Submitted By: Ruchira Pokhriyal
Student ID: 801085619
E-mail ID: rpokhriy@uncc.edu

ITIS 6200/8200 Principles of Information Security and Privacy

Weichao Wang

Fall Semester of 2018
Homework 2
Hand out: September 12th, 2018
Due time: September 21st, 2018 before 11:00 am

1. We have a symmetric encryption algorithm $E_K(M)=C$. Here K is the secret key, M is the plaintext, and C is the ciphertext. We (and the attacker) know that the key length is 192 bits. The attacker eavesdrops on the communication line and gets a copy of the ciphertext C1. Now the attacker decides to conduct the brute force attack and try every possible key to get the plaintext M1 (that means, every possible key of the 2^{192} keys). Let us assume that there is only one possible M1 and if the attacker sees it, he will know that this is the correct one. The attacker has 1,000,000 machines, with each machine having the capabilities to try 6,000,000 decryption of C1 with different keys per second. If one machine finds the right key, it will automatically notify the attacker. Now please answer, how many years (roughly) does the attacker need to try 10% of the keys?

   Hint: Note that Google has around 3 million servers. So we assume that the attacker is as powerful as 1/3 Google. Also, check the Internet and see what the expected life time of the Sun is. Can you crack the key before that? This question is to show that after a certain key length, brute force is not the best way of cracking it. Side channel, or some other mechanisms will serve the purpose better.

Answer:

We are given that, $\mathbf{E_K(M)=C}$, where
K-> Secret Key
M-> Plaintext
C-> Ciphertext and
Key Length = 192 bits.

If the attacker wants to conduct a brute force attack, he/she will have to try 2^{192} possible combinations of key to find the correct secret key 'K'**.** Therefore, number of possible combinations= $\mathbf{2^{192}}$

Let '**X**' be the number of years the attacker needs, to try 10% of the keys:

- Therefore, $0.1 * 2^{192}$ => $\mathbf{6.277 * 10^{56}}$
- Number of machines the attacker has= 1,000,000
- Operations performed by a single machine in a second=60,00,000
- Hence, total number of combinations checks per second, by all the machines= 1,000,000 * 6,000,000 => $\mathbf{6 * 10^{12}}$

Submitted By: Ruchira Pokhriyal
Student ID: 801085619
E-mail ID: rpokhriy@uncc.edu

- Total number of seconds in a year= 365 * 24 * 60 * 60=> **31536000 seconds**
- Finally, the number of years (X) to crack this 256-bit key symmetric encryption algorithm:

$$X = \frac{6.277 * 10^{56}}{[(6 * 10^{12}) \ * 31536000]}$$

Therefore, **X= 3.317 * 10^{36} Years**

From the above calculations, it can be concluded that the attacker would take roughly **3.317 * 10^{36} years** to crack 10% of 192-bit symmetric encryption key, using brute force attack. This value is more than the expected lifetime of sun, which is about 5 billion years.

2. Bob has a public-private key pair (pub_Bob, pri_Bob). Alice needs to send some information to Bob. She wants to make sure that when Bob opens the message, he can verify that this is from Alice but not anyone else. So she sends out the message as: [Alice, E_{pub_Bob}(message)] to Bob. Basically, she first sends out her name in clear text, then encrypts the message with Bob's public key. Please discuss, can an attacker M impersonate Alice and send out a packet in Alice's name? How can he do it? Here we assume that M also has the public key of Bob. For the same question, if Alice sends out [E_{pub_Bob}(Alice, message)], can M still impersonate Alice? (Here Alice puts her name in the encryption.)

Answer:

Case 1: [Alice, E_{pub_Bob}(message)]:

Since Alice sends out her name as clear text first (and then encrypts the message with Bob's public key), the attacker M can intercept this information and use replay attack to imitate Alice's identity. Attacker M can then send malicious messages, thereby confusing the intended receiver-Bob into believing that the communication is indeed happening with Alice and not M. Therefore, Alice's name being sent as cleartext provides the attacker M with a chance to send any fraudulent messages to Bob (since Bob's public key is known to all) and frame Alice as a sender of the message, even when she's actually not the legitimate initiator of the message.

Submitted By: Ruchira Pokhriyal
Student ID: 801085619
E-mail ID: rpokhriy@uncc.edu

Case 2: [E_{pub_Bob}(Alice, message)]:

In this case, since the entire message has been encrypted using Bob’s public key and therefore, it can only be decrypted using Bob’s private key. This particular message may initially seem to be secured; However, consider that, everyone has access to Bob’s public key including M. This means that M can draft a malicious message and include Alice’s name in it to trick Bob into believing that Alice is the sender. The malicious message and Alice’s name would then be encrypted in a single block and sent to Bob. Bob would decrypt this message using his private key and see that Alice is the sender, which actually isn’t the case. Thus, even in this scenario, M can impersonate Alice and there is no way for Bob to ensure that the sender of the message is legitimate.

Submitted By: Ruchira Pokhriyal
Student ID: 801085619
E-mail ID: rpokhriy@uncc.edu

3. Alice's computer stores the files in the following way: for every file F, the computer will calculate the hash value of the file hash(F) and store it after the file. Every time when Alice login, the machine will automatically hash all the files and compare the results to the stored hash values. In this way, if by accident the hard drive is mis-functioning and flips a few bits in a file, Alice can immediately detect it since the hash value will be different. Now an attacker hacks into Alice's machine and he tries to change several files. The attacker also knows the hash function that the computer uses. Please describe what the attacker needs to do so that the next time Alice login, the machine will not detect the changes. **Also, please discuss how we should improve the mechanism to detect such changes.**

   Hint: this is actually directly related to how to use hash functions to protect integrity of data. The defense capability of a hash function and that of a keyed hash function are actually different.

Answer:

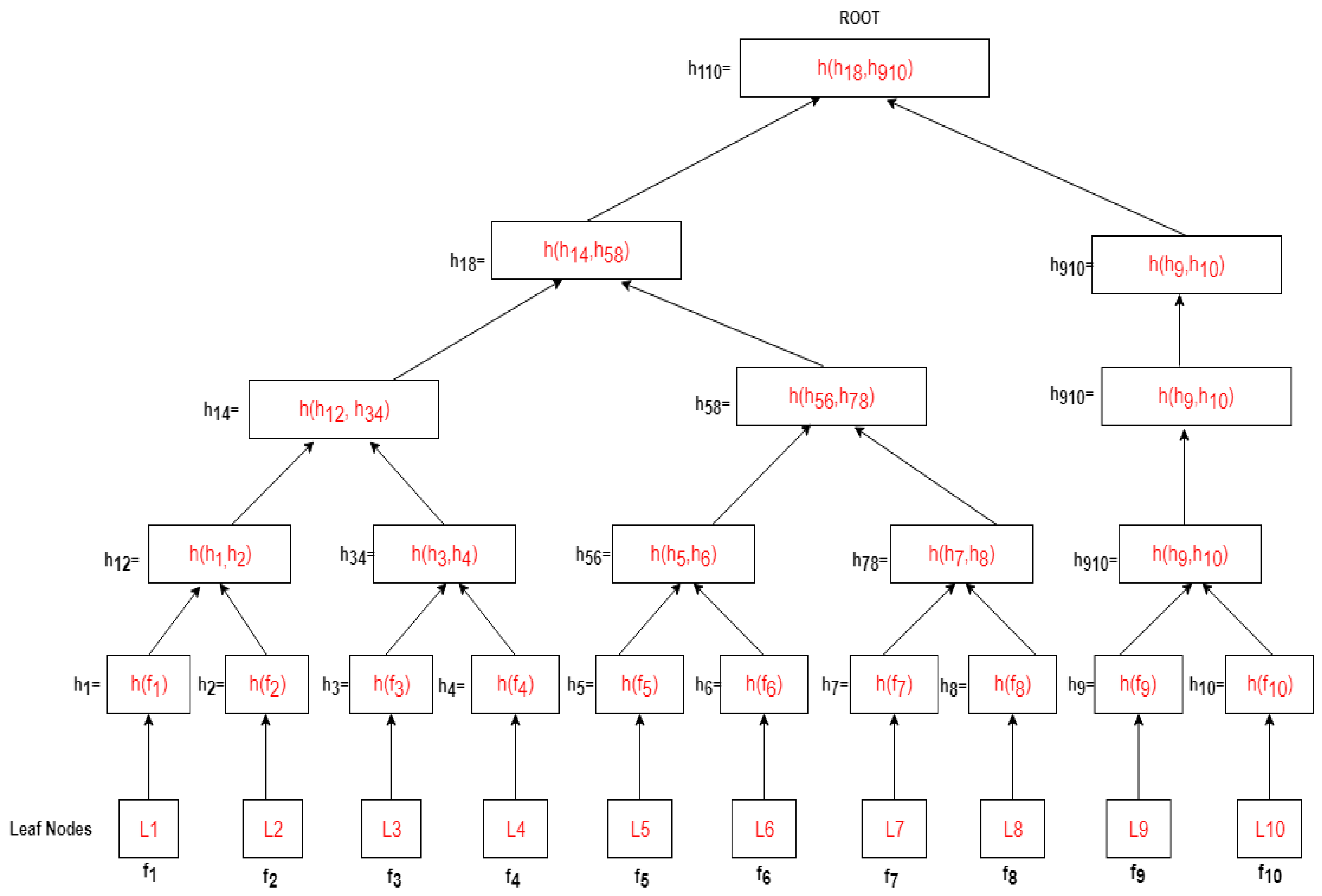
In the given scenario, Alice's computer uses hash function to compute hash values of the files. Hence, when the attacker hacks into Alice's system and attempts to modify the files, the hash values of these files would change and Alice would be notified about these unauthorized changes when she logs in the next time.
Notice that the attacker knows the hash function that Alice's computer is using. Therefore, the attacker can change the stored hash values of the files which he has modified, and the system will not detect any changes when Alice logs in later because now, the hash values will remain same. So, it appears to Alice as if the files on her system are untampered since the hash values seem to remain intact. But in reality, the stored hash values have been tampered with to further modify the files. This is a major data security threat.
To improve the security mechanism of such a system, Message Authentication Codes (MAC) should be used. MAC is a cryptographic checksum on data which uses a session key along with the hash to generate a tag. This helps to detect any intentional or unintentional modifications made to data. In this scenario, the attacker probably does not know the secret key to generate tags. The OS will compute the tags for the files and store them along with the files (Tags consist of file names and key). Now when Alice logs in, the OS will delete the key. The attacker would still be able to modify the files if he breaks into the system but now he won't be able to alter the hash values because he does not know what the key is. Hence, If Alice places her files inside MAC, she'd be able to detect all the files modified by the attacker.

Submitted By: Ruchira Pokhriyal
Student ID: 801085619
E-mail ID: rpokhriy@uncc.edu

4. Please draw a binary Merkle's hash tree with 10 leaf nodes. The leaf nodes are labeled as Leaf1 to Leaf10, which correspond to the hash values of the files F1 to F10, respectively. Now please answer:

   (1) For each node in the tree, please label clearly how the hash value is calculated based on its children; Please note that the number of leaf nodes is not power of 2. Therefore, you may need to change the way in which intermediate nodes are calculated. There are different ways to handle this. Please label clearly how you calculate each node.

Answer:



Submitted By: Ruchira Pokhriyal
Student ID: 801085619
E-mail ID: rpokhriy@uncc.edu

(2) Now the creator of the file F_7 needs to verify that his file's hash value is integrated in the root of the tree. Please show the minimum number of hash values in the tree that the creator needs to accomplish the task. **Please also show how the verification is accomplished.**

Answer:

The creator of file F_7 needs to verify that his file's hash value is integrated in the root of the tree- h_{110}. The creator already has f_7, using which he can generate h_7. Along with this, the minimum number of hash values in the tree that the creator needs to accomplish the task are: **h_8, h_{56}, h_{14}, h_{910}**.

Following are the verification steps to demonstrate how this works:

- h_7 concatenated with h_8 would generate h_{78}
- h_{56} concatenated with h_{78} would generate h_{58}
- h_{14} concatenated with h_{58} would generate h_{18}
- Finally, h_{18} concatenated with h_{910} would generate **h_{110}** which is the **final hash value.**

Since the calculated h_{110} value matches with the h_{110} value of the tree, the creator will know that his file's hash value (h_7) is indeed integrated in the root of the tree (h_{110}).