Urmi Patel

Ryerson University

Project Report: ESTATE

Introduction

Lightweight cryptography is an encryption method used to run with very low computing power. In other words, lightweight cryptography is a balance between lightweightness and security. ESTATE is an authenticated encryption scheme based on tweakable block cipher and follows a MAC-thenencrypt style of construction. Generally, cipher has two inputs, one is key and second is message but in the case of tweakable block cipher, there is an additional input called tweak. Key and tweak are both used to select permutations, and the reason behind the use of tweak is its cost-effectiveness, as changing key is far more expensive as compared to changing tweaks.

MAC (message authentication code) is used to protect a messages authenticity with the help of a tag. There are mainly three type of encryptions using MAC, but ESTATE uses MAC-then-encrypt because it hides the plaintext tag, so it cannot leak information about the plaintext. Lastly, ESTATE is known as **E**nergy efficient and **S**ingle-state **T**weakable block cipher-based M**A**C-**T**hen-**E**ncrypt [1] .

ESTATE is an improved version of SUNDAE [2]. SUNDAE is a mac-then-encrypt type of scheme where MAC part is performed by OMAC (One-Key CBC MAC) and the encryption part is performed by OFB (Output Feedback). This encryption method is nonce-misuse resistant, but it does not provide RUP security [3]. The motivation behind constructing ESTATE is to improve on the weaknesses of SUNDAE approach.

ESTATE proposes two tweakable block ciphers named TweGIFT and TweAES. These are variants of GIFT and AES block ciphers. Overall, ESTATE is an inverse-free, single-state, RUP secure creation that does not require any field multiplications.

Background Information

FCBC

ESTATE uses FCBC which is a simple variant of CBC MAC. The algorithm is similar, but the only difference is the number of keys used and how the last block of the message is treated. There are mainly two cases: one where the size of message, M, is a multiple of n which denotes the block length and second where it is not. There are three keys K_1, K_2 and K_3 required for FCBC. The key used depends on whether padding is added. The algorithm can be summarized in the following steps [4]:

1. Algorithm takes original message, M, and three set of keys K_1, K_2 and K_3
2. If the size |M| of M is a multiple of n, then define the key to be used for last block as $K \leftarrow K_2$
3. Otherwise, if the size |M| of M is not a multiple of n, then pad the message with 1 following by *l*
4. 0s where $l \leftarrow n - 1 - |M| \bmod n$ and the key to be used for last block as $K \leftarrow K_3$
5. Define P to be the resulting message from either step 1 or step 2 such it can be represented in m blocks of size n each, such that $P \leftarrow P_1 \ldots P_m$ and $|P_1| = \ldots = |P_1| = n$.
6. Define the initial cipher block C_0 as *n* 0s $C_0 \leftarrow 0^n$.

7. Now starting with $i \leftarrow 1$ and going until m-1, perform the following actions: $C_i \leftarrow E_{K1}(P_i \oplus C_{i-1})$, where P_i refers to the block of P, $\oplus$ refers to the exclusive OR and E_{K1} means applying the encryption E with key value K_1. This part will continuously take the cipher text from the previous block and use that in combination with the current block *i* of P in order to generate the cipher *i*. This action happens until the second last block.
8. For the last block perform the following action: $E_K(P_m \oplus C_{m-1})$, where the key, K, would be either K_2 or K_3 depending on the result from step 2 and step 3. The outcome of this step provides the tag T which represents the MAC of the original message M.

The visual representation for first case of FCBC can be seen in Figure 1 and the second case of FCBC can be seen in Figure 2.

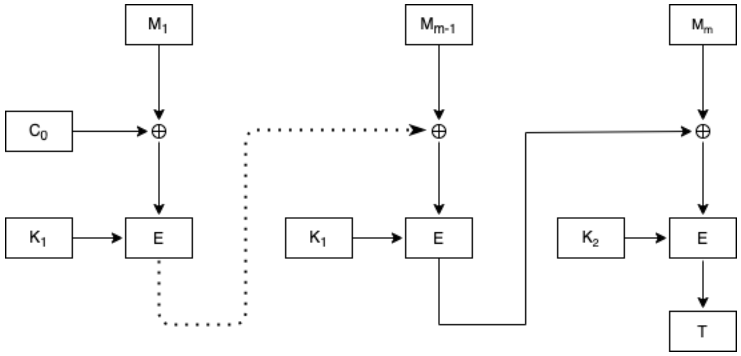

Figure 1: FCBC without padding

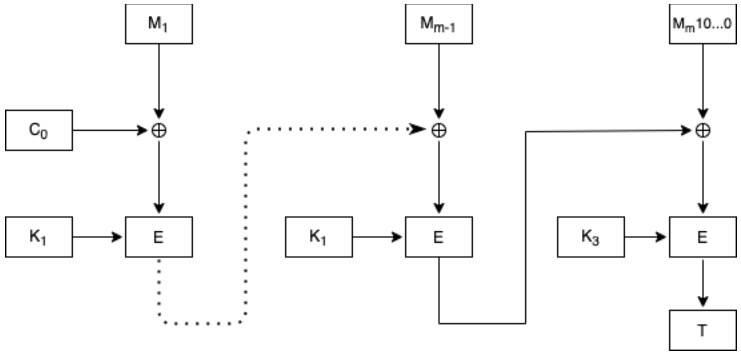

Figure 2: FCBC with padding

One cannot add '000' in padding because the message remains the same after padding zero. The idea is to pad with '100..00', where 1 indicates the beginning of padding. One can also add a new dummy block but for FCBC, no dummy blocks are required as the ambiguity is resolved by use of keys K_2 and K_3.

OFB

Output feedback (OFB) is a mode of operation for a block cipher and is used to encrypt a message and decrypt a cipher text. OFB is similar to CFB [5], it uses a mode of operation to operate a block cipher as a stream cipher. This mode provides confidentiality by use of nonce as an initial vector (IV). Nonce

meaning that the value must be unique for each execution of the algorithm for the given key K. At the same time, nonce helps to increase the complexity of the algorithm. The encryption algorithm can be explained in the following steps [6]:

1. The IV is used as a first input l_1 into the cipher function of the block cipher to retrieve the first output O_1.
2. An exclusive OR is performed between the output O_1 and plain text P_1 in order to obtain the first cipher text C_1.
3. The output O_1 is used as the input for the cipher function in order to obtain second output O_2.
4. An exclusive OR is performed between the output O_2 and plain text P_2 in order to obtain the first cipher text C_2.
5. This process is continuously performed on each previous output and plain text until the last block.
6. For the last block, if it is a partial block of size, *u*, then most significant bits of the previous output O_{n-1} are used for the exclusive OR along with the plain text P_n in order to retrieve cipher text C_n from which *b* - *u* bits are discarded where *b* refers to the size of a full block.

The main advantages behind the use of OFB is, if there are any bit errors out in the output of the encryption algorithm, it will not be propagated because C_1, C_2, up to C_n is generated perfect which you can see in below Figure 3.

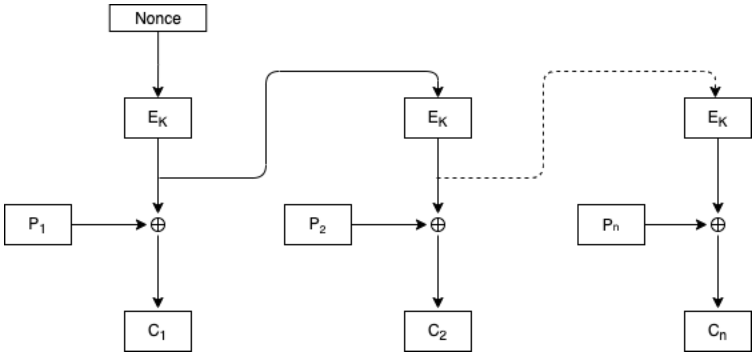

Figure 3: OFB encryption process

AEAD Approach for ESTATE

AEAD stands for authenticated encryption with associated data where AE provides privacy and confidentiality, and AD provides integrity. ESTATE uses AEAD approach and receives an encryption key K, a nonce N, an associated data A, and a message M, as inputs and returns a ciphertext C, and a tag T. ESTATE have four different cases with associated data and message, as each one will be discussed as follows.

Case1:

This case involves both associated data and message. Figure 4 shows the tag generation process and further encryption process with associated data blocks and message blocks.

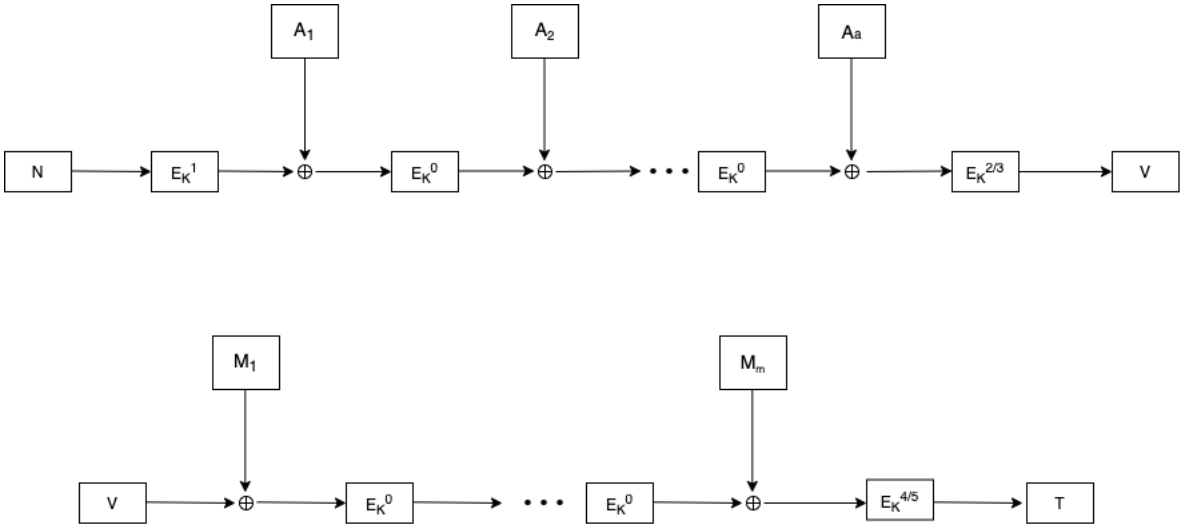

Figure 4: ESTATE with associated data blocks (AD) and message blocks (M)

Here A stands for associated data (AD) and m block stands for message (M). Firstly, encrypt the nonce using tweak value 1. Output of E_K^1 use as an initial value in CBC-MAC like operation over the associated data. Last call is differentiated with tweaks 2 and 3, depending upon whether the data is full or partial. At last V is generated as an output where V stands for intermediate tag on the associated data. Again, compute FCBC like operation over the message to generate the final tag, T. This tag is used as an initial value for the OFB encryption, and this encryption is done the same way as normal OFB operation. You can see the OFB operation in context of ESTATE, in Figure 5.

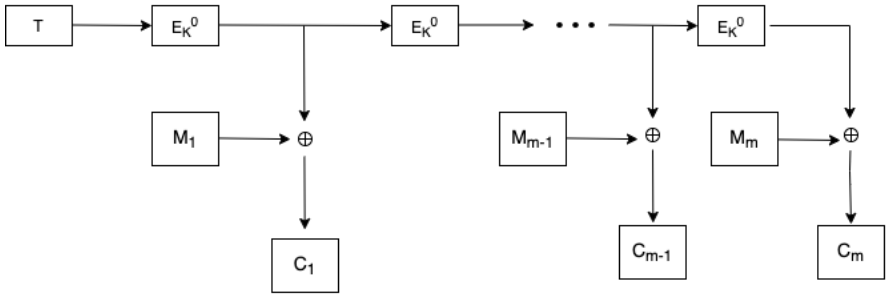

Figure 5: OFB operation in context of ESTATE

Tweaks are used for a domain separation where tweak 1 is used only for nonce processing, but not used anywhere else in the whole process. Moreover, 0 has been used to process all the associated data and messages while 2,3,4,5 used to differentiate the last block.

Case 2:

In this case associated with empty data, simply ignore the first layer of operation and directly process with message blocks. Figure 6 describes detail process for this case.

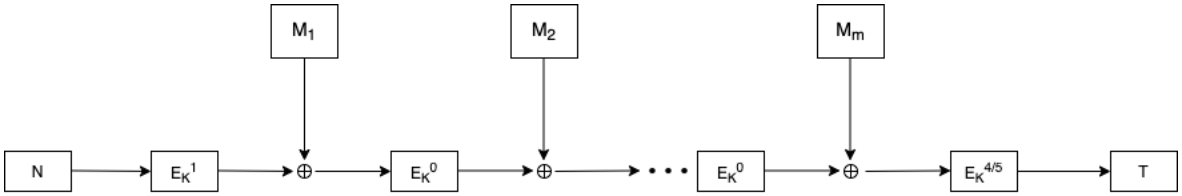

Figure 6: ESTATE with message blocks (M) and empty AD blocks

Again, nonce is used as an initial value and uses FCBC type of operation for tag generation process. Tag T is generated and used as an input in OFB encryption while this process is same as in case 1 and Figure 5 can be viewed for this encryption process.

Case 3:

In this case message data is empty and simply go for AD operation and get tag T. This scheme uses different tweaks 6 and 7 instead of 2,3,4,5. The reason behind use of different tweaks here is that adversary cannot construct any forgery query using this tag T where message is nonempty.

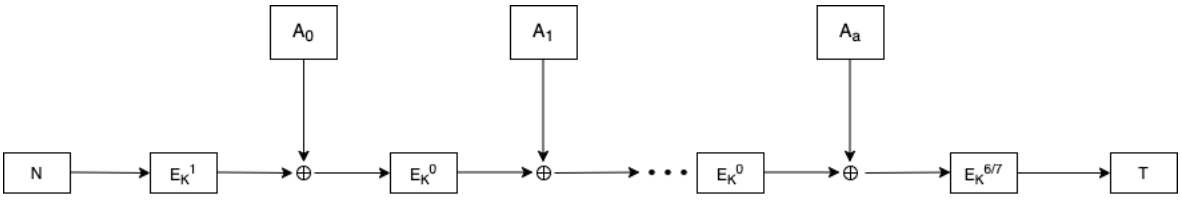

Figure 7: ESTATE with AD blocks and empty message blocks (M)

Case 4:

Suppose AD blocks and message blocks both are empty, simply use unique tweak 8 to encrypt the nonce and that is the tag value.

$$T := E_K^8\ (N)$$

AEAD Approach for sESTATE

Along with ESTATE, a lighter (smaller) version of ESTATE is defined as sESTATE. It replaces the full block cipher E with round reduce block cipher F. This F is always used to process intermediate blocks and uses tweak 15 every time. Whereas the last block of processing uses full block cipher E in both case of associated data and message. sESTATE also has four different cases with empty and nonempty block.

Case 1:

This case involves both associated data and message. Nonce is used as an input and performs FCBC type of operations to get intermediate tag V. The last block uses the same tweak as ESTATE. Generated intermediate tag used as an input in second layer and again performs FCBC operation to get tag T. Obtained tag T used as an input in OFB encryption process which was already seen in Figure 8.

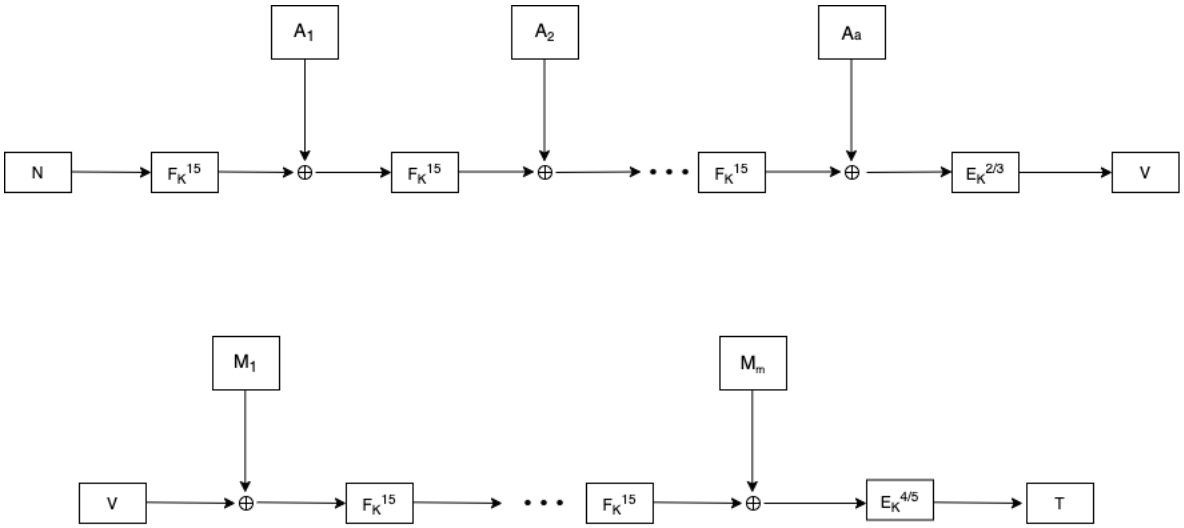

Figure 8: sESTATE with associated data blocks (AD) and message blocks (M)

Case 2:

In this case, associated data blocks are empty and only perform tasks with message blocks. Tag generation process use FCBC type of operation and after that, the use OFB for encryption same as before.

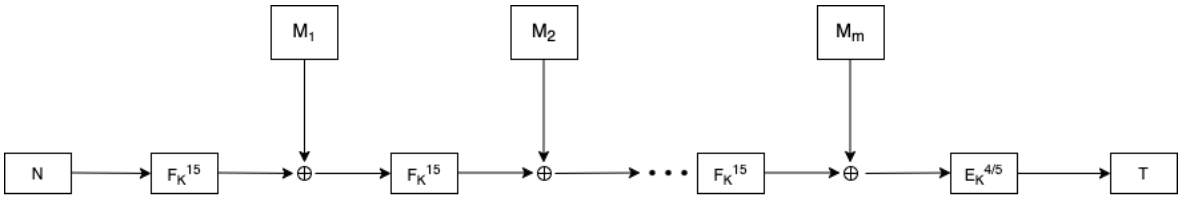

Figure 9: sESTATE with message blocks (M) and empty AD blocks

Case 3:

This case shows empty message bocks and generating tag T, with the help of associated data blocks only. Furthermore, OFB encryption process is done using tag T.

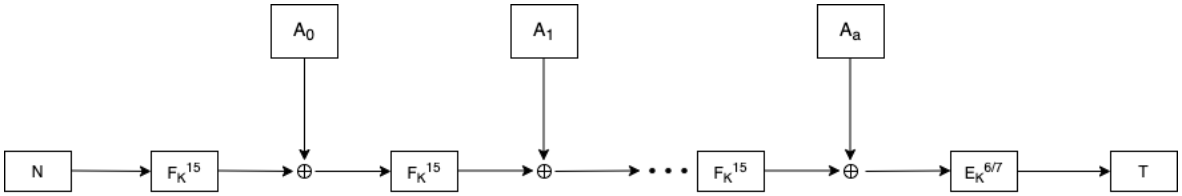

Figure 10: sESTATE with AD blocks and empty message blocks (M)

Case 4:

In the case of empty AD blocks and message blocks, the same ESTATE approach is used for tag generation where tweak 8 is used for encryption process.

$$T := E_K^8\ (N)$$

Choice of Tweaks

SUNDAE is only a block cipher and ESTATE uses the tweakable block cipher approach for domain separation. In the case of ESTATE, 0-8 tweaks are used whereas 0-15 tweaks were used for sESTATE. A small summary of tweaks used during the whole process and why they were used is shown below.

Table 1: Choice of tweaks used for ESTATE

Tweaks	Where	Why
Tweak 0	To process the bulk of messages	Identical to block cipher
Tweak 1	First block cipher call	Ensures RUP security
Tweak 2 and 3	Full and partial AD block processing	For Domain separation
Tweak 4 and 5	Full and partial final plaintext block processing	
Tweak 6 and 7	Non-empty AD and empty message	
Tweak 8	Empty AD and message	

Tweakable Block ciphers

The authors provide three types of block ciphers named TweAES-128, TweAES-128-6 and TweGIFT-128. Each one will be discussed below.

TweAES-128

This block cipher is a modified version of AES-128 or AES-128-128 [7]. The main difference is the use of tweaks on different rounds of encryption. The table below describes specifications for TweAES-128

Table 2: Specification of TweAES-128

Key Size	128 bits
Tweak	4-bit
Block Size	128-bits
Number of Rounds	10

The number of rounds performed by an algorithm is totally dependent on the size of the key. In the key expansion process, the given 128 bits cipher key is stored in [4]*[4] bytes matrix and then the four column words of the key matrix is expanded into a schedule of 44 words. Moreover, the last round of the TweAES encryption does not contain mix column phase which is the same as AES logic. The Figure 11 below illustrates how the tweakable AES encryption works.

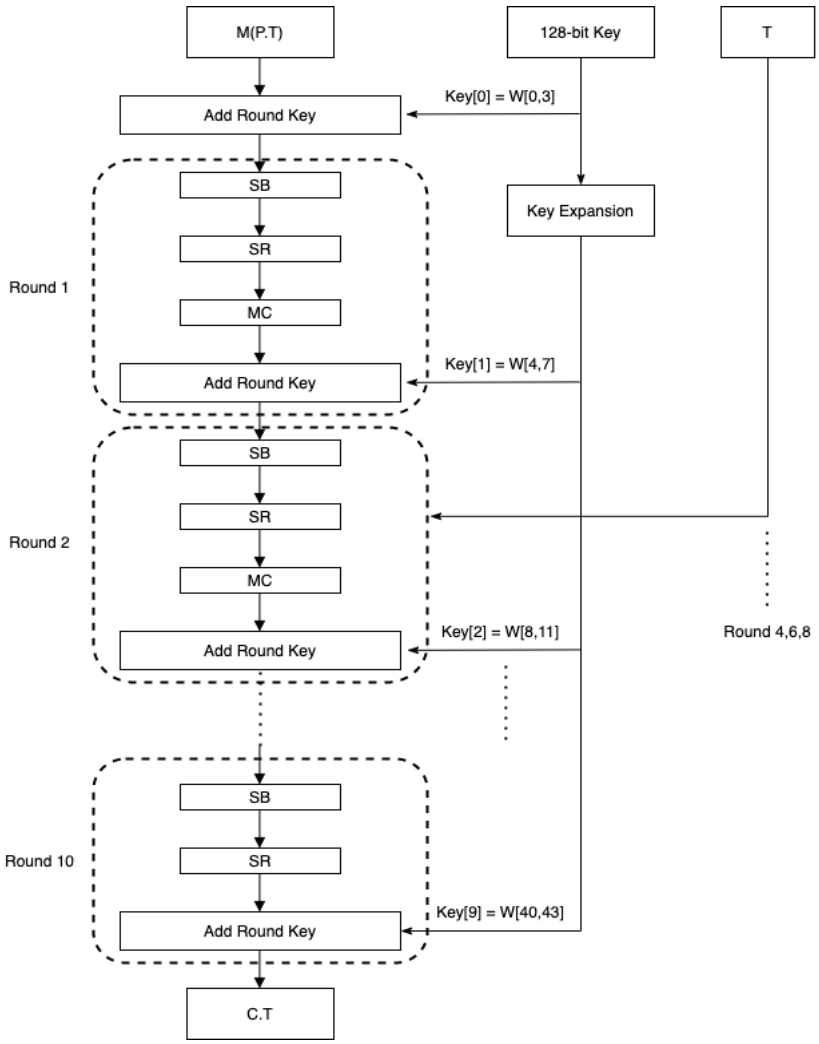

Figure 11: Encryption process of TweAES-128

Main steps of TweAES-128

1. SubBytes (SB): Uses same invertible 8-bit S-box as AES-128 and applies it to each byte of the cipher state. The elements of the S-box are written in hexadecimal system.
2. ShiftRows (SR): Rows of the block are cylindrically shifted in the left direction where the first row is untouched.
3. MixColumns (MC): This step is very important to achieve good diffusion where the block is multiplied with a fixed matrix and multiplication is done using GF(2^8) corresponding with irreducible polynomial of degree 8.
4. AddRoundKey: 128 bits key is extracted from the master key K and XORed with corresponding element of key's matrix. Once this step is done, the keys are no longer available.
5. AddTweak: the 4-bit tweak first expanded to the 8-bit value with the help of linear code and in next step the 8-bit value is XORed to the state at an interval of 2 rounds (2,4,6,8).

TweAES-128-6

This is the rounded reduced version of TweAES-128 and the reason behind adding 6 in the name is the number of rounds performed by an algorithm. The main change with this approach is that the last round (6^{th} round) includes the mix column operation. The table below describes the specification of the algorithm.

Table 3: Specification of TweAES-128-6

Key Size	128 bits
Tweak	4-bit
Block Size	128-bits
Number of Rounds	6

SubBytes, ShiftRows, MixColumns and AddRoundKey operations are performed the same as TweAES-128 but the AddTweak operation is performed with some modification. Again, the 4-bit tweak is expanded to 8-bit tweak and then XORed to the state at an interval of 2 rounds. Here the AddTweak operation called in round 2 and 4 because the algorithm itself has an if condition for ($i < 6$) and that is why the 6^{th} round is not performing AddTweak operation.

TweGIFT-128

This algorithm is the variant of GIFT-128 block cipher [8]. GIFT has some advantages compared to other algorithms such as, smaller area because of smaller S-box, lesser subkey additions and faster key schedule. The table below describes the specification of the algorithm

Table 4: Specification of TweGIFT-128

Key Size	128 bits
Tweak	4-bit
Block Size	128-bits
Number of Rounds	40

Main steps of TweAES-128

1. SubCells: TweGIFT uses the same 4-bit S-box as GIFT and applies a total of 16 4-bit S-boxes in parallel to every nibble of the state.
2. PermBits: This operation uses the same bit permutation as GIFT where the pure bit permutation is done without any XOR gate. Like map bit i to bit P(i).
3. AddRoundKey: In this step, a 64-bit round key is extracted from the master key and added to the cipher state.
4. AddRoundConstant: Add a single bit '1' to the most significant bit, and a 6-bit round contact is XORed to bit 3 of the first 6 nibbles. Cipher state at bit position 127,23,19,15,11,7 and 3 would be affected. The round constant is generated using a 6-bit affine LFSR with 1 XNOR gate same as SKINNY [9].
5. AddTweak: Here the 4-bit tweak is expanded to the 32-bit value with the help of linear code and the 32-bit value is XORed to the state at an interval of 5 rounds. Hence, the rounds 4,9,14,19,24,29 and 34 are affected.

ESTATE With Block Ciphers:

ESTATE_TweAES-128:

The authors use AEAD scheme defined in ESTATE mode of operation with tweakable block cipher TweAES-128. The size of the key, nonce and tag are 128 bits.

ESTATE_TweGIFT-128:

The reason behind use of this combination is to achieve hardware-oriented ultra-lightweight applications. The key size, nonce and tag remain same as before (128 bits).

sESTATE_TweAES-128-6:

For this approach, authors used smaller version of ESTATE and the tweakable block cipher TweAES128 where the nonce, tag and size of key remain 128 bits. The reason behind this approach is to get a higher throughput demanding and energy constrained applications.

Security

The table below describes the information about the amount of time and data required to make higher probability of the attack. The attacker will take 2^{128} time to break an encryption where the size of data is 2^{64}. sESTATE_TweAES-128-6 allows a little bit less data and time as compared to ESTATE_TweAES-128 due to the round reduced function.

Table 5: Security related information

Approaches	Privacy		Integrity	
	Time	Data (in bytes)	Time	Data (in bytes)
ESTATE_TweAES-128	2_{128}	2_{64}	2_{128}	2_{64}
ESTATE_TweGIFT-128	2_{128}	2_{64}	2_{128}	2_{64}
sESTATE_TweAES-128	2_{112}	2_{60}	2_{112}	2_{60}

The authors guarantees that all the algorithms are secure in nonce-misusing situation. They also declared that there are no hidden weaknesses in ESTATE and sESTATE modes of operations.

ESTATE and sESTATE Features

Here we discuss some main features of the ESTATE and sESTATE.

Nonce-misuse Resistant:
ESTATE provides full security even when the nonce can be repeated. Nonce can be any random numbers used to protect private communications. In other words, we can say ESTATE performs deterministic authenticated encryption where the nonce is taken as a first block of the associated data.

Single State:
State size of the ESTATE is small as block size is used in a cipher. At the same time, it will satisfy both the characteristics of lightweightness and high performance.
Multiplication Free:
ESTATE doesn't require any field multiplication. Infact, it is free from any other operations apart from the tweakable block cipher operation itself.

Optimal:
The algorithm tries to reduce the number of additional primitive calls to make the system more optimal. On the other hand, SUNDAE requires additional primitive calls to prepare states depending upon emptiness of data. ESTATE requires some primitive calls to process a block of message and associated data which are the optimal number of non-linear calls required for authenticated encryption.

RUP Secure:
RUP security is where the adversary can observe the unverified plaintext and SUNDAE does not provide RUP security. ESTATE is proven to be INT-RUP [3] secure meaning adversary should not be able to forge, even when given access to unverified plaintext. In more detail, tweak values for the first block cipher call in tag generation and encryption phase which are always distinct. So even if adversary get some unverified plaintext, it cannot get any information regarding tag generation phase.

Robustness:
AEAD requires nonce value, and those values are totally unique and random; this helps to achieve high security. There is always a scenario where the security becomes weak, which is totally dependent on the situation.

Hardware Results

Authors gave some comparison related to the hardware implementation between TweAES-128, ESTATE_TweAES-128, TweGIFT-128 and ESTATE_TweGIFT-128. Virtex 7 FPGA is the hardware used while testing these algorithms. Below is a table to show a comparison of the results.

Table 6: Comparison of hardware results

Primitives	LUTs	FF	Slices	Frequency (MHz)	Clock cycles	Throughput (Mbps)
TweAES-128	1617	524	574	328.27	11	3819.87

ESTATE_TweAES-128	2235	679	1110	314.71	20	2014.14
TweGIFT-128	790	408	336	597.59	41	1865.65
ESTATE_TweGIFT-128	1413	698	616	580.11	80	928.27

LUTs stands for lookup tables that can be prefilled with a bunch of key value pairs that are utilized at some point in the future to quickly find values by key. FF stands for flipflops, used as a switch which can flip the value based on the previous and current incoming values. With more LUTs, it is very hard to crack any system but at the same time it needs more hardware space to store the data structure.

Throughput for ESTATE_TweAES-128 is 50% higher than the ESTATE_TweGIFT-128 and the reason behind this is used number of cycles to process one block. GIFT requires 80 cycles to process one block while AES requires only 20. Hence, a greater number of cycles requires more time to perform and also gives a low throughput.

ESTATE is very efficient for short messages. Throughput of the algorithms also increases with the size of the message. Simply put, when the message size increases, the throughput also increases. Moreover, throughput of ESTATE_TweAES-128 is two times the magnitude of the throughput of ESTATE_TweGIFT-128, meaning ESTATE_TweAES-128 is better in terms of performance.

Conclusion

Overall, ESTATE uses tweakable block cipher based lightweight AEAD. ESTATE has a single-state, RUP secure, energy-efficient, nonce-misuse resistance and robust type of construction. Smaller version of AEAD and ESTATE is very efficient but at the same time, some drawbacks are present in terms of security. As it is well known, light weight cryptography design never satisfied all three modules namely security, cost and speed. At some point, one of those three have to be sacrificed.

Future scope

ESTATE is the improved version of SUNDAE hence it uses the mac-then-encrypt functionality. It might be possible that the use of encrypt-then-mac will be beneficial because it is a more strong system than mac-then-encrypt. The second issue is the design of the algorithm; what if the user wants to verify the encrypted text before decrypting it? It is not possible with this scenario, firstly decrypting the cipher text and second, using FCBC to check the integrity of the message.

Works Cited

[1] N. D. A. J. C. M. L. M. N. Y. S. Avik Chakraborti, "ESTATE," p. 22, March 29, 2019.

[2] A. B. A. L. E. T. Subhadeep Banik, "SUNDAE: Small Universal Deterministic Authenticated Encryption for the Internet of Things," p. 35, 2018.

[3] P. W. H. H. Ping Zhang, "The INT-RUP Security of OCB with Intermediate (Parity) Checksum," p. 34.

[4] P. R. John Black, "CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions," p. 19, 2000.

[5] A. R. R. Mohan H.S, "REVISED AES AND ITS MODES OF OPERATION," *International Journal of Information Technology and Knowledge Management,* p. 8, June, 2012.

[6] M. Dworkin, "Recommendation for Block Cipher Modes of Operation," *NIST,* p. 65, 2001.

[7] "ADVANCED ENCRYPTION STANDARD (AES)," *Federal Information Processing Standards Publication 197,* p. 51, November, 2001.

[8] S. K. P. T. P. Y. S. S. M. S. Y. T. Subhadeep Banik, "GIFT: A Small Present Towards Reaching the Limit of Lightweight Encryption," p. 50, 2017.

[9] C. J. J. K.¨ S. L. G. M. A. P. T. S. Y. S. P. S. S. Beierle, "The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS," *Springer,* pp. 123-153, 2016.

[10] A. B. A. L. B. M. N. M. a. K. Y. Elena Andreeva, "How to securely release unverified plaintext in authenticated encryption," *20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C.,* p. 105–125, 2014.

[11] N. D. A. J. C. M. L. M. N. Y. S. Avik Chakraborti, "Elastic-Tweak: A Framework for Short Tweak Tweakable Block Cipher," p. 39.