4^{th} Assignment Information Security

Francesco Segala 3521885 Manos Gionanidis 3542068

October 10, 2017

Excercise 18

A sha-256 hashing function fo a document D compute a hash value of length 256 bit, so we have 2^{256} different permutation in wicht the hash value can be output once we compute h(D) .
Now we want to know how much time should we wait untill there is a 100% probability that we occurr in a collision if everyone in the world produce and sign a document every day (7*10^9 humans $\approxeq 10^{10}$).
Let's work with this assumption, now we have to compute $2^{256}+1$ different documents, (assuming the perfect hash function). 2^{256} different messages $\approxeq 10^{85}$ different messages $\approxeq \frac{10^{85}}{10^{10}} = 10^{75}$ days needed to compute all this different documents, that are $3 * 10^{72}$ years.
Now we want to know how many document D we have to compute such that there is an high probability (not 100%) to have a collision.
For this purpose we start from the Birthday paradox and apply the insight to our problem. So we know from the Birthday paradox that with N = 23 people we have a probability higher than 50% that 2 of them are born in the same day, so we stated with this one that in a hashing space of 365 we need $\approxeq$ 23 elements to have an high probability of collision. From this we can easily see that for our problem we need much less documents Ds to be confident in finding a collision.
Generalizing this problem we have that the probability to have a collision is equal to 1 - the probability to have no collision , so given a space of N=2^{256} possible hash values and P=#peopele-in-the-world * D (supposed different) the second day we should pick N-P hash values and so on.
In general, the probability of randomly generating P documents every day that are all unique is: $\frac{N-1}{N} * \frac{N-2}{N} * \frac{N-3}{N} * .... * \frac{N-(P-1)}{N}$.
The above expression is approximable to $e^{\frac{-P(P-1)}{2N}}$. A more easy approximation is made up by seeing that every document should be different to each other so every day we need P(P-1)/2 $\approxeq P^2$ comparison. It follows that $\sqrt{N = 2^{256}} = 2^{128}$ document in input leads to an high probabilty of collision, we have that $2^{128} \approxeq 10^{43}, 7, 6x10^9 \approxeq 10^{10}$ so we need $\frac{10^{43}}{10^{10}} = 10^{33}$ days $= 3 * 10^{30}$ years to generate such a number of different documents.

Excercise 19

Exercise 8 chapter 5, section 5.11; Stamp (2011, p. 155):
Message X=10101011
Divisor d=10011
X1, X2 alternative message that produces the same CRC:
X1=01011001
X2=11000001
The steps we did to complete this excercise are the same explained in the textbook and in the lecture's slides, infact we've just modified the first 3 bit of the message at our pleasure and then wisely choose the other bit in such a way that the same CRC=1010 shows up following the CRC calculation protocol.
Exercise 9 chapter 5, section 5.11; Stamp (2011, p. 155):
This excercise is just a generalization of the previous one, the result is: 11100011 and 11110000 . Here is the code for this excercise:

```
#!/usr/bin/env python3
import itertools
import zlib

def comb_calc_bit(num):
    tuples = list(itertools.product([0, 1], repeat=num))
    res=[]
    for elem in tuples:
        elem = str(elem)[1:-1]
        elem = "".join(elem.split(", "))
        res.append(elem)
    return res

def crc(message, div):
    pad='0'*(len(div)-1)
    message = message + pad
    message = list(message)
    div = list(div)
    for i in range(len(message)-len(pad)):
        if message[i] == '1':
            for j in range(len(div)):
                message[i+j] = str((int(message[i+j])^int(div[j])))
    return ''.join(message[-len(pad):])

def calculate_all_collision(message,div,prefixList=None):
    final_crc=crc(message,div)
    comb5=comb_calc_bit(len(div))
```

```
    curr_msg=""
    res=[]
    if prefixList==None:
        prefixList=comb_calc_bit(len(message)-len(div))
    for prefix in prefixList:
        for calc_bit in comb5:
            curr_msg=prefix+calc_bit
            if crc(curr_msg,div) == final_crc:
                res.append(curr_msg)
    return res

print calculate_all_collision("11010110","10011",["111"])
```

Excercise 20

The checksum is not a security guard as stated in the website. That form of checksum guarantees only that the file has not been atered during the download. This procedure is wrong because if a cracker is able to modify 1 of this files, he is also able to corrupt both of them , or to do a MITM attack if the checksum is distributed as a separate file. This procedure do not offer a safeguard for those reasons.
Another step to increase the security would be let the file to be downloaded as the previous scenario , but distribute the chechsum via a secure channel (like SSL over HTTPS for example) on a different server. This would ensure the integrity and the validity of the checksum file (you can also post the checksum on the website for double checking) and so you can easily check the downloaded file.
However this is not the final step for ensure security, infact you should prefer to send that checksum file encrypted by the reciever public key, this guarantees on the integrity of such file for sure.

Excercise 21

We also did excercise 21 using SMTP protocol to write an email to the professor sent by himself. We already got the point for this excercise so we suppose the explanation is not required here.

Excercise 22

Unfortunately we did not found the insight of how the message was embedded in the picture.

Excercise 23

This is the HMAC we computed for the file that the TA sent to us using the openssl software via command line and the secret "Daniel Riciardo" as:
`$ openssl dgst -sha256 -hex -hmac "Daniel Riciardo" ./segala.plain`
, digest: 48083b874b6525275f801dd8032c206e3797366865fc83c96d75a6689e0$dece$2
To double check this result we also tried some online services to do this task but surprisingly we came up with different HMAC values :
4fc28663069cc613a764e5229723e3be699b09c52f2dd3aa4c22f4d654325a2d
this is from https://www.freeformatter.com/hmac-generator.html , or even
cf505c254500c8e54db22b36ea09a8948003f70d7e173cae78e10a5a1ab51146
from https://hash.online-convert.com/sha256-generator that takes the file as input.

Excercise 24

Trying to adjust this problem with the birthday paradox problem.
the number of the people on earth is n $=7.5 * 10^9$ we define event {A} as the event of finding 2 PGP keys with the same value. It is easier to find the possibility that all the people have different keys so P(A'). But we want P(A) so we find P(A) = 1 - P(A')

So we are trying to compute P(A') = $(1/k)^n$, generalized
as n we have the number of the population of earth n = 7.5*10^9 as k we have the possible PGP fingerprints. Analytically the PGP fingerprint consist of charachters(A-F) and numbers(0-9) so we have 6 charachters and 10 numbers as a total of 16. So the possible PGP fingerprints are $(16)^{40}$
Assume $\varphi = 16^{40}$:

P(A') = (1 / (φ)) * (φ/n) * (φ-1 / n) * (φ-2 / n) *.....(φ-n / n) = $\frac{1}{\varphi}^n$ * (φ * φ-1 * φ-2 *....* φ-n)

We can continue the calculations:
P(A') = 1* (1-1/φ) * (1-2/φ) *.....* (1- n-1/ φ) = (φ * φ-1 * φ-2 *.....* φ-n-1) / (φ^n) = (φ)! / (φ^n) * (φ-n)!
This is the generalized mathematical type. We tried to make the calculations and the P(A') bearly touched the 1 so the result was a little above of zero. But because we had really big number as a dividors and it is now when you do number/(number which goes to infinity) = 0. Our results was only zero so we made the maths with the traditional way with pen and paper.

We have 40 charachters and every hecademical charachter consists from 4 bytes . Because of this we have 40 * 4 = 160

The combinations of the PGP figerprints are 2^{160} and we want at least two people have the same fingerprint(collision) so the number of key to compute to have an high probability of finding a collision is $(2^{160})^{1/2} = 2^{80}$. We tried to round the numbers here and just to make an approximate solution. We calculate that $2^{80} \simeq 10^{25}$ and $7,5 * 10^9 \simeq 10 * 10^9 = 10^{10}$ and we use these numbers.

P(A) = number of people in the earth / possible PGP fingerprints = $(10^{10})/(10^{25}) = 10^{-15}$ this number is so close to zero as we expected. This is an informal approach . But the point is that the possibility touches to zero and the possibility of no collision touches the 1.