Solutions Report

Modern Cryptology (CS641)

Computer Science and Engineering

Level 7

Team: **team58**
Ayush Bansal (160177)
Aman Deep Singh (15807084)
Gunjan Jalori (170283)

Date: June 15, 2020

Contents

1 Chapter 7 (WECCAK, WEAK-KECCAK)

This assignment is based on a variant of **KECCAK** hash function which we call **WECCAK**.
The description of WECCAK is as follows:

1. Input to the hash function is a message $M \in \{0,1\}*$.
2. M is padded with minimum number of zeros such that bit-length of padded message is a multiple of 184.
3. The padded message is divided into blocks of 184 bits. Let's call them $M_1, M_2, \ldots, M_k$.
4. A state in WECCAK hash function is a $5 \times 5 \times 8$ 3-dimensional array.
5. Initial state S contains all zeros.
6. The first message block M_1 is appended with 16 zeros to form M_1' and is XORed with S, similar to KECCAK.
7. This state is given as input to a function F (defined below) and let's call its output as O_1. The output F is also a $5 \times 5 \times 8$ 3-dimensional array.
8. The second message block M_2 is appended with 16 zeros to form M_2' and is XORed with O_1 and is given as input to F.
9. This is continued for k times.
10. The output of WECCAK is the initial 80 bits of O_k, similar to KECCAK.

For the above WECCAK hash function, we have to answer the following questions:

1. Compute the inverse of χ and θ.
2. Claim about the security of WECCAK with $F = R \cdot R$, give a preimage, collision and second preimage attack.

here $R = \chi \cdot \rho \cdot \pi \cdot \theta$ (θ, ρ, π, χ are the same defined in KECCAK).

1.1 Thinking about the Problem

KECCAK (in our case WECCAK) hash functions has 3 parameters:

- r is the bitrate, it's value is 184 here.
- c is the capacity, it's value is 16 here.
- n is the output length, it's value is 80 here.

KECCAK-p permutation is denoted by KECCAK-p$[b, n_r]$

- b is length of input string (called width of permutation), it is 200 for WECCAK.
- n_r is number of rounds of internal transformation, it is 2 for our custom WECCAK function since $F = R \cdot R$.

The b bit input string can be represented as a $5 \times 5 \times w$ 3-dimensional array, $w = 8$ in our case.
A lane in a state S is denoted by $S[x][y]$ which is a substring $S[x][y][0]|S[x][y][1]|\ldots|S[x][y][w-1]$ where $|$ is the concatenation function, hence the length of lane is w.

Usually in KECCAK, the internal transformation consists of 5 step mappings θ, ρ, π, χ and ι which act on a state, but in our case there are only 4 step mappings, since $R = \chi \cdot \rho \cdot \pi \cdot \theta$, note the absence of ι. Let our initial matrix be S, then mappings will be as follows:

1. θ:

$$S'[x][y][z] = S[x][y][z] \oplus E[x,z]$$

and

$$E[x,z] = P[(x+1)\ mod\ 5][(z-1)\ mod\ 8] \oplus P[(x-1)\ mod\ 5][z]$$

where $P[x][z]$ is the parity of a column, i.e.,

$$P[x][z] = \bigoplus_{i=0}^{4} S[x][i][z]$$

2. ρ: simply rotates each lane by a predefined (known) value

$$S'[x][y] = S[x][y] << r[x][y]$$

where << is bitwise rotation towards MSB of the 8 bit word. The values of $r[x][y]$ are predefined constants.

3. π: interchanges lanes

$$S'[y][2x+3y] = S[x][y]$$

4. χ: non-linear operation

$$S'[x][y][z] = S[x][y][z] \oplus ((S[(x+1)\ mod\ 5][y][z] \oplus 1) \cdot S[(x+2)\ mod\ 5][y][z])$$

We will make use of the above notations and definitions in our solution ahead.

1.2 Solution

Our solution will be broadly divided into 2 parts:

1. We will analyze each of the 4 steps in the internal transformation and compute the inverse of each.
2. Analyse the security of WECCAK by performing 3 attacks:
   - Preimage Attack
   - Collision Attack
   - Second Preimage Attack

R is the internal transformation step for us, it includes 4 computations, as follows:

$$R = \chi \cdot \rho \cdot \pi \cdot \theta$$

Now, we want to compute R^{-1} (inverse of R), it will be denoted as the following composition of functions (or steps):

$$R^{-1} = \theta^{-1} \cdot \pi^{-1} \cdot \rho^{-1} \cdot \chi^{-1}$$

The problem statement requires us to find χ^{-1} and θ^{-1}, so we will talk about these 2 in detail and briefly about computation of π^{-1} and ρ^{-1}.

1.2.1 Computing χ^{-1}

Firstly, let's analyse the χ step and compute χ^{-1}.
An important observation here is that χ is a row operation, i.e. it does not depend on the value of y and z. Using this fact, we are going to make the following claim.

Claim 1.1
If the output of χ for an entire row is known, i.e. $\chi([a_0,a_1,a_2,a_3,a_4]) = [b_0,b_1,b_2,b_3,b_4]$, then we have

$$a_i = b_i \oplus (1 \oplus b_{i+1}) \cdot (b_{i+2} \oplus (1 \oplus b_{i+3}) \cdot b_{i+4})$$

Proof 1.1
For the χ step, in a row, we have atmost 5 values, so the equation for the computation of b_i's will be as follows:

$$b_i = a_i \oplus (1 \oplus a_{i+1}) \cdot a_{i+2} \tag{1}$$
$$b_{i+1} = a_{i+1} \oplus (1 \oplus a_{i+2}) \cdot a_{i+3} \tag{2}$$
$$b_{i+2} = a_{i+2} \oplus (1 \oplus a_{i+3}) \cdot a_{i+4} \tag{3}$$
$$b_{i+3} = a_{i+3} \oplus (1 \oplus a_{i+4}) \cdot a_i \tag{4}$$
$$b_{i+4} = a_{i+4} \oplus (1 \oplus a_i) \cdot a_{i+1} \tag{5}$$

We know the values of $\{b_0,b_1,b_2,b_3,b_4\}$, and we need to find a_i's.

$$\begin{aligned}
a_i &= b_i \oplus (1 \oplus a_{i+1}) \cdot a_{i+2} && \text{From eq. (1)} \\
a_i &= b_i \oplus (1 \oplus b_{i+1} \oplus (1 \oplus a_{i+2}) \cdot a_{i+3}) \cdot a_{i+2} && \text{From eq. (2)} \\
a_i &= b_i \oplus (1 \oplus b_{i+1}) \cdot a_{i+2} \oplus (1 \oplus a_{i+2}) \cdot a_{i+2} \cdot a_{i+3} && \\
a_i &= b_i \oplus (1 \oplus b_{i+1}) \cdot a_{i+2} && (1 \oplus a_{i+2}) \cdot a_{i+2} = 0 \\
a_i &= b_i \oplus (1 \oplus b_{i+1}) \cdot (b_{i+2} \oplus (1 \oplus a_{i+3}) \cdot a_{i+4}) && \text{From eq. (3)} \\
a_i &= b_i \oplus (1 \oplus b_{i+1}) \cdot (b_{i+2} \oplus (1 \oplus b_{i+3} \oplus (1 \oplus a_{i+4}) \cdot a_i) \cdot a_{i+4}) && \text{From eq. (4)} \\
a_i &= b_i \oplus (1 \oplus b_{i+1}) \cdot (b_{i+2} \oplus (1 \oplus b_{i+3}) \cdot a_{i+4}) && (1 \oplus a_{i+4}) \cdot a_{i+4} = 0 \\
a_i &= b_i \oplus (1 \oplus b_{i+1}) \cdot (b_{i+2} \oplus (1 \oplus b_{i+3}) \cdot (b_{i+4} \oplus (1 \oplus a_i) \cdot a_{i+1})) && \text{From eq. (5)}
\end{aligned}$$

Rewriting the last equation above, we get the following result:

$$\begin{aligned}
a_i = b_i &\oplus (1 \oplus b_{i+1}) \cdot (b_{i+2} \oplus (1 \oplus b_{i+3}) \cdot b_{i+4}) \\
&\oplus (1 \oplus b_{i+1}) \cdot (1 \oplus b_{i+3}) \cdot (1 \oplus a_i) \cdot a_{i+1}
\end{aligned} \tag{6}$$

Let's look at the 2nd part of the equation above:

$$\begin{aligned}
&(1 \oplus b_{i+1}) \cdot (1 \oplus b_{i+3}) \cdot (1 \oplus a_i) \cdot a_{i+1} \\
&= ((1 \oplus a_{i+1}) \oplus (1 \oplus a_{i+2}) \cdot a_{i+3}) \cdot ((1 \oplus a_{i+3}) \oplus (1 \oplus a_{i+4}) \cdot a_i) \cdot (1 \oplus a_i) \cdot a_{i+1} && \text{From eq. (2) and eq. (4)} \\
&= (1 \oplus a_{i+1}) \cdot (1 \oplus a_{i+3}) \cdot (1 \oplus a_i) \cdot a_{i+1} \\
&\quad \oplus (1 \oplus a_{i+1}) \cdot (1 \oplus a_{i+4}) \cdot (1 \oplus a_i) \cdot a_{i+1} \cdot a_i \\
&\quad \oplus (1 \oplus a_{i+2}) \cdot (1 \oplus a_{i+3}) \cdot (1 \oplus a_i) \cdot a_{i+3} \cdot a_{i+1} \\
&\quad \oplus (1 \oplus a_{i+2}) \cdot (1 \oplus a_{i+4}) \cdot (1 \oplus a_i) \cdot a_{i+3} \cdot a_i \cdot a_{i+1} && \text{Expanding the equation} \\
&= 0 && (1 \oplus a_i) \cdot a_i = 0 \text{ for all the } 4 \text{ terms}
\end{aligned}$$

Putting back the above result in eq. (6), we get the following result:

$$a_i = b_i \oplus (1 \oplus b_{i+1}) \cdot (b_{i+2} \oplus (1 \oplus b_{i+3}) \cdot b_{i+4})$$

The proof of Claim 1.1 gives us the result for the value of χ^{-1}

$$\begin{aligned}
S[x][y][z] = S'[x][y][z] &\oplus (S'[(x+1) \bmod 5][y][z] \oplus 1) \cdot \\
&(S'[(x+2) \bmod 5][y][z] \oplus (S'[(x+3) \bmod 5][y][z] \oplus 1) \cdot \\
&S'[(x+4) \bmod 5][y][z])
\end{aligned} \tag{7}$$

1.2.2 Computing θ^{-1}

Nextly, we analyze the θ step and compute θ^{-1}.
The θ step proceeds as follows, it works on columns:

$$S'[x][y][z] = S[x][y][z] \oplus E[x][z] \tag{8}$$

and

$$E[x][z] = P[(x+1)\ mod\ 5][(z-1)\ mod\ 8] \oplus P[(x-1)\ mod\ 5][z] \tag{9}$$

where $P[x][z]$ is the parity of a column, i.e.,

$$P[x][z] = \bigoplus_{y=0}^{4} S[x][y][z] \tag{10}$$

Instead of the following the above procedure repeatedly for each element, we will try to accomplish the above result by computing it differently.
Let's look at it from another perspective, take a look at the following figure:

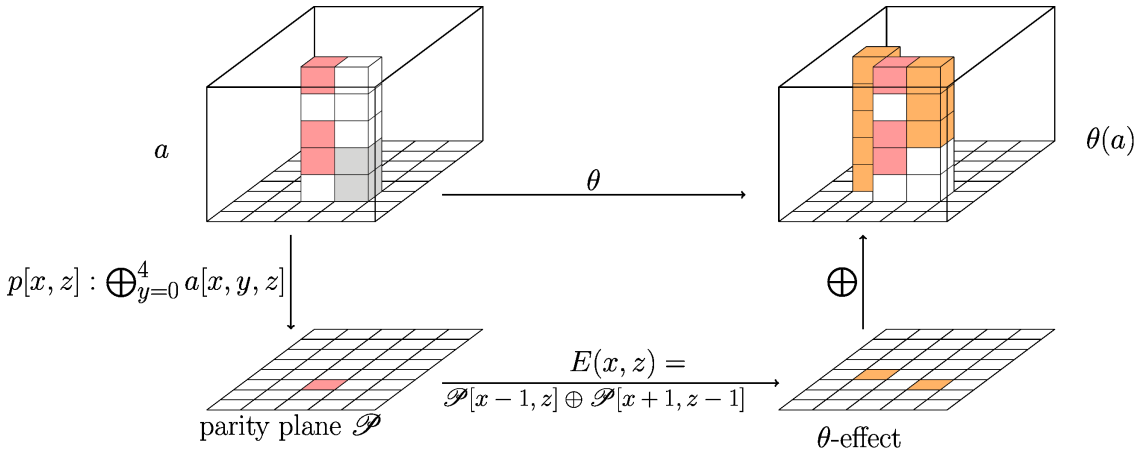

Figure 1: KECCAK THETA

The stepwise computation process of the θ step will be as follows:

1. We flatten the KECCAK Structure along the column, computing $P[x][z]$ 2-dimensional array by calculating parity of each column.
2. We apply the θ-effect, i.e. compute $E[x][z]$ 2-dimensional array by using the following formula

$$E[x][z] = P[(x+1)\ mod\ 5][(z-1)\ mod\ 8] \oplus P[(x-1)\ mod\ 5][z]$$

3. Re-generate the 3-dimensional array by XORing the original column with the corresponding $E[x][z]$ to get the updated column, i.e.

$$S'[x][y][z] = S[x][y][z] \oplus E[x][z]$$

The $P[x][z]$ and $E[x][z]$ are $5 \times w$ (in our case 5×8) 2-dimensional arrays.

To compute the θ^{-1}, we are going to prove the following claim using the result we discussed above.

Claim 1.2
θ is invertible, i.e. knowing all the values of S', we can find all the values of S.

Proof 1.2
Instead of seeing $P[x][z]$ (parity) of the input state S as $5 \times w$ bit-array, we can see it as a $5w$-bit vector W, i.e. we have a unique mapping σ between x, z and i^{th} coordinate in the $5w$-bit vector

$$W : i = x \times w + z$$

Similarly, $E[x][z]$ can be seen as a $5w$-bit vector as well, let's call it W'.
The θ-effect can be seen as the result of a matrix multiplication:

$$W' = W \cdot T \tag{11}$$

In the above equation, T is a $5w \times 5w$ 2-dimensional matrix applying the θ-effect, it is already known.

Two interesting observations about T:

- *It is not always invertible, an example of T can be taken with $w = 4$, $det(T) = 0$ for the same.*
- *All of its elements are non-negative (0 or 1).*

So, we cannot invert the θ-effect step, so let's go back to θ for the inverse.

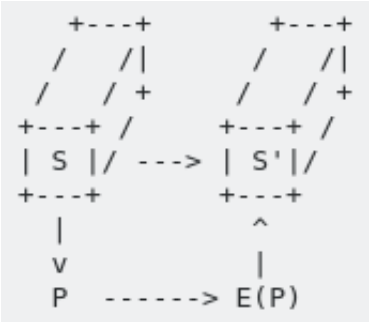

Figure 2: THETA STEP

In the above figure, the values of P and $E(P)$ defined in eq. (10) and eq. (9) respectively, we have omitted x, z from $P[x][z]$ and $E[x][z]$ for ease,
Let's use the above results

$$\bigoplus_{y=0}^{4} S[x][y][z] = P \qquad \text{Recall eq. (10)}$$

$$\bigoplus_{y=0}^{4} S'[x][y][z] = P \oplus (5 \times E(P)) \qquad \text{From eq. (8) and above equation}$$

Using the fact that $E \oplus E = 0$, we have

$$\bigoplus_{y=0}^{4} S'[x][y][z] = P \oplus E(P) \tag{12}$$

Coming back to our earlier result, eq. (11)

$$W' = W \cdot T \qquad \text{Recall eq. (11)}$$
$$W' \oplus W = W \cdot (I \oplus T) \qquad \text{Adding } W \text{ on both sides}$$
$$E(P) \oplus P = P \cdot (I \oplus T) \qquad \text{Remember } W = P \text{ and } W' = E(P) \text{ from the } \theta\text{-effect}$$
$$\bigoplus_{y=0}^{4} S'[x][y][z] = P \cdot (I \oplus T) \qquad \text{From eq. (12)}$$

From the above result, we can retrieve the value of P, since $I \oplus T$ will be invertible.

$$P = \bigoplus_{y=0}^{4} S'[x][y][z] \cdot (I \oplus T)^{-1} \tag{13}$$

Now, we have retrieved P, we can simply apply the θ-effect and get $E(P)$, and

$$\begin{aligned} S'[x][y][z] &= S[x][y][z] \oplus E[x][z] && \textit{Recall eq. (8)} \\ S[x][y][z] &= S'[x][y][z] \oplus E[x][z] && \textit{XOR's additive inverse is itself} \end{aligned}$$

So, the final inverse of θ will look as follows:

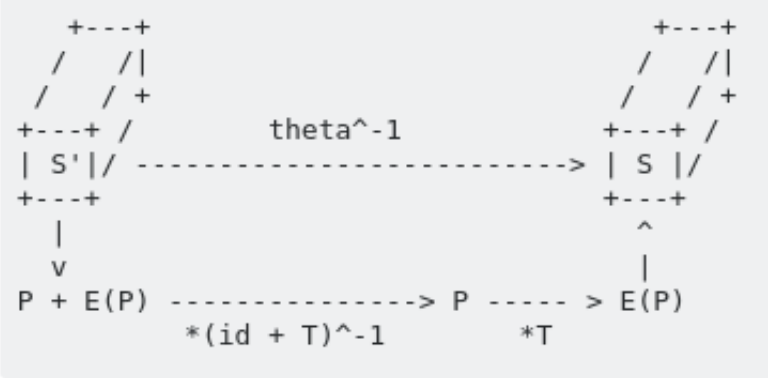

Figure 3: THETA INVERSE

The major time consuming step in the above operation is computation of $(I \oplus T)^{-1}$ ($5w \times 5w$ 2-dimensional matrix).

1.2.3 Computing π^{-1} and ρ^{-1}

The ρ step simply rotates each lane by a predefined lane towards MSB, so to invert it, we can rotate each lane by the original amount towards LSB, i.e.

$$S[x][y] = S'[x][y] >> r[x][y]$$

Applying above equation for each lane will give us ρ^{-1}.

The π step simply interchanges the lanes of the state S, just like the above step, we can get the π^{-1} by simply putting each lane in its original place.

1.2.4 Preimage Attack on WECCAK

We have the internal transformation function $R = \chi \cdot \rho \cdot \pi \cdot \theta$ which contains 4 individual steps which we already discussed in sections above.

In the last 3 sub-sections, we proved that all of the 4 steps are invertible, this makes the function (step) R invertible

$$R^{-1} = \theta^{-1} \cdot \pi^{-1} \cdot \rho^{-1} \cdot \chi^{-1}$$

Since R is invertible, the WECCAK function $F = R \cdot R$ is also invertible,

$$F^{-1} = R^{-1} \cdot R^{-1}$$

We will use the above result to perform the **Preimage Attack on WECCAK**.

Solution
For the Preimage Attack, we will be given a Hash Value, which is 80 *bits in WECCAK* (10 *lanes, since each lane is of* 8 *bits).*

We know that F is invertible and we have already devised the method calculate the inverse of F, let's use it to move backwards from our hash output.

Since hash value is of only 10 *lanes, rest* 15 *lanes are unknown to us, and their specific value don't matter to us because we just want to find an input string which results in this hash value (first* 10 *lanes).*

The steps to perform the Preimage Attack will be as follows:

1. *Populate the first* 10 *lanes with the given hash value, pad the rest* 15 *lanes as all zeroes*
2. *Now we know the complete output* O_1 *(all* 25 *lanes) of WECCAK, apply the function* F^{-1} *on* O_1.
3. $F^{-1}(O_1)$ *gives us the value* M_1'
4. *Take the first* 184 *bits of* M_1', *call it* M_1.
5. M_1 *is the Preimage for the provided Hash Value.*

In the above solution we are only finding a message of length atmost 184 *bits (i.e. only one block) which will give us the required hash value.*
If we try to find hash value of M_1,

- *We pad* M_1 *with* 16 *zeroes to get* M_1'.
- M_1' *is XORed with S which is all zeroes right now, so* M_1' *will not change*
- *F is applied to* M_1' *to get the output* O_1.
- *The first* 80 *bits of* O_1 *will be same as our hash value.*

The time complexity of the F^{-1} *computation will be driven by the* θ^{-1} *step but since the size of matrix to invert is* 40×40, *it is well within the bounds of computation.*

1.2.5 Collision Attack on WECCAK

We have already discussed the **Preimage Attack** on WECCAK in section 1.2.4.

For performing the **Collision Attack**, we will be given a Hash Value H, and we have to find 2 distinct messages M_1 and N_1 such that both of their hashes are H.

Solution
Since we know the hash value H, we know 10 *lanes of the output, rest* 15 *lanes are unknown.*
The steps to perform the Preimage Attack will be as follows:

1. *Perform the Preimage Attack with rest* 15 *lanes as zeroes to get* M_1.
2. *Pick a different value for the rest* 15 *lanes and perform the Preimage Attack to get* N_1.
3. *If* $N_1 \neq M_1$, *then we are done, report* M_1 *and* N_1 *as the result.*
4. *If* $N_1 = M_1$, *then go to Step 2.*

Both the input strings M_1 *and* N_1 *retrieved from above will result in the same hash value H.*

1.2.6 Second Preimage Attack

We have already discussed the **Preimage Attack** on WECCAK in section 1.2.4.

For performing the **Second Preimage Attack**, we will be given a Hash Value $H(M)$, and the original input string M, we have to find N_1 such that the hash of N_1 is same as $H(M)$.

Solution
Since we know the hash value H, we know 10 *lanes of the output, rest* 15 *lanes are unknown.*

Our solution will be based on the value of M.
The solution steps will be as follows:

- *If the size of M is greater than* 184 *bits, i.e. the input string consists of more that* 1 *block, then simply perform the* ***Preimage Attack*** *using $H(M)$ to get N_1 and report the result.*
- *If the size of M is atmost* 184 *bits, then do the following:*

    1. *Calculate the O_1 value by performing WECCAK operations on $H(M)$ and note the rest* 15 *lanes of O_1.*
    2. *Pick a different value for the rest* 15 *lanes and perform the Preimage Attack to get N_1.*
    3. *If $N_1 \neq M$, then we are done, report N_1 as the result.*
    4. *If $N_1 = M$, then go to Step 2.*

In the first case, when M is input string of more than 1 *blocks, we are sure that N_1 will be distinct since we are computing input strings of only* 1 *block.*

The second case is similar to what we did in the ***Collision Attack*** *and it will give us a N_1 which is distinct from M.*