Side Channel Attacks on GIFT

Sudarshan Sharma

Senior Undergraduate,
Dual Degree with Specialisation in Microelectronics and VLSI Design,
Electronics and Electrical Communication Engineering,
IIT Kharagpur.

sudarshansharma04@gmail.com

April 29, 2019

Certificate

This is to certify that this report titled, **"Side Channel Attacks on GIFT"** submitted by **Sudarshan Sharma (Roll No.: 15EC32005)** to Indian Institute of Technology, Kharagpur, for the fulfilment of the Bachelor of Technology, is a record of work carried out by him under my supervision and guidance. The report, in my opinion, is worthy of consideration for the fulfilment of award of the degree of Bachelor of Technology in Engineering in accordance with the rules and regulations of the Institute.

Date: 29/04/2019
Place: IIT Kharagpur

Dr. Debdeep Mukhopadhyay
Professor
Department of Computer Science and Engineering,
IIT Kharagpur

Outline

GIFT- Introduction

- GIFT[1]is an improved version of PRESENT, with modification in bit permutation and Sbox layers.
- Advantages over PRESENT.
  - Smaller size due to efficient bit permutation layer and smaller Sboxes.
  - Lesser rounds and higher throughput.(PRESENT-31 round)
  - Simple and faster key schedule.
  - Better resistance against Linear Cryptanalysis.
- Two versions of GIFT, both the versions have 128 bit key.
  - GIFT-64, 28-round with 64-bit block size, discussion based on this.
  - GIFT-128, 40-round with 128-bit block size.

[1]Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, Yosuke Todo: GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption. CHES 2017: 321-345

GIFT- Design Specification

The GIFT cipher encryption can be simplified into three main steps as follows.

- **SubCell**- 16 4-bit Sboxes.
- **PermBits**-jth bit is mapped to jth bit, no xor gate required.
- **AddRoundKey**- Out of the 128-bit Key, 32 bit from the LSB side are used in the Key XOR layer in addition to this a 6-bit round constant is used.

Moreover, the Keys are updated after the completion of each round and round constants are updated before usage.

GIFT- Hardware Implementation

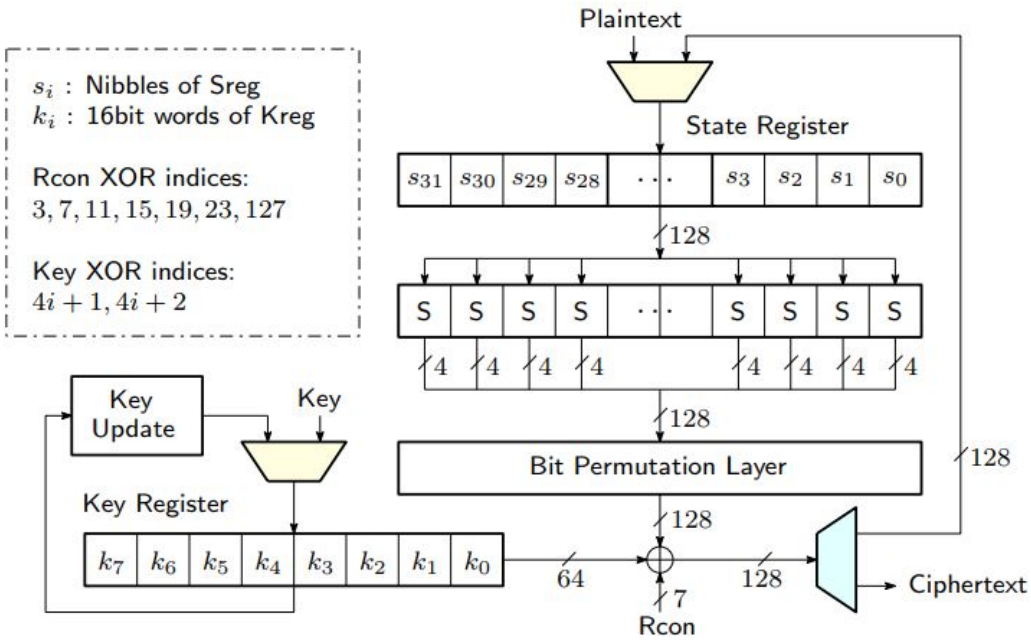

Figure: GIFT-128 Encryption Hardware Architecture[2]

[2]Naina Gupta, Arpan Jati, Anupam Chattopadhyay, Somitra Kumar Sanadhya, Donghoon Chang: Threshold Implementations of GIFT: A Trade-off Analysis. IACR Cryptology ePrint Archive 2017: 1040 (2017)

GIFT64- Serial Implementation on FPGA

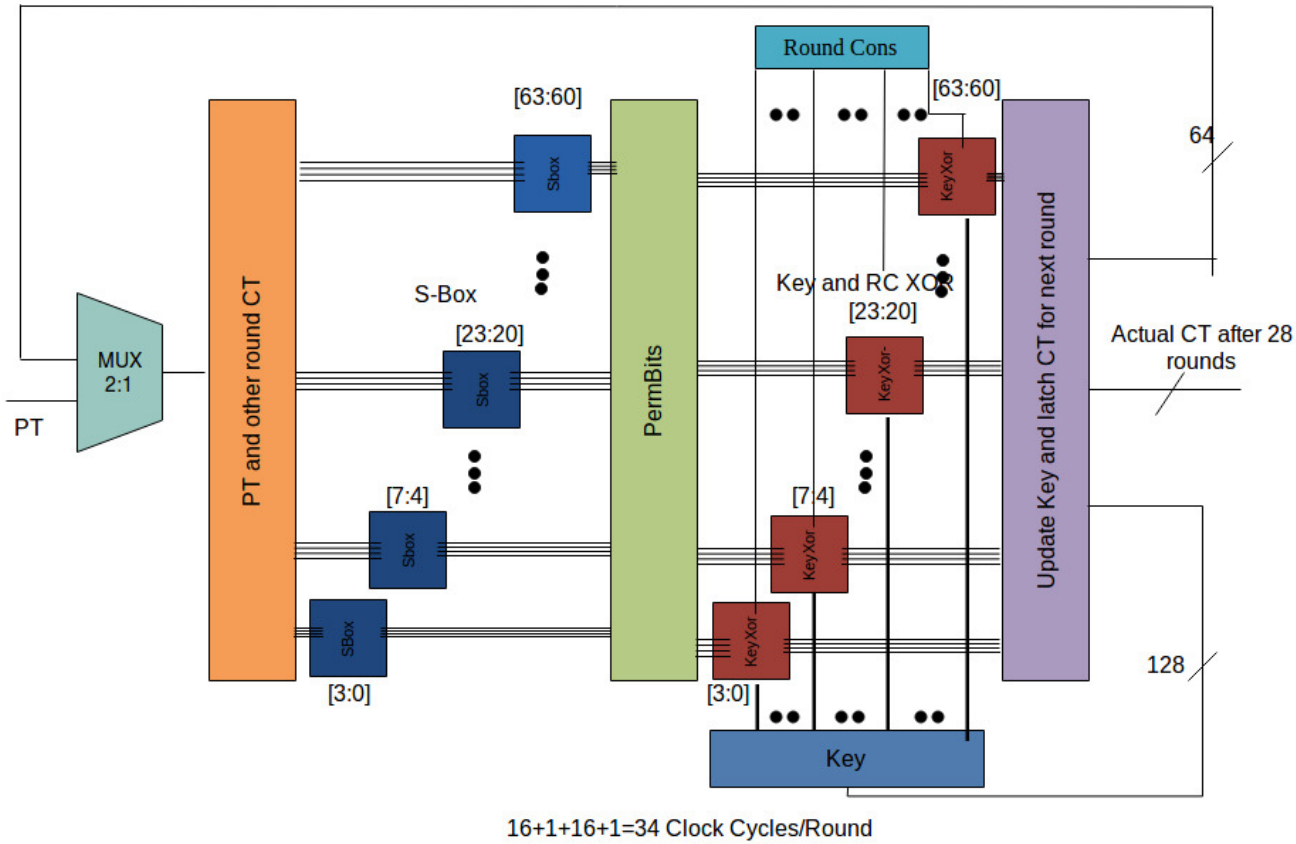

Figure: GIFT64- Serial Implementation Architecture on FPGA

GIFT64-Serial Implementation on FPGA

1. **Clock Cycle Description**
   GIFT64 is a 28 round cipher, the serial implementation is per nibble (4 bit) so each round takes 34 clock cycles.
   - **SubCell**-16 clock cycle
   - **PermBits**-1 clock cycle
   - **AddRoundKey**-16 clock cycle

   One clock cycle is used to update the Keys and feed the output of the present round to the input of the next round.

GIFT64-Serial Implementation on FPGA

1. **Design Summary Report Details**
   The entire architecture is written in verilog and fit perfectly well in Virtex7, following are key architectural details.
   - Number of Slice Registers: 1,053
     - Number used as Flip Flops: 861
     - Number used as Latches: 192
   - Number of Slice LUTs: 1,099
     - Number used as logic: 1,099
     - Number using O6 output only: 766
     - Number using O5 output only: 0
     - Number using O5 and O6: 333
   - Best case available: **2.956ns** (Least Time period of the Clock)

GIFT-Property

GIFT Bit Permutation Property

The output bits of a Sbox in Qx go to 4 distinct Sboxes in Rx in the next round after AddRoundKey layer.

- $Q_x = Sb_{4x}, Sb_{4x+1}, Sb_{4x+2}, Sb_{4x+3}$
- $R_x = Sb_x, Sb_{q+x}, Sb_{2q+x}, Sb_{3q+x}$ where q = s/4 , 0 $\geq$ x $\leq$ q-1, s=16 in GIFT-64

$Q_0 = Sb_0, Sb_1, Sb_2, Sb_3$ and $R_0 = Sb_0, Sb_4, Sb_8, Sb_{12}$ are the two examples of Quotient and Remainder set respectively.

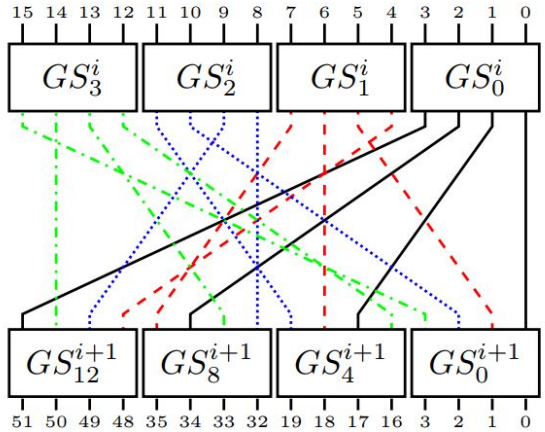

Figure: GIFT Bit Permutation Property

Worked out Example

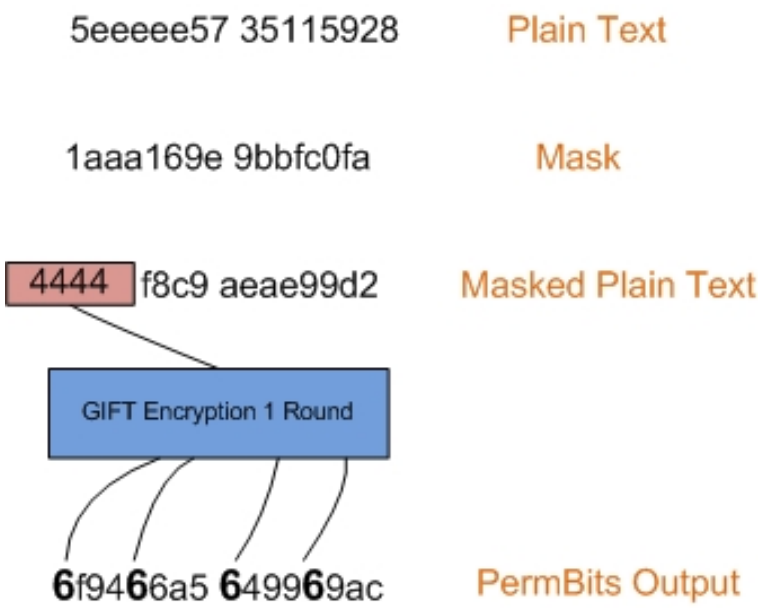

Figure: GIFT Property Explained

If all the input to the Sbox at the Quotient Group Q_x are same.
The output of the PermBits at the Remainder Group R_x are same.

Experiment I- Check the Occurrence of the Property

- Generated a random Plain Text and Mask value using random function(sage and python).
- Obtain the PermBit output in the first round of the GIFT Encryption when the property satisfies.
- Number of iteration for which the script was executed 2^{26}
- The frequency of occurrence of the property $= 2^{10}$

This experiment also helps in creating test cases with the property explained.

Experiment II- Extracting Mask, Simulation

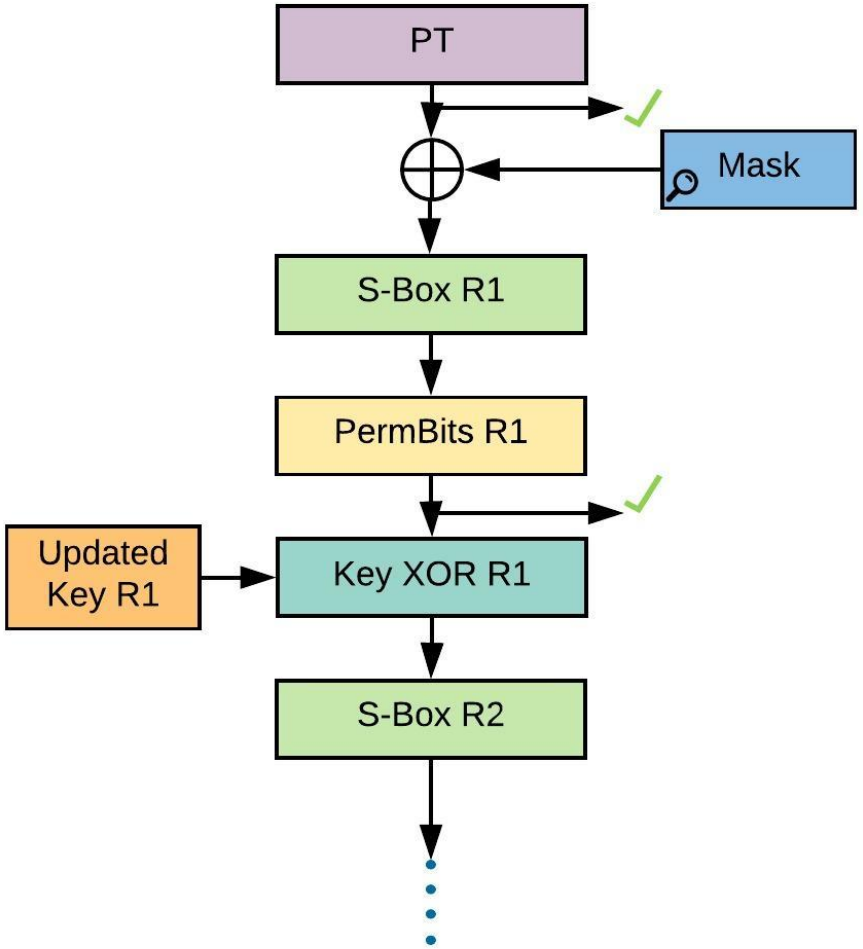

Figure: GIFT Unfolded Architecture: Extracting Mask

Experiment II- Extracting Mask

- PT and the PermBit Output (Y_i) satisfying the property known.
- **Obtain the masks values**.
- PT is denoted as x_i(1 nibble) and Mask m_i(1 nibble)
  According to the Property if the remainder set of the Y_i is same its corresponding quotient set of the input to Sboxes must be same.
  So, $x_0 \oplus m_0 = x_1 \oplus m_1 = x_2 \oplus m_2 = x_3 \oplus m_3$.
  Since x_0, x_1, x_2, x_3 are know following can be written for m_i.
  $m_0 = 0$ to ff (16 possible values)
  $m_1 = (x_0 \oplus m_0) \oplus x_1$.
  $m_2 = (x_1 \oplus m_1) \oplus x_2$.
  $m_3 = (x_2 \oplus m_2) \oplus x_3$.
  Now, the entropy for masks value is reduced from 2^{16} to 2^4 which is reasonable enough.

Experiment II- Extracting Mask, Results

- PT (in hex)-**031764ad ae2e85a5**
  Mask (in hex)-**657147ae 0ad535bf**
  Masked PT(in hex)- **4444f8c9 aeae99d2**
  PermBits Out(in hex)-**6**f94**6**6a5 **6**4996**9**ac
  Actual Mask (in hex)-**6571**
- Following are the obtained Masks (in hex) all 16 possible
  0317 1206 2135 3024 4753 5642 6571 7460 8b9f 9a8e a9bd b8ac cfdb deca edf9 fce8

Experiment III- Extracting Keys, Simulation

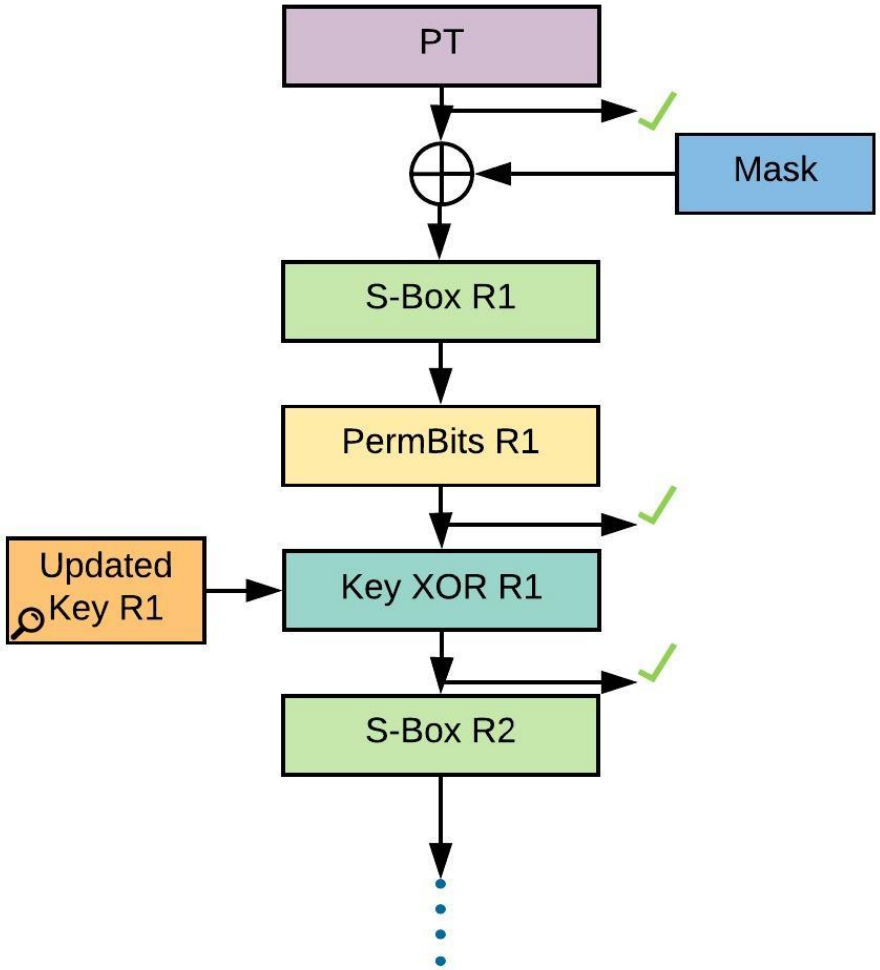

Figure: GIFT Unfolded Architecture: Extracting Keys

Experiment III- Extracting Keys

- PT, PermBit Output Y_i, CT at the end of first round Z_i satisfying the property is known.
- Since each Sbox in case of GIFT uses 2 bit of Key, **obtain the 2x4(Sbox)=8 bit of key.**
- Using the previous Experiment we get the 16 possible masks value m_i.
- Create a matrix of all possible Keys (Row) and masks value (Cols).

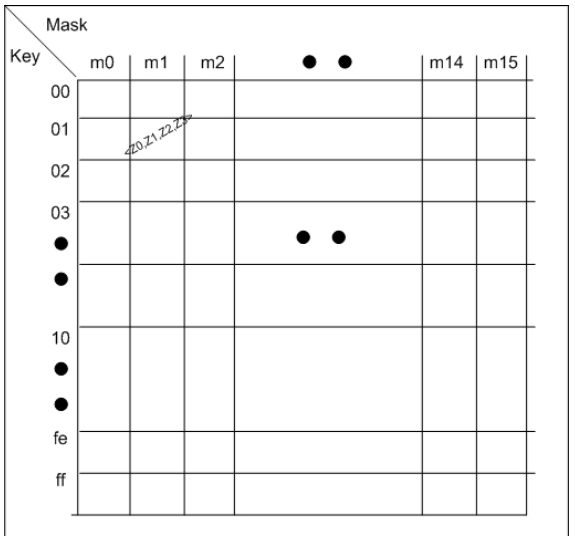

Figure: Key Mask Matrix

Experiment III- Extracting Keys Cont.

- Now for each value of the key and mask pair obtain the corresponding Z_i, values coming from the Remainder set of the PerBits Output.
- Use the actual Z_i values and try to obtain the Key mask pair through matrix element matching.

Experiment III- Extracting Keys, Results

- **Obtained 4 matching cases for the (Key, mask) pair** for every test cases tried and Key turn out to be same but masks are obviously different for a particular key.
- PT(in hex)-**031764ad ae2e85a5**
  Mask(in hex)-**657147ae 0ad535bf**
  Actual Key(4 bit U, 4 bit V)(in hex)-**7f**
  Actual Zi(in hex) from the corresponding remainder set of PermBits-**a000**
  Following are the (key, mask) pair obtained

Keys(UV)	Mask
70	8b9f
7f	6571
80	deca
8f	0317

- It can be observed that the Hamming Distances of the Possible Keys with the Actual Keys are very less.

Experiment IV- Extracting Key based on Hamming Weight(HM), Simulation

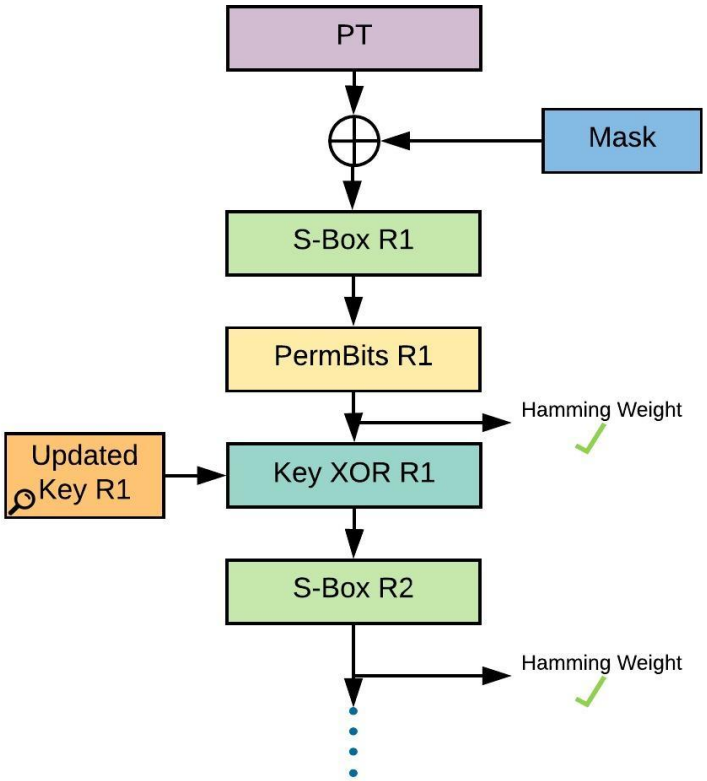

Figure: GIFT Unfolded Architecture: Extracting Keys based on Hamming Weight

Experiment IV- Extracting Key based on Hamming Weight(HM)

- HM of PermBit Output Y_i, HM of next round Sbox Output Z_i known.
- Creating a matrix for all possible values of Y_i where Remainder Set elements are all same (16 values Eg:- 0000, 1111, ..., aaaa, ..., ffff) (Cols) and all possible values of Keys involved (Rows) i.e from 0 to ff because four Sbox each uses 2 bits of Keys.
- Use the actual HM of Z_i and try to find the possible Key Y_i pairs.

Experiment IV- Extracting Key based on Hamming Weight(HM)-Results

- It was observed that the number of possible pairs after matching the HM varies for different PT and Keys. However, for all the possible variation of the PT for a particular key, the 4 possible Keys can be extracted after unification of the pairs for different PT.
- Key(in hex)= **BD1320002B4E3AA946DBA6547C9F9AF9**
  The table(next slide) shows the possible collisions with different PermBits Out for same key. The 4 possible keys obtained after unification are the same as in Experiment 3 shown below with actual key being 7f in hex.

Key	HM
70	3444
7f	3444
80	3444
8f	3444

Same Remainder Set in PermBits Out (Yi) for Different PT	No. of Poss (Key,Yi) by Matching the HM Weight Output of Round II SBox (Zi)
1e191204 189e1611	40
3d2c34eb 387334f3	96
6f9466a5 649969ac	4
aeb0a7a3 a855acaf	228
03b20d74 0a3900c3	204
21f52661 22212e05	204
f8cef6e4 f94ff40e	120
c590cf71 c314cb6e	4
8cee8b96 8c0386bd	228
efa2eddc e08eee1d	12
53ea5025 5605528d	120
7aaa73ba 7a437a09	140
b2a0b76a bc60b268	40
d0dbdc2b d370d3a3	20
92399d23 995696ea	216
4d654dc6 4a944c7c	108

Correlation Power Analysis on GIFT

Algorithm 1: Correlation Power Analysis

Input: trace[NSample][NPoint], hPower[NSample][NKey], Ciphertext[NSample],meanH[i],meanTrace[j],i, j
Outputresult[NKey][NPoint]

```
for k ← 0 to NCipher by 1 do
    temph = hConsumedPower[i[k] - meanH[i]
    tempt = trace[j][k]- meanTrace[j]
    sumHT += temph*tempt
    sumSqH += temph*temph
    sumSqT += tempt*tempt
end
result[i][j] = sumHT/sqrt(sumSqH*sumSqT)
end
```

Side Channel Attack Setup

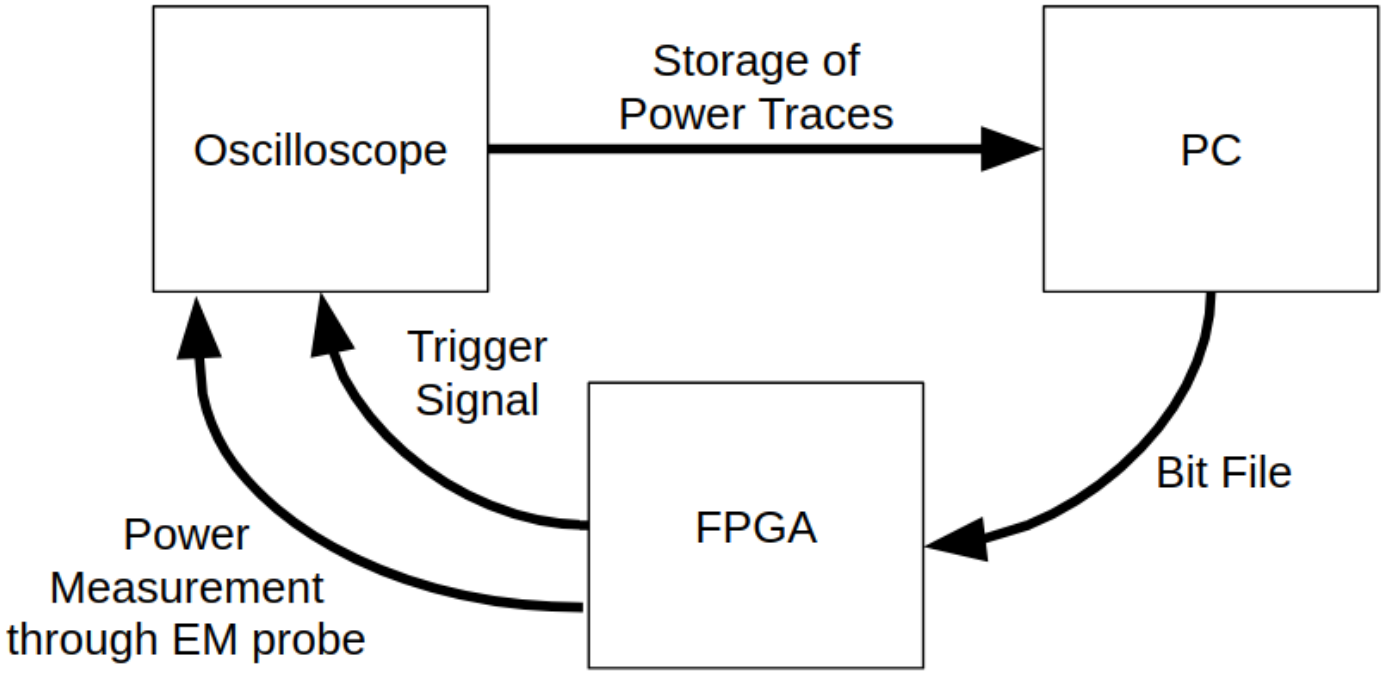

Figure: Power Trace Collection Setup.

Sample Traces

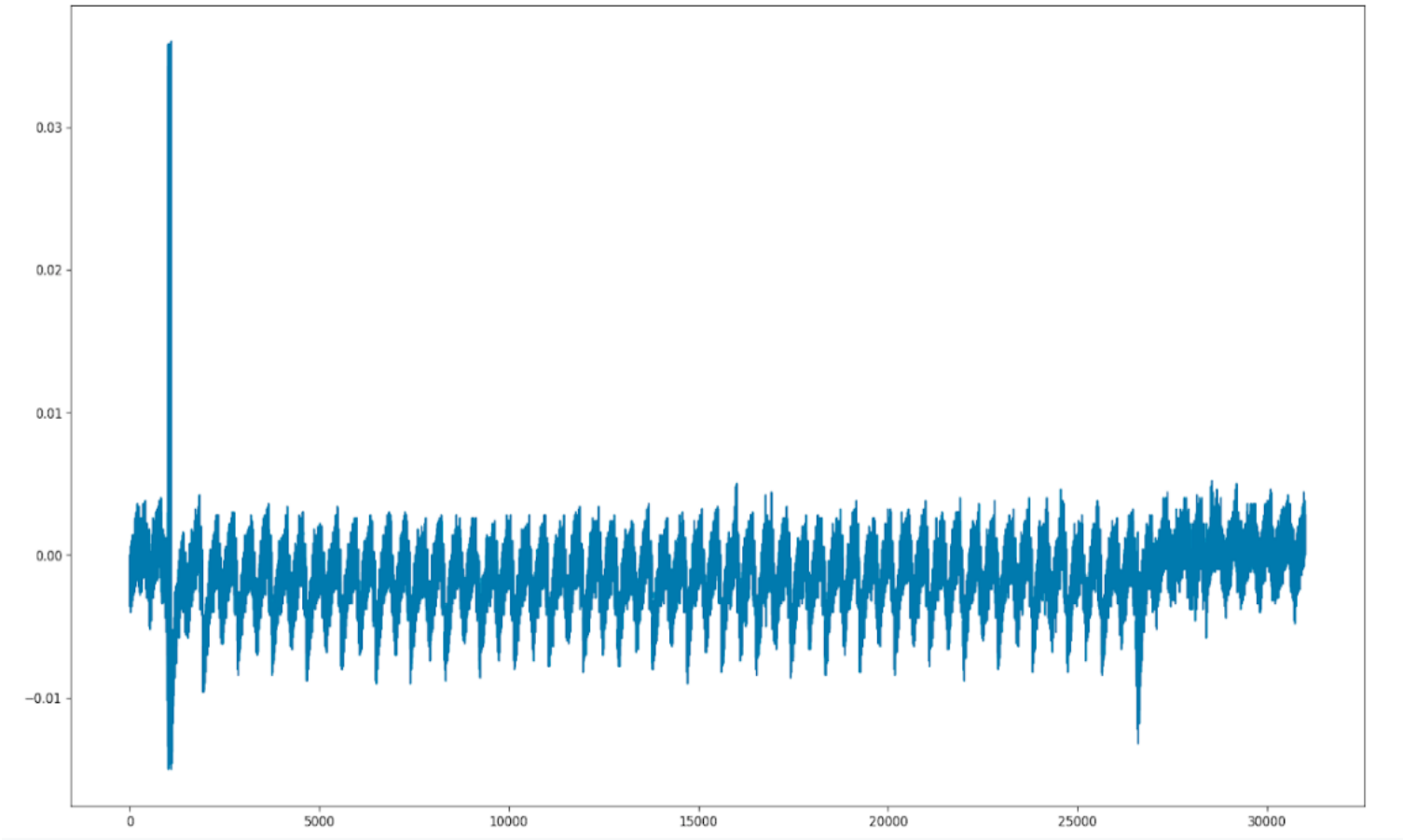

Figure: GIFT-64 entire 28 rounds power trace.

Enlarged Round One Power Trace

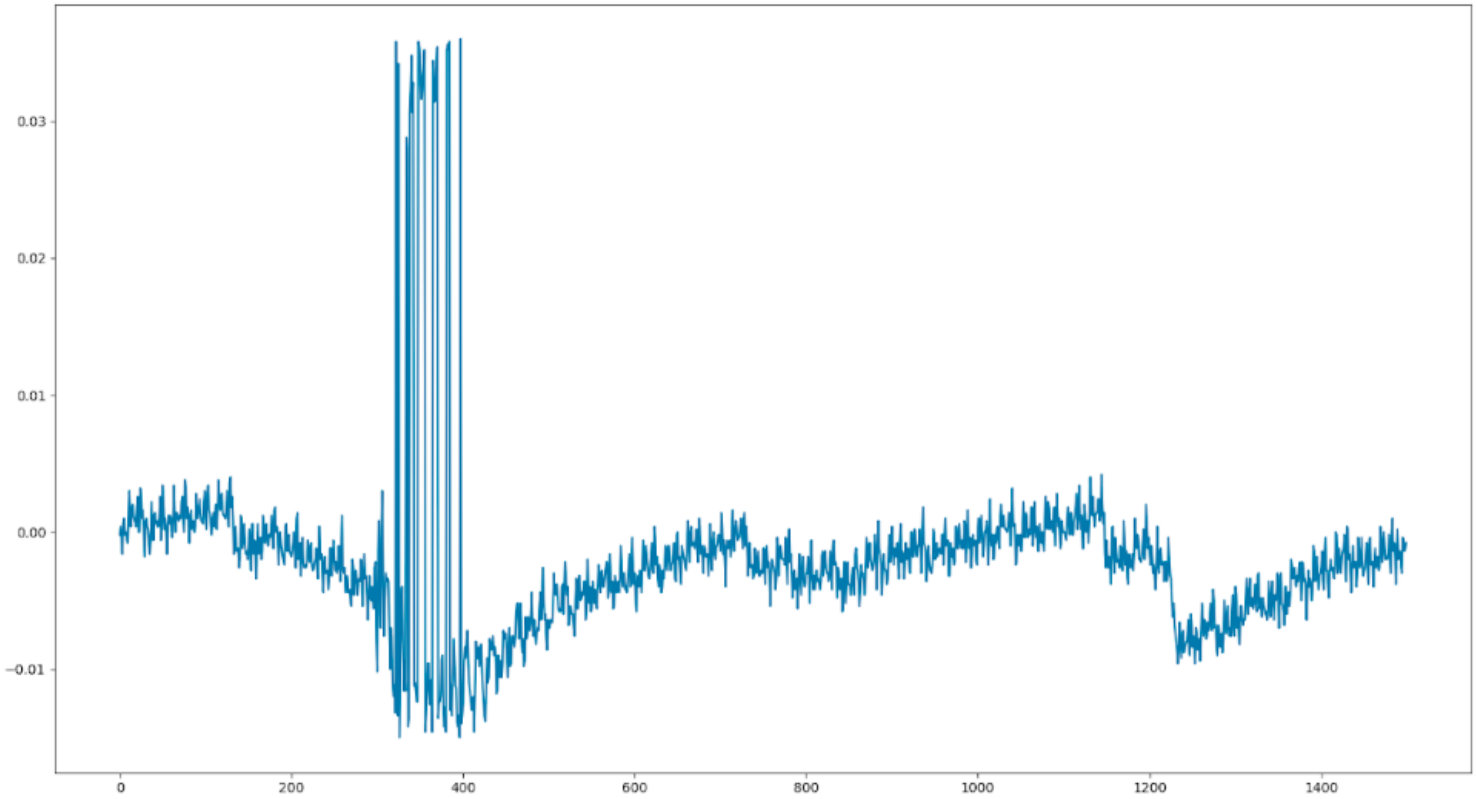

Figure: Enlarged Round One Trace.

CPA Results

- The window corresponding to the first round where we wish to attack the Key bits is shown in the previous slide. **The spike**, in the beginning, corresponds to the **latching operation** where all the inputs variables are latched to the registers which are used later on.
- The CPA was performed on **2000** traces to obtain the key involved in the first two Sboxes. As we know in GIFT each of the S-Box uses **two bits of keys**. We performed CPA attacking two nibbles a time in a divide and conquer strategy. The correlation results for the guessing 4 bit or one nibble of the key for the first two S-boxes is shown in the next slide.

CPA Results

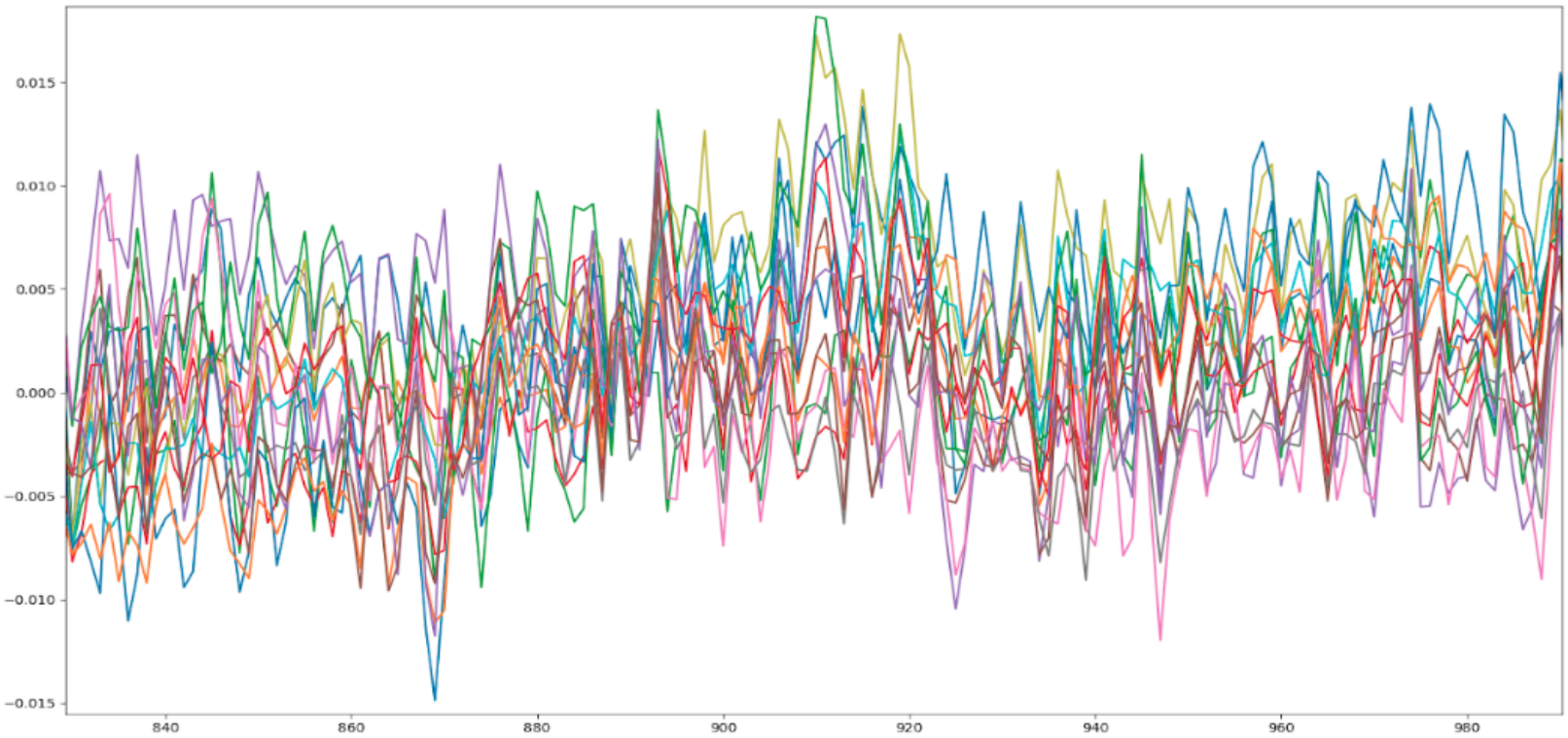

Figure: The correct key being 0xc is verified corresponding to the peak of the trace.

Threshold Implementation(TI) GIFT[3]

TI techniques randomizes the cipher execution using different variation of secret sharing and implementation techniques. **Makes Side Channel Attacks difficult**.

Popular TI techniques includes following:

- Sharing using Decomposition of SBox with cubic algebraic degree-3-Share
- Sharing using Decomposition with one SBox for all- combined 3-shares
- Direct Sharing 4 share

N., Gupta et.al(2017) proposes TI implementation of GIFT (round-based implementation) based on trade off analysis of Throughput and Area.

[3]Naina Gupta, Arpan Jati, Anupam Chattopadhyay, Somitra Kumar Sanadhya, Donghoon Chang: Threshold Implementations of GIFT: A Trade-off Analysis. IACR Cryptology ePrint Archive 2017: 1040 (2017)

Conclusion and Future Scope

- GIFT is the **most energy efficient lightweight cipher** presently.
- Need to modify the existing property based on the Bit Permutation in Threshold Implementation of GIFT.
- More simulation attack based on Hamming Weight (HM) needs to be performed considering the Differential Power Analysis SCA.
- The proposed attack works for a simple masking scheme. But when the TI is used the search space increase and the attack is difficult.

Questions

Questions?