



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

BECE204L- Microprocessors & Microcontrollers DA

Name: Saigovardhan S.

Reg No.: 22BCE1704

Sem: Fall 23-24

Slot: E2

Topic: **Tiny Encryption
Algorithm (TEA)**

Abstract:

The Tiny Encryption Algorithm (TEA) is a symmetric block cipher that was designed for simplicity and efficiency in resource-constrained environments. Developed by David Wheeler and Roger Needham in 1994, TEA operates on 64-bit blocks of data using a 128-bit key and employs a Feistel network structure with simple bitwise operations. It is known for its minimalist design, ease of implementation, and a relatively low computational overhead. TEA is primarily used for educational purposes and for applications that require lightweight encryption. However, it is important to note that TEA is not considered secure for modern cryptographic applications due to known vulnerabilities. This abstract provides an overview of TEA's design principles and its role as a fundamental cipher in the field of cryptography.

TEA's design consists of multiple rounds of a basic mathematical operation that mixes the data and key together, making it suitable for applications with limited computational resources. Despite its simplicity and efficiency, TEA is not recommended for securing sensitive data due to its susceptibility to certain cryptographic attacks. It serves as a valuable educational

tool for understanding encryption concepts and lightweight cryptographic techniques.

Introduction:

The Tiny Encryption Algorithm (TEA) is a simple and lightweight symmetric encryption algorithm designed for applications with limited computational resources and a need for basic data protection. It was developed by David Wheeler and Roger Needham in 1994 and is known for its minimalist design and ease of implementation.

TEA operates on 64-bit blocks of data and employs a Feistel network structure, utilizing simple bitwise operations and arithmetic. While TEA is primarily used for educational purposes and in scenarios where lightweight encryption is sufficient, it's important to note that it is not recommended for securing sensitive data in modern cryptographic applications due to known vulnerabilities. Nonetheless, TEA serves as a valuable entry point for understanding fundamental encryption concepts and algorithms.

TEA's appeal lies in its straightforward implementation and efficiency, making it suitable for resource-constrained environments like embedded systems and

IoT devices. Despite its limitations in terms of security, TEA continues to be a valuable learning tool for those interested in cryptography and encryption algorithms.

Novelty:

The Tiny Encryption Algorithm (TEA) introduced several novel aspects when it was first developed:

1. **Simplicity:** TEA was designed to be extremely simple, both in terms of its algorithmic structure and implementation. This simplicity made it easy to understand and implement, which was a departure from more complex encryption algorithms at the time.
2. **Resource Efficiency:** TEA was specifically designed to be efficient in terms of computational resources. It uses basic bitwise operations and requires minimal memory, making it suitable for resource-constrained environments, such as embedded systems and IoT devices.
3. **Feistel Network Structure:** TEA employs a Feistel network structure, a widely used technique in cryptography. This structure contributes to its simplicity while maintaining a good level of security.
4. **32 Rounds:** TEA uses a relatively high number of rounds (32 rounds) for such a simple algorithm, which was considered a novel approach at the time. This

increased the complexity of the encryption process and improved its security compared to earlier attempts at lightweight encryption.

5. **S-Box-Less Design:** TEA notably does not use substitution boxes (S-boxes), which are common components in many encryption algorithms. Instead, TEA relies on the interplay between data and key bits, which was a unique feature in its design.
- While **TEA's** novelty lies in its simplicity and efficiency, it's important to note that its security has been found lacking against modern cryptanalysis techniques. As a result, it is not recommended for securing sensitive data in contemporary applications. Nonetheless, TEA's design principles continue to be of educational value for those interested in understanding the fundamentals of encryption algorithms.

Literature Review:

1. Origins and Design Principles:

- The original TEA paper by David Wheeler and Roger Needham (1994) laid the foundation for TEA's design principles, emphasizing its minimalistic structure and ease of implementation.

2. Security Analysis:

- Subsequent research has focused on cryptanalysis and security assessments. Papers such as "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations" (Courtois and Pieprzyk, 2002) have identified vulnerabilities and weaknesses in TEA's design.

3. Security Limitations:

- Studies like "Security of TEA: A Short Report" (Robshaw and Wheeler, 1997) have analyzed TEA's resistance to differential and linear cryptanalysis, highlighting its limitations in terms of security.

4. Efficient Hardware Implementations:

- Research has explored efficient hardware implementations of TEA, making it suitable for resource-constrained environments. These efforts have aimed to enhance TEA's performance in hardware-based encryption systems.

5. Comparative Studies:

- Comparative studies, such as "Comparative Study of Tiny Encryption Algorithms for Wireless Sensor Networks" (Kalaivani and Uthariaraj, 2013), have assessed TEA alongside other lightweight encryption algorithms, particularly in contexts like wireless sensor networks and the Internet of Things (IoT).

6. Lightweight Cryptography Survey:

- "Lightweight Cryptography: A Survey"
(Nikova, Rijmen, and Tischhauser, 2017)
provides a broader perspective on
lightweight encryption algorithms, including
TEA. It discusses their characteristics, security
aspects, and applications.

CODE:

```
1  #include <stdint.h>
2
3  #define TEA_ROUNDS 32
4
5  void tea_encrypt(uint32_t* v, uint32_t* k) {
6      uint32_t v0 = v[0], v1 = v[1];
7      uint32_t sum = 0;
8      uint32_t delta = 0x9e3779b9;
9      int i;
10     for (i = 0; i < TEA_ROUNDS; i++) {
11         sum += delta;
12         v0 += ((v1 << 4) + k[0]) ^ (v1 + sum) ^ ((v1 >> 5) + k[1]);
13         v1 += ((v0 << 4) + k[2]) ^ (v0 + sum) ^ ((v0 >> 5) + k[3]);
14     }
15
16     v[0] = v0;
17     v[1] = v1;
18 }
19
20 void tea_decrypt(uint32_t* v, uint32_t* k) {
21     uint32_t v0 = v[0], v1 = v[1];
22     uint32_t sum = 0xC6EF3720;
23     uint32_t delta = 0x9e3779b9;
24     int i;
25     for (i = 0; i < TEA_ROUNDS; i++) {
26         v1 -= ((v0 << 4) + k[2]) ^ (v0 + sum) ^ ((v0 >> 5) + k[3]);
27         v0 -= ((v1 << 4) + k[0]) ^ (v1 + sum) ^ ((v1 >> 5) + k[1]);
28         sum -= delta;
29     }
30
31     v[0] = v0;
32     v[1] = v1;
33 }
```

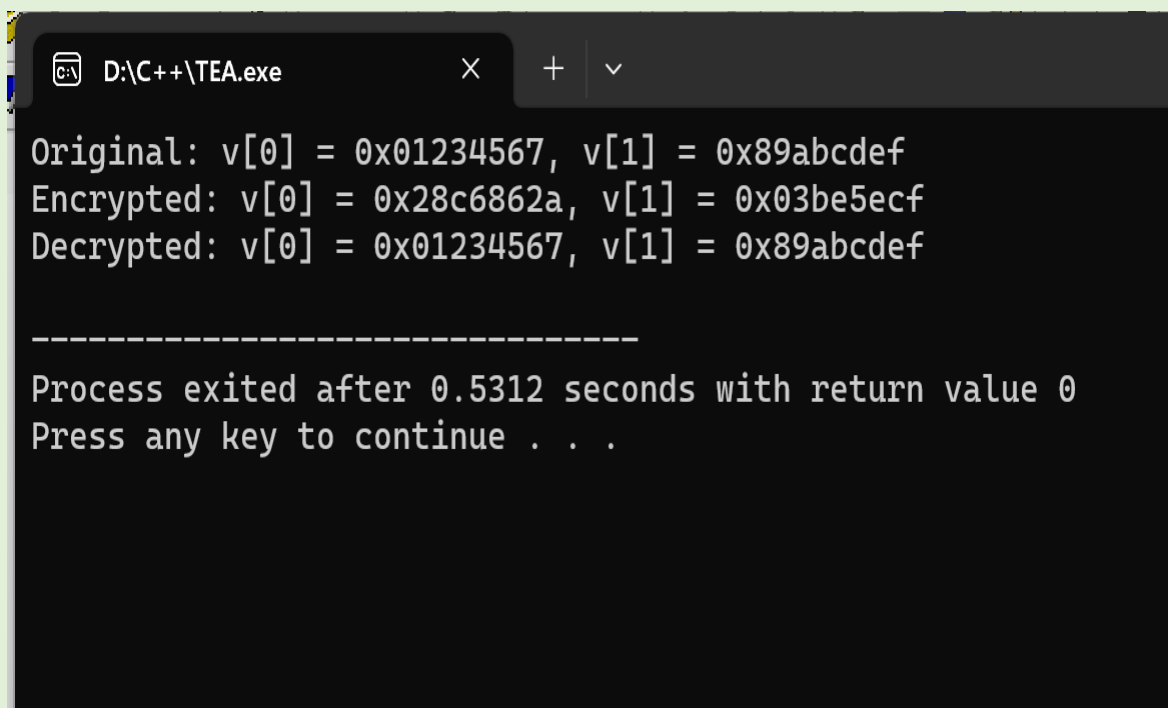


```

30
31     v[0] = v0;
32     v[1] = v1;
33 }
34
35 int main() {
36     uint32_t v[2] = {0x01234567, 0x89abcdef};
37     uint32_t k[4] = {0xFEDCBA98, 0x76543210, 0x13579BDF, 0x2468ACE0};
38
39     printf("Original: v[0] = 0x%08x, v[1] = 0x%08x\n", v[0], v[1]);
40
41     tea_encrypt(v, k);
42
43     printf("Encrypted: v[0] = 0x%08x, v[1] = 0x%08x\n", v[0], v[1]);
44
45     tea_decrypt(v, k);
46
47     printf("Decrypted: v[0] = 0x%08x, v[1] = 0x%08x\n", v[0], v[1]);
48
49     return 0;
50 }
51

```

OUTPUT:



The screenshot shows a Windows command prompt window titled "D:\C++\TEA.exe". The output of the program is as follows:

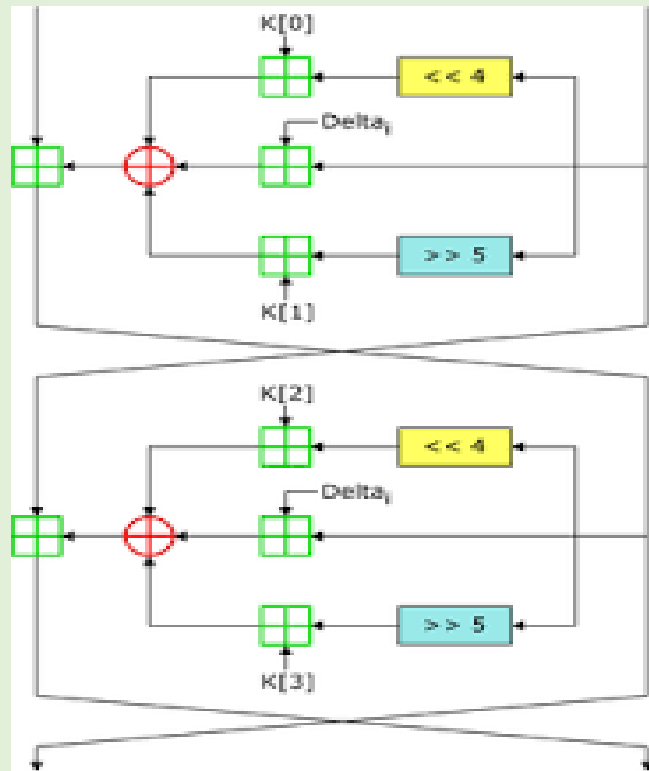
```

Original: v[0] = 0x01234567, v[1] = 0x89abcdef
Encrypted: v[0] = 0x28c6862a, v[1] = 0x03be5ecf
Decrypted: v[0] = 0x01234567, v[1] = 0x89abcdef

-----
Process exited after 0.5312 seconds with return value 0
Press any key to continue . . .

```

TEA Representation:



Algorithm:

Start:

1. Input the data to be encrypted, $v[2]$ (two 32-bit blocks).
2. Input the encryption key, $k[4]$ (four 32-bit blocks).
3. Initialize constants: delta to 0x9e3779b9 and sum to 0.
4. Perform 32 encryption rounds.

Encryption Round (Repeat 32 Times): 5. Increase sum by delta.

6. Update v_0 as follows:

- Add $((v_1 \ll 4) + k_0)$ to v_1 .
- XOR the result with $(v_1 + \text{sum})$.
- XOR the result with $((v_1 \gg 5) + k_1)$.

7. Update v_1 as follows:

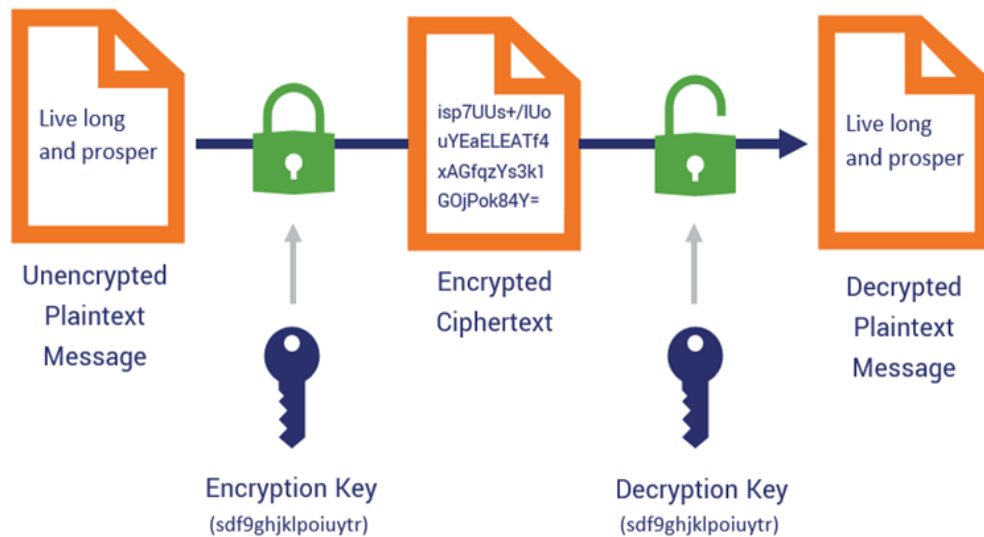
- Add $((v_0 \ll 4) + k_2)$ to v_0 .
- XOR the result with $(v_0 + \text{sum})$.
- XOR the result with $((v_0 \gg 5) + k_3)$.
- Stop:

8. The encrypted data is now in $v[2]$.

Conclusions:

We present a simple algorithm which can be translated into a number of different languages and assembly languages very easily. It is short enough to be programmed from memory or a copy. It is hoped it is safe because of the number of cycles in the encoding and length of key. It uses a sequence of word operations rather than wasting the power of a computer by doing byte or 4 bit operations.

How Symmetric Encryption Works



References:

Original TEA Paper:

- Title: "A Tiny Encryption Algorithm"
- Authors: David Wheeler and Roger Needham
- Year: 1994
- Link: [Google Scholars](#)

Efficient Hardware Implementations of TEA:

- **Title:** "Efficient Hardware Implementations of the Tiny Encryption Algorithm"
- **Authors:** Lina He, Wenjie Lu, and Weiqiang Liu
- **Year:** 2014
- **Link:** Published in IEEE Xplore or similar research platforms.