Modern Cryptology (CS641A)

Assignment 7
WECCAK (WEAK-KECCAK)

Instructor: Prof. Manindra Agrawal
Group Name: CMEN

Roll No	Names of Students
170500	Pratyush
160508	Pravar Deep Singh
160749	Tushar Garg
160755	Umesh Meena

Date of Submission: 15th June 2020

Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh, India – 208016

1 WECCAK (WEAK-KECCAK)

Consider a variant of KECCAK hash function which we will call as WECCAK (WEAK-KECCAK). The following is the description of WECCAK.

1. Input to the hash function is a message $M \in \{0,1\}^{\cdot}$.
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 $M_1, M_2, ..., M_r$.
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. (This procedure is similar to KECCAK).
7. This state is given as input to a function F (which will be defined later) and let's call its output as O_1. The output of 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 r times.
10. The output of WECCAK is the initial 80 bits of O_r. (This is similar to KECCAK).

2 Problem

Let $R = \chi \circ \rho \circ \pi \circ \theta$ (χ, ρ, π, θ are the same as defined in KECCAK). Please note that now in all operations z indices are modulo 8.

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

3 Solution

3.1 Finding θ Inverse

3.1.1 The θ Operation

Input: state array A.

Output: state array A'.

Steps:

1. For all pairs (x, z) such that $0 \leq x < 5$ and $0 \leq z < 8$, let
$$C[x, z] = A[x, 0, z] \oplus A[x, 1, z] \oplus A[x, 2, z] \oplus A[x, 3, z] \oplus A[x, 4, z]$$
2. For all pairs (x, z) such that $0 \leq x < 5$ and $0 \leq z < 8$, let
$$D[x, z] = C[(x - 1) \bmod 5, z] \oplus C[(x + 1) \bmod 5, (z - 1) \bmod 8].$$
3. For all triples (x, y, z) such that $0 \leq x < 5$, $0 \leq y < 5$ and $0 \leq z < 8$, let
$$A'[x, y, z] = A[x, y, z] \oplus D[x, z]$$

3.1.2 Visualizing the operations

Now to find the inverse of the θ operation we visualize the term used in defining θ operation.

1. **The Parity Plane (C)**:- The C[x, z] term represent the parity of the column in the state matrix where a column is defined as the set of all points for which (x, z) values are the same. This is because entry of our state matrix is either 0 or 1 and the XOR operator has the property that $1 \oplus 1 = 0$ and $1 \oplus 0 = 1$. Hence, odd number of 1's will produce 1 and even number of 1's will produce 0 as output hence the name Parity Plane.
2. **The θ-effect (D)**:- This is the 2D-plane whose entry are $D[x, z]$.

3.1.3 Construction of Inverse operation

Given A' we want to find A.
Firstly We want to visualize the 2D matrix C as a 1D array W. Since there are $5 \times 8 = 40$ element in C we create a 1D vector W of size 40. From a 2D 5×8 matrix we can easily convert it into a 40 length vector where $(x, z)^{\text{th}}$ element of the matrix is same as $(8 \cdot x + z)^{\text{th}}$ element of the vector. Also, from Division theorem we know "Let a and b be integers with $b > 0$. Then there exist unique integers q and r with the property that $a = b \cdot q + r$, where $0 \leq r < b$.". Now consider the i^{th} element of the vector W. From the Division theorem we know that we can find unique x and z s.t.,

$$i = 8 \cdot x + z, \quad \text{for some } 0 \leq z < 8$$

Now since the maximum value of i is 39 (assuming 0 base indexing), the maximum value of x can also be 4. Hence every element of vector W uniquely corresponds to a element in the 2D matrix C. So, we can write,

$$W[8 \cdot x + z] = C[x, z]$$

Therefore from a 40 length vector W we can construct back the 5×8 Parity plane by finding the coordinate (x,z) in the parity plane of i^{th} element of W in the 2D-Matrix as follows

$$x = \lfloor i/8 \rfloor, \quad z = i \bmod 8$$

We name the mapping to convert 5×8 matrix to 40 length vector as φ . Also let

$$\varphi(C) = W \text{ and } \quad \varphi^{-1}(W) = C$$

Here, φ^{-1} is well defined because we have given the inverse mapping already.
Now Since the θ-effect is also a 2D matrix we can write it as vector $W^{'}$ of length 40 by applying the map φ.
Now, let us construct a 2D-matrix T of size 40×40 as follows. To get the i^{th} column of the Matrix T we do the following

$$x := \lfloor i/8 \rfloor, z := i\%8$$

$$p := (x-1) \bmod 5, \quad q := (x+1) \bmod 5, \quad r = (z-1) \bmod 8$$

$$j := 8 \cdot p + z, \quad k := 8 \cdot q + r$$

T for our case is given in file "T.txt". Now we make all the entry in the i^{th} column to be 0 except the j^{th} and k^{th} row which we make equal to 1. Now if we Right multiply this to the W vector, where addition and multiplication happens in field GF(2), then this will have same effect as assigning the $[x, z]^{th}$ element of Parity plane matrix as XOR of $[x-1, z]^{th}$ and $[x+1, z-1]^{th}$ element. Hence we can say that

$$W^{'} = W \cdot T$$

Which implies

$$D = \varphi^{-1} \circ (\lambda x \to x \cdot T) \circ \varphi(C)$$

Now we want to find the parity plane of the output state: First of all,

$$A^{'}[x, y, z] = A[x, y, z] \oplus D[x, z]$$

hence

$$\oplus_{y=0}^{y=4} A^{'}[x, y, z] = \oplus_{y=0}^{y=4} (A[x, y, z] \oplus D[x, z])$$

Now we know $a \oplus a = 0$

$$\oplus_{y=0}^{y=4} A^{'}[x, y, z] = (\oplus_{y=0}^{y=4} A[x, y, z]) \oplus D[x, z]$$

Putting $\oplus_{y=0}^{y=4} A[x, y, z] = C[x, z]$ we get,

$$C^{'}[x, z] = \oplus_{y=0}^{y=4} A^{'}[x, y, z] = C[x, z] \oplus D[x, z]$$

If the operation are done in field GF(2) then $\oplus$ is equivalent to addition. Hence, Parity Plane of Output state $(C^{'}) = C+D$. Now the φ operation has the property $\varphi(A+B) = \varphi(A)+\varphi(B)$ because it is immaterial if we map the bit first to the vector or take the XOR first since both operation apply on individual bits. So, applying map φ on both side

$$\varphi(C^{'}) = \varphi(C) + \varphi(D)$$

Hence,

$$\varphi(C^{'}) = W + W \cdot T$$
$$\varphi(C^{'}) = W \cdot (id + T)$$

Since $C^{'}$ can be computed from the given state matrix $A^{'}$, so if we know the inverse of the matrix $(id + T)$ in field GF(2), then we can get the vector W and hence the matrix C. For our case, matrix $(id + T)$ and $(id + T)^{-1}$ are given in files "id_T.txt" and "id_T_inv.txt" respectively.
Now we know

$$C^{'} = C + D$$

adding C both side we get

$$D = C + C^{'}$$

Since we now know know both C and $C^{'}$ we can find D.
Now we know that

$$A^{'}[x, y, z] = A[x, y, z] \oplus D[x, z]$$

taking XOR of $D[x, z]$ on both side we get,

$$A[x, y, z] = A^{'}[x, y, z] \oplus D[x, z]$$

Now we know both $A^{'}$ and D so we can compute A.
And hence inverse of θ for any input $A^{'}$.

3.2 The χ Inverse

3.2.1 The χ operation

Input: state array A.

Output: state array $A^{'}$.

Steps:

1. For all triples (x, y, z) such that $0 \leq x < 5$, $0 \leq y < 5$ and $0 \leq z < 8$, let

$$A^{'}[x, y, z] = A[x, y, z] \oplus ((A[(x + 1) \bmod 5, y, z] \oplus 1) \cdot A[(x + 2) \bmod 5, y, z])$$

3.2.2 Inverse of χ

From the equation above, it is clear that the χ operation operates on a row where row is defined same as in Keccak. So, let us assume we are given a row represented as a_0, a_1, a_2, a_3, a_4 and corresponding output row as b_0, b_1, b_2, b_3, b_4. So, We can rewrite operation on individual rows as:

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

Here subscript is modulo 5 and each of a_i and b_i is either 0 or 1.
Now we know if a is a binary operand then $a \oplus 1 = \overline{a}$, where $\overline{a}$ represents complement of a. we get,

$$b_i = a_i \oplus (\overline{a_{i+1}} \cdot a_{i+2}) \tag{2}$$

So now taking XOR with 1 on both side of equation 1 we get,

$$b_i \oplus 1 = a_i \oplus 1 \oplus (\overline{a_{i+1}} \cdot a_{i+2})$$

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

Now since each of a_i and b_i is either 0 or 1 we can replace the '$\oplus$' operator with addition and '$\cdot$' operator with multiplication in field GF(2). So the equation transform into the following equation:

$$b_i = a_i + \overline{a_{i+1}} \cdot a_{i+2} \tag{4}$$

adding $a_i + b_i$ on the both side we get,

$$a_i = b_i + \overline{a_{i+1}} \cdot a_{i+2} \tag{5}$$

Similarly adding $\overline{a_i} + \overline{b_i}$ in equation 3 we get,

$$\overline{a_i} = \overline{b_i} + \overline{a_{i+1}} \cdot a_{i+2} \tag{6}$$

Now we consider equation 5

$$a_i = b_i + \overline{a_{i+1}} \cdot a_{i+2} \tag{7}$$

We put the value of $\overline{a_{i+1}}$ using the equation 6

$$a_i = b_i + (\overline{b_{i+1}} + \overline{a_{i+2}} \cdot a_{i+3}) \cdot a_{i+2} \tag{8}$$

Now the equation 8 simplifies to

$$a_i = b_i + \overline{b_{i+1}} \cdot a_{i+2} \tag{9}$$

Now from equation 5 we put the value of a_{i+2} we get,

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

Now we put the value of $\overline{a_{i+3}}$ from equation 6

$$\begin{aligned} a_i &= b_i + \overline{b_{i+1}} \cdot b_{i+2} + \overline{b_{i+1}} \cdot (\overline{b_{i+3}} + \overline{a_{i+4}} \cdot a_{i+5}) \cdot a_{i+4} \\ &= b_i + \overline{b_{i+1}} \cdot b_{i+2} + \overline{b_{i+1}} \cdot \overline{b_{i+3}} \cdot a_{i+4} \end{aligned}$$

Now we put the value of a_{i+4} from the equation 5

$$\begin{aligned} a_i &= b_i + \overline{b_{i+1}} \cdot b_{i+2} + \overline{b_{i+1}} \cdot \overline{b_{i+3}} \cdot (b_{i+4} + \overline{a_{i+5}} \cdot a_{i+6}) \\ &= b_i + \overline{b_{i+1}} \cdot b_{i+2} + \overline{b_{i+1}} \cdot \overline{b_{i+3}} \cdot (b_{i+4} + \overline{a_i} \cdot a_{i+1}) \end{aligned}$$

$$a_i = b_i + \overline{b_{i+1}} \cdot b_{i+2} + \overline{b_{i+1}} \cdot \overline{b_{i+3}} \cdot b_{i+4} + \overline{b_{i+1}} \cdot \overline{b_{i+3}} \cdot \overline{a_i} \cdot a_{i+1} \tag{10}$$

Now we since we used all the five equation we expanded the last term of equation i.e. $\overline{b_{i+1}} \cdot \overline{b_{i+3}} \cdot \overline{a_i} \cdot a_{i+1}$

$$\begin{aligned} \overline{b_{i+3}} \cdot \overline{a_i} &= (1 + b_{i+3}) \cdot \overline{a_i} \\ &= (1 + a_{i+3} + \overline{a_{i+4}} \cdot a_i) \cdot \overline{a_i} \\ &= \overline{a_i} + a_{i+3} \cdot \overline{a_i} + \overline{a_{i+4}} \cdot a_i \cdot \overline{a_i} \\ &= \overline{a_i} + a_{i+3} \cdot \overline{a_i} \\ &= \overline{a_i} \cdot \overline{a_{i+3}} \end{aligned}$$

$$\begin{aligned} \overline{b_{i+1}} \cdot a_{i+1} &= (1 + b_{i+1}) \cdot a_{i+1} \\ &= (1 + a_{i+1} + \overline{a_{i+2}} \cdot a_{i+3}) \cdot a_{i+1} \\ &= a_{i+1} + a_{i+1} + \overline{a_{i+2}} \cdot a_{i+3} \cdot a_{i+1} \\ &= \overline{a_{i+2}} \cdot a_{i+3} \cdot a_{i+1} \end{aligned}$$

Hence,

$$\begin{aligned} \overline{b_{i+1}} \cdot \overline{b_{i+3}} \cdot \overline{a_i} \cdot a_{i+1} &= \overline{a_i} \cdot \overline{a_{i+3}} \cdot \overline{a_{i+2}} \cdot a_{i+3} \cdot a_{i+1} \\ \overline{b_{i+1}} \cdot \overline{b_{i+3}} \cdot \overline{a_i} \cdot a_{i+1} &= 0 \end{aligned}$$

So we put this in equation 10 and we get the final answer to be

$$a_i = b_i + \overline{b_{i+1}} \cdot b_{i+2} + \overline{b_{i+1}} \cdot \overline{b_{i+3}} \cdot b_{i+4} \tag{11}$$

which can again be converted to XOR notation as follows

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

Hence for a given input $A^{'}$ every row can be inverted to find the corresponding row in inverse and therefore whole $A^{'}$ can be inverted.

3.3 Collision Attack

3.3.1 Definition

Given a hash function $h : X \to Y$ then we have to find two inputs $x_1 \in X$ and $x_2 \in X$ s.t. $x_1 \neq x_2$ and $h(x_1) = h(x_2)$.

3.3.2 Method 1

Consider two input binary strings, first $x \in X$ s.t. $|x| \bmod 184 \neq 0$ and second string $y \in X$ s.t.$y = x \,||\, 0$ (here $||$ means concatenation of both the strings). Now while performing WECCAK operation, in the second step we pad the input with minimum number of extra zeros such that bit-length of padded message is a multiple of 184. So in this step both the string will become equal hence the same hashing.

3.3.3 Method 2

First of all let us define a function $H : X \to Z$ s.t. it operates all the step of Weccak except the last one i.e. it does not output only the first 80 bits but the whole string of 200 length. (here X is a binary string corresponding to a message)
Now firstly we want to find two different input messages x_1 and x_2 such that $H(x_1)$ and $H(x_2)$ has same last 16 bits. For this we Generate $2^{16} + 1$ different binary strings of length multiple of 184. Then we compute H for all of these strings. Now since the last 16 bits of output of H has only 2^{16} possible values,by pigeonhole principle, at least two of our $2^{16} + 1$ generated binary strings will have the same last 16 bits of output of H. Lets call them x_1 and x_2. So the last 16 bits of $H(x_1)$ and $H(x_2)$ are the same. Lets call the first 184 bits of $H(x_1)$ and $H(x_2)$ as z_1 and z_2. So we see that

$$H(x_1) \oplus H(x_2) = (z_1 \oplus z_2) \,||\, 0^{16}$$

Now we consider $y_1, y_2 \in X$ s.t. $y_1 = x_1 \,||\, (z_1 \oplus z_2)$ and $y_2 = x_2 \,||\, 0^{184}$. Now we can see that,

$$\begin{aligned}h(y_1) &= h(x_1 \,||\, (z_1 \oplus z_2)) \\ &= h(H(x_1) \oplus ((z_1 \oplus z_2) \,||\, 0^{16})) \\ &= h(H(x_1) \oplus (H(x_1) \oplus H(x_2))) \\ &= h(H(x_1) \oplus H(x_1) \oplus H(x_2)) \\ &= h(H(x_2))\end{aligned}$$

Also,

$$\begin{aligned} h(y_2) &= h(x_2 \,||\, 0^{184}) \\ &= h(H(x_2) \oplus (0^{184} \,||\, 0^{16})) \\ &= h(H(x_2) \oplus 0^{200}) \\ &= h(H(x_2)) \end{aligned}$$

Here h is our hash function for which we had to analyze collision attack. Now, since we generated different inputs we get $x_1 \neq x_2$, Hence $y_1 \neq y_2$ but $h(y_1) = h(y_2)$.
Hence we have found a collision.
Also, the time complexity to find the collision is the time needed to generate 2^{16} different binary strings which is of order of 2^{16}. Also the memory consumption is of order of 2^{16}. Therefore our collision attack is practical.

3.4 Preimage Attack

3.4.1 Definition

Given a hash function $h : X \to Y$ and $y \in Y$ then we have to find some $x \in X$ s.t. $h(x) = y$

3.4.2 Method

Let we are given a hashing h_1 of length 80. These 80 bits corresponds to 10 lanes in the state matrix as shown in the image and can be inverted till θ^{-1} operation without considering other 120 bits as follows:

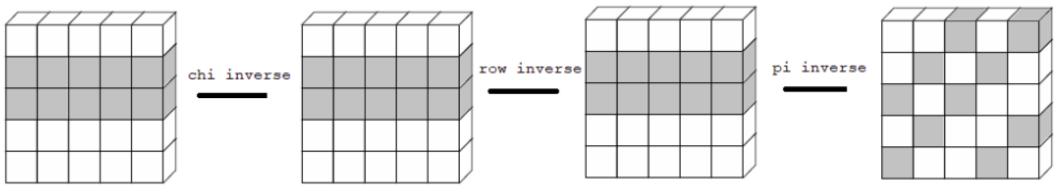

Figure 1: Location of first 80 bits of output after inversion of each operation

Here we know the exact value of the Gray bits and can set the white bits as we want. Now we want to have control over the last 16 bits of some state (say G) s.t. $R \circ R \circ (G)$ has first 80 bits equal to h_1. So we consider the last 16 bits of G and see their propagation as follows

Now to get as many different suffixes of length 16 of state G as possible, we try to set other independent bits as follows:

1. From the θ inverse section we know that if we know the parity plane matrix of $R \circ (G)$ then we can reverse the theta operation. So we consider all the $2^{5\times 8} = 2^{40}$ possible parity plane matrices and invert the known values (Color: Gray) of the matrix after θ operation(It also posses condition on other known bits).

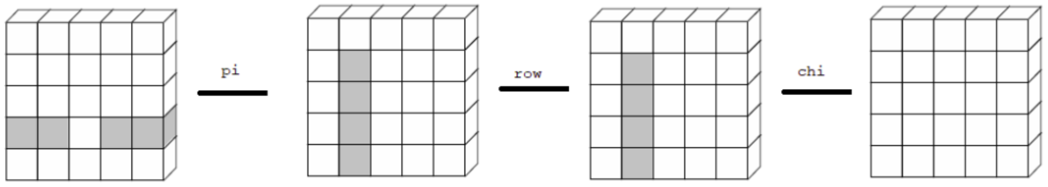

Figure 2: Location of last 16 bits after each operation

2. From all the 2^{40} possible known values we get, we set the rest of the bits s.t. after performing chi inverse we get as many different values as possible at the gray position in Figure 2. This also should be done in accordance with the Parity Plane.

Now let we know state G and its string representation is x_1 s.t. $R \circ R \circ (x_1)$ has first starting bits equal to h_1. Now we randomly generate a input x_2 of length multiple of 184 s.t. last 16 bits of $H(x_2)$ are same as last 16 bits of x_1. This is possible with very high probability since we have many state with different possibilities of the suffixes of length 16 of x_1 Now let

$$y = \text{first 184 bits of } (H(x_2) \oplus x_1))$$

then we have,

$$H(x_2) \oplus x_1 = y \,||\, 0^{16}$$

Consider $z = x_2 \,||\, y$ Now we can see that

$$\begin{aligned} h(z) &= h(x_2 \,||\, y) \\ &= h(h(x_2) \oplus (y \,||\, 0^{16})) \\ &= h(h(x_2) \oplus h(x_2) \oplus x_1) \\ &= h(x_1) \end{aligned}$$

4 Reference

1. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
2. https://crypto.stackexchange.com/questions/52876/what-is-the-current-time-limit-of-keccak-inversion
3. https://eprint.iacr.org/2013/561.pdf