EP2500 Networked Systems Security
Homework 1

Emonet Camille
Fleitz Pierre
12/01/2015

Work distribution :
We did the entire homework together, but Pierre is the one who wrote the final report. We decided then to say that Camille did 45% of the work and Pierre 55%.

1. Basic Cryptographic Primitives and Protocols

Exercice 1 : Key establishment =

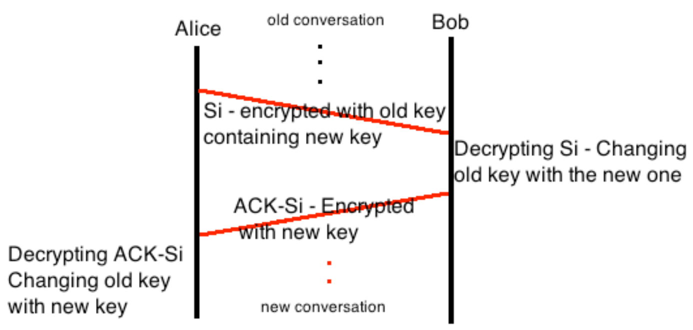
a) 1. We know that Alice and Bob use symmetric key cryptography, therefore they only need one key (for encryption and decryption).

2. And both of them need to know the key in order to communicate.

b) 1. This time Alice and Bob are using asymmetric key cryptography, therefore they gonna need 2 keys. One public key for encryption, and one private key for decryption.

2. Alice and Bob need to know the public key. Then Alice needs to know her private key only, just like Bob needs to know his private key only.

3. This is a solution we thought about :
- Alice send a packet to Bob encrypted using the old share key and containing the new key.
- Bob receives it and can decrypt it. The packet contains a new shared key.
- Bob changes his public key by the new one.
- Bob sends an ACK-Si to Alice with the new key that she sent him earlier.
- Alice receives the ACK-Si and she tries to decrypt is using the new shared key. If she can it means that Bob is using the new shared key, if she can't it means that the process failed.


Alice
old conversation
Bob
Si - encrypted with old key containing new key
Decrypting Si - Changing old key with the new one
ACK-Si - Encrypted with new key
Decrypting ACK-Si Changing old key with new key
new conversation

Exercice 2 : XOR encryption =

We know that we have C1= M1 ⊕ K1 and C2 = M1 ⊕ K2 but also that K1 == K2. We want to give the ASCII and the binary representations of M1, M2, and K1.

M1 : Eve assume that M1 ASCII's representation is the word : net.
In binary it gives us : 011011100110010101110100

Now we want to have the binary and ASCII of M2 and K2. If Eve want to have M2 she gonna have to proceed that way :

C1 = M1 ⊕ K1
C1 ⊕M1 = M1 ⊕ K1 ⊕M1
C1⊕M1=K1

So Eve is gonna have to do the calcul C1 ⊕M1 in order to have K1 representations :

C1	001101000100110001101111
M1	011011100110010101110100
C1 ⊕M1 = K1	010110100010100100011011
ASCII	Z)

Now Eve do have M1, K1. She wants M2 also. Here's what she gonna have to do :

C2 = M2 ⊕ K2
C2 = M2 ⊕ K1 (because we know that K1 == K2)
C2 ⊕ K1 = M2 ⊕ K1 ⊕ K1
C2 ⊕ K1 = M2

So this time Eve is gonna have to do the calcul C2 ⊕ K1 in order to have M2 representations :

C2	001010010100110001111000
K1	010110100010100100011011
C1 ⊕K1 = M2	011100110110010101100011
ASCII	sec

Proceeding that way will allow Eve to obtain M2 and K1.

Exercice 3 : Symetric Key =

First attack : Man-in-the-middle Attack.
Camille (the attacker) is listening the entire conversation. Doing that she receives RA from Alice and Ek(RA) from Bob. Then she can try to find K from the two information. If she finds K then she can impersonate Bob, or Alice in a future conversation by sending [RC(a random number), Alice] to Bob then EK(RC) later authenticating herself as Alice to Bob.

Second attack : Key Search (Brute force) Attack
In this attack Camille (the attacker) send a packet to Bob containing "Alice, Ra", then Bob answers "Bob, Rb, Ek(Ra)". Then in order to impersonate Alice, Camille gonna have to find the key and send Ek(Rb), to do that Camille is just going to attempt to decrypt the message with every possible key until Bob says "Ok it's you Alice, let's talk". It can take a while and there could be some security preventing Camille to try a limited number of times but if there is no such things then eventually it's gonna work.

Exercice 4 : Block Cipher =

a) If you want to obtain the original message from c you gonna have to : Decrypt c, there you will obtain R||m. Therefore if you want m, you gonna have to only keep the last 64 bits of R||m. There you have m, the original message.

b) What you want if you are the attacker: Send enough pairs so the oracles gives back 2 identicals block ciphers. Indeed, if you send (A1,A2) [knowing that A1 = A2] to the oracle and the oracle send back (C1,C2) (because you reach the probability to have the exact same 64 random bits) then you don't care to know it it's C1 or C2 that is encrypted because they are the same. You have the original message and the encrypted message, so you can try to break the algorithm. So yes the attacker can indeed infer something about the algorithm if he sends enough pairs to reach that point.

c) When we want to know if the scheme is secure, we want to know if is realisticly possible to reach the probability that both 64 bits of randoms parts are the same.
The probability that 1 bits from each are the same is (2^2 / 2*64). And the probability that 64 bits from both randoms messages are exactly the same is : (2^2 / 2*64)^64 wich is approximately 5^47. It means that if the attacker wants to be sure that he will have 2 exacts same randoms 64 bits he gonna have to try 5^47 pairs. Wich is an insane number of pairs to send.
Therefore, even if it is indeed possible to break the scheme, we can't say that it is not secure !

2. Physical Layer Security

Exercice 5 : Jamming =

i) Cjams = 4 channels.
The probability that t first messages are jammed p is :
p = Pr[X1 ∈ Y1, … , Xt ∈ Yt] = (Cjam/C)^t * ((1-Cjam)/C)^(n-t).
So for t = 5 and n = 5 :
p = (Cjam/C)^5 * 1
= (4/10)^5
= 0.01024

So the probability that they are unjammed q is :
q = 1 - p = 0.98976
So there is approximately 99% chances that a 5 seconds transmission is unjammed.

We know that 60% of 5 messages represent 3 messages jammed.

The probability that 3 messages out of 5 are jammed p2 is :

p2 = (4/10)^3 * (1 – 4/10)^(5-3)
= 0.023

So there is approximately 2.3% chances that 60% of a 5 seconds transmission is jammed.

ii) a) and rest : We tried to find some answers to these questions but we couldn't no find anything correct after all.

3. Denial of Service (DoS)

Exercice 6 : Flooding =

- I agree. Because in that way, bots from network A will only be able to consume a small amount of ressources than requests from network B. Indeed if the cache dedicated to requests coming from network A is smaller than it can only consume a limited amount of ressources. But still it will consume ressources for nothing and the large amount of legitimate hosts from A will suffer from this restriction as a lot of their requests will be dropped.
- Yes it would work, but it also mean that you gonna have to spy every packet coming on the server, so it could take some time and ressources. And again, it would block any requests coming from legitimate hosts from A.
- Note : SYN = 1 means that this is a SYN request, a resquest to start a TCP connection.
  It would then prevent the SYN-flooding, but it would also break every legitimate TCP request as said before.

- No it would not work, because at the end the only possible working value will be (H,F,G) so we can't know if it comes from A or not. It would work if you could mark only the 3 first (or even only the first router passed through) in that case you could know where the packet comes from. And again, legitimate hosts from A will be impacted by this solution.

  Even if it's not asked in the question at the end of this exercise we realized that we couldn't find out a solution were legitimate hosts from A would not be impacted. We gonna ask this question in the next lecture, but if we have a corrected version of our paper is it possible to give us a solution where we only block traffic coming from bots ?

4. Unauthorized Access

Exercice 7 : Password Entropy =

a) We are calculating the entropy per symbol of alphanumeric characters :

N = {0,9} + {a-z} + {A-Z} = 62
L = 1
H = 1*log(62) = 5.95 bits.

The entropy per symbol of alphanumeric characters is 5.95 bits.

b) We want to compute the entropy of a password that consists of 8 random symbols chosen from the extended ASCII table, that means that we gonna have a password composed of 8 random characters from a 256 characters dictionnary.
Then H = 8 * log(256) = 64 bits.

c) Both of them are « guess attacks ». It means that we gonna break the password by guessing it, trying some combinaisons. The difference between them is that for a Brute Force Attack, we gonna determine a secret by trying *every possible combinaisons*. On the other hand for a Dictionnary attack we gonna use precombiled list of option. It means that we are not going to try every combinaisons like a Brute Force Attack but only these wich have the highest probability to work.

d) The example we are going to use is the word « @nticOnstitutiOnnelement » (it is the longest word in the french dictionnary and we replaced "a" by "@" and "o" by "0"). Its entropy is : H = 25 * log(256) = 200, wich is a very good entropy because of the use of @ and 0. But still it's gonna be easy for a malware doing a dictionnary attack to understand that letters has been changed by some characters and that usually a lot of people replace "a " by "@" or/and "o" by "0", then it will be able to break the password easily.

e) It's gonna be a huge value, because we gonna have a combinaison of 16 random symbols and each of them could be one of the 256 ASCII symbols. Therefore the number of combinaison that we need to try during a Brute Force Attack will be 256^16 = 3^38.

f) This time the number of possible combinaison will be N = 16^6 and we now that we can try 100 000 combinaisons per seconde. Then it's gonna take (16^6)/100 000 = 168 seconds to crack the entire key space password. If the adversary is doing a brute-force attack, since he gonna have to try every combinaison then it's gonna take 168 seconds for him to crack the password.

   For the previous question we hade N = 3^38, then the corresponding time is : N/100 000 = 13 508 517 176 730 seconds. It's the time that it will take for an adversary doing a brute-force attack to crack the password. Note that it represent approximately 14 000 years.

Exercice 8 : Password Storage =

a) No it's not, it needs to be encrypted before you stock it in your database. It is pretty easy to break into a database, if nothing is encrypted in it you can have access to every information stored.
b) One-way hash fonction are pretty secure but an attacker could send a list of word to the system, wait for it to be hashed and the compare the non-hashed list and the hashed list in order to break the function. Therefore it is not completely secure to store them hashed in a database.
c) One way to store passwords securely could be to use a very "slow" hash function so it would cost a lot to hackers to try to break them, like the "Blowfish" hash function.
   Another way would be to add a "salt grain" to our dictionnary, to compromise it intentionnaly and in a way that we are the only one to understand so it can become really hard for the attacker to understand the dictionnary.

5. Secure Routing

Exercice 9 : Secure Routing Theory =

1) If we don't update frequently our routing table then a packet that we forward could be sent on a route that doesn't even exist anymore or we could create a loop and eventually the packet would never reach its destination. Plus, if an attacker is trying to inject a false route in the network he gonna have to inject it at every update, and that could prevent him trying to.
2) You need to know that everytime you send an update, you are sending the entire topology of the network that you know to the rest of the network. It represent already a lot of informations to forward in the network. Increasing it could, in my opinion, only mess with the integrity of the network. If every nodes don't have the time to process the latest update and already receiving a new one. And also because it can represent a broadcast storm and then cause traffic congestion.
3) **<u>We are not sure to understand correctly this question and we can't see what type of network we are supposed to study here. The answer could be 42 as every hope represent a cost of 2 and there is 20 hopes between them, but it sounds a little be to simple to be true. It must be a missunderstanding of the question.</u>**
4) Yes I agree. Because anyone in possession of the publick key certificate will be able to validate information digitally signed with the private key associated to the public key. If we have a Public Key Infrastructure then we can secure authentication.
5) Count-to-infinity is a problem of rooting loop that we can met in a distance-vectore protocol like RIP has. Nodes in RIP are not aware that the advertised cost is associated with the path that it advertised previously, causing a rooting loop between nodes and eventually a count-to-infinity problem.
6) BGP is a path-vector protocol, so when a speaker hears a BGP announcement containing itself in the path, it will know that this path should not be used.

Exercice 10 : Routing Information Protocol (RIP) =

1) Yes it could, a part of it actually. By sending a RIP advertisement to its neighbours e ;c ;b saying them that it is a better path to join the network usually reachable through f, so they could send him the traffic originally destined to f. There will be e ;c ;b also affected by that.
2) **<u>a.</u>**
Let's assume that we have a node B that want to enter in the network.
B authenticate in the network by using the password.
B generate a public key, using its network interface characteristics and associate to that key a time to live of x seconds.
B broadcast in the network its public key asking to the other nodes if it is unique or not.
-> If yes, B keep its public key until x expire. When x expire, B will renew its public key or create a new one and then will rebroadcast it.

-> If no, then B need to generate a new public key.
Every nodes in the network need to keep a track or a table where every nodes are associated to their known public key and their timer, so we can prevent identity usurpation.
We want to use a timer in order to discredit a key when the node is dead and doesn't give any news to the network for a certain amount of time.

b.
Let's say that a node B wants to join the network "governed" by a PKI.
B will generate a hash chain H(R)^1 .... H(R)^n with R a secret that B generated randomly. Then B will send to A (the PKI's CA) H(R)^n and n.
A will decide, or not, to authenticate B.
If it does then A gives a public key that will be associated to B and keep that key associated to H(R)^n and n in a table.
When B's public key will expire, B gonna have to send to A H(R)^(n-1), A will then be able to be sure that it is B that is asking for a new key by checking if H(R)^n match with H(H(R)^(n-1)), because only B will be able to generate H(R)^(n-1). B will be able to renew its public key n times.
In that way A is the only one that allow a node to enter in the network and once a node is in, it cannot be impersonate.

Exercice 11 : Border Gateway Protocol (BGP) =

1. If the attacker use the compromised AS-2 to modify, drop or introduce fake BGP updates, the consequences could be huge ! Indeed AS-1 and AS-3 can therefore have incorrect views of the network, leading to blackholing, redirection, or instability.
   Here we are interested in the case of redirecting, or AS BGP hijacking attacks.
   The attacker could send an update to AS-1 saying that it is a better/shorter (less cost effective) router if AS-1 want to reach 214.50.0.0/21 network. Once it would have done that AS-1 will believe that in order to reach 214.50.0.0/21 it should be better to go through AS-2 than through AS-3.
   The attacker can potentially advertise any prefixe he wants,
2. The system can be secured by using attestations. Attestations claim the right to originate a prefix. Each address destination is a signed statement of delegation of address space from one AS to another and the right to originate a prefix is checked through the validation of a delegation chain from ICANN to the advertising AS. (Note : ICANN = Internet Corporation for Assigned Names and Numbers).
   Route attestations are signed by each AS as it traverses the network. All signatures on the path sign previously attached signature.
   Therefore, using attestations, the router will be able to : validate the path, validate the fact that the path was traversed the Ases in the order indicated in the path and no intermediate Ases were added or removed by an adversary.
   The prefix that an AS is advertising is supposed to be checked and valid. Then it is impossible to attract traffic that is not dedicated to us.