Assignment 7: CS641 Modern Cryptology

Nikhil Naidu (170758)
Dewansh Kr Singh (170241)
Parv Jain (170464)

Group Name: objectStrongly

June 15, 2020

Introduction WECCAK

This assignment is on analysis on a weak variant of **Keccak**, known as **Weccak**, which has the following description:

- Input to the hash function is a message M $\in \{0,1\}^*$. M is padded with minimum number of zeros such that bit-length of padded message is a multiple of 184.
- M is broken down into chunks of 184-bit length messages M_1, M_2,..M_r.
- A state **S** in WECCAK hash function is a **5**×**5**×**8** 3-dimensional array. i.e, a state consists of **200 bits**. Initially all states are zeros.
- 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) and sent into a function F.
- $\mathbf{F = R \circ R}$ ie a two round function where round R = $\chi \circ \rho \circ \pi \circ \theta$
- This is continued till all the message blocks have been XORed.
- The output of WECCAK is first 80 bits of final state

Problem 1

Compute the inverse of χ(chi) and θ (theta)

Solution 1

χ (chi) inverse

Algorithm for χ
Input : State array A
Output : State Array B
Steps:

- 1. For all triples (x, y, z) such that $0 \leq x < 5$, $0 \leq y < 5$, and $0 \leq z < 8$, let B[x,y,z] = A[x,y,z] $\oplus$ ((A[(x+1) mod 5, y, z] $\oplus$ 1) ·and A[(x+2) mod 5, y, z]).
- 2. Return B

Computing Inverse The χ is a row-dependent operation and we simply observed that if we know all the bits of a row then we can simply find its corresponding inverse easily as χ offers a clear **Bijective mapping** of 2^5 to 2^5 pairs i.e for a given 5 bit sequence F, the output of χ(F) comes out unique.
The Table1(Refer Table 1) gives the complete mapping of 5bits (all 32 possibilities) to their corresponding χ values and from here one can simply look at the inverse of a chi function i.e to find the chi inverse of S look at the corresponding value of χ^{-1}(S) in the table

$A_0A_1A_2A_3A_4 \xrightarrow{\chi^{-1}} B_0B_1B_2B_3B_4$
$B_i = A_i \oplus (\neg\, A_{i+1}) \cap (A_{i+2} \oplus (\neg\, A_{i+3}) \cap A_{i+4})$

Table 1: χ(Chi) values

F or χ^{-1}(S)	$\chi(F)$ or S	F or χ^{-1}(S)	$\chi(F)$ or S
00000	00000	10000	10010
00001	00101	10001	10101
00010	01010	10010	11000
00011	01011	10011	11011
00100	10100	10100	00110
00101	10001	10101	00001
00110	10110	10110	00100
00111	10111	10111	00111
01000	01001	11000	11010
01001	01100	11001	11101
01010	00011	11010	10000
01011	00010	11011	10011
01100	01101	11100	11110
01101	01000	11101	11001
01110	01111	11110	11100
01111	01110	11111	11111

θ (theta) inverse

Algorithm for θ
Input : State array A
Output : state array B
Function: θ(theta) XORs each bit a[x][y][z] to bitwise sum of the parities of two columns: a[x−1][·][z] and a[x+1][·][z−1].
Steps:

- 1. For all triplets (x, y, z) such that $0 \leq$ x $<$ 5, $0 \leq$ y $<$ 5, and $0 \leq$ z $<$ 8, let:
  Parity[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 triplets (x, y, z) such that $0 \leq$ x $<$ 5, $0 \leq$ y $<$ 5, and $0 \leq$ z $<$ 8, let:
  B[x,y,z] = A[x,y,z] $\bigoplus$ Parity[(x - 1) mod 5, z] $\bigoplus$ Parity[(x + 1) mod 5, (z - 1) mod 8]

Computing Inverse

- Computing the inverse of θ can be done by adopting a polynomial notation. the coefficient of the monomial X^i Y^j Z^k denotes the value of bit a[i][j][k].The exponents i and j range from 0 to 4 and the exponent k ranges from 0 to 7.
- The polynomial representing the state of WECCAK is an element of a polynomial quotient ring defined by the polynomial ring over Gaussian Field GF(2) [x,y,z] modulo the ideal generated by $<1+ x^5, 1+y^5, 1+z^8 >$
- When the state is represented by a polynomial, the mapping θ can be expressed as the multiplication(in the quotient ring defined above) by the following polynomial:

$$1 + \bar{y}(x + x^4 z) \quad where \quad \bar{y} = \sum_{i=0}^{4} y^i = \frac{1 + y^4}{1 + y} \qquad - eq1$$

- The inverse of θ(theta) is simply the inverse of above polynomial(eq1) in the quotient ring defined above.
- First, we assume that the inverse would be of form $(1 + \bar{y}$ Q) with Q in a polynomial of x and z only and simplified the equation from the reference to be:

$$1 + \bar{y}(x + x^4 z) * (1 + \bar{y}Q) = 1 \quad mod < 1 + x^5, 1 + y^5, 1 + z^8 >$$

- Putting $\bar{y}^2 = \bar{y}$ gives

$$Q = 1 + (1 + x + x^4 z)^{-1} mod < 1 + x^5, 1 + z^8 >$$

- We computed the inverse of 1+x+x^4z and thus inverse of Q using the code from SAGE given below.

```
1. R.<x,z> =GF(2)[]
2. I = R.ideal(1+x^5,1+z^8)
3. f= 1+x+z*x**4
4. Q = f.inverse_mod(I)+1
5. Return Q
```

The inverse of Q for w=8 was found out to be:
$Q = x^4.z^7 + x^3.z^7 + x^4.z^5 + x^3.z^6 + x^2.z^7 + x^3.z^5 + x^3.z^4 + x^2.z^5 + x.z^6 + x^4.z^2 + x^3.z^3 + x^2.z^4 + x^4.z + z^5 + x^2.z^2 + x.z^2 + z^3 + x^2 + z^2 + z$

- Hence, after finding this Inverse function, θ inverse can be simply applied by multiplying the state matrix polynomial with the inverse polynimial (1+ $\bar{y}$ Q) where the coefficient X^i Y^j Z^k in the product simply denotes the value of bit at a[i][j][k]

Problem 2

Claim about the security of WECCAK with F = R ∘ R (Give a preimage, collision and second preimage attack)

Solution 2

Preimage Attack

- Definition: Given a hash output H(M) , find the corresponding message M.
- **Weakness of WECCAK:**
  - Two-round function
  - The round does not include ι(iota) function (Without it, the round function would be translation-invariant in the z-direction and all rounds would be equal making Keccak-f subject to attacks exploiting symmetry)
  - The output hash is 80 bits thus it requires exactly 10 lanes (80/8) which corresponds to 2 rows in the output matrix completely known. Thus allowing us to use the Chi inverse mapping from table directly.
- **Attack Construction:**
  - The entire Weccak function is summarized in these four states given in figure 1:
    State 1 $\xrightarrow{\theta,\pi,\rho}$ State 2 $\xrightarrow{\chi,\theta}$ State 3 $\xrightarrow{\pi,\rho,\chi}$ State 4
  - State 1 : This is an initial state XORed with message M_i of 184 bits, which are arranged in a lane fashion with a lane size (w= 8) thus making total 23 lanes. Hence, every slice (x-y plane) consisits of 23 bits from the message M_i and remaining two bits are padded to be 0.
  - State 2 :After applying θ, π, ρ, to state-1, we obtain the state 2.
  - State 4 : This is the matrix of the Hashed message, where only the first two rows are considered as output (Remaining three rows are flushed out/don't care).
  - State 3 : This state is obtained either by applying χ^{-1}, π^{-1},ρ^{-1} on State-4 or by applying χ and θ on state-2. This common intermediate state helps us in solving necessary attacks. The white spaces in State 3 are those we dont care.
- **Implementation:**
  - As the Hash message H(M)(output 80bits) is known, the first two rows of the state-4 are known(i.e 80/8 = 10 lanes) and the rest we don't care(*). and by applying inverse χ on stage-4, this uniquely maps to another state with same two known rows (since, inverse χ for an entire known row maps uniquely and to the same row.).
  - Now, we apply inverse π, inverse ρ,on state 4 since they just change the (x,y) and rotate respectively ,corresponding to each known lane in state 4 {h0,h1,..h9} we also get 10 lanes {h0',h1',...h9'} known in output state 3 as well , the rest we don't care.
  - In state-1,we have total 23 lanes and thus 23*8 variables . we would like to impose the following conditions on the input state:
    - each column in the input slice must maintain a Zero-Parity condition, which makes the effect of applying θ on the state an Identity.

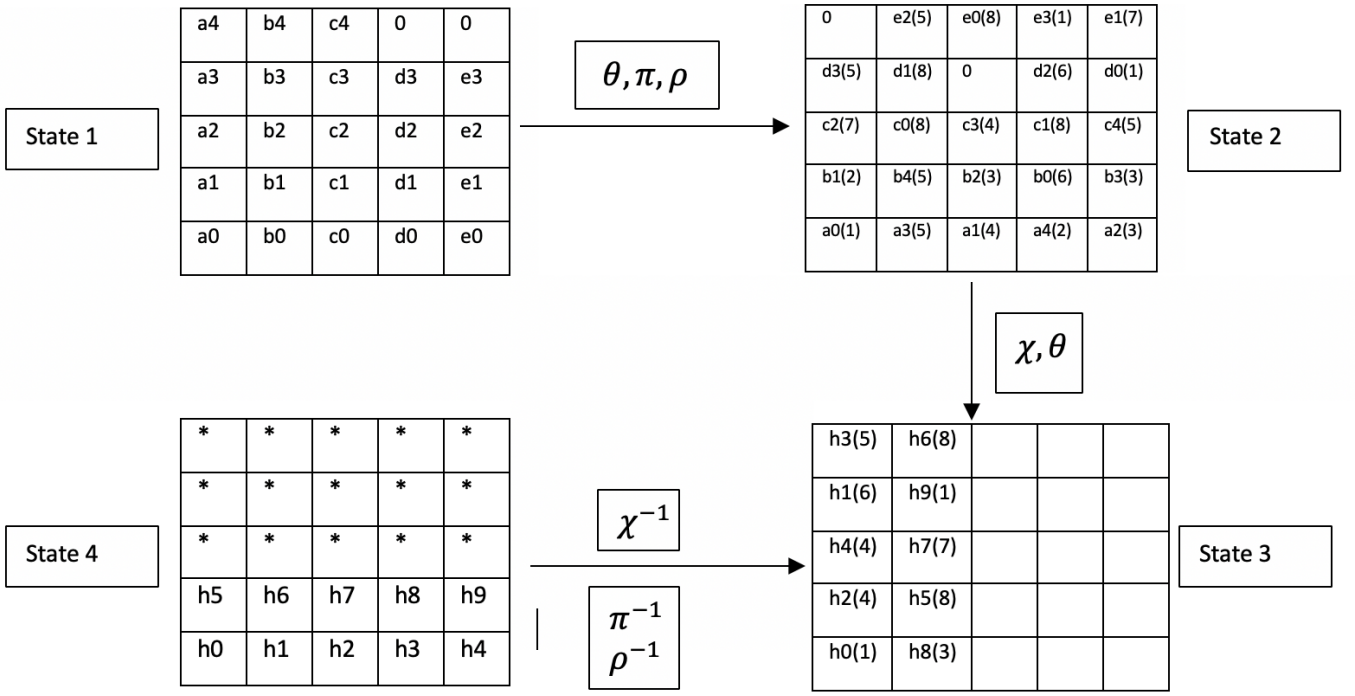

Figure 1: State diagrams

* Moreover we also assign some values to the columns (reason will be explained later).
* We set a_i's = 0 , b_i's =0
  c4 = c0 ⊕ c1 ⊕ c2 ⊕ c3
  d3 = d0 ⊕ d1 ⊕ d2
  e3 = e0 ⊕ e1 ⊕ e2
* this gives us a total of (5+5+1+1+1)*8 = 13*8 equations

– We now apply θ,π and ρ on state 1 and get output state 2

– At state two we apply χ θ to get to state 3

– But we already know 10 lanes in this state, so we compare both the outputs and thus get 10*8 equations more.

– This makes it a total to 10*8 + 13*8 = 23*8 equations and the same number of variables.Since the number of variables and the number of equations is the same, we can expect to find a solution

- **Analysis:**

  – Possible Solution for 2 slice

    * There are total 13 variables in a single slice (if we exclude the a's and b's which currently are set to 0) so it means for two slices we have 2^{2*13} possibilities
    * Given a 2-slice in the State 2, we need to apply $\theta \circ \chi$ mapping to get an output in the State 3. Since the θ mapping depends on the values of two slices; the current slice and one preceding it, we will only able to get the correct output for second slice.
    * In the State 3, we have the values of 10 lanes. So for one slice, we already have 10 fixed bit values , thus reducing the number of variables by 10 and hence for two slices now we have only $2^{2*13-10} = 2^{16}$ order of time complexity and memory.
    * There are four such pairs of two slices which makes it to $4*2^{16} = 2^{18}$ solutions

  – Possible Solution for 4 slice

    * We now concatenate first two slices to make it a four slice group. After joining, the topmost layer of second two pair, finds its required parity slice for application of θ function on itself.So the time complexity is: $2^{2*16-10} = 2^{22}$.
    * As there are two such pairs of 4 slices, so the order of time complexity would be;$2*2^{22}= 2^{23}$.

- Possible Solution for 8 slice
  - In this final act, we now combine the final 2 four-slice groups into a single eight slice group.
  - Again, after joining, the top layer of 2^{nd} four slice group finds its required parity slice. And also as this is the final piece, the topmost layer of the eight slice group also finds its parity slice, which is the last slice of eight slice group.
    Hence, total time complexity would be: $2^{2*22-10-10} = 2^{24}$.
  - Additional to these constraints, there is another important condition, the condition of parities in the input.
  - For example, the input choices of d_0, d_1, d_2 will decide the value of d_3. Hence, while all D's are present in a slice group, Then the One of the D's become no longer independent and hence remains fixed.
  - During the transition from step-1 to step-2, after applying π, ρ on first state, the elements which were present in same column in state-1 are now dispersed into different slices, with diffusion maintaining by more than four slices. i.e, all the column elements cannot be present in consecutive four slices.
  - Hence, for a single column we need to remove random possibility of one element. And for a slice there are three such columns which depend on parity. So, the new time complexity for eight-slice group would be: $2^{2*24-3*(8)} = 2^0 = 1$.
- **Order of Total Time** = $2^{18} + 2^{23} + 1 \leq 2^{24}$
- This procedure is not complete as we have fixed the values for both a and b columns. If the procedure fails then we can try for the possible pairs of a's and b's such that column parity remains zero
  Total such pairs of a and b
- all bits of a are 0's = 1
- two 1's and other 0's = 5C2 = 10
- four 1's and one 0 = 5C1 = 5
- Total =16
- thus total a and b pairs = 16*16
- **Order of Total Time** = 2^{24} * 16*16 = 2^{32}

Collision Attack

- **Definition :** Find two messages M_1,M_2 such that they produce the same Hash output H(M1)=H(M2).
- **Attack construction :**
  - This attack requires the same construction that was used for preimage attack. But another state is constructed between state-2 and state-3 for a better explanation.
  - State 2.5: This state is attained after applying only χ function to state-2. After applying θ to this state, we obtain State-3.
    $\text{State}\,1 \xrightarrow{\theta,\pi,\rho} \text{State}\,2 \xrightarrow{\chi} \text{State}\,2.5 \xrightarrow{\theta} \text{State}\,3 \xrightarrow{\pi,\rho,\chi} \text{State}\,4$
- **Implementation :**
  - We reach the state 3 from state 4 by applying χ^{-1}, ρ^{-1} and π^{-1}
  - If we observe closely in State-3,the output hash bits gets transferred to only the first two columns of each slice in state 3. Hence we will now focus only on these two columns.
  - To obtain these two columns from state-2.5 to stage-3, θ is being applied. As we know that, θ function depends only on the bit value, the previous column (x - 1) and next column (x+1,z-1).
  - So, the first column of state-3 depends on column-1,2,5 of state-2.5 and second column of state-3 depends on column-1,2,3 of state-2.5. Which clearly implies that **Column-4 of state-2.5 is not used**.
  - We can now exploit this weakness as per our convenience.
  - Consider the following example:
    - Let's start with any random input message (I_0) in sate 1 . suppose after arriving to state-2 from state-1 we get the first row of a slice as for eg **'11001'**. Now to proceed to state-2.5, χ function is applied. Which on applying to **'11001'** gives **'11101'**
    - As we know that fourth column of state-2.5 does not contribute anything to the output in state 3, we can tweak the fourth column and get same output in state 3.

* Now flipping the fourth bit of our selected row in state-2.5 gives **'11111'**, which on going to state-2 by applying χ^{-1}, gives **'11111'**.
* Notice the change in Initial value and final value of first row in state-2:
  Initial value : 11001
  Final value : 11111
* These change in bits can be re-propagated back to State-1 by applying ρ^{-1}, π^{-1} and θ^{-1}; obtaining different value of input in state-1 (I'$_0$) than the one we started as initial input (I_0)

– Hence, we can obtain two different messages (set $I_0 = M_1$ and I'$_0 = M_2$), for a same hash value.

- **Time analysis :**
  – During the implementation, we reached till state-2.5 from state-1 and returned to the initial state.
  – So, time complexity of moving from state-1 to state-2 would be: Time taken for π function(T_π) + ρ function(T_ρ) $\implies$ $T_\pi + T_\rho$
  – From state-2 to state-2.5, we apply χ function. So, time complexity is T_χ.

– During back propagation from state-2.5 to state-1, time complexity is ${T_\chi}^{-1}+{T_\rho}^{-1}+{T_\pi}^{-1}$.
Total time : $T_\pi + T_\rho + T_\chi + {T_\chi}^{-1}+{T_\rho}^{-1}+{T_\pi}^{-1}$

Second Preimage Attack

- **Definition :** Given a Hash value H(M_1) and its corresponding message M_1, finding another message M_2 with same Hash.
- **Attack construction :**
  – For second Preimage attack, We use same attack construction as collision attack's construction.
- **Implementation :**
  – We use same implementation of attack as we used in collision attack before. But with only difference is that the input message(I_0) here is fixed (= M_1).
  – So, similar to collision attack, we proceed till state-2.5 with M_1 as input and flip the fourth column bits and revert back to the initial state, but with different input this time (I'$_0 = M_2$).
- **Time analysis:**
  – As the process is very much similar to the collision attack, except for fixed inputs, the time complexities also will not change.

– Hence, **Total time :** $T_\pi + T_\rho + T_\chi + {T_\chi}^{-1}+{T_\rho}^{-1}+{T_\pi}^{-1}$.

References

- https://eprint.iacr.org/2018/1191.pdf
- https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
- http://naya.plasencia.free.fr/Maria/papers/Keccak_differential.pdf
- https://keccak.team/obsolete/Keccak-main-1.1.pdf