{msg3} &: A \rightarrow B : \{\text{challenge}\}_{K_{AB}} \end{aligned}$$

where  $K_{AB}$  is a secret key (password) shared between  $A$  and  $B$  and  $\text{challenge}$  is a random four bit value generated using `unix rand()`.

Identify any flaws in the design of this protocol and suggest how it may be improved. (4 marks)

**A** *The purpose of a challenge response authentication protocol is to ensure that no two authentication runs of the protocol look the same and are therefore resilient to a replay attack by an eavesdropping attacker. In this case the challenge is too small (4 bits) and therefore its value (and consequent protocol run) will repeat at least after 16 runs of the protocol.*

*The attacker simply records a small number ( $\leq 16$ ) of authentication interactions between  $A$  and  $B$ , recording  $(\text{challenge}, \{\text{challenge}\}_{K_{AB}})$  pairs from message 2 and message 3. Then, the attacker simply connects to  $B$  pretending to be  $A$  and when presented with challenge  $\text{challenge}$ , simply looks up the corresponding  $\{\text{challenge}\}_{K_{AB}}$  from the recorded interactions, replays it to  $B$ , whereupon  $B$  thinks he is speaking with  $A$ .*

*The protocol can be improved by using a larger challenge (eg 128 bits) that is highly unlikely to be repeated during the lifetime of the use of the protocol. Alternatively, we could simply include a timestamp as a way of ensuring message freshness.* A



Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

## Mid-term 2013 Sample – Answers

Simon Foley,  
Department of Computer Science,  
University College, Cork.

**CS4614 Network Security**  
**Mid-Term test**  
Time: 45 mins

STUDENT NAME	
STUDENT NUMBER	

TOTAL MARKS = 10/2 = 5%

1. Explain the desirable properties of a one-way hash function (2 marks)

**A** A hash function  $h$  maps arbitrary length value  $x$  to fixed length value  $y$  such that:

- *Hard to reverse.* Given value  $y$  not feasible to find  $x$  with  $y = h(x)$ .
- *Collision freeness.* Hard to find values  $x, x'$  such that  $h(x) = h(x')$ .
- *Unpredictability.* The hash value  $h(x)$  does not give any information about any part of its operand  $x$ .

A

2. A document editor provides an option to store documents in encrypted form using DES in ECB mode and based on a user-provided eight character passphrase as the key. Prior to encryption, a block of null values is appended to the end of the plaintext document. When the document is loaded/decrypted, this last block is checked (for nulls) and used to confirm the integrity of the document.

Comment on the effectiveness of this design and suggest how it might be improved. (4 marks)

**A** Using nulls as a recognizer at the end of a message does not act as a check for integrity. An attacker can corrupt earlier ciphertext blocks without interfering with the encrypted block that contains the nulls.

*DES-ECB is not a suitable cipher mode for encrypting a stream of plaintext blocks since it encrypts each block independently. As a consequence*

- *patterns in the plaintext show through in the ciphertext*
- *it is subject to a cut and paste attack whereby an attacker can replace any ciphertext block by the contents of another ciphertext block without interfering with the content of the surrounding blocks.*

*Note that DES uses a 56-bit key which is moderately weak and given resources it is feasible for an attacker to carry out a brute force attack on the  $2^{56}$  bit key space. However, if we limit ourselves to an 8-character password then in practice it is likely that the effective key space is smaller (meaning less work for the attacker) assuming that the user limits their passwords to the more usual (printable) ASCII characters*

A





Title : CS4614 Study
Student Name : Brian O Regan
Student Number : 110707163
Module : CS4614
Exam Date: Wednesday 17 <sup>th</sup> December 2014 @ 16.30

3. Alice logs in to Bob's remote server  $B$  over the Internet. Alice ( $A$ ) uses the following challenge-response user authentication protocol to prove her identity to  $B$ :

$msg1 : A \rightarrow B : \text{userid\_of\_}A$   
 $msg2 : B \rightarrow A : \text{challenge}$   
 $msg3 : A \rightarrow B : \{\text{challenge}\}_{K_{AB}}$

where  $K_{AB}$  is a secret key (password) shared between  $A$  and  $B$  and  $challenge$  is a random four bit value generated using `unix rand()`.

Identify any flaws in the design of this protocol and suggest how it may be improved. (4 marks)

- A** The purpose of a challenge response authentication protocol is to ensure that no two authentication runs of the protocol look the same and are therefore resilient to a replay attack by an eavesdropping attacker. In this case the challenge is too small (4 bits) and therefore its value (and consequent protocol run) will repeat at least after 16 runs of the protocol.

The attacker simply records a small number ( $\leq 16$ ) of authentication interactions between  $A$  and  $B$ , recording  $(challenge, \{\text{challenge}\}_{K_{AB}})$  pairs from message 2 and message 3. Then, the attacker simply connects to  $B$  pretending to be  $A$  and when presented with challenge  $challenge$ , simply looks up the corresponding  $\{\text{challenge}\}_{K_{AB}}$  from the recorded interactions, replays it to  $B$ , whereupon  $B$  thinks he is speaking with  $A$ .

The standard `unix random number generator rand()` is not suitable since it generates a predictable sequence of challenges and potentially repeats challenges. For example, knowing the sequence of future challenges, the attacker records past interactions and can predict when a past challenge will repeat (and will have recorded the encrypted response).

The protocol can be improved by using a larger challenge (eg 128 bits) generated by a secure random number generator that is unlikely to be repeated during the lifetime of the use of the protocol. Alternatively, we could simply include a timestamp as a way of ensuring message freshness. **A**