Thomas Mackintosh

ECE 4802

21 November 2016

Assignment 4

1. 12.3:
2. For *c = ek(x* || *h(x))*, if an attacker knows the whole plaintext *x* then they know the entire input into the encryption. With knowledge of both the input and ciphertext output of the encryption function, the key to the encryption function could be brute forced if the key space is small enough. With knowledge of the key *k*, an attacker could potentially verify any message *x1*. This would only work in a situation where the key *k* does not change, i.e. this attack would not work with a one-time pad.
3. For *c = ek1(x* || *MACk2(x))*, the above attack would not work. Assuming the length of *k2* is long enough that the attacker cannot reasonable brute force it, the attacker will not be able to crack the key *k2* to get the correct output of the MAC. Without knowledge of the key to the MAC the attacker cannot get the key *k1* and thus will not be able to verify any message *x1.*
4. Draw the block diagram for both constructions

Matyas-Meyer-Oseas (modified)

E

mi

Hi-1

Matyas-Meyer-Oseas

mi

Hi-1

E

Hi

**⊕**

Hi

1. Show why the modified construction is not secure by using the decryption function of the block cipher to obtain a (second) preimage. Assume H0 is all zeroes.

If we use the decryption function on the modified Matyas-Meyer-Oseas block, we getthe message directly as an output: Dec(EncHo(m1)) = m1 for the first block of the block chain H0. If we move to the next block, we know the input to the block will be some message m2 and the hash result from the previous block, H1. We can easily use the decryption function on this block to obtain a second preimage: Dec(EncH1(m2) = m2.

1. Given the above method for finding preimages, describe how to find collisions.

We can do EncHi-1(mi) = Hi to find an m such that mi != mi1 and compute whether H(mi) = H(mi1).

1. Below is the code for my implementation of an AES CBC-MAC:



This code provides the following output:

Message: b'The quick brown fox jumps over the lazy dog'

Need: b'\x94maSb\x14\x08\x15\xef<\x8c:\xbe\xb9LF'

Have: b'\x94maSb\x14\x08\x15\xef<\x8c:\xbe\xb9LF'

Message: b'The quick brown fox jumps over the lazy doh'

Need: b'|K\x8b\x06\x96K#\x1d\x87\xdd\x1e\xca\xa9o\xad\x83'

Have: b'|K\x8b\x06\x96K#\x1d\x87\xdd\x1e\xca\xa9o\xad\x83'

1. Below is the code for cracking hashed passwords using SHA512:



The above code provides the following output:

Airmont

Ansonia

Anguilla

Apple Grove

Altus

Algonquin

Algerita

Annandale

Alvwood

Allenhurst

Ambler

Alamance

Allen City

Anselma

Ambridge

Agency

Adgateville

Accord

Abeytas

Advance

If we output the code with the time taken to crack each password, we get the following output:

PASSWORD: Airmont Time taken: 5.550299601018196 seconds

PASSWORD: Ansonia Time taken: 20.517417489987565 seconds

PASSWORD: Anguilla Time taken: 18.958472188998712 seconds

PASSWORD: Apple Grove Time taken: 23.13965288698091 seconds

PASSWORD: Altus Time taken: 14.389990436990047 seconds

PASSWORD: Algonquin Time taken: 9.697414041991578 seconds

PASSWORD: Algerita Time taken: 9.250819558015792 seconds

PASSWORD: Annandale Time taken: 19.625334457989084 seconds

PASSWORD: Alvwood Time taken: 14.420188821997726 seconds

PASSWORD: Allenhurst Time taken: 22.174967784027103 seconds

PASSWORD: Ambler Time taken: 15.48839113197755 seconds

PASSWORD: Alamance Time taken: 6.359459736995632 seconds

PASSWORD: Allen City Time taken: 10.89969251700677 seconds

PASSWORD: Anselma Time taken: 20.171272645995487 seconds

PASSWORD: Ambridge Time taken: 17.4873442449898 seconds

PASSWORD: Agency Time taken: 22.654370269010542 seconds

PASSWORD: Adgateville Time taken: 16.702048188017216 seconds

PASSWORD: Accord Time taken: 6.616238359012641 seconds

PASSWORD: Abeytas Time taken: 2.9110286129871383 seconds

PASSWORD: Advance Time taken: 18.245870003011078 seconds

From the above results, we see that it takes 81.04014845800702 seconds to get through the first 5 passwords which are preceded with a salt at 5000 rounds (average 16.2080296916 seconds per password.) We also see it takes 130.680724372 seconds to get through the next 10 passwords without a hash at 5000 rounds (average 13.068072437 seconds per password.) Lastly, it takes 44.840311862 seconds to get through the last 5 passwords which are hashed at 25000 rounds without a salt (average 8.9680623724 seconds.) From this data we can see the most secure form of SHA512 is with a salt at 5000 rounds. SHA512 hashed at 25000 rounds is the least secure as it can be broken the quickest.