Solutions, exercises, set 1

Computer Security

Exercise 1.1

As preventative measures, one would check id and authentication of everybody wanting to use (log in) to the application. All the communication between the clients and the server should be encrypted and authenticated. Different user levels could give different access levels, although it might be appreciated with some sort of override system for emergencies.

Detection means finding out when someone has broken into the system and what they have done, but also when a legitimate user has been overriding their rights to look at something not intended for them.

To recover, we need good and frequent backups to be able to restore the data after a technical problem or an illwilling person. Broadening the scope, we might react to a system misuse by taking legal actions against the suspect data thieves or "curious George".

Exercise 1.2

a) With 2^n possible hash values we need to try, according to the birthday paradox, $2^{n/2}$ different inputs before we find a collision. Using this fact we can create about $2^{n/2}$ "evil" documents and $2^{n/2}$ "good" documents. This gives roughly $2^{n/2}$ documents in total and it is likely that two hash values will collide. With probabilty $1/2$ the collision will be between a "good" and an "evil" document. We let the victim sign the good document. After that we move the signature to the evil document since it is valid for both.

Some may find it more intuitive to see this attack from a different point of view: With $2^{n/2}$ "evil" documents, we cover a fraction $2^{-n/2}$ of the total number of hash values. Any newly created hash value has a probability $2^{-n/2}$ of colliding with one of the hash values from the "evil" documents. Thus, the expected number of "good" documents needed is geometrically distributed with an expected value of $2^{n/2}$.

This attack gives a lower bound on n when creating a hash function.

b) Since SHA-1 produces hash values consisting of $n = 160$ bits, we need to create about 2^{80} contracts during our attack.

Exercise 1.3

Let's use some small numbers. The hint suggests that $q = 23$ and $p = 2q + 1 = 47$ is a good choice. Choosing $\alpha = 2$ gives $g = 4$. For the private key, let's try $a = 13$. It yields $y = 4^{13} \bmod p = 8$.

(If you have trouble calculating $4^{13} \bmod p$ by hand, break it up: $4^2 \bmod p = 16$, $4^4 \bmod p = 21$ and $4^8 \bmod p = 18$. This gives $4^{13} \bmod p = 4^{8+4+1} \bmod p = 4^8 4^4 4 \bmod p = 8$. This way, all you need to do are some squarings and some multiplications modulo p where all calculations will be doable by hand.)

We are now ready to publish our public key $(p, q, g, y) = (47, 23, 4, 8)$ while keeping our private key $a = 13$ to ourselves. We sign some message m with

$h = h(m) = 11$. Choosing $k = 3$ we find $r = (4^3 \bmod p) \bmod q = 17$.

We need to find $k^{-1} = 3^{-1} \bmod q$ and might find ourselves tempted to claim that $3^{-1} \bmod q = \frac{1}{3}$. This is however not true as we only have the integers to play with. In short, we need to find *an integer* such that $3 \cdot 3^{-1} \bmod q = 1$. The interested reader may read up on Euclid's algorithm, but we can resort to a brute force search or some thinking. Anyway, we soon realize that $3 \cdot 8 = 24 \equiv 1 \pmod{23}$ so $k^{-1} \bmod q = 8$.

Using this, we find $s = 8(11 + 13 \cdot 17) \bmod q = 16$ and the signature is published as $(r, s) = (17, 16)$.

To verify the signature we do the following (note that a is not needed). $w = s^{-1} \bmod q = 13$ by brute force and we get the two[1] help variables as $u_1 = wh \bmod q = 5$ and $u_2 = rw \bmod q = 14$. Using square and multiply as above, we find that $8^2 \bmod p = 17$, $8^4 \bmod p = 7$, $8^8 \bmod p = 2$ and thus $8^{14} \bmod p = 3$. $4^5 \bmod q = 37$ is simple enough to find at once and we finally arrive at $v = (4^5 8^{14} \bmod p) \bmod q = 17 = r$ which verifies the signature.

Exercise 1.4

The private key is (p, q, e) and the public one is (n, d) where we assume that you know the relations between, and properties of, these numbers. Let's say an attacker acquires two messages and their corresponding signatures. Thus, she has s_i, m_i such that $s_i^d \equiv m_i \pmod{n}$, $i = 1, 2$ (this is the check someone would do to verify the signatures). What happens if the attacker publishes $(m, s) = (m_1 m_2 \bmod n, s_1 s_2 \bmod n)$? An attempt to verify the signature would give

$$\begin{aligned} s^d \bmod n &= (s_1 s_2 \bmod n)^d \bmod n \\ &= s_1^d \bmod n \cdot s_2^d \bmod n \\ &= m_1 m_2 \bmod n = m \bmod n, \end{aligned}$$

tricking the reciever into accepting the signature! Convince yourself that the use of a hash function means the attacker needs to perform a preimage attack on the hash function in order to utilize the above attack idea.

Exercise 1.5

a) Two blocks that are identical will be encrypted the same. This preserves some of the structure in the plaintext.

b) Since two identical blocks will only be encrypted into the same cryptotext if the previous blocks of cryptotext match, there is a very small probability of such a "collision".

c) Together with an IV, a counter is encrypted in order to create a stream cipher-like application where the plaintext is XOR-ed with the output of the block cipher. The key is the same for all invocations of the block cipher. Without the IV and always restarting the counter at 0 for a new message, we would reproduce keystream used for a previous plaintext, which is bad.

Exercise 1.6

Let's study this system from Eve's point of view: She first sees

$$X = M + S_A$$

flying by from Alice to Bob. Then Bob responds with

$$Y = X + S_B = M + S_A + S_B$$

[1] There's a typo in the book here. There are of course two variables u_1 and u_2.

before the last transmission,

$$Z = Y + S_A = M + S_B.$$

Since the scheme is entirely based on XOR, Eve might start with trying just that using some of the variables she has at hand:

$$X + Y = M + S_A + M + S_A + S_B = S_B.$$

Oops, she now has Bob's secret key! This means she can find

$$Z + S_B = M + S_B + S_B = M.$$

So it seems as if all Eve needs to do is add up all three messages on the channel:

$$X + Y + Z = M + S_A + M + S_A + S_B + M + S_B = M.$$

Eve will be able to decrypt the transmission in real-time in all senses of the word since she can do it just as fast as Bob. Clearly, there is a problem with this protocol.

Exercise 1.7
a) With $p = 13$ and $q = 17$, we get $n = 11 \cdot 17 = 187$ and $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1) = 160$. We compute d such that $e \cdot d \equiv 1 \bmod 160 \rightarrow e \cdot d = 1 + k \cdot 160$. With $k = 5$ we get $d = 801/9 = 89$.
b) To encrypt we compute $14^9 \bmod 187 \equiv 14^{8+1} \bmod 187$. We can use the fact that $14^{2x} \bmod n = (14^x \bmod n)(14^x \bmod n)$ and construct the following table:
$14^1 \equiv 14 \bmod 187$
$14^2 \equiv 196 \equiv 9 \bmod 187$
$14^4 \equiv 9^2 \equiv 81 \bmod 187$
$14^8 \equiv 81^2 \equiv 16 \bmod 187$
So $14^9 \equiv 14 \cdot 16 \equiv 37 \bmod 187$. The encryption of $m = 14$ is given by $c = 37$.

Exercise 1.8
A MAC takes a key as input and only the parties with knowledge of the key will be able to compute the MAC. Thus it will protect against both accidental and intentional changes to the message. A hash value does not take a secret key as input and anyone with knowledge of the algorithm will be able to calculate the hash of a message. Thus it will protect against accidental changes but not intentional changes.

Exercise 1.9
In a Merkle-Damgård construction the internal variable h_i is computed as $h_i = f(x_i \| h_{i-1})$ and the MAC is given by $MAC(x) = h_m$. The MAC of the message $x = x_1 \dots x_m$ is the hash of the message $k, x_1, \dots, x_m$. If we know this, we can find the MAC of another message $x' = x_1 \dots x_m x_{m+1}$ as $f(x_{m+1} \| MAC(x))$. (By Kerckhoffs' principle we assume that the function $f(\cdot)$ is known.)