Chapter 7
Authenticated Encryption
Primitives

7.1 Introduction

Stream ciphers are useful where the data must be processed in real-time or where data comes in quantities of unknown length and padding or buffering must be avoided. The distinction between block and stream ciphers is not always clear, as block ciphers provide stream cipher properties in certain modes of operation, e.g. cipher feedback (CFB) mode, output feedback (OFB) mode and counter (CTR) mode. Usually stream ciphers employ a symmetric key for generating a pseudorandom bit stream (or keystream), which is then combined with the plaintext, typically using an XOR operation. Therefore basic stream ciphers only provide confidentiality without integrity protection. In general an additional cryptographic operation is needed to guarantee the integrity of a message.

The protection of a message typically requires the protection of both confidentiality and authenticity. There are two main approaches to authenticate and encrypt a message. One approach is to treat the encryption and authentication separately. That is, a plaintext is encrypted with a block cipher or stream cipher, and a separate MAC algorithm (example: keyed hash function) is used to authenticate the ciphertext. The other approach is to find an encryption scheme that can accomplish both message confidentiality and authenticity in a single cryptographic primitive with the focus
to achieve high throughput and minimal overhead. The scheme that delivers both confidentiality and authenticity in a single cryptographic primitive may be more efficient than using separate confidentiality and authenticity techniques. It is more difficult to design a good authenticated cipher than a usual encryption cipher as the security of authenticated ciphers depends on both encryption and authentication. An attacker possesses extra information such as authentication tag other than plaintext/ciphertext and has more choices in the execution of attacks. A new competition known as CAESAR has been recently called for submissions of authenticated ciphers. This competition follows a long tradition of focused competitions in symmetric key cryptography like NISSIE and ECRYPT projects. The rest of this chapter is organized as follows: Section 7.2 describes different authenticated encryption primitives, section 7.3 proposes an authenticated encryption cipher and section 7.4 presents its security evaluation. Finally section 7.5 concludes the chapter.

7.2 Authenticated Encryption Primitives

The different ways to design authenticated encryption (AE) schemes are identified as generic compositions, single-pass modes, two pass combined modes and authenticated encryption using stream ciphers. Generic composition [11] [113] is a conventional way of achieving both authenticity and confidentiality. They are relatively slow, because of their two pass constructions, which means that every block of message is processed twice, once by the encryption algorithm and then by the authentication algorithm with the use of two independent keys. An alternative to the generic composition are the dedicated block cipher modes which are divided into single pass modes and two pass modes. Single pass modes provide authenticated encryption immediately after one-time processing of a message, which reduces the computational cost to about half that of a two-pass scheme. Some examples of such schemes include IAPM [114], OCB [8] and XCBC [9]. As most of these single-pass modes have been protected by patents, two pass combined modes have appeared as an alternative. The two pass combined modes process data in two steps like generic compositions, and differ from them by using a single key for both encryption and authentication. Some of
the examples for this type are the CCM (Counter with CBC-MAC) [115], EAX [116], CWC [117] and GCM (Galois/Counter mode) [10]. Both the schemes CCM and EAX are based on Authenticated Encryption with Associated Data (AEAD) like the packet header in the TCP/IP protocol. But unlike CCM scheme, EAX is online for both the plaintext and the associated data, which means that one can process streaming data in real time. GCM is a parallelizable two-pass scheme based on multiplication over the finite field. Examples of stream ciphers with AEAD mechanisms are Helix [118], SOBER-128 [119], AEGIS [91] etc. Grain 128a [120] is a stream cipher based single pass AE producing an authentication tag of size 32 bits without AEAD.

Deriving motivation from the failures in designing an open efficient single pass authenticated encryption algorithm, a group of European Cryptology team initiated a competition known as CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) in 2013. The purpose of CAESAR is to evaluate the state of the art in authenticated encryption and gather community input regarding desired future directions.

**Rotation Symmetric Substitution Boxes**

The substitution boxes (S-boxes) are the fundamental building blocks and play a vital role in the security of almost all modern symmetric ciphers. So they have to be designed or selected carefully to make the cipher resistant to all kinds of cryptanalytic attacks. They are used as one the major nonlinear components in a cipher. An S-box also known as vectorial Boolean function is represented by the vector $S_B = (f_0, f_1, ..., f_{m-1})$ where each $f_i$, $0 \leq i \leq m - 1$ is a Boolean function from $\mathbb{F}_2^n$ into $\mathbb{F}_2$ which are known as the component functions of the S-box. An S-box of order $n \times m$ is also represented as a lookup table with $2^m$ words of $m$ bits each. Some of the basic criterion for the selection of an S-box that is widely accepted for cryptographic purposes are chosen by a tradeoff with the properties like high nonlinearity, Strict Avalanche Criterion (SAC), balancedness (or bijective), completeness, low differential uniformity, high algebraic immunity and robustness. In lightweight cryptography the choice of $4 \times 4$ S-boxes are
hardware driven rather than $8 \times 8$ S-boxes. However, to improve on this we have selected and worked with an $8 \times 8$ rotation symmetric nonlinear S-box [121] in the proposed design, as they are cryptographically stronger than $4 \times 4$ S-boxes and reduces the amount of space for table lookups when implemented in hardware. Rotation symmetric functions are the class of functions which are invariant under the cyclic rotation operation of inputs which results in partitions or equivalence classes of inputs called orbits where an orbit comprises of all cyclic rotations of an input.

**Definition 7.2.1 (Cyclic Rotation).** Let $x_i \in \mathbb{F}_2$, for any $1 \leq i \leq n$ and $0 \leq k \leq n - 1$ we define

$$
\rho_n^k(x_i) = \begin{cases} 
    x_{i+k} & \text{if } i + k < n \\
    x_{i+k-n} & \text{if } i + k \geq n
\end{cases}
$$

Let $x = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^n$. Then we can extend the definition of $\rho_n^k$ on tuples as follows:

$$
\rho_n^k(x) = (\rho_n^k(x_0), \rho_n^k(x_1), \ldots, \rho_n^k(x_{n-1}))
$$

**Definition 7.2.2 (Rotation Symmetric Boolean Function [121]).** A Boolean function $f(x_0, x_1, \ldots, x_{n-1})$ is called rotation symmetric (Rots) if for each input $(x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^n$

$$
f(\rho_n^1(x_0, x_1, \ldots, x_{n-1})) = f(x_0, x_1, \ldots, x_{n-1})
$$

**Definition 7.2.3 (Rotation Symmetric S-box).** An S-box $S_B$ is rotation symmetric if

$$
\rho_n^1(S_B(x_0, x_1, \ldots, x_{n-1})) = S_B(\rho_n^1(x_0, x_1, \ldots, x_{n-1}))
$$

for all $(x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^n$.

The proposed $8 \times 8$ bit S-box is given in table 7.1. A representative element from each of the 36 orbits of the S-box is chosen and their corresponding output from the S-box $S_B$ is given in the table 7.1. The output corresponding to an arbitrary input $(x_0, x_1, \ldots, x_7) \in \mathbb{F}_2^8$ of the S-box $S_B$ is worked out as $S_B(x) = \rho_8^k(S_B(r))$ for some fixed $k$, $1 \leq k \leq 8$, such that
\( \rho^k(x) = r \), where \( r \) is one of the representatives of the 36 orbits of the S-box given in input column of table 7.1. This helps us to greatly reduce the 256 bytes S-box lookup table to 72 bytes reduced lookup table by introducing a negligibly small computation as defined above.

### Table 7.1: Reduced Truth Table of S-box

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[1, 1, 1, 1, 1, 1, 1, 1]</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 1]</td>
<td>[1, 0, 1, 0, 1, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 0, 0, 1, 0, 0, 1, 1]</td>
<td>[0, 1, 0, 0, 0, 1, 0, 1]</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 1, 0, 0, 1]</td>
<td>[0, 0, 1, 0, 0, 1, 0, 1]</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 1, 0]</td>
<td>[0, 0, 1, 0, 1, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 1]</td>
<td>[0, 0, 1, 0, 0, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 1, 1]</td>
<td>[0, 1, 0, 0, 0, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 0, 1, 0, 0, 0, 1, 0]</td>
<td>[0, 0, 1, 0, 1, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 0, 1, 0, 0, 0, 0, 1]</td>
<td>[0, 1, 0, 0, 0, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 1, 0, 0, 0, 0, 0, 1]</td>
<td>[0, 1, 0, 0, 1, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 1, 0, 0, 0, 0, 1, 1]</td>
<td>[0, 0, 1, 0, 0, 1, 0, 1]</td>
</tr>
<tr>
<td>[0, 0, 1, 0, 0, 0, 0, 0]</td>
<td>[0, 1, 0, 0, 0, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 1, 0, 0, 0, 0, 1, 0]</td>
<td>[0, 1, 0, 0, 1, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 0, 1, 0, 0, 0, 0, 0]</td>
<td>[0, 0, 0, 0, 0, 1, 0, 1]</td>
</tr>
<tr>
<td>[0, 1, 0, 0, 0, 0, 1, 1]</td>
<td>[0, 0, 0, 0, 1, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 0, 1, 0, 0, 0, 0, 0]</td>
<td>[0, 1, 0, 0, 0, 0, 1, 0]</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[0, 1, 0, 0, 0, 0, 0, 0]</td>
</tr>
</tbody>
</table>

The selected 8 bit rotation symmetric S-box has the following properties: nonlinearity 102, bijective, complete, robustness 0.96875, SAC (refer table 7.2), differential uniformity 8. Each of the eight component Boolean functions associated with the S-box has algebraic immunity 4 and nonlinearity 106. These properties form a measure of security against linear, differential and algebraic cryptanalysis. With respect to the definition given in section 5.2 of chapter 5, we define the table 7.2.

### Table 7.2: SAC of the rotation symmetric S-box

<table>
<thead>
<tr>
<th>Input</th>
<th>Error Vector</th>
<th>( f_0 )</th>
<th>( f_1 )</th>
<th>( f_2 )</th>
<th>( f_3 )</th>
<th>( f_4 )</th>
<th>( f_5 )</th>
<th>( f_6 )</th>
<th>( f_7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[0, 1, 0, 0, 0, 0, 0, 0]</td>
<td>122</td>
<td>120</td>
<td>132</td>
<td>144</td>
<td>136</td>
<td>120</td>
<td>124</td>
<td>140</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[0, 0, 1, 0, 0, 0, 0, 0]</td>
<td>140</td>
<td>112</td>
<td>120</td>
<td>132</td>
<td>144</td>
<td>136</td>
<td>120</td>
<td>124</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[0, 0, 0, 1, 0, 0, 0, 0]</td>
<td>124</td>
<td>140</td>
<td>112</td>
<td>120</td>
<td>132</td>
<td>144</td>
<td>136</td>
<td>120</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[0, 0, 0, 0, 1, 0, 0, 0]</td>
<td>120</td>
<td>124</td>
<td>140</td>
<td>112</td>
<td>120</td>
<td>132</td>
<td>144</td>
<td>136</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[0, 0, 0, 0, 0, 1, 0, 0]</td>
<td>136</td>
<td>120</td>
<td>124</td>
<td>140</td>
<td>112</td>
<td>120</td>
<td>132</td>
<td>144</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[0, 0, 0, 0, 0, 0, 1, 0]</td>
<td>144</td>
<td>136</td>
<td>120</td>
<td>124</td>
<td>140</td>
<td>112</td>
<td>120</td>
<td>132</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[0, 0, 0, 0, 0, 0, 0, 1]</td>
<td>132</td>
<td>144</td>
<td>136</td>
<td>120</td>
<td>124</td>
<td>140</td>
<td>112</td>
<td>120</td>
</tr>
<tr>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>[0, 0, 0, 0, 0, 0, 0, 0]</td>
<td>120</td>
<td>132</td>
<td>144</td>
<td>136</td>
<td>120</td>
<td>124</td>
<td>140</td>
<td>128</td>
</tr>
</tbody>
</table>
7.3 An Overview of the Proposed Cipher

The proposed algorithm takes as input a key of size 128 bits, an initial vector (IV) of size 128-bit (16-byte). The key is secret, and the initial vector is typically public. The input key is represented as a sequence of bytes \( k_0, k_1, ..., k_{15} \) and the initial vector represented as a sequence of bytes \( v_0, v_1, ..., v_{15} \). At the end, the encryption process produces a ciphertext message and a tag that provides message authentication. The decryption function takes the key \( K \), initial vector \( IV \), ciphertext \( C \), and tag \( T \), and produces either the plaintext message \( P \) or an error if the authentication failed. The operations used in the algorithm are exclusive or, addition modulo 256 and binary AND. The size of the internal state is about 280 bits, which consists of 35 bytes, represented as a sequence \( S_0, S_1, S_2, ..., S_{34} \). The internal state is divided into two parts of 144 bits \( S_0, S_1, S_2, ..., S_{17} \) and 136 bits \( S_{18}, S_{19}, ..., S_{34} \) which are updated independently using the output of the nonlinear filter NLF. The cipher also contains an accumulator which is a shift register consisting of 16 bytes of memory denoted by \( A_0, A_1, ..., A_{15} \). This accumulator is updated regularly using the internal state of the cipher and one byte output of the NLF. The updated values in the accumulator are used for generating the 128-bit authentication tag for a message. In the proposed scheme, both the ciphertext and the authentication tag (or MAC) are generated simultaneously. Here the plaintext as a whole is being authenticated.

7.3.1 Nonlinear Filter Function

The filter function takes \((\ell_1, \ell_2, \ell_3, \ell_4, \ell_5)\) as input and gives \((\ell'_1, \ell'_2, \ell'_3, \ell'_4, \ell'_5)\) as output where \(\ell_i, \ell'_i \) for \(i = 1, 2, 3, 4, 5\) are bytes. Let \(S_R\) denote the \(8 \times 8\) substitution box used in the cipher. The NLF function processes the input data as follows:
\( \partial \leftarrow \ell_5; \quad \partial' \leftarrow \ell_4 \oplus \partial; \quad \partial'' \leftarrow S_B(\ell_3 \oplus \ell_4) \oplus \partial' \)

\( \partial'' \leftarrow S_B(\ell_2 \oplus \ell_3) \oplus \ell_3 \oplus \ell_4 \oplus \partial''; \quad \ell'_4 \leftarrow (\ell_1 \oplus \partial) \oplus S_B(\partial' \oplus \partial'' \oplus S_B(\partial'')) \)

\( \ell'_2 \leftarrow \left[ (\ell_1 \oplus \ell_2 \oplus S_B(\ell_1 \oplus \partial)) \oplus ((\ell_1 \oplus \partial) \oplus S_B(\partial')) \right] \oplus S_B(\partial'' \oplus (\ell_1 \oplus \partial) \oplus S_B(\partial')) \)

\( \ell'_3 \leftarrow \left[ (\ell_2 \oplus \ell_3) \oplus S_B(\ell_1 \oplus \ell_2) \right] \oplus \left[ S_B(\partial \oplus \ell_1) \oplus \ell_1 \oplus \ell_2 \right] \oplus S_B((\ell_1 \oplus \ell_2) \oplus S_B(\partial \oplus \ell_1)) \oplus (S_B(\partial') \oplus (\partial \oplus \ell_1)) \)

\( \ell'_4 \leftarrow [\ell_4 \oplus \ell_3 \oplus S_B(\ell_2 \oplus \ell_3)] \oplus [((\ell_2 \oplus \ell_3) \oplus S_B(\ell_1 \oplus \ell_2)) \oplus (\ell_1 \oplus \ell_2 \oplus S_B(\partial \oplus \ell_1))] \)

\( \ell'_5 \leftarrow (\ell_5 \oplus \ell_4) \oplus S_B(\ell_3 \oplus \ell_4) \oplus (\ell_4 \oplus \ell_4) \oplus S_B(\ell_2 \oplus \ell_3) \)

\( \oplus S_B((\ell_3 \oplus \ell_4 \oplus S_B(\ell_2 \oplus \ell_3)) \oplus ((\ell_2 \oplus \ell_3) \oplus S_B(\ell_1 \oplus \ell_2))) \)

Figure 7.1: Block diagram of NLF
The NLF is chosen to be complete, which means that each output byte is influenced by all of the input bytes. The only nonlinear component in the filter is the S-box.

### 7.3.2 Key/IV Initialization Phase

Before the keystream is generated the cipher must be initialized with the key and the IV. The initialization of the key and IV are done as follows,

\[
S_i \leftarrow k_i \oplus v_i, \quad 0 \leq i \leq 15; \quad S_{16} \leftarrow k_0; \quad S_{17} \leftarrow (1, 1, 0, 1, 0, 0, 1, 0)
\]

\[
S_i \leftarrow k_i \oplus v_i, \quad 18 \leq i \leq 33; \quad S_{34} \leftarrow (1, 0, 1, 0, 1, 0, 0, 1); \quad A_i \leftarrow 0, \quad 0 \leq i \leq 15
\]

\[
(\ell_1, \ell_2, \ell_3, \ell_4, \ell_5) \leftarrow (S_0, S_{11}, S_{13}, S_{18}, S_{31});
\]

Then, the cipher is clocked 140 times without producing any keystream, instead the internal state registers and the accumulator is updated with the output bytes of the filter NLF.

**Pseudo code for state updation process**

\[
\{ \\
(\ell_1, \ell_2, \ell_3, \ell_4, \ell_5) \leftarrow (S_0, S_{11}, S_{13}, S_{18}, S_{31}) \\
(\ell'_1, \ell'_2, \ell'_3, \ell'_4, \ell'_5) \leftarrow NLF(\ell_1, \ell_2, \ell_3, \ell_4, \ell_5) \\
\delta_1 = \ell'_1 \\
\delta_2 = \ell'_2 \oplus \ell'_3 \\
\delta_3 = \ell'_4 \oplus \ell'_5 \\
(S_0, S_1, ..., S_{17}) \leftarrow (S_1, S_2, ..., S_{17}, \delta_2) \\
(S_{18}, S_{19}, ..., S_{34}) \leftarrow (S_{19}, S_{20}, ..., S_{34}, \delta_3)
\}
\]
\[(A_0, A_1, ..., A_{15}) \leftarrow (A_1, A_2, ..., A_{15}, \delta_1)\]
\}
repeat 140 times

Figure 7.2: Block diagram for Key/IV initialization

### 7.3.3 Keystream Generation/Encryption

The proposed cipher is a byte oriented stream cipher producing 1 byte/clock in its basic implementation. However, the rate can be increased up to 32 bits/clock if some additional hardware is used. In this phase the algorithm produces the keystream which is then xored with the plaintext to get the ciphertext.

**Pseudo code**

\[
\begin{align*}
(\ell_1, \ell_2, \ell_3, \ell_4, \ell_5) &\leftarrow (S_0, S_{11}, S_{13}, S_{18}, S_{31}) \\
(\ell'_1, \ell'_2, \ell'_3, \ell'_4, \ell'_5) &\leftarrow NLF(\ell_1, \ell_2, \ell_3, \ell_4, \ell_5) \\
\delta_1 &= \ell'_1 \\
\delta_2 &= \ell'_2 \oplus \ell'_3 \\
\delta_3 &= \ell'_4 \oplus \ell'_5 \\
(S_0, S_1, ..., S_{17}) &\leftarrow (S_1, S_2, ..., S_{17}, \delta_2)
\end{align*}
\]
\[(S_{18}, S_{19}, \ldots, S_{34}) \leftarrow (S_{19}, S_{20}, \ldots, S_{34}, \delta_3)\]

Update \((A_0, A_1, \ldots, A_{15})\) (refer section 7.3.4)

\[Z \leftarrow \delta_1 \oplus \delta_2 \oplus \delta_3 \text{ (keystream byte)}\]

\[C \leftarrow P \oplus Z \text{ (ciphertext)}\]

Figure 7.3: Block diagram for keystream generation process

### 7.3.4 Authentication Phase

At the end of encrypting the last bit of a message, a 128-bit MAC is produced to authenticate the message. The MAC generation and encryption are done simultaneously. As with most of the stream ciphers, the ciphertext is created by xoring the plaintext with the keystream. The accumulator \((A_0, A_1, \ldots, A_{15})\) is updated with the output of the NLF as follows.

**Update** \((A_0, A_1, \ldots, A_{15})\)

\[
\{ \\
(A_0, A_1, \ldots, A_{15}) \leftarrow (A_1, A_2, \ldots, A_{15}, \ell'_1) \\
A_i \leftarrow A_i \oplus S_{i+3} \oplus P, \ 0 \leq i \leq 15 \\
\}
\]
7.4 Security Evaluation

7.4.1 Periodicity

Since the 280 bit internal state of the cipher is updated in a nonlinear way, finding the period of the cipher is assumed to be difficult. During initialization phase, the cipher is clocked 140 times in a nonlinear way making the cipher to behave as a random function. Hence the periodicity of the cipher for a given Key/IV pair is considered to be equiprobable. Assuming a maximum periodicity of $2^{280}$ for the cipher, the probability for a given key/IV pair to cause a periodicity smaller than $2^{100}$ is $2^{-180}$, which is negligibly small.

7.4.2 Algebraic Attack

The algebraic degree of the output keystream bits when expressed as a function of the nonlinear state bits are large and varies with time. The choice of the S-box is such that it has a high nonlinearity equal to 102 and the component functions are having an algebraic immunity equal to 4 and also the state bits are nonlinearly updated which makes it is so difficult for solving the 280 bit internal state of the keystream generator using algebraic attacks.

7.4.3 Diffusion

To avoid statistical chosen IV attacks, any stream cipher has to pass the diffusion criterion [4]. The proposed cipher satisfies the diffusion property due to the choice of the number of the rounds during initialization and because of the filter function and the S-box which has been selected to provide good SAC property as given in table 7.2. Hence the proposed
algorithm is found to be good in diffusion.

### 7.4.4 Time-Memory-Data Tradeoff Attack

A generic attack that can be applied to a large class of cryptographic primitives, and on stream ciphers in particular, is the time-memory-data tradeoff attack. If the size of the internal state of a cipher is \( n \) then the expected time complexity of this attack is at least \( O(2^n) \) [122]. Hence for the proposed cipher, the 280 bit internal state gives a time complexity of at least \( O(2^{140}) \), which exceeds the complexity of the brute force attack.

### 7.4.5 Cube Attack

Cube attack is effective against almost any cipher where at least one of its output in polynomial representation has low degree. For cube attack we need only a black-box access to the cipher, whereby we can assign various values to the IV bits and obtain the output bits. The cycling of the state of our cipher for at least 72 times is sufficient for each state bit to depend on each key and IV bit in a nonlinear way. We fixed the number of clockings of the state in the initialization phase of the cipher to be 140, which we assume to be sufficient to protect the cipher against both cube attack and resynchronization attacks. The table 7.3 shows that for the randomly chosen cubes, there does not exist any maxterms. The reason for this could be the choice of a nonlinear filter function NLF and the choice of 140 rounds of the initialization phase, which could force the degree of the output polynomial to be very high. The IV variables other than the cube variables are set to zero during the offline process of the attack.

### 7.4.6 Adaptive Chosen Ciphertext Attack

In order to prevent adaptive chosen ciphertext attack, it is necessary to use an encryption scheme that limits ciphertext malleability. Chosen ciphertext attacks, may be adaptive or non-adaptive. In a non-adaptive attack, the attacker chooses the ciphertext or ciphertexts to decrypt in advance, and does not use the resulting plaintexts to inform their choice for more ciphertexts. In an adaptive chosen-ciphertext attack, the attacker makes
Table 7.3: Cube attack for random cubes

<table>
<thead>
<tr>
<th>Sl.No.</th>
<th>Cube</th>
<th>Output bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>{0,1,4,5,6,7,8,11,12,13,15,16}</td>
<td>1 to 200</td>
</tr>
<tr>
<td>2.</td>
<td>{0,11,17,25,26,32,33,34,39,43,45,53}</td>
<td>1 to 200</td>
</tr>
<tr>
<td>3.</td>
<td>{0,1,3,5,6,7,8,11,12,13,16}</td>
<td>1 to 200</td>
</tr>
<tr>
<td>4.</td>
<td>{0,1,2,4,5,6,7,8,11,12,13}</td>
<td>1 to 200</td>
</tr>
<tr>
<td>5.</td>
<td>{21,22,29,41,45,46,47,48,51,62,63}</td>
<td>1 to 200</td>
</tr>
<tr>
<td>6.</td>
<td>{1,2,11,23,24,25,31,43,37,38,41,43,46,48,49,71,72}</td>
<td>1 to 160</td>
</tr>
<tr>
<td>7.</td>
<td>{4,11,23,24,25,31,32,37,38,41,43,46,48,49,50,52}</td>
<td>1 to 160</td>
</tr>
<tr>
<td>8.</td>
<td>{2,13,33,41,45,66,67,68,79}</td>
<td>1 to 200</td>
</tr>
<tr>
<td>9.</td>
<td>{10,20,31,41,42,49,56,57,58,62,65,67,68,70,71,72,74,79}</td>
<td>1 to 160</td>
</tr>
<tr>
<td>10.</td>
<td>{0,1,10,11,22,29,30,31,37,38,44,47,51,60,71,73,74,78}</td>
<td>1 to 160</td>
</tr>
</tbody>
</table>

their ciphertext choices adaptively, that is, depending on the result of prior decryptions. This attack is not feasible in an authenticated encryption scheme, as the choice of a ciphertext requires a corresponding valid authentication tag for decryption and thus restricts ciphertext malleability.

### 7.5 Conclusion

Most applications that require symmetric cryptography actually require both encryption and authentication. To meet this goal, we presented an authenticated encryption algorithm which we believe to be resistant to most of the attacks such as linear, differential, key recovery, and algebraic cryptanalysis. The algorithm is also incorporated with an $8 \times 8$ rotation symmetric cryptographically secure S-box for reducing the amount of space in table lookups, if implemented in hardware. The proposed cipher is a synchronous stream cipher which appears to be suited for applications which requires a flexible software and hardware implementation.