References :

https://faculty.e-ce.uth.gr/cda/docs/FCCM11_SEAL.pdf

https://en.wikipedia.org/wiki/SEAL_(cipher)

https://github.com/jkstpierre/seal

https://faculty.e-ce.uth.gr/cda/docs/FCCM11_SEAL.pdf

https://www.cs.ucdavis.edu/~rogaway/papers/seal.pdf

FINLE RESEARCH

Software Optimized Encryption Algorithm

SEAL is a stream cipher optimised for machines with a 32-bit word size and plenty of RAM with a reported performance of around 4 cycles per byte. SEAL is actually a pseudorandom function family in that it can easily generate arbitrary portions of the keystream without having to start from the beginning. This makes it particularly well suited for applications like encrypting hard drives

سلمان عبدالرحمن القصمه

عبد السلام العنزي

INTRODUCTION

HISTORY OF SAEL

The full name of the cipher described in this paper is SEAL 3.0. An earlier version of this cipher was described in 1993 and denoted SEAL 1.0. Though SEAL 3.0 is the first modification to SEAL 1.0 which the authors have described, a variant known as SEAL 2.0 had already appeared in the literature : it was identical to SEAL 1.0 apart from using NIST's revised Secure Hash Algorithm (SHA-1) instead of the original one (SHA) . While SEAL 3.0 retains that change, the more significant adjustment is responsive to an attack by Handschuh and Gilbert . See Section 5 for further information on their attack and the differences between SEAL 3.0 and SEAL 1.0. In this paper the name SEAL, by itself, always refers to SEAL 3.0.

ENCRYPTING FAST IN SOFTWARW

Encryption must often be performed at high data rates, a requirement usually achieved, when at all, with the help of sup- porting cryptographic hardware. Unfortunately, fast cryptographic hardware is often absent and data confidentiality is sacrificed because the cost of software cryptography is deemed too expensive

HOW WE DESIGNED SEAL?

we have designed SEAL (Software Encryption Algorithm; to be known as SEAL 1.0 should other versions arise). It is intended to be used as a stream cipher, providing strong data confidentiality. On a modern 32-bit processor SEAL can encrypt messages at a rate of about 5 instructions per byte. In comparison, the DES algorithm is some 10-30 times as expensive. Even a Cyclic Redundancy Code (CRC) is more costly.

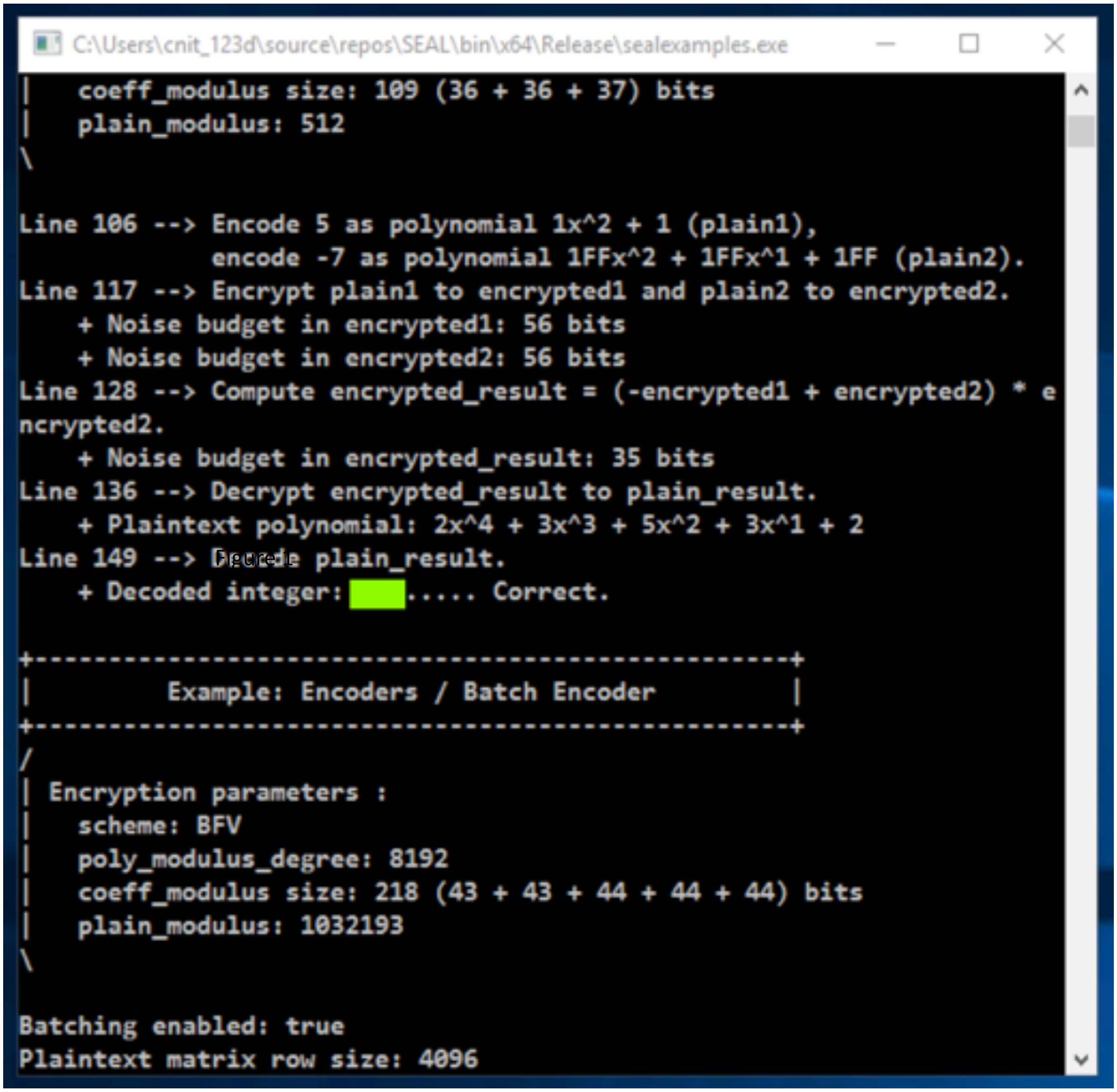

C:\Users\cnit_123d\source\repos\SEAL\bin\x64\Release\sealexamples.exe
| coeff_modulus size: 109 (36 + 36 + 37) bits
| plain_modulus: 512
\
Line 106 --> Encode 5 as polynomial 1x^2 + 1 (plain1),
encode -7 as polynomial 1FFx^2 + 1FFx^1 + 1FF (plain2).
Line 117 --> Encrypt plain1 to encrypted1 and plain2 to encrypted2.
+ Noise budget in encrypted1: 56 bits
+ Noise budget in encrypted2: 56 bits
Line 128 --> Compute encrypted_result = (-encrypted1 + encrypted2) * encrypted2.
+ Noise budget in encrypted_result: 35 bits
Line 136 --> Decrypt encrypted_result to plain_result.
+ Plaintext polynomial: 2x^4 + 3x^3 + 5x^2 + 3x^1 + 2
Line 149 --> plain_result.
+ Decoded integer: ...... Correct.
+--+
| Example: Encoders / Batch Encoder |
+--+
/
| Encryption parameters :
| scheme: BFV
| poly_modulus_degree: 8192
| coeff_modulus size: 218 (43 + 43 + 44 + 44 + 44) bits
| plain_modulus: 1032193
\
Batching enabled: true
Plaintext matrix row size: 4096

Figure 1

CHARACTERISTICS OF THR CIPHER

PREPRCESSING THE KEY

In typical applications requiring fast software cryptography, data encryption is required over the course of a communication session to a remote partner, or over the course of a login session to a particular machine. In either case the key a which protects the session is determined at session setup. Typically this session setup takes at least a few milliseconds and is not a time-critical operation. It is therefore acceptable, in most applications, to spend some number of milliseconds to map the (short) key a to a (less concise) representation of the cryptographic transformation specialized to this key. Our cipher has this characteristic. As such, SEAL is an inappropriate choice for applications which require rapid key setup.

LENGTH-INCREASING FUNCTION

The function SEAL is a type of cryptographic object called a *pseudorandom function family* (PRF). Such objects were first defined in [3]. SEAL is a *length-increasing* PRF: under control of a 160-bit key a, SEAL maps a 32-bit string n to an L-bit string SEAL(a, n, L). The number L can be made as large or as small as is needed for a target application, but output lengths ranging from a few bytes to a few thousand bytes are anticipated. An arbi- trary length key a' can be used as the key for SEAL simply by selecting a = SHA-1(a').

TARGET PLATFORMS

Execution vehicles that should run the algorithm well include the Intel386™/Intel486™/Pentium™ processors, and contemporary 32-bit RISC machines. Because of the particular challenges involved in having a cipher run well on the 386/486/Pentium, and because of the pervasiveness of this processor family, we have optimized our cipher with the characteristics of this processor family particularly in mind. By doing well on these difficult-to-optimize-for vehicles we expect to do well on any modern 32-bit processor

- **Confidentiality**
- **Authentication**
- **Data Integrity**

SEAL optimized

- cost on a 32-bit processor is about 5 elementary machine instructions per byte of text
- with a reported performance of around 4 cycles per byte

SEAL KEY

- The encryption standard relies on a pseudorandom family that uses a **length-increasing function** and a 160-bit key to map the 32-bit string to a string of any length.

FUNCTION SEAL

function $\mathsf{WEAK}(a, n)$
!Unexpected End of Formula

$$y \leftarrow \lambda;$$

$$\mathsf{Initialize}_a(n, 0,\ A, B, C, D,\ \cdots);$$

for $i \leftarrow 1$ to 64 do

$$P \leftarrow A \,\&\, \mathtt{0x1ff};\ B \leftarrow B \oplus T[P];\ A \leftarrow A \rangle\rangle\rangle 9;\ B \leftarrow B \oplus A;$$
$$P \leftarrow B \,\&\, \mathtt{0x1ff};\ C \leftarrow C \oplus T[P];\ B \leftarrow B \rangle\rangle\rangle 9;\ C \leftarrow C \oplus B;$$
$$P \leftarrow C \,\&\, \mathtt{0x1ff};\ D \leftarrow D \oplus T[P];\ C \leftarrow C \rangle\rangle\rangle 9;\ D \leftarrow D \oplus C;$$
$$P \leftarrow D \,\&\, \mathtt{0x1ff};\ A \leftarrow A \oplus T[P];\ D \leftarrow D \rangle\rangle\rangle 9;\ A \leftarrow A \oplus D;$$
$$P \leftarrow A \,\&\, \mathtt{0x1ff};\ B \leftarrow B \oplus T[P];\ A \leftarrow A \rangle\rangle\rangle 9;$$
$$P \leftarrow B \,\&\, \mathtt{0x1ff};\ C \leftarrow C \oplus T[P];\ B \leftarrow B \rangle\rangle\rangle 9;$$
$$P \leftarrow C \,\&\, \mathtt{0x1ff};\ D \leftarrow D \oplus T[P];\ C \leftarrow C \rangle\rangle\rangle 9;$$
$$P \leftarrow D \,\&\, \mathtt{0x1ff};\ A \leftarrow A \oplus T[P];\ D \leftarrow D \rangle\rangle\rangle 9;$$
$$y \leftarrow y \ \|\ B \oplus S[4i-4] \ \|\ C \oplus S[4i-3] \ \|\ D \oplus S[4i-2] \ \|\ A \oplus S[4i-1];$$

return y;

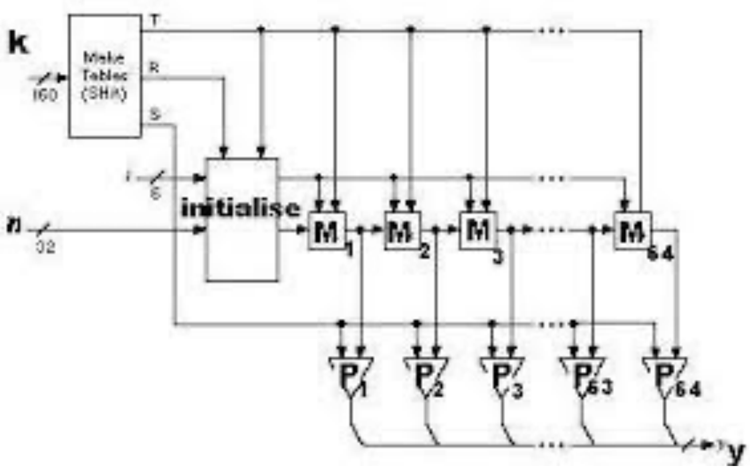
The cipher WEAK, attacks on which are given in the text. Under the control of *a*-derived tables *T* , *R*, and *S* (computed exactly as with SEAL) this cipher maps 32-bit position index *n* to 256-word string WEAK(*a*, *n*).

Can we broke SEAL ?

We present an attack on the SEAL Pseudorandom Function Family that is able to efficiently distinguish it from a truly random function with 2^{43} bytes output. While this is not a practical attack on any use of SEAL, it does demonstrate that SEAL does not achieve its design goals

DESCRIPTION OF SEAL ALGORITHM WITH DIAGRAMS

Initialization is dependent on *T* and *R* tables.
Initialization of *A*, *B*, *C*, *D* , *nl* , *n*2, *n*3, *n*4 from *n*.



The block diagram of SEAL hardware implementation.