LED - Block Cipher

Team - Crypton

Department of EECS
Indian Institute of Technology Bhilai

November 27, 2020

Outline

Introduction from Applications Perspective

- AES and the ciphers based on it are often optimised for fast software implementations
- Most of these ciphers generally lag behind in providing a light implementation in hardware.
- Huge surge in Internet of Things(IoT) and highly-constrained devices and their applications
- Need for a compact cipher for hardware along with good performance in software implementation
- That's where LED comes into the picture!

Three Key Goals of LED

- Super-light key schedule Algorithm
- Resistance to related key or single key attacks
- A compact cipher for hardware along with good performance in software implementation

Comparison with other ciphers

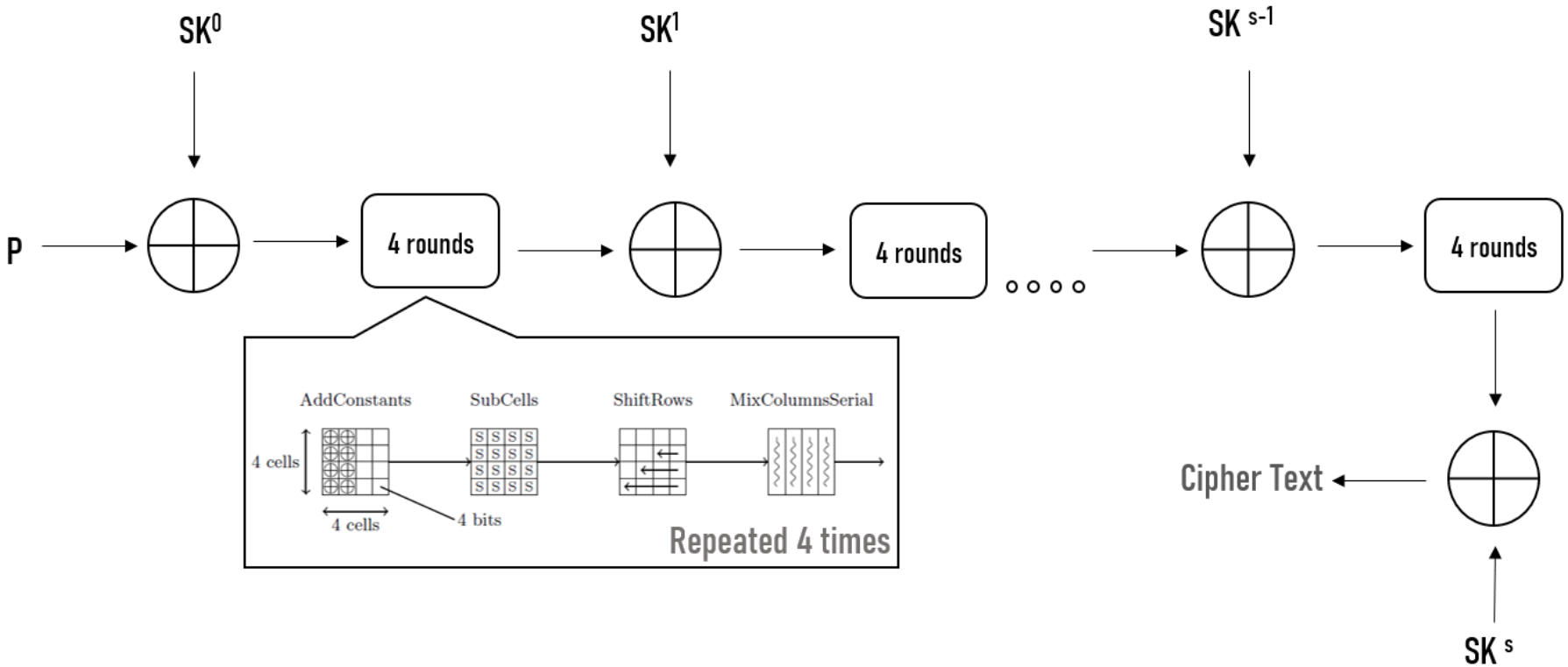
- In general, we can say that most of the lightweight block ciphers are vulnerable to key-related attacks
- HIGHT has a known related-key attack (K+-2010). Hummingbird-1 and KTANTAN are known to have practical related-key attacks(S-2011 and A-2011 respectively).
- LED is resistant from any such attacks in spite of having a practically non existent key-schedule.

Outline

Basic Operations

- `AddConstants`: For any single round, this is the first step where we xor a round dependent constant in the first two columns of the state.
- `SubCells`: For the purpose of confusion, every nibble in the state is replaced using the s-box of LED which is identical to that of PRESENT.
- `Shiftrows`: Here permutation is done by shifting rows. The nibbles are rotated such that the ith line is pushed by i positions onto the left.
- `MixColumns Serial`: For the purpose of diffusion, this is performed but this operation is not identical to that of the AES. Here we apply the special construction of the MDS matrix to every column independently. This operation can be seen as applying a hardware-friendly matrix A four times in order to get the MDS matrix M.

Overall Design



Outline

Related-key attacks

- One of the primary motivation of LED was the resistance against related key attacks.
- This is the major reason why total number of rounds is largely conservative when compared to AES. For instance, the 64-bit variant has a total of 32 AES-like rounds (8 steps * 4 rounds)
- Well known 4 rounds proof for AES which shows that there are at least 25 active sboxes in it. Since one step consists of four rounds, we can say that any differential path of LED will have at least 25 * s active s-boxes.
- The simple or non-existent key scheduling algorithm as stated above becomes before useful to set upper bounds on the number of active s-boxes even under the setting of related key

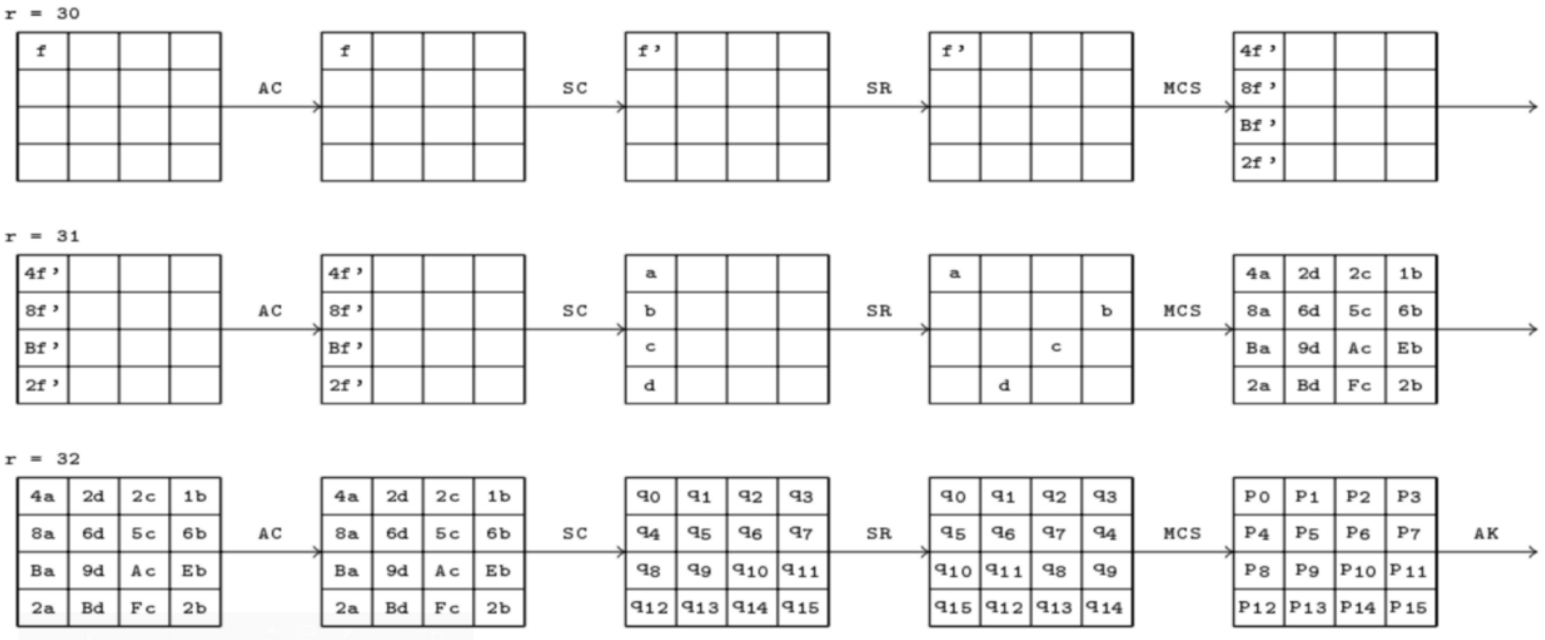
Other cryptanlysis techniques

- LED is practically resistant to them due to its conservative approach in the number of rounds.
- In integral attacks with known-key model, it reaches only 7 rounds with a time complexity of 2^{28}.
- The same goes for zero-sum partitions attack where there is a distinguisher for at most 12 rounds with a time complexity of 2^{64}.
- The classical differential attack shows that we have the upper bound of 2^{-32} on the best differential path probability on 4 active rounds of LED

Outline

Fault Attack

We have implemented a Fault attack on LED64 which consists of 32 rounds.



Invert Functions

The XOR difference derived from the expressions of C and C'(faulty) is computed and compared with the corresponding fault value. To derive the fault equations, we need to invert each round.The inverse calculation is as follows:
(**AddKey**)$^{-1}$: $c_i + k_i$ and $c_i' + k_i$
(**MixColumnsSerial**)$^{-1}$: The inverse matrix is calculated and then is multiplied with the state matrix. Inverse matrix is:

$$M^{-1} = \begin{pmatrix} C & C & D & 4 \\ 3 & 8 & 4 & 5 \\ 7 & 6 & 2 & E \\ D & 9 & 9 & D \end{pmatrix}$$

(**ShiftRows**)$^{-1}$:The nibbles are rotated such that the i^{th} line is pushed by i positions onto the right.
(**SubCells**)$^{-1}$: Inverse of the LED Sbox is taken.

Fault Equations

Fault equations for a.

$$\begin{aligned}
\mathtt{4}\cdot a = {} & S^{-1}(\mathtt{C}\cdot(c_0+k_0)+\mathtt{C}\cdot(c_4+k_4)+\mathtt{D}\cdot(c_8+k_8)+\mathtt{4}\cdot(c_{12}+k_{12}))+ \\
& S^{-1}(\mathtt{C}\cdot(c'_0+k_0)+\mathtt{C}\cdot(c'_4+k_4)+\mathtt{D}\cdot(c'_8+k_8)+\mathtt{4}\cdot(c'_{12}+k_{12})) \qquad (E_{a,0}) \\
\mathtt{8}\cdot a = {} & S^{-1}(\mathtt{3}\cdot(c_3+k_3)+\mathtt{8}\cdot(c_7+k_7)+\mathtt{4}\cdot(c_{11}+k_{11})+\mathtt{5}\cdot(c_{15}+k_{15}))+ \\
& S^{-1}(\mathtt{3}\cdot(c'_3+k_3)+\mathtt{8}\cdot(c'_7+k_7)+\mathtt{4}\cdot(c'_{11}+k_{11})+\mathtt{5}\cdot(c'_{15}+k_{15})) \qquad (E_{a,1}) \\
\mathtt{B}\cdot a = {} & S^{-1}(\mathtt{7}\cdot(c_2+k_2)+\mathtt{6}\cdot(c_6+k_6)+\mathtt{2}\cdot(c_{10}+k_{10})+\mathtt{E}\cdot(c_{14}+k_{14}))+ \\
& S^{-1}(\mathtt{7}\cdot(c'_2+k_2)+\mathtt{6}\cdot(c'_6+k_6)+\mathtt{2}\cdot(c'_{10}+k_{10})+\mathtt{E}\cdot(c'_{14}+k_{14})) \qquad (E_{a,2}) \\
\mathtt{2}\cdot a = {} & S^{-1}(\mathtt{D}\cdot(c_1+k_1)+\mathtt{9}\cdot(c_5+k_5)+\mathtt{9}\cdot(c_9+k_9)+\mathtt{D}\cdot(c_{13}+k_{13}))+ \\
& S^{-1}(\mathtt{D}\cdot(c'_1+k_1)+\mathtt{9}\cdot(c'_5+k_5)+\mathtt{9}\cdot(c'_9+k_9)+\mathtt{D}\cdot(c'_{13}+k_{13})) \qquad (E_{a,3})
\end{aligned}$$

The correct key will satisfy the fault equations derived above.In this attack, we identify large sets of key candidates which are inconsistent with some fault equations and exclude them from our keyspace.

Key Filtering

- Key Tuple Filtering:
  - Every fault equation depends on 4 key indeterminates i.e. from k_0 to k_{15}.
  - The result of the fault equation must belong to Galois Field 16.
  - The correct key will satisfy all the 16 fault equations.
  - All such possible key intermediates are stored as 4 tuples.
- Key Set Filtering: Now we have the fault values of a,b,c and d. We can use this information to further reduce the keyspace by calculating which of these values regularly satisfy all 16 fault equations.
- Exhaustive Search: Now we have the set of all possible key values in the keyspace and now we will apply exhaustive search(brute force).

Finally, we were able to reduce our keyspace from 2^{64} to 2^{24}.

Outline

Conclusion

- It is apparent that its design and the non-existent key schedule in particular is very interesting.
- As mentioned by its designers, the cipher has not been much cryptanalysed independently. Therefore, it should not be deployed straightaway into devices.
- A major takeaway from this study has been the revival of the neglected field of key-schedule design
- This block cipher might also act as a gateway for other researchers to come up with efficient ciphers in the design space for constrained hardware implementations due to its growing applications every day.

Thanks

Team Members

- Aniket Raj
- Kumar Shivendu
- Aastha Asthana

Implementation Info

- Github Link : ¡link¿