Cryptographic hash functions and MACs

Solved Exercises for Cryptographic Hash Functions and MACs

Enes Pasalic
University of Primorska
Koper, 2014

Contents

1 Preface **3**

2 Problems **4**

1 Preface

This is a collection of solved exercises and problems concerning the design and analysis of cryptographic hash functions and message authentication codes (MAC). These cryptographic primitives are widely used in compressing the binary data and their cryptographic security is of crucial importance in modern society. These exercises are primarily meant for students who have a moderate knowledge in cryptography, and therefore the material is more suitable for graduate students that may wish to extend their understanding about these important primitives. Its aim is to gain a deeper knowledge of the most important design rationales of hash functions and the security evaluation of some standard approaches. The contents are arranged to permit enough flexibility to allow the presentation of a traditional introduction to the subject.

Enes Pasalic
enes.pasalic@upr.si

2 Problems

In these exercises we consider some basic but more advanced approaches related to the design and security of cryptographic hash functions and message authentication codes (MACs). The students are advised to consult "Handbook of applied cryptography" by Menezes, von Oorschot, and Vanstone for all relevant definitions and facts related to the topics treated here.

1. A cryptographic hash function h takes as input a message of arbitrary length and produces as output a message digest of fixed length, for example 160 bits. Certain properties should be however satisfied:

   (a) Given a message m, the message digest $h(m)$ can be calculated very quickly.

   (b) Given a message digest y, it is computationally infeasible to find an m with $h(m) = y$(in other words, h is a one-way, or preimage resistant function).

   (c) It is computationally infeasible to find messages m_1 and m_2 with $h(m_1) = h(m_2)$ (in this case, the function h is said to be strongly collision-free).

   Let n be a large integer. Let $h(m) = m \pmod{n}$ be regarded as an integer between 0 and $n - 1$. Show that h satisfies (a) but not (b) and (c).

   Solution: This function clearly satisfies (a). However, (b) and (c) fail. Given y, let $m = y$. Then $h(m) = y$. So h is not one-way. Similarly, choose any two values m_1 and m_2 that are congruent mod n. Then,

   $$h(m_1) = h(m_2),$$

   so h is not strongly collision-free.

2. This example illustrates how certain simple constructions fail to ensure the cryptographic strength of the proposed hash algorithms. More precisely, the collisions for the proposed methods are easily found.

   (a) Let the input data be of the form $X = (X_0, X_1, X_2, \ldots, X_{n-1})$ where each X_i is a byte. Consider the following hash function :

   $$h(X) = X_0 + X_1 + X_2 + \ldots + X_{n-1},$$

   where + stands for bitwise modulo two addition. Is this a secure hashing method in the sense that collisions are hard to find ?

   Solution: One can verify that for instance

   $$h = (10101010, 00001111) = h(00001111, 10101010).$$

   Notice that it is easy to find other collision pairs easily.

(b) Consider now the case

$$h(X) = nX_0 + (n-1)X_1 + (n-2)X_2 + \ldots + X_{n-1}.$$

Can you find some collision pairs in this case ?

Solution: In this case we at least have

$$h(10101010, 00001111) \neq h(00001111, 10101010).$$

On the other hand, one can also easily found other collision pairs.

3. (**Hard**) The following example, sometimes called the discrete log hash function, is due to Chaum, van Heijst, and Pfitzmann. It satisfies (b) and (c) above but is much too slow to be used in practice. However, it demonstrates the basic idea of a hash function. Certain knowledge in number theory is however required.
The choice of the parameters used in hashing is as below :

- First we select a large prime number p such that $q = (p-1)/2$ is also prime.
- We now choose two primitive roots α and β in $\mathbb{Z}_p$. Since α is a primitive root, there exists a such that $\alpha^a \equiv \beta \pmod{p}$. However, we assume that a is not known (finding a, if it is not given in advance, involves solving a discrete log problem, which we assume is hard).
- Write $m = x_0 + x_1 q$ with $0 \leq x_0, x_1 \leq q-1$. Then define,

$$h(m) \equiv \alpha^{x_0}\beta^{x_1} \pmod{p}.$$

The hash function h will map integers $\pmod{q^2}$ to integers $\pmod{p}$. Therefore, the message digest contains approximately half as many bits as the message. This is not a drastic reduction in size as is usually required in practice, but it suffices for our purposes.

Show that the function h is probably strongly collision-free, where probably means that it is up to the difficulty of solving a discrete log problem.

Solution: We show this by proving that if we know messages $m \neq m'$ for which $h(m) = h(m')$, then we can solve the discrete logarithm $a = \log_\alpha(\beta)$, that is, for given α and β satisfying $\alpha^a \equiv \beta \pmod{p}$ we can find this unknown a. The proof of our claim is as follows. Write

$$m = x_0 + x_1 q,$$

and

$$m' = x_0' + x_1' q,$$

and suppose

$$\alpha^{x_0}\beta^{x_1} \equiv \alpha^{x_0'}\beta^{x_1'} \pmod{p}.$$

Using the fact that $\alpha^a \equiv \beta \pmod{p}$, we rewrite the above equation as

$$\alpha^{a(x_1-x_1')-(x_0'-x_0)} \equiv 1 \pmod{p}.$$

Since α is a primitive root $\pmod{p}$, we know that

$$\alpha^k \equiv 1 \pmod{p}$$

if and only if $k \equiv 0 \pmod{p-1}$. In our case, this means that

$$a(x_1 - x_1') \equiv x_0' - x_0 \pmod{p-1}.$$

Let now $d = \gcd(x_1 x_1', p-1)$. There are exactly d solutions to the preceding congruence, and they can be found quickly. By the choice of p, the only factors of $p-1$ are $1, 2, q, p-1$. Since $0 \leq x_1, x_1' \leq q-1$, it follows that

$$-(q-1) \leq x_1 - x_1' \leq q-1.$$

Therefore, if $x_1 - x_1' \neq 0$, then it is a nonzero multiple of d of absolute value less than q. This means that $d \neq q, p-1$, so $d = 1$ or 2.
Therefore there are at most 2 possibilities for a. Calculate α^a for each possibility; only one of them will yield β. Therefore, we obtain a (thus solve a discrete log problem), as desired.
On the other hand, if

$$x_1 - x_1' = 0,$$

then the preceding yields

$$x_0' - x_0 \equiv 0 \pmod{p-1}.$$

Since $-(q-1) \leq x_1 - x_1' \leq q-1$, we must have $x_0' = x_0$. Therefore, $m = m'$, contrary to our assumption.

Based on the above result, it is now easy to show that h is preimage resistant. Suppose we have an algorithm g that starts with a message digest y and quickly finds an m with $h(m) = y$. In this case, it is easy to find $m_1 \neq m_2$ with $h(m_1) = h(m_2)$:

- Choose a random m and compute $y = h(m)$, then compute $g(y) = m$. Since h maps q^2 messages to $p-1 = 2q$ message digests, there are many messages m' with $h(m') = h(m)$. It is therefore not very likely that $m' = m$.
- If we still get $m = m'$, try another random m. Soon, we should find a collision, that is, messages $m_1 \neq m_2$ with $h(m_1) = h(m_2)$.

The preceding result shows that we can then solve a discrete log problem. Therefore, it is unlikely that such a function g exists.

4. The Davis-Price model is used as a hash scheme,

$$H_i = H_{i-1} \oplus E(M_i, H_{i-1})$$

and recall the complementation property of DES : If $Y = E(K, X)$ then $Y' = E(K', X')$. Use this property to find a collision for a message $M = M_1, M_2, \ldots, M_N$.

Solution: Let us consider $M = M_1^2, M_2^2, \ldots, M_N^2$, and use $M_1^2 = M_1'$ and IV' as the initial value. Then,

$$H_1' = H_0 \oplus E(M_1', IV') = IV' \oplus E(M_1, IV)' = IV + E(M_1, IV) = H_1.$$

The remaining blocks may be taken to be the same $M_i^2 = M_i$ for $i \geq 2$.

b) Show that a similar attack succeeds for,

$$H_i = M_i \oplus E(H_{i-1}, M_i)$$

Solution: Here the same reasoning is valid as above. Complementing IV and M_1 gives the same hash value H_1.

5. There are many variations to the DSS signature scheme. In the Nyberg-Rueppel variation the signature is computed as When generating the signature, it is checked that $(g^k \bmod p) \bmod q \neq 0$. Show how the hash code $H, 0 < H < q$, can be recovered from the signature (r, s) using public information only, including the public key $g^a \bmod p$.

Solution: The hash code H is recovered from the signature (r, s) given the public key $(g^a \bmod p)$ by calculating

$$\begin{aligned} H &= r(g^k \pmod p)^{-1} \pmod q \\ &= r(g^{s+ar} \pmod p)^{-1} \pmod q \\ &= r(g^s(g^a)^r \pmod p)^{-1} \pmod q \end{aligned}$$

6. Consider the following signature scheme. The scheme group is $\mathbb{Z}_q$ for a large prime q and a generator g of $\mathbb{Z}_q$. A user has a private key α and a public key $X = g^x$.

To sign message m, one first computes $h = H(m)$ for some hash function H. Then one computes $z = \alpha/h$ (assuming $h \neq 0$). The signature is g^z.

Verification of signature s consists of checking that $s^h = X$. Is this a good scheme, i.e.

(a) Will correct signatures be accepted?

Solution: (a) Correct signatures will be accepted, since

$$s^h = g^{zh} = g^{\alpha} = X.$$

(b) Is it infeasible to sign an arbitrary message without knowing α ?
(b) **NO !**

We have from the above equation that

$$s = X^{h^{-1}}$$

Given a message, anyone can hash it to get h, compute h^{-1} by the extended Euclidean algorithm and then compute the signature using the user's public key.

7. Let $h : \{0,1\}^* \to \{0,1\}^n$ be a hash function that is second-preimage and collision resistant. Let $h' : \{0,1\}^* \to \{0,1\}^{n+1}$ be the hash function given by the rule

$$h'(x) = \begin{cases} 0||x & x \in \{0,1\}^n, \\ 1||h(x) & \text{otherwise.} \end{cases}$$

Prove that h' is not preimage resistant, but still second-preimage and collision resistant.

Solution: The modified hash function h' is not preimage resistant, since for any hash value y of the form $0||x$, a preimage is x.

Therefore, we can find a preimage for at least one half of all possible hash values.

Next we prove that h' inherits second-preimage and collision resistance from h. We show that if we can find a collision or a second preimage for h', then we can easily do so for h. Suppose

$$\exists x_0 \neq x_1 : h'(x_0) = h'(x_1).$$

Two cases:

(a) First bit of $h'(x_0)$ is 0. Impossible as implies $x_0 = x_1$.
(b) First bit of $h'(x_0)$ is 1. Then $h(x_0) = h(x_1)$ a contradiction, as h is collision resistant.

8. Given are two protocols in which the sender's party performs the following operation:

Protocol A:

$$y = e_{k_1}(x||H(k_2||x)),$$

where x is the message, H is a hash function such as SHA-1, e is a symmetric-key encryption algorithm, E is a public key encryption, "||" denotes simple concatenation, and k_1, k_2 are secret keys which are only known to the sender and the receiver.

Protocol B:

$$y = x, E_{k_{pub}}(H(x)),$$

where k is a shared secret key, and k_{pr} is a private key of the sender (not shared with the receiver) and k_{pub} is a public key of the receiver.

a) Provide a step-by-step description (e.g., with an itemized list) of what the receiver does upon reception of y.

Solution a) The receiver does the following, protocol A:

- Decrypt using k_1 and obtain $x||H(k_2||x)$. Extract x.
- Hash $k_2||x$ and compare to $H(k_2||x)$.

In case of protocol B (which is completely flawed):

- Decrypt using k_{priv} and obtain $H(x)$. Extract x.
- Hash x and compare to $H(x)$.

b) State whether the following security services:

- confidentiality
- integrity
- non-repudiation (preventing an entity from denying previous commitments or actions)

is given for each of the two protocols given in the previous problem. You have to justify your answer in every case (1-2 sentences every time can be sufficient).

Solution b) In the case of protocol A:

- confidentiality - YES through encryption
- integrity - YES through hashing
- non-repudiation - NO both parties can claim that the other party has created the message.

In case of protocol B:

- confidentiality - NO, no encryption
- integrity - NO, as anybody can replace the message and compute the hash
- non-repudiation - NO anybody can create such a message.

The following few exercises treat the use of block ciphers in specific modes for constructing hash functions and MACs. This however requires careful use of the operating mode and a suitable embedding of a block cipher to resist simple attacks that can be mounted.

9. Let $E_k : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n$ be a block cipher with key length n and block length n (the key is the first input, the block the second). We can make a compression function $f : \{0,1\}^{2n} \mapsto \{0,1\}^n$ from the block cipher with the rule

$$f(H_{i-1}, m_i) = E_{H_{i-1}}(m_i) \;\; H_0 = IV,$$

where $m = m_1||m_2||\cdots||m_l$ and $h(m) = H_l$. This also defines hash function $h : \{0,1\}^* \to \{0,1\}^n$.
Show how one can find a collision in h.

Solution: The function h is not preimage resistant. Since $f(H_{i-1}, m_i) = E_{H_{i-1}}(m_i)$ one can easily obtain a preimage of a single block message using the decryption function.

To obtain a second preimage or a collision, we use the same idea. Let $m = m_1||m_2$ be a string of length $2n$ and we want another string with the same hash value. Let

$$H_0 = IV, H_1 = E_{IV}(m_1), H_2 = E_{H_1}(m_2).$$

That is

$$h(m_1||m_2) = H_2.$$

Note that we ignore padding.

Now choose any m_1' arbitrarily and set

$$m_2' = D_{E_{IV}(m_1')}(H_2).$$

Then

$$\begin{aligned} h(m_1'||m_2') &= E_{H_1'}(m_2') = E_{E_{IV}(m_1')}(m_2') = \\ &= E_{E_{IV}(m_1')}(D_{E_{IV}(m_1')}(H_2)) = \\ &= H_2 = h(m_1||m_2). \end{aligned}$$

10. We consider the encryption of an n-block message $x = x_1||\ldots||x_n$, by a block cipher E in CBC mode. We denote by $y = y_1||\ldots||y_n$ the n-block ciphertext produced by the CBC encryption mode.

    (a) Show that one can extract information about the plaintext if we get a collision, i.e., if $y_i = y_j$ with $i \neq j$.

    Solution: If $y_i = y_j$ for $i \neq j$, then

    $$y_{i-1} \oplus x_i = y_{j-1} \oplus x_j.$$

As y_{i-1} and y_{j-1} are known, we can deduce the value

$$x_i \oplus x_j = y_{i-1} \oplus y_{j-l}.$$

(b) What is the probability of getting a collision when the block size of E is 64 bits ?

Solution: Using the Birthday Paradox, we know that the probability of getting a collision when we have $n = \theta\sqrt{2^{64}}$ blocks at disposal is approximately equal to $1 - e^{-\sqrt{\theta^2}2}$.

11. The CBC-MAC construction (see Handbook of applied cryptography for the definition of CBC mode) builds a MAC function from a block cipher by taking the last encrypted block of the CBC-mode encryption. Can we similarly invent an ECB-MAC or an OFB-MAC and obtain secure constructions ?

Solution: ECB-MAC: We consider the last block of the ECB encryption of a message as the MAC of this message. Obviously, this scheme is highly insecure since the MAC of the message

$$m = m_1||\dots||m_n||m_{n+l},$$

where $||$ stands for concatenation, is equal to the MAC of

$$m' = m'_1||\dots||m'_n||m_{n+l}$$

for any blocks $m_l, \dots, m_n$.

OFB-MAC: We consider the last block of the OFB encryption of a message as the MAC of this message. Once again, the scheme is insecure. Given the MAC c of a one-block message m, i.e., $c = m \oplus E(IV)$, it is easy to forge the MAC of the message

$$m' = m \oplus \alpha$$

as it simply is

$$c' = m' \oplus E(IV) = c \oplus \alpha.$$

12. In this problem, we study a MAC scheme based on the CFB encryption mode. We consider a block cipher

$$E : \{0,1\}^{64} \times \{0,1\}^{64} \to \{0,1\}^{64},$$

where $E_k(x) = E(k,x)$ denotes the encryption of the plaintext x under the key k. The CFB-MAC of a given message $m \in \{0,1\}^*$ with the key k is obtained by first encrypting m with E_k using the CFB encryption mode and then combining the output blocks by XORing them together.
More precisely, for a message

$$m = x_l||x_2||\dots||x_n,$$

we have

$$\text{CFB-MAC}_k(m) = y_1 \oplus y_2 \oplus \ldots \oplus y_n,$$

where $y_i = E_k(y_{i-1}) \oplus x_i$ for $i = 2, \ldots, n$ and $y_1 = E_k(IV) \oplus x_1$, IV being an initialization vector. For the sake of simplicity, we assume that all messages have a length that is a multiple of 64 bits. We also assume in all the questions of this problem that IV is constant and known.

(a) Assume we have access to an oracle $\mathcal{O}$ that computes the CFB-MAC under a given secret key k and a fixed known IV. Show that you can recover $E_k(IV)$ by querying only one message to the oracle.

Solution : We can simply query a message x of one block to the oracle $\mathcal{O}$. The oracle returns the value

$$y = x \oplus E_k(IV).$$

Hence, $E_k(IV)$ is found by computing $x \oplus y$.

(b) Given a message m of n blocks and $h = \text{CFB-MAC}_k(m)$. Show how it is possible to generate a new message m' of n blocks and a $h' \in \{0, l\}^{64}$ such that $m' \neq m$ and $\text{CFB-MAC}_k(m') = h'$.

Solution : Let $m = x_1||x_2||\ldots||x_n$ and $h = \text{CFB-MAC}_k(m)$. We take another message $m' = x_1||x_2||\ldots||x_{n-1}||x'_n$, where x'_n is any block of 64 bits. Since CFB mode is used with a fixed IV the output blocks of m' will be identical to those of m except the last one.
Since

$$h' = \text{CFB-MAC}_k(m') = y_l \oplus \ldots \oplus y_{n-1} \oplus y'_n$$

and

$$h = y_l \oplus \ldots \oplus y_n,$$

we have

$$h \oplus h' = y \oplus y'_n.$$

We also know that

$$y_n = E_k(y_{n-1}) \oplus x_n$$

and

$$y'_n = E_k(y_{n-1}) \oplus x'_n.$$

Using these two relations, we finally deduce that

$$h' = h \oplus y_n \oplus y'_n = h \oplus x_n \oplus x'_n.$$

(c) Assume we are given IV, $E_k(IV)$, and a $h \in \{0,l\}^{64}$. Show how it is possible to generate a message m of two blocks, such that CFB-MAC$_k(m) = h$.

Solution : Set $m = x_l||x_2$ and $x_l = E_k(IV) \oplus IV$. We then have $y_l = IV$ and $y_2 = h \oplus IV$. Thus,

$$x_2 = E_k(y_1) \oplus y_2 = E_k(IV) \oplus IV \oplus h.$$

(d) Can we extend the attack of the previous question to messages m of more than two blocks ?

Solution : Yes, this works in the same way! Set

$$m = x_1||x_2||\ldots||x_n,$$

where

$$x_1 = x_2 = \ldots = x_{n-1} = Ek(IV) \oplus IV.$$

Hence,

$$y_1 = y_2 = \ldots = y_{n-1} = IV.$$

If n is even, then setting

$$x_n = h \oplus IV \oplus E_k(IV)$$

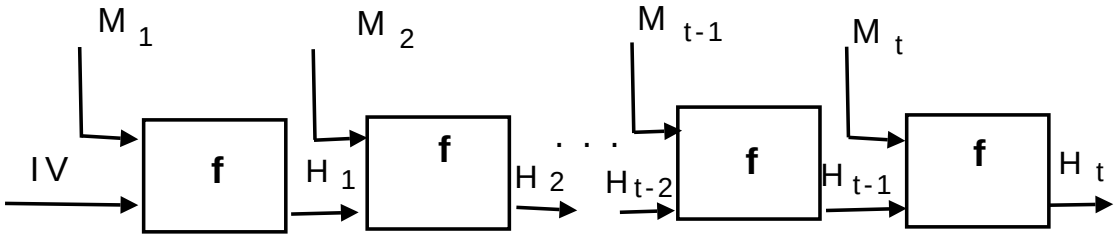
gives $y_n = h \oplus IV$ and thus

$$y_l \oplus \ldots \oplus y_n = h.$$

If n is odd, setting $x_n = h \oplus Ek(IV)$ gives $y_n = h$ and thus

$$y_1 \oplus \ldots \oplus y_n = h.$$

13. In this problem we consider the security of so-called compression function used in connection to the Merkle-Damgård construction given in the figure below.



a) Assume that

$$\mathbf{f} : \{0,1\}^m \to \{0,1\}^n, \;\; m > n,$$

that is m binary bits are compressed to n binary bits. Assume the function $\mathbf{f}$ is uniformly distributed, that is for each given image $f(x) = y$ there are 2^{m-n} input values $x \in \{0,1\}^m$ such that $f(x) = y$, i.e.,

$$\#\{x \in \{0,1\}^m : \mathbf{f}(x) = y\} = 2^{m-n},$$

for all $y \in \{0,1\}^n$.

The second preimage in the context of the compression function is defined as follows.

Given some input value x and its corresponding image $\mathbf{f}(x)$, the second preimage is any $x' \neq x$ such that $\mathbf{f}(x') = \mathbf{f}(x)$.

Compute the probability of finding the second preimage. Does it depend on m ?

Solution a) Given any x and its hash value $\mathbf{f}(x)$ there are 2^{m-n} inputs that map to the same value $\mathbf{f}(x)$. Hence the probability of finding second preimage is simply,

$$\frac{2^{m-n}}{2^m} = 2^{-n},$$

and it is independent of the input space.

b) The idea of the Merkle-Damgard construction is to split the message M into blocks of constant length

$$M = M_1||M_2|| \cdots M_{t-1}||M_t,$$

(each M_i is typically 512 bits) and to process these blocks along with the intermediate hash values $H_1, \ldots, H_{t-1}$ through the compression function $\mathbf{f}$. H_t is the hash value of M, that is $h(M) = H_t$.

Assume that for two messages

$$M = M_1||M_2|| \cdots M_{t-1}||M_t$$

and

$$M' = M'_1||M'_2|| \cdots M'_{t-1}||M'_t$$

one can find the inner collision of the type

$$H_i = \mathbf{f}(M_i, H_{i-1}) = H'_i = \mathbf{f}(M'_i, H'_{i-1}),$$

for some $1 \leq i < t$, $M_i \neq M'_i$, where by definition (see the figure) $H_0 = IV$.

Hence, the collision is not on the hash value H_t but on some intermediate value H_i, $i < t$. Using this construct

$$M'' = M''_1||M''_2||\cdots M''_{t-1}||M''_t$$

so that $h(M) = h(M'')$, that is find the second preimage of M.

Solution b)

$$M'' = M'_1||M'_2||\cdots||M'_i||M_{i+1}||\cdots||M_t.$$

c) Describe how you would find arbitrary collision, that is any M, M', $M \neq M'$ so that $h(M) = h(M')$ in time complexity much less than 2^n hash computations.

Solution c) Create $2^{n/2}$ different messages

$$M^1, \ldots, M^{2^{n/2}}$$

of same length (say all the messages have t blocks) and compute the hash values $h(M^i)$. Then the birthday paradox implies that with high probability there are $M^i \neq M^j$ such that

$$h(M^i) = h(M^j)$$

for some $1 \leq i < j \leq 2^{n/2}$.

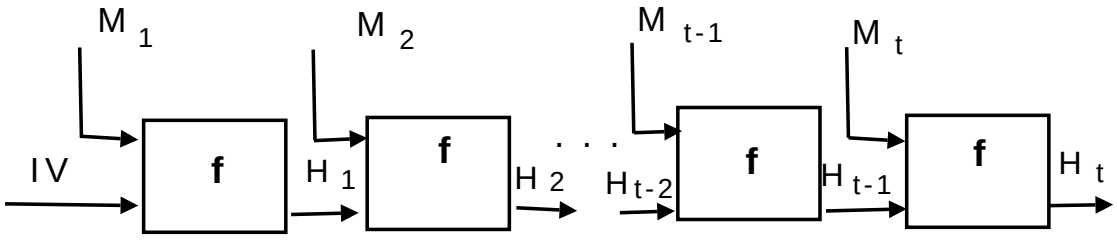
14. In this problem we explore the different ways of constructing a **MAC** out of a non-keyed hash function. Let $h : \{0,1\}^* \to \{0,1\}^n$ be a hash function constructed by iterating a collision resistant compression function using the Merkle-Damgård construction given in the figure below. The idea is to split the message M into blocks of constant length

$$M = M_1||M_2\cdots M_{t-1}||M_t$$

(each M_i is typically 512 bits) and to process these blocks along with the intermediate hash values

$$H_1, \ldots, H_{t-1}$$

through the compression function **f**. H_t is the hash value of M, that is $h(M) = H_t$.



a) Show that defining $MAC_k(M) = h(k||M)$ results in an insecure **MAC**. That is, show that given a valid text/MAC pair (M, H) one can efficiently construct another valid text/MAC pair (M', H') without knowing the key k. Assume for simplicity that the key length is the same as the length of the message block $|k| = |M_i|$.

Hint: The compression function **f** is of course known, that is given the two inputs anybody can compute its output.

b) Show that appending the secret key k, that is defining $MAC_k(M) = h(M||k)$ results in an insecure **MAC**. Recall that by definition the property of being collision resistant for MAC means that finding two messages $x \neq x'$ such that

$$MAC(x, k) = MAC(x', k),$$

implies the computational effort of 2^n operations, where n is the size of MAC (and hash). Describe an attack (based on the Merkle-Damgard structure above) that uses less than 2^n operations to create the MAC forgery, a legitimate MAC value for some message x' without revealing the key k.

Solution a) An adversary creates the message $M' = x||y$ so he needs to compute $MAC_k(k||x||y)$. But the iterative structure of Merkle-Damgard scheme implies that

$$MAC_k(k||x||y) = f(MAC_k(k||x), y).$$

So given the value $MAC_k(k||x)$ the adversary simply evaluates the publicly known compression function.

Solution b) Here processing of the message is not dependant on the key until the last block - we have birthday attack of course. That is, create $2^{n/2}$ different messages

$$M^1, \ldots, M^{2^{n/2}},$$

of same length (say all the messages have t blocks) and compute the hash values $h(M^i)$. Then the birthday paradox implies that there are $M^i \neq M^j$ such that $h(M^i) = h(M^j)$ for some $1 \leq i < j \leq 2^{n/2}$. Then,

$$MAC_k(M^i||k) = \mathbf{f}(h(M^i), k) = \mathbf{f}(h(M^j), k) = MAC_k(M^j||k).$$

The adversary may simply obtain the MAC value on message M^i, and then produce a correct text-MAC pair $(M^j, MAC_k(M^j||k))$.

15. Consider the following proposal for a cryptographic hash function, that makes use of a block cipher E with block size k bits to produce k bit hash values. The message to be hashed is split into a sequence $M_1 M_2 \cdots M_n$ of k bit blocks.

    For simplicity, we ignore padding and consider only messages whose length is a multiple of the block size. Hashing works as follows:

$$\begin{aligned} h_0 &= IV \\ h_i &= E_{M_i}(h_{i-1}), for \;\; i = 1, 2, \ldots, n. \end{aligned}$$

The hash of the message is h_n. Here IV is a fixed, public initialisation vector, which is part of the definition of the hash function.

Show that one can apply meet-in-the-middle attack (with birthday paradox) to find a preimage of the form $M||X||Y$ for a given hash value h.

Solution: As the first step we hash M and get, say, h^M. We want to produce a message of the form $M||X||Y$ with hash value h (preimage attack).

We procede as follows :

- we choose $2^{k/2}$ random blocks and for each such block X we compute $h_X = E_X(h^M)$.
- We store the pairs (h_X, X) in a hash table, indexed by h_X.

Then we start from the end, trying to meet in the middle:

- We choose again random blocks and for each such block Y we compute $h_Y = E_Y^{-1}(h^M)$ and try to look up h_Y in the hash table.
- If we find it, we have found the suitable X and Y and the attack has succeeded.

We would compute,

$$\begin{aligned} h(M||X||Y) &= E_Y(h_{n+1}) = E_Y(E_X(h^M)) = \\ &= E_Y(E_Y^{-1}(h^M)) = h^M. \end{aligned}$$

The birthday paradox tells us that the number of blocks Y that we have to try is in the order of $2^{k/2}$.
Time complexity is, around $2^{k/2}$ steps, where a step includes one encryption, one hash table insertion, one decryption and one hash table lookup. Space requirements for the hash table is $2^{k/2}$ pairs, where each pair is $2k$ bits.

16. Meet-in-the-middle attack seeks collisions on intermediate results, that is chaining variables, rather than the overall hash result. The attacker must be able to efficiently go backwords, for the attack to work. Show that the following hash scheme allows going backwards through chaining variables:

$$H_0 = IV, H_i = f(H_{i-1}, x_i) = E_{K_i}(H_{i-1}),$$

where H_i are intermediate hash values (chaining variables), and $K_i = x_i \oplus H_{i-1}$, x_i being the i-th message block.

Solution: Assumption is that we know the value H_{i+1}. Choose a fixed value

$$k_{i+1} = x_{i+1} \oplus H_i$$

and compute

$$H_i = D_{k_{i+1}}(H_{i+1}).$$

Finally choose the message block

$$x_{i+1} = k_{i+1} \oplus H_i.$$

17. Analyze the security of MASH-1 without feedforward, that is when

$$H_i = ((H_{i-1} + X_i) \vee A)^2 \pmod{N},$$

$A = 0xF00\ldots00$.

Solution: Without feedforward MASH-1 is not 2nd preimage resistant. Note that

$$C \vee A = C' \vee A$$

for any C, C' differing only in the first four bits (MSB). Therefore, given H_{i-1} and X_i one can compute $C = H_{i-1} + X_i$. Then select C' only differing from C in the first four bits and compute:

$$X_i' = C' - H_{i-1}.$$

The messages $X_1X_2\cdots X_i$ and $X_1X_2\cdots X_i'$ will then collide.

18. Consider the MAC scheme based on a block cipher E_k of block size n and a collision resistant hash function h with output size n as well. The MAC generation works as follows:

    (a) For any message m of size $N > n$, the MAC is computed as $\mathsf{MAC}(m) = E_k(h(m))$

    (b) For the messages of size n exactly compute $\mathsf{MAC}(m) = E_k(m)$.

    a) Show that this MAC scheme is not secure (find an efficient forgery)

Solution a) Take an arbitrary message $m \in \{0,1\}^{2n}$ so that $\mathsf{MAC}(m) = E_k(h(m))$. Now taking $m' = h(m)$ of length n gives,

$$\mathsf{MAC}(m') = E_k(m') = E_k(h(m)) = \mathsf{MAC}(m).$$

To make the scheme more secure it suffices to modify part 2 so that even short messages are hashed as well, that is,

$$\mathsf{MAC}(m) = E_k(h(m)) \text{ for } m \in \{0,1\}^n$$

19. Consider the keyed MAC (Message Authentication Code) scheme based on a block cipher E_{key} of block size n and a collision resistant hash function h with output size n as well. The MAC on message m is computed as

$$\mathsf{MAC}(m) = E_{h(m)}(k).$$

Note that the standard usage mode is interchanged: hashed value (message tag) is used as the encryption key whereas the secret key is a message input to symmetric-key encryption algorithm.

a) Show that this MAC scheme is not secure (find an efficient forgery). Hint : Recover the key from the MAC.

Solution a) Given a pair

$$(m, MAC(m)) = (m, E_{h(m)}(k))$$

one can compute $h(m)$ (h is public hash algorithm) and then to decrypt $E_{h(m)}(k))$ to obtain k.

b) Generalize the result above by showing that adversary is capable of forging any message.

Solution b) Since key k is known the attacker can compute MAC for any message as $(m', E_{h(m')}(k))$.

20. This problem considers some standard security flaws related to the design of hash functions.

a) Show that defining

$$MAC_k(M) = h(k||M)$$

results in an insecure **MAC**. That is, show that given a valid text/MAC pair (M, H) one can efficiently construct another valid text/MAC pair (M', H') without knowing the key k. Assume for simplicity that the key length is the same as the length of the message block $|k| = |M_i|$.

Solution a) An adversary creates the message $M' = x||y$ so he needs to compute $MAC_k(k||x||y)$.

But the iterative structure of M-D scheme implies that

$$MAC_k(k||x||y) = f(MAC_k(k||x), y).$$

So given the value $MAC_k(k||x)$ the adversary simply evaluates the publicly known compression function.

b) Show that appending the secret key k, that is defining

$$MAC_k(M) = h(M||k)$$

results in an insecure **MAC**. Recall that by definition the property of being collision resistant for MAC means that finding two messages $x \neq x'$ such that $MAC(x,k) = MAC(x',k)$ implies the computational effort of 2^n operations, where n is the size of MAC (and hash).

Solution b) Here processing of the message is not dependant on the key until the last block - we have birthday attack of course. That is, create $2^{n/2}$ different messages

$$M^1, \ldots, M^{2^{n/2}}$$

of same length (say all the messages have t blocks) and compute the hash values $h(M^i)$. Then the birthday paradox implies that there are $M^i \neq M^j$ such that $h(M^i) = h(M^j)$ for some $1 \leq i < j \leq 2^{n/2}$. Then,

$$\begin{aligned} MAC_k(M^i||k) &= \mathbf{f}(h(M^i),k) = \mathbf{f}(h(M^j),k) \\ &= MAC_k(M^j||k). \end{aligned}$$

The adversary may simply obtain the MAC value on message M^i, and then produce a correct text-MAC pair $(M^j, MAC_k(M^j||k))$.

21. In this problem we will consider a signature scheme that once was proposed as a more efficient alternative to RSA. The setting is as follows. User Alice chooses two large secret primes p and q. We put $N = pq$ as in RSA. She also chooses an element $g \in \mathbb{Z}_N^*$. that generates a subgroup of prime order r. Alices public key is (N, g); her private key is r. Here p and q should be big enough to make factoring of N infeasible, i.e., at least 1024 bits. Further, r should be big enough to make the discrete log problem for the subgroup infeasible; hence r could be a 160 bit number. The community of users have also agreed on a hash function H. To sign message m, Alice first hashes m and computes x such that

$$x \cdot H(m) = 1 \pmod{r}.$$

The signature is then

$$s = gx \pmod{N}.$$

The intended advantage of the scheme is that $x < r$, so the exponent is much smaller than for RSA signatures.

a) How will Bob, after receiving (s, m), verify that s is Alices signature on m ? Show

that a correct signature will be verified.

Solution a) This only requires straightforward computation and is left to the reader.

b) The proposed scheme has several vulnerabilities and is completely broken. We will now demonstrate some of the problems with the scheme. i.) Show that r is a divisor of at least one of $p1$ or $q1$. ii. Show that, if r is a divisor of $p1$ but not of $q1$, then one can factor N, using only the public key.

Hint: Show that $g \pmod{q} = 1$.

Solution b) This problem is also left as an exercise for the reader.

22. (Improper combination of CBC-MAC and CBC encryption) Consider using the data integrity mechanism given by,

$$C' = E_k(x||h_{k'}(x)),$$

with:

- E_k being CBC-encryption with key k and initialization vector IV ,
- $h_{k'}(x)$ being CBC-MAC with k' and IV',
- and $k = k', IV = IV'$.

Show that this scheme fails to provide the message integrity.

Solution: The scheme provides both authenticity (via MAC) and secrecy via encryption with key k.

The usual procedure when $k \neq k'$ is to produce a MAC for $x_1, x_2, \ldots, x_t$ which per definition is c_t where c_t is obtained by encrypting $x_1, x_2, \ldots, x_t$ in the CBC mode with key k',

$$c_i = E_{k'}(c_{i-1} \oplus x_i) \;\; 1 \leq i \leq t; \;\; c_0 = IV.$$

Then one should encrypt the sequence

$$x_1, x_2, \ldots, x_t, x_{t+1} = c_t$$

with the key k again using the CBC mode.
Since the keys are same the data $x = x_1 x_2 \ldots x_t$ can then be processed in a single CBC pass.

The CBC-MAC is equal to the last ciphertext block

$$c_t = E_k(c_{t-1} \oplus x_t); \; 1 \leq i \leq t$$

, and the last data block is $x_{t+1} = c_t$. This gives a final ciphertext block

$$c_{t+1} = E_k(c_t \oplus x_{t+1}) = E_k(0)$$

The encrypted MAC is thus independent of both plaintext and ciphertext, rendering the integrity mechanism completely insecure.

23. One may consider the MAC (Message Authentication Code) scheme based on a block cipher E_k of block size n and a collision resistant hash function h with output size n as well. The MAC generation works as follows:

    (a) For any message m of size $N > n$, the MAC is computed as $\mathsf{MAC}(m) = E_k(h(m))$

    (b) For the messages of size n exactly compute $\mathsf{MAC}(m) = E_k(m)$.

    a) Show that this MAC scheme is not secure (find an efficient forgery)
    Hint: A collision is found for one message of length $2n$ and one of length n.

    b) Propose a small modification which would make the scheme more secure. That is, no trivial attacks as in a) should be possible.

    Solution a) Take an arbitrary message $m \in \{0,1\}^{2n}$ so that $\mathsf{MAC}(m) = E_k(h(m))$. Now taking $m' = h(m)$ of length n gives,

    $$\mathsf{MAC}(m') = E_k(m') = E_k(h(m)) = \mathsf{MAC}(m).$$

    b) It suffices to modify part b) so that even short messages are hashed as well, that is, $\mathsf{MAC}(m) = E_k(h(m))$ for $m \in \{0,1\}^n$.

24. Consider the keyed MAC (Message Authentication Code) scheme based on a block cipher E_{key} of block size n and a collision resistant hash function h with output size n as well. The MAC on message m is computed as

    $$\mathsf{MAC}(m) = E_{h(m)}(k).$$

    Note that the standard usage mode is interchanged: hashed value (message tag) is used as the encryption key whereas the secret key is a message input to symmetric-key encryption algorithm.

    a) Show that this MAC scheme is not secure (find an efficient forgery). Hint : Recover the key from the MAC.

    b) Generalize the result above by showing that adversary is capable of forging any

message.

Solution a) Given a pair $(m, MAC(m)) = (m, E_{h(m)}(k))$ one can compute $h(m)$ (h is a public hash algorithm) and then to decrypt $E_{h(m)}(k))$ to obtain k.

b) Since key k is known the attacker can compute MAC for any message as $(m', E_{h(m')}(k))$.

25. The symmetric-key cryptosystems may be used for generation of message authentication codes (MAC). This however requires a careful use of the underlying algorithm. The message m is split into l blocks of length n each, i.e., $m = m_1 m_2 \dots m_l$ and the MAC(m) is generated by repeatedly applying encryption algorithm $E_k()$ with key length n as follows.

$$h_i = E_{m_i}(h_{i-1}), \text{ for } 1 < i \leq l.$$

The initial hash value $h_1 = m_1$, and h_l is the message digest (MAC) on message m, i.e. $h(m) = h_l$. Find the collisions (at least two messages) for the symmetric cryptosystems specified below.

a) Let E be one-time pad defined by $E_k(m) = k \oplus m$. Note that h_i's are generated using the above rule, i.e. m_i is considered to be the encryption key in step i.

b) Let E be the DES encryption algorithm, that is $h_i = E_k(h_{i-1})$ and $h_1 = m_1$.
Hint: You choose a suitable l (small l of course) to demonstrate the collisions. In addition, use the fact that DES has weak keys for which $E_k(m) = m$.

Solution a) Let E be one-time pad defined by $E_k(m) = k \oplus m$. Note that for $h(m_1, m_2)$ we have $h_1 = m_1$ and $h_2 = m_2 \oplus h_1$. Thus,

$$h(m_1, m_2) = m_1 \oplus m_2.$$

A collision is found for any two strings

$$(m_1, m_2) \neq (m_1', m_2'),$$

such that $m_1 \oplus m_2 = m_1' \oplus m_2'$.

b) Note that DES has weak keys k for which $E_k(E_k(m)) = m$. In particular, this is true for $k = 0^{56}$ and $k = 1^{56}$. This implies that for

$$m = (x, 0^{56}, 0^{56}),$$

and

$$m' = (x, 1^{56}, 1^{56}),$$

it holds that

$$h(m) = h(m') = x.$$

26. Consider the following (flawed) authentication protocol, where s_A denotes the signature operation of party A , and it is assumed that all parties have authentic copies of all others' public keys.

$$\begin{array}{lcll} A & & B & \\ \rightarrow & r_A & & (1) \\ & r_B, s_B(r_B, r_A, A) & \leftarrow & (2) \\ \rightarrow & r_{A'}, s_A(r_{A'}, r_B, B) & & (3) \end{array}$$

The intention is that the random numbers chosen by A and B , respectively, together with the signatures, provide a guarantee of freshness and entity authentication. However, an enemy E can initiate one protocol with B (pretending to be A), and another with A (pretending to be B), to successfully deceive B into believing E is A (and that A initiated the protocol).

a) Provide the details of the attack by writing down the sent messages in the same form as for the original protocol above. Note that in the attack a few more messages have to be exchanged.

b) Propose a simple modification of the above protocol to prevent from this attack.

Solution a) The attack is performed by exchanging the following messages:

$$\begin{array}{lcccl} A & & E & & B \\ & & \rightarrow & r_A & (1) \\ & & & r_B, s_B(r_B, r_A, A) \quad \leftarrow & (2) \\ & r_B & \leftarrow & & (1') \\ \rightarrow & r_{A'}, s_A(r_{A'}, r_B, B) & & & (2') \\ & & \rightarrow & r_{A'}, s_A(r_{A'}, r_B, B) & (3) \end{array}$$

At the end of the protocol run B is decieved into believing E is A (and that A initiated the protocol).

b) Use the same r_A in the step (3) of the original protocol, that is $r_A = r_{A'}$.

27. Meet-in-the-middle attack seek collisions on intermediate results, that is chaining variables, rather than the overall hash result. The attacker must be able to efficiently go backwords, for the attack to work. Show that the following hash scheme allows going backwards through chaining variables:

$$H_0 = IV, H_i = f(H_{i-1}, x_i) = E_{K_i}(H_{i-1}),$$

where H_i are intermediate hash values (chaining variables), and $K_i = x_i + H_{i-1}$, x_i being the i-th message block.

Solution: The assumption is that we know the value H_{i+1}. Choose a fixed value

$$k_{i+1} = x_{i+1} + H_i,$$

and compute

$$H_i = D_{k_{i+1}}(H_{i+1}).$$

Finally choose the message block

$$x_{i+1} = k_{i+1} \oplus H_i.$$