Cryptography and Security of Digital Devices

Exercise book

Alessandro Barenghi, Gerardo Pelosi

February 1, 2018- Rev 1.4.2

Contents

Chapter 1

Historical Ciphers and Basic Principles

Cryptanalyzing historical ciphers allows to see clearly what does it happen whenever pre-Kerchoff principle cryptographic schemes are scrutinized with full knowledge of their structure. More in detail, we will be breaking a couple of ciphers employing a known ciphertext only attack: the weakest possible condition for an attacker, sometimes despite the fact that the cipher keyspace is not negligible.

1.1 Substitution Ciphers

1.1.1 Distinguishing ciphers

Examining the frequency of the single letters in an encrypted plaintext yields the results depicted in Figure 1.1. From the results of the frequency analysis, what can you infer on the following points:

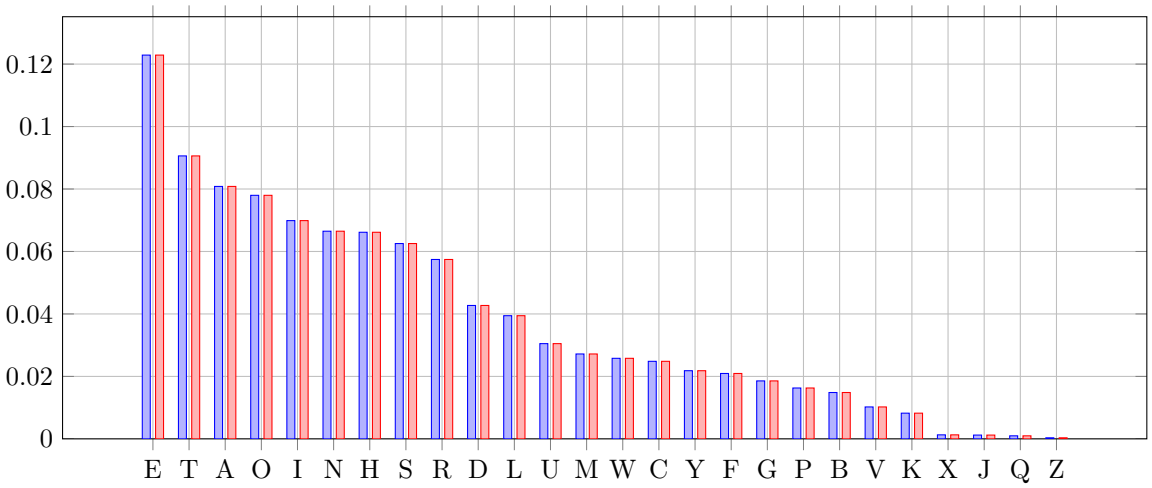


Figure 1.1: Letter frequencies of a large English text (the actual letter is employed as the x-axis label), and ciphertext.

1. Which is the plaintext language?
2. Is the employed cipher a substitution cipher of some kind?
3. Does the employed cipher include a permutation?

Solution:

1. The letter frequencies match perfectly the ones of plain English: it is quite likely that the plaintext is expressed in it.
2. The match between the two histograms suggests a monoalphabetic substitution to be the technique in use.
3. It is not possible to infer whether or not the cipher includes a permutation from a single-letter frequency analysis, as permutations leave this quantity unaltered.

1.1.2 Breaking Vigenére

The polyalphabetic substitution cipher known as the Vigenére cipher was invented by Giovan Battista Bellaso back in 1553, and later misattributed to Vigenére.
The scheme exploits a keyword to define a set of shift ciphers which are applied to the plaintext depending on the plaintext letter position. This is done mapping each letter of the alphabet to a shift amount, and applying the circular shifts in the same order of the key letters, to the plaintext. The shifts are repeated enough times to get the whole plaintext mapped into ciphertext. As a simple running example, given `HAL` as a plaintext, and `BBB` as the key, the resulting shift amounts are 1-1-1, thus yielding `IBM` as ciphertext. The main advantage of the Vigenére cipher is a flattening effect on the frequency histogram of the ciphertext letters, as it can be seen in Figure 1.2. The flattening effect is more and more evident as the length of the cipher key grows.

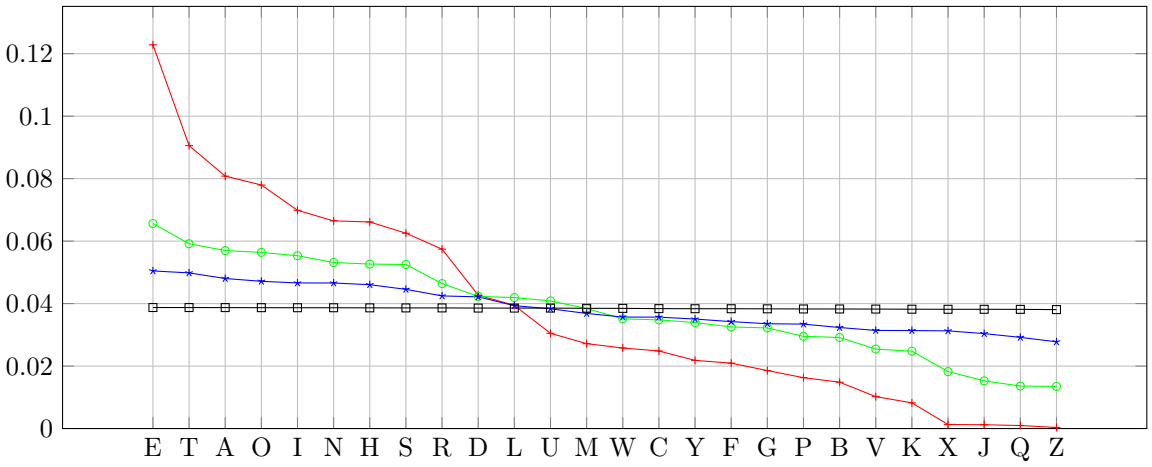


Figure 1.2: Letter Frequencies of different encrypted texts computed on a ≈450k characters ciphertext: Monoalphabetic Substitution —+—, Vigenére w/ 6 letters key —○—,Vigenére w/ 30 letters key —⋆—, One-Time-Pad —□—

The Vigenére cipher can be broken observing that, the ciphertext characters in sharing the same position modulo the key length are actually shifted by the same amount. This in turn leads to the possibility of finding out the key length via Kasiski examination, i.e. looking for repeated patterns in the ciphertext, and computing the greatest common divisor of their distances.
This is rather evident in the following example, keyed with `abcde`

```
PTX: cryptodoesreallymeancryptography
KEY: abcdeabcdeabcdeabcdeabcdeabcdeab
CTX: csasxoeqhwrfcopyngdrcsasxohtdthz
```

Where the distance between the two ciphertext `csasxo` strings is 20, a multiple of the key length.

Assuming that the results of a Kasiski examination on a Vigenére ciphertext report that the three most frequent distances at which repeated patterns occour are $6, 24, 30$.

- Which are the plausible lengths for the key?
- Considering Vigenére ciphertexts obtained encrypting English messages, (assume the entropy of a single English symbol to be 1.5b) with a key generated as a random sequence of letters, compute the unicity distance for the ciphertext.
- Repeat the calculations in case the key is a single English word, recalling that the English dictionary is about 2^{17} words wide, and in case the key is 5 random English words concatenated together.

Solution:

- Computing the gcd among and 6,24 and 30 yields 6, which is the most plausible length for the key. However, note that key lengths of 2 and 3 are still plausible, as they do not contradict the results of the analysis.
- Considering the entropy of a single english symbol to be 1.5b, we have that the language redundancy of English R_{eng} is $R_{eng} = 1 - \frac{1.5}{\log_2(26)} = 0.681$. As a consequence, the unicity distance for a Vigenére encrypted text is $n_0 = \frac{\log_2 26^6}{0.681 \log_2(26)} = 8.81$. This in turn implies that a 9 letter ciphertext is long enough to be able to tell apart which one is the correct Vigenére key examining the plaintext obtained with it.
- In the former case, the unicity distance becomes $n_0 = \frac{\log_2 2^{17}}{0.681 \log_2(26)} = 5.31$ showing that it is easier to tell which key is the correct one if the keyword is a random english word, with respect to a 6 random character string. The unicity distance of the latter case can be easily derived to be 5 times larger than the former, since the only changing quantity is the keyspace, yielding a distance of 26.55

Assume to be encrypting a message for the SMS message system (which, we recall, is 160 characters wide), with the Vigenére cipher, picking a random 256 characters key. Discuss the security of the system against a known ciphertext attack, assuming that:

1. A fresh key is used for each SMS
2. The same key is reused for all SMS

Solution:

1. Since the encryption is employing a new key, selected fully independently from the plaintext, and at least as long as the plaintext, and combines it with a bijective function (addition modulo 26), this encryption scheme is perfectly secure.
2. This is a simple instance of a Vigenére cipher, which can be routinely broken once enough ciphertext material is gathered, so to perform the usual frequency analysis. Note that the effective key length in this case is 160, as the last characters of the key are never used.
   As a side note, a similar, 256 B key Vigenére cipher was employed (and broken) to protect the confidentiality of Pentax DSLRs Firmware. Note that raw exhaustive search would have definitely been out of reach: the number of possible 256 B values is actually $2^{8\times256} = 2^{2048}$.

Assume to be attacking Vigenére with stronger assumptions on the attacker model. Can you do something better than frequency analysis assuming :

1. a chosen plaintext attack?
2. a known plaintext attack?

Solution:

1. Yes. It is possible to extract the key deterministically, supplying a long string of `A`s to the encryption oracle. The oracle will output the key, possibly repeated one or more times as the ciphertext. To confirm the correct key extraction, the plaintext should be at least twice as long as the key, so that it is possible to detect the periodicity in the ciphertext.
2. Again, yes. The strategy is the result of a modification of the previous one, observing that there is no need of having a chosen plaintext to derive the shift amounts caused by the encryption. In particular, computing the shift amounts letter-by-letter on paired ptx-ctx pairs will yield all the required information to obtain the key. Take for instance the previous encryption example

```
CTX: csasxoeqhwrfcopyngdrcsasxohtdthz
PTX: cryptodoesreallymeancryptography
SAM: 01234012340123401234012340123401
```

The derived shift amounts (SAM line) show a clear period of length 5, and can be converted back into the `abcde` key in a straightforward fashion.

1.1.3 Breaking Playfair

The Playfair cipher was invented by Charles Wheatstone in 1854 and later on its use was encouraged by lord Playfair.

To encrypt with the Playfair cipher, the first step is to pick a keyword. Given a keyword a 5×5 table is filled, row by row, employing the letters of the keyword in order, without repetitions (i.e., if a letter has already been inserted in the table, it is not inserted again). Finally, fill in the remaining places in the table with the letters of the alphabet which have not been inserted yet, in alphabetic order. Typically, when employing the English alphabet the letters I and J are mapped to the same table cell. Figure 1.3 reports an example of table building for the keyword "WINTERMUTE".

	1	2	3	4	5
1					
2					
3					
4					
5					

(a) Empty Table

	1	2	3	4	5
1	W	I	N	T	E
2	R	M	U		
3					
4					
5					

(b) Keyword inserted

	1	2	3	4	5
1	W	I	N	T	E
2	R	M	U	A	B
3	C	D	F	G	H
4	K	L	O	P	Q
5	S	V	X	Y	Z

(c) Same column

Figure 1.3: Example of the construction of the Playfair cipher table

Once the table is ready, the plaintext is encrypted one digram (pair of letters) at once. The encryption of a digram depends on the position of its letters within the table according to three rules depicted in Figure 1.4 In case the positions of the two letters of the digram occupy the corners of a proper rectangle on the table, each one of them is encrypted with the one lying on the opposite corner of it, row-wise. For instance, in the example the digram MZ is encrypted as BV. In case the letters are aligned row wise, each one is encrypted taking the first one on its right, wrapping around the table if needed. In the example RU gets encrypted to MA. Finally, if the letters lie on the same column, each one is encrypted with the one right below it, again wrapping around the table if needed. In the example IL is encrypted as MV.

1. How large is the keyspace of Playfair? Is it amenable to a bruteforce attack?
2. Break Playfair with a known ciphertext only attack.

	1	2	3	4	5
1	W	I	N	T	E
2	R	M	U	A	B
3	C	D	F	G	H
4	K	L	O	P	Q
5	S	V	X	Y	Z

(a) Different row/column

	1	2	3	4	5
1	W	I	N	T	E
2	R	M	U	A	B
3	C	D	F	G	H
4	K	L	O	P	Q
5	S	V	X	Y	Z

(b) Same Row

	1	2	3	4	5
1	W	I	N	T	E
2	R	M	U	A	B
3	C	D	F	G	H
4	K	L	O	P	Q
5	S	V	X	Y	Z

(c) Same column

Figure 1.4: Playfair encryption rules: plaintext in red, ciphertext in green

3. Break Playfair with a chosen plaintext attack. Can you do better than known ciphertext only?

4. Break Playfair with a known plaintext attack. Can you do better than chosen ciphertext only?

Solution:

1. Considering that each Playfair table yields a different encryption map, the keyspace for playfair is $25!$. Recalling that $\ln(25!) \approx (25\ln(25) - 25 + \ln(25))$ we get a keyspace of $\approx 2^{120}$. A straightforward bruteforce approach is thus out of question.
2. Observe that, given a specific digram, the Playfair cipher will encrypt it always to the same digram. Consequentially, it is basically the two-letter analog of a simple substitution cipher. The Playfair cipher can be broken simply through digram frequency analysis in the same way a letter substitution cipher is.
3. Breaking playfair via chosen plaintext attack can be easily done supplying a text made out of all the possible ($26^2 = 676$) digrams, and tabulating the answers of the encryption oracle. The total plaintext length is 1352 characters. Albeit this will not yield the actual 5×5 encryption table employed in the usual construction of the cipher, the tabulated information equivalent to it.
4. Breaking Playfair via known plaintext attacks can be done examining the encrypted digrams output by the encryption oracle, and filling in gradually the 5×5 encryption table solving a constraint satisfaction problem akin to Sudoku.
   Alternatively, the attacker may simply re-apply the strategy employed for the chosen plaintext attack, assuming that all digrams will eventually occur in the plaintext. Assuming that all the digrams are equally probable would yield an expected plaintext length of $1352\log(1352) \approx 9700$ letters.

Consider the ciphertext of a Playfair cipher and provide a reasonable estimate for its entropy under the assumption that the source message is in English and the key is picked uniformly over all the possible 25! keys.

Solution:

A first rough estimate can be provided noting that a Playfair cipher never encrypts a letter into itself due to its algorithm. Thus assuming coarsely that the probability of a ptx letter being enciphered into one of the possible ctx ones are equal, we get a first rough estimate $H(C) \leq -\sum_{c \in C} \Pr(C = c)\log_2(\Pr(C = c)) = -\sum_{c \in C} \frac{1}{24}\log_2(\frac{1}{24}) = \log_2(24) \approx 4.6b$ considering the ciphertext alphabet to be made of single characters.

1.1.4 Breaking Nihilist

The Nihilist cipher was employed by both the members of the Nihilist movement, when emprisoned by russian czars and US prisoners in the Vietnam war.

In order to encrypt a message with the Nihilist cipher, two keywords k_1 and k_2 are needed. The first keyword is employed to build a table in the same way the Playfair cipher does (see previous exercise).

For the Nihilist cipher, the row index and column index of the table are the relevant portion of it. The table is in fact used as to encode each plaintext letter as the row index/column index pair where it appears in the table. The second keyword is also encoded following the same scheme. Note that, this encoding effectively reduces the number of symbols of the ciphertext alphabet down to 5. This technique had been known for a long time even then as a Polybius Square, as the greek historian and scholar Polybius was (and is) credited with inventing them.

Table 1.1: Polybius Square constructed employing `WINTERMUTE` as the key

	1	2	3	4	5
1	W	I	N	T	E
2	R	M	U	A	B
3	C	D	F	G	H
4	K	L	O	P	Q
5	S	V	X	Y	Z

For instance, employing Table 1.1 the word `Rincewind` would be encoded as `21 12 13 31 15 11 12 13 32`. Once both k_2 and the plaintext have been encoded as numbers, the Nihilist ciphertext is obtained as follows: given the length l_2 of k_2, the i-th ciphertext element is obtained as the sum of the i-th encoded plaintext number with the ($i \bmod l_2$)-th encoded k_2 element. For instance, picking k_2=`luggage` $\rightarrow$ `42 23 34 34 24 34 15` we compute:

```
PTX: 21 12 13 31 15 11 12 13 32
KEY: 42 23 34 34 24 34 15 42 23
CTX: 63 35 47 65 39 45 27 55 55
```

and get `633547653945275555` as ciphertext .

1. Let l_1 and l_2 be the key lengths of the two Nihilist keys; how large is the Nihilist cipher keyspace?
2. Break Nihilist with a known ciphertext only attack.

Solution:

1. The first Nihilist key is only employed to generate the encoding table in the same way as it was in Playfair, thus, there are $25! \approx 2^{84}$ tables. The second key keyspace is simply $26^{l_2} \approx 2^{4.6l_2}$ wide, thus, combining the effect of the two keys we obtain a $2^{84+4.6l_2}$ sized keyspace. A brute force approach is *way* out of question.
2. Exploiting the fact that the encoding table acts digram-wise, it is easy to see that Nihilist is simply a digram-based version of the common Vigenére cipher, and can thus be broken exploiting the same technique.
   In particular, considering the ciphertext as a sequence of symbols, each one constituted of **two** digits, Kasiski examination can be applied to it retrieving the length n of the second key. Once this has been done, note that the encoding provided by the Polybius square is a simple monoalphabetic substitution, and thus, a ciphertext symbol in a given position $i \mod n$ will be the result of two subsequent fixed monoalphabetic substitutions applied to the plaintext (the Polyibius encoding, and the addition). It is thus possible to apply frequency analysis to the **two** digit symbols of the ciphertext, retrieving the n values of a key equivalent to

performing both the monoalphabetic substitutions at once.
To optimize the analysis, note that the final key addition is performed with a simple sum. As a consequence some of the values for the elements of k_2 may leak more easily if they are high. For instance, assume that the value of a ciphertext element is ≥ 100; the only admissible values for the key are the ones in $[45, 55]$.

1.2 Permutation Ciphers

1.2.1 A simple permutation cipher

Take π to be the following permutation of the elements $\{1, \ldots, 8\}$:

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 1 & 6 & 2 & 7 & 3 & 8 & 5 \end{pmatrix}$$

Compute the permutation π^{-1} and decrypt with it the following ciphertext:

ETEGENLMDNTNEOORDAHATECOESAHLRMI

Solution:

From the previous notation, it is easy to note that the inverse permutation is

$$\pi = \begin{pmatrix} 4 & 1 & 6 & 2 & 7 & 3 & 8 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

and obtain "Gentlemen really don't read each others email" as the plaintext

1.2.2 Matrix Based Permutation Cipher

A given transposition cipher (a.k.a. permutation cipher) splits the plaintext into snippets of equal size of k bytes (representing ASCII characters) each and rearranges every snippet separately using a permutation key. For example, for $k = 3$ the key $\pi = (2\ 1\ 3)$ can be used and the text *minefield ahead* will be encrypted as *imnfeileda haed*. This transposition cipher can be described using the following encryption matrix:

$$[x_1 x_2 x_3] \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [y_1 y_2 y_3]$$

where $[x_1 x_2 x_3]$ is the plaintext and $[y_1 y_2 y_3]$ is the ciphertext. Note that, a permutation matrix (as the one shown above) is defined over $0, 1$, and each column includes a single cell different from zero. Remember that the set of permutations with a given length defines a group with the internal composition law being the natural composition (i.e., subsequent applications) of two permutations over the same plaintext snippet.

- Find the corresponding decryption key and show the decryption matrix, and decrypt the ciphertext *ememtenoteusady*.
- Suggest how to attack this cipher, specifying one or more attack models.
- To increase the difficulty of breaking the cipher Alice decided to encrypt her plaintext twice using the transposition cipher but with two different keys. She considers three options, one with $k_1 = k_2 = 3$, another one with $k_1 = k_2 = 5$ and one with $k_1 = 5$, $k_2 = 6$. Will these three schemes increase the complexity of breaking the cipher?

Solution:

- The decryption key can be obtained as the inverse permutation of $\pi = (2\ 1\ 3)$, which is $\pi^{-1} = (2\ 1\ 3)$ (π is actually its own inverse). The corresponding decryption matrix is :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  The decrypted plaintext is : $meetmeontuesday$

- A simple permutation cipher is vulnerable to a chosen plaintext attack: it is sufficient to encrypt a long sequence of distinct characters (f.i, the alphabet) to infer from the corresponding ciphertext the permutation. A known plaintext attack can be able to infer the permutation in a similar fashion, although some ambiguity from repeated letters may ensue. However, given enough plaintext material, such ambiguities can be solved.
- The choice of $k_1 = k_2 = 3$ does not increase the overall security margin: it is possible to obtain a length-3 permutation equivalent to the subsequent application of 2 lenght-3 ones. Similarly, the choice of $k_1 = k_2 = 5$ provides the same security margin of a single length-5 permutation. Picking $k_1 = 5$, $k_2 = 6$ instead provides at best the security margin of choosing a permutation with length $lcm(5, 6) = 30$.

1.3 Composition of historical ciphers

1.3.1 Double Substitution

Suppose you have two substitution ciphers $S_{k_1}(m)$ and $S_{k_2}(m)$ with the substitutions defined in the following table.

m	ABCDEFGHIJKLMNOPQRSTUVWXYZ
$S_{k_1}(m)$	EKMFLGDQVZNTOWYHXUSPAIBRCJ
$S_{k_2}(m)$	AJDKSIRUXBLHWTMCQGZNPYFVOE

Describe the cipher c=$S_{k_1}(S_{k_2}(m))$, and comment on its security.

Solution:

The composition of two substitution ciphers gives again a substitution cipher. Consequently, the composition of two substitution ciphers is no stronger than a single substitution cipher.

1.3.2 Double Vigenére

Given two keys for a Vigenére $E(\cdot)$ k_1 and k_2 with respective lengths l_1, l_2:

1. Comment on the security of $E_{k_1}(E_{k_2}(\cdot))$: is it better or worse than $E_{k_1}(\cdot)$?
2. Comment on the security of $E_{k_1}(E_{k_2}(E_{k_1}(\cdot)))$: is it better or worse than $E_{k_1}(E_{k_2}(\cdot))$?

Solution:

1. The change in security depends on the values of l_1 and l_2:
   - If $l_1 = l_2$ the security margin of $E_{k_1}(E_{k_2}(\cdot))$ is exactly the same of $E_{k_1}(\cdot)$. This is due to the monoalphabetic substitutions corresponding to each letter of k_1 acting separately on the ones corresponding to the letters of k_2. Since the keyword lengths match, it is possible to obtain a k_3 equivalent to the effect of applying a Vigenére cipher with k_2 followed by one with k_1. In particular $k_3 = E_{k_1}(k_2)$.

- If $l_1 \neq l_2$ the security margin of $E_{k_1}(E_{k_2}(\cdot))$ greater or equal to the one of $E_{k_1}(\cdot)$. In particular, the application of the two ciphers is equivalent to the one of a Vigenére cipher with key k_3 as long as the least common multiple l_3 of l_1 and l_2 and built as follows: concatenate k_2 $\frac{l_3}{l_2}$ times, obtaining s, then $k_3 = E_{k_1}(s)$.

2. For the reasons explained before, applying once more $E_{k_1}(\cdot)$ does not change the security margin.

1.3.3 Double Permutation

Consider two permutation ciphers $P_{k_3}(m)$ and $P_{k_4}(m)$ which take blocks of six letters and permute them according to the permutations

$$k_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 3 & 5 & 4 & 1 & 2 \end{pmatrix} \quad k_4 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 5 & 4 & 3 & 6 & 1 \end{pmatrix}$$

Describe the cipher $c = P_{k_3}(P_{k_4}(m))$, and comment on its security.

Solution:

The composition of two permutation ciphers acting on 6 characters at once gives again a permutation cipher acting on 6 characters at once. It is thus possible to define a permutation P_{k_5} which performs the equivalent action of applying P_{k_4} and P_{k_3}, and is still a permutation on 6 elements. Consequently, the keyspace does not grow, thus composition of two permutation ciphers defined on sets of elements with the same length is no stronger than a single permutation cipher.

1.3.4 Mixing Substitutions and Permutations

Take the substitution and permutation ciphers from the previous questions and comment on the security of $c = S_{k_1}(P_{k_3}(m))$ and $c = S_{k_1}(P_{k_3}(S_{k_2}(P_{k_4}(m))))$.

Solution:

The composition of two substitution-permutation ciphers gives again a substitution-permutation cipher (as the operations commute one can re-arrange the composition such that we have substitution-substitution- permutation-permutation which we know is the same as substitution-permutation) Consequently, the composition of two substitution-permutation ciphers is no stronger than a single substitution-permutation cipher.

Consider the three ciphers $K_{k_5}(m)$, $K_{k_6}(m)$, $K_{k_7}(m)$ which 'exclusive-ors' (adds) blocks of six letters with the keys (modulo 26)

$$\begin{array}{lcl} k_5 & = & \text{AHIXGA} \\ k_6 & = & \text{LIQRTS} \\ k_7 & = & \text{QRYLAF} \end{array}$$

Consider the cipher $c = K_{k_5}(S_{k_1}(P_{k_3}(K_{k_6}(S_{k_2}(P_{k_4}(K_{k_7}(m)))))))$ Comment on its security. What about the security when $k_1, \ldots k_4$ are made public but the secret key is simply k_5, k_6 and k_7?

Solution:

The composition of layers of addition-substitution-permutation ciphers is stronger than a single such layer as the (modular) addition does not commute with the other operations! Making the keys $k_1, \ldots, k_4$ public does not seem to affect security. Essentially the S and P players introduce confusion and diffusion, with the K layers making the output depend on the key.

1.4 Enigma

With the advent of mechanical support to encryption, substution ciphers found their embodiment in rotor machines. Among them the most famous example is Enigma, the encryption machine employed by the Germans during WWII to protect the confidentiality of their communications. The key idea of the mechanism is to mechanically apply a substitution cipher, picking a fresh substitution map for each character. Following the picture in Figure 1.5, we will delineate how this substitution is done:

1. The current supplied by a battery (1) flows to the switches of a keyboard (2): pressing one of the keys corresponding to a letter closes the corresponding switch, selecting it as the current plaintext.

2. The current flows to a plugboard, where a plug socket for each letter ispresent. An empty socket allows the current to flow through acting as a short circuit, mapping the letter to itself (3). A populated socked acts as a letter swapping device, effectively exchanging the mapping the letter to the one in the socket where the other end of the cable is plugged ((7) and (8)).

3. The current flows to a fixed connector (4), and henceforth to a set of cogs, named *rotors*. Each rotor has two sets of contacts, one for each side, which are cross wired (5). The rotors effectively act as letter substitution tables.

4. Once the current has flown through the rotors, it passes through a so-called *reflector*: it is a fixed connector which maps back the incoming current into the circuit corresponding to a different letter.

5. The current flows back through the plugboard once again, and, finally through a lightbulb (there is a lightbulb for each letter of the alphabet). The lit-up letter is the actual ciphertext.

In addition to this, it is possible to rotate the outgoing contacts of a rotor with respect to the inner wirings, effectively defining 26 possible substitution maps for each rotor. This setting, known as ring setting or *ringstellung*, does not change during an encryption operation. Each time the operator presses a key on the keyboard, the first rotor is advanced by one step, changing effectively the applied substitution map. Each of the rotors has a pin in correspondence to one specific letter: this pin lodges into a notch in the following rotor, effectively moving the next one once in a full turn of the previous. Moreover, if the rotor is the middle one, it causes the left rotor to step on once as well as itself (this mechanism is known as double stepping). The rotors can be chosen by the operator during the setup of the machine among a set provided by the machine constructor: at first only three rotors were available, but their number was increased to five for infantry operated machines and eight for navy ones. It was also possible to pick the reflector among a set of available ones.

The secret key of the Enigma cryptoscheme is the actual configuration of the Enigma Machine, that is: the choice of the rotors, their placement, the *ringstellung*, their initial positions, the reflector type, and the plugboard settings.

1. Consider a three-rotor Enigma, where the choice is made among three possible rotors, and the reflector is picked between two, with a plugboard populated with 6 plugs. Compute the effective keyspace of the machine.

2. Is it better in terms of security improvement, to add two more rotors to choose from for the configuration, or add two more cables on the plugboard?

3. Point out the flaws in the Enigma design, and delineate a way to exploit them.

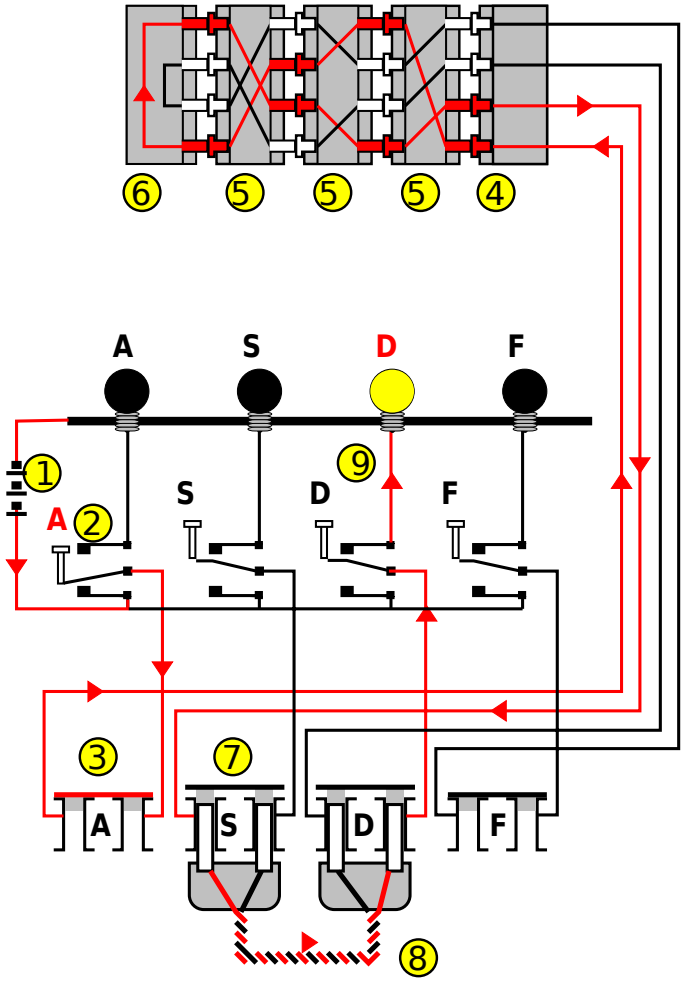


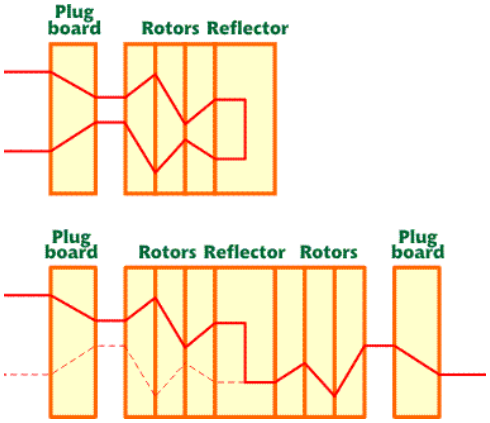
Figure 1.5: A diagram of a simplified three rotor Enigma machine. Differs from the full version only for being defined on a reduced alphabet

Solution:

1. The best way to compute the Enigma keyspace is to compute separately the possible configurations of each part separately, and then combine them together.
   - The choice of three rotors among three implies that there are $3! = 6$ possible choices
   - The ring settings for the rotors are independent, and each rotor has 26 possible settings, yielding $26^3 \approx 2^{4.7*3} \approx 2^{14}$ possibile settings
   - The starting positions of the rotors account for the same amount of possible settings of the ring settings
   - The choice of the reflector accounts for 2 possibilities.
   - The plugboard configuration is a little bit trickier to sort out. Consider that, when adding a cable to a plugboard with o occupied places, there are $(26 - o)(26 - o - 1)$ possible ways to insert it. Consequentially adding 6 plugs to a plugboard amounts to $(26 \cdot 25) \cdot (24 \cdot 23) \cdot (22 \cdot 21) \cdot (20 \cdot 19) \cdot (18 \cdot 17) \cdot (16 \cdot 15)$

Consequentially, the keyspace of the aforementioned Enigma Machine is $2^{31.5} \times 2^{52.1} \approx 2^{83}$, where the first term takes into account the effects of rotors, reflector and their configuration, and the second the effect of the plugboard

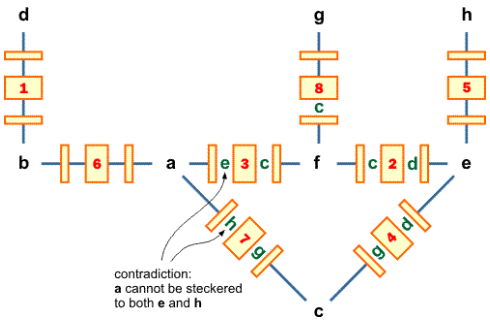
2. Adding two rotors to the set from which they are chosen improves the rotor choice term from $3! = 6$ to $\binom{5}{3} \times 3! = \frac{5!}{3!(5-3)!} \times 3! = 60$, while adding two more cables adds a factor $(14 \cdot 13) \cdot (12 \cdot 11) = 24024$, thus adding two cables is substantially better than adding more rotors.
3. The main critical flaw of the Enigma machine can be spotted looking at the wiring scheme: since pressing a key on the keyboard automatically disconnects the corresponding lamp, a letter will never be encrypted as itself. This observation is the key to Enigma cryptanalysis as the following actions may be performed:
   (a) Assume that the ciphertext contains a long, common word: typically *Wettervorhersage* (weather forecast) was used. It is possible to find a portion of the ciphertext which, letter by letter, does not match any of the letters in *Wettervorhersage*. Such a portion of the ciphertext is a possible cipertext for *Wettervorhersage*. From this point on assume to be in a known plaintext scenario.
   (b) Represent the Enigma machine as follows:



   i.e., consider the double circuit obtained from mirroring the rotor set across the reflector.
   (c) Consider the following alleged plaintext/ciphertext pair to be known through the analysis at point (a):

Plain text	b	e	a	c	h	b	a	g
Encrypted text	d	f	f	e	e	a	c	f

   (d) Assume the rotors are in a given position (1 in the Figure) when encrypting the first character of the plaintext b $\rightarrow$ d:



   And deduce what should the other letters be mapped to according to the following steps until either a contradiction is obtained (a letter should be mapped to two different

ones), or all the constraints have been applied. If the rotor configuration respects the constraints, mark it as a possible one.

(e) Note that the contradictions in the deductions are independent from the effects of the plugboard → it is possible to bruteforce the configuration of the double scrambler independently!

(f) Once the double scrambler configuration has been sorted out, Enigma is reduced to the composition of three substitution ciphers, of which the central one is known.

A detailed description of the Enigma cryptanalysis is available in Nigel P. Smart, Cryptography, An Introduction, Chapter 4
`http://www.cs.bris.ac.uk/~nigel/Crypto_Book/book.ps`

1.5 Principles of Information Theory

1.5.1 Computing the entropy of random variables

Consider the sample space as the outcome of a fair, six-faced die, $\Omega\{1,2,3,4,5,6\}$. Consider three different random variables on Ω, as follows: X is the number obtained with the die, Y takes values *even* and *odd* depending on the die outcome, and Z takes values *low* and *high* depending on whether the die roll is below 4 or not.

- Compute $H(X), H(Y), H(Z)$
- Compute $H(X,Y), H(X,Z), H(Y,Z)$
- Compute $H(X|Y), H(Y|X)$

Solution:

- Following the definition of entropy we have that $H(X) = 6\frac{1}{6}\log_2(6) \approx 2.58$ as all the outcomes of X are equally probable, and have probability $\frac{1}{6}$. Following the same line of reasoning, we obtain $H(Y) = H(Z) = 2\frac{1}{2}\log_2(2) = 2$
- To compute the values of joint entropy required, consider the joint probability distributions of the variable pairs, reported as in Tables 1.2. Applying, in a straightforward fashion the definition of entropy to them, we have that $H(X,Y) = -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}}\Pr(x,y)\log_2(\Pr(x,y)) = 6\frac{1}{6}\log_2(6) \approx 2.58$b (matching the intuition that the uncertainty of knowing the result, and whether it's odd or not is the same as knowing the result), $H(X,Z) = 2.58$b (the same goes for the result and *low* or *high*), and $H(X,Z) = -(2\frac{1}{6}\log_2(\frac{1}{6}) + 2\frac{2}{6}\log_2(\frac{2}{6})) = \frac{1}{3}\log_2(6) + \frac{2}{3}\log_2(3) = 0.86 + 1.05 = 1.91$b.

X	Y	
	odd	*even*
1	$\frac{1}{6}$	0
2	0	$\frac{1}{6}$
3	$\frac{1}{6}$	0
4	0	$\frac{1}{6}$
5	$\frac{1}{6}$	0
6	0	$\frac{1}{6}$

X	Z	
	low	*high*
1	$\frac{1}{6}$	0
2	$\frac{1}{6}$	0
3	$\frac{1}{6}$	0
4	0	$\frac{1}{6}$
5	0	$\frac{1}{6}$
6	0	$\frac{1}{6}$

Y	Z	
	odd	*even*
low	$\frac{2}{6}$	$\frac{1}{6}$
high	$\frac{1}{6}$	$\frac{2}{6}$

Table 1.2: Joint probability distributions for (X, Y), (X, Z), (Y, Z)

Chapter 2

Block Ciphers

2.1 Data Encryption Standard

2.1.1 DES-V and DES-W

Consider two ciphers, DES-V and DES-W defined as follows:

- $\texttt{DES-V}(k, k_1, m) = \texttt{DES}(k, m) \oplus k_1$
- $\texttt{DES-W}(k, k_1, m) = \texttt{DES}(k, m \oplus k_1)$

where m is the plaintext block to be encrypted, with $|m|= 64$ bit, and k, k_1 indicate two symmetric keys with lengths $|k|= 56$ bit e $|k_1|= 64$ bit respectively.

- Point out for each cipher the complexity of a bruteforce attack, showing how many ptx-ctx pairs are needed.
- Discuss the security margin of DES-V and DES-W with respect to the common $\texttt{DES-X}(k, k_1, k_2, m) = k_2 \oplus \texttt{DES}(k; m \oplus k_1)$

Solution:

Discussion for DES-V: A first impression suggests that a single (m, c) pair should be enough for a bruteforce attack. However, there are 2^{56} possible key pairs k, k_1 which make the equivalence $\texttt{DES}(k, m) \oplus c = k_1$ hold. Employing another ptx-ctx pair (m', c'), it is possible to devise a proper attack on the scheme. Consequentially, with two pairs we can extract the correct value of the key k since the following relations must hold:

- $\texttt{DES}_k(m) \oplus c = k_1$
- $\texttt{DES}_k(m') \oplus c' = k_1$

Consequentially, also $\texttt{DES}_k(m) \oplus \texttt{DES}_k(m') = c \oplus c'$ holds. It is thus possible to perform a bruteforce on the value of the key k employing only 2^{57} DES encryptions. Once we obtain the value of k, we can derive the key k_1 as $k_1 = \texttt{DES}_k(m) \oplus c$. Summing up, the total effort to crack DES-V is at most 2^{57} DES encryptions and two ptx-ctx pairs against a total key length of 120 bits.

Discussion for DES-W: The same line of reasoning applies to $\texttt{DES}_k^{-1}(c) \oplus \texttt{DES}_k^{-1}(c') = m \oplus m'$. Therefore, the total effort to crack DES-W is at most 2^{57} DES encryptions and two ptx-ctx pairs against a total key length of 120 bits.

Comparison with DES-X: Both DES-V and DES-W offer a significantly lower security margin than DES-X, as basically the bruteforcing effort is the same of a single DES, while the DES-X scheme needs an effort in the range of 2^{120} DES operations with a key length of 2^{184}.

2.1.2 DES-A

Consider a strengthened version of DES, DES-A defined as follows

$$\text{DES-A}(k_1, k_2, m) \stackrel{\text{def}}{=} \text{DES}(k_1, m \boxplus k_2)$$

where m is the plaintext, k_1 a 56-bit key k_2 a 64-bit key, and $\boxplus$ the common addition modulo 2^{64}. Assuming that an attacker is granted access to two plaintext/ciphertext pairs $(m_a, c_a), (m_b, c_b)$ coming from two unrelated plaintexts m_a, m_b, which is the minimum computational effort required to breach the security of DES-A?

Solution:

Although at a first glance it may seem that no better strategy than a bruteforce effort through a 2^{120}-bit wide keyspace is possible, the attacker is actually able to reduce the computational effort significantly. Defining the intermediate state i of DES-A as the sum of the plaintext with k_2, notice the fact that

$$i_a \boxminus i_b = (m_a \boxplus k_2) \boxminus (m_b \boxplus k_2) = m_a \boxminus m_b$$

where $\boxminus$ is the subtraction modulo 2^{64}. The attacker is able to obtain from the two plaintext/ciphertext pairs the value $m_a \boxminus m_b$ easily. Exploiting this value, the attacker is able to bruteforce the value of k_1 independently through computing the candidate values for $\bar{i}_a$ and $\bar{i}_b$ as $\bar{i}_a = \text{DES}^{-1}(k_1, c_a)$ and $\bar{i}_b = \text{DES}^{-1}(k_1, c_b)$ and checking whether $\bar{i}_b \boxminus \bar{i}_b = m_a \boxminus m_b$. This can be performed with a worst-case computational cost of $2 \cdot 2^{56}$ DES decryptions. When the correct value of k_1 is retrieved , the attacker is simply able to compute the value of k_2 as either $i_a \boxminus m_a$ or $i_b \boxminus m_b$ at the cost of a single subtraction. The security margin of DES-A is thus reduced at 2^{57} DES encryptions.

2.1.3 DES Collision attacks

DES is a 64-bit block cipher which uses a 56 bit key. We now consider the CBC mode of operation. Assuming we encrypt a very large amount of data with the same key, it is possible that two different ciphertext blocks c_i, c_j are the same.

- Does the fact that $c_i = c_j$ with $i \neq j$ leak any information regarding the plaintext blocks m_i, m_j?
- Does replacing DES with 3DES avoid the information leakage?
- How can we protect ourselves against this attack?

Solution:

- We recall that a ciphertext block c_i in CBC mode is the result of $c_i = \mathtt{DES}_k(c_{i-1} \oplus m_i)$, for all $i \geq 1$, $c_0 = \text{IV}$. Thus, in case $c_i = c_j$ for some $i, j, i \neq j$ we can deduce that $c_{i-1} \oplus m_i = c_{j-1} \oplus m_j$ (i.e. the inputs to the DES cipher are the same). Therefore the following differential information on the plaintext blocks is leaked: $m_i \oplus m_j = c_{i-1} \oplus c_{j-1}$. For instance, it is possible to detect if two encrypted plaintext blocks are the same.
- If we replace DES by 3DES the block length of the cipher primitive will remain the same. As the probability of a collision (assuming both DES and 3DES to be equally random permutation of bits) depends only on the block size, the security against the collision attack is *not* increased.
- The only way to protect against this attack, while still employing the CBC mode of operation, is an increase in the cipher primitive block size, to lower the probability of a collision. A viable alternative is to change the mode of operation of the cipher, for instance adopting the counter mode.

2.1.4 DES Block Widening

The DES cipher suffers from both a short key length (DES is in fact practically breakable through straightforward exhaustive search), and a small block size (64 bit). Figure 2.1 proposes three alternatives to combine 4 DES primitives to achieve a higher security margin

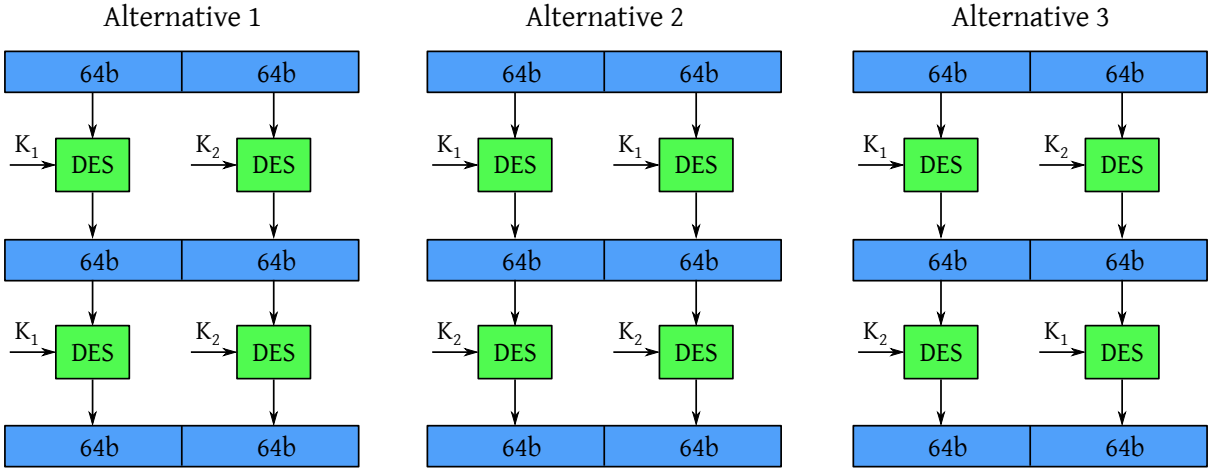


Figure 2.1: Three schemes to widen DES block size to 128 bits, employing two different keys.

- Do the proposed alternatives yield an increased security block cipher with respect to a single DES, against an attacker with a limited amount of storage (e.g. less than 1GB).
- Do the proposed alternatives yield an increased security block cipher with respect to a single DES, against an attacker with an unlimited amount of storage?

Solution:

- The computational effort to break Alternative 1 is only marginally higher than the one of breaking a single DES: in fact, the attacker is able to bruteforce separately K_1 and K_2 with a worst-case cost of $2^{56} \cdot 2$ DES encryptions. The total computational effort is thus 2^{58}. Alternative 2 requires the attacker to exhaustively search for both K_1 and K_2 together, thus yielding a $2^{56} \cdot 2^{56}$ computational effort. Similarly, also Alternative 3 requires a $2^{56} \cdot 2^{56}$ computational effort.
- Allowing the attacker to use unlimited storage, Meet-in-the-Middle attacks become feasible. This approach allows to reduce the computational complexity to break Alternatives 2 and 3 at the expense of a significant amount of storage. In particular, Alternative 2 can be attacked computing, and storing in an indexed table, the values of the decryption of a ciphertext with every possible K_2, costing a 2^{56} effort in both time and space. Once this has been done, computing the result of the encryption of the plaintext with all the possible values for K_1, and checking whether the result matches an entry in the table, will effectively reveal the correct value for both K_1 and K_2 at the cost of 2^{56} DES encryptions. The computational effort is thus reduced from $2^{56} \cdot 2^{56}$ to $2 \cdot 2^{56}$, plus 2^{56} memory slots. Concerning Alternative 3, it is possible to mount a similar attack, although with higher costs. The first phase of the meet in the middle attack memorizes in two indexed tables the results of both the encryption of the plaintext, and the decryption of the ciphertext with all the possible values for K_2: the cost of such an operation is 2^{57} in both time and space. Once the tables are complete, computing the result of both the encryption and the decryption of plaintext and ciphertext with all the possible values of K_1 and looking up for them in the table will yield the correct value of both K_1 and K_2 with an additional 2^{57} computational effort. Summing up, breaking Alternative 3 requires a computational effort of $2 \cdot 2^{57}$ and 2^{57} memory slots.

2.2 Modes of Operation

2.2.1 Error Recovery in CBC and ECB

How many steps are required for error recovery from a ciphertext transmission error in `ECB` and `CBC` modes?

Solution:

- For ECB: just one. The decryption of every block is fully independent
- For CBC: two blocks. The decryptions going wrong will be the one of the incorrect ciphertext, and the one where the incorrect ciphertext is used as a pad.

2.2.2 CBC Malleability

Assume a disk volume is fully encrypted through AES-256-CBC with a properly selected random IV. Can you exploit the encryption scheme to conduct malicious activity on the disk? Justify the answer and propose an effective countermeasure

Solution:

Yes, recalling that CBC provides confidentiality but not integrity. In particular, it is possible to alter the contents of the disk of a block at the cost of causing a misdecryption of the previous one (or, tampering with the IV, to alter the first block with no decryption errors). In particular, flipping a bit in a ciphertext block will flip the corresponding bit in the plaintext obtained from the decryption of the next block as depicted in Figure 2.2 Such an exploitation of the CBC malleability allows the attacker to change standard known disk areas (e.g. MBR) in an arbitrary fashion[1].

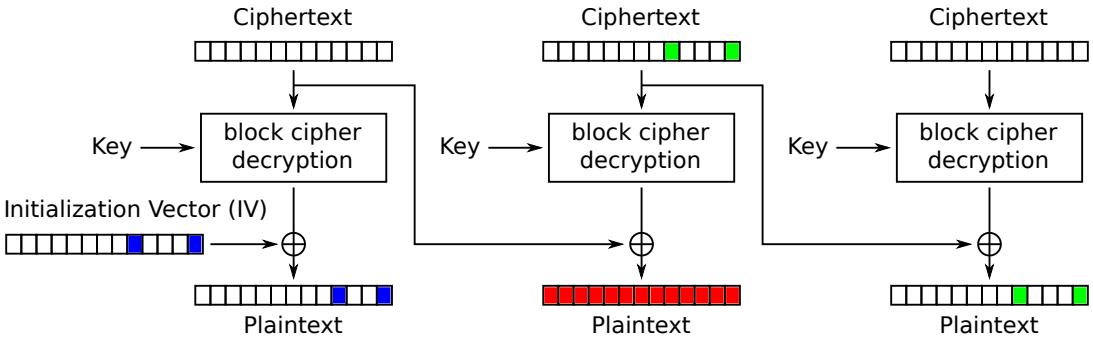


Figure 2.2: Examples of malleability of the CBC encryption: in blue a malleability attack altering in a known way the IV, in green the one altering a ciphertext block, with the resulting decryption error highlighted in red, and corresponding bit flips in green

Such a data-alteration attack can be prevented either through adding error detection codes to the blocks, or employing a proper mode of operation for data at rest encryption such as the XTS mode of operation.

[1]For a practical explanation on how to lead the attack see `http://www.jakoblell.com/blog/2013/12/22/practical-malleability-attack-against-cbc-encrypted-luks-partitions/`.

2.2.3 CTR With Repeated IV

A remote file storage provides encryption support on the stored files, employing AES-256-CounTeR , initializing the counter with a 16 byte fixed user ID. Anyone, upon registration, deposits a secret key which will be used to encrypt his files, and is bound to his user ID. Anyone can ask for the encryption and storage of a file through simply providing the ID for which it should be stored. Anyone can ask to obtain an encrypted file bundle, providing its name.

1. Assume you know that the user with ID `44722e4a616d65734d6f726961727479` has stored a file of your interest, of which you know the name `Qlanzvpfbsna Nfgrebvq`. Can you obtain the plaintext contents of it?
2. Propose a solution to mitigate this issue, without resorting to authentication of the users.

Solution:

1. Recalling that the `CTR` mode of operation basically employs a block cipher to obtain a pseudorandom keystream which is then `xor`-added to the plaintext to obtain the ciphertext. Since the pseudorandom keystream depends only on the secret key and the value of the `IV`, in the described setting, it will be unique, and always the same, for each user. It is thus possible to retrieve the cleartext of `Qlanzvpfbsna Nfgrebvq` as follows:
   (a) Retrieve the encrypted version of `Qlanzvpfbsna Nfgrebvq`
   (b) Generate a zero-filled file as big as `Qlanzvpfbsna Nfgrebvq`
   (c) Upload the zero-filled file asking for it to be encrypted for ID `44722e4a616d65734d6f726961727479`
   (d) Retrieve the result of the zero-filled file encryption, add it via `xor` to `Qlanzvpfbsna Nfgrebvq` and get the cleartext.
2. Changing the `IV` of `CTR` mode is a paramount requirement to its security. The previous attack can be easily thwarted picking a random `IV` for each file.

2.2.4 Faulty Disk Encryption

Rosen Association inc. mandates the use of encrypted disk volumes for its own employees enforcing the following:

- All the disks have two partitions, one containing the OS binaries (with a non-integrity checking filesystem), while the other hosts the employee data. The OS partition is to be encrypted with AES-256-CBC with randomized IV, while the user data partition should employ AES-256-CTR, using the string "NEXUS6-ROSENInc." as IV.
- All the employee partitions are set up by the system administrator which, during formatting, takes care to fill the space within the encrypted volume with zeroes, so to fill the effective physical diskspace with random data.
- The employees are enforced to pick their disk encryption passwords as 16 character random strings containing decimal digits. The password is hashed through SHA-2-256 to fill in the AES-128 key.

(a) Discuss the choices of Rosen inc. concerning the choice of the encryption algorithm for the OS partition, highlighting issues and proposing a way to fix them.

(b) Discuss the choice of the encryption algorithm for the employee partition, and the decision of the system adminstrator to wipe the internal space of the disk volume.

(c) Discuss the password policy chosen by Rosen, providing a quantitative estimate for the attacker effort to be spent to completely break the whole system, in the most efficient way possible

Solution:

(a) Picking a block cipher in CBC mode, without any integrity check, allows an attacker to exploit the CBC malleability to his own advantage. In particular, considering the case of a disk volume where an OS is stored, it is possible to alter partially the OS binaries without deciphering the volume.

(b) The choice of a counter-mode of operation for disk encryption is not problematic as long as the CTR IV is picked at random. Given the choice of Rosen inc. to employ a fixed IV, the pseudorandom keystream obtained out of the AES-128 in CTR mode is the same for all the partitions of the employees using the same password. This in turn allows to extract information without the secret key, adding via `xor` two instances of encrypted data with the same password (e.g. two full disk dumps taken in different moments). The decision of the system administrators to fill with zeroes the partitions effectively worsens the issue: the attacker is basically provided with the whole keystream, if he is able to dump the disk of a laptop right after it has been issued.

(c) Considering the password policy, the effective password space is 10^{16}, which is equivalent to roughly 2^{53}. This in turn implies that a full exhaustive search for the password is within feasibility for moderately motivated attackers. As a further issue, the passwords are used as-is, without any salted hashing scheme, thus allowing an attacker to successfully exploit TMTO strategies.

2.2.5 Altering messages with CBC and CTR

Two friends agree on using a block cipher for encryption and need to choose a mode of operation between CBC and CTR. An adversary is able to intercept and change messages exchanged between them. Now consider the following scenarios.

(a) For some messages, the adversary knows the first plaintext block M_1 and wants to replace it by another block A_1 of his choice, leaving the rest of the message unchanged.
Show that the adversary can achieve this if CTR mode is used.
Do you think he can do it with CBC mode?

(b) The adversary knows that in some messages the plaintext content of the *last* encrypted block is a randomly generated secret value v. Decide for the two modes whether the adversary can corrupt these blocks, so that the receiver of the communication gets a message that looks good after decryption, but contains the wrong secret value v.

Solution:

(a) The adversary can achieve this if the encryption is in CTR mode.
The encryption of the first block is $C_1{=}M_1{\oplus}\mathrm{Enc}_k(\mathtt{IV}+1)$, from which he can compute $\mathrm{Enc}_k(\mathtt{IV}{+}1){=}M_1{\oplus}C_1$. He can successfully replace C_1 with $C_1'{=}A_1{\oplus}\mathrm{Enc}_k(\mathtt{IV}{+}1)$ through computing $C_1'{=}A_1{\oplus}(M_1{\oplus}C_1)$. The encryption of the other blocks remains unaltered.
For CBC mode, we need to look at the decryption process and observe that:
$M_1{=}\mathrm{Dec}_k(C_1){\oplus}C_0$, thus $\mathrm{Dec}_k(C_1){=}M_1{\oplus}C_0$ (Both M_1 and C_0 are known by the adversary). The adversary cannot change C_1, since that would affect the correct decryption of C_2. Instead, he will replace C_0 with C_0', where C_0' is such that $A_1{=}\mathrm{Dec}_k(C_1){\oplus}C_0'$. Solving for C_0', he gets $C_0'{=}A_1{\oplus}\mathrm{Dec}_k(C_1){=}A_1{\oplus}(M_1{\oplus}C_0)$.

(b) For both modes the adversary can replace the last ciphertext block with any other block. The decryption of all blocks but the last one will provide a correct message. The plaintext corresponding to the last block will not be distinguished from a corrupt plaintext, as it includes a random value v. Therefore, in the proposed scenario there is no way for the receiver to detect the alteration.

2.3 Block Cipher Cryptanalysis

2.3.1 S-Box Design

While designing a new 4-to-4 bit S-Box, we came up with the following three definitions (I_j is the j-th input bit, O_j is the j-th output bit):

1. $\forall\ j \in \{0,1,2,3\},\ O_j = I_{j-3 \bmod 4} \oplus I_{j-2 \bmod 4} \oplus I_{j+5 \bmod 4}$
2. $\forall\ j \in \{0,1,2,3\},\ O_j = (\neg I_{j-3 \bmod 4} \wedge I_{j-2 \bmod 4}) \vee (I_{j-3 \bmod 4} \wedge \neg I_{j-2 \bmod 4})$
3. $\forall\ j \in \{0,1,2,3\},\ O_j = I_{j-3 \bmod 4} \oplus I_{j-2 \bmod 4} \wedge I_j$

Highlight which choice among these three is the most secure against linear and differential cryptanalyses, and describe why the other two choices are vulnerable.

Solution:

The first S-box definition is trivially linear, thus it does not offer any resistance against linear and differential cryptanalyses. The second S-Box is equivalent to $\forall\ j{\in}\{0,1,2,3\}\ O_j{=}I_{j-3 \bmod 4} \oplus I_{j-2 \bmod 4}$, and is thus linear too, having the same weakness of the first one. The third S-box, albeit not particularly strong in fact, it is the stronger one of the three as it incorporates a nonlinear term.

2.3.2 Linear and Differential Cryptanalysis - (I)

Consider the simple substitution-key addition cipher depicted in Figure 2.3

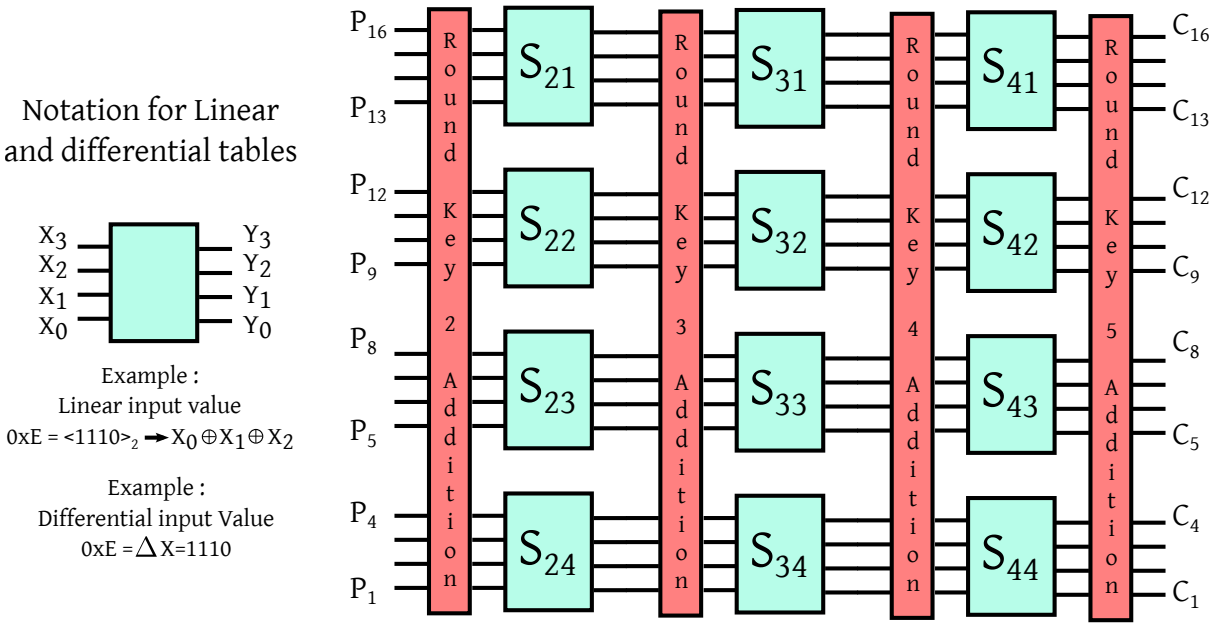


Figure 2.3: Simple cipher with only substitution and key addition layers

Given the reduced DES Sbox described in Table 2.1 and its linear biases and differential probabilities described in Table 2.2 and Table 2.3
As an exercise, complete the following tasks:

- Point out the input-output relation(s) providing the best linear bias(es) for the considered SBox
- Point out the input-output difference relation(s) providing the best differential probability(-ies) for the considered SBox
- Up to which point (operation of the cipher, be it a substitution or a round key addition) should we provide approximations?
- Considering the path depicted in Figure 2.4, and the provided tables, compute the linear bias for the expression highlighted expliciting the relations chosen to approximate each Sbox
- Considering the path depicted in Figure 2.4, and the provided tables, compute the differential probability for the expression highlighted expliciting the differentials chosen to approximate each Sbox
- Point out how many ptx-ctx pairs are required to carry out the two aforementioned attacks

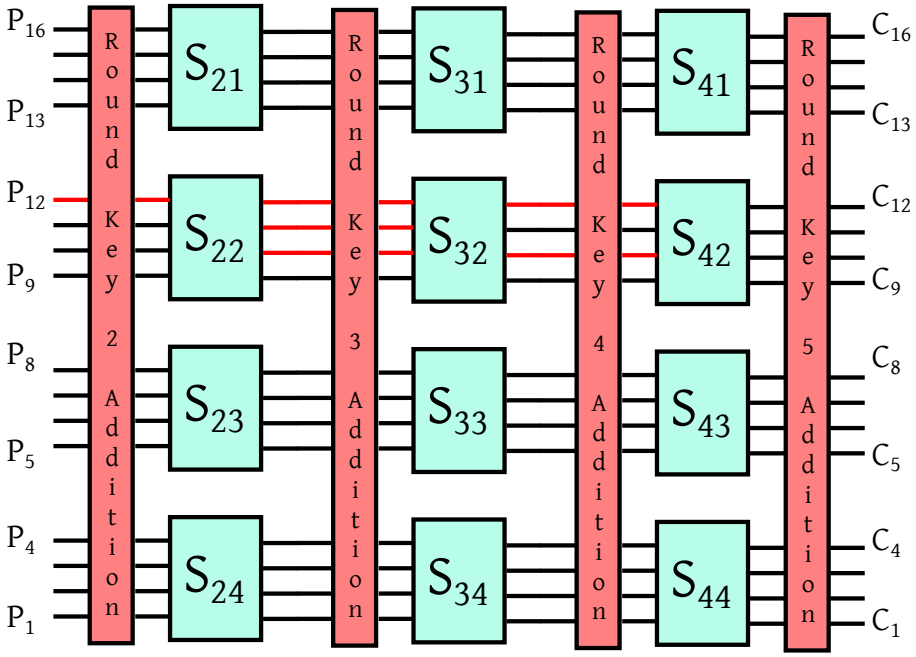


Figure 2.4: Approximation path for the cipher

Table 2.1: SmallDES 4×4 Sbox, values in hexadecimal

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

Table 2.2: Linear biases table for the SmallDES SBox (bitwise input relations on rows, bitwise output relations on columns). Biases are expressed in 16th. We recall that given a bias $\epsilon_{a,b}$, the probability for the relation $a = b$ to hold is $\frac{1}{2} + \epsilon_{a,b}$. Red values are positive biases, blue values are negative biases.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	-2	-2	0	0	-2	6	2	2	0	0	2	2	0	0
2	0	0	-2	-2	0	0	-2	-2	0	0	2	2	0	0	-6	2
3	0	0	0	0	0	0	0	0	2	-6	-2	-2	2	2	-2	-2
4	0	2	0	-2	-2	-4	-2	0	0	-2	0	2	2	-4	2	0
5	0	-2	-2	0	-2	0	4	2	-2	0	-4	2	0	-2	-2	0
6	0	2	-2	4	2	0	0	2	0	-2	2	4	-2	0	0	-2
7	0	-2	0	2	2	-4	2	0	-2	0	2	0	4	2	0	2
8	0	0	0	0	0	0	0	0	-2	2	2	-2	2	-2	-2	-6
9	0	0	-2	-2	0	0	-2	-2	-4	0	-2	2	0	4	2	-2
A	0	4	-2	2	-4	0	2	-2	2	2	0	0	2	2	0	0
B	0	4	0	-4	4	0	4	0	0	0	0	0	0	0	0	0
C	0	-2	4	-2	-2	0	2	0	2	0	2	4	0	2	0	-2
D	0	2	2	0	-2	4	0	2	-4	-2	2	0	2	0	0	2
E	0	2	2	0	-2	-4	0	2	-2	0	0	-2	-4	2	-2	0
F	0	-2	-4	-2	-2	0	2	0	0	-2	4	-2	-2	0	2	0

Table 2.3: Differential probabilities (expressed in 16th) table for the TinyDES SBox (inputs on rows, outputs on columns)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Solution:

- The best (i.e., highest in absolute value) linear biases are achieved by the relations:
  - $X_3 = Y_1 \oplus Y_2 \oplus Y_3$, bias $\frac{6}{16}$
  - $X_2 \oplus X_3 = Y_0 \oplus Y_3$, bias $-\frac{6}{16}$
  - $X_2 = Y_0 \oplus Y_1 \oplus Y_2$, bias $-\frac{6}{16}$
  - $X_0 = Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_3$, bias $-\frac{6}{16}$
- The best (i.e. highest) differential probability is achieved by the differential:
  - $(\Delta X = 1011 = \texttt{0xB}, \Delta Y = 0010 = \texttt{0x2})$, probability of holding $\frac{8}{16} = \frac{1}{2}$
- We should provide approximations up to the input of the last nonlinear layer, i.e. the inputs of the S_4. SBoxes
- The linear approximation employed in the highlighted linear path employs the following relations:
  - For S_{22}, the relation $X_3 = Y_1 \oplus Y_2 \oplus Y_3$, with bias $\frac{6}{16}$
  - For S_{32}, the relation $X_1 \oplus X_2 \oplus X_3 = Y_1 \oplus Y_3$, with bias $-\frac{4}{16}$

  Combining the biases through the Pile-Up Lemma, the approximations holds with bias $2^{2-1}(\frac{6}{16})(-\frac{4}{16}) = \frac{3}{16}$
- The differential approximation employed in the highlighted linear path employs the following relations:
  - For S_{22}, the relation $\Delta X = 0001, \Delta Y = 0111$, with bias $\frac{2}{16}$
  - For S_{32}, the relation $\Delta X = 0111, \Delta Y = 0101$, with bias $-\frac{2}{16}$

  The total probability for the differential approximation to hold is thus: $(\frac{2}{16})(\frac{2}{16}) = \frac{1}{64}$
- The number of ptx-ctx pairs required to carry out the linear cryptanalysis attack is roughly $N_p = \frac{1}{\epsilon^2}$, where ϵ is the bias of the approximation relation, thus in this case $N_p = \frac{256}{9} = 28.4 \approx 29$. For the differential cryptanalysis the estimated number of pairs needed is $N_p = \frac{1}{p}$, where p is the probability for the differential approximation to hold, thus in this case $N_p \approx 64$

2.3.3 Linear and Differential Cryptanalysis - (II)

Consider the simple SPN block cipher depicted in Figure 2.5, with identical Sboxes $\mathbf{S}_{i,j}$, $i, j \in \{1, \ldots, 4\}$ defined in Table 2.4.

Table 2.4: Sbox, values in hexadecimal

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	2	4	0	8	3	6	12	11	5	10	7	14	15	13	9	1

Complete the following tasks, employing the linear and differential biases tables included.

- Complete, for both linear and differential cryptanalysis, the partial path dashed in Figure 2.5 through choosing suitable (i.e. among the best possible) relations. The chosen relations can be different for linear and differential cryptanalysis.
- Compute the linear bias and differential probability for your proposed linear/differential paths.
- Which of the two paths you proposed is more advantageous for an attacker? Evaluate the resistance of the cipher in terms of how many plaintext/ciphertext pairs you need to retrieve the subkey.

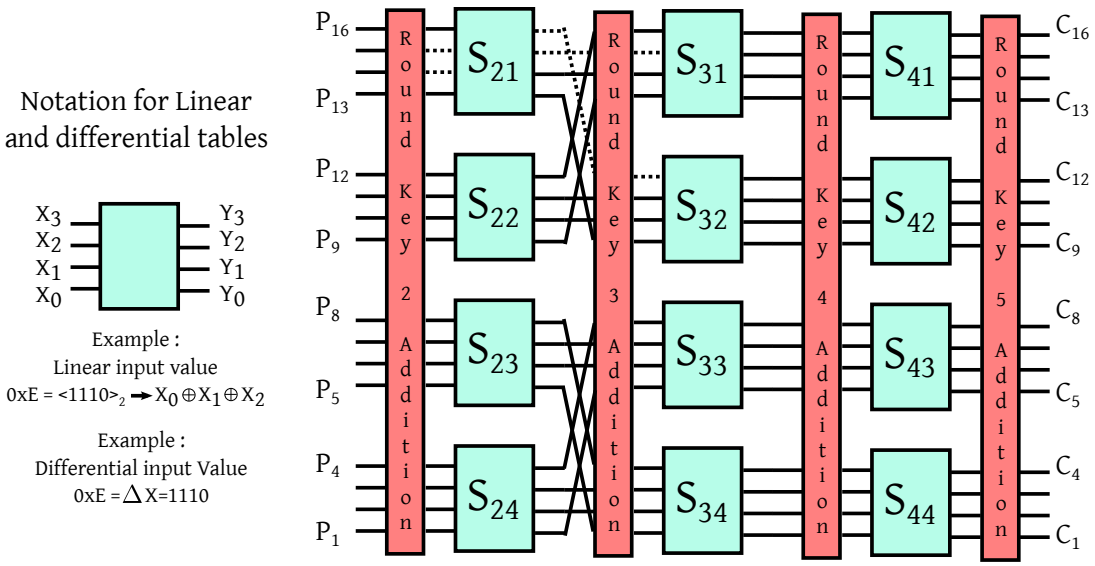


Figure 2.5: Partial approximation path for the cipher

Table 2.5: Linear biases table for the SBox (bitwise input relations on rows, bitwise output relations on columns). Biases are expressed in 16th. We recall that given a bias $\epsilon_{a,b}$, the probability for the relation $a = b$ to hold is $\frac{1}{2} + \epsilon_{a,b}$.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	-2	0	2	0	2	0	-2	2	0	-2	0	2	4	-2	4
02	0	0	-2	-2	-2	2	-4	0	2	2	0	0	0	-4	-2	2
03	0	-2	-2	0	2	0	0	2	0	-2	2	-4	2	0	-4	-2
04	0	4	0	0	0	4	0	0	2	-2	2	2	-2	2	-2	-2
05	0	-2	0	-2	0	2	0	2	4	2	0	-2	0	2	4	-2
06	0	0	2	-2	2	-2	-4	-4	0	0	2	-2	-2	2	0	0
07	0	2	2	-4	-2	0	0	2	-2	0	-4	-2	0	2	-2	0
08	0	4	0	4	2	-2	-2	2	2	2	-2	-2	0	0	0	0
09	0	2	0	-2	2	0	-2	0	0	-2	0	2	6	0	2	0
0A	0	0	-2	-2	0	-4	2	-2	4	0	-2	2	0	0	-2	-2
0B	0	-2	-2	0	4	2	-2	0	-2	0	-4	2	-2	0	0	-2
0C	0	0	4	0	2	2	2	-2	0	4	0	0	2	-2	-2	-2
0D	0	2	-4	-2	2	0	2	0	-2	4	2	0	0	2	0	2
0E	0	0	-2	2	-4	0	-2	-2	-2	2	0	0	2	2	0	-4
0F	0	2	-2	0	0	2	2	-4	0	-2	-2	-4	0	-2	2	0

Table 2.6: Differential probabilities (expressed in 16th) table for the SBox (inputs on rows, outputs on columns)

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	**16**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	0	**2**	0	0	**2**	**2**	**2**	**4**	**2**	0	0	0	0	0	**2**
02	0	0	**4**	0	**2**	0	**2**	0	0	0	0	0	**4**	**2**	0	**2**
03	0	0	0	0	**4**	0	0	0	**2**	0	**4**	**2**	0	**2**	**2**	0
04	0	**2**	**2**	**2**	0	0	0	**2**	0	0	**2**	0	**2**	0	**2**	**2**
05	0	0	0	0	**4**	**2**	**2**	**4**	**2**	0	0	**2**	0	0	0	0
06	0	0	0	**4**	0	0	0	0	**2**	0	0	**2**	**2**	0	**4**	**2**
07	0	**2**	0	**2**	**2**	0	**2**	0	**2**	**2**	**2**	**2**	0	0	0	0
08	0	0	0	0	0	**2**	**2**	**4**	0	0	**2**	**2**	**2**	0	**2**	0
09	0	**2**	**2**	0	0	0	0	0	**2**	**2**	0	0	0	**2**	**4**	**2**
0A	0	0	**2**	**2**	0	**4**	**2**	**2**	0	0	**4**	0	0	0	0	0
0B	0	**2**	**2**	**2**	**2**	0	0	0	0	0	**2**	0	**2**	**2**	0	**2**
0C	0	0	0	0	0	**2**	**2**	0	0	**6**	0	**2**	**2**	**2**	0	0
0D	0	**4**	**2**	**2**	0	0	0	0	0	**2**	0	**2**	**2**	0	0	**2**
0E	0	**2**	0	0	**2**	**4**	0	0	**2**	**2**	0	**2**	0	0	0	**2**
0F	0	**2**	0	**2**	0	0	**2**	**2**	0	0	0	0	0	**6**	**2**	0

Solution:

(a) The linear path should be completed through choosing a linear biases for the S_{31} and S_{32} $S-$boxes starting from X_2 and X_3 respectively: we pick $X_2 \rightarrow X_1 \oplus X_3$ and $X_3 \rightarrow X_0 \oplus X_1$ with bias $\frac{-4}{16}$ and $\frac{4}{16}$ respectively. The differential path should be completed through choosing biases for the S_{31} and S_{32} $S-$boxes starting from X_2 and X_3 respectively: we pick $\Delta Y = Y_2$ and $\Delta Y = Y_0$ with bias $\frac{4}{16}$ and $\frac{4}{16}$ respectively.

(b) The complete linear bias for the relation can be computed piling up the linear biases as $2^2(\frac{-2}{16} \cdot \frac{-4}{16} \cdot \frac{4}{16}) = \frac{1}{32}$. The differential bias is computed by combining together all the biases as $\frac{4}{16} \cdot \frac{4}{16} \cdot \frac{4}{16} = \frac{1}{64}$.

(c) Recalling that linear cryptanalysis needs $\frac{1}{\varepsilon^2}$ pairs ($\lceil \frac{32^2}{1} \rceil = 1024$ in our case), while differential cryptanalysis needs roughly $\frac{1}{p}$ ($\lceil \frac{64}{1} \rceil = 64$ in our case) we can state in this differential cryptanalysis is more effective.

2.3.4 Linear and Differential Cryptanalysis - (III)

Consider a substitution-permutation block cipher with a 128-bit block and a 256-bit key, employing 32 identical 4×4 bit S-boxes for the substitution layer. The best (i.e., highest) linear bias for the S-Box is $\varepsilon = \frac{1}{8}$, while the highest differential probability is $\frac{1}{4}$. The permutation layer is built so that the 4 output bits of each S-Box are employed as inputs to four different boxes in the subsequent layer. The diffusion of the cipher is such that given a change in an S-box input at round i, 4 s-boxes will be involved at round $i+1$, 16 at round $i+2$ and all of them from round $i+3$ onwards. The cipher round acts on the state performing with the substitution layer, the permutation layer and the round key addition, in this order. A single key extra key addition is present before the first round takes place.

(a) Compute the value of a conservative estimate of the linear bias useful for retrieving the last round key for the described block cipher assuming it is 4 rounds long, and the amount of plaintext-ciphertext pairs available. Is it possibly broken by linear cryptanalysis?

(b) Compute the value of the differential probability useful for retrieving the last round key for the described block cipher assuming it is 4 rounds long, and the amount of plaintext-ciphertext pairs available. Is it possibly broken by differential cryptanalysis?

(c) Keeping the same round structure, and the same key length, is it possible to render the block linear and differential cryptanalysis immune? If yes, describe what should be tuned and to which extent.

Solution:

To provide a conservative estimate of the linear bias, assume that the best linear bias can always be employed to approximate an S-Box. Consequentially, to build a linear approximation able to retrieve the last round the best achievable linear bias piles up 1 S-Box from round 1, 4 from round 2, and 16 from round 3, for a total of 21 active S-Boxes, considering for all of them the best linear bias $\varepsilon{=}\frac{1}{8}$. The bias for the full approximation is thus $2^{21-1}(\frac{1}{8})^{21} = 2^{20}\frac{1}{2^{62}} = \frac{1}{2^{42}}$, requiring $\approx 2^{84}$ plaintext-ciphertext pairs to be exploited. Since a bruteforce over the full keyspace requires 2^{256}, the cipher is possibly broken by linear cryptanalysis.

Similarly to the linear cryptanalysis case, we will be exploiting the best differential probability for all the S-Boxes. The number of active S-Boxes does not change with respect to the previous analysis, thus we obtain a differential probability for the full approximation which is $(\frac{1}{4})^{21} = \frac{1}{2^{42}}$, requiring 2^{42} plaintext-ciphertext pairs to be exploited. The cipher is quite likely to be broken by differential cryptanalysis.

Given the constraints, the cipher can be tweaked to be immune to both cryptanalysis raising the number of rounds. In detail, adding a round after the fourth activates 32 S-boxes per added round. To provide linear cryptanalysis immunity, consider that adding round will provide a multiplicative factor of $2(\frac{1}{8})^{32}) = \frac{1}{2^{64}}$ on the linear bias over the current one of the cipher, raising the number of required ciphertexts by a factor $(2^{64}) = 2^{128}$. Two extra rounds are thus sufficient to provide linear cryptanalysis immunity (2^{42+256} ptx-ctx pairs required). Concerning differential cryptanalysis, a round provides a multiplicative factor on the differential probability equal to $(\frac{1}{8})^{32} = 2^{96}$. As a consequence, the differential probability for r rounds is $2^{42} \cdot r2^{96}$. To achieve differential cryptanalysis immunity, it should hold that $2^{42} \cdot (r-4)2^{96} \geq 2^{256}$, thus $r \geq 7$, i.e. three more rounds should be added.

Chapter 3

Stream Ciphers

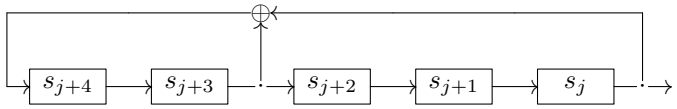
3.1 Linear Feedback Registers

3.1.1 LFSR (I)

Consider an LFSR characterised by the following primitive Connection polynomial: $C_1(x) = x^5 + x^2 + 1$. Draw the structure of the LFSR and write down both the period of the output sequence and the recurrence equation of the register.

Solution:

Recalling the definition of the connection polynomial of a length L LFSR as $C(x) = 1 + \sum_{i=1}^{L} c_i x^i$, we know from the form of C_1 that the feedback coefficients of the LFSR under exam are: $c_1 = 0, c_2 = 1, c_3 = 0, c_4 = 0, c_5 = 1$ and $L = 5$, thus leading to the following structure



The recurrence equation of the LFSR can be consequently written down as:
$s_{j+5} = s_{j+3} + s_j \mod 2$.
Since the connection polynomial is primitive, we expect the LFSR to have the maximum achievable period given the length $L = 5$, i.e. $N = 2^L - 1 = 31$.

Given the initial state $S_0 = \langle s_4, s_3, s_2, s_1, s_0 \rangle = \langle 1, 1, 0, 1, 1 \rangle$ write down the first 10 output bits from the LFSR.

Solution:

Through simple computations employing the LFSR structure drawn before, we obtain:

s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
1	1	0	1	1	0	0	1	0	1

3.1.2 LFSR (II)

Consider the following sequence of bits which has been output by a LFSR with length $L = 5$.

Keystream									
s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
1	1	0	1	1	1	1	1	0	1

Rebuild the LFSR structure from the output bits and draw the structure of the corresponding LFSR, stating its connection polynomial.

Solution:

We recall that the each output bit can be obtained as:

$$s_{j+L} = \sum_{i=1}^{L} c_i \, s_{j+L-i} \bmod 2$$

Consequently, we can write down, for each clock cycle starting from $j = 0$ to $j = L - 1$ the following equations:

Cycle	Output bit equation
$j = 0$	$s_L = c_1 \, s_{L-1} + \ldots + c_L \, s_0 \bmod 2$
$j = 1$	$s_{L+1} = c_1 \, s_L + \ldots + c_L \, s_1 \bmod 2$
$\ldots$	$\ldots \ \ldots \ \ldots$
$j = L - 1$	$s_{2L-1} = c_1 \, s_{2L-2} + \ldots + c_L \, s_{L-1} \bmod 2$

The previous set of equations can be used to build a system of equations, employing as unknowns the values $c_i, i \in \{1, \ldots, 5\}$, in order to rebuild the structure of our LFSR. Such a system looks like this:

$$\begin{pmatrix} s_{L-1} & \cdots & s_0 \\ s_L & & s_1 \\ \vdots & & \vdots \\ s_{2L-2} & \cdots & s_{L-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_L \end{pmatrix} = \begin{pmatrix} s_L \\ s_{L+1} \\ \vdots \\ s_{2L-1} \end{pmatrix}$$

As we know the values of $s_0, \ldots, s_{L-1}$, we can substitute them in the previous equation system, thus obtaining the following one:

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

We now recall from basic linear algebra that the unknowns in the previous system are obtained through multiplying the inverse of the matrix by the constant terms vector as follows:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

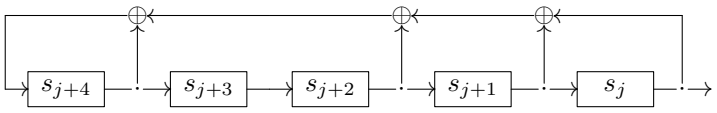
Now we have retrieved the values of the connection polynomial coefficients, and we are thus able to write it down as:

$$C_2(x) = x^5 + x^4 + x^3 + x + 1$$

We recall the fact that we also know the inner state of the LFSR from the first bits of the keystream as:

$$S_0 = \langle s_4, s_3, s_2, s_1, s_0 \rangle = \langle 1, 1, 0, 1, 1 \rangle$$

Employing these pieces of information we can fully reconstruct the initial state and structure of the LFSR under exam as:



The recurrence equation characterising this LFSR can be written down easily by simply examining the structure of the connections as:

$$s_{j+5} = s_{j+4} + s_{j+2} + s_{j+1} + s_j \bmod 2,$$

Even if not explicitly required by the exercise, We note that, as $C_2(x) = x^5 + x^4 + x^3 + x + 1$ is a primitive polynomial the period of the LFSR N is maximum, and corresponds to $N = 2^L - 1 = 31$

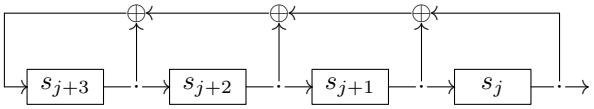
3.1.3 LFSR (III)

Consider an LFSR characterized by the following "Connection Polynomial":
$C(x) = x^4 + x^3 + x^2 + x + 1 \in \mathbb{F}_2[x]$.

- Draw the structure of the LFSR and write down the recurrence equation of the register
- What is the hypothetical maximum period of the output sequence? What are the consequences of having an irreducible connection polynomial that is not also primitive?
- Given the initial state $S_0{=}\langle s_3, s_2, s_1, s_0 \rangle{=}\langle 1, 0, 1, 1 \rangle$, write down all the possible transitions of the register values

Solution:

- $L = 4,\ C(x) = 1 + \sum_{i=1}^{L} c_i x^i = 1 + x + x^2 + x^3 + x^4 \in \mathbb{F}_2[x] \Rightarrow c_1 = 1,\ c_2 = 1,\ c_3 = 1,\ c_4 = 1$



$$s_{j+4} = s_{j+3} + s_{j+2} + s_{j+1} + s_j \bmod 2$$

- $N{=}2^L{-}1,\ L{=}4, N{=}15$ a primitive connection polynomial guarantees that $N{=}2^L{-}1$, while an irreducible but not primitive connection polynomial implies that the period of the output sequence N is not maximum but it divides $2^L{-}1$
- $s_{11} \rightarrow s_7 \rightarrow s_{15} \rightarrow s_{14} \rightarrow s_{13} \rightarrow s_{11}$

Chapter 4

Cryptographic Hashes

4.0.1 Hashes with multiplicative groups

Let p be a large prime and g be a generator of $(\mathbb{Z}_p^*, \cdot)$. Suppose we are considering the function $h : \mathbb{Z} \mapsto \mathbb{Z}_p^*$ for use as a hash function, where $h(x) = g^x \mod p$, assuming x to be an arbitrary binary string interpreted as a positive integer number, i.e.: $x \in \mathbb{Z}$, $x \geq 0$.
Note that the *compression property* of typical hash functions is satisfied by the above definition as messages x of arbitrary bit-length are hashed into a fixed size digest.

Assuming the computational difficulty of the discrete logarithm problem in $(\mathbb{Z}_p^*, \cdot)$, which properties of cryptographic hash functions does h satisfy? Discuss if h is a sound cryptographic hash function or not.

Solution:

The *first pre-image* property is guaranteed by the difficulty of the discrete log problem. The *second pre-image* property is not satisfied by the proposed construction as given x and $h(x)$, it is trivial to compute $x' = x + l(p-1)$, with $l = \{1, 2, 3, \ldots, \}$ such that $h(x') = h(x)$. The *collision resistance* property is also not satisfied as two messages with the same digest can be found picking them in the set $\{x \text{ s.t. } x = x_0 + l(p-1),\ l \geq 1,\ \forall\ x_0 \geq 0\}$

4.1 Collisions

4.1.1 Saving on digest size

A company is employing RIPEMD-64 hashes (64 bit digest) as an integrity checking mechanism for files on a disk. The security officer is currently concerned with the security margin against intentional replacement of files with garbage data and suggests to change the hashing mechanism to SHA-2-256. The commercial department points out that employing SHA-2-256 would increment the amount of required disk space and, to reduce the migration costs, proposes to employ SHA-2-256 to hash the file contents, and store only the first quarter of the digest for integrity checking.

(a) Is the security officer concern well justified? Provide a quantitative motivation to it.

(b) Is it possible to adopt the commercial department solution? Justify quantitatively the answer.

Solution:

(a) Yes: a 64 bit hash implies the possibility of obtaining a collision through exhaustive search in around 2^{32} operations, which is well within the realm of feasibility.

(b) No: performing the comparison only on the first 64 bits of the digest of SHA-2-256, and assuming that the bits of the first quarter of the hash follow the same uniform distribution as the rest of it, it is possible to obtain a partially colliding digest with the same computational effort of the previous hashing mechanism.

4.1.2 Strengthening MD5

The MD5 hash function suffers from both collision attacks in 2^{41}, simply appending a properly crafted block to the message and in 2^{18} appending two blocks. Besides this, the size of the MD5 digest (128 bits) limits its maximum collision resistance to 2^{64}, which is high, but within range for a well founded organization. Willing to strengthen the aging MD5, three proposals are made:

- $\texttt{MD5-A}(m)$ is defined as $\texttt{SHA-2-256}(\texttt{MD5}(m))$
- $\texttt{MD5-B}(m)$ is defined as $\texttt{MD5}(\texttt{SHA-2-256}(m))$
- $\texttt{MD5-C}(m)$ is defined as $\texttt{AES}_{\texttt{k}}(\texttt{MD5}(m))$, with a per-user-fixed key k
- $\texttt{MD5-D}(m)$ is defined as $\texttt{SHA-2-256}(\texttt{MD5}(\texttt{SHA-2-256}(m)))$

Comment on the improvements to the collision resistance of MD5, in particular assuming that 2^{64} is not a sufficient security margin for current exhaustive search capabilities.

Solution:

- MD5−A does not improve the collision resistance of plain MD5 at all, as a message pair colliding in MD5 will also collide in MD5−A, since SHA−2−256 is a deterministic algorithm. It is thus possible to have collisions for MD5−A in 2^{18}, appending two well crafted block padding to any message.
- MD5−B Prevents the attacker from exploiting the collisions relying on the inner structure of MD5: this is due to the computational infeasibility of controlling the output of the SHA−2−256 hash function which is the input of MD5 (note that controlling the output of SHA−2−256 implies that SHA−2−256 is not first-preimage attack robust). However, MD5−B leaves still open the possibility of obtaining a collision in 2^{64} as the output of the MD5−B digest is 128 bit wide.
- MD5−C Does not provide any improvements on the collision resistance front just like MD5−C, as having a deterministic bijective primitive processing the output of the original MD5 will still result in a collision in MD5 being propagated to the output.
- MD5−D basically feeds the output of MD5−B into SHA−2−256. Thus, as MD5−B has a collision resistance of 2^{64}, and equal inputs to SHA−2−256 will yield equal outputs also MD5−D has a collision resistance of 2^{64}.

4.1.3 Keyed and strengthened SHA-31337

In an effort to provide a strengthened and keyed version of SHA-1, a designer proposes to initialize the first three out of five variables composing the 160-bit state with three message dependent integers, one of which is secret. The designer claims that:

- This modification significantly enhances the collision resistance of SHA-1
- The lack of knowledge of the secret state register provides effective security, preventing the signature of a message by someone who does not know its content.

Prove or dispute the claims of the designer.
Solution:

- The claim concerning the collision resistance of SHA-31337 is false: since the message digest is still 160 bits wide, the collision resistance of SHA-31337 is still 2^{80}
- The secret value employed to initialize the state is effectively long $\frac{160}{5} = 32$ bits, which is well within the feasibility range for an exhaustive search. To be able to produce "keyed" signatures on an arbitrary message the attacker simply needs a valid message/keyed-digest pair on which the exhaustive search for the secret value will be performed.

4.1.4 Triple Collisions

Consider $h : M \mapsto D$, a cryptographic hash function.

1. How many random samples (i.e. input messages) do we need to take from M to get a collision, assuming h behaves like a perfectly random function.
2. How many random samples we need to take from M to obtain a three-way collision (i.e., How many random samples, $m_i \in M$, do we need to take in order to observe that three different inputs $m_1, m_2, m_3 \in M$ map to the same value, $h(m_1)=h(m_2)=h(m_3)$?)

Solution:

1. Given a pair of different messages m_1, m_2, the probability of m_1 being mapped to a given $h(m_1) \in D$ is $\frac{1}{|D|}$. Consequentially, m_1 has $\frac{1}{|D|}$ probability of being mapped to the same value of $h(x_2)$, thus the probability of a collision $h(x_1)=h(x_2)$ is $\frac{1}{|D|}$. Since there is no constraint on picking a specific m_1 or m_2, consider that, drawing n input messages from M, the number of possible message pairs formed is $\binom{n}{2} \approx n^2$. Recall that, for any finite or countable set of events, the probability that at least one of the events happens is no greater than the sum of the probabilities of the individual events (a.k.a. *Union Bound*). Thus, the probability to obtain a collision in one of the message pairs which can be built drawing n random hash inputs is: $\frac{n^2}{|D|}$.
   Considering that the previous quantity is a probability, and thus must be lesser than 1: $\frac{n^2}{|D|} < 1$ the upper bound for the collision probability is $\mathbf{n < |D|^{\frac{1}{2}}}$.
2. Similarly to the previous case, the probability that $h(m_1)=h(m_2)$ is $\frac{1}{|D|}$. Assuming that this event happened, $h(m_3)$ has only $\frac{1}{|D|}$ probability of being mapped to the same value as both m_1 and m_2, so the probability of all three digests being equal is $\text{Prob}(h(x_1)=h(x_2) \wedge h(x_2)=h(x_3))=\frac{1}{|D|^2}$.
   Drawing n random samples from the domain M, the number of possible triples is $\binom{n}{3} \approx n^3$. Applying the *union bound* approximation, the probability to obtain a 3-way collision among triplets of messages picked from a set of n samples is given by the summation n^3-times of $\frac{1}{|D|^2}$. Therefore, an upper bound for the probability to get a 3-way collision is: $\frac{n^3}{|D|^2}$.
   Considering that, in the same way as the previous derivation, $\frac{n^3}{|D|^2}$ must be less than 1, the sought upper bound is $\mathbf{n < |D|^{\frac{2}{3}}}$.

4.2 Length extension attacks

4.2.1 The Gambling House

A gambling house provides a perfectly fair game to his clients following this game routine:

1. The client picks the odds of winning as a power of two $o = 2^i, 1 < 140$, and sends the value o, together with an arbitrarily long *nonce*, and the amount of money m to the gambling house
2. The gambling house computes $r = \texttt{SHA} - \texttt{1}(secret|nonce) \bmod 2^i$, and reveals r to the gambler
3. If $r = 0$, the gambling house pays the client $m * 2^i$

The gambling house actively checks if the client employs the same *nonce* multiple times, changing the value of *secret* and forfeiting the current game when this happens.

- Is this system free from cheating? If not, describe how it is possible to cheat, and the computational effort required to do so.

- Would employing SHA-2-256 instead of SHA-1 compensate for the system weakness?

Solution:

- Unfortunately, the gambling house can be cheated around exploiting the properties of the Merkle-Damgaard structure of MD5. In particular, we know that the Merkle-Damgaard structure is vulnerable to length-extension attacks, allowing an attacker to obtain $H(A|B)$, given $H(A)$ and B. This fact can be exploited to drive the generation of r at the client's will, provided that it is possible to obtain the full value of r. Since the gambling house reveals at most the 140 less significant bits of it, the attacker needs to reconstruct the remaining 20 as follows:
  1. Gamble with 2^{140} and a nonce n_1, obtaining a 140 bit long truncated response r_1
  2. Gamble with 2^{140} and craft a nonce as $n_1|n_2$, where n_2 is a second random nonce, obtaning the truncated response r_2
  3. For all the possible values $0 \leq a \leq 2^{20} - 1$ the attacker checks when extending the SHA-1 state $a|r_1$ yields n_2 as the 140 least significant bits of the hash value for a given $\bar{a}$.
  4. The value $\bar{a}|r_1$ is the actual result of $r = SHA(secret|n_1)$, under the assumption that comparing only the last 140 bits of the result of the length extended SHA-1 is enough to determine a match of the whole state (in practice, this works with a very high probability)

  Once the value of r is known to the attacker, he proceeds to cheat the gambling house, with a computational effort of 2^i as follows:
  - Pick a value for the odds of winning 2^{i+1}
  - Pick a random value n_3 Craft a nonce as $n_1|n_3$, and compute the result of the SHA-1 length extension of r with n_3, until the last i bits are zero: this will take, on average 2^i trials
  - Send to the gambling house the triple (2^{i+1}, $n_1|n_3$, lots_of_money)

  Note that, for this attack to be carried out in practice, the attacker should take care of considering the mandatory SHA-1 padding, which is added at the end of n_1. The pad, constituted by a single one, a run of zeroes and the length in bits of n_1 expressed as a 64 bit unsigned integer is thus considered by the attacker as a prefix of n_2 and n_3. Note that the fact that the length of the secret is not known does not represent an issue in the forgery process, as sensible message lengths can be bruteforced easily.
- Although SHA-2-256 is built on a Merkle-Damgaard structure, and thus is still vulnerable to length extension attacks, the attacker will have significantly more difficulty in reconstructing the actual value of the hash from the truncated value provided by the gambling house. In particular, as only 140 out of 256 output bits are known to him, the attacker will need to perform a bruteforce effort in the range of $2^{256-140} = 2^{116}$ which is not feasible.

Chapter 5

Multiplicative Groups and Finite Fields

5.1 Multiplicative groups

5.1.1 Characterize $(\mathbb{Z}_{27}^*, \cdot)$

Consider the multiplicative group of $\mathbb{Z}_{27}$, with support $\mathbb{Z}_{27}^*$:

a) Compute its cardinality and the number of its generators

b) Compute a generator for each one of its subgroups

Solution:

a) The elements of $\mathbb{Z}_{27}^*$ must be invertible $\bmod 27$, and thus coprime with 27. We have thus that $|\mathbb{Z}_{27}^*| = \varphi(27) = 3^3 - 3^2 = 18$. $\mathbb{Z}_{27}^*$ will have $\varphi(18) = (2^1 - 2^0)(3^2 - 3^1) = 6$ generators.

b) Given that $|\mathbb{Z}_{27}^*| = 18$ we have that there are non trivial subgroups of $\mathbb{Z}_{27}^*$ with cardinality $2, 3, 6, 9$. After checking that indeed 2 is a generator of $\mathbb{Z}_{27}^*$, i.e. $2^i \neq_{27} 1$ for $i \in \{2, 3, 6, 9\}$, we obtain the generator of the subgroups according to their cardinality s as $2^{\frac{18}{s}} \bmod 27$. Thus we obtain

- $s = 2 \rightarrow 2^9 \bmod 27 = 26 \bmod 27 = -1 \bmod 27$
- $s = 3 \rightarrow 2^6 \bmod 27 = 10 \bmod 27$
- $s = 6 \rightarrow 2^3 \bmod 27 = 8 \bmod 27$
- $s = 9 \rightarrow 2^2 \bmod 27 = 4 \bmod 27$

5.2 Prime Fields

5.2.1 Quick Computations over F_p^*

Compute, exploiting the properties of the underlying multiplicative group, the following values:

- $3^{49^4} \bmod 16$
- $126^{77} \bmod 31$
- $6^{49^4} \bmod 16$

Solution:

- First of all, note that $\gcd(3,16)=1$, thus $3 \in \mathbb{Z}_{16}^*$.
  The size of $\mathbb{Z}_{16}^*$ is $\phi(15)=(3-1)(5-1)=8$, thus $3^8=3 \bmod 16$.
  Consequentially, $3^{49^4} \bmod 16 = 3^{49^4 \bmod 8} \bmod 16 = 3^{1^4 \bmod 8} \bmod 16 = 3^1 \bmod 16 = 3$
- First of all, $126 \bmod 31 = 2$, thus $126^{77} \bmod 31 = 2^{77} \bmod 31$. Then, checking that $\gcd(2,31)=1$ we have that $2 \in \mathbb{Z}_{31}^*$ (actually, since 31 is prime, we could have skipped this check). Note that $|\mathbb{Z}_{31}^*|=30$, thus $2^{77} \bmod 31 = 2^{77 \bmod 30} \bmod 31 = 2^{17} \bmod 31$. Note that $2^5 \bmod 31 = 1$, thus $2^{17} \bmod 31 = 2^{15} * 2^2 \bmod 31 = 4$
- Note that $\gcd(6,16)=2 \neq 1$, thus $6 \notin \mathbb{Z}_{16}^*$. In fact, note that $16k = 6 \cdot \frac{16}{\gcd(6,16)} = 6 \cdot 8 = 48 = 3 \cdot 16 = 0 \bmod 16$. Consequentially, $(6)^4 \bmod 16 = (2 \cdot 3)^4 \bmod 16 = 2^4 \cdot 3^4 \bmod 16 = 0 \cdot 3^4 \bmod 16$, thus $6^{49^4} \bmod 16 = 0$.

5.3 Polynomial Rings

5.3.1 Irreducible and Primitive polynomials

Consider the following polynomial $f(x)=x^4+x+1 \in \mathbb{F}_2[x]$.

- Verify it is both irreducible and primitive
- Given $\beta(x)=x^2+x+1 \in \mathbb{F}_{2^4} \cong \mathbb{F}_2[x]/\langle f(x) \rangle$, compute $\beta(x)^{64^{-1}}$.

Solution:

- The first step is to apply the irreducibility test: in this case, it amounts to verifying that $\gcd(x^4+x+1, x^{2^1}-x)$, and $\gcd(x^4+x+1, x^{2^2}-x)$ are constant (i.e., zero degree polynomials).
  Being $f(x)$ irreducible, it can be used to build a representation of the elements in $\mathbb{F}_{2^4}$.
  If the roots of $f(x)$ are generators of the multiplicative group $\mathbb{F}_{2^4}^*$, then it is also a primitive polynomial for $\mathbb{F}_{2^4}$.
  In order to verify that a root of $f(x)$, say $\alpha \in \mathbb{F}_{2^4} \setminus \mathbb{F}_2$, $f(\alpha)=0 \Leftrightarrow \alpha^4=\alpha+1$, is a primitive elemen it should be checked that $\alpha^h \neq 1$, for all h dividing $|\mathbb{F}_{2^4}^*|$. Recalling that $|\mathbb{F}_{2^4}^*|=15$ α^3 and checking that $\alpha^3 \neq 1$, $\alpha^5=\alpha^4\alpha=(\alpha+1)\alpha \neq 1$, we obtain that $f(x)=x^4+x+1$ is in fact primitive (and thus irreducible).

- $\beta(x)^{64^{-1}} \equiv \beta(x)^{4^{-1}} \equiv \beta(x)^4 \equiv (\beta(x)^2)^2 \equiv (x^4+x^2+1)^2 \equiv (x^2+x)^2 \equiv (x^4+x^2) \equiv x^2+x+1 \equiv \beta(x)$.

5.3.2 Irreducible and Primitive polynomials - 2

Consider the polynomial $P(x)=x^6+x^5+x^4+x+1 \in \mathbb{F}_2[x]$

(a) Show that $P(x)$ is a primitive polynomial for the field $\mathbb{F}_{2^6}$

(b) Compute the values of all the roots of $P(x)$

(c) Compute the value of $(\alpha^3+\alpha)^{1260128}$

Solution:

(a) $P(x)$ is primitive if its roots are generators of the multiplicative group $(\mathbb{F}_{2^6}^*, \cdot)$.
Remember that $\mathbb{F}_{2^6} \cong \mathbb{F}_2[x]/\langle f(x) \rangle \cong \mathbb{F}_2(\alpha)$, where $\alpha \notin \mathbb{F}_2$, $P(\alpha)=0$ and the elements of the field can be represented as $\mathbb{F}_2(\alpha)=\{\theta_5\alpha^5+\ldots+\theta_1\alpha+\theta_0\}$, $\theta_i \in \{0,1\}$, $0 \leq i \leq 5$.

Therefore, $P(x)$ is primitive over $\mathbb{F}_{2^6}$, $n{=}|\mathbb{F}_{2^6}^*|{=}63{=}3^2{\cdot}7$,
iif assuming $P(\alpha){=}0 \Leftrightarrow \alpha^6 \equiv \alpha^5{+}\alpha^4{+}\alpha{+}1$, the following relations hold

$$\begin{cases} \alpha^3 \overset{?}{\neq} 1 \\ \alpha^7 \overset{?}{\neq} 1 \\ \alpha^9 \overset{?}{\neq} 1 \\ \alpha^{21} \overset{?}{\neq} 1 \end{cases} \Rightarrow \begin{cases} \alpha^3 \neq 1 \\ \alpha^7 \equiv \alpha \cdot \alpha^6 \equiv \alpha \cdot (\alpha^5 + \alpha^4 + \alpha + 1) \equiv \alpha^4 + \alpha^2 + 1 \neq 1 \\ \alpha^9 \equiv \alpha^7 \cdot \alpha^2 \equiv \ldots \equiv \alpha^5 + \alpha^2 + \alpha + 1 \neq 1 \\ \alpha^{21} \equiv \alpha^3 \cdot (\alpha^9)^2 \equiv \ldots \equiv \alpha^4 + \alpha^3 + 1 \neq 1 \end{cases}$$

(b) Since $\alpha{\in}\mathbb{F}_{2^6} \setminus \mathbb{F}_2$ is a root of $P(x){\in}\mathbb{F}_2[x]$, the other five roots are conjugate elements of α, i.e.:

$$\begin{aligned} \alpha^{2^1} &\equiv \alpha^2 \\ \alpha^{2^2} &\equiv \alpha^4 \\ \alpha^{2^3} &\equiv \alpha^8 \equiv \alpha^5 + \alpha^3 + \alpha \\ \alpha^{2^4} &\equiv (\alpha^8)^2 \equiv \alpha^{10} + \alpha^6 + \alpha^2 \equiv \ldots \equiv \alpha^3 + \alpha \\ \alpha^{2^5} &\equiv (\alpha^{16})^2 \equiv (\alpha^3 + \alpha)^2 \equiv \alpha^6 + \alpha^2 \equiv \alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1 \end{aligned}$$

(c) $(\alpha^3 + \alpha)^{1260128} \equiv (\alpha^3 + \alpha)^{1260128 \bmod 63} \equiv (\alpha^3 + \alpha)^2 \equiv \alpha^6 + \alpha^2 \Leftrightarrow$
$(\alpha^3 + \alpha)^{1260128} \equiv \alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1$

5.3.3 Irreducible and Primitive polynomials - 3

Consider the finite field $\mathbb{F}_{3^4}$:

Find the number of its irreducible and primitive polynomials.

Determine if $\pi(x) = x^4 + x + 2 \in F_3[x]$ is a primitive polynnomial and show all its roots.

Solution:

Given the finite field at hand, we know that $X^{3^4} - x$ is the product of all the irreducible polynomials over F_3 with degree d, and $d|4$.

5.3.4 Zech Logarithm

Considering a Finite Field $\mathbb{F}_q$, $q{=}p^m$, with p prime, $m{\geq}1$, and a primitive element $\alpha{\in}\mathbb{F}_q^*$, $\text{ord}(\alpha){=}q{-}1$, the *Zech logarithm* relative to the base α is defined as:

$$Z_\alpha(i) \overset{\text{def}}{=} \log_\alpha(1 + \alpha^i) \Leftrightarrow \alpha^{Z_\alpha(i)} \overset{\text{def}}{=} 1 + \alpha^i$$

with the following conventions:

$$\begin{aligned} &\exists\, e{\in}\mathbb{F}_q^* \text{ s.t. } \alpha^e{=}{-}1 \Rightarrow Z_\alpha(e) \overset{\text{def}}{=} \log_\alpha 0{=}{-}\infty \\ &Z_\alpha(-\infty) \overset{\text{def}}{=} 0 \\ &\alpha^{-\infty} \overset{\text{def}}{=} 0 \end{aligned}$$

Note that: $Z_\alpha(\cdot) \in \{0, 1, \ldots, q-2\}$

For sufficiently small finite fields, a table of Zech logarithms allows an especially efficient implementation of all finite field arithmetic in terms of a small number of integer addition/subtractions and table look-ups, through repeatedly employing the following chain of equalities:

$$\alpha^n + \alpha^m = \alpha^n \cdot (1 + \alpha^{m-n}) = \alpha^n \cdot \alpha^{Z(m-n)} = \alpha^{n+Z(m-n)}$$

- Compute the Zech logarithm table for the field $\mathbb{F}_{2^3} \cong \mathbb{F}_2(\alpha)$, $f(\alpha)=0$, where $f(x)=x^3+x^2+1 \in \mathbb{F}_2[x]$ is the primitive polynomial, $\text{ord}(\alpha)=7$.

i	$Z_\alpha(i)=\log_\alpha(1+\alpha^i)$
0	$-\infty$
1	...
...	...
6	...

- Compute $x = \log_\alpha(\alpha^4 + \alpha + 1)$ using the corresponding Zech table

Solution:

- considering $i = 0, 1, \ldots, 6$ we have
  $\alpha^0 = 1, \alpha^1 = \alpha, \alpha^2 = \alpha^2$
  $\alpha^3 = \alpha^2 + 1, \alpha^4 = \alpha^2 + \alpha + 1, \alpha^5 = \alpha + 1, \alpha^6 = \alpha^2 + \alpha$

i	$Z_\alpha(i)=\log_\alpha(1+\alpha^i)$
0	$-\infty$
1	5
2	3
3	2
4	6
5	1
6	4

- $\alpha^4 + \alpha + 1 = \alpha(1 + \alpha^3) + 1 = \alpha^{1+Z(3)} + 1 = \alpha^3 + 1 = \alpha^{Z(3)} = \alpha^2$
  $x = \log_\alpha(\alpha^4 + \alpha + 1) = \log_\alpha(\alpha^2) = 2.$

Chapter 6

Public-Key Cryptosystems

6.1 RSA Cryptosystem

6.1.1 Computing RSA

Consider the RSA modulus $n = p \cdot q = 713 = 23 \cdot 31$

(a) Apply the Miller-Rabin primality test to the factor $p = 23$ employing as bases $a = 3$, $b = 7$.

(b) Given the public exponent $e{=}7 \in \mathbb{Z}^*_{\varphi(n)}$, show the value of the RSA private key, $k_{priv}{=}(p, q, \varphi(n), d)$ and specify every step of the computation.

(c) Apply a Right-to-Left Square & Multiply strategy to compute the ciphertext $c_1 = m^{13} \bmod n$, with $m = 101 \bmod n$ employing a radix-2 encoding of the exponent, and apply a Left-to-Right Square & Multiply strategy to re-compute the same value employing a radix-4 encoding of the exponent.

Solution:

(a) Decompose $p - 1$ as $d2^s$, $d = 11, s = 1$. Check if $a^d \bmod p \neq \pm 1$: $3^{11} \equiv_{23} 1$, $7^{11} \equiv_{23} 22 \equiv_{23} -1$. 23 is prime with probability 4^{-2}.

(b) $\varphi(n) = 22{\cdot}30 = 2^2{\cdot}3{\cdot}5{\cdot}11 = 660$, $\varphi(\varphi(n)) = \varphi(660) = (2^2{-}2)(3{-}1)(5{-}1)(11{-}1) = 160$
$d = e^{\varphi(\varphi(n))-1} \bmod \varphi(n) = 7^{159} \bmod 660 = 283$.

(c) Right-to-Left S&M:
$c_1 \equiv_n m^{13_{\text{dec}}} \equiv_{713} 101^{\langle 1101 \rangle_2} \equiv_{713} (101^{2^3})^1 \cdot (101^{2^2})^1 \cdot (101^{2^1})^0 \cdot (101^{2^0})^1 \equiv_{713} \ldots$
$\equiv_{713} 357$.
To apply the Left-to-Right S&M with a 4-radix expansion of the exponent it is convenient to be able to compute the 4-th power of an integer modulo 713, efficiently;
in addition, we need to pre-compute: $m \equiv_{713} 101$, $m^2 \equiv_{713} 219$, $m^3 \equiv_{713} 16$. Thus,
$c_1 \equiv_n m^{\langle 13 \rangle_{10}} \equiv_{713} 101^{\langle 31 \rangle_4} \equiv_{713} (101^3)^4 \cdot 101 \equiv_{713} 653 \cdot 101 \equiv_{713} 357$.

6.1.2 Computing RSA - 2

Let $n = p \cdot q = 437 = 19 \cdot 23$ be the modulus of an RSA cryptosystem.

- Choose which one of the possible exponents among $e = 3$, $e = 11$, $e = 17$ can be actually employed as RSA public exponent in conjunction with the public modulus $n = 437$.
- Compute the value of the corresponding RSA private key, $k_{priv} = (p, q, \varphi(n), d)$, and specify every passage of the computation.

- Sign the message m=101 (without employing any padding scheme) through applying the Chinese Remainder Theorem to speedup the computation of $s = m^d \bmod n$. Describe each passage of the procedure.

Solution:

- We recall that the public exponent must belong to $\mathbb{Z}^*_{\varphi(n)} \Leftrightarrow 0 < e < n$, $\gcd(e, \varphi(n)) = 1$. Thus considering that $\varphi(n) = (p-1)(q-1) = 18 \cdot 22 = 2^2 \cdot 3^2 \cdot 11 = 396$, the only possible choice is $e = 17$.
- $d = e^{-1} \bmod \varphi(n) \Rightarrow d \equiv_{\varphi(n)} e^{\varphi(\varphi(n))-1} \equiv_{396} 17^{119} \equiv_{396} 17^{\langle 1110111 \rangle_2} \equiv ((((((17)^2 \cdot 17)^2 \cdot 17)^2)^2 \cdot 17)^2 \cdot 17)^2 \cdot 17 \equiv_{396} 233$. The private key of this RSA cryptosystem instance is thus $k_{priv} = (p, q, \varphi(n), d) = (19, 23, 396, 233)$
- The signature computation process goes as follows:

$$m = c^{d \bmod \varphi(n)} \bmod n \Leftrightarrow \begin{cases} m = m_p \bmod p \\ m = m_q \bmod q \end{cases} \text{ where:}$$

$m_p = c^{d \bmod \varphi(p)} \bmod p \equiv_{19} 101^{233 \bmod 18} \equiv_{19} 101^{17} \equiv_{19} 16$ and
$m_q = c^{d \bmod \varphi(q)} \bmod q \equiv_{23} 101^{233 \bmod 22} \equiv_{23} 101^{13} \equiv_{23} 12$

$m \equiv_n \left(M_p \cdot M_p^{'} \cdot m_p + M_q \cdot M_q^{'} \cdot m_q \right) \bmod n$
$M_p = q = 23,\ M_p^{'} = q^{-1} \bmod p = 23^{-1} \bmod 19 \equiv_{19} 4^{-1} \equiv_{19} 5$
$M_q = p = 19,\ M_q^{'} = p^{-1} \bmod q = 19^{-1} \bmod 23 \equiv_{23} -4^{-1} \equiv_{23} -6 \equiv_{23} 17$

$$\Rightarrow m \equiv_{437} (23 \cdot 5 \cdot 16 + 19 \cdot 17 \cdot 12) \equiv_{437} 1840 + 3876 \equiv_{437} 5716 \equiv_{437} 35$$

6.1.3 Mental Poker

Consider an RSA cryptosystem and a user private key $K_{\text{priv}-\text{A}}$=$(p, q, \varphi(n), d)$=$(7, 19, 108, 13)$.

Mental Poker[SRA76] is a protocol by which two parties each of whom distrusts the other can deal each other cards from a deck without either being able to cheat.

The protocol for A to deal B a card goes like this:

1. A and B agree on a set X=$\{x_1, \ldots, x_{52}\}$; x∈$\mathbb{Z}_n$, of random numbers where n=$p \cdot q$, p and q prime and known to both A and B. These numbers represent the deck of cards: x_i represents the i-th card in the deck.
2. A randomly picks s such that $\gcd(s; \varphi(n))$=1, and t such that $s \cdot t$=1 mod $\varphi(n)$ secretly. B does the same for e and f. (i.e., $e \cdot f$=1 mod $\varphi(n)$)
3. A calculates $(x_i)^s$ mod n for i=$1, \cdots, 52$, shuffles the numbers, and sends them to B.
4. B calculates $((x_i)^s)^e$ mod n for i=$1, \cdots, 52$, shuffles the numbers, and sends them to A.
5. A calculates $(((x_i)^s)^e)^t$ mod n for i=$1, \cdots, 52$, A then chooses a card randomly (i.e., picks $(x_j)^e$ where j∈$[1, \cdots, 52]$) and sends it to B.
6. B then takes $((x_j)^e)^f$ mod n. This is the card B has been dealt.

[SRA76] A. Shamir, R. L. Rivest, and L. M. Adleman. *Mental poker.* In D. Klarner, editor, The Mathematical Gardner, pages 3743. Wadsworth, Belmont, California, 1981.

- Find the corresponding public key $K_{\text{pub}-\text{A}}$=(n, e_{A}). Show each step of the computation.

- Emulate the digital signature process on the message M=$\{10101\}_2$=21 mod n, showing each passage of the calculus.
- Consider the scenario shown in the following and compute the minimal number of modular multiplications to compute one round of the "Mental Poker" protocol (Steps from (3) to (6)).

Assume $\varphi(n)\approx n$.

Solution:

- $e, d \in \mathbb{Z}^*_{\varphi(n)}$

  $e=d^{-1} \bmod \varphi(n)=d^{\varphi(\varphi(n))-1} \bmod \varphi(n)=d^{35} \bmod \varphi(n)=$
  $=d^{\{100011\}_2} \bmod \varphi(n)=((((d^2)^2)^2)^2 \cdot d)^2 \cdot d \bmod \varphi(n)$
  $=((((13^2)^2)^2)^2 \cdot 13)^2 \cdot 13 \bmod 108 \equiv_{108}(((61^2)^2)^2 \cdot 13)^2 \cdot 13 \equiv_{108}((49^2)^2 \cdot 13)^2 \cdot 13 \equiv_{108}$
  $\equiv_{108}(25^2 \cdot 13)^2 \cdot 13 \equiv_{108}(85 \cdot 13)^2 \cdot 13 \equiv_{108}(25)^2 \cdot 13 \equiv_{108} 85 \cdot 13 \equiv_{108} 25.$

  $K_{\text{pub}-\text{A}}=(n, e)=(133, 25)$

(b) M=21 mod 133,

$S=Sig_{\text{Kpriv}}(M)=M^d \bmod n=21^{13} \bmod 133 \equiv_{133} 21^{\{1101\}_2} \equiv_{133} (((21)^2 \cdot 21)^2)^2 \cdot 21 \equiv_{133}$
$\equiv_{133}((42 \cdot 21)^2)^2 \cdot 21 \equiv_{133}((84)^2)^2 \cdot 21 \equiv_{133} 72 \cdot 21 \equiv_{133} 98.$

$\langle M, S\rangle=\langle\{10101\}_2, \{1100010\}_2\rangle$

(c) Applying a *Square & Multiply* strategy, the cost of a single modular exponentiation can be stated as: $\lceil log_2 n\rceil-1$ modular squarings plus $0.5 \cdot \lceil log_2 n\rceil-1$ modular multiplications. (average case – the exponent is randomly chosen in $\{1, \ldots, \varphi(n) \approx n\}$).

If the cost of a modular squaring is assumed to be equal to the cost of a modular multiplication then the total cost of a single modular exponentiation can be expressed as $1.5 \cdot \lceil log_2 n\rceil-1$ modular mul.s.

Cost (3)–(6):
$(52+52+52) \cdot 1.5 \cdot \lceil log_2 n\rceil-1 + 1.5 \cdot \lceil log_2 n\rceil-1=157 \cdot (1.5 \cdot \lceil log_2 n\rceil-1)=235.5 \cdot \lceil log_2 n\rceil-157$
modular mul.s.

6.1.4 Factoring n with collateral information

In the following, we consider an RSA-cryptosystem with public key $k_{\texttt{pub}} = \langle n, e\rangle$,
where n=9179 and e=4321. Furthermore, suppose you find a notice saying $\varphi(n^2)$=82390704.

- Show[1] for any two positive integers a, b that $\varphi(ab)=\varphi(a)\varphi(b)$ holds if $\texttt{gcd}(a, b)=1$.
- Show without factoring n that $\varphi(n)$=8976.
- Determine the private key d without factoring n. Show each step of the computation.
- Factor n using $\varphi(n)$.

Solution:

- Recalling the definition: $\varphi(ab) = |\mathcal{X}|$, with $\mathcal{X} = \{x | 1 \leq x < ab, \texttt{gcd}(x, ab) = 1\}|$.
  If $\texttt{gcd}(a, b) = 1$, then each *prime divisor* of ab is a prime divisor of a or (exclusively) it is a prime divisor of b. Therefore $x \in \{y | 1 \leq y < a, \texttt{gcd}(x, a) = 1\}$, or (exclusively) $x \in \{y | 1 \leq y < b, \texttt{gcd}(y, b) = 1\}$, thus

$$\varphi(ab) = \varphi(a) \cdot \varphi(b)$$

[1] Remark: if $\texttt{gcd}(a, b)$=1 then $\texttt{gcd}(ab, c)$=$\texttt{gcd}(a, c)\texttt{gcd}(b, c)$

Alternatively:
If $a = 1$ or $b = 1$, then the claim holds.
Suppose now that $a > 1$ and $b > 1$, and denote:
$\mathcal{A} = \{x | 1 \leq x < a, \gcd(x, a) = 1\}$
$\mathcal{B} = \{y | 1 \leq y < b, \gcd(y, b) = 1\}$
$\mathcal{C} = \{w | 1 \leq w < a\,b,\ \gcd(w, a\,b) = 1\}$.
Then we have that $|A| = \varphi(a)$, $|B| = \varphi(b)$, and $|C| = \varphi(a\,b)$.
The claim will follow through showing that $\mathcal{C}$ has equally many elements as the set $A \times B = \{(x, y) | x \in A, y \in B\}$. Since $\gcd(a, b) = 1$, we can use the CRT noting that the map $\pi(\cdot)$ is bijective:

$$\pi : \mathbb{Z}_{ab} \leftrightarrow \mathbb{Z}_a \times \mathbb{Z}_b, \quad \pi(v) = (v \bmod a, v \bmod b)$$

Observe that $\mathcal{A} \subset \mathbb{Z}_a$, $\mathcal{B} \subset \mathbb{Z}_b$, and $\mathcal{C} \subset \mathbb{Z}_{ab}$.
It holds that $w \in \mathcal{C}$ *if and only if* $\pi(v) \in \mathcal{A} \times \mathcal{B}$, as:

$$\gcd(w, a\,b) = 1 \Leftrightarrow gcd(x, a) = 1 \text{ and } \gcd(y, b) = 1 \Leftrightarrow$$

$$\Leftrightarrow \gcd(x \bmod a, b) = 1 \text{ and } \gcd(y \bmod b, b) = 1$$

.

Alternatively, note that each *prime divisor* of ab is either a prime divisor of a or of b, and those primes which divide both a and b also divide $\gcd(a, b)$. Hence, starting from

$$\varphi(ab) = (ab) \prod_{p|ab} (1 - \frac{1}{p})$$

we can write

$$\frac{\varphi(ab)}{(ab)} = \prod_{p|ab} (1 - \frac{1}{p}) = \frac{\prod_{p|a}(1 - \frac{1}{p}) \cdot \prod_{p|b}(1 - \frac{1}{p})}{\prod_{p|\gcd(\mathtt{m},\mathtt{n})}(1 - \frac{1}{p})} = \frac{\frac{\varphi(a)}{a} \cdot \frac{\varphi(b)}{b}}{\frac{\varphi(d)}{d}}, \text{ where } d = \gcd(a, b)$$

This is equivalent to write

$$\varphi(ab) = \varphi(a)\varphi(b)\frac{d}{\varphi(d)}$$

Therefore, if $\gcd(a, b) = 1$, then $\varphi(ab) = \varphi(a)\varphi(b)$.

- $$n = p \cdot q, \varphi(n^2) = (p^2 - p) \cdot (q^2 - q) = n\varphi(n) \Rightarrow \varphi(n) = \frac{\varphi(n^2)}{n} = 8976$$
- $$\gcd(\varphi(n), e) = 1 \Leftrightarrow \gcd(8976, 4321) = 1 \overset{\text{EEA}}{\Leftrightarrow} \varphi(n) \cdot (-2057) + e \cdot 4273 = 1$$

  $$d = e^{-1} \bmod \varphi(n) \equiv_{8976} 4273$$

- Recall that $n - \varphi(n) + 1 = pq - pq + p + q - 1 + 1 = p + q$ and solve, by substitution the simultaneous equation set given by $n - \varphi(n) + 1 = p + q = 204$ and $pq = 9179$ for p and q as:

$$Z^2 + (8976 - 9179 - 1)\,Z + 9179 = 0 \Rightarrow Z^2 - 204\,Z + 9179 = 0 \Rightarrow \begin{cases} p = 102 - \sqrt{1225} &= 67 \\ q = 102 + \sqrt{1225} &= 137 \end{cases}$$

6.2 Diffie-Hellman Cryptosystem

6.2.1 Prime Field Choice

We use the Diffie-Hellman Key exchange with public group $(\langle g \rangle, \cdot)$, where $g \in \mathbb{Z}_p^*$, $n = |\langle g \rangle|$. Two pairs of private and public keys are: $(\mathtt{K}_{\mathtt{priv,A}}, \mathtt{K}_{\mathtt{pub,A}}) = (a, A)$, with $A = g^a \bmod p$, and $(\mathtt{K}_{\mathtt{priv,B}}, \mathtt{K}_{\mathtt{pub,B}}) = (b, B)$, with $B = g^b \bmod p$.

- Would any of the following p and g be good choices for the Diffie-Hellman algorithm (ignoring the fact that the numbers are too small to be secure)? Motivate your answer.
  1. $p = 179$, $g = 1$
  2. $p = 17$, $g = 2$
  3. $p = 195$, $g = 14$
- Assuming $p = 71$, $g = 7$
  1. Is it possible to find two pairs $a, b \in \{2, \ldots, n-1\}$ such that the common key $\mathtt{K}_{\mathtt{shared}} = 1$?
  2. An attacker knows that the product $A \cdot B = g \bmod p$. Provide two possible pairs (a, b) that satisfy the attackers knowledge.
- What is a good choice for the parameters of a Diffie-Hellman protocol?

Solution:

- 1. Picking $p = 179$, $g = 1$ is clearly wrong, as it is trivial to verify that $g^a = 1$ for any value of a.
  2. Picking $p = 17$, $g = 2$ calls for checking that 2 generates $\mathbb{Z}_{17}^*$. Since $|\mathbb{Z}_{17}^*| = 16$, the values $2^2 \bmod 17$, $2^4 \bmod 17$, and $2^8 \bmod 17$ should not be 1. This is not the case as $2^8 = 256 = 1 \bmod 17$.
  3. The same reasoning applies to $p = 195$, $g = 14$. Note that $195 = 3 \cdot 5 \cdot 13$ where $|\mathbb{Z}_{195}^*| = \varphi(195) = 96 = 2^5 \cdot 3$. Since $14^2 \equiv_{195} 1$, 14 is not a generator of $\mathbb{Z}_{195}^*$
- g is a generator for the group $(\mathbb{Z}_p^*, \cdot)$ $\ldots$ $g^{ab} \equiv_p 1 \Leftrightarrow g^{ab} \equiv_p g^0 \Leftrightarrow ab \equiv_n 0 \Leftrightarrow ab \equiv_{70} 0 \Rightarrow (a, b) = \{(2, 35), (7, 10)\}$
- $g^{a+b} \equiv_p 7 \Leftrightarrow g^{a+b} \equiv_p g^1 \Leftrightarrow a + b \equiv_n 1 \Leftrightarrow a + b \equiv_{70} 1 \Rightarrow (a, b) = \{(7, 64), (2, 69)\}$

6.2.2 Breaking DSS-DSA

Consider a Digital Signature Standard Algorithm (DSS-DSA) on the cyclic group $(G, \cdot)$ where $G = \mathbb{Z}_p^* = \langle\, g \,\rangle$ and $p = 4 \cdot 13 + 1 = 53$, $g = 2$. The order of the working subgroup H is $q = |H| = 13$.

- Find one generator of the working subgroup $H = \langle h \rangle$
- Consider a message m with digest $H(m) = 11_{\mathtt{dec}}$.
  Given the Signature $\langle \gamma, \delta \rangle = \langle 2_{\mathtt{dec}}, 9_{\mathtt{dec}} \rangle$, find the secret key $s \in \mathbb{Z}_q^*$.

Hint-1: $\gamma = (h^l \bmod p) \bmod q$,
$\delta = l^{-1} \cdot (H(m) - s \cdot \gamma) \bmod q$,
$l \in \mathbb{Z}_q^*$ is randomly chosen

Hint-2: Find $l \in \mathbb{Z}_q^*$ from the first part of the signature, γ, through employing a BSGS strategy to extract the discrete logs. Note that $l = \log_h \bar{\gamma} \bmod p$, where $\bar{\gamma}$ can be any value in the set $\{\gamma = 2, \gamma + q = 15, \gamma + 2q = 28, \gamma + 3q = 41\}$. Try to compute the dlog through picking the aforementioned values in the written order (from left to right), the first time you find out that the dlog exists you will also find the correct value of l. The secret key s can be derived from the 2nd part of the signature as: $s = \gamma^{-1} \cdot (l \cdot \delta - H(m)) \bmod q$

Solution:

Consider a Digital Signature Standard Algorithm (DSS-DSA) on the cyclic group $(G, \cdot)$ where $G = \mathbb{Z}_p^* = \langle g \rangle$ and $p = 4 \cdot 13 + 1 = 53$, $g = 2$. The order of the working subgroup H is $q = |H| = 13$.

- *Find one generator of the working subgroup $H = \langle h \rangle$*

  $h = g^{\frac{p-1}{q}} \bmod p \equiv_{53} 2^{\frac{52}{13}} \equiv_{53} 16$

- *Consider a message m with digest $H(m) = 11_{dec}$.*
  Given the Signature $\langle \gamma, \delta \rangle = \langle 2_{dec}, 9_{dec} \rangle$, find the secret key $s \in \mathbb{Z}_q^*$.

  $q = |H| = |\langle h \rangle| = 13$, generator $h = 16 \bmod 53$

  $\bar{\gamma} = h^l \bmod p$, if $l = (i \cdot \lceil \sqrt{q} \rceil + j) \bmod q$, with $0 \leq i, j \leq \lceil \sqrt{q} \rceil$, then

  $$\underbrace{\bar{\gamma} \cdot \left(h^{-\lceil \sqrt{q} \rceil}\right)^i}_{\text{Giant-Steps}} = \underbrace{h^j}_{\text{Baby-Steps}}$$

  $h^{-\lceil \sqrt{q} \rceil} \bmod p = 16^{-4} \bmod 53 \equiv_{53} 16^{53-1} 16^{-4} \equiv_{53} 16^{48} \equiv_{53} 36$
  (or $h^{-\lceil \sqrt{q} \rceil} \bmod p = (16^{-1})^4 \bmod 53 \equiv_{53} 10^4 \equiv_{53} 36$)

  Baby Steps

j:	0	1	2	3	4
$h^j = 16^j$:	1	16	44	**15**	28

  $\bar{\gamma} \in \{\gamma = 2, \gamma + q = 15, \gamma + 2q = 28, \gamma + 3q = 41\}$
  Giant Steps

i:	0	1	2	3	4
$(h^{-m})^i = 36^i$:	1	36	24	16	46
$2 \cdot 36^i$	2	19	48	32	39
$15 \cdot 36^i$	**15**	–	–	–	–
$28 \cdot 36^i$	–	–	–	–	–
$41 \cdot 36^i$	–	–	–	–	–

  $i = 0$, $j = 3$, matching value: $15 \Rightarrow$
  $\mathbf{l} = (\mathbf{i} \cdot \lceil \sqrt{\mathbf{q}} \rceil + \mathbf{j}) \bmod \mathbf{q} = (\mathbf{0} \cdot \mathbf{4} + \mathbf{3}) \bmod \mathbf{13} \equiv_{\mathbf{13}} \mathbf{3}$
  $\mathbf{s} = \gamma^{-\mathbf{1}} \cdot (\mathbf{H(m)} - \mathbf{l} \cdot \delta) \bmod \mathbf{q} \equiv_{\mathbf{13}} \mathbf{2^{-1}} \cdot (\mathbf{11 - 3 \cdot 9}) \equiv_{\mathbf{13}} \mathbf{7} \cdot (\mathbf{-16}) \equiv_{\mathbf{13}} \mathbf{-21} \equiv_{\mathbf{13}} \mathbf{5}$

6.3 Elliptic Curve Cryptosystems

6.3.1 Elliptic Curve Characterization - (I)

Consider the elliptic curve

$$\mathbb{E}(\mathbb{F}_5) : Y^2 = X^3 + b$$

- For which b does this equation describe a non-singular elliptic curve over the field $\mathbb{F}_5$?
- For which b both points $(3, 1)$ and $(4, 4)$ are on $\mathbb{E}(\mathbb{F}_5)$?

Now, let $b = 3$:

- Calculate all points on $\mathbb{E}(\mathbb{F}_5)$.
- Show that point $\mathtt{P} = (1, 2)$ generates the group $(\mathbb{E}(\mathbb{F}_5), +)$.

Solution:

- $4 \cdot 0^3 + 27b^2 \neq 0 \bmod 5 \Rightarrow b = \{1, 2, 3, 4\}$
- $\nexists b$
- $(1, 2)$, $(1, 3)$, $(2, 1)$, $(2, 4)$, $(3, 0)$, $\mathcal{O}$
- $n = |\mathbb{E}(\mathbb{F}_5)| = 6 = 2 \cdot 3$.
  $\mathtt{P} = (1, 2)$

  $[2]\mathtt{P} \neq \mathcal{O}$
  $\lambda \equiv_5 3 \cdot 4^{-1} \equiv_5 2, \quad x_{[2]\mathtt{P}} = \lambda^2 - 2x_{\mathtt{P}} \equiv_5 2, \quad y_{[2]\mathtt{P}} = -y_{\mathtt{P}} + \lambda(x_{\mathtt{P}} - x_{[2]\mathtt{P}}) \equiv_5 1$
  $[2]\mathtt{P} = (2, 1) \neq \mathcal{O}$.

  $[3]\mathtt{P} = [2]\mathtt{P} + \mathtt{P} \neq \mathcal{O}$
  $\lambda \equiv_5 3 \cdot 4^{-1} \equiv_5 -1, \quad x_{[3]\mathtt{P}} = \lambda^2 - x_{[2]\mathtt{P}} - x_{\mathtt{P}} \equiv_5 3, \quad y_{[3]\mathtt{P}} = -y_{\mathtt{P}} + \lambda(x_{\mathtt{P}} - x_{[3]\mathtt{P}}) \equiv_5 0$
  $[3]\mathtt{P} = (3, 0) \neq \mathcal{O}$.

6.3.2 Elliptic Curve Characterization - (2)

Consider an elliptic curve cryptosystem defined over the elliptic curve $\mathbb{E}(\mathbb{Z}_{11})$ with equation $y^2 = x^3 + 1$ over $\mathbb{Z}_{11}$.

(a) What is the order of the additive group $(\mathbb{E}(\mathbb{Z}_{11}),\ +)$?

(b) What is the sum of the points (2, 3) and (5, 4)?

(c) Describe the encryption and decryption functions of the Elliptic Curve ElGamal cryptosystem.

Solution:

(a) $|\mathbb{E}(\mathbb{Z}_{11})|$=12

x, y:	0	1	2	3	4	5	6	7	8	9	10
$x^3 + 1 \bmod 11$:	1	2	9	6	10	5	8	3	7	4	0
$y^2 \bmod 11$:	0	1	4	9	5	3	3	5	9	4	1

Punti sulla curva	
(0, 1)	(0, 10)
(2, 3)	(2, 8)
(5, 4)	(5, 7)
(3, 5)	(3, 6)
(9, 2)	(9, 9)
(10, 0)	$\mathcal{O}$

(b) $\mathtt{S}(x_{\mathtt{S}}, y_{\mathtt{S}}) = \mathtt{P}(x_{\mathtt{P}}, y_{\mathtt{P}}) + \mathtt{Q}(x_{\mathtt{Q}}, y_{\mathtt{Q}})$=(2, 3)+(5, 4)
$\lambda \equiv_{11} (y_{\mathtt{Q}} - y_{\mathtt{P}})(x_{\mathtt{Q}} - x_{\mathtt{P}})^{-1} \equiv_{11} (4-3)(5-2)^{-1} \equiv_{11} 4$o
$x_{\mathtt{S}} = \lambda^2 - x_{\mathtt{P}} - x_{\mathtt{Q}} \equiv_{11} 16 - 2 - 5 \equiv_{11} 9$
$y_{\mathtt{S}} = \lambda(x_{\mathtt{P}} - x_{\mathtt{S}}) - y_{\mathtt{P}} \equiv_{11} 4(2-9) - 3 \equiv_{11} 2$
$\mathtt{S}(x_{\mathtt{S}}, y_{\mathtt{S}})$=(9, 2)

(c) see lectures ...

6.3.3 Faulty RNG and ECDSA

Consider a computing platform having a bootloader, that before loading and transferring control to the proper binary program $m_1 \in \{0,1\}^*$, at first checks the digital signature of the binary and only runs it only if the signature is correct.

Assume that the employed digital signature primitive is the Elliptic Curve Digital Signature Algorithm (ECDSA) on the cyclic group $(G,+)$ where $G{=}(\mathbb{E}(\mathbb{F}_{17}),+)$, with $\mathbb{E}(\mathbb{F}_{17})$: $y^2{=}\ x^3{+}2x{+}11$, $n{=}|G|{=}11$, and generator $\mathrm{P}{=}(x_\mathrm{P}, y_\mathrm{P}){=}(4,10)$. The EC-DSA implementation under exam has a critical flaw as it employs a fixed "nonce" r for the signature computation of every binary program.
In particular, the following information are known.

- $m_1{\in}\{0,1\}^*$, $e_1 = \text{SHA-1}(m) \bmod n \equiv_{11} 13$, $\text{Sign}^{\text{EC-DSA}}_{k_{priv}}(m_1){=}(k,\, z_1){=}(5,4)$
- $m_2{\in}\{0,1\}^*$, $e_2 = \text{SHA-1}(m_2) \bmod n \equiv_{11} 11$, $\text{Sign}^{\text{EC-DSA}}_{k_{priv}}(m_2){=}(k,\, z_2){=}(5,10)$

Hint: the *EC-DSA Signature Algorithm* works as follows: $k_{priv} \leftarrow (s)$, $s{\in}\mathbb{Z}_n$ and $k_{pub} \leftarrow (n, \mathrm{P}, [s]\mathrm{P})$

1. $r \overset{\text{Random}}{\leftarrow} \{1,\ldots,n-1\},\ \gcd(r,n) = 1$
2. $[r]\mathrm{P} = (x_1,\, y_1)$, $k \leftarrow x_1 \bmod n$. if $k = 0$ then go to step 1.
3. $e \leftarrow \text{SHA-1}(m)$
4. $z \leftarrow r^{-1}(e + s \cdot k) \bmod n$. if $z = 0$ then go to step 1.
5. $\text{Sign}^{\text{EC-DSA}}_{k_{priv}}(m) = (k,\, z)$

- Show how to derive the EC-DSA private key in such a way to nullify the trusted boot process.

Solution:

$$\left\{ \begin{array}{l} z_1 \cdot r \equiv_n e_1 + s \cdot k \\ z_2 \cdot r \equiv_n e_2 + s \cdot k \end{array} \right. \Rightarrow \left\{ \begin{array}{l} 4 \cdot r \equiv_{11} 13 + s \cdot 5 \\ 10 \cdot r \equiv_{11} 11 + s \cdot 5 \end{array} \right. \Rightarrow \left\{ \begin{array}{l} r \equiv_{11} 7 \\ s \equiv_{11} 3 \end{array} \right.$$

Chapter 7

Fast Arithmetics, Discrete Logs and Factoring

7.1 Montgomery Multiplication

Assume to be working in the Montgomery domain: $(\widetilde{\mathbb{Z}}_p, +, \times), p = 17$

- Report the definition of the Montgomery Multiplication and the smallest admissible value for the Montgomery Radix: R

- Compute a pair of integer values R', p' that satisfy the relation: $\gcd(R, p)=R\,R'-p\,p'=1$, showing each step of the procedure

- Compute the Montgomery multiplication $C = A \times B$, where $A = 16_{\text{dec}}$ and $B = 5_{\text{dec}}$, assuming a binary encoding of the operands.

Solution:

- We recall that the definition of the Montgomery Multplication is $a, b \in (\widetilde{\mathbb{Z}}_p, +, \times)$, $\text{MonPro}(a, b) = a \times b \stackrel{\text{def}}{=} a \cdot b \cdot R^{-1} \bmod p$. Consequentially, the smallest admissible value for R is obtained as: $t = \lceil \log_2 p \rceil = \lceil \log_2 17 \rceil = 5$, $R > p$, $gcd(R, p) = 1 \Rightarrow R = 2^t = 32$

- Exploiting the relation $\gcd(R, n)=R\,R'-n\,n'=1$ we obtain that $\gcd(32, 17)=32\,(8)-17\,(-15)=1$. Consequentially, the desired values are: $R' = R^{-1} \bmod p \Rightarrow R'=32^{-1} \bmod 17 \equiv_{17} 13^{15} \equiv_{17} 8$, and $p' = p^{-1} \bmod R \Rightarrow p'=17^{-1} \bmod 32 \equiv_{32} 17^{30} \bmod 32 \equiv_{32} 17 \equiv_{32} -15$.

- $p = 17_{\text{dec}} = \langle 10001 \rangle_2, \qquad p'_0 = (p' \bmod 2) = 1$
  $B = \langle 00101 \rangle_2$
  $A = \langle 10000 \rangle_2$

00000	+
00000	$A_0B = \langle 00000 \rangle_2$
00000	+
00000	$t\,p = (p'_0 x_0)\,p = \langle 00000 \rangle_2$
00000	perform a right-shift of 1 bit
00000	
00000	+
00000	$A_1B = \langle 00000 \rangle_2$
00000	+
00000	$t\,p = (p'_0 x_0)\,p = \langle 00000 \rangle_2$
00000	perform a right-shift of 1 bit
00000	
00000	+
00000	$A_2B = \langle 00000 \rangle_2$
00000	+
00000	$t\,p = (p'_0 x_0)\,p = \langle 00000 \rangle_2$
00000	perform a right-shift of 1 bit
00000	
00000	+
00000	$A_3B = \langle 00000 \rangle_2$
00000	+
00000	$t\,p = (p'_0 x_0)\,p = \langle 00000 \rangle_2$
00000	perform a right-shift of 1 bit
00000	
00000	+
00101	$A_4B = \langle 00101 \rangle_2$
00101	+
10001	$t\,p = (p'_0 x_0)\,p = \langle 10001 \rangle_2$
10110	perform a right-shift of 1 bit
01011	

$C = \langle 01011 \rangle_2 = 11_{\text{dec}} < p \Rightarrow C = \mathsf{MonPro}(16, 5) = 11 \bmod 17$

Validation:

$C = \mathsf{MonPro}(16, 5) \stackrel{\text{def}}{=} 16 \cdot 5 \cdot R^{-1} \bmod p \equiv_{17} 80 \cdot 15^{-1} \equiv_{17} 12 \cdot 8 \equiv_{17} 11_{\text{dec}}$

7.1.1 Montgomery Multiplication in radix-4

Assume to work into the Montgomery domain: $(\mathbb{Z}_N, +, \times)$, $N = 21$.
Compute the Montgomery multiplication $C = A \times B \bmod N$, where $A = 16_{\text{decimal}} = 100_4$ and $B = 18_{\text{decimal}} = 102_4$ are Radix-4 encoded values in the Montgomery domain. Show every step of the procedure, performing all the computations in radix-4.

Solution:

Radix $b = 4$, $\lceil \log_4 N \rceil = 3$, $R = b^3 = 64$,
$\gcd(R, N) = RR' - NN' = 1 \Leftrightarrow \gcd(64, 21) = 64(1) - 21(3) = 1$,
$R' \stackrel{\text{def}}{=} R^{-1} \bmod N = 1 \bmod 21 = 1 = 1_4$,
$N' \stackrel{\text{def}}{=} N^{-1} \bmod R = 3 \bmod 64 = 3 = 3_4$; $N'_0 = 3$

$B = \langle B_2 B_1 B_0 \rangle_b = \langle 102 \rangle_b$
$A = \langle A_2 A_1 A_0 \rangle_b = \langle 100 \rangle_b$

$x = \langle \mathbf{000} \rangle_b$	+
000	$A_0 B = 0 \cdot \langle 102 \rangle_b$
000	+
000	$t\,N = (N'_0 x_0 \bmod b)\,N = 0$
000	perform a right-shift of 1 digit
000	+
000	$A_1 B = 0 \cdot \langle 102 \rangle_b$
000	+
000	$t\,N = (N'_0 x_0 \bmod b)\,N = 0$
000	perform a right-shift of 1 digit
000	+
102	$A_2 B = 1 \cdot \langle 102 \rangle_b$
102	+
222	$t\,N = (N'_0 x_0 \bmod b)\,N = 42_{decimal} = 222_b$
330	perform a right-shift of 1 digit
$\mathbf{033}_b$	

$C = \langle 033 \rangle_4 = 15_{dec} < N \Rightarrow C = 15_{dec} \times 18_{dec} = 15_{dec} \in (\mathbb{Z}_N, +, \times)$

Validation: $C{=}16_{dec} \times 18_{dec} \stackrel{\text{def}}{=} 16 \cdot 18 \cdot R^{-1} \bmod N{=}16 \cdot 18 \cdot 1 \bmod 21{=}15{\equiv_{21}}15$

7.2 Factoring

7.2.1 Pollard's $P-1$

Consider the RSA modulus $n{=}p{\cdot}q{=}899$ Apply the Pollard's P-1 factorization method to compute the two factors p and q (assuming $p{<}q$). Consider the factor p being B-power smooth, with $B{=}6$, while the factor q is not.

Solution:

$a = 2^{B!} \bmod n = 2^{2\cdot3\cdot4\cdot5\cdot6} \bmod 899 \equiv_{899} (2^{24})^{5\cdot6} \equiv_{899} 78^{5\cdot6} \equiv_{899} 807^6 \equiv_{899} 342$

$\mathbf{gcd(a-1, n)}$=gcd$(341, 899)$= gcd$(217, 341)$=gcd$(124, 217)$= gcd$(93, 124)$=gcd$(31, 93)$=**31**

$n/31{=}29$

$p = 29,\ q = 31$

7.2.2 Fermat's Method

Apply the Fermat's method to factor the following RSA public modulus: $n = 437$. Show either the pseudo-code of the employed algorithm or each passage of the procedure.
Hint: Note that a simple test to detect if a number in decimal representation is a perfect square can be done remembering that the last two decimal digits of a perfect square are: within the set $\{00, e1, e4, 25, o6, e9\}$, where 'e' stands for an even digit and 'o' for an odd digit.

Solution:

$n = 437$ is an odd integer, therefore there exist two numbers x, y s.t. $n = x^2 - y^2 \Rightarrow$ if $\gcd(x-y, n) \neq 1$ then you have a non-trivial factor of n

$x \leftarrow \lceil \sqrt{n} \rceil = \lceil \sqrt{437} \rceil,\ x{=}21$
$y \leftarrow x^2 - n,\ y = 4$

y is a perfect square, therefore gcd$(x-y, n)$=gcd$(23-4, 437)$=gcd$(19, 437) = 19$

$n = p \cdot q = 437 = 19 \cdot 23$

7.2.3 Pollard Rho

Apply the Pollard's ρ method to factor the following RSA public modulus: $n = p \cdot q = 629$. Assume $f(x) = x^2 + 1 \bmod n$ as "random-walking" function. Show every passage of the computation.

Solution:

Picking randomly as a starting point for the Pollard's Rho Starting point $x_0 = 2$

$\mathbf{i}$	$\mathbf{x = x_i}$	$\mathbf{y = x_{2i}}$	$\lambda = \mathbf{gcd}(\lvert\mathbf{x - y}\rvert, \mathbf{n})$
0	2	2	1
1	5	26	1
2	26	418	1
3	48	529	37

$q = 37$, $p = n/q = 17$

7.3 Discrete Logarithms

7.3.1 Pohlig-Hellmann method - ex1 -

Consider the following relation: $11 \equiv 2^x \bmod 13$. Compute the discrete logarithm $x \equiv_{\varphi(13)} \log_2^{\mathrm{D}} 11$ applying the Pohlig-Hellman method.

Solution:

We are considering the algebra of the group $(\mathbb{Z}_{13}^*, \cdot)$,
with $|\mathbb{Z}_{13}|$=$\varphi(13)$=12=$2^2 \cdot 3$ elements.

$$x \equiv_{\varphi(13)} \log_2^{\mathrm{D}} 11 \quad \Leftrightarrow \quad \begin{cases} x \equiv x_1 \bmod 2^2 \\ x \equiv x_2 \bmod 3 \end{cases} \quad \Rightarrow$$

$$x \equiv_{12} x_1 \cdot 3 \cdot (3^{-1} \bmod 4) + x_2 \cdot 4 \cdot (4^{-1} \bmod 3) \Rightarrow x \equiv_{12} 9x_1 + 4x_2$$

The Pohlig-Hellman algorithm compute $(x_1 \bmod p_1^2)$, with p_1=2, as follows:
let the 2-radix expansion of $x_1 \bmod 2^2$ be $x_1 = l_0 + l_1 2$ and denote as: g=2 the base of the logarithm, β=11, n=12 the order of the group
$\eta = g^{\frac{n}{p_1}} \equiv_{13} 2^6 \equiv_{13} 12$
$\gamma_0 = 1$, $\delta_0 \equiv (\beta\gamma_0^{-1})^{\frac{n}{p_1}} \equiv_{13} 11^6 \equiv_{13} 12$.
Knowing that $\delta_0 = (g^x\gamma_0^{-1})^{\frac{n}{p_1}} \equiv g^{x_1 \frac{n}{p_1}} \Rightarrow \delta_0 \equiv (g^{\frac{n}{p_1}})^{l_0}$, we have
$\delta_0 \equiv_{13} \eta^{l_0} \Leftrightarrow 12 \equiv_{13} 12^{l_0}$, therefore: $l_0 = 1$.
$\gamma_1 = \gamma_0 \cdot g^{l_0 p_1^0} \equiv_{13} 2$, $\delta_1 \equiv (\beta\gamma_1^{-1})^{\frac{n}{p_1^2}} \equiv_{13} (11 \cdot 2^{-1})^3 \equiv_{13} (11 \cdot 7)^3 \equiv_{13} 12$.
Knowing that $\delta_1 = (g^x\gamma_1^{-1})^{\frac{n}{p_1^2}} \equiv (g^{x_1 - l_0})^{\frac{n}{p_1^2}} \Rightarrow \delta_1 \equiv (g^{\frac{n}{p_1}})^{l_1}$, we have
$12 \equiv_{13} \eta^{l_1} \Leftrightarrow 12 \equiv_{13} 12^{l_1}$, therefore: $l_1 = 1$.
$x_1 = 1 + 1 \cdot 2 = 3$.

The Pohlig-Hellman algorithm compute $(x_2 \bmod p_2^1)$, with p_2=3, as follows:
let the 3-radix expansion of $x_2 \bmod 3$ be $x_2 = l_0$ and denote as: g=2 the base of the logarithm, β=11, n=12 the order of the group
$\eta = g^{\frac{n}{p_2}} \equiv_{13} 2^4 \equiv_{13} 3$
$\gamma_0 = 1$, $\delta_0 \equiv (\beta\gamma_0^{-1})^{\frac{n}{p_2}} \equiv_{13} 11^4 \equiv_{13} 3$.
from $\delta_0 \equiv (g^{\frac{n}{p_2}})^{l_0}$, we have
$\delta_0 \equiv_{13} \eta^{l_0} \Leftrightarrow 3 \equiv_{13} 3^{l_0}$, therefore: $l_0 = 1$.
$x_2 = 1$.

$x \equiv_{12} 9x_1 + 4x_2 \equiv_{12} 27 + 4 \equiv_{12} 7$.

Validation: $2^x \stackrel{?}{\equiv}_{13} 11$, taking x=7, it is true that $2^7 \equiv_{13} 11$.

7.3.2 Pohlig-Hellmann method - ex2 -

Consider the following cyclic group: $G=(\mathbb{Z}_{54}^*, \cdot)=\langle 29 \rangle$, with order $n=\varphi(54)=18=2 \cdot 3^2$.
Compute the discrete logarithms:

- $x \equiv_n \log_g^{\mathrm{D}} \alpha$, with g=29, β=11
- $y \equiv_n \log_g^{\mathrm{D}} \beta$, with g=29, β=12
- $z \equiv_n \log_h^{\mathrm{D}} \rho$, with h=19, ρ=13

Solution:

- Concerning the computation of
  $x \equiv_n \log_g^{\mathrm{D}} \alpha$, with g=29, β=11 we note that $\langle g \rangle = G$ and $\alpha \in G$ as gcd$(54, \alpha)$=1.
  Thus, the logarithm x exists.

  $$x \equiv_{18} \log_{29}^{\mathrm{D}} 11 \quad \Leftrightarrow \quad \begin{cases} x \equiv_2 x_1 \\ x \equiv_{3^2} x_2 \end{cases} \quad \Rightarrow$$

  $$x \equiv_{18} x_1 \cdot 9 \cdot (9^{-1} \bmod 2) + x_2 \cdot 2 \cdot (2^{-1} \bmod 9) \Rightarrow x \equiv_{18} 9 \cdot x_1 + 10 \cdot x_2$$

  The Pohlig-Hellman algorithm compute $(x_1 \bmod 2)$ as follows:
  let the 2-radix expansion of x_1 be $x_1 = l_0$ and denote as: g=29 the base of the logarithm, α=11, n=18 the order of the group
  $\eta = g^{\frac{n}{2}} \equiv_{54} 29^9 \equiv_{54} -1$
  $\gamma_0 = 1$, $\delta_0 \equiv (\alpha \gamma_0^{-1})^{\frac{n}{2}} \equiv_{54} 11^9 \equiv_{54} -1$.
  $l_0 \equiv_2 \log_\eta \delta_0 \equiv_2 \log_{-1}(-1) \equiv_2 1$, Thus $x_1 = 1$

  The Pohlig-Hellman algorithm compute $(x_2 \bmod 3^2)$ as follows:
  let the 3-radix expansion of x_2 be $x_2 = l_0 + l_1 \cdot 3$ and denote as: g=29 the base of the logarithm, α=11, n=18 the order of the group
  $\eta = g^{\frac{n}{3}} \equiv_{54} 29^6 \equiv_{54} 37$
  $\gamma_0 = 1$, $\delta_0 \equiv (\alpha \gamma_0^{-1})^{\frac{n}{3}} \equiv_{54} 11^6 \equiv_{54} 37$.
  $l_0 \equiv_3 log_\eta \delta_0 \equiv_3 \log_{37}(37) \equiv_3 1$.
  $\gamma_1 = g^{l_0} = 29$, $\delta_1 \equiv (\alpha \gamma_1^{-1})^{\frac{n}{3^2}} \equiv_{54} (29^{-1} \cdot 11)^2 \equiv_{54} (41 \cdot 11)^2 \equiv_{54} 37$.
  $l_1 \equiv_3 \log_\eta \delta_1 \equiv_3 \log_{37}(37) \equiv_3 1$, Thus, $x_2 = 1 + 1 \cdot 3 = 4$.

  $x \equiv_{18} 9 \cdot x_1 + 10 \cdot x_2 \equiv_{18} 9 \cdot 1 + 10 \cdot 4 \equiv_{18} 49 \equiv_{18} 13$.
  Validation: $g^x \stackrel{?}{\equiv}_{18} \alpha$, taking x=13, it is true that $29^{13} \equiv_{18} 11$
- Concerning the computation of
  $y \equiv_n \log_g^{\mathrm{D}} \beta$, with g=29, β=12 we note that $\langle g \rangle = G$, but $\beta \notin G$ as gcd$(54, \beta) \neq 1$.
  Thus, the logarithm y does not exist.
- Concerning the computation of
  $z \equiv_n \log_h^{\mathrm{D}} \rho$, with h=19, ρ=13 we note that $\langle h \rangle$ is a generator of the subgroup $H = \langle h \rangle = \{1, 19, 19^2 \equiv_{54} 37\}$ which has order equal to 3.
  Therefore, even if the argument of the logarithm ρ=13$\in G$ as gcd$(54, \rho)$=1, ρ does not belong to the group generated by h.
  Thus, the logarithm z does not exist.

  It would have been more meaningful to ask the value of $w \equiv_3 \log_h^{\mathrm{D}} \rho$.
  indeed in this case $(\rho \bmod 3) \equiv 1$ and $w \equiv_3 0$

Chapter 8

Protocols

8.1 TLS

8.1.1 Ciphersuite choices

Consider a TLS ciphersuite as a tuple $(\mathcal{A}_{kex}, \mathcal{A}_{auth}, \mathcal{A}_{sym}, \mathcal{A}_{hash})$

(a) Which choice between *(Diffie-Hellman-2048, RSA-1024, AES-128-CBC, SHA-2-256)* and *(Diffie-Hellman-Ephemeral-512, RSA-2048, AES-128-CBC, SHA-2-256)* is the one providing the highest security margin?

(b) During the TLS key exchange, is it possible for an active attacker to alter the value of the session nonce, setting it to an arbitrary value decided by him?
Does this action get detected before the end of the TLS handshake?

(c) Is there any difference between picking *(Diffie-Hellman-Ephemeral-2048, RSA-2048, None, SHA-2-256)* and *(Diffie-Hellman-2048, RSA-2048, None, SHA-2-256)* as a TLS ciphersuite?

Solution:

(a) Despite providing perfect forward secrecy, the second ciphersuite is weaker than the first one, as it is possible to break DHE-512 with a relatively small computational effort, thus deriving the session key. The first choice, despite employing a suboptimal choice for key lenghts (the RSA keylength voids the effort of the 2048 bit DH key)

(b) It is possible for an active attacker to tamper precisely with the value of the session nonce, however, as the final message in the TLS handshake includes the hash of all the handshake messages, such tampering will be detected on the server side as the hash computed locally will not match with the one computed by the client.

(c) No. The temporary session key which is needed to provide confidentiality on the transported data is not used, as there is no symmetric bulk encryption. (Both choices are valid ones in TLS, although, obviously, not among the ideal ones).

Remember: In the TLS jargon, Diffie-Hellman-Ephemeral (DHE) differs from the static Diffie-Hellman (DH) in the way that static Diffie-Hellman key exchanges always use the same Diffie-Hellman (half)keys.

When a key exchange uses Ephemeral Diffie-Hellman a temporary DH key is generated for every connection and thus the same shared key is never used twice. This enable *Perfect Forwards Secrecy*, which means that if the private key of the server gets leaked, past communication is still secure.

8.1.2 SSLv3 IV Flaw

While reviewing an implementation of AES-128-CBC, you discover that it simply uses the last ciphertext block from the previously encrypted message as the `IV` value C_0 for encrypting the next message. The implementation's authors argue that as long as the `IV` of the very first message was chosen uniformly at random, all resulting subsequent ciphertext blocks will also be distributed uniformly at random, thus providing a secure solution[1].
Discuss the robustness of this CBC implementation with respect to a *Chosen Plaintext Attack*.
Solution:

The described implementation cannot be recommended as a secure one because it enables an adversary (who record every data transmission) with the ability to verify guesses on the plaintext blocks of the encrypted communication.

Given a plaintext split up in series of a blocks $P_1, P_2, \dots P_L$, the CBC mode of operation applies the symmetric key cipher of choice (say $\mathbb{E}_k(...)$) to compute the ciphertext blocks in the following way: $C_0 = \texttt{IV}$, $C_i = \mathbb{E}_k(P_i \oplus C_{i-1})$, with $1 \leq i \leq L$.

The usual practice of applying a CBC mode of operation is to choose a new, **unpredictable** random `IV` for every message that is encrypted.

The implementation mentioned above chooses all but the initial `IV` by setting it equal to the final ciphertext block of the preceding encrypted message. This enable a passive adversary to mount a chosen-plaintext attack (CPA) to verify his guess as to whether a particular plaintext block has a particular value.

Say an adversary who has observed the ciphertext $C_0, \dots, C_{j-1}, C_j, \dots C_L$ wants to determine whether plaintext block P_j, with $1 \leq j < L$, is equal to some string P^*.

Note that the adversary knows the Initialization Vector that will be used when encrypting the next message, that is C_L.

The adversary causes the sender to encrypt a message M whose initial block P'_1 is equal to $P'_1 = C_{j-1} \oplus C_L \oplus P^*$.
The first ciphertext block C'_1 will be computed as: $C'_1 = \mathbb{E}_k(P'_1 \oplus C_L) = \mathbb{E}_k(C_{j-1} \oplus P^*)$.

However, the adversary also know that $C_j = \mathbb{E}_k(P_j \oplus C_{j-1})$.
This implies that $C'_1 = C_j$ iff $P_j = P^*$.

In this way, an adversary can verify a guess P^* for the value of any plaintext block P_j.

Note that, if the adversary knows that P_j *is one of two possible values* then the adversary can determine the actual value of P_j by executing the above attack a single time.

Similarly, if the adversary knows that P_j is one out of N possible values, then by repeating the above attack $N/2$ times (on average) he can determine the actual value of P_j.

[1] The initial `IV` used by SSL 3.0 (TLS 1.0) was a (pseudo)random string generated and shared during the initial handshake phase, subsequent `IV`s were chosen following the deterministic pattern previously described.

8.2 Custom Protocols Analysis

8.2.1 (Un)Safe communication with symmetric key only

To establish a way to communicate safely over an insecure channel, Alice and Bob both generate a random string employed as a secret parameter (s_a and s_b respectively). All the encryption and decryption transformations are realized via a common $\oplus$ operations. The communication protocol between Alice and Bob performs the following operations:

1. Alice encrypts the message m with s_a obtaining $c = E_{s_a}(m)$ and sends it to Bob
2. Bob encrypts the received message c with his secret s_b obtaining $d = E_{s_b}(d)$
3. Alice decrypts the received message d and sends back to Bob $e = D_{s_a}(d)$
4. Bob decrypts the received message e obtaining the original plaintext m

Discuss the security of the protocol.

Solution:

The protocol is fundamentally broken: an eavesdropper Eve obtains the following values

- $c = m \oplus s_a$
- $d = m \oplus s_a \oplus s_b$
- $e = m \oplus s_b$

It is quite evident that $c \oplus d = s_b$ and $d \oplus e = s_a$ thus revealing both the secrets at the cost of a simple eavesdropping of a single communication!

8.2.2 Secure Password Storage

Willing to prevent an attacker which comes into possess of the storage of our passwords, we decided to store only the hashes of our passwords on permanent storage. As we are aware of common Time-to-Memory Trade-Off (TMTO) attacks such as "Rainbow Tables", we chose a salted hashing technique operating as follows (consider p to be the plaintext password, and s to be a 128-bit random salt):

$$h = \mathrm{MD5}^{1000}(p) \oplus s$$

storing the pair (h, s) on disk.

- Is there anything wrong with this choice?

A friend of ours suggested us to employ the following salted hash function:

$$h = \mathrm{MD5}^{1000}(p||s)$$

storing (h, s).

- Is this choice better or worse than ours?
- If our users employ 8 characters passwords chosen over the lowercase alphanumeric characters, how many bits long should be the salt to prevent an attacker able to compute 2^{70} passwords from employing a TMTO strategy?
- Does the previous amount of random salt prevent an attacker from bruteforcing a single password?

Solution:

- Yes, the chosen salted hash strategy allows an attacker to easily remove the effect of the salt through simply computing $h \oplus s$ and then lead any TMTO attack of his choice.
- The choice is a better one: there is no trivial way to remove the salt from the hash.
- Considering that an 8 characters password over lowercase alphanumeric characters lies in a $(26+10)^8 \approx 2^{41}$ wide password space, we need to employ a salt longer than $70-41=29$ bits to prevent an attacker to mount a TMTO attack precomputing all the possible hashes for all the possible salts.
- No: the cost for the computation of a single password is 2^{41} operations, thus it is feasible to bruteforce it.

8.2.3 Time-To-Memory tradeoff for bruteforcing - (I)

Concerning password storage schemes, and password choice policy, the following three are proposed:

1. Passwords are chosen to be 5-character strings composed of lowercase latin letters and numbers, and are stored as $\texttt{SHA256}(\texttt{password} \oplus s)||s$, with s chosen as a 5 byte array filled with random content

2. Passwords are chosen to be 5-character alphanumeric strings (lowercase and uppercase latin letters allowed), and stored as $\texttt{SHA256}(\texttt{password}||s)||s$, with s chosen as a 5 byte array filled with random content

3. Passwords are built concatenating 5 random words out of the first edition of the Oxford Dictionary, which contained $\approx 256 \times 2^{10}$ words, and stored as $\texttt{SHA256}(\texttt{password})$

Analyze the feasibility of a password recovery attack lead by an attacker endowed with at most 1 TiB storage, and planning on breaking a large amount of passwords, following the same choice policy and storage. Evaluate the computational effort required in all cases in terms of SHA256 executions, highlighting which password choice-storage strategy is the best one.[2]

Solution:

1. Analyzing the password storage policy leaves the attacker with two strategies: either compute $2^{25.85}$ SHA256 hashes to exhaustively search for a password, given a specific salt, or build a rainbow table for all the possible strings resulting from the combination of the password and the salt. Since the combination of salt and password yields a random 5 byte array, this amounts to building a rainbow table for a 2^{40} password-space. Given the limitation of 1 TiB, and the fact that a SHA256 hash takes 32B, it is possible to store 2^{34} chains in the rainbow table, thus requiring 2^6 SHA256 computations per look-up. It is thus feasible to perform both a direct exhaustive search against a single password, and to employ a TMTO strategy.
2. This strategy yields a $2^{29.75}$ SHA256 computation to perform an exhaustive search of the entire password-space. However, trying to apply a TMTO strategy would need to build a rainbow table encompassing a $2^{69.75}$ password-and-salt space, which, given the aforementioned storage capabilities, would result in a $2^{35.75}$ SHA256 computations effort to perform a single lookup. As a consequence, with this password choice-storage strategy, the attacker is better off performing exhaustive password searches for each password, as a TMTO strategy is both less efficient and borderline feasible given the $2^{69.75}$ computations effort required to build the rainbow table.
3. Picking 5 random words out of the Oxford dictionary yields $(256 \times 2^{10})^5 = 2^{90}$ possible passwords, which is well beyond feasibility for both an exhaustive search and the construction of the corresponding rainbow table.

[2] Recall that $\log_2(36) \approx 5.17$, $\log_2(62) \approx 5.95$

Time-To-Memory tradeoff for bruteforcing - (II)
Threefish is a block cipher with a 256-bit (32 bytes) sized block and allows the use of three different key lengths: 256, 512 and 1024 bits. Consider the three following password hashing strategies relying on Threefish:

(a) Split the password p in 9-byte wide blocks $p_0 \dots p_n$, and generate n+1 23-byte unpredictable random salt blocks s_i, $0 \leq i \leq n$. Store on the disk a sequence of blocks $h_i = \text{THREEFISH}_{p_i||s_i}(p_i||s_i)$ followed by the concatenation of all the s_i, i.e.: $h_0||h_1||\dots||h_n||s_0||s_1||\dots||s_n$.

(b) Split the password p in 9 bytes wide blocks $p_0 \dots p_n$, and generate n+1 23-byte salt blocks s_i obtained as the concatenation of a 1-byte random value for each s_i. Store on the disk a sequence of blocks $h_i = \text{THREEFISH}_{p_i||s_i}(p_i||s_i)$ followed by the concatenation of all the s_i.

(c) Split the password p in 32-byte wide blocks $p_0 \dots p_n$, obtain a 1024-bit unpredictable random salt s and store on the disk a sequence of blocks h_i, with
$h_0 = s$ and $\forall i > 0, h_i = \text{THREEFISH}_s(p_{i-1} \oplus h_{i-1})$

Assume you have 16 TiB of disk space available (average access time 1ms), 32 GiB of RAM (average access time 0.1μs) and you are able to compute 2^{32} THREEFISH encryptions or decryptions per second on a good GPU.

State how much computational effort (i.e., how much time) is required to break the proposed password hashing schemes with the best possible technique, and whether this is practically feasible considering only fully lowercase passwords.

Solution:

(a) This password strategy effectively cuts down the computational effort to perform a bruteforce to the one required to find each p_i exhaustively. The keyspace of a 9 character lowercase password is $\approx 26^9 = (2^{4.7})^9 = 2^{42}$, thus amounting to around 2^{10} seconds in computation time i.e., ≈ 20 minutes, which is feasible. The presence of a 23-byte (184-bit) unpredictable salt however prevents time-to-memory-tradeoffs (TMTOs) to be employed.

(b) This password strategy suffers from a key recovery attack in the same fashion as the previous one. However, considering the fact that the possible values for the salt are limited to 2^8, it is possible to apply a TMTO strategy to reduce the time to break a password after a precomputation effort of 2^{18}s, that is, around 72 hours. Employing a rainbow table strategy, we know that besides the initial computation, the worst time to break a password is given by l encryptions and l rainbow table lookups. Assuming that the chains are stored in such a fashion to allow $O(log(n))$ access, the following efforts are required: for disk stored rainbow tables, 2^{38} chains can be stored, resulting in a chain length of 2^{12}, and thus $\approx 1\mu$s in computation time and $2^{12} * 38 * 1$ms (≈ 15.5s)for the disk lookups. For RAM stored rainbow tables, 2^31 chains are stored, resulting in a 2^{19} chain length yielding a ≈ 122ms computation and a $2^{19} * 31 * 0.1\mu$s lookup time (≈ 1.6s). The best strategy to break this password hashing thus retrieves one password in 1.6 seconds employing RAM housed rainbow tables.

(c) This password hashing strategy is effectively encrypting the password employing THREEFISH in CBC mode, using the salt as both the IV and the key. Since the salt is stored in cleartext together with the hashed password, retrieving the password is just a matter of decrypting it in negligible time.

8.3 Commitment schemes

8.3.1 Washing Dishes

Alice and Bob need to decide who is going to do the dishwashing for the next week, and decide to do so via tossing a coin and transmitting the result via e-mail. To this end, Alice has devised the following commitment scheme:

1. Alice commits to the value $v \in \{\texttt{head}, \texttt{tails}\}$ through sending to Bob $c = v \oplus \texttt{rnd}$, where `rnd` is a random value as long as v and completely uncorrelated with v
2. Alice claims to have committed to the value revealing $(v, \texttt{rnd})$

- Alice claims that this scheme is very nice, as, from her point of view, Bob will not be able to cheat. Is Alice right? Support or dismiss her commenting the binding and/or concealing properties of the scheme.

Bob points out to Alice that he does not feel confident in using the previous scheme, as claiming that a simple $\oplus$ is not enough to provide security and proposes the following scheme, for the sake of Alice's good:

1. Alice commits to the value $v \in \{\texttt{head}, \texttt{tails}\}$ through sending to Bob $c = \texttt{SHA256}(k||v)$, where `k` is a 1024-bit random string
2. Alice claims to have committed to the value revealing $(v, \texttt{k})$

- Is Bob right? Support him or disprove him analyzing the binding and concealing properties of the scheme.

Solution:

- Alice is right: the current scheme is effectively information theoretically concealing, thus Bob will never learn the committed value before Alice reveals `rnd`
- The scheme is still information theoretically concealing, thus Bob is lying on the improved warranties for Alice. However, analyzing the binding properties of the scheme we note that the computational effort needed for Alice to cheat (i.e. circumvent the binding property) has dramatically risen. In particular, in the first scheme, Alice could trivially derive two values $\texttt{rnd}_h$, $\texttt{rnd}_t$ which, reveled together with the desired value, matched a commitment. In the second scheme proposed by Bob, She needs to find a collision on the output of SHA-256, which is currently computationally unfeasible.

Chapter 9

Side Channel Attacks

9.1 Passive Side Channel Attacks

9.1.1 Simple and Differential Power Analysis

- Which relation between the key values and the program execution is leveraged by Simple Power Analysis (SPA) and which by Differential Power Analysis (DPA)?

- Which one of the following two code snippets is vulnerable to SPA? Which one to DPA? (Assume that the key value is represented by the variable k, and the value of m can be chosen by the attacker)

```
1 s=m;
2 a=1;
3 for(i=0;i<ceil(log(n));i++){
4   set=((k&(2<<i))>>i);
5   to_mult= s*set + (!set)*1;
6   a=a*to_mult;
7   s=s*s;
8 }
```

(a) Snippet 1

```
1 s=m;
2 a=1;
3 for(i=0;i<ceil(log(n));i++){
4   if(k&(2<<i)){
5     a=a*s;
6   }
7   s=s*s;
8 }
```

(b) Snippet 2

Solution:

- Simple Power Analysis relies on the dependence of the control flow of the program from the key values to deduce the value of the secret key from the instantaneous power consumption of the circuit during the encryption. By contrast, Differential Power Analysis exploits dependencies of the data flow of the program from the values of the secret key.
- Of the two code snippets, the second one is vulnerable to SPA, as the `if` statement relies directly on the value of a bit of the key to drive the execution of the `a*s` multiplication, while the first code snippet relies on a sequence of arithmetic instructions which are always run, regardless of th e key bits. Both snippets however are vulnerable to DPA as the value of the key bit influences directly the data flow, and they are directly employed together with a value known to the attacker (the message).

9.2 Fault Attacks

9.2.1 Faulty RSA-CRT signatures

A smartcard using an RSA schoolbook implementation for digital signatures contains a CRT-private key, consisting of $\mathtt{K}_{\mathtt{CRT-priv}} = \langle\ p, q, d_p \equiv_{\varphi(p)} d, d_q \equiv_{\varphi(q)} d, u \equiv_q p^{-1}, v \equiv_p q^{-1}\ \rangle$. The corresponding public key consists of $\mathtt{K}_{\mathtt{pub}} = (n, e)$, with n=10807, e=17. Somebody runs the smartcard twice in order to sign the message $m = 123$ on both a regular and glitchy power supply. The fist time the smartcard returns the signature $s = 851$, the second one it returns a the faulty signature $\tilde{s} = 1053$.

- Find the values of the CRT-private key[1]

Solution:

- p=$\mathtt{gcd}(n, \tilde{s} - s)$=$\mathtt{gcd}(10807, 1053 - 851)$=$\mathtt{gcd}(10807, 202)$=$\mathtt{gcd}(202, 101)$=101.
  $q = n/p = 107$.
  $\varphi(n) = 100 \cdot 106 = 10600 = 2^3 \cdot 5^2 \cdot 53,\ \varphi(\varphi(n)) = 4160$,
  $d \equiv_{\varphi(n)} e^{\varphi(\varphi(n))-1} \equiv_{10600} 17^{4159} \equiv_{10600} 9353$.
  $\mathtt{K}_{\mathtt{CRT-priv}} = \langle\ p, q, d_p \equiv_{\varphi(p)} d, d_q \equiv_{\varphi(q)} d, u \equiv_q p^{-1}, v \equiv_p q^{-1}\ \rangle =$
  $= \langle\ p$=$101, q$=$107, d_p \equiv_{\varphi(101)} 53, d_q \equiv_{\varphi(107)} 25, u \equiv_{107} 89, v \equiv_{101} 17\ \rangle$

[1] The complete solution does not require to write all the intermediate values of modular exponentiations.

Appendix A

Useful notions for fast approximations

It is useful, when computing the size of keyspaces, to be able to express their size in terms of powers of two. To this end recall that: $2^{10} = 1024 \approx 1000 = 10^3$ (stated first by Shannon, in *A Mathematical Theory of Communication*) and memorize the first ten powers of two.
Proceed as follows to obtain a quick ballpark estimate of the closest power of two to a given number n.

1. Consider only the three most significant digits of n zeroing out the others
2. Express n as $h10^{3k}$ with the largest value of k you can use leaving h integer
3. Note that $n = h10^{3k} \approx h2^{10k}$, which leaves you with the need to estimate only h in terms of powers of two
4. Estimate the value of h using the powers of two you memorized $\Rightarrow\ h \approx 2^g$
5. Combine the estimates obtaining $n = 2^{g+10k}$

Moreover, these notions do come in handy:

- by Stirling's approximation, $\ln(n!) \approx n\ln(n) - n + O(\ln(n))$
- Memorizing the constant required for the base change $e \to 2$ in logarithms helps: $\log_2(a) = \frac{\ln(a)}{\ln(2)} \approx 1.44\ln(a)$
- Recalling that $\log_2(3) \approx 1.6$ and $\log_2(5) \approx 2.3$ is also handy
- Given the size of the English alphabet, it may come in handy to know that $log_2(26) \approx 4.7004 \approx 4.7$

Appendix B

Summary of linear algebra: Determinant and Matrix Inversion

In linear algebra, the *Laplace expansion* is an expression for the determinant ($\det(A)$ or $|A|$) of an $n \times n$ square matrix A that is a weighted sum of the determinants of n sub-matrices of A, each of size $(n-1) \times (n-1)$. The Laplace expansion is of theoretical interest as one of several ways to view the determinant, as well as of practical use in determinant computation.

Theorem .1 (Laplace Expansion)**.** *Given a $n \times n$ square matrix A,*

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

denote as $A^{(i,j)}$ the sub-matrix obtained through removing from A its i-th row and j-th column. The **determinant** *of the matrix, $\det(A)$, is equivalent to the computation of the following recursive procedure:*

- $\det(A) = a_{i,j}$, *if the order of the matrix is unitary, i.e. $A = [a_{i,j}]$*
- $\det(A) = \sum_{k=1}^{n} (-1)^{k+i} a_{i,k} \det(A^{(i,k)})$ for a chosen row $1 \leq i \leq n$,
  if the order of the matrix is greater than 1

  Equivalently,
  $\det(A) = \sum_{k=1}^{n} (-1)^{k+j} a_{k,j} \det(A^{(k,j)})$ for a chosen column $1 \leq j \leq n$,
  if the order of the matrix is greater than 1

Observation .1. *Given a square matrix A of order n, the determinant of the sub-matrix obtained through removing the i-th row and the j-th column is called (i,j)-***minor** *of A and denoted by $M_{i,j}$=$\det(A^{(i,j)})$*

Observation .2. *Given a square matrix A of order n, the product $(-1)^{i+j} M_{i,j}$ is called (i,j)-***cofactor** *of A and denoted by $C_{i,j}$=$(-1)^{i,j} M_{i,j}$=$(-1)^{i+j} \det(A^{(i,j)})$*

Observation .3. *The Laplace expansion for the computation of the determinant of a square matrix A of order n can be written as the sum of cofactors of any row or column of the matrix multiplied by the entries that generated them.*

- *Cofactor expansion along the i-th row:*

$$det(A) = a_{i,1}C_{i,1} + a_{i,2}C_{i,2} + a_{i,3}C_{i,3} + \ldots + a_{i,n}C_{i,n} = \sum_{k=1}^{n} a_{i,k}C_{i,k}$$

- *Cofactor expansion along the j-th column:*

$$det(A) = a_{1,j}C_{1,j} + a_{2,j}C_{2,j} + a_{3,j}C_{3,j} + \ldots + a_{n,j}C_{n,j} = \sum_{k=1}^{n} a_{k,j}C_{k,j}$$

Example .1. *Given a square matrix A of order $n = 3$,*

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

the computation of the (2,3)-minor $M_{2,3}$ removes from A the 2nd row and the 3rd column:

$$A^{(2,3)} = \begin{bmatrix} a_{1,1} & a_{1,2} & \square \\ \square & \square & \square \\ a_{3,1} & a_{3,2} & \square \end{bmatrix}$$

and computes the determinant of the sub-matrix according to the recursive procedure of the Laplace expansion (applied on the first row)

$$M_{23} = \det(A^{(2,3)}) = \begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{3,1} & a_{3,2} \end{vmatrix} = a_{1,1}\,(-1)^{1+1}\,\left|a_{3,2}\right| + a_{1,2}\,(-1)^{1+2}\,\left|a_{3,1}\right|$$

$M_{23} = a_{1,1}a_{3,2} - a_{1,2}a_{3,1}$

Note: the vertical lines are an equivalent notation for $\det(matrix)$.

Definition .1 (Cofactor Matrix)**.** Given a square matrix A of order n, the matrix of cofactors is the square matrix whose (i, j) entry is the cofactor $C_{i,j}$ of A.

For instance, if

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

the cofactor matrix of A is

$$C = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,n} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n,1} & C_{n,2} & \cdots & C_{n,n} \end{bmatrix}$$

where $C_{i,j} = (-1)^{i+j}\,M_{i,j} = (-1)^{i+j}\,\det(A^{(i,j)}) = (-1)^{i+j}\,|A^{(i,j)}|$ is the cofactor bound to the element $a_{i,j}$.

Theorem .2 (Inverse matrix)**.** *Given a square matrix A of order n, if* $\det(A) \neq 0$ *(equivalently, A is non-singular), the inverse matrix A^{-1} is computed through transposing the related matrix of cofactors and multiplying each of its elements by the inverse of the determinant of A*

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,n} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n,1} & C_{n,2} & \cdots & C_{n,n} \end{bmatrix}^{T} = \frac{1}{\det(A)} \begin{bmatrix} C_{1,1} & C_{2,1} & \cdots & C_{n,1} \\ C_{1,2} & C_{2,2} & \cdots & C_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ C_{1,n} & C_{2,n} & \cdots & C_{n,n} \end{bmatrix}$$

Techniques for computing a determinant quickly over binary matrices

Since the matrices involved in the LFSR state/feedback network structure reconstruction are defined over $(\mathbb{Z}_2, +, \cdot)$, the arithmetics involved is simplified and allows fast computations. In particular:

- The only valid value for the determinant of a nonsingular matrix A is 1. Consequentially, once you checked that the matrix is nonsingular, the inverse matrix is simply the transposed of the cofactor matrix, as the term $\frac{1}{det(A)}$ is equal to 1
- Recalling that the determinant of a matrix is zero if and only if two or more of the vectors composing it are in linear dependence, it is possible to avoid computing the determinant of a matrix if you notice that there are two columns or two rows containing the same values
- Due to the nature of the Laplace expansion, it is faster to compute the determinant of a matrix if you choose a row (or column) with a high number of zeros, as they will cancel out the contribution of their cofactor, thus avoiding the need to compute it
- As we are computing on $(\mathbb{Z}_2, +, \cdot)$ (i.e. modulo 2), rembember that $1 = -1$, i.e. there are no signs to take care about