ELmD v2.0

Designers : Nilanjan Datta and Mridul Nandi

Submitters : Nilanjan Datta and Mridul Nandi

nilanjan_isi_jrf@yahoo.com

August 29, 2015

Chapter 1

Mode Specification

ELmD is a Encrypt-Linear mix-Decrypt block cipher mode, designed to provide misuse resistant, fully parallelizable authenticated encryption, secured against blockwise adaptive adversaries. In this section, we provide a complete definition of the ELmD family of authenticated ciphers, which include the complete parameter space defining the family and include a list of all the recommended parameter sets.

1.1 Notation

Assume, $\mathbb{B} = \{0,1\}^{128}$, denotes the size of a complete block. Any string $B \in \{0,1\}^r$, can be represented as $(B[1],\ B[2],\ \ldots,\ B[\ell-1],\ B^*[\ell])$, where $\ell = \lceil \frac{r}{128} \rceil$. Here $B[i]$ is the i^{th} complete block of B and $B^*[\ell]$ is the final block of B which may or may not be complete. We call $r := |B|$, the no. of bits of B and $l := l(B)$, the block-length of B. $(B[i])_s$ denotes the most significant s bits of the block $B[i]$. For $0 \leq a \leq b < \ell$ we denote $B[a..b] := (B[a],\ B[a+1], \ldots,\ B[b])$, $B[..b] = B[1..b]$. We use E_K and E_K^{-1} to denote block cipher encryption and decryption respectively.

The underlying field for the construction is $GF(2^{128})$. An element $a \in GF(2^{128})$ can be representated in any of the following ways :

- An integer between 0 to $2^{128} - 1$.
- A 128-bit string : $a_{127} \cdots a_1 a_0 \in GF(2^{128})$ where $a_i \in \{0,1\}$, which is the binary representation of the integer.
- A polynomial : $a(x) = a_{127}x^{127} + \cdots + a_1 x + a_0$ where $a_i \in \{0,1\}$.

For example, we present the following table to show various representations of some field elements 2, 3 and 7 :

The addition and multiplication of two elements $\alpha, \beta \in GF(2^{128})$ is denoted by $\alpha \oplus \beta$ and $\alpha \cdot \beta$ respectively.

Integer Representation	128-bit String Representation	Polynomial Representation
2	$0^{126}10$	x
3	$0^{126}11$	$x+1$
7	$0^{125}111$	x^2+x+1

Table 1.1: Various representations of some field elements

1.2 External Parameters :

In this subsection, we include a list of 5 external parameters and briefly describe the values these parameters can take and the recommended choices.

1.2.1 External Parameter List

In this subsection, we define out external parameters :

1. rd: It denotes the number of AES rounds used in E_K. It is required that, $\mathsf{rd} \geq 6$.

2. t : It denotes the number of blocks after which intermediate tags are generated. Conventionally, $\mathsf{t} = 0$ suggests that the construction doesn't use any intermediate tag. The values t can take, varies for different applications. When the application runs in a limited buffer t can take any integer value from 0 to 127. Everywhere else, it can take any positive integer value less than 2^{32}.

3. $\mathsf{l_t}$: Denotes the length of intermediate tag. It can take any value from 64 to 128.

4. f : It's a boolean value to indicate whether the tag is **fixed length** or not. $\mathsf{f} = 1$ means that the tag size is 128 bit (fixed size) and $\mathsf{f} = 0$ means that the tag size is flexible and the range is from 128 bit to 255 bit to make the tagged ciphertext multiple of 128 bits.

5. $\mathsf{l_{tag}}$: Denotes the tagsize. It can take any value from 1 to 128.

1.2.2 Recommended Parameter Set

The recommended external parameter values are : $\mathsf{rd} \in \{10, 6\}$, $\mathsf{t} \in \{0, 127\}$, $\mathsf{l_t} = 128$, $\mathsf{f} \in \{0, 1\}$, $\mathsf{l_{tag}} = 128$. As the recommended choices varies for the values of rd, t and f only, we have 8 recommended parameter sets and denote these sets by $\mathsf{ELmD}_{(\mathsf{rd},\mathsf{t},\mathsf{f})}$. The priority is given only for the parameter round number rd as this has some security issues. So, priority wise, our primary recommended parameter set is $\mathsf{ELmD}_{(10,\mathsf{t},\mathsf{f})}$ and secondary recommended parameter set is $\mathsf{ELmD}_{(6,\mathsf{t},\mathsf{f})}$ where $(\mathsf{t}, \mathsf{f}) \in \{(0, 0), (127, 0), (0, 1), (127, 1)\}$. The choices of t and f entirely depends on the environment. For low end devices, when we need intermediate tags, we set $\mathsf{t} = 127$. Otherwise we set $\mathsf{t} = 0$. When the application require minimal ciphertext expansion or fixed tag, we set $\mathsf{f} = 1$. Otherwise

to reduce some clock-cycles during decryption, we set $\mathsf{f} = 0$.

1.3 Input and Output Data

To encrypt a message with an associated data, one needs to provide the informations given below.

- An encryption key $K \in \{0,1\}^{128}$, suitable for the block cipher.
- Public message number $\mathsf{pub} \in \{0,1\}^{64}$.
- The Parameter set $\mathsf{param} \in \{0,1\}^{64}$. The most significant 8 bits of param denote the number of AES rounds rd used. Next 16 bits represent the intermediate tag interval t, next 7 bits denotes the intermediate tag length $\mathsf{l_t}$, next bit represents f, to indicate whether we use fixed length tag or flexible length tags are to be used. Next 8-bits denote the tag size $\mathsf{l_{tag}}$. The last 24 bits are kept as optional for future use and currently assigned to the fixed value 0^{24}.
- Associated data $D \in \{0,1\}^*$, with the following restriction of associated data size : $0 \leq |D| \leq 2^{64}$.
- A message (or plaintext) $M \in \{0,1\}^*$, where $1 \leq |M| \leq 2^{64}$. The most significant 64 bits of the first block of the message is priv, is the Private message number.

ELmD authenticated encryption produces the following output data :

- Tagged ciphertext $C \in \{0,1\}^{|M|+l_{tag}}$ where $l_{tag} = 128$ when $\mathsf{f} = 1$ and otherwise (i.e. $\mathsf{f} = 0$) l_{tag} takes the value in between 128 to 255 for which 128 divides $|C|$.
- Intermediate Tags $T \in \{0,1\}^{\mathsf{l_t}h}$ for $\mathsf{t} \neq 0$, where $h = \lfloor \frac{l-1}{\mathsf{t}} \rfloor$, denotes the no. of intermediate tags generated and $l = \lceil \frac{M}{128} \rceil$, denotes the no. of blocks of M. Note that, if $\mathsf{t} = 0$, no intermediate tag is generated i.e. T is empty.

1.4 Mathematical Components

1.4.1 Block Cipher.

We use $\mathsf{AES}^{\mathsf{rd}}$ as our block-cipher where rd denotes the no. of rounds used during the AES encryption or decryption. We usually use $\mathsf{rd} = 10$ for AES encryption and decryption but here we keep it as option, as we may not require that many rounds always. E_K is used in the specification to mean $\mathsf{AES}^{\mathsf{rd}}$.

$\mathsf{AES}^{\mathsf{rd}}$ encryption is done as follows : First KeyExpansion is done where separate 128-bit round key blocks are requires for each of the rd rounds, derived from the cipher key using Rijndael's key schedule. The rd round of encryption is performed, where each round of encryption consists of the following operations : The 128-bit plaintext that is to be encrypted is represented in a 4×4 array of bytes, called the state array, which gets modified in rd rounds of encryption where each round consist of the following operations :

- AddRoundKey : In this stage, the state array is updated by xoring it with the round subkey.
- SubBytes : It is a non-linear substitution step where each byte is replaced with another according to a single fixed lookup table (called S-box).
- ShiftRows : In this phase, the bytes in i^{th} row is cyclically shifted $i-1$ places to the left for all i varies from 1 to 4.
- MixColumns : In this step, an invertible linear transformation is applied to each column. It can be think of as a matrix multiplication over the underlying field $GF(2^8)$.

Note that, $\mathsf{AES}^{\mathsf{rd}}$ is the standard 10 round AES-encryption, where the mix column operation is skipped in the last round. But in general when $\mathsf{rd} < 10$, for $\mathsf{AES}^{\mathsf{rd}}$, we will have the last round mix-column operation - hence have rd full round encryption.

$\mathsf{AES}^{\mathsf{rd}}$ decryption is done as follows : Similarly the rd round of decryption is performed, where each round of encryption consists of Inverse MixColumn, Inverse ShiftRows, Inverse SubBytes and AddRoundKey. The detailed description can be found in [4].

1.4.2 Field Operations.

Here we define the required field operations operations : addition and multiplication. Note that, the underlying field is $GF(2^{128})$.

Field Addition. As our operations are performed in binary field, field addition is equivalent to exclusive-or operation.

Field Multiplication. We will use field multiplication by 2, 3 and 7. We take $p(x) = x^{128} + x^7 + x^2 + x + 1$ [19] as the primitive polynomial.

- Multiplication by 2. It is computationally simple to multiply $a \in \{0,1\}^{128}$ by 2. Suppose $a = a_{127} \cdots a_1 a_0$. Multiplying it by 2 means computing $a(x) \cdot x$ modulo $p(x)$ which can be easily computed as :

$$\begin{aligned} a \cdot 2 &= a \ll 1, \quad \text{if } a_{127} = 0 \\ &= (a \ll 1) \oplus 0^{120}10000111, \quad \text{else} \end{aligned}$$

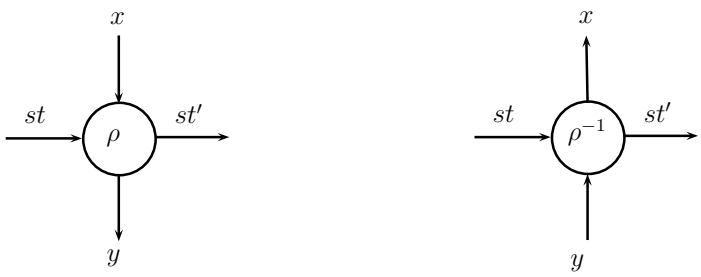

Figure 1.4.1: Linear Mixing Function ρ and ρ^{-1}

- Multiplication by 3. Multiplication by 3 (i.e by the polynomial $x + 1$) is easily computed using the multiplication by 2 :

$$a \cdot 3 = (a \cdot 2) \oplus a$$

- Multiplication by 7. Multiplication by 7 (i.e. by the polynomial $x^2 + x + 1$) is easily computed using the multiplication by 2 and 3 :

$$a \cdot 7 = ((a \cdot 2) \cdot 3) \oplus a$$

1.4.3 Linear Mix Function ρ and ρ^{-1}.

ρ is a linear function that takes two inputs $x \in \mathbb{B}$ and $st \in \mathbb{B}$ and gives $y \in \mathbb{B}$ and $st' \in \mathbb{B}$ in the following way :

$$\begin{aligned} y &= x \oplus 3 \cdot st \\ st' &= x \oplus 2 \cdot st \end{aligned}$$

Now, as y and st' are linear functions of x and st, we can represent x and st' as a linear combination of y and st : $x = y \oplus 3 \cdot st$ and $st = (y \oplus 3 \cdot st) \oplus 2 \cdot st = y \oplus st$. We call this linear function ρ^{-1}. So, ρ^{-1} is a linear function that takes two inputs $y \in \mathbb{B}$ and $st \in \mathbb{B}$ and gives $x \in \mathbb{B}$ and $st' \in \mathbb{B}$ in the following way :

$$\begin{aligned} x &= y \oplus 3 \cdot st \\ st' &= y \oplus st \end{aligned}$$

1.5 ELmD Authenticated Encryption Function

ELmD authenticated encryption takes an associated data $D \in \{0,1\}^*$, a messages $M \in \{0,1\}^*$, a non-negetive integer t and generates a tagged-ciphertext (including the intermediate tags) $(C, T) \in \{0,1\}^{|M|+l_{tag}} \times \{0,1\}^{l_t h}$ in two steps, as described below. We define $L := E_K(0)$ when $E_K = \mathsf{AES}^{10}$ and $L := E_K(E_K(0))$ when $E_K = \mathsf{AES}^{6}$. We use L to generate masks, during the tagged ciphertext generation.

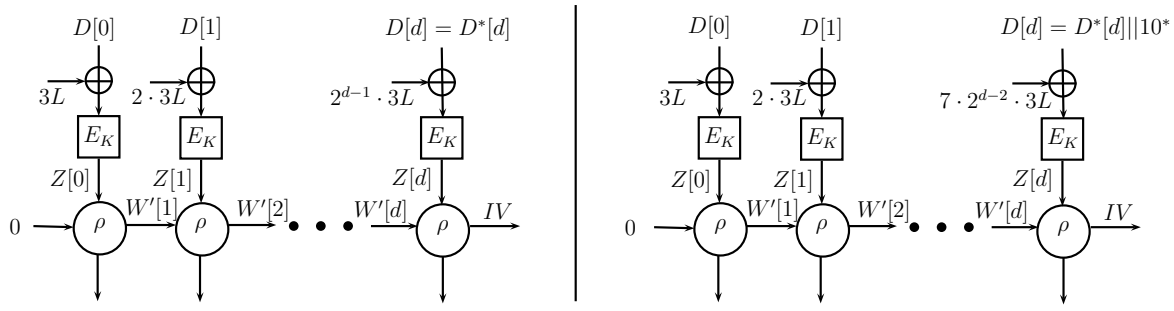

Figure 1.5.1: Processing of Associated data in ELmD Authenticated Encryption : For complete final data block (in the left) and incomplete final data block (in the right)

1.5.1 Initial Value Generation.

Suppose we have a public message number pub, the parameter set param and an associated data $D = (D[1], D[2], \cdots, D[d-1], D^*[d])$. Before the main processing, we first apply the following padding to make the nonce and the final associated data block complete:

$$D[d] = \begin{cases} D^*[d] \;||\; 10^* & \text{if } |D^*[d]| \neq 128 \\ D^*[d] & \text{else} \end{cases}$$

We assign, $W'[0] = 0$ and $D[0] = \mathsf{pub} \;||\; \mathsf{param}$. Now, the processing of the $D[0]$, called the initial block and $D[..d]$ to generate the IV, is done as shown below.

$$\begin{aligned} DD[i] &= D[i] \oplus 3 \cdot 2^i \cdot L \quad \text{for } i = 0 \text{ to } d-1 \\ DD[d] &= \begin{cases} D[d] \oplus 3 \cdot 2^d \cdot L & \text{if } |D^*[d]| = 128 \\ D[d] \oplus 3 \cdot 7 \cdot 2^{d-1} \cdot L & \text{else} \end{cases} \\ Z[i] &= E_K(DD[i]) \quad \text{for } i = 0 \text{ to } d \\ (Y'[i], W'[i+1]) &= \rho(Z[i], W'[i]) \quad \text{for } i = 0 \text{ to } d \\ IV &= W'[d+1] \end{aligned}$$

Note that, when associated data is empty, $IV = E_K(D[0] \oplus 3 \cdot L)$.

1.5.2 Tagged Ciphertext Generation.

The tagged ciphertext is generated using the message M and the IV, generated as described above using the nonce N and the associated data D. Suppose $M = (M[1], M[2], \cdots, M[l-1], M^*[l])$. Depending on the completeness of the last block, we first perform the following padding to make the last block

complete :

$$M[l] \quad = \quad \begin{cases} (\oplus_{i=1}^{l-1} M[i]) \oplus (M^*[l] \;||\; 10^*) & \text{if } 0 < |M^*[l]| < 128 \\ (\oplus_{i=1}^{l-1} M[i]) \oplus M^*[l] & \text{else} \end{cases}$$

Now, the tagged ciphertext C along with the intermediate tags T is generated using the following equations :

$$\begin{aligned}
W[0] &= IV \\
M[l+1] &= M[l] \\
MM[i] &= M[i] \oplus 2^{i-1} \cdot L \quad \text{for } i = 1 \text{ to } (l-1) \\
MM[l] &= \begin{cases} M[l] \oplus 2^{l-1} \cdot L & \text{if } |M^*[l]| = 128 \\ M[l] \oplus 7 \cdot 2^{l-2} \cdot L & \text{else} \end{cases} \\
MM[l+1] &= \begin{cases} M[l+1] \oplus 2^{l} \cdot L & \text{if } |M^*[l]| = 128 \\ M[l+1] \oplus 7 \cdot 2^{l-1} \cdot L & \text{else} \end{cases} \\
X[i] &= E_K(MM[i]) \quad \text{for } i = 1 \text{ to } (l+1) \\
(Y[i], W[i]) &= \rho(X[i], W[i-1]) \quad \text{for } i = 1 \text{ to } (l+1) \\
CC[i] &= E_K^{-1}(Y[i]) \quad \text{for } i = 1 \text{ to } l \\
C[i] &= CC[i] \oplus 3^2 \cdot 2^{i-1+\lfloor \frac{i-1}{\mathsf{t}} \rfloor} \cdot L \quad \text{for } i = 1 \text{ to } l \\
TT[j] &= E_K^{-1}(W[j.\mathsf{t}]) \quad \text{for } j = 1 \text{ to } h \\
T[j] &= TT[j] \oplus 3^2 \cdot 2^{j\mathsf{t}+j-1} \cdot L \quad \text{for } j = 1 \text{ to } h \\
CC[l+1] &= E_K^{-1}(Y[l+1] \oplus 1) \\
C[l+1] &= CC[l+1] \oplus 3^2 \cdot 2^{l+h} \cdot L
\end{aligned}$$

The algorithm returns tagged ciphertext and intermediate tags (C, T); where

$$\begin{aligned}
C &= \begin{cases} (\; C[1..l], (C[l+1])_{|M^*[l]|} \;) & \text{if } \mathsf{f} = 1 \\ C[1..(l+1)] & \text{else} \end{cases} \\
T &= ((T[1])_{\mathsf{l_t}},\; (T[2])_{\mathsf{l_t}},\; \cdots,\; (T[h])_{\mathsf{l_t}}) \in \{0,1\}^{\mathsf{l_t}.h}
\end{aligned}$$

When $\mathsf{t} = 0$, the term $\lfloor \frac{i-1}{\mathsf{t}} \rfloor$ is defined to be 0. Hence, we have $C[i] = CC[i] \oplus 3^2 \cdot 2^{i-1} \cdot L$, for $\mathsf{t} = 0$.

1.6 ELmD Verified Decryption Function

ELmD verified decryption is the algorithm that takes the public message number pub, parameter set param, an associated data D, a tagged-ciphertext C, intermediate tags $T \in \{0,1\}^{\mathsf{l_t}h}$, a bit $b \in \{0,1\}$ if $\mathsf{f} = 0$ and returns a message M or $\perp$, depending on the verification. For flexible tag length i.e. the case when $\mathsf{f} = 0$, the additional bit b is required to determine whether the last block of the message is complete ($b = 1$) or incomplete ($b = 0$). Let $l + 1 = \lceil \frac{C}{128} \rceil$, denotes

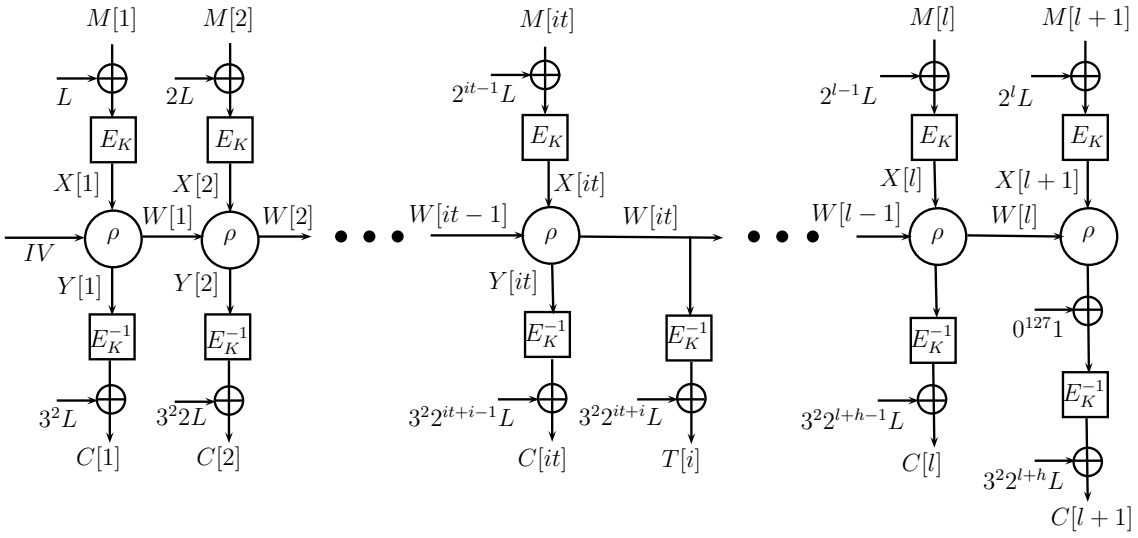

Figure 1.5.2: Construction of ELmD Authenticated Encryption

the no. of blocks of the ciphertext.

The Decryption function is a three step process : First IV is generated using pub, param and D, which is identical to the initial value generation during the authenticated encryption. Then the decryption is performed using the tagged ciphertext and then the verification is performed and if verified, corresponding message is returned. The decryption and the verification procedure is dscribed below.

1.6.1 Decryption.

It first process the associated data exactly similar to the encryption. Then using the inverse online linear function ρ^{-1}, the algorithm computes M.

$$
\begin{aligned}
W[0] &= IV \\
CC[i] &= C[i] \oplus 3^2 \cdot 2^{i-1+\lfloor \frac{i-1}{t} \rfloor} \cdot L \quad \text{for } i = 1 \text{ to } l \\
Y[i] &= E_K(CC[i]) \quad \text{for } i = 1 \text{ to } l \\
(X[i], W[i]) &= \rho^{-1}(Y[i], W[i-1]) \quad \text{for } i = 1 \text{ to } l \\
MM[i] &= E_K^{-1}(Y[i]) \quad \text{for } i = 1 \text{ to } l \\
M[i] &= MM[i] \oplus 2^{i-1} \cdot L \quad \text{for } i = 1 \text{ to } (l-1) \\
M[l] &= \begin{cases} MM[l] \oplus 2^{l-1} \cdot L & \text{if } |C^*[l+1]| = 128 \text{ for } \mathsf{f} = 1 \text{ or } b = 1 \text{ for } \mathsf{f} = 0 \\ M[l] \oplus 7 \cdot 2^{l-2} \cdot L & \text{else} \end{cases} \\
M^*[l] &= \oplus_{i=1}^{l} M[i] \\
M[l+1] &= M^*[l]
\end{aligned}
$$

1.6.2 Verification.

The verification phase is done differently depending on whether the tag size is fixed or flexible.

For fixed sized tags (i.e. when $\mathsf{f} = 1$), the verification is done as described below. First the following computations are performed :

$$\begin{aligned}
TT[j] &= T[j] \oplus 3^2 \cdot 2^{jt+j-1} \cdot L \quad \text{for } j = 1 \text{ to } h \\
W'[j.\mathsf{t}] &= E_K(TT[j]) \quad \text{for } j = 1 \text{ to } h \\
X[l+1] &= E_K(MM[l+1]) \\
(Y[l+1], W[l+1]) &= \rho(X[l+1], W[l]) \\
CC[l+1] &= E_K^{-1}(Y[l+1] \oplus 1) \\
C'[l+1] &= CC[l+1] \oplus 3^2 \cdot 2^{l+h} \cdot L
\end{aligned}$$

The verification succeeds if both of the following two conditions holds :

- Intermediate tags are Verified (if $\mathsf{t} > 0$): $\forall i \leq h$, $W[i.\mathsf{t}] = W'[i.\mathsf{t}]$
- Tagged Ciphertext is Verified : If $|C^*[l+1]| = 128$, then the tagged ciphertext is verified if $C^*[l+1] = C'[l+1]$. Otherwise, the ciphertext is verified if $C^*[l+1] = (C'[l+1])_{|C^*[l+1]|}$ and the last $(128 - |C^*[l+1]|)$ bits of $M^*[l]$ is 10^*.

On the otherhand, for flexible tag size (i.e. when $\mathsf{f} = 0$), we first compute the following :

$$\begin{aligned}
TT[j] &= T[j] \oplus 3^2 \cdot 2^{jt+j-1} \cdot L \quad \text{for } j = 1 \text{ to } h \\
W[j.\mathsf{t}] &= E_K(TT[j]) \quad \text{for } j = 1 \text{ to } h \\
CC[l+1] &= C[l] \oplus 3^2 \cdot 2^{l+h} \cdot L \\
Y[l+1] &= E_K(CC[l+1]) \oplus 1 \\
(X[l+1], W[l+1]) &= \rho^{-1}(Y[l], W[l-1]) \\
MM[l+1] &= E_K^{-1}(X[l+1]) \\
M'[l+1] &= \begin{cases} MM[l+1] \oplus 2^l \cdot L & \text{if } b = 1 \\ MM[l+1] \oplus 7 \cdot 2^{l-1} \cdot L & \text{else} \end{cases}
\end{aligned}$$

Now the verification succeeds if both the following holds :

- Intermediate tags are Verified (if $\mathsf{t} > 0$) : $\forall i \leq h$, $W[i.\mathsf{t}] = W'[i.\mathsf{t}]$.
- Tagged Ciphertext is Verified : $M'[l+1] = M^*[l+1]$.

In both the cases, if i^{th} intermediate tag verification is successful, we realease plaintext $M[..i\mathsf{t}]$. If any of the verification is not successful, we return $\perp$. If all

the intermediate tag verification succeeds, then the final tag is veryfied and M is returned :

$$M \quad = \quad \begin{cases} (M[..(l-1)],\ (M^*[l])_{r'}) & \text{if } b = 0 \text{ for } \mathsf{f} = 0 \text{ or } |C^*[l+1]| < 128 \text{ for } \mathsf{f} = 1 \\ (M[..(l-1)],\ M^*[l]) & \text{else} \end{cases}$$

where r' is defined such that, $M^*[l] = (M^*[l])_{r'}||10^*$.

Chapter 2

Proposed Modification from ELmD v1.0

ELmD v2.0 has following modifications from the version ELmD v1.0 :

- The definition of $M[l]$ and $M[l+1]$ is modified, while processing a message M of length l.
- The secondary round parameter is considered to be $(6, 6)$ instead of $(5, 10)$.
- Definition of L is modified when AES^6 is used as the block-cipher.

Here, we discuss details of the these modifications with proper justification.

2.1 Modification in the Specification.

We propose a small modification in padding in the specification. We define $M[l]$ and $M[l+1]$ as :

$$\begin{aligned} M[l] &= \begin{cases} (M^*[l] \mid\mid 10^*) \oplus (\oplus_{i=1}^{l-1} M[i]) & \text{if } |M^*[l]| \neq 128 \\ M^*[l] \oplus (\oplus_{i=1}^{l-1} M[i]) & \text{else} \end{cases} \\ M[l+1] &= M[l] \end{aligned}$$

Brief Explanation of the Modification. Our previous analysis did not take care processing of incomplete blocks properly. Consider the case of an adversary, makes some encryption queries and tries to forge for an incomplete final block message. Suppose the last ciphertext block has i bits. Now, in order to construst a valid forging, it must satisfy the two consitions: (i) The last i bit ciphertext matches and (ii) The last $(128-i)$ bit of the last plaintext block is 10^*. As only i bit ciphertext matching (this i-bit ciphertext depends on the checksum) is done and the last plaintext block doesn't depend on the checksum (in particular it doesn't depend on any previous message blocks), the entropy of the checksum is

reduced to i bits instead of 128 bit. This entropy loss may cause some problems for security analysis. Now, to gain entropy, we define this new padding rule such that the last padded plaintext block depends on all the message blocks.

2.2 Modification in Round Parameter.

We use only one AES-round parameter rd which denotes the number of rounds used in AES encryption, as well as in AES decryption. We chose $\mathsf{rd} = 10$ and $\mathsf{rd} = 6$ as the primary and secondary recommended parameters respectively. For $\mathsf{rd} = 6$ versions, we compute L as $L := \mathsf{AES}^6(\mathsf{AES}^6(0))$.

Brief Explanation of the Modification. Having equal no. AES rounds in upper as well as lower layer, will ensure full pipelined implementation implying better performance. Moreover having equal no. of rounds will make similar structure for both encryption and decryption. This helps us in minimizing the combined hardware implementation. Details of this will be discussed later in chapter 6. We use 6 round as the recommended parameter set, as 6 round AES in the upper layer provides a good collision resistant hash and the total of 12 $(= 6 + 6)$ round AES in the combined upper-lower layer provides the desired randomness. The detailed argument that 6 round AES is supposed to give the desired security, is discussed in chapter 4.

Chapter 3

Security Goals

$\mathsf{ELmd}_{10,t,f}$ uses full round AES encryption-decryption, and hence promises to provide the online privacy whenever the legitimate key holder can use same nonce to encrypt two different (plaintext, associated data) pairs and full privacy if nonce can be ensured distinct for each invocation. The concatanation of the public message number and private message number is used as nonce. The no. of bits of security for confidentiality is 62.8, as mentioned in the table, when we consider the distinguishing attack. That means, the expected number of queries an adversary needs to distinguish our construction from an online function chosen uniformly at random, is $2^{62.8}$. **Note that, one can not use this distinguishing attack to mount a plaintext or key recovery attack and we believe that our construction provides** 128 **bits of security, against plaintext or key recovery attack**.

The integrity goal of a forger is to make some forward queries and then attempt to forge against the construction for several times. When intermediate tags are used i.e. $\mathsf{t} \neq 0$, if the forger can compute a valid intermediate tag such that the ciphertext upto that is not identical to any of previous ciphertexts then forger succeeds. We claim that $2^{62.4}$ and $2^{62.3}$ are the expected number of online forgery attempts for a successful forgery for $\mathsf{t} = 0$ and $\mathsf{t} = 127$ respectively. The explation of the desired number of bits of security is given in the next chapter.

On the other hand, for $\mathsf{ELmD}_{6,t,f}$ recommended choices, as 6 rounds of AES encryption-decryption is being used, although we believe that the desired security given in the table is achievable, but it may contain some weakness, as full round AES has not been used. We discuss this issue in details, in the next section.

Goal	$\mathsf{ELmD}_{\mathsf{rd},0,\mathsf{f}}$	$\mathsf{ELmD}_{\mathsf{rd},127,\mathsf{f}}$
confidentiality for the plaintext	62.8	62.8
integrity for the plaintext	62.4	62.3
integrity for the associated data	62.4	62.3
integrity for the public message number	62.4	62.3
confidentiality for the private message number	62.8	62.8
integrity for the private message number	62.4	62.3

Table 3.1: Table quantifying, for each of the recommended parameter sets, the intended number of bits of security : Here $(\mathsf{rd}, \mathsf{f}) \in \{(10,0),(10,1),(6,0),(6,1)\}$. **For confidentiality we consider distinguishing advantage and for integrity we consider the forging advantage**.

Chapter 4

Security Analysis

4.1 Confidentiality and Integrity of the recommended choices.

In this section, we provide the security bounds of all the recommended parameter sets. Note that, the parameter f is used for implementation issues and doesn't have any effect from functional point of view. So, the security bounds doesn't depend on the value of f. Let $n = 128$ represents the blocksize.

Now applying the following mentioned theorems and using standard hybrid technique, we obtain the following results :

- Theorem 3.1 : $\mathbf{Adv}^{\text{opriv}}_{\mathsf{ELmD}_{10,0,\mathsf{f}}}(A) \leq \eta(\sigma_{\text{priv}}) + \frac{5\sigma^2_{\text{priv}}}{2^n}$.
- Theorem 3.2 : $\mathbf{Adv}^{\text{opriv}}_{\mathsf{ELmD}_{10,127,\mathsf{f}}}(A) \leq \eta(\sigma_{\text{priv}}) + \frac{91}{18}\frac{\sigma^2_{\text{priv}}}{2^n}$
- Theorem 3.3 : $\mathbf{Adv}^{\text{auth}}_{\mathsf{ELmD}_{10,0,\mathsf{f}}}(A) \leq \eta(\sigma_{\text{auth}}) + \frac{9\sigma^2_{\text{auth}}}{2^n}$.
- Theorem 3.4 : $\mathbf{Adv}^{\text{auth}}_{\mathsf{ELmD}_{10,127,\mathsf{f}}}(A) \leq \eta(\sigma_{\text{auth}}) + \frac{81}{8}\frac{\sigma^2_{\text{auth}}}{2^n}$

Here $\eta(i)$ denotes the maximum AES advantage over all adversaries, making at most i queries. As full rounds of AES is used, we can assume $\eta(i)$ to be negligible.

Now, the above argument doesn't work if we use 6 as the round parameter because for 6 round AES, the maximum AES advantage over all adversaries is high. So, we have to argue in a different way. From the structure of construction, we observe that we need to resist collision in the upper layer encryption and want high randomness in the combined two layer encryption. We use 6 round AES in the upper layer, as 6 round AES is a good collision resistant hash and the total of 12 $(= 6 + 6)$ rounds of AES in the combined upper-lower layer provides the desired randomness. In this context, note that AES-6 has many

key-recovery attacks [2, 5, 6] but all those attacks uses the property that the chosen plaintexts has certain differential characteristic. Here these attacks are not applicable due to the upper layer masking and the randomness of L.

With these arguments in hand, we have the following conjecture: $\mathsf{ELmD}_{6,\mathsf{t},\mathsf{f}}$ has 62.8 bit security for confidentiality for the plaintext and private message number. $\mathsf{ELmD}_{6,0,\mathsf{f}}$ has 62.4 bit security for integrity for the plaintext, associated data, public message number, private message number. $\mathsf{ELmD}_{6,127,\mathsf{f}}$ has 62.3 bit security for integrity for the plaintext, associated data, public message number, private message number.

4.2 Confidentiality of ELmD

We give a particularly strong definition of confidentiality or privacy, one asserting indistinguishability from random strings. Consider an adversary A who has access of one of two types of oracles: a "real" encryption oracle or an "ideal" authenticated encryption oracle. A real authenticated encryption oracle, F_K, takes as input (D, M) and returns $(C, T) = F_K(D, M)$. Whereas an ideal authenticated encryption oracle \$ returns a random string R with $|R| = |M| + 1$ for every fresh pair (D, M). Given an adversary A (w.o.l.g. we assume a **deterministic adversary**) and an authenticated encryption scheme F, we define the (full) **privacy-advantage** of A by the distinguishing advantage of A distinguishing F from \$. More formally,

$$\mathbf{Adv}_F^{\mathrm{priv}}(A) := \mathbf{Adv}_F^{\$}(A) = \mathsf{Pr}_K[A^{F_K} = 1] - \mathsf{Pr}_{\$}[A^{\$} = 1].$$

Similarly, we define online privacy for which the the ideal online authenticated encryption oracle $\$_{ol}$ responses random string keeping the online property. The online privacy advantage of an adversary A against F is defined as $\mathbf{Adv}_F^{\mathrm{opriv}}(A) := \mathbf{Adv}_F^{\$_{ol}}(A)$. In this section, we'll prove online privacy of our construction, which extends to provide full privacy in nonce respecting scenario.

We use the notation ELmD_Π denotes our algorithm ELmD, when the blockcipher encryption is replaced by random permutation Π.

4.2.1 Confidentiality of ELmD_Π when $\mathbf{t} = 0$

Theorem 4.1 *Let A be an adversary which can make q queries at an aggregrate of total σ associated data and message blocks to distinguish ELmD_Π with $\mathsf{t} = 0$, from an online cipher chosen uniformly at random. Let $\sigma_{\mathrm{priv}} = \sigma + q$. The online privacy advantage of the adversary A is given by,*

$$\mathbf{Adv}_{\mathsf{ELmD}_\Pi}^{\mathrm{opriv}}(A) \;\leq\; \frac{5\sigma_{\mathrm{priv}}^2}{2^n}.$$

Proof. The main idea is as follows : Suppose an adversary A tries to distinguish our construction from a PRP, making q queries $M_1, \cdots, M_q$ and get

responces $C_1, \cdots, C_q$. $(C_1, \cdots, C_q)$ is called a view of the adversary with respect to $(M_1, \cdots, M_q)$. We call some of these views as good views and show that for those views, our construction has high interpolation probability i.e. A can not distinguish our construction from a PRP for those views with more than negligeable probability. Moreover we show that the probability of having a view to be good is overwhelming. Hence applying patarin's coefficient H technique, we obtain the result. More formally, the proof follows directly from Patarin's coefficient H technique [17] and following the two Lemmas 4.2.1 and 4.2.4.

More formally, let us fix q associate data with initial block, plaintext pairs $P_1 = (D_1, M_1), \ldots, P_q = (D_q, M_q)$ with $|D_i| = d_i, |M_i| = l_i$, $\sigma = \sum_i d_i + l_i$, $\sigma_{\text{priv}} = \sigma + q$. This q is added to incorporate the checksum blocks. We denote $(P_1, \ldots, P_q)$ by τ_{in}. We assume that all P_i's are distinct. A tagged ciphertext tuple $\tau_{out} = (C_1, \ldots, C_q)$ (also the complete view $\tau = (\tau_{in}, \tau_{out})$) is called **good** online view (belongs to τ_{good}) w.r.t. τ_{in} if (τ_{in}, τ_{out}) is an online view (i.e. $(M_i[1..j] = M_{i'}[1..j]) \Rightarrow (C_i[j] = C_{i'}[j])$) and the following conditions hold:

1. $C_i[j] = C_{i'}[j']$ implies that $j = j'$, $(D_i, M_i[..j]) = (D_{i'}, M_{i'}[..j])$
2. $\forall\ (i, l_i + 1) \neq (i', j'),\ C_i[l_i + 1] \neq C_{i'}[j']$.

The conditions says that a view is good if we can have collision of ciphertext blocks or intermediate tags in a position only if they are ciphertexts of two messages with same prefixes up to that block and all the final tags are fresh. We can easily show that the bad online views (views those are not good) has negligible probability :

Lemma 4.2.1 (Obtaining a Good view has high probability)

$$\mathsf{Pr}[\tau(A^{\$_{ol}}) \notin \tau_{good}] \ \leq\ \epsilon_1$$

where $\epsilon_1 \ = \ \frac{\sigma_{\text{priv}}^2}{2^n}$

Proof. The probability of having each such bad case is $\frac{1}{2^n}$. As at most $\binom{\sigma_{\text{priv}}}{2}$ pairs are there for the 1st and $q\sigma_{\text{priv}}$ pairs for 2nd case, the result follows.

We now fix a good view $\tau = (\tau_{in}, \tau_{out})$ as mentioned above. In the following result, we'll prove that the interpolation probability, i.e. $\mathsf{Pr}[\tau(A^{\mathsf{ELmD}}) = \tau]$ is high for τ. [1] Note that $\mathsf{Pr}[\tau(A^{\$^{ol}}) = \tau] = 2^{-nP}$ where P denotes the number of non-empty prefixes of (D_i, M_i), $1 \leq i \leq q$ as for every different prefixes, $\$^{ol}$ assigns an independent and uniform ciphertext blocks.

Lemma 4.2.2 (High interpolation probability of ELmD) $\forall \tau \in \tau_{good}$,

[1] Note that, if we define L from E_K then we need to revise the proof of the Lemma 4.2.4 to obtain a modified value of ϵ_2. The revision is mainly by defining more internal bad events that some of the Π inputs is 0 (the inputs are used to generate L-values). As this adds notational complexity and does not increase the order of advantage (except a very small constant factor will increase) we skip it.

$$\mathsf{Pr}[\tau(A^{\mathsf{ELmD}_\Pi}) = \tau] \geq (1 - \epsilon_2) \times \mathsf{Pr}[\tau(A^{\$^{ol}}) = \tau].$$

where $\epsilon_2 = \frac{4\sigma^2_{\text{priv}}}{2^n}$.

Proof. As adversary is deterministic, we restrict to those good views which can be obtained by A. Hence the probability $\mathsf{Pr}[\tau(A^{\mathsf{ELmD}}) = \tau]$ is same as

$$\mathsf{Pr}[\mathsf{ELmD}_\Pi(D_i, M_i) = C_i, 1 \leq i \leq q].$$

Before computing interpolation probability we denote all intermediate variables while computing $\mathsf{ELmD}_\Pi(D_i, M_i) = C_i$. Let for all i and j whenever defined

1. $DD_i[j] = 3 \cdot L \cdot 2^{j-1} + D_i[j]$, $MM_i[j] = L \cdot 2^{j-1} + M_i[j]$
2. $\Pi(DD_i[j]) = Z_i[j]$, $\Pi(MM_i[j]) = X_i[j]$,
3. $\mathsf{mix}(Z_i, X_i) = Y_i$ and
4. $CC_i[j] = 3^2 \cdot L \cdot 2^{j-1} + C_i[j]$

Here mix is the linear mixing layer (defined through the ρ functions) that takes Z_i and X_i to define Y_i. Observe that CC has been defined through tagged-ciphertext and L instead of applying Π on Y blocks. We denote the q-tuple $(DD_1, \ldots, DD_q)$ as $\mathbf{DD}$. We define $\mathbf{MM}$, $\mathbf{Z}, \mathbf{X}, \mathbf{Y}$ and $\mathbf{CC}$ similarly. We have $\mathsf{mix}(\mathbf{Z}, \mathbf{X}) = \mathbf{Y}$ with the extended definition of mix which applies mix function for each (Z_i, X_i).

Collision Relation. Now we define a collision relation of a vector $(x_1, \ldots, x_t)$ by the equivalence relation $\mathsf{coll}(x)$ for which i is related to j if and only if $x_i = x_j$.

The irreducible polynomial $f(x) = x^{128} + x^7 + x^2 + x + 1$ ensures distinct values (none equal to 1) of $2^\alpha 3^\beta 7^\gamma$ for $\alpha \in [-2^{108}, 2^{108}]$ and $\beta, \gamma \in [-2^7, 2^7]$[19]. This property ensures that the masking values corresponding to D_i, M_i and C_i's (of the form $3.2^i.L$, $2^i.L$ and $3^2.2^i.L$ respectively) are distinct. Moreover the masking value $S_i[j]$ will be equal to the masking value of $S_{i'}[j']$ if and only if $j = j'$ where $S \in \{D, M, C\}$.
L is called **valid** if it computes $(\mathbf{DD}, \mathbf{MM}, \mathbf{CC})$ for which only equality among the blocks occurs in $SS_i[j] = SS_{i'}[j]$ where $S_i[j] = S_{i'}[j]$. Now, the primitivity of 2 and uniform independent choice of L ensures that each equality violating has probability 2^{-n} and there are at most $\binom{2\sigma_{\text{priv}}}{2}$ equality the result follows. So, using union bound applied to all the cases which violates that L is valid, we have

$$\mathsf{Pr}[L \text{ is valid}] \geq (1 - \epsilon_2') \text{ where } \epsilon_2' = \frac{2\sigma^2_{\text{priv}}}{2^n}.$$

Now we establish two collision relations γ_1 and γ_2 which are consistent with the linear mix function. These relations are defined based on a good view τ. Let γ_1 be the collision relation defined on the set $\{(i, j, M) : i \leq q,\ j \leq l_i + 1\} \bigcup \{(i, j, D) : i \leq q, j \leq d_i\}$. A pair $((i, j, S), (i', j', S'))$ is related if $S = S'$,

$j = j'$ and $S_i[j] = S_{i'}[j]$. All other pairs are unrelated. Let γ_2 be the a collision relation defined on the set $\{(i, j, C) : i \leq q,\ j \leq l_i + 1\}$ for which only pairs $((i, j, C), (i', j', C))$ if $j = j'$ and $C_i[j] = C_{i'}[j]$. Let the no. of equivalence class of γ_i be s_i, $i = 1, 2$. Note that $s_2 = P$, the number of prefixes of (D_i, M_i) containing at least one message block. Now, Let $\mathbf{Y} = (Y_1 := \mathsf{mix}(Z_1, X_1), \ldots, Y_q := \mathsf{mix}(Z_q.X_q))$. Since the view is good, $C_i[j] = C_{i'}[j]$ can happen if $D_i = D_{i'}$ and $M_i[..j] = M_{i'}[..j]$. For these cases, clearly, $Y_i[j] = Y_{i'}[j]$. Now for any other pair $((i, j), (i', j'))$, it is easy to see that mix function leads to a non-trivial equation $\mathsf{mix}_j(X_i^{\gamma_1}) = \mathsf{mix}_{j'}(X_{i'}^{\gamma_1})$. Hence,

$$(\gamma_1, \gamma_2) \textbf{ is consistent with } \mathsf{mix}.$$

Now, applying the consistent collision relations for linear functions (see Lemma 3.2.5), we have :

$$\#\{(Z, X) : \mathsf{coll}(Z, X) = \gamma_1, \mathsf{coll}(Y) = \gamma_2\} \geq 2^{ns_1}(1 - \epsilon_2'') \text{ where } \epsilon_2'' = \frac{2\sigma_{\text{priv}}^2}{2^n}$$

Now, for a fixed valid L, the conditional interpolation probability is

$$\sum_{(Z,X)} \frac{\#\Pi : \Pi(MM) = X, \Pi(DD) = Z, \Pi(CC) = Y}{\#\Pi} \geq (1 - \epsilon_2'') \times 2^{-nP}.$$

So by multiplying the probability for validness of L the proof of the lemma completes.

4.2.2 Confidentiality for ELmD_Π when $\mathbf{t} > 0$

Theorem 4.2 *Let A be an adversary which can make q queries at an aggregrate of total σ associated data and message blocks. Let $\sigma_{\text{priv}} = \sigma + q$. The online privacy advantage of the adversary A is given by,*

$$\mathbf{Adv}_{\mathsf{ELmD}}^{\text{opriv}}(A) \ \leq \ \frac{5\sigma_{\text{priv}}^2}{2^n} + \frac{7\sigma_{\text{priv}}^2}{t.2^n}.$$

Proof. The previous proof can be extended to deal with intermediate tags. Here, also we fix q associated data, plaintext pairs - $P_1 = (D_1, M_1), \cdots, P_q = (D_q, M_q)$ and denote $(P_1, \cdots, P_q)$ as τ_{in}. A tagged ciphertext tuple with intermediate tags is denoted by $\tau_{out} = ((C_1, T_1), \cdots, (C_q, T_q))$. Now, we call a online view good if the previous two conditions hold and moreover we make sure that all the ciphertexts along with the final tag are fresh from all the intermediate tags and two intermediate tags are same only if associated data, plaintext pair upto that is same. It is straightforward to see that :

Lemma 4.2.3 (Obtaining a Good view has high probability)

$$Pr[\tau(A^{\$_{ol}}) \notin \tau_{good}] \ \leq \ \epsilon_1$$

where $\epsilon_1 = \frac{\sigma_{\text{priv}}^2}{2^{n+1}} + \frac{q(\sigma_{\text{priv}} + \frac{\sigma_{\text{priv}}}{t})}{2^n} \ + \ \frac{\sigma_{\text{priv}}^2}{t.2^n} \ + \ \frac{\sigma_{\text{priv}}^2}{t^2.2^{n1}} \leq \frac{\sigma_{\text{priv}}^2}{2^n} + \frac{2\sigma_{\text{priv}}^2}{t.2^n}$

Now, we fix a good view $\tau = (\tau_{in}, \tau_{out})$ and will prove the following lemma :

Lemma 4.2.4 (High interpolation probability of ELmD) $\forall \tau \in \tau_{good}$,

$$\mathsf{Pr}[\tau(A^{\mathsf{ELmD}_{\Pi,\mathbf{L}}}) = \tau] \geq (1 - \epsilon_2) \times \mathsf{Pr}[\tau(A^{\$^{ol}}) = \tau].$$

where $\epsilon_2 = \frac{4\sigma^2_{\text{priv}}}{2^n} + \frac{5\sigma^2_{\text{priv}}}{t.2^n}$.

To prove this, we define $\mathbf{DD}, \mathbf{MM}, \mathbf{Z}, \mathbf{X}, \mathbf{Y}$ as earlier. We modify the definition of $\mathbf{CC}$ and define $\mathbf{TT}$ and incorporate $\mathbf{H}$ as follows :

1. $\mathsf{mix}(Z_i, X_i) = (Y_i, H_i)$,
2. $CC_i[j] = 3^2 \cdot 2^{j-1+\lfloor \frac{j-1}{\mathtt{t}} \rfloor} \cdot L + C_i[j]$
3. $TT_i[j] = 3^2 \cdot 2^{j\mathtt{t}+j-1} \cdot L + T_i[j]$

We call L valid if it computes $(\mathbf{DD}, \mathbf{MM}, \mathbf{CC}, \mathbf{TT})$ such that equally occurs only when $SS_i[j] = SS_{i'}[j]$ where $S_i[j] = S_{i'}[j']$. It is easy to check that,

$$\mathsf{Pr}[L \text{ is valid}] \geq (1 - \epsilon'_2) \text{ where } \epsilon'_2 = \frac{2\sigma^2_{\text{priv}}}{2^n} + \frac{3\sigma^2_{\text{priv}}}{\mathtt{t}.2^n}.$$

Now, we define the collision relation γ_1 and γ_2. The definition of γ_1 remains same as earlier. γ_2 is modified and defined on the set $\{(i, j, C) : i \leq q, j \leq l_i + 1\} \bigcup \{(i, j, T) : i \leq q, j \leq h_i\}$. A pair $(i, j, S), (i', j', S')$ is related if $S = S'$, $j = j'$ and $S_i[j] = S_{i'}[j]$. It is easy to check that the modified relation (γ_1, γ_2) is consistent with mix. So, applying the consistent collision relations for linear functions (see Lemma 3.2.5), we have :

$$\#\{(Z, X) : \mathsf{coll}(Z, X) = \gamma_1, \mathsf{coll}(Y) = \gamma_2\} \geq 2^{ns_1}(1 - \epsilon''_2)$$

where $\epsilon''_2 = \frac{2\sigma^2_{\text{priv}}}{2^n} + \frac{2\sigma^2_{\text{priv}}}{\mathtt{t}2^n}$

Now, for a fixed valid L, the conditional interpolation probability is

$$\sum_{(Z,X)} \frac{\#\Pi : \Pi(MM) = X, \Pi(DD) = Z, \Pi(CC) = Y, \Pi(TT) = H}{\#\Pi} \geq (1 - \epsilon''_2) \times 2^{-nP}.$$

So by multiplying the probability for validness of L the proof of the proposition completes.

4.2.3 Consistent collision relations for a linear function

Suppose $X = X[..r_1]$ is a r_1-tuple of variables of $\mathbb{B}$ and $\mathcal{L} : \mathbb{B}^{r_1} \to \mathbb{B}^{r_2}$ be a linear function. We denote $Y = \mathcal{L}(X)$ which is a r_2-tuple of variables from $\mathbb{B}$. We can have an equivalent represent through matrix notation: $Y = \mathcal{L}.X$ where $\mathcal{L}$ is a matrix. We denote the i^{th} row of the matrix $\mathcal{L}$ by $\mathcal{L}_i$. Let γ_1 and γ_2 be two equivalence relations defined on the sets respectively $[1..r_1]$ and $[1..r_2]$. Suppose X^{γ_1} denotes the tuple of variables which satisfies the collision relation

γ_1 by replacing identical variables by the variable which occurred with minimum index. (γ_1, γ_2) is said to be **consistent** with $\mathcal{L}$ if $\mathcal{L}_i(X^{\gamma_1}) \equiv \mathcal{L}_j(X^{\gamma_1})$ if and only if i and j are related in γ_2. Clearly, given any γ_1 and L there is exactly one γ_2 for which (γ_1, γ_2) is consistent with $\mathcal{L}$. We write $\gamma_1 \Rightarrow_L \gamma_2$.

Example. If $\gamma_1 = \{\{1,3\},\{2\},\{4,6\},\{5\}\}$ for $r_1 = 6$, then we write $X^{\gamma_1} = (X_1, X_2, X_1, X_4, X_5, X_4)$. Let $\mathcal{L}$ map into three variables (i.e., $r_2 = 3$ such that $\mathcal{L}_1 = X_1 + X_2 + X_3 + X_6$, $\mathcal{L}_2 = X_4 + X_5 + X_6$ and $\mathcal{L}_3 = X_2 + X_4$ then $\mathcal{L}_1(X^{\gamma_1}) = \mathcal{L}_3(X^{\gamma_1}) = X_2 + X_4$ and $\mathcal{L}_2(X^{\gamma_1}) = X_5$ (we work it here in binary field). So $\gamma_1 \Rightarrow_{\mathcal{L}} \gamma_2$ where $\gamma_2 = \{\{1,3\},\{2\}\}$. Note, that here + denotes modulo 2 addition.

Lemma 4.2.5 *[Number of Solutions for Consistent relations] Let (γ_1, γ_2) be consistent with $\mathcal{L} : \mathbb{B}^{r_1} \to \mathbb{B}^{r_2}$ then*

$$|\{X : Coll(X) = \gamma_1, Coll(\mathcal{L}(X)) = \gamma_2\}| \geq 2^{ns_1} \times (1 - \frac{s^2}{2^{n+1}})$$

where s_1 and s_2 denote the number of equivalence classes of γ_1 and γ_2 respectively and $s = s_1 + s_2$.

Let $Y = \mathcal{L}(X)$. Because of consistency, for all related i, j in γ_2, $Y_i = Y_j$. There may be additional equality which must be avoided. For all unrelated pair (i, j) in γ_2 we must choose X in a manner such that $Y_i \neq Y_j$ and similarly for all unrelated pair (i, j) in γ_1 we have $X_i \neq X_j$. Due to consistency, any one can happen for at most $2^{n(s_1-1)}$ many X's as $\mathcal{L}_i(X^{\gamma_1}) = \mathcal{L}_j(X^{\gamma_1})$ gives a non-trivial equation. So the result follows as we have at most $\binom{s}{2}$ such equalities.

4.3 Integrity of ELmD_Π

We say that an adversary A **forges** an authenticated encryption F if A outputs (D, C) where $F_K(D, C) \neq \perp$ (i.e. it accepts and returns a plaintext), and A made no earlier query (D, M) for which the F-response is C. It can make s attempts to forge after making q queries. We define that A forges if it makes at least one forges in all s attempts and the **authenticity-advantage** of A by

$$\mathbf{Adv}^{\text{auth}}_F(A) = \mathsf{Pr}_K[A^{F_K} \text{ forges}].$$

Suppose for any valid tuple of associate data and tagged ciphertext (D, C), with $|C| = l + 1$, the tag $C[l+1]$ can be computed from $(D, C[..l])$. We write $C[l+1] = T_K(D, C[..l])$. So (D, C) is a valid tagged ciphertext if and only if $T_K(D, C[..l]) = C[l+1]$. Almost all known authenticated encryptions F (including those following encrypt-then-mac paradigm) have this property for a suitably defined ciphertext C and tag function T. We know that PRF implies MAC. We use similar concept to bound authenticity. More formally, for any forgery B, there is a distinguisher A such that

$$\mathbf{Adv}^{\text{auth}}_F(B) \leq \mathbf{Adv}^{\mathcal{O},\$}_{(F,T)}(A) + \frac{s}{2^n} \tag{4.1}$$

4.3.1 Integrity of ELmD_{Π} when $\mathbf{t} = 0$

Theorem 4.3 *Let A be an adversary which can make q forward queries and tries to forge s many times at an overall aggregrate of total σ associated data and message blocks. Let $\sigma_{\text{auth}} = \sigma + q$. Then the forging advantage of the adversary is given by,*

$$\mathbf{Adv}^{\text{auth}}_{\mathsf{ELmD}_{\Pi}}(\mathcal{A}) \;\leq\; \frac{9\sigma^2_{\text{auth}}}{2^n} + \frac{s}{2^n}$$

Proof. The proof is done using the coefficient H technique and following lemma 3.3.1 and 3.3.2 and then using equation 3.1. For notational simplicity, we write $\mathsf{ELmD}_{\Pi,\mathbf{L}}$ by F. As we have observed in Eq 3.1, we only need to show indistinguishability for which we apply the coefficient H technique again. For this, we need to identify set of good views for which we have high interpolation probability.

A (F,T)-view of a distinguisher A is the pair $v = (\tau_F, \tau_T)$ where $\tau_F = (D_i, M_i, C_i)_{1\leq i\leq q}$ is an q-tuple of F-online view and $\tau_T = (D_j, C_j)_{q<j\leq q+s}$ is an s-tuple non-trivial T-view. It is called **good** online forge view, if τ_F is **good** online view (as defined in the privacy prove) and for all $q < j \leq q+s$, $C_j[l_j+1]$'s are fresh - distinct and different from all other $C_i[j]$'s. Suppose, $|D_i| = d_i$, $|M_i| = l_i$ and $|C_i| = l_i + 1$. Let $\sigma = \sum_{i=1}^{q+s}(d_i + l_i)$ and $\sigma_{auth} = \sigma + q$ (the total number of tagged ciphertext blocks). The forging advantage of ELmD is given by: Since F is online function we consider pair of independent oracles $(\$_{ol}, \$)$ where $\$_{ol}$ denotes the random online function and $\$$ is simply a random function.

Lemma 4.3.1 (Realizing Good Forge View has high probability) *For all adversary A,*

$$\mathsf{Pr}[\tau(A^{\$_{ol},\$}) \notin \tau_{good}] \leq \frac{(q + \sum_{i=1}^{q}(d_i + l_i))^2}{2^{n+1}} + \frac{s(q + s + \sum_{i=1}^{q+s}(d_i + l_i))}{2^n} \leq \epsilon_1.$$

where $\epsilon_1 = \frac{2\sigma^2_{auth}}{2^n}$

As in Lemma 4.2.1, we can similarly prove the above. The first summand takes care the collisions in $C_i[j]$'s (i.e., the bad view for τ_F as in Proposition 4.2.1) and the second summand takes care the collision between $C_i[l_i+1]$'s $(q < i \leq q+s)$ and all other $C_i[j]$'s. Now we fix a good view $\tau = (\tau_F, \tau_T)$ as defined above (following same notations). Now it is easy to see that obtaining τ interacting with $(\$_{ol}, \$)$ has probability $2^{-nP} \times 2^{-ns} = 2^{-n(P+s)}$ where P denotes the number of non-empty prefixes of C_i, $1 \leq i \leq q+s$ (at those blocks random online function returns randomly).

Lemma 4.3.2 (Good Forge View has high interpolation probability) *For any good (F,T)-view τ, we have*

$$\mathsf{Pr}[F(D_i, M_i) = C_i, \forall i \leq q,\; T(D_j, C_j[..l_j]) = C_j[l_j{+}1], q < j \leq q{+}s] \;\geq\; \frac{(1-\epsilon_2)}{2^{n(P+s)}}$$

where $\epsilon_2 = 7\sigma^2_{\text{auth}}/2^n$.

Proof. We choose $X_1, \ldots, X_q$ and then $Y_{q+1}, \ldots, Y_{s+q}$ which fix all internal X and Y values except the last block for the s many T-queries. We explicitly provide counting steps by steps. We choose valid L which fixes MM's for the first q messages and, CC's and DD's for all $s+q$ queries. We can then choose MM for these s queries so that checksums are all fresh and for all these fresh checksums we can ensure last Y blocks fresh by choosing X blocks appropriately. Now we make these choices one by one more formally.

Choices of Valid L.

We first define valid L-triples as defined in privacy. L is called valid w.r.t. the fixed good (F,T)-view τ if the computed MM, DD, CC values satisfy the collision relations described below and whenever C_j, $j > q$, is a strictly prefix of C_i, $i \leq q$ and $D_i = D_j$ then $MM_i[l_i] \neq MM_j[l_j]$, i.e., equivalently $M_i[l_j+1]+\ldots M_i[l_i]+L(2^{l_j}+\cdots+2^{l_i}) \neq 0$. To define the collision (equivalence) relation, we mention those places where equivalence occurs. In all other places these are not related. $SS_i[j] \equiv SS_{i'}[j]$ if $S_i[j] = S_{i'}[j]$ where S represents any one of the four symbols M, D, and C. So they can be identical only if their positions as well as symbols (or types of the input) match. The simple counting argument with union bound applied to all individual bad events proves the following result.

$$\mathsf{Pr}[L \text{ is valid}] \;\geq\; (1-\epsilon_1').$$

where $\epsilon_1' = \frac{2\sigma^2_{\text{auth}}}{2^n}$

Choices of valid Z, X, Y except the last blocks for the last s queries.

As in the privacy proof, τ_F induces consistent collision relations of $(\mathbf{Z}, \mathbf{X}) := (\mathbf{Z}_1, \ldots, \mathbf{Z}_q, \mathbf{X}_1, \ldots, \mathbf{X}_q)$ and $\mathbf{Y} := (\mathbf{Y}_1, \ldots, \mathbf{Y}_q)$. Now we extend this collision relation to $(\mathbf{Z}_{q+1}, \mathbf{Y}_{q+1}, \ldots, \mathbf{Z}_{q+s}, \mathbf{Y}_{q+s})$ as follows for $j < i \leq q+s$:

1. $\mathbf{Z}_i[j] \equiv \mathbf{Z}_{i'}[j']$ if $j = j'$ and $D_i[j] \equiv D_{i'}[j]$.
2. $\mathbf{Y}_i[j] \equiv \mathbf{Y}_{i'}[j']$ if $j = j'$ and $C_i[j] \equiv C_{i'}[j]$.

The collision relation on $(\mathbf{Z}, \mathbf{Y})$ induces a collision relation on $\mathbf{X}_f := (\mathbf{X}_{q+1}, \ldots, \mathbf{X}_{q+s})$ through the linear mix^{-1} function. That is, $(\mathbf{Z}, \mathbf{Y}) \Rightarrow_{\mathsf{mix}^{-1}} \mathbf{X}_f$. Let γ_1' be the extended collision relation on $(\mathbf{Z}, \mathbf{X})$ and γ_2' be that of $\mathbf{Y}$. We denote the number of equivalence classes by s_1' and s_2'. By using the counting on consistency relations (see Lemma 4.2.5) the number of (Z, X, Y) with $\mathsf{mix}(Z, X) = Y$ and $\mathsf{coll}(Z, X) = \gamma_1'$, $\mathsf{coll}(Y) = \gamma_2'$ is at least

$$2^{n(s_1+s_3)}(1 - \frac{(s_1'+s_2')^2}{2^{n+1}}) \geq 2^{n(s_1+s_3)}(1-\epsilon_2')$$

where $\epsilon_2' = \frac{2\sigma_{\text{auth}}^2}{2^n}$ and s_3 denotes the number of additional equivalence classes in $\mathbf{Y}_f$ which are not present in $(\mathbf{Y}_1, \ldots, \mathbf{Y}_q)$.

Now, the remaining last s blocks can be chosen freely which determines all other blocks. Before that, we state an important property of these collision relations γ_1' and γ_2' :

$$(\exists\ j > q, \forall k \leq \ell_j,\ X_j[k] \equiv X_{r_k}[k],\ r_k \leq q) \Rightarrow (\exists i \leq q\ \forall k \leq \ell_j,\ X_j[k] \equiv X_i[k]).$$

This means the message corresponding to a forged ciphertext is the prefix of some other messages, queried previously by the adversary.

Proof. Let us fix $j = q + 1$ (for all other j, the argument is similar) and denote l_j by l. Now we have the following identities: $X_{q+1}[k] \equiv X_{r_k}[k]$ for all k. This can happen only if $Y_{q+1}[j] \equiv Y_{t_j}[j]$ for some $t_j \leq q$, otherwise $X_{q+1}[j]$ would get completely new variable which is not present in all first q queries. Now if we write $X_{q+1}[j]$ in terms of these X_{t_j}'s variable one can obtain the desired result.

Choices of MM for forging s queries.

Given the choices of valid L and those of X, Y, Z as described above we can now choose remaining MM values satisfying same collision relation as $(X_{q+1}, \ldots, X_{q+s})$. More precisely, we can choose all those MM values for which $X_j[i]$'s are fresh. Let s_4 denote the number of additional distinct blocks in $(X_{q+1}, \ldots, X_{q+s})$ which are not present in $(X_1, \ldots, X_q)$. The number of these s_4 blocks MM different from all other defined MM, DD and CC blocks such the all last blocks of MM_j's $(j > q)$ are fresh is at least $2^{ns_4}(1 - \epsilon_3')$, where $\epsilon_3' = \frac{2\sigma_{\text{auth}}^2}{2^n}$. Note that $MM_i[l_i + 1] = MM_{i'}[l_{i'} + 1]$ induces a restriction on choices of MM.

Choices of last block of X for these s queries.

For any such previous choices, we now choose the blocks of $X_j[l_j + 1]$, $j > q$ so that the last block of Y_j's are fresh. This can be chosen in $2^{ns}(1 - \epsilon_4')$ ways, where $\epsilon_4' = \frac{\sigma_{\text{auth}}^2}{2^n}$.

Armed with all these counting, the interpolation probability is at least

$$(1 - \epsilon_2) \times 2^{-n(P+s)}.$$

where $\epsilon_2 = \epsilon_1' + \epsilon_2' + \epsilon_3' + \epsilon_4' = \frac{7\sigma_{\text{auth}}^2}{2^n}$

4.3.2 Integrity of ELmD_Π when $0 \leq \mathbf{t} \leq 127$

We say that an adversary A **forges the intermediate tag** in an authenticated encryption F if A outputs $(D, C[..\tau k], T[..k])$ where $F_K(D, C[..\tau k], T[..k]) \neq \perp$ (i.e. it accepts and returns a plaintext). We assume, A made no earlier query

(D, M) for which $(C[..\tau k], T[..k])$ is the F-response. Moreover, we assume that A has made some queries (D, M) for which $(C[..\tau(k-1)], T[..(k-1)])$ is a prefix of F's response. We need this condition, because we are looking for first position where the intermediate tag forging occurs.

Theorem 4.4 *Let A be an adversary which can make q forward queries and tries to forge (the final tag and/or intermediate tag) s many times at an overall aggregrate of total σ associated data and message blocks. Let $\sigma_{\text{auth}} = \sigma + q$. Then the forging advantage of the adversary is given by,*

$$\mathbf{Adv}^{\text{auth}}_{\mathsf{ELmD}_\Pi}(\mathcal{A}) \;\leq\; \frac{10\sigma^2_{\text{auth}}}{2^n} + \frac{16\sigma^2_{\text{auth}}}{\mathtt{t}\ 2^n} + \frac{s}{2^n}$$

Proof. The proof is done again using the coefficient H technique and following lemma 3.3.1 and 3.3.2 and then using equation 3.1. the details is given below :

A (F, T)-view of a distinguisher A is the pair $v = (\tau_F, \tau_T)$ where $\tau_F = (D_i, M_i, C_i, T_i)_{1\leq i\leq q}$ is an q-tuple of F-online view and $\tau_T = (D_j, C_j, T_j)_{q<j\leq q+s}$ is an s-tuple non-trivial T-view. Note, there are two types of forging queries - some with intermediate tag forging and others that forges the final tag (in case forging query's length is less than $\mathtt{t}$ or the length is long but upto last generated intermediate tag, the ciphertext is a prefix of some previous queried message). Let q_t and q_{it} denotes the no. of attempted forging queries against the final tag and intermediate tag respectively. Clearly $q = q_t + q_{it}$. W.l.o.g assume that, all the forging queries against final tags are performed first and then the queries against the intermediate tag forgings are done. It is called **good** online intermediate tag forge view, if τ_F is Good Online view (as defined in the privacy prove) and for all $q < j \leq q + q_t$, $C_j[l_j + 1]$'s are fresh and for all $q + q_t < j < q + s$, $T_j[l_j]$'s are fresh - distinct and different from all other $C_i[j]$'s and $T_i[j]$'s. Suppose, $\forall i \leq q + s$, $|D_i| = d_i$ $|C_i| = l_i$. Let $\sigma_{auth} = \sum_{i=1}^{q+s}(d_i + l_i) + q$. Since F is online function we consider pair of independent oracles $(\$_{ol}, \$)$ where $\$_{ol}$ denotes the random online function and $\$$ is simply a random function. It is easy to see from the previous proof that,

Lemma 4.3.3 (Realizing Good Forge View has high probability) *For all adversary A,*

$$Pr[\tau(A^{\$_{ol},\$}) \notin \tau_{good}] \leq \epsilon_1$$

where $\epsilon_1 = \frac{2\sigma^2_{auth}}{2^n} + \frac{3\sigma^2_{auth}}{\mathtt{t}\ 2^n}$. Now we fix a good view $\tau = (\tau_F, \tau_T)$ as defined above (following same notations). Now it is easy to see that obtaining τ interacting with $(\$_{ol}, \$)$ has probability $2^{-nP} \times 2^{-ns} = 2^{-n(P+s)}$ where P denotes the number of non-empty prefixes of C_i, $1 \leq i \leq q + s$ (at those blocks random online function returns randomly).

Lemma 4.3.4 (Good Forge View has high interpolation probability) *For any good (F, T)-view τ, we have,*

$$\mathsf{Pr}[F(D_i, M_i) = C_i \ \forall i \le q,\ T(D_j, C_j[..l_j]) = C_j[L_j + 1] \ \forall q < j \le q + q_t,$$
$$T(D_j, C_j[..l_j]) = T_j[l_j] \ \forall q + q_t < j \le q + s] \ \ge \ \tfrac{(1-\epsilon_2)}{2^{n(P+s)}}$$

where $\epsilon_2 = \frac{8\sigma^2_{\text{auth}}}{2^n} + \frac{13\sigma^2_{\text{auth}}}{t\ 2^n}$.

Proof. We choose $(Z_1, \ldots, Z_{q+s})$, $(X_1, \ldots, X_q)$ for the first q queries and then $(Y_{q+1}, \ldots, Y_{s+q})$, which fix all internal Z, X, Y and W values. We first choose valid L which fixes MM's for the first q messages and CC's, TT's and DD's for all $s+q$ queries. Now, for the first q_t queries, like the previous proof, we can then choose MM's so that checksums are all fresh and for all these fresh checksums we can ensure last Y blocks fresh by choosing X blocks appropriately. Now we make these choices one by one more formally. For the next q_{it} queries, we choose X such that last W values for them are fresh.

Choices of Valid L.

We first define valid L-triples as defined in privacy. L is called valid w.r.t. the fixed good (F,T)-view τ if the computed MM, DD, CC, TT values satisfy the collision relations as described earlier. The simple counting argument with union bound applied to all individual bad events proves the following result.

$$\mathsf{Pr}[L \text{ is valid}] \ \ge \ (1 - \epsilon'_1)$$

where $\epsilon'_1 = \frac{2\sigma^2_{\text{auth}}}{2^n} + \frac{3\sigma^2_{\text{auth}}}{\mathsf{t}\ 2^n}$.

Choices of valid Z, X, Y, H except the last X blocks of the q_t queries

As in the previous proof, τ_F induces consistent collision relations of $(\mathbf{Z}, \mathbf{X}) := (\mathbf{Z}_1, \ldots, \mathbf{Z}_q, \mathbf{X}_1, \ldots, \mathbf{X}_q)$. Now we extend this collision relation to $(\mathbf{Z}_{q+1}, \mathbf{Y}_{q+1}, \mathbf{H}_{q+1}$ $\ldots, \mathbf{Z}_{q+s}, \mathbf{Y}_{q+s}, \mathbf{H}_{q+s})$ as follows for $j < i \le q + s$:

1. $\mathbf{Z}_i[j] \equiv \mathbf{Z}_{i'}[j']$ if $j = j'$ and $D_i[j] \equiv D_{i'}[j]$.
2. $\mathbf{Y}_i[j] \equiv \mathbf{Y}_{i'}[j']$ if $j = j'$ and $C_i[j] \equiv C_{i'}[j]$.

The extended collision relation on $(\mathbf{Z}, \mathbf{Y})$ induces a collision relation on $\mathbf{X}_f := (\mathbf{X}_{q+1}, \ldots, \mathbf{X}_{q+s})$ through the linear mix^{-1} function. That is, $(\mathbf{Z}, \mathbf{Y}) \Rightarrow_{\mathsf{mix}^{-1}} \mathbf{X}_f$. Let γ'_1 be the extended collision relation on $(\mathbf{Z}, \mathbf{X})$ and γ'_2 be that of $\mathbf{Y}$. We denote the number of equivalence classes by s'_1 and s'_2. By using the counting on consistency relations (see Lemma 4.2.5) the number of (Z, X, Y) with $\mathsf{mix}(Z, X) = Y$ and $\mathsf{coll}(Z, X) = \gamma'_1$, $\mathsf{coll}(Y) = \gamma'_2$ is at least

$$2^{n(s_1+s_3)}(1 - \frac{(s'_1 + s'_2)^2}{2^{n+1}}) \ge 2^{n(s_1+s_3)}(1 - \epsilon'_2)$$

where $\epsilon'_2 = \frac{2\sigma^2_{\text{auth}}}{2^n} + \frac{3\sigma^2_{\text{auth}}}{\mathsf{t}\ 2^n}$ and s_3 denotes the number of additional equivalence classes in $\mathbf{Y}_f$ which are not present in $(\mathbf{Y}_1, \ldots, \mathbf{Y}_q)$.

Now, for the q_t queries, it is easy to check, we have the property that the message corresponding to a forged ciphertext is the prefix of some other messages, queried previously by the adversary. The argument is similar to that used in the previous integrity proof.

Choices of X for the first q_t forging queries.

Given the choices of valid L and those of X, Y, Z as described above we can now choose remaining MM values satisfying same collision relation as $(X_{q+1}, \ldots, X_{q+q_t})$. More precisely, we can choose all those MM values for which $X_j[i]$'s are fresh. Let s_4 denote the number of additional distinct blocks in $(X_{q+1}, \ldots, X_{q+q_t})$ which are not present in $(X_1, \ldots, X_q)$. The number of these s_4 blocks MM different from all other defined MM, DD and CC blocks such the all last blocks of MM_j's $(j > q)$ are fresh is at least $2^{ns_4}(1 - \epsilon_3')$, where $\epsilon_3' = \frac{2\sigma_{\text{auth}}^2}{2^n} + \frac{3\sigma_{\text{auth}}^2}{\mathtt{t}\ 2^n}$

Choices of last block of X for these q_t queries.

For any such previous choices, we now choose the blocks of $X_j[l_j+1]$, $q < j \leq q_t$ so that the last block of Y_j's are fresh. This can be chosen in $2^{ns}(1 - \epsilon_4')$ ways, where $\epsilon_4' = \frac{\sigma_{\text{auth}}^2}{2^n} + \frac{2\sigma_{\text{auth}}^2}{\mathtt{t}\ 2^n}$.

Now we claim that, last blocks of T_j's for the q_{it} queries are fresh. The claim is proved in Lemma 3.3.5. Using the result, now we have to choose the X's such that the last block of H_j's are fresh.

Choices of X for the last q_{it} queries.

For the last q_{it} queries, we choose the blocks of X_j's so that the last block of W_j's are fresh. This can be done $2^{ns}(1 - \epsilon_5')$ ways where $\epsilon_5' = \frac{\sigma_{\text{auth}}^2}{2^n} + \frac{2\sigma_{\text{auth}}^2}{\mathtt{t}\ 2^n}$.

Armed with all these counting, the interpolation probability is at least

$$(1 - \epsilon_2) \times 2^{-n(P+s)}.$$

where $\epsilon_2 = \epsilon_1' + \epsilon_2' + \epsilon_3' + \epsilon_4' + \epsilon_5' = \frac{8\sigma_{\text{auth}}^2}{2^n} + \frac{13\sigma_{\text{auth}}^2}{\mathtt{t}\ 2^n}$

Lemma 4.3.5 *A forged ciphertext upto c^{th} intermediate tag is valid, if it is a prefix of a ciphertext produced by some previous forward query.*

Proof. First we consider the case where $c = 1$. Then will generalize for any c. One can check that, if any of the blocks in the forged ciphertext is not identical to a previous response, then $T_i[1]$ will not be valid due to the randomness of Y value corresponding to that block. Similarly taking $T_i[1]$ as some final tag output will give some non-trivial equations. Hence, assume the forged Ciphertext be $(C_{i_1}[1]\ C_{i_2}[2] \cdots C_{i_t}[\mathtt{t}]\ T_i[1])$, where $C_{i_j}[j]$ is the j^{th} block of the ciphertext of (D_{i_j}, M_{i_j}) and $T_i[1]$ is the first intermediate tag block of message (D_i, M_i). If

the forged intermediate tag $T_i[1]$ is valid, then we have the following set of equalities in the X blocks : $\forall j \leq \mathsf{t}$,

$$2^{\mathsf{t}-j}X_i[j] \equiv 3 \cdot 2^{\mathsf{t}-j-1}X_{i_\mathsf{t}}[j] + 3 \cdot 2^{\mathsf{t}-j-2}X_{i_{\mathsf{t}-1}}[j] + \cdots + 3X_{i_{j+1}}[j] + X_{i_j}[j]$$

For $\mathsf{t} \leq 127$, this equation is trivial only if $\forall j \leq \mathsf{t},\ X_{i_\mathsf{t}}[j] \equiv \cdots \equiv X_{i_j}[j] \equiv X_i[j]$, otherwise we have an polynomial of 2 with degree $\leq \mathsf{t} - j < 128$, whose value is 0, which contradicts the primitivity of 2 in $GF(2^{128})$. Now, we have to consider the equalities in the Z blocks. Let d_z denotes the no. of associated data blocks in the message M_z. There are two cases :

- $d_{i_1} = \cdots = d_{i_\mathsf{t}} = d_i = d_f$: In this case, we has the following equalities in the Z blocks : $\forall j \leq d_i$,

  $$2^{\mathsf{t}}Z_i[j] \equiv 3 \cdot 2^{\mathsf{t}-1}Z_{i_\mathsf{t}}[j] + 3 \cdot 2^{\mathsf{t}-2}Z_{i_{\mathsf{t}-1}}[j] + \cdots + 3Z_{i_1}[j] + Z_f[j]$$

  For $\mathsf{t} \leq 127$, this equation is trivial only if $\forall j \leq k,\ Z_{i_\mathsf{t}}[j] \equiv \cdots \equiv Z_{i_j}[j] \equiv Z_f[j]$, otherwise again we have an polynomial of 2 with degree $\leq \mathsf{t} - j < 128$, whose value would be 0. Hence, the forged ciphertext is a prefix of the ciphertext corresponding to the i^{th} message. Note that, assigning $\mathsf{t} > 127$, vialotes this claim.

- Otherwise, Let $d_{max} = max\{d_{i_1}, \cdots, d_{i_\mathsf{t}}, d_i, d_f\}$. For $T_i[1]$ to be valid, the equality in the block $Z[d_{max}]$, violates the primitivity of 2. Note, that as for some message, there is no contribution to this block, assigning same value in this block for the remaining messages also gives a polynomial of 2, with degree less than 128, whose value is 0. So, this case doesn't occur.

This makes $\forall j \leq k,\ X_{i_\mathsf{t}}[j] \equiv X_i[j]$ meaning that the forged ciphertext block $C_f[1..\mathsf{t}] = C_i[1..\mathsf{t}]$, i^{th} ciphertext response.

Now we prove this for general c using induction. Suppose, our claim is true for c intermediate tag blocks. We have to show it for ciphertexts upto $(c+1)^{th}$ block. Consider the forged Ciphertext is $(Z_i, C_i[1] \cdots C_i[c\mathsf{t}]\ T_i[c]\ C_{i_{c\mathsf{t}+1}}[c\mathsf{t}+1]\ C_{i_{c\mathsf{t}+2}}[c\mathsf{t}+2] \cdots T_{i'}[c+1])$. If the ciphertext is valid, we have the following set of equalities for all $c\mathsf{t} < j \leq (c+1)\mathsf{t}$,

$$2^{c\mathsf{t}+\mathsf{t}-j}X_i[j] \equiv 3 \cdot 2^{c\mathsf{t}+\mathsf{t}-1-j}X_{i_{(c+1)\mathsf{t}}}[j] + \cdots + 3 \cdot 2^{c\mathsf{t}-j}X_{i_{c\mathsf{t}+1}}[j] + 2^{c\mathsf{t}-j}X_i[j]$$

which is again violating the primitivity of 2 unless $\forall c\mathsf{t} < j \leq c(\mathsf{t}+1),\ X_{i_{(c+1)\mathsf{t}}}[j] \equiv \cdots \equiv X_{i_j}[j] \equiv X_i[j]$. Hence the forge ciphertext is identical with the ciphertext of i^{th} query.

Chapter 5

Features

5.1 Main Features of the Cipher

5.1.1 Efficient and Nonce Misuse Resistant

Most of the nonce based authenticated encryption [18] schemes like AES-GCM [15], OCB3 [13], required the nonce to be distinct in every invocation. Failure to do so, leads easy attacks on the privacy of the scheme. Usually, one applies a counter or we choose it randomly (then repetition can happen with negligible probability) to ensure that distinct nonces are used during the tagged-encryption. But in practice, it is challenging to ensure that a nonce is never reused. For example, in lightweight applications, it is quite challenging to generate distinct nonce as it either needs to store a non-tamperable state or require some hardware source of randomness. Apart from that, there are various issues like flawed implementations or bad management by the user, for example where users with same key uses the same nonce. Nonce Misuse Resistance is an important criteria in the design of AE schemes.

Our construction ELmD is a nonce misuse resistant AE. Recall that, the concatanation of public message number pub and private message number priv is used as nonce. We proved that our scheme provides **online privacy** under the repeatation of nonce and **full privacy** in nonce-respecting scenario.

Various Nonce Misuse Resistant AE Schemes (for example, the deterministic AE Schemes like SIV [20], BTM [11], HBS [10]) doesn't use any nonce. Instead it uses a derived IV using the message and the associated data, which ensures that IV is distinct for each different associated data-message tuples. A limitation of such constructions is that these constructions are two pass, meaning they have to process message twice - once for the authentication and again for encryption making it less efficient. Having Encrypt mix Encrypt type layered design and using lightweight linear mixing (instead of non-linear mixing used in EME), makes our construction single pass and hence efficient.

5.1.2 Online and Fully Pipeline Implementable

The cipher is online, allowing encryption to produce ciphertext blocks before subsequent plaintext blocks (or the plaintext length) are known, and decryption to produce plaintext blocks before subsequent ciphertext blocks (or the ciphertext length) are known. Some of the online authenticated constructions like McOE-D [7] uses TC3 [21] type of structure whose lower level has CBC type structure, which is sequential and hence can not be pipelined. Our construction ELmD has a Encrypt-mix-Encrypt type structure and processes associated data and message in identical fashion, which makes it fully parallel and pipeline implementable. Note that, in some of the existing constructions like COPA [1], during the processing of associated data, the last blockcipher input depends on the previous blockcipher outputs, making the construction sequential for one block-cipher invocation. So, complete parallelization can not be achieved. Our construction ELmD doesn't have such bottleneck and is completely parallel.

5.1.3 Resistant against Block-wise Adaptive Adversaries

We consider the scenario where an authenticated encryption scheme has to release some portion of the plaintexts before the verification succeeds. This case can occur when devices have limited buffer to store the entire plaintext, or when the decrypted plaintext may be needed to be processed quickly due to real time requirements. This raises some attacks on popular constructions [12]. We consider blockwise adaptive adversaries in this scenario, which have similar advantages such as privacy and authenticity, however the adversaries would have access of partial decryption oracles for authenticity security. To resist against such attacks, intermediate tags can be used. Our construction ELmD incorporates intermediate tags efficiently and provides security against Block-wise adaptive adversaries, if required.

5.1.4 Provision for Skipping Intermediate Tags during Decryption

ELmD generates intermediate tags in a such a manner that during decryption, the plaintext computation is independent of the intermediate tag computations. Hence, if intermediate verifications are not required, the extra computations required for verifying the intermediate tags, can be skipped. Note that, Sponge duplex [3], is another authenticated encryption that incorporates intermediate tags but doesn't have this advantage.

5.1.5 Robustness

In this subsection, we discuss about the robustness of ELmD.

Empty associated data.

ELmD works perfectly even if associated data is empty. In that case, the initial value is computed using public message number only and then message processing is done as earlier.

Using the scheme as tweakable encryption scheme.

For complete final block messages, ELmD acts as a length-preserving (tweakable) encryption scheme (without message/data integrity), with IV being the tweak, if we set associated data as empty and return the ciphertext without the tag. For incomplete final block messages, some standard techniques like [16] can be used to achieve this.

Using the scheme as IV based Stream-cipher.

We can use ELmD as IV-based stream-cipher by setting the message blocks zero as long as required. The ciphertexts are used as the stream.

Using the scheme as MAC only.

ELmD acts as MAC only if we put the message in the associated data part, set message part as empty and checksum to 0. The tag is the desired output.

Integrity of associated data.

Considering message as empty and assigning the checksum block to 0^n, ELmD provides integrity of associated data.

Multiple Message Encryption under same Associated Data.

To encrypt multiple messages under same associated data, one can have separate modules for AD processing and message-ciphertext processing. To ensure that AD processing module does not need any inverse, we define L to be $E_K(E_K(0))$ (instead of something like $E_K(E_K^{-1}(0) + const)$) when 6-round of AES is used.

5.2 Justification for Recommended Parameter Sets.

In this section, we provide the details that why we have chosen the external parameters and their recommended values.

We have AES round rd as a parameter. Usual convension is to use full 10 rounds of AES encryption because of it's security gurantee. So choosing the round parameter as 10 is obvious. We also observe that, for some applications, even with much lesser rounds of encryption or decryption is good enough to

provide the desired security. In the upper layer encryption, we do not require much randomness, we just need to ensure that collision probability is low and the desired randomness is required through the overall upper and lower layer encryption. As AES gives good differential probability after 6 rounds and enough randomness with 12 ($= 6 + 6$) rounds, we believe that choosing 6 rounds in both the layers will provide the desired security. Hence, we opt for the round parameter 6 as one of our recommended choice.

If implemented in a low-end device, then to resist against block-wise adversaries, ELmD must use intermediate tags and set the value of t in between 1 to 127. We recommend to generate tags after the maximum possible gap i.e. $\mathsf{t} = 127$. But if the buffer is less than 127, then we set t as the size of the buffer. Applications that don't have limited buffer problem and during decryption, release message only after the final tag verification, can use any values of t from it's range. If the application demands quick rejection of invalid ciphertexts then it sets t to an appropriate nonzero value. For example, suppose an application want to verify the ciphertext after each 256 blocks. Then it sets $\mathsf{t} = 256$. On the otherhand, if an application doesn't require any quick rejection, then it sets $\mathsf{t} = 0$ and skip the extra operations, required to compute the intermediate tags.

Conventionally, Authenticated Encryptions have fixed tag sizes inorder to minimize ciphertext expansion, but we keep the option of having flexible (i.e. $\mathsf{f} = 0$) tag size such that the tagged ciphertext is multiple of block size because having flexible tag lengths, ELmD has some advantages during decryption. Consider the decryption with fixed tag lengths. As the final block of the tagged ciphertext may not be full, one can not use decryption to get the checksum block and verify in the upper level, having full pipelining. Rather, the verification has to be done in the lower level by first obtaining the plaintext, then calculating the checksum and generating tag in forward ELmD encryption and then verifying the tag. Since the checksum value has to pass through a block-cipher encryption and then a decryption and these can not be pipelined with any previous blockcipher encryption and decryption (as the output of previous block cipher invocations are needed to calculate the checksum) - it requiring extra 2.rd-round clock cycles. To avoid these extra clock-cycles, we can use flexible tags and having the final tagged ciphertext block complete, we can compute plaintext along with the final checksum block in a fully pipelined manner and verify in the upper level.

5.3 Comparative Study with AES-GCM

ELmD has the following advantages over AES-GCM :

- AES-GCM needed to ensure distinct nonce for each invocation. Repeatation of nonce doesn't provide any security for AES-GCM. Our construction ELmD is a Nonce Misuse Resistant Authenticated cipher and provides online security when nonce is repeated.

- AES-GCM does not provide security against blockwise adaptive adversaries. It leaks partial informations during the decryption of invalid ciphertexts, when implemented in low end device. As ELmD incorporates intermediate tags, it resist against such attacks.

- AES-GCM requires one block cipher call plus one field multiplication operation per message block and one field multiplication operation per associated data blocks, whereas our construction requires two block cipher call per message block and one block cipher call per associated data block. Clearly AES-GCM has additional implementation cost, in particular with hardware as it has to implement both AES blockcipher and field multiplication. Moreover, on the latest Intel CPUs, one call of AES is faster than one multiplication over the finite field $GF(2^{128})$. Hence, ELmD has much better performance in software also.

On the other hand, AES-GCM has following advantages over ELmD :

- AES-GCM uses only the forward direction of the blockcipher is used in the mode where as ELmD uses both forward as well as backward direction of the blockcipher. This reduces chip area in AES-GCM when implemented in hardware. But overall, ELmD require less chip area compared to AES-GCM as ELmD doesn't need to implement field multiplications, which takes a good amount of area.

- On decryption, the authenticity of the associated data can be verified independently from the recovery of the plaintext in AES-GCM. So, the blockcipher calls required during decryption, are not needed. Our construction doesn't have this facility as the tag is computed using the checksum of the plaintexts. Note that, this advantage for AES GCM, has the following drawback as well : If the verification is done independently and the verification succeeds, then to recover the plaintext, again the ciphertexts has be read, making the construction two-pass.

Chapter 6

Design Rationale

The main goal of the cipher is to be efficient, provide high performance and able to perform well in low end devices. For efficiency, we want our the cipher to be one pass, nonce misuse resistant. To obtain high performance, we want our cipher efficient as well as fully pipeline implementable. To perform well in low end devices, we require that our cipher to be secure against blockwise adaptive adversaries.

We know that, Encrypt Mix Encrypt or EME [9] is a block-cipher mode of operation, that turns a block cipher into a tweakable enciphering scheme. The mode is parallelizable, but as serial-efficient as the non-parallelizable mode CMC [8]. EME algorithm entails two layers of ECB encryption and a non-linear mixing in between. In the non-linear mixing, the blockcipher is again used. EME is proved to provide sprp [14] security in the standard, provable security model assuming that the underling block cipher is sprp secure. We observed that replacing non-linear mixing by an efficient online linear mixing actually helps to have faster and parallel implementation of the construction and gives online prp [14] security, which is good enough to construct an authenticated encryption scheme, if tags are properly generated.

We use online linear mixing to make the construction online and efficiently implemented in low end device or any platform with limited memory. Moreover, we choose ρ as our online linear mixing, as it is lightweight, efficiently computable and at the same time, intermediate tags can be generated very efficiently. Note that, we could have used more lightweight mixing like simple xor operation in the linear mixing, but then generating intermediate tags wouldn't have been efficient.

We have used the doubling method [19] to generate the masks because this method produces many different values of the mask from just one secret value L and it doesn't require any precomputations and memory. As, we want full randomness of L, we compute $L = E_K(0)$ when 10 rounds of AES is being used

and $L = E_K(E_K(0))$ when 6 rounds of AES is being used.

Consider an authenticated encryption construction with intermediate tag verification is done after each t blocks. Before each intermediate tag verification, the device where it is implemented, has to hold the t blocks of message as well as some intermediate computations, required for the verification. So, the device required to have appropriate amount of buffer. As intermediate tags are used in low end devices, one needs ensure that the buffer size is minimized. The choice of ρ helps ELmD to verify after one layer which makes the verification is faster and it requires to store only rd_2 blocks of intermediate computation for the next rd_2 subsequent ciphertext.

We replace the second layer encryption by decryption which makes authenticated encryption and verified decryption almost identical. This helps us to minimize the combined implementd area when both encryption and decryption is implemented in the same device. Nowadays in all application environment, both encryption and decryption of blockciphers are needed to be implemented and hence we can share the architectures to have a compact combined hardware implementation of it.

The designers have not hidden any weaknesses in this cipher. The recommended parameter set $\mathsf{ELmD}_{10,\mathsf{t},\mathsf{f}}$ rules out any weaknesses outside AES. The only possible weakness in case of $\mathsf{ELmD}_{6,\mathsf{t},\mathsf{f}}$ is the use of 6 rounds AES encryption in both the layers, which we have clearly mentioned in chapter 2 and 3.

Chapter 7

Intellectual Property

This family of authenticated ciphers or any parts of the cipher, do not have an kind of patents. Existance of any kind of patent on any parts of the cipher is not known to the submitters. If any of this information changes, the submitters will promptly (and within at most one month) announce these changes on the crypto-competitions mailing list.

Chapter 8

Consent

The submitters hereby consent to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitters understand that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analyses that led to the selection of the algorithm. The submitters understand that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitters understand that if they disagree with published analyses then they are expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

Bibliography

[1] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar W. Tischhauser and Kan Yasuda, *Parallelizable (authenticated) online ciphers* **8269** (2013), 424–443, *Asiacrypt*, Lecture Notes in Computer Science, 2013. Citations in this document: §5.1.2.

[2] A.Biryukov, *The Boomerang Attack on 5 and 6-Round Reduced AES* **3373** (2005), 11–15, *Advanced Encryption Standard - AES*, Lecture Notes in Computer Science, 2005.

[3] G.Bertoni, J.Daeman, M.Peeters and G.V.Assche, *Duplexing the Sponge : Single Pass Authenticated Encryption and Other Applications* **7118** (2011), 320–337, *Selected Areas in Cryptography*, Lecture Notes in Computer Science, 2011. Citations in this document: §5.1.4.

[4] Joan Daemen and Vincent Rijmen, *AES Submission Document on Rijndael* (1998). URL: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf. Citations in this document: §1.4.1.

[5] Patrick Derbez, Pierre-Alain Fouque, Jeremy Jean, *Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting* **7881** (2013), 371–387, *Eurocrypt 2013*, Lecture Notes in Computer Science, 2013.

[6] N. Ferguson, J.Kelsey, S.Lucks, B.Schneier, M.Stay, D.Wagner, D.Whiting, *Improved Cryptanalysis of Rijndael* **1978** (2000), 213–230, *Fast Software Encryption*, Lecture Notes in Computer Science, 2000.

[7] E. Fleischmann, C. Forler, S. Lucks, *McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes* **7549** (2012), 196–215, *Fast Software Encryption*, Lecture Notes in Computer Science, 2012. Citations in this document: §5.1.2.

[8] S. Halevi and P. Rogaway., *A Tweakable Enciphering Mode.* **2729** (2003), *CRYPTO*, Lecture Notes in Computer Science, 2003. Citations in this document: §6.

[9] Shai Halevi and Phillip Rogaway, *A parallelizable enciphering mode* **2964** (2004), 292–304, *CTRSA*, Lecture Notes in Computer Science, 2004. Citations in this document: §6.

[10] Tetsu Iwata and Kan Yasuda, *HBS : A Single-Key mode of Operation for Deterministic Authenticated Encryption* **5665** (2009), 394–415, *Fast Software Encryption*, Lecture Notes in Computer Science, 2009. Citations in this document: §5.1.1.

[11] Tetsu Iwata and Kan Yasuda, *BTM : A Single-Key, Inverse-Cipher-Free Mode for Deterministic Authenticated Encryption* **5867** (2009), 313–330, *Selected Areas in Cryptography*, Lecture Notes in Computer Science, 2009. Citations in this document: §5.1.1.

[12] Antoine Joux, Gwenlle Martinet and Fredric Valette, *Blockwise-Adaptive Attackers: Revisiting the (In)Security of Some Provably Secure Encryption Models: CBC, GEM, IACBC* **2442** (2002), 17–30, *CRYPTO*, Lecture Notes in Computer Science, 2002. Citations in this document: §5.1.3.

[13] T. Krovetz and P. Rogaway, *The Software Performance of Authenticated-Encryption Modes* **6733** (2011), 306–327, *Fast Software Encryption*, Lecture Notes in Computer Science, 2011. Citations in this document: §5.1.1.

[14] Michael Luby, Charles Rackoff, *How to construct pseudorandom permutations from pseudorandom functions*, SIAM Journal of Computing (1988), 373–386. Citations in this document: §6, §6.

[15] D. McGrew, J. Viega, *The Galois/Counter Mode of Operation (GCM), Submission to NIST Modes of Operation Process, January 2* (2004). URL: http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf. Citations in this document: §5.1.1.

[16] Mridul Nandi, *A Generic Method to Extend Message Space of a Strong Pseudorandom Permutation.*, Computación y Sistemas (2009). Citations in this document: §5.1.5.

[17] Jacques Patarin, *The "Coefficients H" Technique* **5381** (2009), 328–345, *Selected Areas in Cryptography*, Lecture Notes in Computer Science, 2009. Citations in this document: §4.2.1.

[18] P. Rogaway, *Nonce-based symmetric encryption* **3017** (2004), 348–359, *FSE 2004*, Lecture Notes in Computer Science, 2004. Citations in this document: §5.1.1.

[19] P. Rogaway, *Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC.* **3329** (2004), 16–31, *Asiacrypt 2004*, Lecture Notes in Computer Science, 2004. Citations in this document: §1.4.2, §4.2.1, §6.

[20] P. Rogaway and T. Shrimpton, *Deterministic Authenticated-Encryption : A Provable-Security Treatment of the Key-Wrap Problem* **4004** (2006), 373–390, *Advances in Cryptology - Eurocrypt*, Lecture Notes in Computer Science, 2006. Citations in this document: §5.1.1.

[21] Phillip Rogaway and Haibin Zhang, *Online Ciphers from Tweakable Block-ciphers* (2011), 237–249, *CT-RSA*, 2011. Citations in this document: §5.1.2.