SHELL v1

Designer and Submitter

Lei Wang

Nanyang Technological University, Singapore
`shellaemode@gmail.com`

March 15, 2014

Webpage of SHELL:
`http://www1.spms.ntu.edu.sg/~syllab/CAESAR/shell`

Contents

1 Specification **2**
1.1 Parameters 2
1.2 Recommended parameter sets 2
1.3 Authenticated encryption mode SHELL 3
1.3.1 Notations 3
1.3.2 Structure Overview of SHELL 3
1.3.3 Preprocessing: Key Setup 7
1.3.4 PRP layer for nonce: CENC 7
1.3.5 Universal hash layer for associated data: PX-MAC 10
1.3.6 Universal hash layer for plaintext encryption: PX-Enc 13
1.3.7 PRP layer for ciphertext: parallel XEX ciphers 16
1.3.8 Tag generation 17
1.3.9 Extension to plaintexts with arbitrary length 17
1.4 SHELL-AES parameters 21

2 Security goals **22**

3 Security analysis **24**
3.1 Ideal Primitives 24
3.2 Security Definition 24
3.3 Security bound claims of SHELL in nonce-respecting environment 26
3.4 Security bound claims of SHELL in nonce-misuse environment . 27
3.5 Security of SHELL-AES 28

4 Features **29**

5 Design rationale **31**

6 Intellectual property **33**

7 Consent **34**

8 Acknowledgement **35**

Chapter 1

Specification

SHELL is a block-cipher-based authenticated encryption mode. We recommend AES [1] as the underlying block cipher of SHELL, since it is the most widely used block cipher nowadays, The specific instantiation of SHELL with AES is denoted as SHELL-AES in this report.

1.1 Parameters

SHELL uses the following parameters.

- **Block cipher** $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$.
- **A set of auxiliary permutations** $\{AP_i\}$: $AP_i : \{0,1\}^{k_i} \times \{0,1\}^n \to \{0,1\}^n$. Note that k_i is not necessarily equal to k.
- **Key length:** k bits or $2k$ bits.
- **Nonce length** ℓ_{nonce}**:** a positive integer such that $1 \leq \ell_{\text{nonce}} < n$.
- **Tag length** ℓ_{tag}: full size $\ell_{\text{tag}} = n$ (no truncation).
- **Frame width** w**:** a positive integer.
- **Maximum plaintext block length** ℓ_{pt}**:** $\ell_{\text{pt}} \leq 2^{n-\ell_{\text{nonce}}-1}$.
- **Maximum number of plaintexts under the same key:** $2^{\ell_{\text{nonce}}}$
- **The number of auxiliary permutations** d**:** a positive integer.

1.2 Recommended parameter sets

We recommend the following as the default parameters.

- **Block cipher** E**:** AES-128. Thus $k = 128$ and $n = 128$.

- **A set of auxiliary permutations** $\{AP_i\}$: all 4-round AES permutations with independent round keys.
- **Key length:** 128 bits.
- **Nonce length:** $\ell_{\text{nonce}} = 64$ or 80.
- **Tag length:** $\ell_{\text{tag}} = 128$.
- **Frame width:** $w = 256$.
- **#auxiliary permutations:** $4 \leq d \leq 8$.

1.3 Authenticated encryption mode SHELL

As an authenticated encryption mode, the encryption algorithm $\mathcal{E}_K$ of SHELL takes a nonce N, an associated data A and a plaintext M as input, and produces a ciphertext C that has the same length with M and a tag value T as output. That is $(C, T) \leftarrow \mathcal{E}_K(N, A, M)$. The decryption algorithm $\mathcal{D}_K$ of SHELL takes a nonce N, an associated data A, a ciphertext C and a tag T as input, and outputs either a plaintext M that has the same length with C if the tag is valid, or $\perp$ otherwise. That is $M/\perp \leftarrow \mathcal{D}_K(N, A, C, T)$.

1.3.1 Notations

Σ denotes the set $\{0,1\}$ and Σ^n denotes the n-bit set $\{0,1\}^n$. Moreover, $(\Sigma^n)^{\leq l}$ denotes the set of all binary sequences those have a bit length of a multiple of n and have a bit length at most nl, namely $(\Sigma^n)^{\leq l} = \bigcup_{i=1}^{l} \Sigma^{in}$. $(\Sigma^n)^+$ denotes the set of all binary sequences with a bit length of a multiple of n, and Σ^* denotes the set of all finite-length binary sequences.

For a finite set $\mathcal{X}$, $X \xleftarrow{\$} \mathcal{X}$ denotes that an element X is uniformly seleted from the set $\mathcal{X}$. For $X_1, X_2 \in \Sigma^*$ and have equal length, $X_1 \oplus X_2$ denotes the bitwise XOR. For $X_1, X_2 \in \Sigma^*$, $X_1 \| X_2$ or simply $X_1 X_2$, denote their concatenation. For $X \in \Sigma^*$, $|X|$ denotes its bit length. For a finite set $\mathcal{X}$, $\|\mathcal{X}\|$ denotes its carnality.

For a bit string X and an integer s with $s \leq |X|$, $\mathsf{msb}_s(X)$ denotes the s MSBs of X, and $\mathsf{lsb}_s(X)$ denotes the s LSBs of X.

Throughout this report, the block sizes of permutations and block cipher are fixed as n. Then for $X \in \Sigma^{n\ell}$ with $\ell \geq 1$, $X_1 \ldots X_\ell \xleftarrow{n} X$ denotes that X is partitioned into n-bit blocks such that $X_1 \| \cdots \| X_\ell = X$.

1.3.2 Structure Overview of SHELL

We present a high-level overview of the structure of SHELL, which is also shown in Figure 1.1. We will omit the description of the decryption procedure since it can be trivially obtained from the encryption procedure.

The nonce N is input to a Pseudo-Random-Permutation (PRP) layer CENC [11], which maps nonce to random strings with a provable security beyond the birthday bound. The output consists of two values (S, F). The first value S has the same length with plaintext M, and is XORed to the plaintext: $I \leftarrow S \oplus M$. The second value F is one-block long, and will be used in the tag generation algorithm. Moreover, the checksum of I is computed as $I_1 \oplus I_2 \oplus \cdots \oplus I_t$, where $I_1 I_2 \ldots I_t \overset{n}{\leftarrow} I$, and the checksum value is used in the tag generation.

The associated data A is input to a message authentication code PX-MAC, which is newly designed by us and utilizes universal hash functions. The output is denoted by V and V is one block long.

Then, V and I are input to an encryption layer called PX-Enc, which shares the same basic primitives with PX-MAC and thus is also universal-hash-based. The outputs consists of two values (U, Z). The first value U is one block long, and is used in the tag generation. The second value Z has the same length with I, and is used to produce the ciphertext.

Then I is input to a PRP layer that utilizes parallel XEX tweakable ciphers [23]. The output is the ciphertext C.

Finally, F, the checksum of I and U are input to tag generation, and a tag T is produced.

The above structure overview also well explains why we named it SHELL as shown in Figure 1.2. Our design consists of three layers. The upper and the lower layers utilize strong cryptographic primitive like pseudo-random-permutations, e.g., full-round AES. The middle layer utilizes much weaker cryptographic primitive like differentially-uniform permutations, e.g., four-round AES with independent round keys.

Moreover, we would like to regard the universal hash layer, more precisely PX-MAC and PX-Enc, as the main novelty of this design (to our best knowledge), or in other words the *meat* of SHELL.

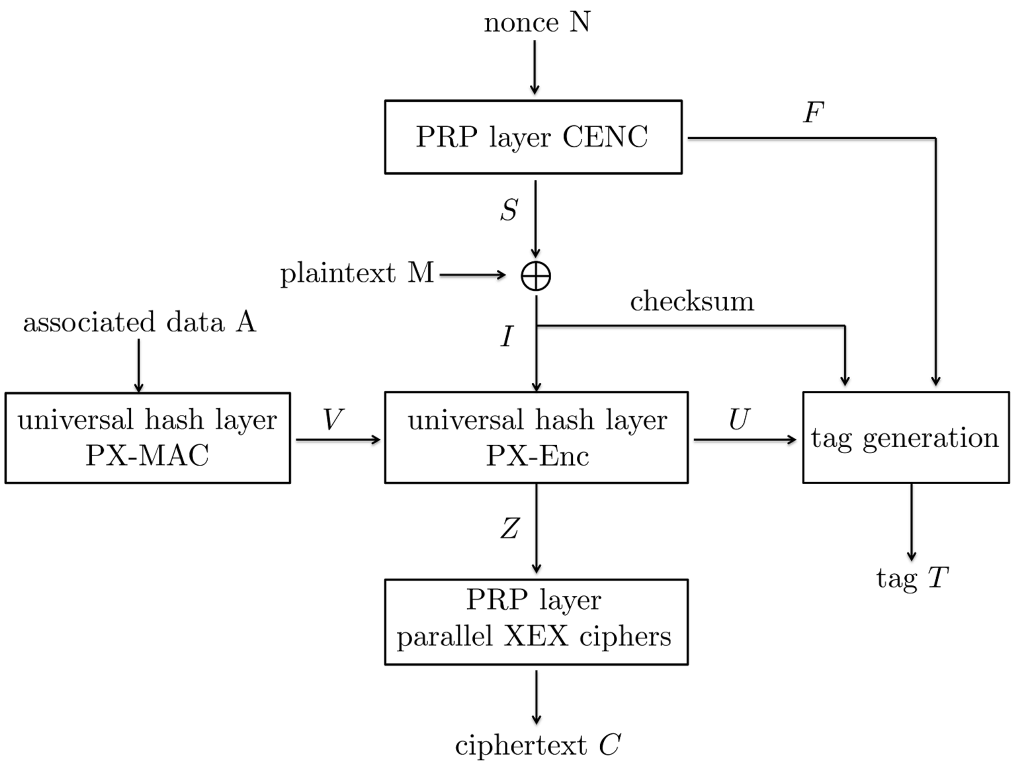

Figure 1.1: Structure Overview of SHELL

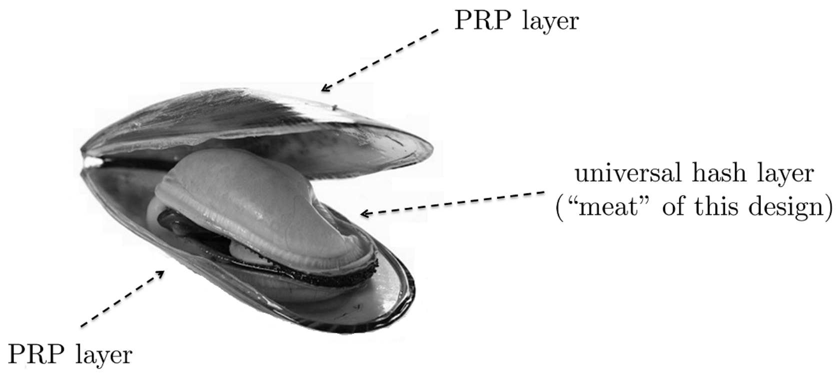

Figure 1.2: The Name "SHELL"

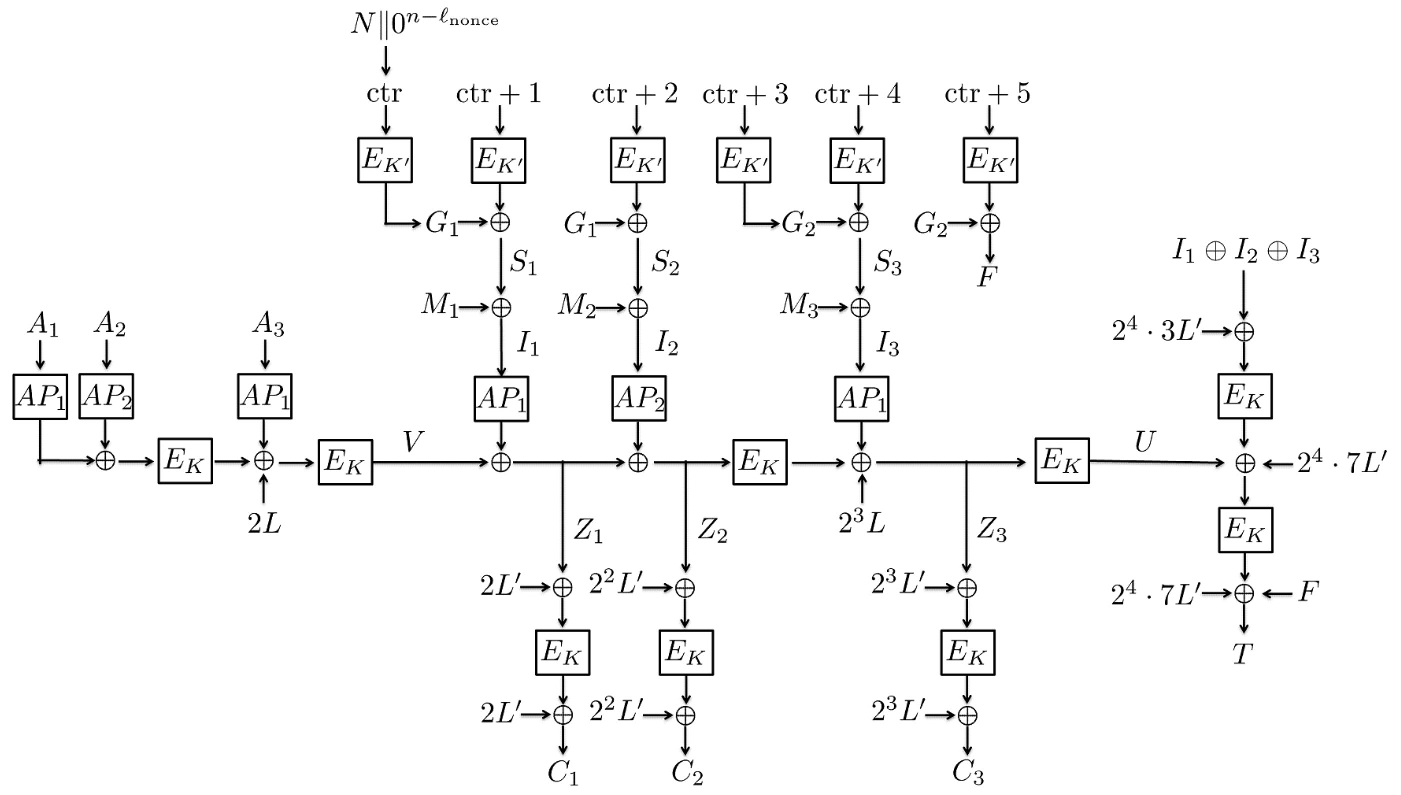

Figure 1.3: A simple example of SHELL with parameters $w = 2$ and $d = 2$. Associated data A is three full blocks $A_1A_2A_3$ and plaintext M is three full blocks $M_1M_2M_3$. During the computation, internal values are $S = S_1S_2S_3$, $I = I_1I_2I_3$, and $Z = Z_1Z_2Z_3$.

In the next sections, we will describe the specification of each layer.

1.3.3 Preprocessing: Key Setup

SHELL utilizes a block cipher E, and needs to key the block cipher E twice by using two keys K and K'.

Recall that SHELL uses a set of d auxiliary permutations denoted as $\{AP_1, AP_2, \ldots, AP_d\}$. Throughout this report, we use K_i to denote the secret key of AP_i for $1 \leq i \leq d$. We note that $|K_i|$ is not necessarily equal to $|K|$.

For PX-MAC and PX-Enc, in total d mask keys denoted as $\{K_1^{\text{mask}}, K_2^{\text{mask}}, \ldots, K_d^{\text{mask}}\}$ are necessary in order to construct universal hash function from the auxiliary permutations. Moreover, a subkey L is necessary for PX-MAC to distinguish messages with multiple block length from those with a length that is not a multiple of n.

Finally, a subkey L' is utilized to tweak ciphers following XEX method.

Key Setup. All these key materials are derived from E_K as detailed below.

- $K' = K \oplus \texttt{0xF0F0} \cdots \texttt{F0}$, if single key K is used as recommended. We assume that K has an integer number of bytes.
- $L = E_K(0)$.
- $SK = K_1 \| K_2 \| \cdots \| K_d$ is the first $|SK|$ bits of $E_K(1) \| E_K(2) \| \cdots \| E_K(a)$, where $a = \lceil |SK|/n \rceil$.
- $K^{\text{mask}} = K_1^{\text{mask}} \| \cdots \| K_d^{\text{mask}} = E_K(a+1) \| \cdots \| E_K(a+d)$.
- $L' = E_K(1^n)$.

1.3.4 PRP layer for nonce: CENC

CENC (Cipher-based ENCryption) is a nonce-based encryption mode proposed by Iwata [11]. It has improved the security of counter (CTR) encryption mode beyond the birthday bound, and meanwhile has also remained most advantage features of CTR mode including highly efficiency, fully parallel and etc.

CENC uses two parameters: a nonce length ℓ_{nonce} with $1 \leq \ell_{\text{nonce}} < n$, and a frame width w. Moreover, for each nonce value N, CENC(N) is restricted to make at most $2^{n-\ell_{\text{nonce}}}$ block cipher calls.

Let ℓ_m denote the block length of plaintext M, which is $\ell_m = \lceil |M|/n \rceil$. Then CENC($N$) produces $\ell_m + 1$ blocks long string Q. Moreover, to produce every w blocks of the string Q, CENC(N) needs to make $w + 1$ block cipher calls to E_K. So we have that

$$\frac{(\ell_m + 1)(w + 1)}{w} \leq 2^{n-\ell_{\text{nonce}}} \Longrightarrow \ell_m \leq \frac{w 2^{n-\ell_{\text{nonce}}}}{w + 1}$$

Thus, SHELL restricts the maximum block length of plaintext ℓ_{pt} as

$$\ell_{\mathrm{pt}} \leq 2^{n-\ell_{\mathrm{nonce}}-1}.$$

The specification of CENC is described in Fig 1.4. Nonce N is firstly padded to a full block (n bits) by adding '0's, which is denoted as $\mathtt{ctr}$. To produce the $(i+1)$-th frame of the string Q, where i counts from 0, CENC computes a value $G_i \leftarrow E_{K'}(\mathtt{ctr}')$, where $\mathtt{ctr}' = \mathtt{ctr}+i(w+1)$, and then computes the i-th frame as $S_{iw+1}\|S_{iw+2}\|\cdots\|S_{iw+w}$, where $S_{iw+j} = E_{K'}(\mathtt{ctr}' + j) \oplus G_i$ for $1 \leq j \leq w$. Let $Q = S_1\|S_2\|\cdots\|S_{\ell_m+1}$.

Note that the last frame may not be w blocks, if ℓ_m is not a multiple of w.

Let S be the first $|M|$ bits of Q, and F be the last n bits of Q, that is $S = \mathsf{msb}_{|M|}(Q)$ and $F = \mathsf{lsb}_n(Q)$.

Finally CENC(N) outputs (S, F).

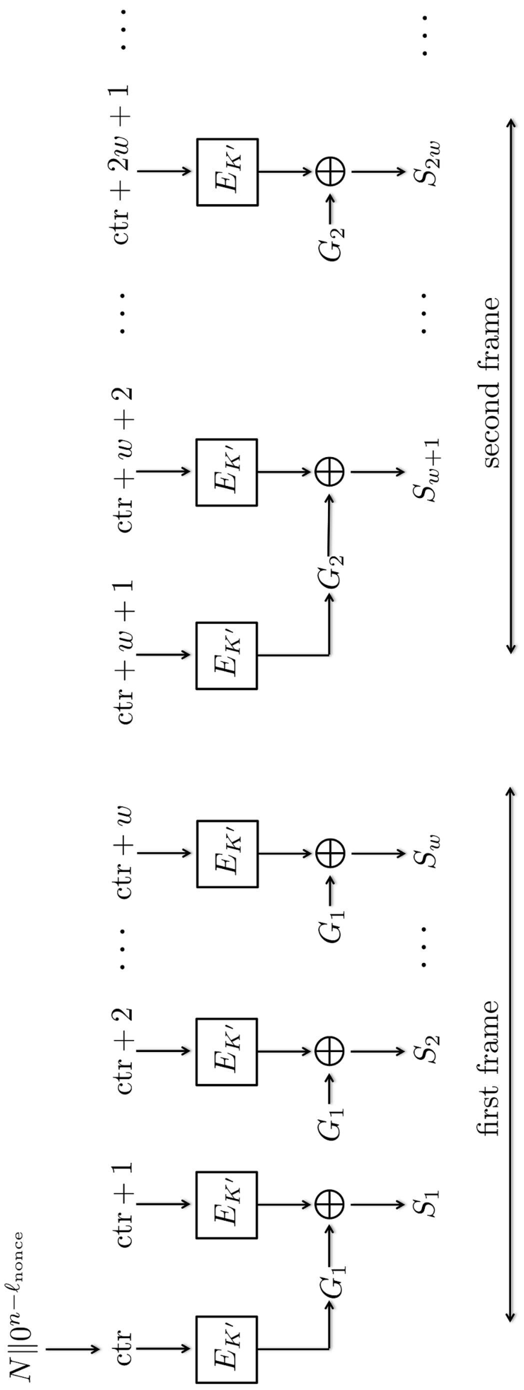

Figure 1.4: The CENC mode [11].

1.3.5 Universal hash layer for associated data: PX-MAC

PX-MAC is a new message authentication code (MAC) designed by us. It utilizes a block cipher E_K and a set of d auxiliary permutations $\{AP_1, AP_2, \ldots, AP_d\}$. Without of generality, assume that each AP_i is keyed, and denote its underlying secret key by K_i. Denote the candidate space of K_i by $\mathcal{K}_i$. AP_{i,K_i} is to denote that AP_i is keyed by K_i. Define $SK \overset{\text{def}}{=} K_1 \| K_2 \| \cdots \| K_d$, and denote its candidate space as $\mathcal{SK}$.

Auxiliary permutations. Each auxiliary permutation should have a small maximum expected differential probability (MEDP). Denote MEDP of the auxiliary permutation AP_i as ϵ_i for $1 \leq i \leq d$. Let ϵ be $\max\{\epsilon_1, \epsilon_2, \ldots, \epsilon_d\}$.

If an auxiliary permutation is unkeyed, then it should have a small value of maximum differential probability (MDP).

Parallel-auxiliary-permutation-Xor (PX). We define a keyed function called PX that maps $(\Sigma^n)^{\leq d}$ to Σ^n.

PX keyed function is built based on the set of auxiliary permutations. It uses two keys. One key is SK, which we recall is the secret keys $K_1 \| K_2 \ldots \| K_d$ for the auxiliary permutations $\{AP_1, AP_2, \ldots, AP_d\}$. The other key is K^{mask} which is used to mask the inputs to the auxiliary permutations. More precisely, K^{mask} is partitioned into n-bit blocks:

$$K_1^{\text{mask}} K_2^{\text{mask}} \ldots K_d^{\text{mask}} \overset{n}{\leftarrow} K^{\text{mask}},$$

such that $K_1^{\text{mask}} \| K_2^{\text{mask}} \| \ldots \| K_d^{\text{mask}} = K^{\text{mask}}$, and K_i^{mask} is used to mask the input for auxiliary permutation AP_i for $1 \leq i \leq d$.

The computation procedure on an input $X \in (\Sigma^n)^{\leq d}$ is as follow, which is also illustrated in Figure 1.5. It firstly partitions X into n-bit blocks: $X_1 X_2 \ldots X_t \overset{n}{\leftarrow} X$, where $|X| = tn$ and $1 \leq t \leq d$. then it applies the auxiliary permutation AP_{i,K_i} to compute the i-th block X_i: $Y_i = AP_{i,K_i}(X_i \oplus K_i^{\text{mask}})$. Finally it computes $Y = \bigoplus_{i=1}^{t} Y_i$ and outputs the value Y.

Arbitrary-input-length universal hash PX-UH. We define an arbitrary-input-length universal hash function called PX-UH, which is built by using the PX keyed function and the pseudo-random-permutation E_K. The algorithm is illustrated in Figure 1.6.

The computation procedure of PX-AU on an input $X \in \Sigma^*$ is as follows.

Firstly, if $|X|$ is not a multiple of n, then X is padded to have a length of multiple of n by $\mathsf{pad}(X) = X \| 10^{n-1-s}$, where $s = |X| \mod n$. Otherwise, $\mathsf{pad}(X) = X$.

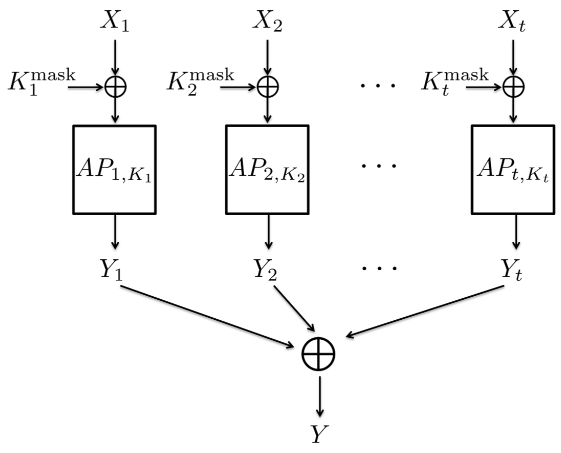

Figure 1.5: PX keyed function with domain $(\Sigma^n)^{\leq d}$. Thus $1 \leq t \leq d$

Then $\mathsf{pad}(X)$ is partitioned into dn-bit frames, which is denoted as $W_1 W_2 \ldots W_{\ell_f} \overset{dn}{\leftarrow} \mathsf{pad}(X)$, where $\ell_f = \lceil |\mathsf{pad}(X)|/(dn) \rceil$. Note that the last frame W_{ℓ_f} may be shorter than dn bits if $|\mathsf{pad}(X)|$ is not a multiple of dn, that is $W_{\ell_f} \in (\Sigma^n)^{\leq d}$. These frames are input to PX keyed function in a parallel manner:

$$Y_i = \mathrm{PX}(W_i), \text{ for } 1 \leq i \leq \ell_f.$$

Then these Y_i's are hashed in a CBC-like sequential manner (more precisely, CMAC [8] or OMAC [12]):

$$\begin{aligned} J_1 &\longleftarrow 0^n; \\ J_{i+1} &\longleftarrow E_K(Y_i \oplus J_i), \text{ for } 1 \leq i \leq \ell_f - 1. \end{aligned}$$

Finally, if the original input X before padding has a length of a multiple of n, the value $J = J_{\ell_f} \oplus Y_{\ell_f} \oplus (2 \cdot L)$ is outputted, where we recall that $L = E_K(0^n)$ and the multiplication "$\cdot$" is over $\mathrm{GF}(2^n)$. Otherwise, the value $J = J_{\ell_f} \oplus Y_{\ell_f} \oplus (2^2 \cdot L)$ is outputted.

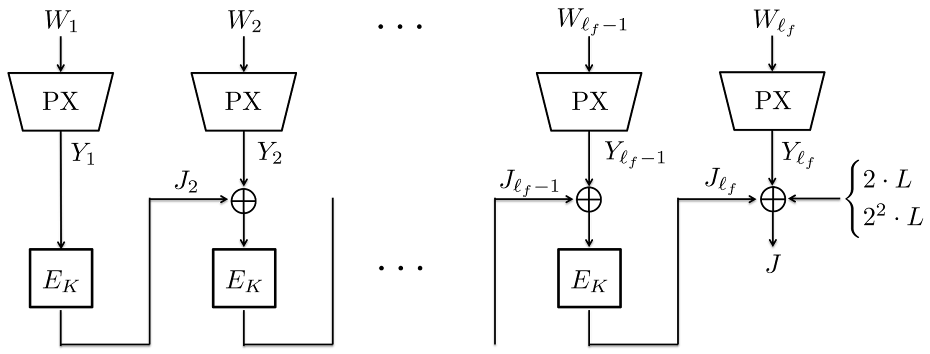

Figure 1.6: The universal hash PX-UH

PX-MAC. Now we describe the completed specification of PX-MAC, which utilizes the block cipher E_K and the universal function PX-UH. We recall that all the key materials L, SK and K^{mask} used in PX-UH are derived from the block cipher E_K.

Key Setup. Derive the values of L, SK and K^{mask} from E_K, which is also illustrated from Figure 1.7. Let $a = \lceil |SK|/n \rceil$.

- $L = E_K(0)$;
- $SK = K_1 \| \cdots \| K_d$ is the first $|SK|$ bits of $E_K(1) \| \cdots \| E_K(a)$;
- $K^{\text{mask}} = K_1^{\text{mask}} \| \cdots \| K_d^{\text{mask}} = E_K(a+1) \| \cdots \| E_K(a+d)$.

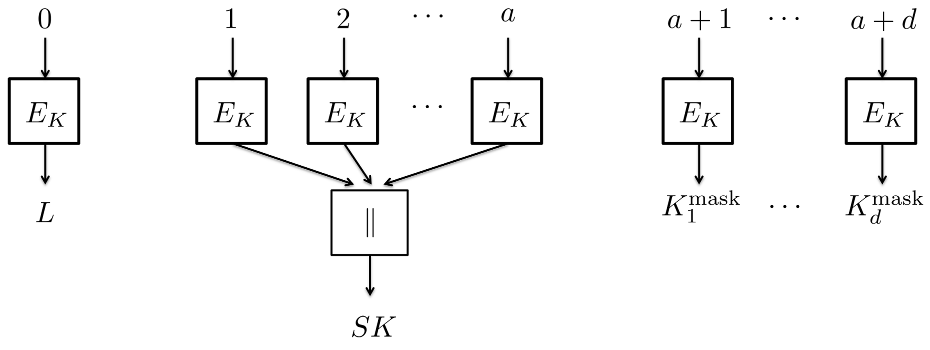

Figure 1.7: The Key Setup of PX-MAC

Tag Computation. For an input $X \in \Sigma^*$, firstly apply PX-UH to hash X, and get J =PX-UH(X). Then compute $T = E_K(J)$, and finally output T as the tag of X.

In SHELL, the procedure of computing associated data A is illustrated in Figure 1.8. The value V is outputted.

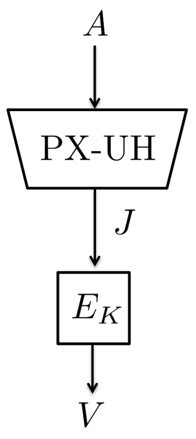

Figure 1.8: The computation procedure of associated data in SHELL

1.3.6 Universal hash layer for plaintext encryption: PX-Enc

We note that the following three sections focus on the case that plaintext $M \in (\Sigma^n)^+$. After these three sections, we will discuss about the extension of plaintexts to Σ^* in Section 1.3.9.

Firstly, plaintext M is XORed with the output S of CENC(N), that is $I \leftarrow M \oplus S$. Then, I and V, which is the output of PX-MAC(A), are input to a universal hash layer, which is very similar with PX-MAC and called PX-Enc.

PX*. We change the keyed function PX, and let the internal values be outputted. The new function is denoted as PX*, which also utilizes the set of auxiliary permutations. All the notations below follows those used in the specification of the keyed function PX.

PX* takes a pair of values (H, X) as input, where $H \in \Sigma^n$ and $X \in \Sigma^{dn}$. It outputs a pair of values (H', Y), where H is n bits long, and Y has the same bit length with X, namely $|Y| = |X|$. The algorithm is depicted in Fig. 1.9.

The computation procedure on input (H, X) is as follows. Firstly, PX* divides X into n-bit blocks: $X_1X_2 \ldots X_d \overset{n}{\leftarrow} X$. Then it computes

$$Y_1 = H \oplus AP_{1,K_1}(X_1 \oplus K_1^{\text{mask}}),$$
$$Y_i = Y_{i-1} \oplus AP_{i,K_i}(X_i \oplus K_i^{\text{mask}}), \text{ for } 2 \leq i \leq d.$$

Let $Y = Y_1\|Y_2\|\cdots\|Y_d$, and $H' = Y_d$. Finally PX* outputs (H', Y).

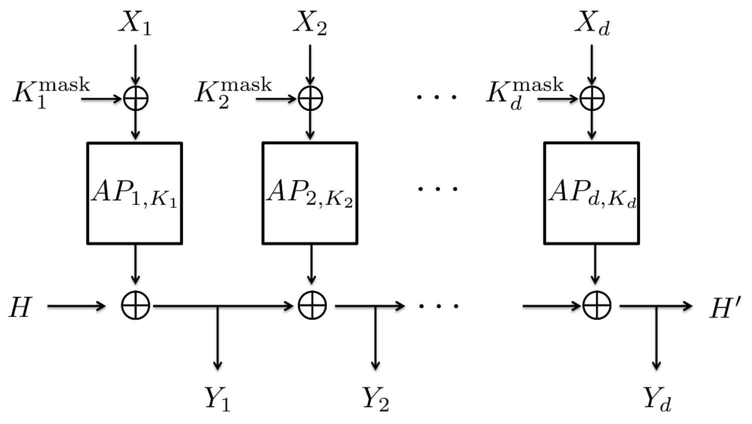

Figure 1.9: The PX* algorithm.

PX-Enc. We build an encryption algorithm called PX-Enc, which utilizes PX* and the block cipher E_K.

In the structure of SHELL, PX-Enc takes (V, I) as input, and $V \in \Sigma^n$ and $S \in (\Sigma^n)^+$. It outputs a pairs of values (U, Z), where U is n bits long and Z has the same length with I, namely $|Z| = |I|$. The algorithm of PX-Enc is illustrated in Fig. 1.10 and 1.11.

The computation procedure on input (V, I) is as follows. Note that I has the same block length with the plaintext M, and thus $|I| = |M|$. Firstly, PX-Enc divides I into dn-bit frames $W_1 W_2 \ldots W_{\ell_{fm}} \overset{dn}{\leftarrow} I$, where $\ell_{fm} = \lceil \ell_m/(dn) \rceil$. Note that the last frame $W_{\ell_{fm}}$ may be short of d blocks, if $|I|$ is not a multiple of dn. Then PX-Enc computes V and $W_1\|\cdots\|W_{\ell_{fm}-1}$ as below.

$$\begin{cases} (U_1, Z_1) & \leftarrow \text{PX*}(W_1, V) \\ V_1 & \leftarrow E_K(U_1) \end{cases}$$

$$\begin{cases} (U_i, Z_i) & \leftarrow \text{PX*}(W_i, V_{i-1}) \\ V_i & \leftarrow E_K(U_i) \end{cases} \quad \text{for} \quad 2 \le i \le \ell_{fm} - 1.$$

Then the computation of the value $V_{\ell_{fm}-1}$ and the last frame $W_{\ell_{fm}}$ is as follows, which is also shown in Fig. 1.11. Let $X_1 X_2 \ldots X_t \overset{n}{\leftarrow} W_{\ell_{fm}}$. The process is exactly the same with PX* for each block except for the last block X_t, where an extra XOR is introduced. More precisely, $Y_t = Y_{t-1} \oplus AP_{t,K_t}(X_t \oplus K_t^{\text{mask}}) \oplus (2^2 \cdot L)$. Let $Z_{\ell_{fm}} = Y_1 \| Y_2 \| \cdots \| Y_t$, and $U = E_K(Y_t)$.

Let $Z = Z_1 \| Z_2 \| \cdots \| Z_{\ell_{fm}}$. Finally PX-Enc outputs the pair (U, Z).

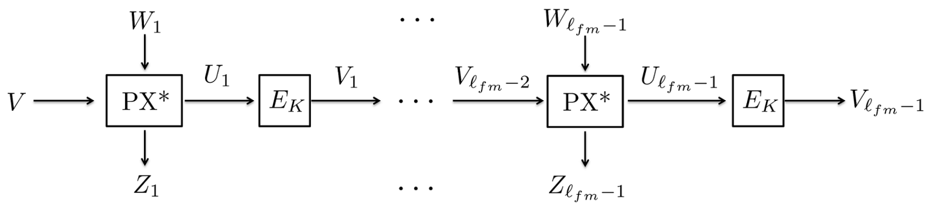

Figure 1.10: The PX-Enc algorithm for computing (V, I) except the last frame of I. $W_1 \dots W_{\ell_{fm}} \overset{dn}{\leftarrow} I$.

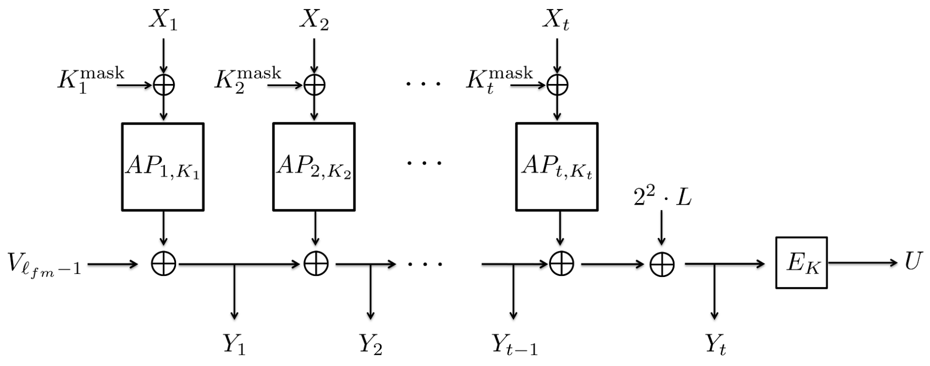

Figure 1.11: The PX-Enc algorithm for computing the last frame of I. $X_1 \| \cdots \| X_t \overset{dn}{\leftarrow} W_{\ell_{fm}}$ and $1 \leq t \leq d$.

1.3.7 PRP layer for ciphertext: parallel XEX ciphers

This layer takes the output Z of PX-Enc as input, and produces the ciphertext C. It utilizes a set of tweakable block ciphers in a parallel manner. These tweakable block ciphers are constructed from E_K following the doubling method by using the subkey L' to derive many distinct masks. Recall that $L' = E_K(1^n)$.

XEX cipher [23]. We can derive a block cipher E'_K from E_K by using a secret mask Δ as follows. Let X be any input from Σ^n.

$$E'_K(X) \overset{\text{def}}{=} E_K(X \oplus \Delta) \oplus \Delta.$$

Interestingly, E'_K behaves like an independent block cipher from E_K (up to the birthday bound).

Rogaway has proposed an efficient approach called the doubling method [23] to enable one to derive a large set of masks from just one secret mask. More precisely, the masks are derived as $2^\alpha 3^\beta 7^\gamma L'$ by changing α, β and γ.

It is important to note that the irreducible polynomial $f(\mathtt{x})$ should be chosen carefully so that we get a large set of distinct masks. For the case $n = 128$, e.g., SHELL-AES, we choose $f(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$, which has been proven to satisfy the requirement for $\alpha \in [-2^{108}, 2^{108}]$ and $\beta, \gamma \in [-2^7, 2^7]$ [23].

Parallel XEX ciphers. The computation procedure for an input $Z \in (\Sigma^n)^+$ is as follows, which is also illustrated in Fig. 1.12. Recall that $|Z| = |M|$. Firstly, divide Z to n-bit blocks $Z_1 \| Z_2 \| \cdots \| Z_{\ell_m} \overset{n}{\leftarrow} Z$. Then, the i-th block of ciphertext denoted as C_i for $1 \le i \le \ell_m$ is computed as

$$C_i = E_K(Z_i \oplus 2^i L') \oplus 2^i L',$$

Let $C = C_1 \| C_2 \| \cdots \| C_{\ell_m}$. Finally this layer outputs C as the ciphertext for M.

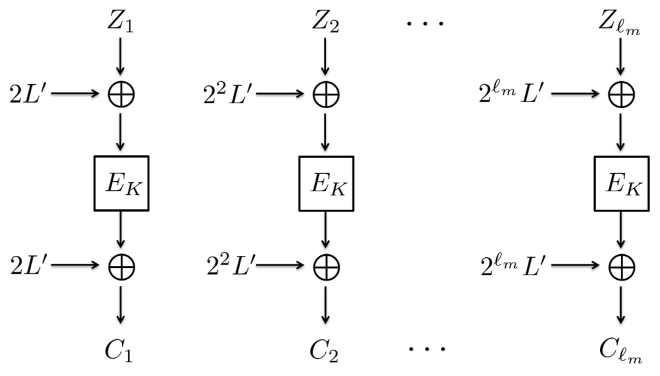

Figure 1.12: Parallel XEX ciphers

1.3.8 Tag generation

The tag generation algorithm takes as the checksum value of I, which we recall is the value of XORing the output S of CENC(N) and the plaintext M, the output F of CENC(N), and the output U of PX-Enc(V, I), where we recall that V is the output value of PX-MAC(A). Recall that all these three inputs are n bits long. It outputs an n-bit tag. The algorithm is illustrated in Fig. 1.13.

The computation procedure is as follows. Firstly, divide I to n-bit blocks. $I_1 \dots I_{\ell_m} \overset{n}{\leftarrow} I$. Then compute as follows.

$$\begin{aligned} Y &\leftarrow E_K(I_1 \oplus I_2 \oplus \cdots \oplus I_{\ell_m} \oplus (2^{\ell_m+1} \cdot 3L')) \oplus U, \\ T &\leftarrow E_K(Y \oplus (2^{\ell_m+1} \cdot 7L')) \oplus (2^{\ell_m+1} \cdot 7L') \oplus F. \end{aligned}$$

Finally the tag generation algorithm outputs T as the tag.

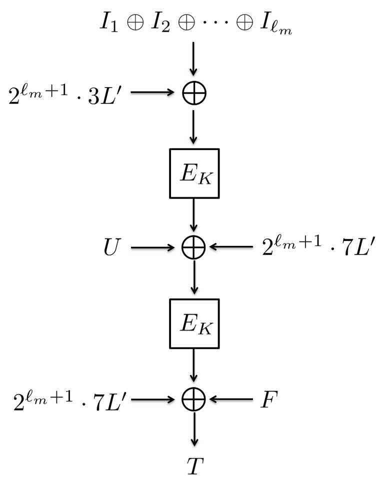

Figure 1.13: The tag generation algorithm

1.3.9 Extension to plaintexts with arbitrary length

Let $\mathcal{E}_K(\cdot)$ denote previously described encryption procedure for plaintexts $M \in (\Sigma^n)^+$. In this section, we extend the encryption algorithm of SHELL to handle plaintxts $M \in \Sigma^*$. More precisely, we present an encryption procedure $\mathcal{E}'_K(\cdot)$ for the plaintexts M such that $|M|$ is not a multiple of n. In the rest of this section, we will always assume that $|M|$ is not a multiple of n. The decryption procedure of $\mathcal{E}'_K(\cdot)$ is omitted since it can be trivially derived from $\mathcal{E}'_K(\cdot)$.

$\mathcal{E}'_K$ uses previous proposed techniques including tag splitting [9] and XLS [20], which has been also used in other authenticated encryption such as TC1-3 [25], MCOE [9], COPA [2] and POET [17].

$\mathcal{E}'_K$ for case $|M| < n$. It takes the output value V of PX-MAC(A), the value I obtained by XORing the output value S of CENC(N) and M, and the output value F of CENC(N) as inputs. Note that $|I|$ is the same with the plaintext $|M|$, and let $s = |X|$. The algorithm is illustrated in Fig. 1.14. We omit the description of the computation procedure of (C, T), and refer it to the Fig. 1.14. After the pair (C, T) is obtained, the s MSBs of C, that is $\mathsf{msb}_s(C)$, is outputted as the ciphertext for M. the $n - s$ LSBs of C and the s MSBs of T are concatenated, that is $\mathsf{lsb}_{n-s}(C)\|\mathsf{msb}_s(T)$, is outputted as the tag for M.

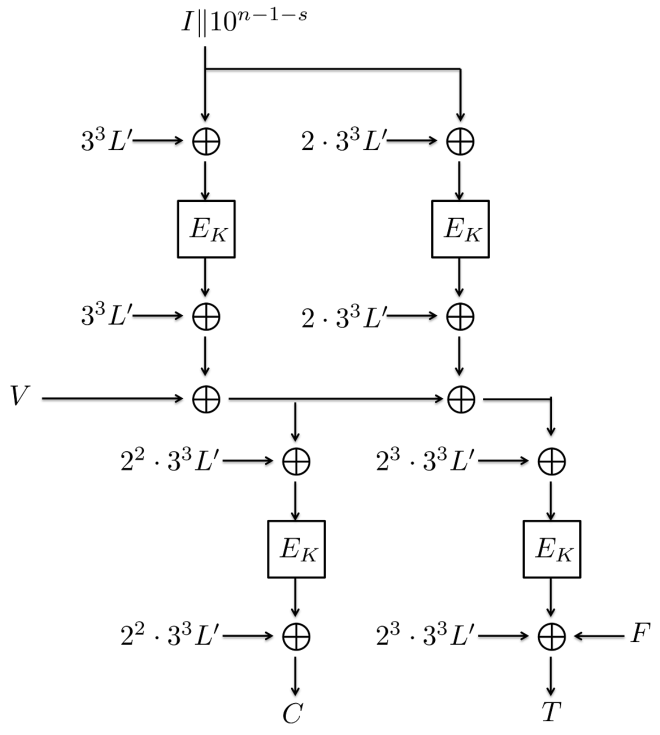

Figure 1.14: The encryption procedure for plaintext M with $|M| < n$. Let $s = |M|$. $\mathsf{msb}_s(C)$ and $\mathsf{lsb}_{n-s}(C)\|\mathsf{msb}_s(T)$ will be outputted as the ciphertext and the tag respectively.

$\mathcal{E}'_K$ for case $|M| > n$. We construct a tweakable block cipher that accepts inputs X with $n < |X| < 2n$, following the XLS method [20]. This tweakable block cipher is denoted as XLS_{ℓ_m}, where we recall that $\ell_m = \lceil |M|/n \rceil$. Let $Y = \mathrm{XLS}_{\ell_m}(X)$, and it always holds that $|Y| = |X|$, for any input X with $n < |X| < 2n$. Let E'_K be defined as follows.

$$E'_K(X) \overset{\text{def}}{=} E_K(X \oplus (2^{\ell_m} \cdot 3^2 L')) \oplus (2^{\ell_m} \cdot 3^2 L').$$

The specification of XLS_{ℓ_m} built from E'_K is illustrated in Fig. 1.15 and 1.16. For more details about the XLS specification, we refer to [20].

The computation procedure of $\mathcal{E}'_K$ is as follows. Firstly, divide M into n-bit blocks $M_1 M_2 \dots M_{\ell_m} \xleftarrow{n} M$. Let $M' = M_1 \| M_2 \| \cdots \| M_{\ell_m - 1}$. Note that $1 \leq |M_{\ell_m - 1}| < n$. Then apply the previously defined encryption $\mathcal{E}_K$ for plaintexts from $(\Sigma^n)^+$ to process (N, A, M') and obtain a pair (C', T'). That is

$$(C', T') \leftarrow \mathcal{E}_K(N, A, M').$$

Then apply XLS_{ℓ_m} to $T' \| I_{\ell_m}$ as below,

$$Y \leftarrow \text{XLS}_{\ell_m}(T' \| I_{\ell_m}).$$

Let $C = C' \| \mathsf{msb}_{|M_{\ell_m}|}(Y)$ and $T = \mathsf{lsb}_n(Y)$. Finally C is outputted as the ciphertext for M, and T is outputted as the tag.

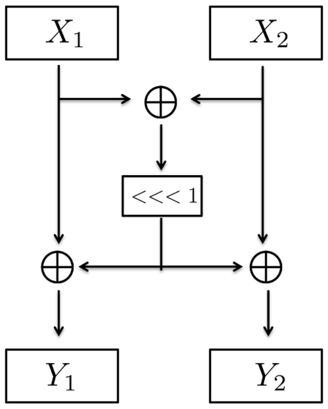

Figure 1.15: The MIX function [20]. Let input $X = X_1 \| X_2$ and $|X_1| = |X_2|$. $Y = Y_1 \| Y_2$ is the output. $\lll 1$ is the left bit rotation by one bit.

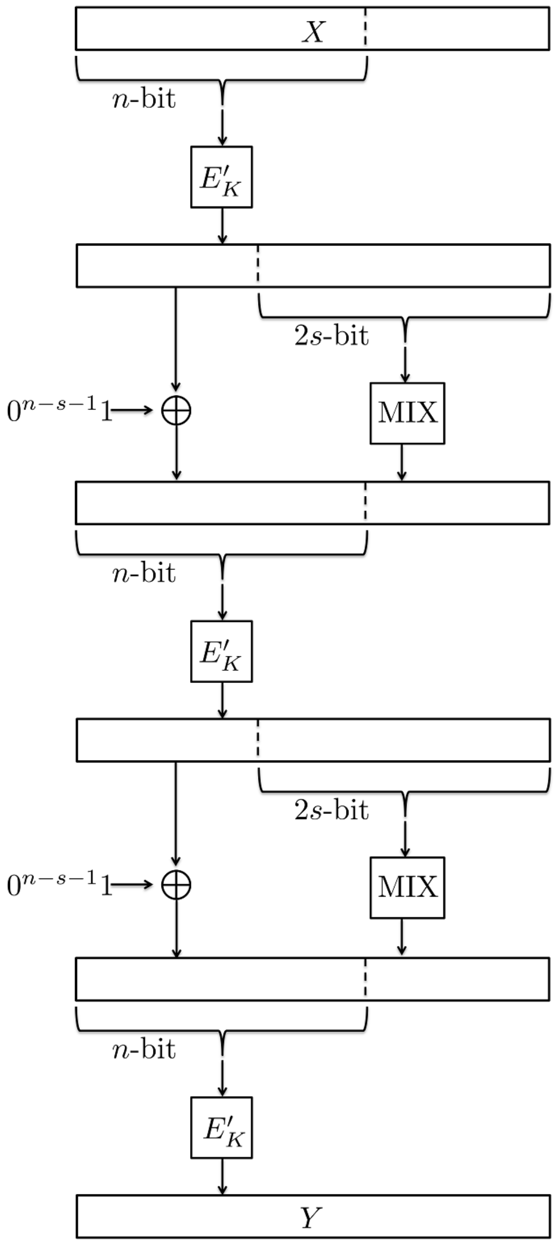

Figure 1.16: The XLS method [20]. Let $|X| = n + s$ and $1 \leq s < n$.

1.4 SHELL-AES parameters

We mainly specify the auxiliary permutations in SHELL-AES. Each AP_i with $1 \leq i \leq d$ is a four-round AES with independent round keys, which is shown in Fig. 1.17. SB, SR and MC are SubBytes, ShiftRows and MixColumns operations defined in AES.

Each key K_i for AP_i consists of three 128-bit round keys. Thus the number of block cipher calls in the pre-processing phase, that is the key setup, is in total $4d + 2$.

The multiplication over GF(2^{128}) uses the the irreducible polynomial $f(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$.

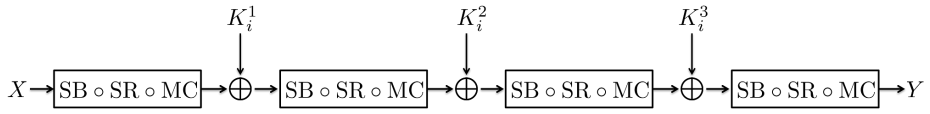

Figure 1.17: The auxiliary permutation AP_i for SHELL-AES. The secret key is $K_i = K_i^1 \| K_i^2 \| K_i^3$.

Chapter 2

Security goals

SHELL uses the public message number as nonce. The secret message number has length of 0 bits.

The security goal of SHELL is twofold. On one hand, in the nonce-respecting environment where no nonce is repeated to the encryption algorithm, we expect SHELL can provide a provable security beyond the birthday bound. On the other hand, in the nonce-misuse environment where the same nonce can be used for distinct queries to the encryption algorithm, we expect SHELL can still provide a provable security bound, which we choose to be lower than the birthday bound. This is because a higher security bound usually comes with an efficiency loss. We have to take the tradeoff into consideration.

The security goal of SHELL mode is provided in Table 2.1. The definitions of security notions are detailed in Section 3.2.

It is important to note that to our best knowledge, the privacy security bound naturally implies (at least) the same security bound for the confidentiality of the plaintext, and the authenticity security bound naturally implies (at least) the same bound for the integrity of nonce and the integrity of associated data.

Table 2.1: The security goal of SHELL mode. The value ϵ is defined as $\max\{\epsilon_1, \epsilon_2, \ldots, \epsilon_d\}$, where ϵ_i denotes MEDP of AP_i. Note that n, w, d and AP_i are parameters of SHELL.

	nonce-respecting environment	nonce-misuse environment
Security notions	(bit security)	(bit security)
Privacy	$(2n - \log_2 w)/3$	$\frac{1}{2}\log_2 \frac{1}{d\epsilon}$
Authenticity	$(2n - \log_2 w)/3$	$\frac{1}{2}\log_2 \frac{1}{d\epsilon}$

Now we discuss about SHELL-AES with recommended parameters. $n = 128$,

$w = 256$, and $4 \leq d \leq 8$. For four-round AES with independent round keys as the recommended auxiliary permutations, we have that $\epsilon \leq 2^{-113}$ [14, 15]. Thus we claim the security of SHELL-AES as shown in Table 2.2.

Table 2.2: The security goal of SHELL-AES with recommended parameters

Security notions	nonce-respecting environment (bit security)	nonce-misuse environment (bit security)
Privacy	80	55
Authenticity	80	55

We point out that SHELL is not decryption-misuse resistant mode.

Chapter 3

Security analysis

In this chapter, we provide security bound claims of SHELL. The proofs will be put online in the webpage of SHELL soon.

3.1 Ideal Primitives

Random permutation. Let $\mathsf{Perm}(n)$ be the set of all n-bit permutations. We write that a permutation P is a n-bit random permutation if it is uniformly selected from $\mathsf{Perm}(n)$, namely $P \overset{\$}{\leftarrow} \mathsf{Perm}(n)$.

Random function. Let $\mathsf{Func}(n)$ be the set of all functions that take $N\|A\|M$ as input and output a string Y such that $|Y| = |M| + n$. We write that a function $\mathcal{R}$ is a random function if it is uniformly selected from $\mathsf{Func}(n)$, namely $\mathcal{R} \overset{\$}{\leftarrow} \mathsf{Func}(n)$.

Random online cipher. Let $\mathsf{OPerm}(n)$ be the set of all ciphers that take $N\|A\|M$ as input, and output a string Y such that $|Y| = |M| + n$, which have an additional feature such that the i-th block of Y only depends on N, A and the first i blocks of M. We write that a random online cipher $\mathcal{P}$ if it is uniformly selected from $\mathsf{OPerm}(n)$, namely $\mathcal{P} \overset{\$}{\leftarrow} \mathsf{OPerm}(n)$.

3.2 Security Definition

The definitions below follow previous works [16, 3, 4, 24, 22, 25]

Security for block cipher. An adversary $\mathcal{A}$ is an algorithm that outputs a bit. E_K is a block cipher with a secret key value K. $E_K^{\pm}$ consist of both E_K and its inverse. P and P' are random permutations. $P^{\pm}$ consists of both P and its inverse. $K' = K \oplus \texttt{0xF0F0}\cdots\texttt{F0}$. We define

$$\begin{aligned}
\mathbf{Adv}_E^{\text{prp}}(\mathcal{A}) &\overset{\text{def}}{=} \left|\Pr\left[\mathcal{A}^{E_K(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{P(\cdot)} \Rightarrow 1\right]\right|,\\
\mathbf{Adv}_E^{\text{sprp}}(\mathcal{A}) &\overset{\text{def}}{=} \left|\Pr\left[\mathcal{A}^{E_K^{\pm}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{P^{\pm}(\cdot)} \Rightarrow 1\right]\right|,\\
\mathbf{Adv}_E^{\text{prp-rka}}(\mathcal{A}) &\overset{\text{def}}{=} \left|\Pr\left[\mathcal{A}^{E_K(\cdot),E_{K'}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{P(\cdot),P'(\cdot)} \Rightarrow 1\right]\right|,\\
\mathbf{Adv}_E^{\text{sprp-rka}}(\mathcal{A}) &\overset{\text{def}}{=} \left|\Pr\left[\mathcal{A}^{E_K^{\pm}(\cdot),E_{K'}^{\pm}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{P^{\pm}(\cdot),P'^{\pm}(\cdot)} \Rightarrow 1\right]\right|.
\end{aligned}$$

Then we define that

$$\begin{aligned}
\mathbf{Adv}_E^{\text{prp}}(t,q) &\overset{\text{def}}{=} \max_{\mathcal{A}} \mathbf{Adv}_E^{\text{prp}}(\mathcal{A}),\\
\mathbf{Adv}_E^{\text{sprp}}(t,q) &\overset{\text{def}}{=} \max_{\mathcal{A}} \mathbf{Adv}_E^{\text{sprp}}(\mathcal{A}),\\
\mathbf{Adv}_E^{\text{prp-rka}}(t,q) &\overset{\text{def}}{=} \max_{\mathcal{A}} \mathbf{Adv}_E^{\text{prp-rka}}(\mathcal{A}),\\
\mathbf{Adv}_E^{\text{sprp-rka}}(t,q) &\overset{\text{def}}{=} \max_{\mathcal{A}} \mathbf{Adv}_E^{\text{sprp-rka}}(\mathcal{A}),
\end{aligned}$$

where the maximum is taken over all adversaries $\mathcal{A}$ whose time complexity is at most t and whose total number of queries is at most q.

Security for authenticated encryption. Let $\mathcal{E}_K$ and $\mathcal{D}_K$ be the encryption oracle and the decryption oracle of an authenticated encryption, respectively. Let $\mathcal{R}$ be a random function, and $\mathcal{P}$ be a random online cipher. Let $\perp$ be an oracle that always outputs $\perp$ for all queries.

Nonce respecting environment. We only need to consider nonce-respecting adversaries. We say that an adversary is nonce-respecting if he never repeats a nonce during the interaction with the encryption oracle $\mathcal{E}_K$.

In such environment, we define the privacy of the authentication encryption as below.

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{E}}^{\text{priv}}(\mathcal{A}) &= \left|\Pr\left[K \overset{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{\mathcal{E}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathcal{R}(\cdot)} \Rightarrow 1\right]\right|\\
\mathbf{Adv}_{\mathcal{E}}^{\text{priv}}(t,q,\ell,\sigma) &= \max_{\mathcal{A}} \mathbf{Adv}_{\mathcal{E}}^{\text{priv}}(\mathcal{A}),
\end{aligned}$$

were the maximum is taken over all adversaries $\mathcal{A}$ whose time complexity is at most t, number of queries is at most q, maximum block length of a query is at most ℓ, and total number of blocks in all queries is at most σ.

Next, we define the authenticity of the authenticated encryption as below.

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{E}}^{\text{auth}}(\mathcal{A}) &= \left|\Pr\left[K \overset{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{\mathcal{E}(\cdot),\mathcal{D}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathcal{R}(\cdot),\perp} \Rightarrow 1\right]\right|\\
\mathbf{Adv}_{\mathcal{E}}^{\text{auth}}(t,q,\ell,\sigma) &= \max_{\mathcal{A}} \mathbf{Adv}_{\mathcal{E}}^{\text{auth}}(\mathcal{A}),
\end{aligned}$$

were the maximum is taken over all adversaries $\mathcal{A}$ whose time complexity is at most t, number of queries at most q, maximum block length of a query at most ℓ, and total number of blocks in all queries at most σ.

Nonce miuse environment. The adversary is allowed to use the same nonce for distinct queries during the interaction with the encryption oracle $\mathcal{E}_K$. An extreme case is that he uses the same nonce for all his queries to $\mathcal{E}_K$.

In such environment, we define the privacy of the authentication encryption as below.

$$\mathbf{Adv}_{\mathcal{E}}^{\text{priv}}(\mathcal{A}) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathcal{P}(\cdot)} \Rightarrow 1\right]\right|$$
$$\mathbf{Adv}_{\mathcal{E}}^{\text{priv}}(t,q,\ell,\sigma) = \max_{\mathcal{A}} \mathbf{Adv}_{\mathcal{E}}^{\text{priv}}(\mathcal{A}),$$

were the maximum is taken over all adversaries $\mathcal{A}$ whose time complexity is at most t, number of queries at most q, maximum block length of a query at most ℓ, and total number of blocks in all queries at most σ.

Next, we define the authenticity of the authenticated encryption as below.

$$\mathbf{Adv}_{\mathcal{E}}^{\text{auth}}(\mathcal{A}) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}(\cdot),\mathcal{D}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathcal{P}(\cdot),\perp} \Rightarrow 1\right]\right|$$
$$\mathbf{Adv}_{\mathcal{E}}^{\text{auth}}(t,q,\ell,\sigma) = \max_{\mathcal{A}} \mathbf{Adv}_{\mathcal{E}}^{\text{auth}}(\mathcal{A}),$$

were the maximum is taken over all adversaries $\mathcal{A}$ whose time complexity is at most t, number of queries at most q, maximum block length of a query at most ℓ, and total number of blocks in all queries at most σ.

3.3 Security bound claims of SHELL in nonce-respecting environment

The security claim on the privacy of SHELL in nonce-respecting environment is as follows. We assume that $K' = K \oplus \texttt{0xF0F0}\cdots\texttt{F0}$.

Theorem 3.3.1 *Let $\mathcal{E}(\cdot)$ be the encryption of* SHELL, *and E be its underlying block cipher. Let c be the number of block cipher calls in the pre-processing phase that is the key setup. Then we have that*

$$\mathbf{Adv}_{\mathcal{E}}^{\text{priv}}(t,q,\ell,\sigma) \le \mathbf{Adv}_{E}^{\text{prp-rka}}(t', 3\sigma + c + 2q) + \frac{O(w\sigma^3)}{2^{2n}}.$$

where $t' = O(t + \sigma + c)$.

Note that the term $\mathbf{Adv}_E^{\text{prp-rka}}(t', 3\sigma{+}c{+}2q)$ will be replaced by $\mathbf{Adv}_E^{\text{prp}}(t', 3\sigma + c + 2q)$ if SHELL uses two independent keys K and K'.

Next, the security claim of the authenticity of SHELL in nonce-respecting environment is as follows.

Theorem 3.3.2 *Let $\mathcal{E}(\cdot)$ and $\mathcal{D}(\cdot)$ be the encryption and the decryption of* SHELL, *and E be its underlying block cipher. Let the set of its underlying auxiliary permutations be $\{AP_1, AP_2, \ldots, AP_d\}$, where the maximum expected differential probability of AP_i is denoted by ϵ_i. Let $\epsilon = \max\{\epsilon_1, \epsilon_2, \ldots, \epsilon_d\}$. Let c be the number of block cipher calls in the pre-processing phase that is the key setup. Then we have that*

$$\mathbf{Adv}_{\mathcal{E}}^{\text{auth}}(t, q, \ell, \sigma) \le \mathbf{Adv}_E^{\text{sprp-rka}}(t', 3\sigma + c + 2q) + O(\frac{w\sigma^3}{2^{2n}} + q\epsilon + \frac{c\ell q}{2^n}).$$

where $t' = O(t + \sigma + c)$.

Note that the term $\mathbf{Adv}_E^{\text{sprp-rka}}(t', 3\sigma{+}c{+}2q)$ will be replaced by $\mathbf{Adv}_E^{\text{sprp}}(t', 3\sigma + c + 2q)$ if SHELL uses two independent keys K and K'.

3.4 Security bound claims of SHELL in nonce-misuse environment

The security claim on the privacy of SHELL in nonce-misuse environment is as follows. We assume that $K' = K \oplus \texttt{0xF0F0}\cdots\texttt{F0}$.

Theorem 3.4.1 *Let $\mathcal{E}(\cdot)$ be the encryption of* SHELL, *and E be its underlying block cipher. Let the set of its underlying auxiliary permutations be $\{AP_1, AP_2, \ldots, AP_d\}$, where the maximum expected differential probability of AP_i is denoted by ϵ_i. Let $\epsilon = \max\{\epsilon_1, \epsilon_2, \ldots, \epsilon_d\}$. Let c be the number of block cipher calls in the pre-processing phase that is the key setup. Then we have that*

$$\mathbf{Adv}_{\mathcal{E}}^{\text{priv}}(t, q, \ell, \sigma) \le \mathbf{Adv}_E^{\text{prp-rka}}(t', 3\sigma + c + 2q) + O(d\ell q^2\epsilon + \frac{\ell q^2}{2^n} + \frac{c\ell q}{2^n}).$$

where $t' = O(t + \sigma + c)$.

Note that the term $\mathbf{Adv}_E^{\text{prp-rka}}(t', 3\sigma{+}c{+}2q)$ will be replaced by $\mathbf{Adv}_E^{\text{prp}}(t', 3\sigma + c + 2q)$ if SHELL uses two independent keys K and K'.

Next, the security claim of the authenticity of SHELL in nonce-misuse environment is as follows.

Theorem 3.4.2 *Let $\mathcal{E}(\cdot)$ and $\mathcal{D}(\cdot)$ be the encryption and the decryption of* SHELL, *and E be its underlying block cipher. Let the set of its underlying auxiliary permutations be $\{AP_1, AP_2, \ldots, AP_d\}$, where the maximum expected* *differential probability of AP_i is denoted by ϵ_i. Let $\epsilon = \max\{\epsilon_1, \epsilon_2, \ldots, \epsilon_d\}$. Let c be the number of block cipher calls in the key setup. Then we have that*

$$\mathbf{Adv}^{\text{Auth}}_{\mathcal{E}}(t, q, \ell, \sigma) \le \mathbf{Adv}^{\text{sprp-rka}}_{E}(t', 3\sigma + c + 2q) + O(d\ell q^2\epsilon + \frac{\ell q^2}{2^n} + \frac{c\ell q}{2^n} + q^2\epsilon).$$

where $t' = O(t + \sigma + c)$.

Note that the term $\mathbf{Adv}^{\text{sprp-rka}}_{E}(t', 3\sigma+c+2q)$ will be replaced by $\mathbf{Adv}^{\text{sprp}}_{E}(t', 3\sigma+c+2q)$ if SHELL uses two independent keys K and K'.

3.5 Security of SHELL-AES

MEDP of auxiliary permutations. For four-round AES with independent round keys, its maximum expected differential probability has been proven upper bounded by 2^{-113} [15, 14]. Thus we have $\epsilon \le 2^{-113}$ for SHELL-AES.

Security of AES. Ever since the publication of the AES (or Rijndael [7]) block cipher, cryptographers have been carrying out continuous evaluation on its security, and so many analysis result papers have been published. Particularly, attacks have been found on full AES-192 with a complexity of 2^{176} [5] and on full AES-256 with a complexity of 2^{99} [6, 5]. The best distinguisher on AES-128 reaches 9 rounds [10], namely one round shorter than full version.
As we can easily see, these analysis results on AES are either with higher complexities than our security goal, or have not even reach to full rounds yet.

Security of SHELL-AES. Let $w = 256$ and $4 \le d \le 8$ as recommended. Then the number of block cipher calls in pre-processing $c \le 34$.

In nonce-respecting environment, we have that both the privacy security and the authenticity security after replacing AES by random permutations are around 80 bits.

In nonce-reuse environment, we have that both the privacy and the authenticity after replacing AES by random permutations are around 55 bits.

Chapter 4

Features

SHELL is mainly a software-oriented design. In hardware, we should choose a small value for d, which is the number of auxiliary permutations, in order to reduce the memory requirement for storing the necessary subkeys. For SHELL-AES with the largest recommended value $d = 8$, it takes around 0.5 KByte memory to store the subkeys.

Advantage over GCM-AES. There are mainly two advantage features of SHELL-AES compared to GCM-AES [18].

The first one is a higher provable security bound under the nonce-respecting environment. SHELL-AES has a security beyond the birthday bound (up to 2^{80}), while GCM-AES has a security up to the birthday bound (that is 2^{64}) [13]. The higher security of SHELL-AES is obtained with a little loss of efficiency. For each block of plaintext, SHELL-AES needs roughly 2.5 AES calls, while GCM-AES needs roughly one AES call and one multiplication over GF(2^{128}). A detailed performance report on SHELL-AES and more accurate efficiency comparison with GCM-AES will be put online soon in the SHELL webpage.

The second one is failure-friendly under the nonce misuse environment. It is possible and actually has already happened that a well-designed cryptographic protocol is later wrongly implemented by software engineers who may know very little about cryptography. Then the security of that protocol may be immediately and completely lost. As for nonce-based authenticated encryption protocols, a possible failure is the so-called nonce-misuse issue, that is the same nonce used to encrypt distinct messages. When such a failure occurs, GCM-AES immediately loses all the security and can be seriously and trivially attacked. On the other hand, SHELL-AES still holds a provable security bound, even under an extreme environment where the nonce is fixed as a constant and never changes for the encryption of all plaintexts.

Other features. Similarly with GCM [18], SHELL is an authenticated encryption mode. Thus compared with dedicated authenticated encryption algorithms, SHELL provides the user with the interface of choosing his/her own preferred underlying primitives, e.g., the block cipher.

The beyond-birthday-bound security of SHELL makes it a more suitable choice for applications, where a block cipher with a smaller block size (say 64 bits) is used.

Chapter 5

Design rationale

In this chapter, we write about the choices that we have made during the design procedure of SHELL.

We start with the design of a new message authentication code, which is expected to be provably secure and efficient in both serial implementation and parallel implementation. From the consideration of the efficiency in the serial implementation, we decide to utilize the auxiliary permutations, say four-round AES, Then from the consideration of the efficiency in the parallel implementation, we decide to compute these auxiliary permutations in a parallel manner. Finally in order to obtain provable security and meanwhile to have a small number of block cipher calls in pre-processing phase, we decide to compute the outputs of auxiliary permutations in a CBC-like sequential manner. Putting everything together, we come up with PX-MAC.

After that, we move to build a provable nonce-misuse-resistant authenticated encryption. We adopts the Encrypt-then-MAC framework. We decide to use a two-layer structure. The upper lay is built by modifying PX-MAC, where all the internal values of PX-MAC are outputted. This is because PX-MAC guarantees that the internal values of distinct queries will not collide up to some bound. Then we choose strong primitives such as pseudo-random permutations as the lower layer in order to transform these non-colliding strings to pseudo-random strings as the ciphertexts. Then we need to define a tag generation, and pay attention to the security of authenticity. In the end, we get a nonce-misuse-resistant authenticated encryption called PX-AE. PX-AE is SHELL without the first PRP layer CENC, and the nonce N is appended to associated data such as $A \leftarrow A\|N$.

Finally, we move to build an authenticated encryption mode that has twofold security goals including provable security beyond the birthday bound in nonce-respecting environment and failure-friend in the nonce-misuse environment. We come up with a framework that uses a stream cipher, which is proven indistinguishable from random oracle with a beyond security bound, and a nonce-misuse-resistant authenticated encryption. [1] The idea is to send nonce to the stream cipher and then use its output to mask the plaintext and the tag of the nonce-misuse-resistant authenticated encryption. Thanks to CENC [11], we got SHELL.

Remark. We tried to find an approach to avoid the operation of keying E by another value K' such that CENC layer also uses E_K, and the resulted mode can yet maintain a provable security beyond the birthday bound. But we failed. If someone can achieve it or has some interesting ideas, we are always happy and feel honored to hear it.

Consent. The designers of SHELL faithfully declare that we have not hidden any weaknesses in this cipher.

[1] It is important to note that we did not claim that this framework is always provably secure for general combinations. Usually one must pay particular attention to the authenticity.

Chapter 6

Intellectual property

There is no patent constraint relevant to the usage of SHELL to our best knowledge. If any of this information changes, the submitter will promptly (and within at most one month) announce these changes on the `crypto-competitions` mailing list.

Chapter 7

Consent

The submitter hereby consents 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 submitter understands 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 submitter understands 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 submitter acknowledges that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitter understands that if he disagrees with published analyses then he is expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitter understands that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

Chapter 8

Acknowledgement

The designer would like to thank the colleagues of Nanyang Technological University Jian Guo, Jérémy Jean, Ivica Nikolić and Thomas Peyrin for their valuable discussion and comments.

The designer also would like to thank Prof. Tetsu Iwata for serial lectures about authenticated encryption and many valuable and inspiring discussions during his visit to Nanyang Technological University.

The author also would like to thank Liting Zhang for valuable discussions during ASK13 workshop.

The designer also would like to thank the designers of PC-MAC [19], CMAC [8], CENC [11] and COPA [2]. These beautiful designs helped the designer a lot to accomplish the design of SHELL.

The designer is supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

Bibliography

[1] : Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce (November 2001)

[2] Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and Authenticated Online Ciphers. In Sako, K., Sarkar, P., eds.: ASIACRYPT (1). Volume 8269 of Lecture Notes in Computer Science., Springer (2013) 424–443

[3] Bellare, M., Kohno, T.: A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In Biham, E., ed.: EUROCRYPT 2003. Volume 2656 of LNCS., Springer (May 2003) 491–506

[4] Bellare, M., Rogaway, P., Wagner, D.: The EAX Mode of Operation. In Roy, B.K., Meier, W., eds.: FSE 2004. Volume 3017 of LNCS., Springer (February 2004) 389–407

[5] Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Matsui, M., ed.: ASIACRYPT 2009. Volume 5912 of LNCS., Springer (December 2009) 1–18

[6] Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and Related-Key Attack on the Full AES-256. In Halevi, S., ed.: CRYPTO 2009. Volume 5677 of LNCS., Springer (August 2009) 231–249

[7] Daemen, J., Rijmen, V.: AES Proposal: Rijndael (1998)

[8] Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (2007)

[9] Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In Canteaut, A., ed.: FSE 2012. Volume 7549 of LNCS., Springer (March 2012) 196–215

[10] Fouque, P.A., Jean, J., Peyrin, T.: Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128. In Canetti, R., Garay, J.A., eds.: CRYPTO 2013, Part I. Volume 8042 of LNCS., Springer (August 2013) 183–203

[11] Iwata, T.: New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. [21] 310–327

[12] Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC. In Johansson, T., ed.: FSE 2003. Volume 2887 of LNCS., Springer (February 2003) 129–153

[13] Iwata, T., Ohashi, K., Minematsu, K.: Breaking and Repairing GCM Security Proofs. In Safavi-Naini, R., Canetti, R., eds.: CRYPTO 2012. Volume 7417 of LNCS., Springer (August 2012) 31–49

[14] Keliher, L., Sui, J.: Exact Maximum Expected Differential and Linear Probability for 2-Round Advanced Encryption Standard (AES). IACR Cryptology ePrint Archive **2005** (2005) 321

[15] Keliher, L., Sui, J.: Exact maximum expected differential and linear probability for two-round Advanced Encryption Standard. IET Information Security **1**(2) (2007) 53–57

[16] Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM J. Comput. **17**(2) (1988) 373–386

[17] McGrew, D., Fluhrer, S., Lucks, S., Forler, C., Wenzel, J., Abed, F., List, E.: Pipelineable On-Line Encryption. In: Fast Software Encryption. (2014)

[18] McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In Canteaut, A., Viswanathan, K., eds.: INDOCRYPT 2004. Volume 3348 of LNCS., Springer (December 2004) 343–355

[19] Minematsu, K., Tsunoo, Y.: Provably Secure MACs from Differentially-Uniform Permutations and AES-Based Implementations. [21] 226–241

[20] Ristenpart, T., Rogaway, P.: How to Enrich the Message Space of a Cipher. In Biryukov, A., ed.: FSE 2007. Volume 4593 of LNCS., Springer (March 2007) 101–118

[21] Robshaw, M.J.B., ed.: FSE 2006. In Robshaw, M.J.B., ed.: FSE 2006. Volume 4047 of LNCS., Springer (March 2006)

[22] Rogaway, P.: Authenticated-Encryption With Associated-Data. In Atluri, V., ed.: ACM CCS 02, ACM Press (November 2002) 98–107

[23] Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In Lee, P.J., ed.: ASIACRYPT 2004. Volume 3329 of LNCS., Springer (December 2004) 16–31

[24] Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In: ACM CCS 01, ACM Press (November 2001) 196–205

[25] Rogaway, P., Zhang, H.: Online Ciphers from Tweakable Blockciphers. In Kiayias, A., ed.: CT-RSA 2011. Volume 6558 of LNCS., Springer (February 2011) 237–249