iFeed[AES] v1

Designers and Submitters:

Liting Zhang[1] Wenling Wu[1] Han Sui[1] Peng Wang[2]

Trusted Computing and Information Assurance Laboratory
State Key Laboratory of Computer Science
Institute of Software[1]
Data Assurance and Communication Security Research Center[2]
Chinese Academy of Sciences[1,2]

liting.zhang@hotmail.com

March 15, 2014

Contents

Chapter 1

Introduction

iFeed is a mode of operation for authenticated encryption, that can use blockciphers or compression functions as its underlying primitives.

In this submission, we specify iFeed[AES] that uses blockcipher AES-128 as its underlying primitive.

By a random key K and a public message number, which should never be repeated, iFeed[AES] can provide both integrity and confidentiality protections for plaintext PT. Since associated data AD needs only integrity protection, we introduce a PMAC-like mode Mac[AES] (using the same K).

Based on pseudorandomness of AES-128, iFeed[AES] is proved to be secure up to a birthday bound with q (the number of queries) and σ (the total block length of queried messages).

iFeed[AES] is rate-1, in the sense that it needs only one forward invocation on AES-128 to process each block in AD and PT on average. It is also one-pass in the sense that in processing PT its mechanisms for integrity and confidentiality protections are closely combined that by a single pass through PT we get both protections for it.

Furthermore, iFeed[AES] has many other good features, such as one-key, on-line, inverse-free, parallel encryption, ciphertext length preserving, endian-less, intermediate tag supporting and etc. All these make it convenient to use in many practical applications, e.g. encrypting Internet packets.

Chapter 2

Specification

2.1 Symbols

Symbols	Descriptions
$\{0,1\}^n$	the set containing all n-bit strings
$\{0,1\}^{a\sim b}$	the set containing all the strings of bit length between a and b
$\{0,1\}^{\leq l}$	the set containing all the strings of bit length no longer than l
$X\|\|Y$	the concatenation of strings X and Y
0^n	the string of n successive "0"
$\|X\|$	the bit length of string X
$\|X\|_n$	the block length of string X, equals to the least number no smaller than $\|X\|/n$, where n is the block size
$K \xleftarrow{\$} \mathcal{K}$	select an element K from set $\mathcal{K}$ uniformly at random
$X \cdot 2$	128-bit value X multiplied by 2 over GF(2^{128}) $X \cdot 2 = X << 1$ if the left most bit of X is 0 otherwise $X \cdot 2 = X << 1 \oplus (0^{120}\|\|10000111)$
AES	the advanced encryption standard, and we specially refer it to AES-128 throughout this submission
AES(K, X)	encrypting 128-bit string X by blockcipher AES-128 under a 128-bit key K
Pad(A)	$A\|\|10^{n-1-\|A\| \mod n}$ if $\|A\| < n$, or A if $\|A\| = n$
Partition(A)	break string A into a successive parts $A_1 A_2 \cdots A_a$ that $\|A_i\| = n$ $(i = 1, 2, \cdots, a-1)$ and $\|A_a\| \leq n$
Cut(A, b)	the string A is cut into two parts, with the left part of bit length $b \leq \|A\|$
Truncate(A, b)	the left part of **Cut**(A, b)

2.2 Parameters

Parameter Size (in bytes)	iFeed[AES]
Key	16
Secret Message Number	0
Public Message Number	[1,15]
Tag	[2,16]

2.3 Recommended Parameter Sets

Parameter Size (in bytes)	1st recommended	2nd recommended
Key	16	16
Secret Message Number	0	0
Public Message Number	12	13
Tag	16	16

2.4 Authenticated Encryption

Although in the following iFeed[AES] is defined on bit strings, the bit lengths of inputs (key, public message number, associated data, plaintext) and outputs (ciphertext, tag) shall all be multiples of 8, so that these values are byte strings.

In authenticated encryption, the inputs to iFeed[AES] include a 128-bit secret key K, a public message number PMN of bit length between 1 and 127, a plaintext PT of bit length between 0 and $|\text{PT}|_{\max}$, and an associated data AD of bit length between 0 and $|\text{AD}|_{\max}$, with $|\text{PT}|_{\max}+|\text{AD}|_{\max} \leq 2^{71}-512$.

The outputs from iFeed[AES] include a ciphertext CT and a tag T. CT is the result of encrypting PT. It can be used to recover PT, but without integrity checking. Its length equals to the length of PT. T is a τ-bit value to be used to check the validity of CT, with $32 \leq \tau \leq 128$.

The public message number PMN should be a nonce, a value that never repeated in different encryptions. If unfortunately PMN is reused by some wrong operations, the security of iFeed[AES] would be reduced. See details in Chapter 3.

iFeed[AES] takes K as the AES-128 key. By calculating a secret value $Z_0 = \text{AES}(K, 0^{128})$, we can pre-compute the secret masks Z_i by the multiplication "$\cdot 2$" over $\text{GF}(2^{128})$. The detailed authenticated encryption procedure is described in the next page, and illustrated in Figure 2.1.

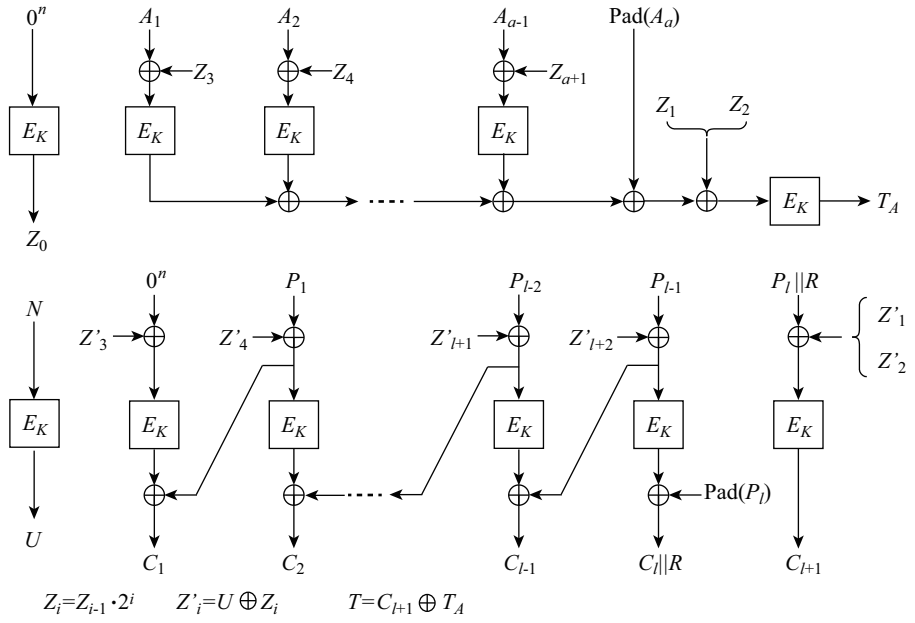

Figure 2.1: Illustration of iFeed[AES].AEnc, with E_K denoted as AES-128 under K.

iFeed[AES].AEnc(K, PMN, AD, PT)

Input: $K \xleftarrow{\$} \{0,1\}^{128}$, PMN $\in \{0,1\}^{1\sim 127}$, AD $\in \{0,1\}^{\leq |\text{AD}|_{\max}}$, PT $\in \{0,1\}^{\leq |\text{PT}|_{\max}}$
Output: CT $\in \{0,1\}^{|\text{PT}|}$, $T \in \{0,1\}^{\tau}$

101. $Z_0 = \text{AES}(K, 0^{128})$
102. **for** $i = 1$ **to** $\max\{|\text{AD}|_{128}, |\text{PT}|_{128}\} + 2$ **do**
103. $\quad Z_i = Z_{i-1} \cdot 2$
104. $N = \text{PMN}||10^{127-|\text{PMN}|}$
105. $U = \text{AES}(K, N)$
106. **if** $|\text{AD}| = 0$ **then** $T_A = 0^{128}$ **else** $T_A = \text{Mac[AES]}(K, Z, \text{AD})$
107. $C_1 C_2 \cdots C_{l+1} = \text{Enc[AES]}(K, Z, U, \text{PT})$
108. $T = \textbf{Truncate}(T_A \oplus C_{l+1}, \tau)$
109. **return** CT $= C_1 C_2 \cdots C_l$, T

iFeed[AES] uses two subroutines Mac[AES] and Enc[AES]. The former is PMAC-like and is used to authenticate AD, with $Z = Z_1 Z_2 \cdots Z_{|\text{AD}|_{128}+1}$. The latter deals with PT for both integrity and confidentiality protections, with $Z = Z_1 Z_2 \cdots Z_{|\text{PT}|_{128}+2}$. Since AES has 128-bit block size, we have $n = 128$ for **Partition** and **Pad** funtions.

Mac[AES](K, Z, AD)	
Input: $K \xleftarrow{\$} \{0,1\}^{128}$, $Z \in \{0,1\}^{128\times\lvert\mathrm{AD}\rvert_{128}+128}$, $\mathrm{AD} \in \{0,1\}^{1\sim\lvert\mathrm{AD}\rvert_{\max}}$	
Output: $T_A \in \{0,1\}^{128}$	
111.	$A_1A_2\cdots A_a = \mathbf{Partition}(\mathrm{AD})$
112.	$\mathrm{SumB} = 0^{128}$
113.	**for** $i = 1$ **to** $a-1$ **do**
114.	$\quad B_i = \mathrm{AES}(K, A_i \oplus Z_{i+2})$
115.	$\quad \mathrm{SumB} = \mathrm{SumB} \oplus B_i$
116.	**if** $\lvert A_a\rvert < 128$ **then** $T_A = \mathrm{AES}(K, \mathrm{SumB} \oplus Z_1 \oplus \mathbf{Pad}(A_a))$
117.	**else** $T_A = \mathrm{AES}(K, \mathrm{SumB} \oplus Z_2 \oplus A_a)$
118.	**return** T_A

Enc[AES](K, Z, U, PT)	
Input: $K \xleftarrow{\$} \{0,1\}^{128}$, $Z \in \{0,1\}^{128\times\lvert\mathrm{PT}\rvert_{128}+256}$, $U \in \{0,1\}^{128}$, $\mathrm{PT} \in \{0,1\}^{\leq\lvert\mathrm{PT}\rvert_{\max}}$	
Output: $C_1C_2\cdots C_{l+1} \in \{0,1\}^{\lvert\mathrm{PT}\rvert+128}$	
121.	$P_1P_2\cdots P_l = \mathbf{Partition}(\mathrm{PT})$
122.	$P_0 = 0^{128}$
123.	**for** $i = 1$ **to** $l-1$ **do**
124.	$\quad C_i = \mathrm{AES}(K, P_{i-1} \oplus Z_{i+2} \oplus U) \oplus P_i \oplus Z_{i+3} \oplus U$
125.	$C_l = \mathrm{AES}(K, P_{l-1} \oplus Z_{l+2} \oplus U) \oplus \mathbf{Pad}(P_l)$
126.	**if** $\lvert P_l\rvert < 128$ **then** $C_lR = \mathbf{Cut}(C_l, \lvert P_l\rvert)$
127.	$\quad\quad C_{l+1} = \mathrm{AES}(K, (P_l\|\|R) \oplus Z_1 \oplus U)$
128.	**else** $C_{l+1} = \mathrm{AES}(K, P_l \oplus Z_2 \oplus U)$
129.	**return** $C_1C_2\cdots C_{l+1}$

2.5 Authenticated Decryption

The authenticated decryption procedure naturally inverses the authenticated encryption procedure. Its inputs include a 128-bit secret key K, a public message number PMN of bit length between 1 and 127, a ciphertext CT of bit length between 0 and $|\mathrm{PT}|_{\max}$, an associated data AD of bit length between 0 and $|\mathrm{AD}|_{\max}$, plus a tag T' of τ bits. Notice that we restrict $|\mathrm{PT}|_{\max} + |\mathrm{AD}|_{\max} \leq 2^{71} - 512$.

Its output is either a plaintext PT of length $|\mathrm{CT}|$ or a failure symbol $\perp$, depending on the tag checking of CT is valid of not.

The detailed authenticated decryption procedure is described in the next page and illustrated in Figure 2.2.

iFeed[AES].ADec$(K, \text{PMN}, \text{AD}, \text{CT}, T')$

Input: $K \in \{0,1\}^{128}$, $\text{PMN} \in \{0,1\}^{1\sim 127}$, $\text{AD} \in \{0,1\}^{\leq |\text{AD}|_{\max}}$, $\text{CT} \in \{0,1\}^{\leq |\text{PT}|_{\max}}$, $T' \in \{0,1\}^{\tau}$

Output: $\text{PT} \in \{0,1\}^{|\text{CT}|}$ or $\perp$

201. $Z_0 = \text{AES}(K, 0^{128})$
202. **for** $i = 1$ **to** $\max\{|\text{AD}|_{128}, |\text{CT}|_{128}\} + 2$ **do**
203. $\quad Z_i = Z_{i-1} \cdot 2$
204. $N = \text{PMN}||10^{127-|\text{PMN}|}$
205. $U = \text{AES}(K, N)$
206. **if** $|\text{AD}| = 0$ **then** $T'_A = 0^{128}$ **else** $T'_A = \text{Mac[AES]}(K, Z, \text{AD})$
207. $P_1 P_2 \cdots P_l, C'_{l+1} = \text{Dec[AES]}(K, Z, U, \text{CT})$
208. **if** $T' \neq \textbf{Truncate}(T'_A \oplus C'_{l+1}, \tau)$ **then return** $\perp$
209. **else return** $\text{CT} = P_1 P_2 \cdots P_l$

iFeed[AES].ADec also uses two subroutines. Mac[AES] has been introduced previously, and Dec[AES] is used to decrypt CT, with $Z = Z_1 Z_2 \cdots Z_{|\text{CT}|_{128}+2}$, as follows.

Dec[AES](K, Z, U, CT)

Input: $K \in \{0,1\}^{128}$, $Z \in \{0,1\}^{128 \times |\text{CT}|_{128}+256}$, $U \in \{0,1\}^{128}$, $\text{CT} \in \{0,1\}^{\leq |\text{PT}|_{\max}}$

Output: $P_1 P_2 \cdots P_l C'_{l+1} \in \{0,1\}^{|\text{CT}|+128}$

211. $C_1 C_2 \cdots C_l = \textbf{Partition}(\text{CT})$
212. $P_0 = 0^{128}$
213. **for** $i = 1$ **to** $l - 1$ **do**
214. $\quad P_i = \text{AES}(K, P_{i-1} \oplus Z_{i+2} \oplus U) \oplus C_i \oplus Z_{i+3} \oplus U$
215. $P_l = \text{AES}(K, P_{l-1} \oplus Z_{l+2} \oplus U) \oplus \textbf{Pad}(C_l)$
216. **if** $|C_l| < 128$ **then** $P_l R = \textbf{Cut}(P_l, |C_l|)$
217. $\quad\quad C'_{l+1} = \text{AES}(K, (P_l||R) \oplus Z_1 \oplus U)$
218. **else** $C'_{l+1} = \text{AES}(K, P_l \oplus Z_2 \oplus U)$
219. **return** $P_1 P_2 \cdots P_l, C'_{l+1}$

2.6 Test Vectors

Inputs	#bytes	Values (in hex)
Key	16	`0123456789abcdeffedcba9876543210`
Public Message Number	13	`6946656564204145204d6f6465`
Associated Data	26	`6162636465666768696a6b6c6d6e6f70` `7172737475767778797a`
Plaintext	36	`4142434445464748494a4b4c4d4e4f505152` `535455565758595a30313233343536373839`
Ciphertext	36	`9f7aecdd989cb5eb26490e69f7d06bf4cfcc` `10b85055f642a1ad15ea4b3f3c6c3efee234`
Tag	16	`ba6239be4e2c687c58b807d6a508c073`

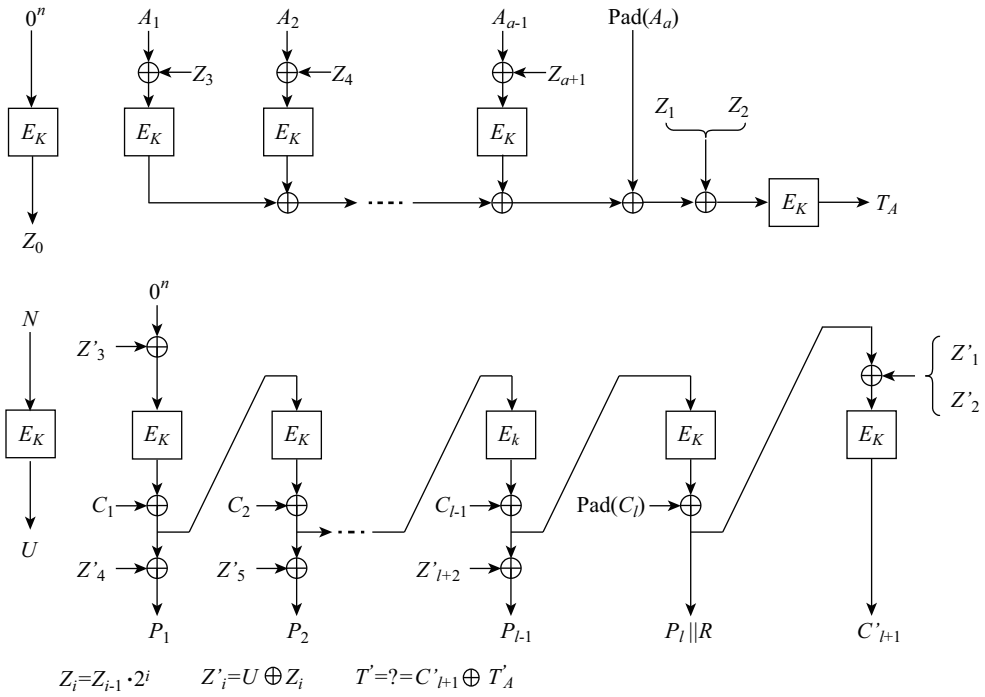

Figure 2.2: Illustration of iFeed[AES].ADec, with E_K denoted as AES-128 under K.

Chapter 3

Security Goals

Goals in nonce-respecting settings	iFeed[AES] bits of security
confidentiality for the plaintext	128
pseudorandomness of the ciphertext	64
integrity for the plaintext	64
integrity for the associated data	64
integrity for the public message number	64

There is no secret message number in iFeed[AES]. Public message number should be used as a nonce, never repeated in any two queries to iFeed[AES]. AEnc.

1. The confidentiality of plaintext is 128-bit secure means the expected number of key guesses to find the secret key is no less than 2^{128}. This in fact depends on the expectation that to guess AES-128 key we need 2^{128} attempts.

2. The pseudorandomness of ciphertext is 64-bit secure means in distinguishing ciphertext from random strings, the expected number of total blocks of queried plaintext and associated data is no less than 2^{64}.

3. The integrity for plaintext, associated data and public message number is 64-bit secure means in forging against iFeed[AES] with a new tuple (PMN$'$, AD$'$, CT$'$, T'), the number of total blocks of queried plaintext, ciphertext and associated data in online attempts is no less than 2^{64}. Note here we let $\tau = 128$.

Any successful forgery, plaintext recovery, or successful key guess should be assumed to completely compromise integrity or confidentiality of all messages. The table assumes that the legitimate key holder does not approach $2^{64} - 4$ blocks of plaintext and associated data under a single key.

iFeed[AES] does not promise any integrity or confidentiality if the legitimate key holder uses the same public message number to encrypt two

different (plaintext, associated data) pairs under the same key. If public message number is reused by some careless operation, the security of iFeed[AES] would be reduced. Specifically,

1. the confidentiality of plaintext is still 128-bit secure, but attackers can find plaintext differentials from ciphertext differentials.

2. the pseudorandomness of ciphertext would completely lose, because attackers can distinguish ciphertext from random strings with only two forward queries.

3. the integrity for plaintext, associated data and public message number, would completely lose, i.e. attackers can make forgeries with a tuple (PMN$'$, AD$'$, CT$'$, T'), where CT$'$ is new, by only two forward queries. However, note that Mac[AES] itself is still 64-bit secure, i.e. it is unforgeable against any new AD.

Security in nonce-reuse settings	iFeed[AES] bits of security
confidentiality for the plaintext	128
pseudorandomness of the ciphertext	$O(1)$
integrity for the plaintext	$O(1)$
integrity for the associated data	$O(1)$
integrity for the public message number	$O(1)$

Chapter 4

Security Analysis

4.1 Definitions

PRP for Blockciphers Let $\mathrm{Perm}(n)$ be the set of all permutations over $\{0,1\}^n$, $P(\cdot)$ is said to be a random permutation if $P(\cdot) \xleftarrow{\$} \mathrm{Perm}(n)$. For a blockcipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, its PRP security is evaluated as

$$\mathbf{Adv}_E^{\mathrm{PRP}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} |\Pr[K \xleftarrow{\$} \{0,1\}^k : \mathcal{A}^{E_K(\cdot)} = 1] - \Pr[\mathcal{A}^{P(\cdot)} = 1]|,$$

where the first probability is taken over $K \xleftarrow{\$} \{0,1\}^k$ and the random choices of $\mathcal{A}$, and the second probability is taken over $P(\cdot) \xleftarrow{\$} \mathrm{Perm}(n)$ and the random choices of $\mathcal{A}$. By $\mathbf{Adv}_E^{\mathrm{PRP}}(t,q)$ we mean the maximum advantage for all adversaries over computation time at most t and number of queries at most q.

Privacy for AE schemes We follow the definitions given by Rogaway [1, 2]. A chosen plaintext adversary $\mathcal{A}$ is given an encryption oracle, which is either iFeed[E].AEnc or a random-bit oracle \$ with equal probability. $\mathcal{A}$ is asked to be nonce-respecting. That is, it should never repeat PMN in queries to its encryption oracle. For any input (PMN, AD, PT) within iFeed domain, iFeed[E] returns $(\mathrm{CT}, T) = \mathrm{iFeed}[E_K].\mathrm{AEnc}(\mathrm{PMN}, \mathrm{AD}, \mathrm{PT})$, and the random-bit oracle always returns random bits of length $|\mathrm{CT}| + |T|$. Let

$$\mathbf{Adv}_{\mathrm{iFeed}[E]}^{\mathrm{Priv}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} |\Pr[K \xleftarrow{\$} \{0,1\}^k : \mathcal{A}^{\mathrm{iFeed}[E_K](\cdot)} = 1] - \Pr[\mathcal{A}^{\$(\cdot)} = 1]|,$$

with the first probability is taken over the random choices of K and $\mathcal{A}$, and the second probability is taken over the random-bit oracle \$ and the random choices of $\mathcal{A}$. By $\mathbf{Adv}_{\mathrm{iFeed}[E]}^{\mathrm{Priv}}(t,q,\sigma)$ we mean the maximum advantage for all adversaries over computation time at most t, number of queries at most q whose total length at most σ blocks.

Authenticity for AE schemes A chosen ciphertext adversary $\mathcal{A}$ is given two oracles iFeed[E].AEnc and iFeed[E].ADec, and it can make forward queries to iFeed[E].AEnc and backward queries to iFeed[E].ADec, where the latter behaves as for any given (PMN, AD, CT, T) returning either PT= iFeed[E_K].ADec(PMN, AD, CT, T) or $\perp$ = iFeed[E_K].ADec(PMN, AD, CT, T). Let

$$\mathbf{Adv}^{\mathrm{Auth}}_{\mathrm{iFeed}[E]}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \Pr[K \xleftarrow{\$} \{0,1\}^k : \mathcal{A}^{\mathrm{iFeed}[E_K].\mathrm{AEnc}(\cdot),\mathrm{iFeed}[E_K].\mathrm{ADec}(\cdot)}\mathrm{forges}],$$

with the probability taken over the random choices of K and $\mathcal{A}$. By $\mathbf{Adv}^{\mathrm{Auth}}_{\mathrm{iFeed}[E]}(t, q, q_v, \sigma, \sigma_v)$ we mean the maximum advantage for all adversaries over computation time at most t, number of forward queries at most q whose total length at most σ blocks, and number of backward queries at most q_v whose total length at most σ_v blocks.

Here adversary $\mathcal{A}$ forges means iFeed[E].ADec returns a plaintext PT= iFeed[E_K].ADec (PMN, AD, CT, T) other than $\perp$, conditioned on (CT, T) was never outputted by iFeed[E_K].AEnc in corresponding to (PMN, AD) and some plaintext.

Note that $\mathcal{A}$ should also be nonce-respecting in queries to iFeed[E].AEnc. However, there is no restrictions for $\mathcal{A}$ to either repeating PMN in queries to iFeed[E].ADec or using the same PMN in both queries to iFeed[E].AEnc and iFeed[E].ADec. Without loss of generality, we assume $\mathcal{A}$ never made trivial queries. That is, $\mathcal{A}$ never repeats queries to iFeed[E].AEnc and iFeed[E].ADec, and $\mathcal{A}$ never queries (PMN, AD, CT, T) to iFeed[E].ADec if it has obtained (CT, T)=iFeed[E_K].Enc(PMN, AD, PT) before, and vise visa.

4.2 Main Results

If AES-128 is secure as a PRP, we get iFeed[AES] is secure as an authenticated encryption scheme against nonce-respecting adversaries. Specifically, we have

Theorem 1. *For any adversary $\mathcal{A}$ making at most q forward queries of total (including both associated data and plaintext) block length no more than σ, and at most q_v backward queries of total (including both associated data and ciphertext) block length no more than σ_v, we have*

$$\begin{cases} \mathbf{Adv}^{\mathrm{Priv}}_{\mathrm{iFeed[AES]}}(t_1, q, \sigma) \le \mathbf{Adv}^{\mathrm{PRP}}_{\mathrm{AES}}(t_2, \sigma_1) + \frac{(\sigma+2q)^2+2.5(\sigma+2q)}{2^{128}}, \\ \mathbf{Adv}^{\mathrm{Auth}}_{\mathrm{iFeed[AES]}}(t_3, q, q_v, \sigma, \sigma_v) \le \mathbf{Adv}^{\mathrm{PRP}}_{\mathrm{AES}}(t_4, \sigma_2) + \frac{(\sigma+\sigma_v+2q+2q_v+1)^2}{2^{128}} + \frac{q_v}{2^\tau}, \end{cases}$$

with $t_2 \approx t_1$, $t_4 \approx t_3$, $\sigma_1 = \sigma + q + 1$ and $\sigma_2 = \sigma + \sigma_v + q + q_v + 1$.

4.3 Analysis

In both privacy and authenticity aspects of iFeed[AES], security relies on the pseudorandomness of AES. Specifically, the privacy of iFeed[AES] is guaranteed by making AES inputs pairwise distinct, and thus iFeed construction outputs pseudorandom values seemingly independent of each other. This is achieved by the none repetition of public message numbers and keeping secrecy of inputs to AES. In any two queries (PMN^i, AD^i, PT^i) and (PMN^j, AD^j, PT^j) to iFeed[AES].AEnc, the inputs to AES are differed by two random values N^i and N^j derived from PMN^i and PMN^j respectively. Within any single query (PMN^i, AD^i, PT^i), the inputs to AES are differed by different Z'_l (derived from $Z_0 = \text{AES}(K, 0^n)$ and N^i) XORed to plaintext blocks. Z'_1 and Z'_2 are specially arranged to the end to prevent sliding attacks, and to differ plaintexts of full blocks long or not.

The authenticity of iFeed[AES] is guaranteed by the secrecy of AES inputs, and the advantage of EMAC structure and PMAC structure [3, 4]. In a forgery with a new PMN, the inputs to AES can hardly collide with those in previously queried messages (including PT, CT, AD, PMN and 0^{128}). In a forgery with an old PMN but not its corresponding CT, the EMAC structure ensures internal states can hardly collide after the common prefix of CTs. In both above cases, no collision in AES inputs implies random tags in the forgery that adversaries can hardly predict. A third case is adversary forges with an old PMN but a new AD, then Mac[AES] would output random values by the advantage of PMAC structure. The remaining two cases of forgery are trivial.

Let $\text{Rand}(n, n)$ be the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$, $F(\cdot)$ is said to be a random function if $F(\cdot) \xleftarrow{\$} \text{Rand}(n, n)$. Define iFeed[$E$], iFeed[$P$] and iFeed[$F$] as iFeed modes with underlying primitives being a blockcipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, a random permutation P and a random function F respectively.

We first prove the security of iFeed[F] in the information-theoretic setting, then turn the results to iFeed[P] by PRP/PRF switching lemma [5], at last we give the results for iFeed[AES].

Proof for Privacy A nonce-respecting adversary $\mathcal{A}$ has access to an encryption oracle, which is either iFeed[F].AEnc or a random-bit oracle \$ with equal probability. Without loss of generality, we assume $\mathcal{A}$ makes exactly q queries. In the following, we check the process details of iFeed[F].AEnc on every $\mathcal{A}$'s query, and show that in most cases it outputs random bits, just like the random-bit oracle \$ does, and its probability to fail to do this is sufficiently small.

Before $\mathcal{A}$ makes its first query, we define several flags $\texttt{Bad}[i]$ ($i = 1, 2, \cdots, 9$), any one of which being `True` indicates iFeed[F].AEnc may fail to output

random strings. We also define five sets CDom, ADom, SDom, NDom and PDom, which are used to record the inputs to F when $\mathcal{A}$ querying iFeed[F].AEnc. Specifically, CDom contains only the constant 0^n, ADom contains all the inputs to F in Mac[F] except the last one, SDom contains the last input to F in Mac[F], NDom contains all the nonces from the public message numbers, and PDom contains all the inputs to F in Dec[F]. Note that these five sets cover all the inputs to F in $\mathcal{A}$ querying iFeed[F].AEnc.

According to iFeed[F].AEnc, a value $Z_0 = F(0^n)$ is secretly produced, and we further calculate $Z_j = Z_{j-1} \cdot 2$ for sufficient times as below, where L_{ADmax} (or L_{PTmax}) stands for the maximum number of blocks of all $\mathcal{A}$'s querying AD (or PT).

301.	$\texttt{Bad}[i] = \texttt{False}$ for $i = 1, 2, \cdots, 9$
302.	CDom $= \{0^n\}$; ADom, SDom, NDom, PDom =EmptySet
303.	$Z_0 = F(0^n)$
304.	**for** $j = 1$ **to** $\max\{L_{\mathrm{ADmax}}, L_{\mathrm{PTmax}}\} + 2$ **do** $Z_j = Z_{j-1} \cdot 2$

Proof. We let $\mathcal{A}$ start to query iFeed[F].AEnc, and return it with the corresponding outputs. Suppose in the previous $i-1$ queries (PMN^j, AD^j, PT^j) for $j = 1, 2, \cdots, i-1$, iFeed[F].AEnc has successfully output $i-1$ random strings (CT^j, T^j) of length $|\mathrm{CT}^j| = |\mathrm{PT}^j|$ and $|T^j| = \tau$. Now, let $\mathcal{A}$ make its i-th query $(\mathrm{PMN}^i, \mathrm{AD}^i, \mathrm{PT}^i)$, and we check the process details of iFeed[F].AEnc$(\mathrm{PMN}^i, \mathrm{AD}^i, \mathrm{PT}^i)$.

311.	$N^i = \mathrm{PMN}^i \| 10^{n-\|\mathrm{PMN}^i\|}$
312.	**if** $N^i \in \mathrm{CDom} \cup \mathrm{NDom}$ **then** reject
313.	**else if** $N^i \in \mathrm{ADom} \cup \mathrm{SDom} \cup \mathrm{PDom}$ **then** $\texttt{Bad}[1] = \texttt{True}$
314.	**else** $U^i = F(N^i)$, $\mathrm{NDom} = \mathrm{NDom} \cup \{N^i\}$

In line 312, $N^i \in$ CDom would never occur by our padding rule for PMN, and $N^i \in$ NDom shouldn't happen because $\mathcal{A}$ is required to be nonce-respecting.

For line 313, we notice that every element in sets ADom, SDom and PDom is related to the random value $Z_0 = F(0^n)$, so $\Pr[\texttt{Bad}[1] = \texttt{True}] \leq (\#\mathrm{ADom} + \#\mathrm{SDom} + \#\mathrm{PDom})/2^n \leq (\sum_{j=1}^{i-1}(L_{\mathrm{AD}}^j - 1) + (i-1) + \sum_{j=1}^{i-1}(L_{\mathrm{PT}}^j + 1))/2^n$, where L_{AD}^j and L_{PT}^j are the block lengths of AD^j and PT^j in $\mathcal{A}$'s j-th query. After line 313, we are sure N^i is a new input to F, so U^i is a random value.

321.	**if** $\|\mathrm{AD}^i\| = 0$ **then** $T_A^i = 0^n$
322.	**else if** $\exists j \in [1, i-1]$ s.t. $\mathrm{AD}^i = \mathrm{AD}^j$ **then** $T_A^i = T_A^j$
323.	**else** $T_A^i = \mathrm{Mac}[F](\mathrm{AD}^i)$
$\mathrm{Mac}[F](\mathrm{AD}^i)$ for a new AD^i	
331.	$A_1^i A_2^i \cdots A_{a_i}^i = \mathbf{Partition}(\mathrm{AD}^i)$
332.	**for** $j = 1$ **to** $a_i - 1$ **do**
333.	**if** $A_j^i \oplus Z_{j+2} \in \mathrm{CDom} \cup \mathrm{NDom} \cup \mathrm{SDom} \cup \mathrm{PDom}$ **then** $\mathtt{Bad}[2] = \mathtt{True}$.
334.	**else if** $\exists j' \neq j, i' \leq i$ s.t. $A_{j'}^{i'} \oplus Z_{j'+2} = A_j^i \oplus Z_{j+2}$ **then** $\mathtt{Bad}[3] = \mathtt{True}$
335.	**else** $B_j^i = F(A_j^i \oplus Z_{j+2})$, $\mathrm{ADom} = \mathrm{ADom} \cup \{A_j^i \oplus Z_{j+2}\}$
336.	**if** $\|A_{a_i}^i\| < n$ **then** $\mathrm{SumB}^i = \bigoplus_{j=1}^{a_i-1} B_j^i \oplus \mathbf{Pad}(A_{a_i}^i) \oplus Z_1$
337.	**else** $\mathrm{SumB}^i = \bigoplus_{j=1}^{a_i-1} B_j^i \oplus A_{a_i}^i \oplus Z_2$
338.	**if** $\mathrm{SumB}^i \in \mathrm{CDom} \cup \mathrm{NDom} \cup \mathrm{ADom} \cup \mathrm{PDom}$ **then** $\mathtt{Bad}[4] = \mathtt{True}$
339.	**else if** $\exists i' < i$ s.t. $\mathrm{AD}^{i'} \neq \mathrm{AD}^i$ and $\mathrm{SumB}^{i'} = \mathrm{SumB}^i$ **then** $\mathtt{Bad}[5] = \mathtt{True}$
340.	**else** $T_A^i = F(\mathrm{SumB}^i)$, $\mathrm{SDom} = \mathrm{SDom} \cup \{\mathrm{SumB}^i\}$

For line 333, $A_j^i \oplus Z_{j+2} = A_j^i \oplus Z_0 \cdot 2^{j+2}$ with $j = 1, 2, \cdots, a_i - 1$, so $\Pr[A_j^i \oplus Z_{j+2} \in \mathrm{CDom} \cup \mathrm{NDom}] \leq (1+i)/2^n$. Notice that every element in SDom is related to either $Z_1 = Z_0 \cdot 2$ or $Z_2 = Z_0 \cdot 2^2$, so $\Pr[A_j^i \oplus Z_{j+2} \in \mathrm{SDom}] \leq (i-1)/2^n$. While in PDom, every element is related to U^j (for some $j \in [1, i-1]$), all of which are independent of Z_0, so $\Pr[A_j^i \oplus Z_{j+2} \in \mathrm{PDom}] \leq \sum_{j=1}^{i-1}(L_{\mathrm{PT}}^j + 1)/2^n$. In a short summary, we have $\Pr[\mathtt{Bad}[2] = \mathtt{True}] \leq \sum_{w=1}^{a_i-1}(\frac{i+1}{2^n} + \frac{i-1}{2^n} + \frac{\sum_{j=1}^{i-1}(L_{\mathrm{PT}}^j+1)}{2^n})$.

For line 334, $A_{j'}^{i'} \oplus Z_{j'+2} = A_j^i \oplus Z_{j+2}$ implies $A_{j'}^{i'} \oplus Z_0 \cdot 2^{j'+2} = A_j^i \oplus Z_0 \cdot 2^{j+2}$. Since $j \neq j'$, this equation has a probability of $1/2^n$ to hold. Then, $\Pr[\mathtt{Bad}[3] = \mathtt{True}] \leq \sum_{j=1}^{a_i-1}(\sum_{i'=1}^{i-1}(L_{\mathrm{AD}}^{i'} - 2) + j - 1)/2^n$.

For line 338, it is easy to get $\Pr[\mathrm{SumB}^i \in \mathrm{CDom} \cup \mathrm{NDom}] \leq (1+i)/2^n$, $\Pr[\mathrm{SumB}^i \in \mathrm{ADom}] \leq \sum_{j=1}^{i}(L_{\mathrm{AD}}^j - 1)/2^n$, and $\Pr[\mathrm{SumB}^i \in \mathrm{PDom}] \leq \sum_{j=1}^{i-1}(L_{\mathrm{PT}}^j + 1)/2^n$, by similar reasons given in analysis for line 333. In a short summary, we have $\Pr[\mathtt{Bad}[4] = \mathtt{True}] \leq ((1+i) + \sum_{j=1}^{i}(L_{\mathrm{AD}}^j - 1) + \sum_{j=1}^{i-1}(L_{\mathrm{PT}}^j + 1))/2^n$.

For line 339, we consider the following cases,

1. $|A_{a_{i'}}^{i'}| = n$ and $|A_{a_i}^i| < n$, then $\mathrm{SumB}^{i'} = \mathrm{SumB}^i$ implies an equation related to the random value Z_0, so $\Pr[\mathrm{SumB}^{i'} = \mathrm{SumB}^i | \text{Case 1}] \leq 1/2^n$.
2. $|A_{a_{i'}}^{i'}| < n$ and $|A_{a_i}^i| = n$, then the same analysis as in Case 1.
3. $|A_{a_{i'}}^{i'}| = n$ and $|A_{a_i}^i| = n$, then consider
   (a) $a_{i'} = a_i$ and $A_j^i = A_j^{i'}$ for $j = 1, 2, \cdots, a_i - 1$. Then by $\mathrm{AD}^{i'} \neq \mathrm{AD}^i$ we know $A_{a_{i'}^{i'}} \neq A_{a_i^i}$, and $\Pr[\mathrm{SumB}^{i'} = \mathrm{SumB}^i | \text{Case 3(a)}] = 0$.

(b) $a_{i'} = a_i$ and $\exists j \in [1, a_i - 1]$, s.t. $A^i_j \neq A^{i'}_j$. Then, $\Pr[\text{SumB}^{i'} = \text{SumB}^i | \text{Case 3(b)}] \leq 1/2^n$, because the equation depends on the randomness of B^i_j and $B^{i'}_j$, which are two independently random values.

(c) $a_{i'} \neq a_i$. Then $\Pr[\text{SumB}^{i'} = \text{SumB}^i | \text{Case 3(c)}] \leq 1/2^n$, for the same reason in Case 3(b).

4. $|A^{i'}_{a_{i'}}| < n$ and $|A^i_{a_i}| < n$, the same as in Case 3.

Then we have $\Pr[\texttt{Bad}[5] = \texttt{True}] \leq \sum_{i'=1}^{i-1} 1/2^n = (i-1)/2^n$.

After line 339, we know when $\text{AD}^i \neq \text{AD}^{i'}$ for $i' = 1, 2, \cdots i-1$, SumB^i is a new input to F, so T^i_A is a random value.

351.	$P^i_1 P^i_2 \cdots P^i_{l_i} = \textbf{Partition}(\text{PT}^i)$, $P^i_0 = 0^n$
352.	**for** $j = 1$ **to** $l_i - 1$ **do**
353.	**if** $P^i_{j-1} \oplus U^i \oplus Z_{j+2} \in \text{CDom} \cup \text{NDom} \cup \text{ADom} \cup \text{SDom} \cup \text{PDom}$
354.	**then** $\texttt{Bad}[6] = \texttt{True}$
355.	**else if** $\exists j' \neq j$ s.t. $P^i_{j'-1} \oplus Z_{j'+2} = P^i_{j-1} \oplus Z_{j+2}$ **then** $\texttt{Bad}[7] = \texttt{True}$
356.	**else** $C^i_j = F(P^i_{j-1} \oplus U^i \oplus Z_{j+2}) \oplus P^i_j \oplus U' \oplus Z_{j+3}$
357.	$\text{PDom} = \text{PDom} \cup \{P^i_{j-1} \oplus U^i \oplus Z_{j+2}\}$
358.	**if** $P^i_{l_i-1} \oplus U^i \oplus Z_{l_i+2} \in \text{CDom} \cup \text{NDom} \cup \text{ADom} \cup \text{SDom} \cup \text{PDom}$
359.	**then** $\texttt{Bad}[8] = \texttt{True}$
360.	**else** $C^i_{l_i} = F(P^i_{l_i-1} \oplus U^i \oplus Z_{l_i+2}) \oplus \textbf{Pad}(P^i_{l_i})$
361.	$\text{PDom} = \text{PDom} \cup \{P^i_{l_i-1} \oplus U^i \oplus Z_{l_i+2}\}$

For line 354, notice $P^i_{j-1} \oplus U^i \oplus Z_{j+2}$ is related to random value U^i, and all the elements in CDom, NDom, ADom, SDom and PDom are not, we have $\Pr[\texttt{Bad}[6] = \texttt{True}] \leq \sum_{j=1}^{l_i-1}(\#\text{CDom} + \#\text{NDom}\# + \text{ADom} + \#\text{SDom} + \#\text{PDom})/2^n \leq \sum_{j=1}^{l_i-1}(1 + i + \sum_{i'=1}^{i}(L^{i'}_{\text{AD}} - 1) + i + \sum_{i'=1}^{i-1}(L^{i'}_{\text{PT}} + 1))/2^n$.

For line 355, $P^i_{j'-1} \oplus Z_{j'+2} = P^i_{j-1} \oplus Z_{j+2}$ with $j' \neq j$ implies an equation depending on the randomness of Z_0, so $\Pr[\texttt{Bad}[7] = \texttt{True}] \leq \binom{L^i_{\text{PT}}-1}{2}/2^n$.

For line 359, the analysis is similar as in lines 354 and 355, and by the randomness of U^i and Z_0 we get $\Pr[\texttt{Bad}[8] = \texttt{True}] \leq (\#\text{CDom} + \#\text{NDom}\# + \text{ADom} + \#\text{SDom} + \#\text{PDom})/2^n \leq (1 + i + \sum_{i'=1}^{i}(L^{i'}_{\text{AD}} - 1) + i + \sum_{i'=1}^{i-1}(L^{i'}_{\text{PT}} + 1) + L^i_{\text{PT}} - 1)/2^n$.

After line 359, we have $C^i_1, C^i_2, \cdots C^i_{l_i}$ are all random values, if the flags $\texttt{Bad}[j] = \texttt{False}$ with $j = 1, 2, \cdots, 8$.

371.	**if** $\lvert P^i_{l_i}\rvert < n$ **then** $C^i_{l_i} R^i = \mathbf{Cut}(C^i_{l_i}, \lvert P^i_{l_i}\rvert)$, Last $= (P^i_{l_i} \lvert\lvert R^i) \oplus U^i \oplus Z_1$
372.	**else** Last $= P^i_{l_i} \oplus U^i \oplus Z_2$
373.	**if** Last $\in$ CDom $\cup$ NDom $\cup$ ADom $\cup$ SDom $\cup$ PDom
374.	**then** `Bad`[9] $=$ `True`
375.	**else** $C^i_{l_i+1} = F$(Last), PDom $=$ PDom $\cup$ {Last}
376.	**return** $\mathrm{CT}^i = C^i_1 C^i_2 \cdots C^i_{l_i}$, $T^i = T^i_A \oplus C^i_{l_i+1}$

For line 374, the analysis is similar as in lines 354 and 355. The probability is $\Pr[\texttt{Bad}[9] = \texttt{True}] \leq (1 + i + \sum_{i'=1}^{i}(L^{i'}_{\mathrm{AD}} - 1) + i + \sum_{i'=1}^{i-1}(L^{i'}_{\mathrm{PT}} + 1) + L^i_{\mathrm{PT}})/2^n$.

After line 374, if flags $\texttt{Bad}[j] = \texttt{False}$ $(j = 1, 2, \cdots, 9)$, we know both CT^i and T^i are random values, and iFeed[F].AEnc successfully cheat $\mathcal{A}$. Its probability to fail to do this is upper bounded by,

$$\begin{aligned} & \Pr[\text{iFeed}[F].\text{AEnc fails in the } i\text{−th time}] \\ \leq\ & \sum_{j=1}^{9} \Pr[\texttt{Bad}[j] = \texttt{True}] \\ \leq\ & \frac{\sum_{j=1}^{i-1}(L^j_{\mathrm{AD}} + L^j_{\mathrm{PT}})W^i + (W^i)^2/2 + 2iW^i}{2^n}, \end{aligned}$$

with $W^i = L^i_{\mathrm{AD}} + L^i_{\mathrm{PT}} + 2$. Finally we get

$$\begin{aligned} \mathbf{Adv}^{\mathrm{Priv}}_{\mathrm{iFeed}[F]}(\mathcal{A}) &= |\Pr[\mathcal{A}^{\mathrm{iFeed}[F].\mathrm{AEnc}(\cdot)} = 1] - \Pr[\mathcal{A}^{\$(\cdot)} = 1]| \\ &\leq \sum_{i=1}^{q} \Pr[\text{iFeed}[F].\text{AEnc fails in the } i\text{−th time}] \\ &\leq \frac{(\sigma + 2q)^2 + 4(\sigma + 2q)}{2^{n+1}}, \end{aligned}$$

with $\sigma = \sum_{i=1}^{q}(L^i_{\mathrm{AD}} + L^i_{\mathrm{PT}})$. This finishes the proof for iFeed[F].AEnc privacy. □

Proof for Authenticity A nonce-respecting adversary $\mathcal{A}$ has access to an encryption oracle iFeed[F].AEnc and a decryption oracle iFeed[F].ADec. $\mathcal{A}$ can make forward queries (PMN^i, AD^i, PT^i) to iFeed[F].AEnc, and get (CT^i, T^i)=iFeed[F].AEnc(PMN^i, AD^i, PT^i). $\mathcal{A}$ can also make backward queries ($\mathrm{PMN}^{i'}$, $\mathrm{AD}^{i'}$, $\mathrm{CT}^{i'}$, $T^{i'}$) to iFeed[F].ADec and get either $\mathrm{PT}^{i'}$ =iFeed[F].ADec($\mathrm{PMN}^{i'}$, $\mathrm{AD}^{i'}$, $\mathrm{CT}^{i'}$, $T^{i'}$) or a failure symbol $\perp$. Without loss of generality, we assume $\mathcal{A}$ makes exactly q forward queries and q' backward queries.

In the following, we check the process details of both oracles on $\mathcal{A}$'s queries, and show that in most cases iFeed[F].AEnc outputs random bits and iFeed[F].ADec returns a failure symbol $\perp$, and their probability to fail is sufficiently small.

As in the previous proof for privacy, before $\mathcal{A}$ makes its first query we define a flag Bad whose value being True means iFeed[F].ADec returns a plaintext other than $\perp$. Let sets CDom, NDom, ADom, SDom and PDom, and values L_{ADmax} and L_{PTmax} be defined as before. Denote L'_{ADmax} and L'_{CTmax} as the maximum block length of AD$'$ and CT$'$ in backward queries to iFeed[F].ADec. At the beginning, we have

401.	$\texttt{Bad} = \texttt{False}$
402.	CDom $= \{0^n\}$; ADom, SDom, NDom, PDom =EmptySet
403.	$Z_0 = F(0^n)$
404.	**for** $j = 1$ **to** $\max\{L_{\text{ADmax}}, L_{\text{PTmax}}, L'_{\text{ADmax}}, L'_{\text{CTmax}}\} + 2$ **do**
405.	$Z_j = Z_{j-1} \cdot 2$

Proof. Suppose now $\mathcal{A}$ has made $i-1$ forward queries (PMN^j, AD^j, PT^j) for $j = 1, 2, \cdots, i-1$, in response to which iFeed[F].AEnc has successfully output $i-1$ random strings (CT^j, T^j) of length $|\text{CT}^j| = |\text{PT}^j|$ and $|T^j| = \tau$, and $\mathcal{A}$ has made $i'-1$ backward queries (PMN^j, AD^j, CT^j, T^j) for $j = 1, 2, \cdots, i'-1$, in response to which iFeed[F].ADec has successfully output $i'-1$ $\perp$.

Then, let $\mathcal{A}$ make its next query. If this is the i-th forward query (PMN^i, AD^i, PT^i), then by the previous analysis iFeed[F].AEnc has a probability at least $1-(\sum_{j=1}^{i-1}(L^j_{\text{AD}}+L^j_{\text{PT}})W+W^2/2+2iW)/2^n$ to still output random bits. If this is the i'-th backward query (PMN$^{i'}$, AD$^{i'}$, CT$^{i'}$, $T^{i'}$), we check the details of how iFeed[F].ADec deals with it.

iFeed[F].ADec(PMN$^{i'}$, AD$^{i'}$, CT$^{i'}$, $T^{i'}$)	
411.	**if** $\|\text{AD}^{i'}\| = 0$ **then** $T^{i'}_A = 0^n$
412.	**else if** $\exists j \in [1, i-1]$ s.t. AD$^{i'}$ = AD^j **then** $T^{i'}_A = T^j_A$
413.	**else** $T^{i'}_A = \text{Mac}[F](\text{AD}^{i'})$
414.	**if** $N^{i'} \in$ NDom
415.	**then** $U^{i'} = F(N^{i'})$, run Subroutine OldNonceinDec[F]$(Z, U^{i'}, \text{CT}^{i'})$
416.	**else if** $N^{i'} \in \text{ADom} \cup \text{SDom} \cup \text{PDom}$ **then** $\texttt{Bad}[10] = \texttt{True}$
417.	**else** $N^{i'} = F(N^{i'})$, run Subroutine NewNonceinDec[F]$(Z, U^{i'}, \text{CT}^{i'})$
418.	**if** $T^{i'} \neq$ **Truncate**$(T^{i'}_A \oplus C^{i'}_{l_{i'}+1}, \tau)$ **then return** $\perp$
419.	**else** $\texttt{Bad}[14] = \texttt{True}$, **return** $\text{CT}^{i'} = P^{i'}_1 P^{i'}_2 \cdots P^{i'}_{l_{i'}}$

For line 413, see details in lines from 331 to 340, and we know if AD$^{i'}$ is new then $T^{i'}_A$ would be a random value with a high probability $1-\varepsilon_1$, with $\varepsilon_1 \leq \sum_{j=2}^{5} \Pr[\texttt{Bad}[j] = \texttt{True}]$, where we replace a_i by $a_{i'} = |\text{AD}^{i'}|_n$

For line 416, $\Pr[\texttt{Bad}[10] = \texttt{True}] \leq (\#\text{ADom} + \#\text{SDom} + \#\text{PDom})/2^n \leq (\sum_{j}^{i-1}(L^j_{\text{AD}} + L^j_{\text{PT}} + 1))/2^n$, because all the elements in these three sets are related to a random value Z_0.

Subroutine NewNonceinDec$[F](Z, U^{i'}, \mathrm{CT}^{i'})$	
421.	$C_1^{i'} C_2^{i'} \cdots C_{l_{i'}}^{i'} = \mathbf{Partition}(\mathrm{CT}^{i'})$
422.	$P_0^{i'} = 0^n$
423.	**for** $j = 1$ **to** $l_{i'} - 1$ **do**
424.	**if** $P_{j-1}^{i'} \oplus Z_{j+2} \oplus U^{i'} \in$ CNASP **then** Bad[11] = True
425.	**else** $P_j^{i'} = F(P_{j-1}^{i'} \oplus Z_{j+2} \oplus U^{i'}) \oplus C_j^{i'} \oplus Z_{j+3} \oplus U^{i'}$
426.	**if** $P_{l^{i'}-1}^{i'} \oplus Z_{l_{i'}+2} \oplus U^{i'} \in$ CNASP **then** Bad[12] = True
427.	**else** $P_{l_{i'}}^{i'} = F(P_{l^{i'}-1}^{i'} \oplus Z_{l_{i'}+2} \oplus U^{i'}) \oplus \mathbf{Pad}(C_{l_{i'}}^{i'})$
428.	**if** $\lvert C_{l_{i'}}^{i'} \rvert < n$ **then** $P_{l_{i'}}^{i'} R^{i'} = \mathbf{Cut}(P_{l_{i'}}^{i'}, \lvert C_{l_{i'}}^{i'} \rvert)$
429.	Last $= (P_{i'}^{i'} \lvert\lvert R^{i'}) \oplus Z_1 \oplus U^{i'}$
430.	**else** Last $= P_{i'}^{i'} \oplus Z_2 \oplus U^{i'}$
431.	**if** Last $\in$ CNASP **then** Bad[13] = True
432.	**else** $C_{l_{i'}+1}^{i'} = F(\text{Last})$
433.	**return** $P_1^{i'} P_2^{i'} \cdots P_{l_{i'}}^{i'}, C_{l_{i'}+1}^{i'}$

For line 417, see details in lines from 421 to 433. From here we Denote CNASP as CDom $\cup$ NDom $\cup$ ADom $\cup$ SDom $\cup$ PDom for short. Since $N^{i'}$ is a newly produced random value, we get $\Pr[\texttt{Bad}[11] = \texttt{True}] \leq \sum_{w=1}^{L_{\mathrm{CT}'}^{i'}-1}$ #CNASP$/2^n$ with #CNASP $= \sum_j^{i-1}(L_{\mathrm{AD}}^j + L_{\mathrm{PT}}^j) + 2i - 1$ for line 424. Similarly, $\Pr[\texttt{Bad}[12] = \texttt{True}] = \Pr[\texttt{Bad}[13] = \texttt{True}] \leq$ #CNASP$/2^n$ for line 426 and 431.

Subroutine OldNonceinDec$[F](Z, U^{i'}, \mathrm{CT}^{i'})$	
441.	$C_1^{i'} C_2^{i'} \cdots C_{l_{i'}}^{i'} = \mathbf{Partition}(\mathrm{CT}^{i'})$
442.	$P_0^{i'} = 0^n$
443.	**for** $j = 1$ **to** $w + 1$ **do**
444.	$P_j^{i'} = F(P_{j-1}^{i'} \oplus Z_{j+2} \oplus U^{i'}) \oplus C_j^{i'} \oplus Z_{j+3} \oplus U^{i'}$
445.	**for** $j = w + 2$ **to** $l_{i'} - 1$ **do**
446.	followed by lines from 424 to 433

For line 415, see details in lines from 441 to 446. Suppose $U^{i'} = U^c$ for some $c \in [1, i-1]$, then we denote w as the block length of common prefix in $\mathrm{CT}^{i'}$ and CT^c. After decrypting their common prefix, we consider the probability of $P_{j-1}^{i'} \oplus Z_{j+2} \oplus U^{i'} \in$ CNASP for $j \in [w+2, l_{i'} - 1]$ in line 424. Note that $P_{w+1}^{i'} = P_{w+1}^c \oplus C_{w+1}^c \oplus C_{w+1}^{i'}$, and adversary $\mathcal{A}$ tries to select a proper $C_{w+1}^{i'}$ to induce $P_{w+1}^{i'} \oplus Z_{w+4} \oplus U^{i'} \in$ CNASP. This is unlikely happen because $\Pr[P_{w+1}^{i'} \oplus Z_{w+4} \oplus U^{i'} \in \mathrm{CDom} \cup \mathrm{NDom}] \leq i/2^n$ by the randomness of Z_0, and $\Pr[P_{w+1}^{i'} \oplus Z_{w+4} \oplus U^{i'} \in \mathrm{ADom} \cup \mathrm{SDom}] \leq \sum_{j=1}^{i-1} L_{\mathrm{AD}}/2^n$ by the randomness of U^c, and $\Pr[P_{w+1}^{i'} \oplus Z_{w+4} \oplus U^{i'} \in \mathrm{PDom}] \leq \sum_{j=1, j\neq c}^{i-1}(L_{\mathrm{PT}} + 1)/2^n + (L_{\mathrm{PT}} + 1 - w)/2^n$ by the randomness of Z_0. Specially note that in the last probability the first term is over the randomness of U^c and the second term is over the randomness of Z_0. Once $P_{w+1}^{i'} \oplus Z_{w+4} \oplus U^{i'}$ is new, all the

subsequent invocations on F can seldom collide with previous inputs, and their colliding probability is upper bounded by $\sum_{j=11}^{13} \Pr[\texttt{Bad}[j] = \texttt{True}]$. If all these three flags are `False`, then $C^{i'}_{l_{i'}+1}$ is a random value.

Denote $X \nsim \$$ as the event that variable X is not an independently random value. Now let us upper bound the probability for iFeed[F].ADec(PMN$^{i'}$, AD$^{i'}$, CT$^{i'}$, $T^{i'}$) to not output $\perp$ by the following cases.

1. $N^{i'} \notin$ NDom. Then, by the analysis for line 416 and 417, the probability for $\Pr[C^{i'}_{l_{i'}+1} \nsim \$] \leq \sum_{j=10}^{13} \Pr[\texttt{Bad}[j] = \texttt{True}]$. If all $\texttt{Bad}[j] = \texttt{False}$ ($j \in [10, 13]$), the probability for iFeed[F].ADec to not output $\perp$ in line 419 is upper bounded by $\Pr[\texttt{Bad}[14] = \texttt{True}] \leq 1/2^\tau$.

2. $\exists j \in [1, i-1]$ s.t. $N^{i'} = N^j$ but CT$^{i'} \neq$ CT^j. By the analysis for line 415, we have $\Pr[C^{i'}_{l_{i'}+1} \nsim \$] \leq \sum_{j=11}^{13} \Pr[\texttt{Bad}[j] = \texttt{True}]$ and $\Pr[\texttt{Bad}[14] = \texttt{True}] \leq 1/2^\tau$.

3. $\exists j \in [1, i-1]$ s.t. $N^{i'} = N^j$ and CT$^{i'}$ = CT^j, $|\text{AD}^{i'}| \neq 0$ and AD$^{i'} \notin \{$AD^1, AD^2, $\cdots$, AD$^{i-1}\}$. By the analysis for line 413, we have $\Pr[T^{i'}_A \nsim \$] \leq \sum_{j=2}^{5} \Pr[\texttt{Bad}[j] = \texttt{True}]$ and $\Pr[\texttt{Bad}[14] = \texttt{True}] \leq 1/2^\tau$.

4. $\exists j \in [1, i-1]$ s.t. $N^{i'} = N^j$ and CT$^{i'}$ = CT^j, $|\text{AD}^{i'}| = 0$, and $|\text{AD}^j| \neq 0$. This leads to the adversary $\mathcal{A}$ guess the random value $C^j_{l_j+1}$, which is hidden in $T^j = C^j_{l_j+1} \oplus T^j_A$, and its probability to win is $\Pr[\texttt{Bad}[14] = \texttt{True}] \leq 1/2^\tau$.

5. $\exists j1, j2 \in [1, i-1]$ s.t. $j1 \neq j2$, $N^{i'} = N^{j1}$, CT$^{i'}$ = CT^{j1} and AD$^{i'}$ = AD^{j2}. This implies $T^{i'} = T^{j2}_A \oplus C^{j1}_{l_{j1}+1}$, and $\mathcal{A}$ has to guess both T^{j2}_A and $C^{j1}_{l_{j1}+1}$. Its probability to win is $\Pr[\texttt{Bad}[14] = \texttt{True}] \leq 1/2^\tau$.

These five cases cover all the different forms that $\mathcal{A}$ may use to forge against iFeed[F].ADec. Based on the condition that iFeed[F].AEnc has output $i-1$ random strings in response to $\mathcal{A}$'s $i-1$ forward queries, and iFeed[F].ADec has output $i'-1$ $\perp$ in response to $\mathcal{A}$'s $i'-1$ backward queries, by the above analysis we know

$$\begin{aligned}
& \Pr[\text{iFeed}[F].\text{ADec doesn}'\text{t output}\perp \text{ in the } i'-\text{th time}] \\
\leq & \sum_{j=2,3,4,5,10,11,12,13,14} \Pr[\texttt{Bad}[j] = \texttt{True}] \\
\leq & \frac{\sum_{j=1}^{i-1}(L^j_{\text{AD}} + L^j_{\text{PT}})W^{i'} + (2i-1)W^{i'} + (L^{i'}_{\text{AD}'})^2/2 + L^{i'}_{\text{AD}'}/2}{2^n} + \frac{1}{2^\tau} \\
\leq & \frac{\sum_{j=1}^{q}(L^j_{\text{AD}} + L^j_{\text{PT}})W^{i'} + 2qW^{i'} + (L^{i'}_{\text{AD}'})^2/2 + L^{i'}_{\text{AD}'}/2}{2^n} + \frac{1}{2^\tau},
\end{aligned}$$

with $W^{i'} = L^{i'}_{\text{AD}'} + L^{i'}_{\text{CT}'} + 2$.

Then suppose $\mathcal{A}$ made exactly q forward queries and q_v backward queries, the probability for $\mathcal{A}$ to forge is upper bounded by

$$\begin{aligned} & \mathbf{Adv}^{\mathrm{Auth}}_{\mathrm{iFeed}[F]}(\mathcal{A}) \\ \leq\ & \mathbf{Adv}^{\mathrm{Priv}}_{\mathrm{iFeed}[F].\mathrm{AEnc}}(\mathcal{A}) + \\ & \sum_{i'=1}^{q_v} \Pr[\mathrm{iFeed}[F].\mathrm{ADec\ doesn't\ output} \perp \mathrm{\ in\ the\ } i'\mathrm{-th\ time}] \\ \leq\ & \frac{(\sigma+\sigma_v+2q+2q_v)^2+8q}{2^{n+1}} + \frac{q_v}{2^\tau}, \end{aligned}$$

with $\sigma = \sum_{i=1}^{q}(L^i_{\mathrm{AD}} + L^i_{\mathrm{PT}})$ and $\sigma_v = \sum_{i=1}^{q_v}(L^i_{\mathrm{AD}'} + L^i_{\mathrm{CT}'})$. The finishes the proof for iFeed$[F]$ authenticity. □

The Results for iFeed$[P]$ For the privacy of iFeed$[P]$, note that after $\mathcal{A}$ has made q forward queries, iFeed$[P]$.AEnc calls E for no more that $1+2q+\sigma$ times, by the PRP/PRF switching lemma [5], we add a term $\binom{1+2q+\sigma}{2}/2^n$ to the privacy bound of iFeed$[F]$, and get

$$\begin{aligned} \mathbf{Adv}^{\mathrm{Priv}}_{\mathrm{iFeed}[P]}(\mathcal{A}) &= \mathbf{Adv}^{\mathrm{Priv}}_{\mathrm{iFeed}[F]}(\mathcal{A}) + \binom{1+2q+\sigma}{2}/2^n \\ &\leq \frac{(\sigma+2q)^2+2.5(\sigma+2q)}{2^n}, \end{aligned}$$

with $\sigma = \sum_{i=1}^{q}(L^i_{\mathrm{AD}} + L^i_{\mathrm{PT}})$.

Similarly, to give the authenticity bound for iFeed$[P]$, we add a term $\binom{1+2q+\sigma+\sigma_v+2q_v}{2}/2^n$ to the authenticity bound of iFeed$[F]$, as follows

$$\begin{aligned} \mathbf{Adv}^{\mathrm{Auth}}_{\mathrm{iFeed}[P]}(\mathcal{A}) &= \mathbf{Adv}^{\mathrm{Auth}}_{\mathrm{iFeed}[F]}(\mathcal{A}) + \binom{1+2q+\sigma+\sigma_v+2q_v}{2}/2^n \\ &\leq \frac{(\sigma+\sigma_v+2q+2q_v+1)^2}{2^n} + \frac{q_v}{2^\tau}, \end{aligned}$$

with $\sigma = \sum_{i=1}^{q}(L^i_{\mathrm{AD}} + L^i_{\mathrm{PT}})$ and $\sigma_v = \sum_{i=1}^{q_v}(L^i_{\mathrm{AD}'} + L^i_{\mathrm{CT}'})$.

Chapter 5

Features

One-Pass Only one pass through plaintext is required to provide both integrity and confidentiality protections for it. This provides an easy and convenient way to protect plaintext.

Rate-1 Only one AES invocation is required in processing associated data and plaintext for their each block. This saves computation loads compared with those of higher rate.

Parallel Authenticated Encryption iFeed[AES] can fully parallelize AES invocations in authenticated encryption. Specially, the tag is generated by only the last plaintext block, and this makes it parallel with other plaintext block encryptions.

Serial Authenticated Decryption iFeed[AES] has to decrypt serially, but this enables intermediate tags for early detection on wrong ciphertext blocks, as mentioned below.

Intermediate Tag Supporting We can append an all zero redundancy block 0^{128} to the end of each w-block plaintext, and use iFeed[AES] to get the ciphertext blocks as $C_1 x_1 C_2 x_2 \cdots C_{l-1} x_{l-1} C_l$, where C_i here is a w-block ciphertext and x_i is its next ciphertext block. For the last C_l, we append no 0^{128} and thus there is no x_l.

Then, in decryption we have at least the following advantages:

1. in serial decryption, we can check the validity of each $C_i x_i$ by seeing whether its last plaintext block is 0^{128} or not. If 0^{128}, output the corresponding w-block plaintext, otherwise stop the whole decryption.

   x_i is called intermediate tag, discussed in another design by Datta and Nandi [6]. It helps to avoid caching too long plaintext blocks before finally confirming the ciphertext validity by the tag.

2. parallelize in decrypting each $C_i x_i$, by knowing its previous plaintext block is 0^{128}.

iFeed[AES] is proved to be secure against any nonce-respecting attackers with any plaintext. This ensures us to inject redundancy into plaintext, conditioned on the total number of queried associated data, plaintext, and redundancy is still within $2^{64} - 4$ blocks limit.

The disadvantage of injecting redundancy is ciphertext length extension by about $|\mathrm{PT}|_n/w$ blocks.

The width w should be carefully selected because it is a tradeoff between ciphertext length extension and decryption efficiency, and it should also be strictly fixed to prevent sliding attacks.

Inverse-Free on AES No inverse queries on AES is required in all the processions of iFeed[AES]. This reduces iFeed[AES] security to PRP (not SPRP) assumption on AES, and thus making it a strong authenticated encryption scheme in theory. Furthermore, this provides a possibility to reduce areas on hardware implementation or to speed up in some platforms with slower AES decryption, as discussed in [7].

This also provides a possibility to turn iFeed into an authenticated encryption scheme using compression functions in hash functions.

One-Key Only a single key is required to initialize AES-128, for all the processions of plaintext and associated data, reaching the minimum key size.

On-Line iFeed[AES] can start authenticated encryption procedures without knowing the length information of plaintext and associated data. This allows encryption on-the-fly.

Ciphertext Length Preserving The length of ciphertext (excluding the tag) obtained from iFeed[AES] is the same with the length of plaintext. This facilitates ciphertext storage.

Endian-Less All the inputs, outputs, and internal states of iFeed[AES] are byte strings, no numeral representation for them is required. Thus iFeed[AES] has no endian preference for platforms.

Incremental without Changing Tags When an iFeed[AES] user wants to modify his i-th plaintext block P_i after having finished the authenticated encryption procedures, he can just use the original public message number to recall the corresponding one AES invocation (the one with input $P_i \oplus Z_{i+3} \oplus U$) and modify both C_{i-1} and C_i consequently. Specially, this update has no influence over other ciphertext blocks and the original tag.

Simple Structure with Mature Techniques The overall structure of iFeed[AES] is quite simple, and its supporting techniques such like PRP assumption on AES, generating secret masks by $\mathrm{AES}(K, 0^{128})$ and "$\cdot 2$" over $\mathrm{GF}(2^{128})$ have been used in many other modes of operation for security reduction, fast implementation and easy-to-understand proof [8, 9]. Furthermore, the adopted EMAC structure (in iFeed[AES].ADec) and PMAC structure (in Mac[AES]) have been thoroughly studied and their security analysis has been commonly accepted [5, 3, 10, 11, 8, 12, 4, 9, 13, 14, 15].

Next we draw a table to carefully compare iFeed[AES] with GCM[AES-128], as below. The first row shows iFeed[AES] advantages over GCM[AES-128], and the second row shows iFeed[AES] disadvantages over GCM[AES-128], and the third row shows their common features.

	iFeed[AES]	GCM[AES-128]
#pass	One	Two
endian-less	yes	no
# multiplication "$\cdot H$"	0	Many$^\sharp$
intermediate tag supporting	yes	hardly
plaintext size	$[0, \lvert\mathrm{PT}\rvert_{\max}]^\S$	$[0, 2^{39} - 256]$
associated data size	$[0, \lvert\mathrm{AD}\rvert_{\max}]^\S$	$[0, 2^{64} - 1]$
# AES calls	$\lvert\mathrm{PT}\rvert_{128} + \lvert\mathrm{AD}\rvert_{128} + q + 1$	$\lvert\mathrm{PT}\rvert_{128} + q + 1$ †
# multiplication "$\cdot 2$"	$\max\{\lvert\mathrm{PT}\rvert_{128}, \lvert\mathrm{AD}\rvert_{128}\} + 2$	0
public message number size	[1,127]	$[1, 2^{64} - 1]$
parallel decryption	no	yes
key size	128	
secret message number size	0	
tag size	$[32, 128]^\ddagger$	
ciphertext length preserving	yes	
on-line	yes	
inverse-free on AES	yes	
parallel encryption	yes	
nonce-reuse settings	forgeable	
nonce-reuse settings	ciphertext leaks plaintext differentials	

$\sharp$ Many= $\lvert\mathrm{AD}\rvert_{128} + \lvert\mathrm{PMN}\rvert_{128} + \lvert\mathrm{PT}\rvert_{128} + c$. where $c = 2$ if $\lvert\mathrm{PMN}\rvert \neq 96$ else $c = 1$.
§ $\lvert\mathrm{PT}\rvert_{\max} + \lvert\mathrm{AD}\rvert_{\max} \leq 2^{71} - 512$.
† q is the total number of encryption queries.
‡ All numbers for parameter size stand for length in bits.

Compared with GCM[AES-128], iFeed[AES] realizes authenticated encryption in a one-pass way, removing multiplications "$\cdot H$" and adopting intermediate tag supporting easily. It can encrypt even longer plaintexts, and its overall structure is quite simple. In many applications using GCM[AES-128], e.g. encrypting Internet packets, iFeed[AES] may do even better, except in those with extreme requirements on fully parallel decryption.

Chapter 6

Design Rationale

iFeed Basic Construction The authenticated encryption part for plaintext in iFeed[AES] is designed from a simple construction iFeed Basic, as illustrated in Fig. 6.1. This simple construction is used to avoid inverse queries on underlying blockciphers in both authenticated encryption and authenticated decryption, by an observation that any ciphertext block itself provides authenticity protection for its corresponding plaintext block, and furthermore this ciphertext block is a random string that can be used to mask other plaintext blocks for their privacy protections.

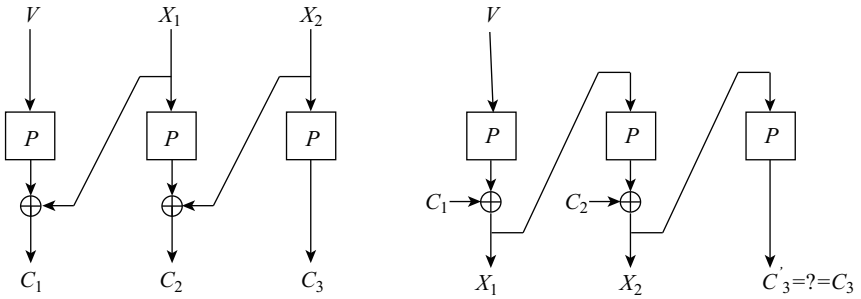

Figure 6.1: Illustration of iFeed Basic. On the left is authenticated encryption part, and on the right is authenticated decryption part.

By this observation, iFeed Basic feeds backward every blockcipher input to its previous blockcipher output in authenticated encryption, and the invocations on blockciphers can be fully parallelized.

In decryption, the overall procedures are serial, and plaintext blocks have to be decrypted one by one in the cipher-block-chaining (CBC) manner. The ciphertext validity would be checked in the end of CBC structure.

We should note that iFeed Basic construction itself is not a secure authenticated encryption scheme, because its inputs to blockciphers should be pariwise distinct and kept secret. This can be reached by using nonces and multiplication "$\cdot 2$" over GF(2^{128}).

We can also feed forward every blockcipher input to its next blockcipher output, see Fig. 6.2. This construction keeps ciphertext length preserving naturally, but in decryption its ciphertext blocks have to be inputted inversely, making it inconvenient.

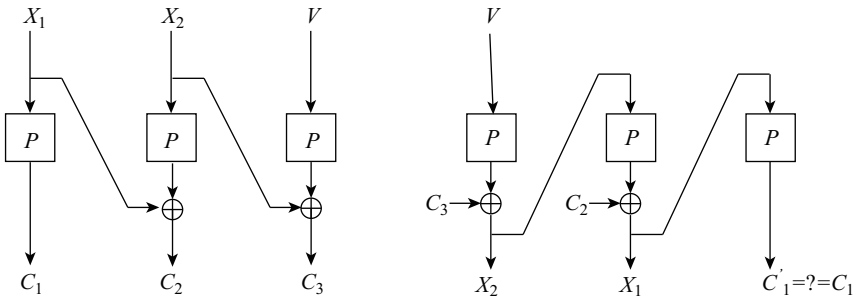

Figure 6.2: Illustration of Mirrored iFeed Basic. On the left is authenticated encryption part, and on the right is authenticated decryption part.

Ciphertext Stealing iFeed Basic only applies to plaintexts of full blocks. We can of course append all plaintexts in iFeed Basic, but this would result in ciphertext length extension. To overcome this, we use the ciphertext stealing technique [16, 17, 18, 19, 20].

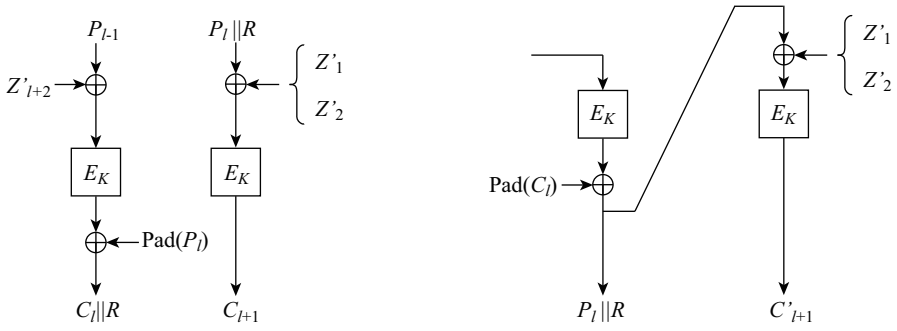

Figure 6.3: Illustration of Ciphertext Stealing in iFeed[AES]. On the left is the part in authenticated encryption, and on the right is the part in authenticated decryption.

If the last plaintext block P_l is not full block long, we cut its previous ciphertext block (the previous blockcipher output XORed by $\mathbf{Pad}(P_l)$) into two parts, where the left part is of length $|P_l|$, denoted as C_l, and the right part is of length $n - |P_l|$, denoted as R. The stolen R is further appended to the right of P_l, and they together (XORed by Z'_1 or Z'_2) consist an input of full block long to the last blockcipher. The final ciphertext is $C_1C_2 \cdots C_l, C_{l+1}$, where R is not included and should be kept secret. The

decryption procedures should be adjusted accordingly.

If the last plaintext block P_l is already full block long, we just use the original iFeed Basic construction. To differ these two cases, two secret masks Z'_1 and Z'_2 are used respectively.

Notice that ciphertext stealing technique makes iFeed[AES] less fully parallelizable in the last two blockcipher invocations.

Multiplication "·2" over GF(2^{128}) To ensure security, the inputs to blockciphers in iFeed Basic construction need to be pairwise distinct and kept secret. We introduce the commonly used multiplication "·2" over GF(2^n) to solve this [8, 9]. By selecting a primitive polynomial $p_n(x)$ over GF(2), we get an extension field GF(2^n) of x, where x is the primitive that can be used to generate all the none-zero elements in GF(2^n).

For iFeed[AES] we use $p_{128}(x) = x^{128} + x^7 + x^2 + x + 1$, then $x = 2$ is the primitive, and any none-zero value X can represent all the elements in GF(2^{128}) by $X \cdot 2^i$ for $i \in [1, 2^{128} - 1]$. Further notice that X multiplied by 2 can be calculated fairly fast and easily. That is,

$$X \cdot 2 = \begin{cases} X << 1, & \text{if the leftmost bit of } X \text{ is 0} \\ X << 1 \oplus (0^{120}||10000111), & \text{if the leftmost bit of } X \text{ is 1} \end{cases}$$

By selecting $X \xleftarrow{\$} \{0,1\}^n$ and keeping it secret, we have $\Pr[M_1 \oplus X \cdot 2^i = M_2 \oplus X \cdot 2^j] \leq 1/2^n$ for any M_1, M_2 with $i \neq j$. In this way, we make the inputs to iFeed[AES] pairwise distinct and secret with a close-to-1 probability.

Independent MAC for Associated Data Mac[AES] is a PMAC-like message authentication mode independent of iFeed construction. This independence allows parallelly processing associated data and plaintext, and thus it provides a possibility to speed up the whole efficiency.

Also due to this independence, Mac[AES] can be easily replaced by some other MAC if required, but the overall security of iFeed[AES] needs reconsideration.

If the speedup is not necessary, we can use Mac[AES] to authenticate both associated data and public message number, and treat their tag as the secret value Z_0 in iFeed[AES]. Then, C_{l+1} itself serves as the tag. By this adjustment we extend the length of public message number to be as long as nearly 2^{64} blocks.

The designers have not hidden any weaknesses in this cipher.

Chapter 7

Intellectual Property

Mac[AES] is inspired by PMAC, which has been patented by Phillip Rogaway [21]. We claim no patent for either iFeed construction or Mac[AES]. As far as we know the iFeed construction is not in known patents, patent applications, or planned patent applications, and it has no intellectual-property constraints relevant to use.

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 analysis 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 analysis then they are expected to promptly and publicly respond to those analysis, 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.

Chapter 9

Acknowledgements

The designers would like to thank Lei Wang and the anonymous reviewers at FSE 2014 for their fruitful comments. Thanks Jian Zou for helping to produce test vectors.

Bibliography

[1] Rogaway, P.: Nonce-based symmetric encryption. [22] 348–359

[2] Bellare, M., Rogaway, P., Wagner, D.: The eax mode of operation. [22] 389–407

[3] Petrank, E., Rackoff, C.: Cbc mac for real-time data sources. J. Cryptology **13**(3) (2000) 315–338

[4] Black, J., Rogaway, P.: A block-cipher mode of operation for parallelizable message authentication. In Knudsen, L.R., ed.: EUROCRYPT. Volume 2332 of Lecture Notes in Computer Science., Springer (2002) 384–397

[5] Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. **61**(3) (2000) 362–399

[6] Datta, N., Nandi, M.: Misuse resistant parallel authenticated encryptions. Cryptology ePrint Archive, Report 2013/767 (2013) `http://eprint.iacr.org/`.

[7] Minematsu, K.: Parallelizable authenticated encryption from functions. Cryptology ePrint Archive, Report 2013/628 (2013) `http://eprint.iacr.org/`.

[8] Iwata, T., Kurosawa, K.: Omac: One-key cbc mac. [23] 129–153

[9] Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes ocb and pmac. In Lee, P.J., ed.: ASIACRYPT. Volume 3329 of Lecture Notes in Computer Science., Springer (2004) 16–31

[10] Black, J., Rogaway, P.: Cbc macs for arbitrary-length messages: The three-key constructions. In Bellare, M., ed.: CRYPTO. Volume 1880 of Lecture Notes in Computer Science., Springer (2000) 197–215

[11] Kurosawa, K., Iwata, T.: Tmac: Two-key cbc mac. In Joye, M., ed.: CT-RSA. Volume 2612 of Lecture Notes in Computer Science., Springer (2003) 33–49

[12] Bellare, M., Pietrzak, K., Rogaway, P.: Improved security analyses for cbc macs. In Shoup, V., ed.: CRYPTO. Volume 3621 of Lecture Notes in Computer Science., Springer (2005) 527–545

[13] Knudsen, L.R., Kohno, T.: Analysis of rmac. [23] 182–191

[14] Yasuda, K.: The sum of cbc macs is a secure prf. In Pieprzyk, J., ed.: CT-RSA. Volume 5985 of Lecture Notes in Computer Science., Springer (2010) 366–381

[15] Yasuda, K.: Pmac with parity: Minimizing the query-length influence. In Dunkelman, O., ed.: CT-RSA. Volume 7178 of Lecture Notes in Computer Science., Springer (2012) 203–214

[16] Meyer, C.H., Matyas, S.M.: Cryptography: a new dimension in data security. John Wiley and Sons, New York, 1982

[17] Schneier, B.: Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C. New York, Wiley, 1996.

[18] Dworkin, M.: Recommendation for block cipher modes of operation: three variants of ciphertext stealing for CBC mode. Addendum to NIST Special Publication 800-38A. October 2010.

[19] Daemen, J.: Hash Function and Cipher Design: Strategies Based on Linear and Differential Cryptanalysis. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium (1995)

[20] Rogaway, P., Wooding, M., Zhang, H.: The security of ciphertext stealing. In Canteaut, A., ed.: FSE. Volume 7549 of Lecture Notes in Computer Science., Springer (2012) 180–195

[21] Rogaway, P.: Method and apparatus for realizing a parallelizable variable-input-length pseudorandom function. Available at: `http://patents.justia.com/patent/20020051537`.

[22] Roy, B.K., Meier, W., eds.: Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers. In Roy, B.K., Meier, W., eds.: FSE. Volume 3017 of Lecture Notes in Computer Science., Springer (2004)

[23] Johansson, T., ed.: Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers. In Johansson, T., ed.: FSE. Volume 2887 of Lecture Notes in Computer Science., Springer (2003)