UConn, School of Computing
Fall 2024
CSE 3400/CSE 5850: Introduction to Computer and Network Security / Introduction to Cybersecurity

Assignment 4

Student: Isaac Piegat (using 1 late day)
Instructor: Prof. Ghada Almashaqbeh
Posted: 10/24/2024
Submission deadline: 11/2/2024, 11:59 pm

Notes:

- Solutions **must be typed** (using latex or any other text editor) and must be submitted as a pdf (not word or source latex files).

Problem 1 [30 points]
Let $h_1 : \{0,1\}^* \to \{0,1\}^n$ be a CRHF hash function (but not OWF) and $h_2 : \{0,1\}^* \to \{0,1\}^n$ be a OWF hash function (but not CRHF), based on that answer the following:

1. Construct a new hash function h' as follows: $h'(m) = h_1(m_0 \parallel h_2(m_1))$ such that $m = m_0 \parallel m_1$. Is h' a CRHF? Justify your answer.

   The function h_1 is not a CRHF because h_2 is not collission resistant. It is possible for the attacker fo find inputs m_1 and m_1' such that $h_2(m_1) = h_2(m_1')$. Once found, the attacker could then construct $m' = m_0 \parallel m_1'$. He could then compute $h_1(m_0 \parallel h_2(m_1))$ and $h_1(m_0 \parallel h_2(m_1'))$ where the different input would lead to the same output. The attackers advantage will always be 1 because the attacker will always be able to find a collision within h_2.

2. Construct a new hash function h'' as follows: $h''(m) = h_1(h_2(h_1(m)))$. Is h'' a OWF? Justify your answer.

   The function h'' is a OWF because h_2 is described as a OWF. The attacker would be able to invert the outer h_1, reducing the problem to $h_2(h_1(m))$; however, the attacker would be unable to invert h_2 thus unable to progress. The attacker could attempt to break the inner h_1, but it would be impossible as it is described as collision resistant thus the attacker would be unable to find $m = m'$ such that $h_1(m') = h_1(m)$. Therefore, there is no attacker advantage.

3. Construct a new hash function h''' as follows: for an input m, parse m as $m = m[1:6] \parallel m'$, where $m[1:6]$ is the first six bits of m and m' is the rest of m. Then compute $h'''(m) = h_2(m[1:6]) \parallel m'$. Is h''' a OWF? Justify your answer.

   The function h''' is not a OWF function. The attacker would first be able to directly extract m' leaving 6 unknown bits in m. The attacker would then be able to brute force search, as 6 bits means there are only 64 possible combinations. Once

found, the attacker could then resemble the original message, thus meaning h''' is not a OWF. The attacker advantage would be 1 as he would always be able to find the original message.

Note: We know that unkeyed hash functions cannot be CRHF, all the functions we have above are keyed but for simplicity the keys are omitted.

Note: If the scheme is insecure (i.e., not a CRHF or not a OWF, based on the question) then provide an attack against it and analyze its success probability. If the scheme is secure (i.e., a CRHF or a OWF, based on the question), just provide a convincing argument (formal security proofs are not required) and state whether the success probability of the attacker is negligible.

Problem 2 [40 points]
Roger has a database composed of 32 files, each of which is of size 135 KB, and he constructed a Merkle tree over this database (i.e., the database files are the leaves of the tree). The files are named as $f_1, f_2, \ldots, f_{32}$, and when constructing the tree these files are ordered from left to right, i.e., f_1 is the leftmost leave and so on. Roger sent the tree root to Serena. Roger has used SHA512 for the hash function h when constructing the tree. SHA512 has an output size of 512 bits (or 64 bytes). Based on that answer the following:

1. How many levels does the resulting Merkle tree have (including the root level)? And how many nodes does the tree have excluding the leaf level?

   6 levels, 31 nodes.

2. What is the total size of the tree including the files?

   $32 * (135 * 1024)B + 63 * 64B = 4427712$.

3. Bob wants to prove to Serena that file f_{29} is a member of the tree. What is the proof of inclusion that Roger will send to Serena? what is the total size of that proof (including the file itself)? How will Serena verify it?

   Level 0: Node $h(f_{29})$ and sibling node $h(f_{30})$.
   Level 1: Computed hash is $H_{29,30}$, sibling node is $H_{31,32} = h(h(f_{31}) \parallel h(_{32}))$.
   Level 2: Computed hash is H_{29-32}, sibling node is $H_{25-28} = h(H_{25,26} \parallel H_{27,28})$.
   Level 3: Computed hash is H_{25-32}, sibling node is H_{17-24}.
   Level 4: Computed hash is H_{17-32}, sibling node is H_{1-16}.

   Serena verifies it by computing each hash from f_{29} using the above path up until the root node. She then compares the computed hash to the root hash, and if they match it is verified. Total size of the proof would be $(135 * 1024) + 5 * 64 = 138560B$.

4. Repeat part (3) above for file f_9.

   Same steps as above. Goes $10 -> (11, 12) -> (13, 16) -> (1, 8) -> (17, 32)$. To verify, use the computed hash from the path and compare with root node. Total size would be $(135 * 1024) + 5 * 64 = 138560B$.

Problem 3 [30 points]

1. Is the use of an SPR hash function (instead of a CRHF) enough to preserve the immutability of the blocks in Bitcoin's blockchain? why?

   The use off an SPR hash function is enough to preserve the immutability of the blocks in a Bitcoin's blockchain. The attacker would need to find a collission for the most recent block, but because the hash function is a SPR, the attacker cannot find the value given the block - thuus still immutable.

2. In Bitcoin, the multi-layer (the binary tree one) Merkle tree construction is used to compute the digest (tree root) over the transactions included in a block. Bob wants to replace that with the two-layer Merkle tree. What are the impacts (both security and efficiency wise) of this decision? Justify your answers.

   There is no impact in terms of security. Efficiency is greatly affected because using a two layer Merkle tree will mean each block can only support up to two transactions, greatly slowing down the chain.

3. Roger has 4 secret documents and wants to prove that he knows them now while he wants to reveal the documents next year. He computed a digest over these documents using the Merkle-Damgard construction (using a hash function h that is OWF and CRHF) and posted the digest in the newspaper.

   Serena has another secret document, but she wants to prove that she knows all the five (hers and Roger's although she does not know Roger's documents) also be revealing the five documents next year after posting a digest of the five documents today.

   How can Serena do that? (recall that she will reveal the documents also next year). Hint: think about the extend function of the Merkle-Damgard construction.

   Serena would be able to use the length extension property of the Merkle-Dangard construction. This property would allow her to create a digest which includes both Roger's and her own documents, whether knowing them or not. She would begin by obtaining Roger's published digest (H_R). Serena would treat H_R as an initial hash value, thus $H_S = H_{H_R}(D_5)$ is what she would publish. Once Roger releases his document next year, Serena's digest would then prove she "knew" the documents at the time of publication.