Fast Parallelizable Misuse-Resistant Authenticated Encryption

Low Latency (Decryption-Fast) SIV

Mustafa Khairallah

Department of Electrical and Information Technology
Lund University, Lund, Sweden
`khairallah@ieee.org`

Abstract. MRAE security is an important goal for many AEAD applications where the nonce uniqueness cannot be maintained and security risks are significant. However, MRAE schemes can be quite expensive. Two of the SoTA MRAE-secure schemes; Deoxys-II and AES-GCM-SIV rely on internal parallelism and special instructions to achieve competitive performance. However, they both suffer from the same bottleneck, they have at least one call to the underlying primitive that cannot be parallelized to any other call. Romulus-M and LM-DAE are two other more recent MRAE secure schemes based on TBCs that target low area hardware. However, they are unparallelizable so they are slower than their counterparts.
In this paper, we present two new AEAD modes and four instantiations based on Tweakable Block Ciphers. These new modes target equipping high-speed applications on parallel platforms with nonce misuse resistant AEAD (MRAE). The first mode, LLSIV, targets similar performance on single-core platforms to SCT-2, while eliminating the bottlenecks that make SCT-2 not fully parallelizable. The enhanced parallelism allows LLSIV to encrypt significantly more blocks on parallel platforms, compared to SCT-2, in the same amount of time. LLSIV is based on the NaT MAC, where each ciphertext block can itself be viewed as an instance of NaT when the plaintext is prepended with 0^n. The trade-off is that LLSIV requires the inverse function of the TBC. However, the inverse function is used only once per message and we demonstrate that for parallel implementations it represents a very small overhead.
We give an instantiation of LLSIV based on the SKINNY-128-384 TBC, and a pruned scheme, dubbed pLLSIV, which targets enhanced performance compared both SCT-2 and LLSIV on all platforms, while having reduced security claims. It relies on the recently popularized *prove-then-prune* methodology to take full advantage of the properties of LLSIV. This leads to a significant performance improvement, making pLLSIV even faster than online TBC-based schemes that are not MRAE-secure. Last but not least, we give an instantiation that uses the primitives used in AES-GCM-SIV: the PolyVal hash function and AES. Our instantiation is faster than AES-GCM-SIV on all platforms and have better bounds. On the other hand, it relies on the ideal cipher model as it uses the ICE2 TBC proposed as part of the Remus AEAD design.

The second mode we describe is LLDFV. It uses ideas from LLSIV combined the Decryption-Fast SIV (DFV) framework proposed recently by Minematsu. The goal is to reduce the number of calls to the TBC by one, while making the scheme as parallelizable as LLSIV. This makes the scheme faster that DFV on all platforms.

Keywords: AEAD · MRAE · TBC · SIV · Deoxys · Skinny.

1 Introduction

Authenticated Encryption with Associated Data (AEAD) has become one of the most important symmetric-key cryptographic primitives. However, providing privacy and authenticity, simultaneously, comes at a cost. Most widely used AEAD schemes, such as AES-GCM [21, 20], and the recently selected NIST stardardization candidate; Ascon [16], fall into the category known as Nonce-based AEAD (NAE). The user must provide an extra non-repeating input (nonce). This adds a significant implementation cost and widens the attack surface. Nonces may repeat due to bad implementations, fault attacks or (malicious) misconfiguration. In some cases, such as AES-GCM, repeating the nonce even once can lead to drastic consequences.

In 2006, Rogaway and Shrimpton [39] presented the Deterministic AEAD (DAE) security notion, which does not require any nonce. They also introduced a construction known as the Synthetic Initial Vector (SIV) construction. It has, ever since, become one of two blueprints for building DAE and nonce-Misuse-Resistant AE (MRAE), the other blueprint being the Encode-then-Encipher (EtE) framework [11]. The SIV construction works as follows: First, the plaintext and associated data (AD) are absorbed by a variable-input-length Pseudo-Random Function (PRF) to generate a block T of fixed length. Then, T is used both as an authentication tag and an IV for an IV-based encryption scheme. This process is depicted in Figure 1(a). Decryption uses a tag provided by the user as an IV for the IV-based decryption. Then, the unverified plaintext and AD are passed to the PRF and the outcome is compared to the provided tag.

The security of the SIV mode boils down to two main requirements: the PRF acts as a deterministic Message Authentication Code (MAC) whose outcome is indistinguishable from random tags and the encryption is an IND-CPA-secure encryption scheme. These arguments mean that the scheme also inherits the limitations of its components. Mainly, the scheme can only offer up to Birthday Bound (upBB) security with respect to the tag size[1]. This limitation has motivated a line of research into combining the concepts DAE and NAE into MRAE. The idea is to build a scheme that acts as an NAE scheme when the nonce uniqueness is ensured, and the security does not drop drastically when the nonce is repeated a small number of times.

[1] We should note that the LM-DAE mode proposed in 2020 [35] claims BBB security, but with respect to the block size n, where the tag size is $2n$. Hence, it does not contradict our statement.

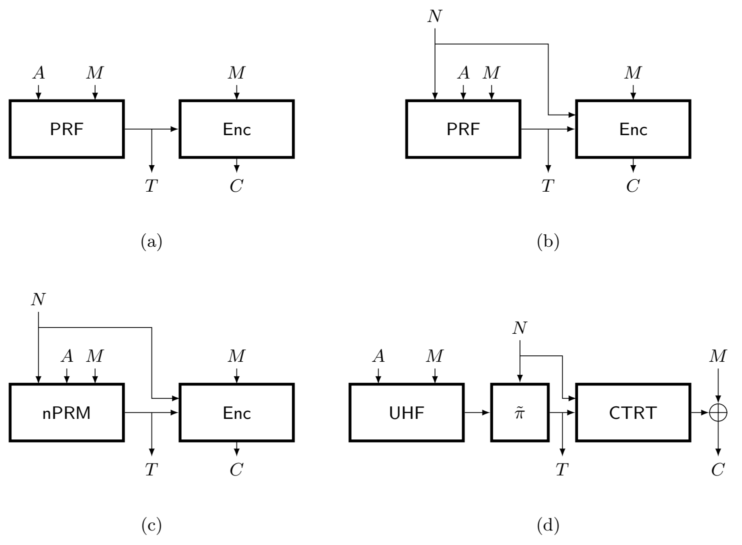

Fig. 1. The evolution of SIV-like constructions. (a) The original SIV construction. (b) The nSIV construction. (c) The nSIV construction using nonce-based Pseudo-Random MAC. (d) The SCT-2k mode.

A simple, yet ineffective, approach is to add the nonce as part of the AD. This does not lead to any gain in terms of security, but it can be useful if a DAE scheme needs to be used in an application that requires an NAE. In fact, this was done to transform the SUNDAE scheme [7] (D-A-E stands for Deterministic AE) into a nonce-based variant [8] compliant with the NIST lightweight standardization project.[2]

Another direction has been exemplified by the line of research surrounding the Deoxys-II AEAD scheme [30, 36, 31, 14], which is the winner of the defence-in-depth track of the CAESAR competition.[3] In [36], Peyrin and Seurin proposed the Synthetic Counter-in-Tweak (SCT) mode. Later, the designers of Deoxys modified it to SCT-2 [30] which uses the Nonce-as-Tweak (NaT) MAC [15] as the PRF and the CounTeR-in-Tweak (CTRT) IV-and-nonce-based encryption mode. The SCT-2 mode is depicted in Figure 1(d). A security proof for a two-key version of SCT-2 was given in [14] under the name GNSIV-N and we shall be using this version in our discussion. We shall be referring to this version as SCT-2k. The goal of these constructions is to use the properties of the Tweakable Block Cipher (TBC) to achieve what is known as *graceful degradation*: If the nonce is unique, the scheme has almost full Beyond Birthday Bound (BBB) security, while the security drops linearly with the maximum number of nonce repetitions, reaching upBB security if the nonce becomes a constant. This means that for

[2] https://csrc.nist.gov/projects/lightweight-cryptography
[3] https://competitions.cr.yp.to/caesar.html

applications where the nonce is only repeated a small number of times, due to faults or weak randomness, the security does not degrade significantly. In [31], the authors extended this concept to define the nonce-based SIV (nSIV) construction, depicted in Figures 1(b) and (c). They also generalized the security analysis of NaT to define the nonce-based Pseudo-Random MAC (nPRM) security notion, which is quite different from and more flexible than simple PRFs. These concepts were also used to design other nSIV-based schemes such as Romulus-M [27].

One of the main features of Deoxys-II that makes it appealing for high speed applications is its internal parallelism. The NaT scheme uses a Universal Hash Function (UHF) followed by a TBC. The UHF is implemented using the sum of TBCs construction used in PMAC-1 [38]. On the other hand, the CTRT mode process the counters and plaintext blocks completely in parallel. It was shown in several works that by exploiting the internal parallelism, the performance of Deoxys-II can be improved drastically [30–32]. In particular, the advantage of pipelined hardware accelerators of block ciphers have been demonstrated in multiple recent works, with sometimes upward of 50× speed-up compared to sequential implementations [44, 46, 26, 42, 40].

This work and the bottlenecks of SIV. Our goal in this paper is to take advantage of the internal parallelism the underlying encryption and hashing modes, and eliminate the bottlenecks of SIV/SCT-2k. Simultaneously, we want to take advantage of the recently popularized *prove-the-prune* methodology. The first bottleneck comes from the requirement of SIV/SCT-2k that the IV/tag T be indistinguishable from a random block. This requires transforming the hashed (A, M) pair using a PRF, as exemplified by the application of $\tilde{\pi}$ in Figure 1(d). This call to a fixed length PRF (in this case implemented using a TBC) cannot be done in parallel to either parts of the construction, and presents a significant bottleneck to speeding up instantiations of SIV/SCT-2k on parallel platforms, especially for short messages.

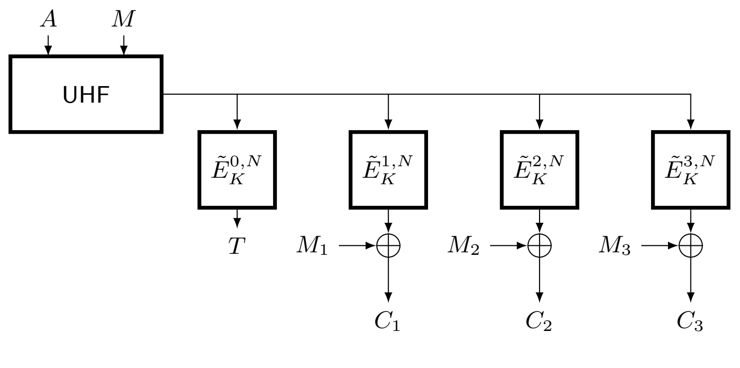

Fig. 2. The LLSIV with 3 plaintext block: $M = M_1 \| M_2 \| M_3$

The other technique that cannot be fully used in SIV-based schemes (including SCT-2k) is the *prove-then-prune* framework. This framework have become popular in the recent years with designs/constructions such as the Orthros PRF [9], the ForkCipher [3], the Iterate-Fork-Iterate (ifi) framework [2], and other forked constructions [19]. The basic idea of this framework is to design constructions that are proven secure when the underlying primitive is an idealized primitive, then argue that the way the primitive is used inside the construction limits the adversary's capability, so the number of rounds inside the primitive can be reduced. In the context of SIV, this can be done to the primitive calls inside the UHF, since the adversary does not observe their outputs and usually the outputs are XORed together. It was indeed done in one instance of the Estate family of AEAD algorithms [13]. However, this cannot be done to the calls that are part of the encryption phase of the algorithm without further studies and without increasing the overhead, since for these calls, the adversary can see, and choose, their inputs and outputs, with almost no restrictions.

To resolve both these bottlenecks, we propose the LLSIV mode of oeration, depicted in Figure 2. Rather than using the output of a PRF as the IV for the CTRT encryption mode, we show that it is sufficient to use the outcome of a UHF as an IV for a a CTRT-like mode. The first block of the CTRT mode is used as the tag. One could see that each output block is an instance of the NaT MAC. Hence, the proposed construction can be shown to achieve both confidentiality and authenticity. Without pruning, the construction has the same speed as SCT-2k on single-core platforms, but has faster encryption performance on parallel platforms. This comes at the cost of using the inverse function (decryption) to be able to compute the IV during decryption. Moreover, by using the ifi mindset, we can prune not only the UHF, but also the TBC calls. We can speed up the TBC calls by about 35%. We shall provide the security proofs, design rationale, performance comparison and practical instantiations of the proposed construction. We give instantiations based on the SKINNY-128-384 TBC and a pruned version of it. We also give an instantiation based on PolyVal and AES for comparison with AES-GCM-SIV.

Next we propose the LLDFV mode, depicted in Figure 3. It uses the same technique used to optimize SCT-2k, which can be also applied to optimize the DFV technique recently proposed by Minematsu [34], reducing the number of TBC calls in a TBC-based instantiation of DFV by one call, and allowing using the fast decryption speed up with the encryption speed up from reducing the number of calls and parallelism.

Outline. Section 2 provides the preliminaries needed to follow the paper. Section 3 provides the specification, security proofs and implementation results of LLSIV and its pruned variant pLLSIV. Section 5 discusses LLDFV and compares it to Minematsu's proposals [34]. Finally, the paper is concluded in Section 6.

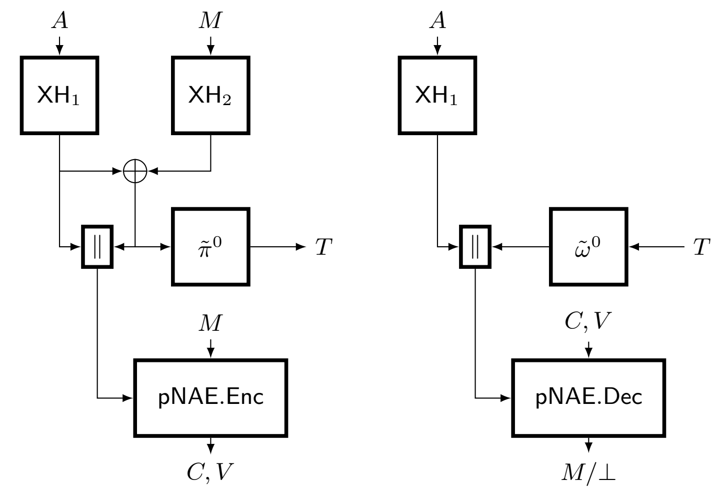

Fig. 3. The proposed LLDFV encryption (left) and decryption (right).

2 Preliminaries

Let $\{0,1\}^*$ be the set of all finite bit strings including the empty string ε. For $X \in \{0,1\}^*$, $|X|$ is its bit length. For an integer $i \geq 0$, $\{0,1\}^i$ is the set of all bit strings of length i bits and $\{0,1\}^{\leq i}$ is the set of all bit strings of length at most i bits. For an integer $l \leq 1$, $|X|_l$ is the length of $X \in \{0,1\}^*$ in l-bit blocks, *i.e.*, $|X|_l = \lceil |X|/l \rceil$ if $X \neq \varepsilon$ and $|X|_l = 1$ if $X = \varepsilon$. For two bit strings X and Y, $X\|Y$ is the concatenation of the two strings. 0^i is the string consisting of i zero bits and $1\|0^i$ is written as 10^i. For $X \in \{0,1\}^*$ with $|X| \geq i$, $\mathtt{msb}_i(X)$ is the leftmost i bits of X and $\mathtt{lsb}_i(X)$ is the rightmost i bits of X. Let X be a uniformly sampled bit string from the set $\mathcal{X}$, we write $X \overset{\$}{\leftarrow} \mathcal{X}$. For $X \in \{0,1\}^*$, $\mathtt{pad}_n(X)$ is the de-facto one-zero padding: Let $|X| \mod n = i$. Then, $\mathtt{pad}_n(X) = 10^{n-1}$ if $X = \varepsilon$, $\mathtt{pad}_n(X) = X$ if $i = 0$, and $\mathtt{pad}_n(X) = X\|10^{n-i-1}$, otherwise. Let $0 \leq i < 2^b$ be an integer, then $\bar{i}_b$ be the b-bit binary representation of i, such that if $i \neq j$, $\bar{i}_b \neq \bar{j}_b$. If b is understood from context, we write $\bar{i}$.

Authenticated Encryption with Associated Data (AEAD) We define the syntax of AEAD using nonce and AD, then we describe the special cases where either of these inputs are not available/not used. Let $\mathsf{NAE} = (\mathsf{NAE.Enc}, \mathsf{NAE.Dec})$ be an NAE scheme. NAE.Enc takes as input a key $K \in \mathcal{K}$ and a tuple $(N, A, M) \in \mathcal{N} \times \mathcal{A} \times \mathcal{M}$, and returns a ciphertext $C \in \mathcal{M}$ and a tag $T \in \mathcal{T}$, such that $|M| = |C|$. We shall define $\mathcal{M} = \{0,1\}^*$ and $\mathcal{T} = \{0,1\}^\tau$ for a fixed small τ. NAE.Dec takes as input a key $K \in \mathcal{K}$ and a tuple $(N, A, C, T) \in \mathcal{N} \times \mathcal{A} \times \mathcal{M} \times \mathcal{T}$, and returns either an invalid symbol $\bot$ or a plaintext $M \in \mathcal{M}$.

If $\mathcal{A} = \phi$, we shall follow the notation of [34] and refer to the scheme as pNAE. If $\mathcal{N} = \phi$, then we refer to the scheme as DAE. If $\mathcal{N} \neq \phi$ but the nonce can be repeated, then we refer to the scheme as MRAE.

MRAE Security Let MRAE be an AEAD scheme. We define two security notions. The privacy notion is the indistinguishability of ciphertexts from random string. Let $\mathbf{A}$ be an nonce-repeating adversary against MRAE. $\mathbf{A}$ has access to one oracle and makes queries on the form of (N, A, M) and does not repeat queries. $\mathbf{A}$ makes q_e queries. Given $N \in \mathcal{N}$, N appears in at most μ queries. If $\mathcal{N} = \phi$, then $\mu = q_e$. Let \$ be the oracle that takes as input the tuple (N, A, M) and returns $C \xleftarrow{\$} \{0,1\}^{|M|}$ and $T \xleftarrow{\$} \{0,1\}^{\tau}$. Then, the advantage of $\mathbf{A}$ against the nonce-misuse privacy of MRAE is defined as

$$\mathrm{Adv}^{\mathtt{nm-priv}}_{\mathsf{MRAE}}(\mathbf{A}) \overset{\mathrm{def}}{=} |\Pr[K \xleftarrow{\$} \mathcal{K} : \mathbf{A}^{\mathsf{MRAE.Enc}} \to 1] - \Pr[\mathbf{A}^{\$} \to 1]|$$

Let $\mathbf{B}$ be an nonce-repeating adversary against MRAE. $\mathbf{B}$ has access to two oracles. It makes q_e queries the form of (N, A, M) to its first oracle and q_d queries the form of (N, A, C, T) to its second oracle. $\mathbf{B}$ does not repeat queries and does not request queries from its second oracles that have been previously generated by the first oracle. Then, the advantage of $\mathbf{B}$ against the nonce-misuse authenticity of MRAE is defined as

$$\mathrm{Adv}^{\mathtt{nm-auth}}_{\mathsf{MRAE}}(\mathbf{B}) \overset{\mathrm{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathbf{B}^{\mathsf{MRAE.Enc},\mathsf{MRAE.Dec}} \text{ forges } \mathsf{MRAE}]$$

where $\mathbf{B}$ forges MRAE means that for any query $\mathbf{B}$ makes to its second oracle, it receives a plaintext $M^\star \neq \perp$.

When understood in context, we will use $\mathtt{nr}-$ to refer to the same security notions when the adversary is nonce respecting and $\mathtt{d}-$ when the scheme is deterministic, *i.e.*, $\mathcal{N} = \phi$.

Tweakable Block Cipher (TBC) A TBC is a mapping $\tilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ such that for any choice of $K \in \mathcal{K}$, and any choice of $T_w \in \mathcal{T}$, $Y \leftarrow \tilde{E}(K, T_w, X)$ is a permutation of $\{0,1\}^n$. Let $\mathsf{perm}_{t,n}$ be the set of all tweakable permutations of $\{0,1\}^n$ with tweak space $\mathcal{T}$. We say $\tilde{\pi} : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is a tweakable permutation if for every choice of $T \in \mathcal{T}$, $Y \leftarrow \tilde{\pi}(T, X)$ is a permutation of $\{0,1\}^n$, and $\tilde{\omega} : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is the inverse tweakable permutation such that $\tilde{\omega}^T(\tilde{\pi}^T(M)) = M$. We write $\tilde{E}^T_K(X)$ to indicate $\tilde{E}(K, T, X)$. A (q_e, q_d, t)-adversary $\mathbf{A}$ against the strong Tweakable Pseudo-Random Permutation (sTPRP) security of $\tilde{E}$ is an algorithm that has oracle access to a tweakable permutation of $\{0,1\}^n$ as well as its inverse, makes q_e queries to the first oracle, q_d queries to the second oracle and runs in time at most t. It outputs a single bit. The advantage of $\mathbf{A}$ against the sTPRP security of $\tilde{E}$ is given by

$$\mathrm{Adv}^{\mathtt{stprp}}_{\tilde{E}}(\mathbf{A}) \overset{\mathrm{def}}{=} |\Pr[K \xleftarrow{\$} \mathcal{K} : \mathbf{A}^{\tilde{E}^{\cdot}_K,(\tilde{E}^{\cdot}_K)^{-1}} \to 1] - \Pr[\tilde{\pi} \xleftarrow{\$} \mathsf{perm}_{t,b} : \mathbf{A}^{\tilde{\pi},\tilde{\omega}} \to 1]|.$$

Let $\mathbf{B}$ be an adversary that has has only access to the first oracle and makes q_e queries, then

$$\mathrm{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathbf{B}) \overset{\mathrm{def}}{=} |\Pr[K \xleftarrow{\$} \mathcal{K} : \mathbf{A}^{\tilde{E}^{\cdot}_K} \to 1] - \Pr[\tilde{\pi} \xleftarrow{\$} \mathsf{perm}_{t,n} : \mathbf{A}^{\tilde{\pi}} \to 1]|.$$

If $\tilde{\pi}$ is selected randomly from $\mathsf{perm}_{t,n}$, we call it a Tweakable Uniformly Random Permutation (TURP). An ideal cipher is a TBC is a that is sampled uniformly from the set of all possible TBCs with the same domain.

Universal Hash Function (UHF) Let $H : \mathcal{K}_h \times \mathcal{X} \to \mathcal{Y}$ be a keyed hash function with key space $\mathcal{K}_h$, input space $\mathcal{X}$ and output space $\mathcal{Y}$. Let $\epsilon > 0$. We say H is ϵ-Almost Universal (ϵ-AU) if for any distinct X_1 and $X_2 \in \mathcal{X}$

$$\Pr[K \xleftarrow{\$} \mathcal{K} : H_K(X_1) = H_K(X_2)] \leq \epsilon.$$

2.1 Useful Constructions

The XOR-Hash Let $\tilde{E} : \mathcal{K} \times (\mathcal{D} \times \mathcal{I} \times \{0,1\}^n) \times \{0,1\}^n \to \times\{0,1\}^n$ be a TBC, where $\mathcal{D} = \{1,2,3,4,5,6\}$ and $\mathcal{I}$ be the set of non-negative integers $\leq l_{\max}$ for a constant $l_{\max} \in \mathbb{N}$. Then, the TBC-based version of XOR-Hash $\mathsf{XH} : \mathcal{K} \times (\{0,1\}^{\leq 2nl_{\max}} \times \{0,1\}^{\leq 2nl_{\max}}) \to \{0,1\}^n$ is given by Algorithm 1. The XOR-Hash is a known construction and was used in other designs [14, 28]. However, it requires careful assignment of message blocks, padding and domain separators. We describe it in this section for the sake of completeness. We shall also discuss it in Section 4. Otherwise, we will refer to it as a black-box construction.

Algorithm 1 The XOR-Hash Function

```
1: $\mathsf{XH}(K, A, M)$
2: **return** $\mathsf{XH}_1(K, A) \oplus \mathsf{XH}_2(K, M)$

3: $\mathsf{XH}_i(K, X)$
4: $X_1, X_2, \ldots, X_x \xleftarrow{n} \mathsf{pad}_{2n}(X)$
5: **if** $X = \epsilon \vee |X| \mod 2n \neq 0$ **then**
6:     $d_l \leftarrow 3 + 3(i-1)$
7: **else**
8:     $d_l \leftarrow 2 + 3(i-1)$
9: **end if**
10: $X_h \leftarrow 0^n$
11: $d \leftarrow 1 + 3(i-1)$
12: **for** $i \in \{0, \ldots, x/2 - 2\}$ **do**
13:     $X_h \leftarrow X \oplus \tilde{E}_K^{d,i,X_{2i+2}}(X_{2i+1})$
14: **end for**
15: $X_h \leftarrow X \oplus \tilde{E}_K^{d_l,x/2-1,X_x}(X_{x-1})$
16: **return** $X_h$
```

Cogliati *et. al.* [14] proved that if the underlying TBC is unpredictable against any adversary running in time $O(l)$ and making at most queries of at most $2l$ queries to the TBC, with advantage at most ϵ, then the described hash function is ϵ-AU against all adversaries making queries of length at most l blocks. They then conjectured that in the case of this hash function,

$$\epsilon \leq \frac{l}{2^k} + \frac{1}{2^n - 2}. \tag{1}$$

We refer the reader to [14] for a discussion on this conjecture, which we will assume is true.

Nonce-as-Tweak (NaT) [15] Let $\tilde{E} : \mathcal{K} \times \mathcal{N} \times \{0,1\}^n \to \{0,1\}^n$ be a TBC and $H : \mathcal{K}_h \times \mathcal{M} \to \{0,1\}^n$ be an ϵ-AU hash function. Then, the NaT MAC is given by

$$\mathsf{NaT}[\tilde{E}, H]_{K,K_h}(M) = \tilde{E}_K^N(H_{K_h}(M)). \tag{2}$$

Besides, let Ver be the oracle that takes as input $M \in \mathcal{M}$ and $T \in \{0,1\}^n$, and returns $\top$ if $\mathsf{NaT}[\tilde{E}, H]_{K,K_h}(M) = T$ and $\perp$, otherwise. Let \$ be the oracle that returns a uniformly random n-bit block for each plaintext-nonce pair $(N, M) \in \mathcal{N} \times \mathcal{M}$ and Rej is the oracle that outputs $\perp$ for all queries. Then, Cogliati *et al.* [15] show that for any adversary $\mathbf{A}$ that makes q_m queries to the first oracle and q_v queries to the second oracle and runs in time t, there exists an adversary $\mathbf{A}^{'}$ than runs in time $O(t + (q_m + q_v)t_H)$ and makes $q_m + q_v$ queries to $\tilde{E}$, such that,

$$|\Pr[K \xleftarrow{\$} \mathcal{K}, K_h \xleftarrow{\$} \mathcal{K}_h : \mathcal{A}^{\mathsf{NaT},\mathsf{Ver}} \Rightarrow 1] - \Pr[\mathcal{A}^{\$,\mathsf{Rej}} \Rightarrow 1]| \leq \mathrm{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathbf{A}^{'}) + 2(\mu - 1)q_m\epsilon + \frac{q_v}{2^n - \mu} + \mu q_v \epsilon. \tag{3}$$

where t_H is the upper bound on the time needed to compute H and μ is the maximum number of times a given $N \in \mathcal{N}$ is repeated in different queries to the first oracle.

PolyVal [22] Let GF(2^{128}) be the binary field of size 2^{128} defined by the irreducible polynomial $x^{128} + x^{127} + x^{126} + x^{121} + 1$. Let M be a message than is divided into a sequence of m 128-bit blocks. Then, $\mathsf{PolyVal}$ is the keyed universal hash function defined by

$$\mathsf{PolyVal}_{K_h}(M) = S_m$$

where $S_0 = 0^n$, and

$$S_i = (S_{i-1} \oplus M_i) \times K_h \times (x^{127} + x^{124} + x^{121} + x^{114} + 1).$$

Gueron *et al.* [22] show that $\mathsf{PolyVal}$ is ϵ-AU, such that

$$\epsilon \leq \frac{l_{\max}}{2^{128}},$$

where $l_{\max}$ is the maximum number of blocks in any input message.

ICE2 Iwata *et al.* [27] proposed a TBC construction that transforms an ideal cipher into a TBC using three calls to the ideal cipher. However, it is optimized particularly for counter-in-tweak style of processing. Given a tweak space $\{0,1\}^n \times \mathcal{I}$, where $\mathcal{I} \subset \mathbb{N}$, and an ideal cipher $E : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$, then $\mathsf{ICE2}$ $(K, (N, i), M)$ is given by

$$2^i V \oplus E(2^i L, 2^i V \oplus M), \tag{4}$$

where $L = E(K, N)$ and $V = E(K \oplus 1, L)$. The multiplication and exponentiation are done in GF(2^n), and we will use $n = 128$ and the same field representation used in $\mathsf{PolyVal}$. $\mathsf{ICE2}$ is very efficient in applications where one part of the tweak is not updated in every TBC call while another part is a sequential counter. Consider a long message than consists of multiple blocks, then the two calls used to generate L and V are only called once, while Equation 4 is evaluated many times, leading to an asymptotic performance of one ideal cipher call per message block. Iwata *et al.* [27] showed that for any unbounded adversary $\mathbf{A}$ that makes q_c queries to $\mathsf{ICE2}$ and q_p chosen-key queries to the underlying ideal cipher,

$$\mathrm{Adv}^{\mathtt{stprp}}_{\mathsf{ICE2}[E]}(\mathbf{A}) \leq \frac{9q_c^2 + 4q_c q_p}{2^{2n}} + \frac{2q_p}{2^n}. \tag{5}$$

3 Low Latency SIV

The LLSIV AEAD scheme is described in details in Algorithm 2. A diagram for the case of 3 full blocks is depicted in Figure 2. In Section 3.1, we study the security of the LLSIV construction. In Section 3.2, we discuss the performance gain of LLSIV compared to SCT-2k and other recently proposed designs.

Algorithm 2 The LLSIV Scheme.

```
1: $\mathsf{Enc}_{K,K_h}(N, A, M)$
2: $IV \leftarrow \mathsf{UHF}(K_h, A, M)$
3: $M_1, \dots, M_m \overset{n}{\leftarrow} M$
4: $T \leftarrow \tilde{E}_K^{0,N}(IV)$
5: for $i \in \{1, \dots, m\}$ do
6:     $C_i \leftarrow M_i \oplus_{|M_i|} \tilde{E}_K^{0,i,N}(IV)$
7: end for
8: $C \leftarrow C_1 \| \dots \| C_m$
9: return $(C, T)$
10: $\mathsf{Dec}_{K,K_h}(N, A, C, T)$
11: $C_1, \dots, C_c \overset{n}{\leftarrow} C$
12: $IV \leftarrow (\tilde{E}_K^{0,0,N})^{-1}(T)$
13: for $i \in \{1, \dots, c\}$ do
14:     $M_i \leftarrow C_i \oplus_{|C_i|} \tilde{E}_K^{i,N}(IV)$
15: end for
16: $M \leftarrow M_1 \| \dots \| M_m$
17: $IV^* \leftarrow \mathsf{UHF}(K_h, A, M)$
18: if $IV = IV^*$ then
19:     return $M$
20: else
21:     return $\perp$
22: end if
```

3.1 Security of LLSIV

The scheme closely resembles the NaT MAC introduced in [15], where instead of generating one nonce-based block, we generate $m + 1$ blocks with the nonce and a counter as a tweak. However, the analysis differs in two main ways:

1. The decryption function uses one call to the TBC in the inverse direction to decrypt the tag. Hence, the forgery analysis relies on the strong TPRP (sTPRP) security of the TBC.
2. The authors of [15] used the H-Coefficient technique for proving that their construction is a secure PRF-MAC when the number of nonce repetitions is small. In the proof, the challenger reveals K_h at the end of the game. The adversary can then link each message with its hash value in both verification and MAC queries. This is not directly possible here, since the adversary

does not know the message during decryption/verification queries. The authors of [15] estimate the number of *inequalities* in the transcript using the hash values, which is not possible here. The *inequalities* are the number of hash/tag pairs observed during verification, where no permutation that is compatible with the attack transcript can have those points in its codebook. The security proof of Romulus-M [27] overcomes this challenge by modifying the authenticity game to to give the adversary oracle access to the encryption and MAC parts, separately, giving the adversary more power to choose the forgeries, then reduce the authenticity to the security of the MAC. Using a sequence of hybrid arguments, we are able to employ this trick here.

Our goal in the security proof of authenticity is to construct an appropriate auxiliary oracle that allows us to:

– prevent the adversary from inferring any information on the decryption of T.
– be able to reduce the security to the security bound of NaT.

Due to these issues, we opted to use game hopping to prove the security. First, we address the $\mathtt{nm}-\mathtt{priv}$ security. We will replace all the TBC calls in encryption queries with random functions, using [27, Lemma (6)]. Then, we will bound the probability that two pairs $(A_1, M_1) \neq (A_2, M_2)$ have the same hash. If the hash value never repeat, then all the function calls have unique inputs. For $\mathtt{nm}-\mathtt{auth}$, we will define a sequence of hybrid games that will be used to reduce the authenticity of LLSIV to the integrity of the NaT MAC [15].

Lemma 1. *(Adapted from [27, Lemma (6)]) Consider a PRF-adversary* $\mathbf{A}$ *against the TURP* $\tilde{\pi} : \mathcal{I} \times \{0,1\}^t \times \{0,1\}^n \rightarrow \{0,1\}^n$. $\mathbf{A}$ *makes* q_i *queries with the first input* $i \in \mathcal{I}$*, such that* $\sum_{i\in\mathcal{I}} q_i = \sigma$. *Any pair* $(i, N) \in \mathcal{I} \times \{0,1\}^t$ *appears in at most* μ *queries. Then, the advantage of* $\mathbf{A}$ *against the PRF security of* $\tilde{\pi}$ *is bound by*

$$\mathrm{Adv}_{\tilde{\pi}}^{\mathtt{prf}}(\mathbf{A}) \leq \frac{(\mu-1)\sigma}{2^n}$$

Proof. Consider $\tilde{\pi}$ is implemented using lazy-sampling. Fix an index $i \in \mathcal{I}$. For a query $j \in \{1, \dots, q_i\}$ with input (i, N_j, P_j), it always returns a random block unless that block has appeared in a previous query $(i, N_j, P^{'})$. For each new call, there are at most $(\mu - 1)$ previous calls on the form $(i, N_j, P^{'})$. Thus, the probability of this collision is at most $(\mu-1)/2^n$. Since the adversary makes at most q_i queries with first input i, the advantage is bounded by $(\mu-1)q_i/2^n$. Using the standard hybrid argument, we apply this sequentially to each input $i \in \mathcal{I}$, getting

$$\mathrm{Adv}(\mathbf{A}) \leq \sum_{i\in\mathcal{I}} \frac{(\mu-1)q_i}{2^n} = \frac{(\mu-1)\sigma}{2^n}.$$

□

Theorem 1. *Let* $\mathbf{A}$ *be an NM privacy adversary against* LLSIV *that can repeat a nonce at most* μ *times in encryption queries.* $\mathbf{A}$ *makes* q_e *queries of total ciphertext size* σ_e *blocks. Let* $\mathbf{A}$ *run in time at most* t*. Then, there exists a* $(q_e + \sigma_e, t + O(q_e t_H + \sigma_e))$*-TPRP adversary* $\mathbf{A}'$ *against the underlying TBC such that*

$$\mathrm{Adv}^{\mathtt{nm-priv}}_{\mathsf{LLSIV}}(\mathbf{A}) \leq \mathrm{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathbf{A}') + (\mu - 1)q_e\epsilon + \frac{(\mu-1)(q_e + \sigma_e)}{2^n}$$

The hash function $\mathsf{UHF} : \mathcal{K}_h \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ *is an* ϵ*-AU hash function and runs in time at most* t_H*.*

Proof. We will define a sequence of hybrid games and bound the transition probability between these games. Let E_i be the event that the adversary wins in game **Gi**.
G0: The oracle implements the real-world construction.
G1: We replace the TBC with a TURP.

$$|\Pr[E_0] - \Pr[E_1]| \leq \mathrm{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathbf{A}').$$

G2: We replace the TURP with a random function $R : \mathbb{N} \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$. In order to bound the transition probability, we need to bound how many permutations are sampled from the TURP and the number of queries made to each permutation. In order to do so, we will define a series of hybrid sub-games, where $\mathbf{G2}^i$ is the where the all the TURP calls with index $i \in \mathbb{N}$ are replaced with a random function $R(i, \cdot, \cdot)$. Let q_i be the number of queries of plaintext length $\geq i$ (partial) blocks. Let $l_{\max}$ be the maximum number of plaintext blocks in one query. Then, applying Lemma 1 [27, Lemma (6)];

$$|\Pr[E_1] - \Pr[E_2^0]| \leq \frac{(\mu-1)q_e}{2^n},$$

$$|\Pr[E_2^{i-1}] - \Pr[E_2^i]| \leq \frac{(\mu-1)q_i}{2^n}$$

and $\mathbf{G2}^{l_{\max}} \equiv \mathbf{G2}$

$$|\Pr[E_1] - \Pr[E_2]| \leq \frac{(\mu-1)q_e}{2^n} + \sum_{i=1}^{l_{\max}} \frac{(\mu-1)q_i}{2^n} =$$

$$\frac{(\mu-1)q_e}{2^n} + \frac{\mu-1}{2^n}\sum_{i=1}^{l_{\max}} q_i = \frac{(\mu-1)(q_e+\sigma_e)}{2^n},$$

where $\sigma_e = \sum_{i=1}^{l_{\max}} q_i$ holds from counting the number of calls made in all queries.
G3: All the calls R are removed and replaced by a random function $R' : \mathbb{N} \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^*$, which takes a nonce, a hash value, and a natural number $l \in \mathbb{N}$ and returns a random string of length $l + n$. **G2** and **G3** are indistinguishable unless the second and third inputs repeat, *i.e.*, if for any query i, there

exists a query $j < i$ such that $N_i = N_j$ and $\mathsf{UHF}_{K_h}(A_i, M_i) = \mathsf{UHF}_{K_h}(A_j, M_j)$. In case of repetition, the oracle of **G2** will return two ciphertexts encrypted with a common prefix, while **G3** will return two independent ciphertexts. Otherwise, all the ciphertexts are indistinguishable. We further define **G3** such that it terminates if such event happens. Since for any query, there are at most $\mu - 1$ queries with the same nonce, the probability of such collision is bounded by

$$|\Pr[E_2] - \Pr[E_3]| = (\mu - 1)q_e\epsilon$$

Besides, **G3** and the ideal world can only be distinguished if $R^{'}$ is called with the same input twice, which is impossible without **G3** terminating. Thus,

$$\Pr[E_3] = 0.$$

Combining all transitions,

$$\Pr[E_0] \leq \Pr[E_3] + \sum_{i=0}^{2} |\Pr[E_i] - \Pr[E_{i+1}]| \leq$$

$$\mathrm{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathbf{A}^{'}) + (\mu - 1)q_e\epsilon + \frac{(\mu - 1)(q_e + \sigma_e)}{2^n}.$$

□

Lemma 2. *Consider a TBC $\tilde{E} : \mathcal{K} \times \mathcal{I} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. Consider the construction Γ:*

$$\tilde{E}(K, i, N, (\tilde{E})^{-1}(K, 0, N, X))$$

where $i \in \mathcal{I} \setminus \{0\}$. Then, for any adversary $\mathbf{G}$ that runs in times t and makes q queries to Γ, there exists an adversary $\mathbf{G}^{'}$ against the strong TPRP security of $\tilde{E}$ that makes q encryption queries, q decryption queries and runs in time $t^{'} = O(t + q)$, such that

$$\mathrm{Adv}^{\mathtt{tprp}}_{\Gamma}(\mathbf{G}) \leq \mathrm{Adv}^{\mathtt{stprp}}_{\tilde{E}}(\mathbf{G}^{'})$$

Proof. First, we replace all the calls of $\tilde{E}$ with a TURP $\tilde{\pi}$. Let $\tilde{P}_0 : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be the tweakable permutation corresponding to $\tilde{\omega}(0, \cdot, \cdot)$. For all $X, Y \in \{0,1\}^n$, and for a given $N \in \{0,1\}^n$, $Y = \tilde{P}_0(N, X)$ is a one-to-one relation between X and Y. Consider a different family of random permutations $\tilde{\sigma}$. We define the game **G1** as the game where the oracle performs the queries $\tilde{\pi}(i, N, \tilde{P}_0(N, X))$ and the game **G2** as the game where the oracle performs the queries $\tilde{\sigma}(i, N, X)$. Let E_1 be the event that the adversary wins in **G1** and E_2 be the event that the adversary wins in **G2**. Since $\tilde{P}_0$ is a one-to-one function and $i = 0$ never appears in any of the forward queries to $\tilde{\pi}$, we can see that

$$|\Pr[E_1] - \Pr[E_2]| = 0,$$

and since $\tilde{\sigma}$ is TURP, then

$$\Pr[E_2] = 0.$$

This concludes the proof. □

Note that the different tweaks play an important role. $\tilde{P}_0$ is indeed part of the same TURP family as the other calls. However, the TURP assumption ensures that since the tweak 0 never appears in any calls other than $\tilde{P}_0$, the outputs of these calls is sampled uniformly and independently of the input-output pairs of $\tilde{P}_0$.

Theorem 2. *Let* $\mathbf{B}$ *be an NM authenticity adversary against* LLSIV *that can repeat a nonce at most* μ *times in encryption queries.* $\mathbf{B}$ *makes* q_e *queries of total ciphertext size* σ_e *blocks and* q_d *decryption/verification queries of total ciphertext size* σ_d*. Let* $\mathbf{B}$ *run in time at most* t_b*. Then, there exists a* $(q_e+q_d+\sigma_e+\sigma_d, t_b+O((q_e+q_d)t_H+\sigma_e+\sigma_d))$*-sTPRP adversary* $\mathbf{B}^{'}$ *against the underlying TBC such that*

$$\mathrm{Adv}^{\mathtt{nm-auth}}_{\mathsf{LLSIV}}(\mathbf{B}) \leq \mathrm{Adv}^{\mathtt{stprp}}_{\tilde{E}}(\mathbf{B}^{'}) + \mathrm{Adv}^{\mathtt{mac}}_{\mathsf{NaT}}(\mathbf{B}^{"})$$
$$\leq \mathrm{Adv}^{\mathtt{stprp}}_{\tilde{E}}(\mathbf{B}^{'}) + 2(\mu-1)q_e\epsilon + \frac{q_d}{2^n-\mu} + \mu q_d\epsilon.$$

The hash function $\mathsf{UHF} : \mathcal{K}_h \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ *is an* ϵ*-AU hash function and runs in time at most* t_H*.*

Proof. We will define a sequence of hybrid games and bound the transition probability between these games. Let E_i be the event that the adversary wins in game **Gi**.
G0: The oracle implements the real-world construction. in Algorithm 2.
G1: We replace the TBC with a TURP $\tilde{\pi}$. Let $(\tilde{\pi})^{-1} \equiv \tilde{\omega}$.

$$|\Pr[E_0] - \Pr[E_1]| \leq \mathrm{Adv}^{\mathtt{stprp}}_{\tilde{E}}(\mathbf{B}^{`}).$$

G2: We modify the calls during the encryption/decryption phase to be a function of T without first calculating IV, as indicated in lines 6 and 14 of Algorithm 3. This change does not affect the security of the scheme. Thus,

$$|\Pr[E_1] - \Pr[E_2]| = 0.$$

G3: We change the oracles by applying the transformation in Lemma 2 to lines 6 and 14 of Algorithm 3. Since $\tilde{\pi}$ is a TURP, then Lemma 2 implies

$$|\Pr[E_2] - \Pr[E_3]| = 0.$$

G4: As shown in Algorithm 4, we change the equality check during verification to check the equality of the tag T instead of IV. Note that the tweakable permutation $\tilde{\pi}$ ensures that

$$(N,T) = (N^*,T^*) \Leftrightarrow (N,IV) = (N^*,IV^*).$$

Thus, changing which variable to check has no implication on the security, *i.e.*,

$$|\Pr[E_3] - \Pr[E_4]| = 0.$$

Algorithm 3 The oracles of game **G2** in the proof of Theorem 2.

```
1: $\mathsf{Enc}_{K,K_h}(N, A, M)$
2: $IV \leftarrow \mathsf{UHF}(K_h, A, M)$
3: $M_1, \ldots, M_m \overset{n}{\leftarrow} M$
4: $T \leftarrow \tilde{\pi}^{0,N}(IV)$
5: for $i \in \{1, \ldots, m\}$ do
6:     $C_i \leftarrow M_i \oplus_{|M_i|} \tilde{\pi}^{i,N}(\tilde{\omega}^{0,N}(T))$
7: end for
8: $C \leftarrow C_1 \| \ldots \| C_m$
9: return $(C, T)$
10: $\mathsf{Dec}_{K,K_h}(N, A, C, T)$
11: $C_1, \ldots, C_c \overset{n}{\leftarrow} C$
12: $IV \leftarrow \tilde{\omega}^{0,N}(T)$
13: for $i \in \{1, \ldots, c\}$ do
14:     $M_i \leftarrow C_i \oplus_{|C_i|} \tilde{\pi}^{i,N}(\tilde{\omega}^{0,N}(T))$
15: end for
16: $M \leftarrow M_1 \| \ldots \| M_m$
17: $IV^* \leftarrow \mathsf{UHF}(K_h, A, M)$
18: if $IV = IV^*$ then
19:     return $M$
20: else
21:     return $\perp$
22: end if
```

In the rest of the proof, we need to show that for any adversary **B** against **G4**, there exists an adversary $\mathbf{B}''$ against the NaT MAC with the same number of MAC and verification queries such that

$$\Pr[E_4] \leq \mathrm{Adv}^{\mathtt{nmauth}}_{\mathbf{G4}}(\mathbf{B}) \leq \mathrm{Adv}^{\mathtt{mac}}_{\mathsf{NaT}}(\mathbf{B}'').$$

In order to do so, we follow the strategy proposed in the security proof of Romulus-M [27].

Algorithm 4 The oracles of game **G4** in the proof of Theorem 2.

```
1: $\mathsf{Enc}_{K,K_h}(N, A, M)$
2: $IV \leftarrow \mathsf{UHF}(K_h, A, M)$
3: $M_1, \ldots, M_m \overset{n}{\leftarrow} M$
4: $T \leftarrow \tilde{\pi}^{0,N}(IV)$
5: for $i \in \{1, \ldots, m\}$ do
6:     $C_i \leftarrow M_i \oplus_{|M_i|} \tilde{\sigma}^{i,N}(T)$
7: end for
8: $C \leftarrow T \| C_1 \| \ldots \| C_m$
9: return $C$
10: $\mathsf{Dec}_{K,K_h}(N, A, C)$
11: $T, C_1, \ldots, C_c \overset{n}{\leftarrow} C$
12: for $i \in \{1, \ldots, c\}$ do
13:     $M_i \leftarrow C_i \oplus_{|C_i|} \tilde{\sigma}^{i,N}(T)$
14: end for
15: $M \leftarrow M_1 \| \ldots \| M_m$
16: $IV^* \leftarrow \mathsf{UHF}(K_h, A, M)$
17: $T^* \leftarrow \tilde{\pi}^{0,N}(IV^*)$
18: if $T = T^*$ then
19:     return $M$
20: else
21:     return $\perp$
22: end if
```

G5: We give the adversary oracle access to $\tilde{\sigma}$, and the adversary makes verification queries on the form (N, A, M, T), rather than (N, A, C). The encryption and decryption oracles in this case are depicted in Algorithm 5, where the adversary uses the $\tilde{\sigma}$ oracle to perform the omitted parts. We say **B** breaks the authenticity of **G5** if the second oracle returns $\top$. This change can only increase the adversary's advantage. Note that the permutations of $\tilde{\sigma}$ and $\tilde{\pi}$ are sampled independently, which was ensured in **G3**. We can see that the oracles in Algorithm 5 are equivalent to the NaT construction. Thus,

$$\Pr[E_4] \leq \Pr[E_5] \leq 2(\mu - 1)q_e\epsilon + \frac{q_d}{2^n - \mu} + \mu q_d \epsilon$$

which follows from Equation 3 [15, Theorem 1].

Algorithm 5 The oracles of game **G5** in the proof of Theorem 2.

```
1: $\mathsf{Enc}_{K,K_h}(N, A, M)$
2: $IV \leftarrow \mathsf{UHF}(K_h, A, M)$
3: $T \leftarrow \tilde{\pi}^{0,N}(IV)$
4: return $T$
```

```
5: $\mathsf{Dec}_{K,K_h}(N, A, M, T)$
6: $IV^* \leftarrow \mathsf{UHF}(K_h, A, M)$
7: $T^* \leftarrow \tilde{\pi}^{0,N}(IV^*)$
8: if $T = T^*$ then
9:     return $\top$
10: else
11:     return $\perp$
12: end if
```

The overall bound is reached by combing the different transition probabilities as follows:

$$\mathrm{Adv}^{\mathtt{nm-auth}}_{\mathsf{LLSIV}}(\mathbf{B}) \leq$$

$$\Pr[E_4] + \sum_{g=0}^{3} |\Pr[E_g] - \Pr[E_{g+1}]| \leq \Pr[E_5] + \sum_{g=0}^{3} |\Pr[E_g] - \Pr[E_{g+1}]|.$$

$\square$

3.2 Performance Comparison

In most SIV-based constructions, the encryption algorithm starts by processing A and M using a UHF. Thus, we can say, for now, that this part of the computation is more less the same. Obviously, different designs of the UHF have different cost and security bounds, but this can be adjusted depending on the application. We shall restrict our discussion in this section to the XOR-Hash XH described in Algorithm 1 as a reference for comparison. In this section, compare the cost of tag generation and encryption. Earlier MRAE-secure design [30, 14, 27, 28, 2, 8, 13] based on TBCs behave in the following fashion:

1. The hash value is processed by one or more TBC calls to generate the tag.
2. The tag is used to encrypt the plaintext using one call to the TBC per plaintext block.

The LLSIV construction, on the other hand, generates both the tag and ciphertext in parallel. As explained in Section 1, earlier SIV-like designs are not pruning-friendly, and require a fully secure TBC for the most part. In the case of LLSIV, this is not necessarily true. Thus, we shall refer to the instantiation using a fully secure TBC as LLSIV, and to the instantiation using a pruned TBC as pLLSIV, with a pruning ratio p, where for a TBC with r rounds, its pruned variant has pr rounds. We will consider the performance in different computational models. For software, we count the cost in terms of TBC calls. For hardware, we count the cost in terms of number of cycles, assuming each round takes 1 cycle. The plaintext consists of m blocks. Table 1 present the comparison between the three schemes in three different computational models. The single-core and multi-core models apply to both software and hardware, where in software a cycle is a call to the round function of the TBC, while in the case of hardware we assume that the implementation computes one round per clock cycle. We need that due to the need of the decryption function, cost of LLSIV is higher that SCT-2k, which translates to area in hardware and code size in software. However, while the performance of LLSIV is the same as SCT-2k on single-core platforms, pLLSIV is faster due to the reduced number of rounds. On parallel platforms, both LLSIV and pLLSIV are faster. They specially excel on pipelined hardware implementations, where only the TBC encryption needs to be pipelined while the decryption can use a small iterative circuit. Thus, the area overhead of LLSIV is very small (one round) compared to SCT-2k, while pLLSIV can have even smaller area than SCT-2k, due to the shallower pipeline. In terms of speed, a pipelined implementation of LLSIV can encrypt $r-1$ more blocks in the same amount of time compared to SCT-2k, while a pipelined implementation of pLLSIV can encrypt $(2-p)r-1$ extra block compared to SCT-2k in teh same amount of time. For short and medium messages, this is a significant speed up and improves the latency of the proposed schemes in time-critical application drastically.

Scheme	**Model**	**TBC Encryption**	**TBC Decryption**	**# cycles**
SCT-2k	Single-Core	Iterative	-	$r(m+1)$
LLSIV		Iterative	Iterative	$r(m+1)$
pLLSIV		Iterative	Iterative	$pr(m+1)$
SCT-2k	Multi-Core	Multi-core	-	$r(\lceil m/c \rceil + 1)$
LLSIV		Multi-core	Iterative	$r\lceil (m+1)/c \rceil$
pLLSIV		Multi-core	Iterative	$pr\lceil (m+1)/c \rceil$
SCT-2k	Pipelined	Pipelined	-	$2r+m-1$
LLSIV		Pipelined	Iterative	$r+m$
pLLSIV		Pipelined	Iterative	$pr+m$

Table 1. Performance comparison between SCT-2k, LLSIV and pLLSIV

4 Instantiations and Implementations

In this section we describe instantiations of the LLSIV and pLLSIV modes. We provide two sets of instantiations. The first set is based on the SKINNY-128-384 [10] TBC. We also describe the hardware architecture and implementation of a fully pipelined accelarator of FPGAs and compare the cost and performance of the proposed designs to the generalized version of SCT-2k described in [14] under the name GNSIV-N.

The second instantiation is based on PolyVal and AES, and is meant to show the advantage of our approach relative to AES-GCM-SIV, which is one of the most popular MRAE-secure schemes.

4.1 Skinny-Based Instantiations

The UHF The topic of designing an ϵ-AU hash function is a rich topic in symmetric key cryptography. UHF can be designed in a variety of ways that are outside the scope of our work. Since our goal is to design a TBC based scheme to be compared with SCT-2k in terms of performance, we shall rely on the same UHF used in SCT-2k: the XOR-hash defined in Algorithm 1. We will use the same hash function for all our instantiations. We will use $n = 128$ bits and $|K_h| = 192$ bits for LLSIV and SCT-2k, and the maximum length or AD or plaintext is 2^{64} blocks. The security of LLSIV in this case is up to $\approx \min\{2^k/l_{\max}\mu, 2^n/\mu\}$ encryption or decryption queries, of size at most $2^n/\mu$ encrypted n-bit plaintext blocks.

While the dependency on the maximum length of associated data and plaintext in the security bound is unpleasant, it seems to be still the best suited solution for our study. UHFs whose security does not depend on the message length have been proposed in other MRAE-secure AEAD schemes such as Romulus-M [27] or ZAE [28]. However, these hash functions are either unparallelizable, or produces $2n$-bit outputs, respectively. We note the design of a fully parallelizable UHF whose security does not depend on the maximum length and outputs only n bits as an interesting open problem.

Tweakable Block Cipher Choosing the TBC for a practical instantiation is not an easy task. Several TBCs with large tweak sizes have been proposed in the past decade, several TBCs with large tweak sizes have been proposed, including Deoxys-TBC [30, 14], SKINNY [10] and QARMA [5, 6]. Any of these TBCs can by used in LLSIV. We choose SKINNY for its hardware-optimized round function and its maturity, with plenty of literature discussing its security.

Pruning When SKINNY-128-384 was first proposed in 2016 [10], it consisted on 56 iterative rounds. Later, Guo *et. al.* [23] conjectured that 56 rounds is an overkill, and proposed reducing the number of rounds to 40 rounds. Henceforth, we shall refer to the 40-round version as the fully secure version and use it as the underlying TBC for any unpruned instantiation. Table 2 includes a list of the most recent cryptanalytic results on SKINNY-128-384. The models that are relevant for both SCT-2k and pLLSIV are the single key and chosen tweak models. We note that the best attacks in these models are those from [12] and [25], respectively. However, this attacks are significantly beyond the security bounds of pLLSIV, where the time complexity is limited to $2^{128}/l\mu$ for $k = 128$. For example, even for only 23 rounds, [12] requires time complexity of more than 2^{362}. The attacks from [25] against SKINNY-128-256 can only reach up to 22 rounds.

Model	**Technique**	**Ref.**	**Number of Rounds**	**Data**	**Time**
Single Key	ID	[45]	22	$2^{92.22}$	$2^{373.48}$
	MitM	[17]	23	2^{120}	2^{368}
	DS-MitM	[41]	23	2^{96}	2^{372}
	Diff-MitM	[12]	25	$2^{122.3}$	$2^{372.5}$
Chosen Tweak	Int	[25]	26	2^{121}	2^{344}
	DS-MitM	[41]	25	2^{96}	$2^{363.83}$
Related Key	Rectangle	[24]	30	2^{125}	2^{361}
		[37]	30	2^{122}	2^{341}
		[18]	32	2^{123}	2^{355}
		[43]	32	2^{123}	2^{345}

Table 2. A summary of the most recent notable cryptanalytic results on SKINNY-128-384. ID: Impossible Differential. MitM: Meet in the Middle. DS-MitM: Demirci-Selçuk MitM. Diff-MitM: Differential MitM.

The situation in pLLSIV loosely resembles the situation of ForkSkinny [4]. The designers have discussed the applicability of different type of attacks to this forked structure. We shall consider the pruned version pLLSIV to use 25 rounds of SKINNY-128-384 for the TBC calls in both the UHF and the rest of the LL-SIV mode. Figure 4 depicts conceptual cryptanalysis targets. Table 2 shows that there are no distinguishers for any of the top or the bottom parts with time complexity 2^{128}. Besides, even if attacks improve, most attacks are not applicable to this setup. For instance, if an adversary wants to attack the bottom permutations alone, they would need either a chosen tweak known ciphertext attack (the plaintext is unknown), which is a very restricted model, or an attack on the inverse permutation cascaded with the forward permutation with a different tweak, which can occur during decryption. However, while this is not exactly the SKINNY TBC, it requires a distinguisher on 50 rounds of SKINNY with dependent, but different, tweakeys. On the other hand, breaking the universality of the UHF requires finding a distinguisher for the top permutations, while observing the effect through the bottom permutations, each including 25 rounds of SKINNY with two different keys. It also requires at least a distinguisher for 25 rounds of SKINNY-128-384 within our security claims in Table 3. Meet-in-the-Middle (MitM) attacks and other attacks that require both chosen plaintext and chosen ciphertext queries, simultaneously, are not applicable to any of the individual TBC calls in pLLSIV. MitM attacks can be applied on P_t and P_b, which generically has complexity of $O(2^{128})$. To beat the generic attack, the adversary needs a distinguisher on 25 rounds of SKINNY-128-384 within our security claims.

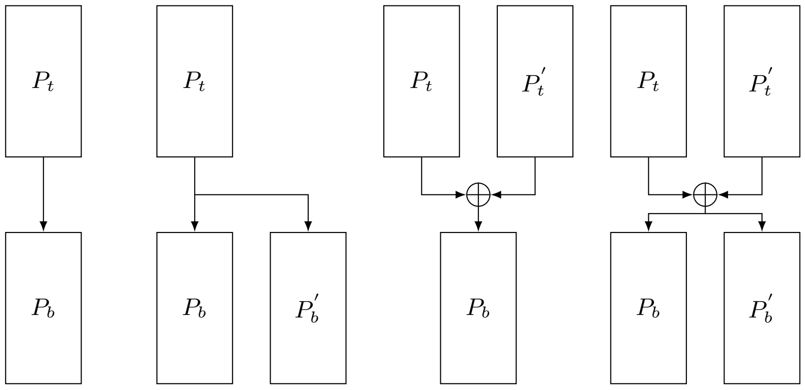

Fig. 4. Conceptual minimalist cryptanalysis targets in pLLSIV.

Last but not least, from the theoretical security bounds of LLSIV and the cryptanalysis of SKINNY-128-384, the security claims are pLLSIV should be limited to the parameters in Table 3. The security claims of pLLSIV are significantly conservative compared to what the attacks allow and are based on the requirements of the recently concluded NIST lightweight cryptography project [1].

Scheme	Max. Length	Data	Time	Key Size
pLLSIV	2^{16}	2^{46}	2^{112}	2×128
LLSIV	2^{64}	$2^{128}/\mu$	2^{128}	$192 + 128$

Table 3. Security claims for pLLSIV and LLSIV. μ is the number of nonce repetitions.

We note that given our parameters, pLLSIV requires 37.5 rounds per plaintext block of 128-bits, which is less than one call of SKINNY (equivalent to 40 rounds). This makes it not only faster than LLSIV and unpruned MRAE-secure TBC-based schemes, but also faster than unpruned online AE schemes such as Deoxys-AE-I or Romulus-N, even using single core implementations.

Domain Separation and Keys In the spirit of being conservative, and given MRAE security requires extra memory overhead anyway, we restrict our self to the case where the UHF uses a different key from other TBC calls. We also use a different domain separator for the TBC calls used to generate the tag and ciphertext blocks compared to the ones used in the UHF. We use 192 bits and 128 bits for the UHF key for LLSIV and pLLSIV, respectively. We use 128 bits for the key of the other calls. On one had, we could use the same key for all TBC calls, assuming the domain separation removes any dependency between the calls. On the other hand, the proofs of Theorems 1 and 2 assume that the UHF and the TBC are completely different functions. Thus, our solution lies somewhere in the middle. It uses the domain separator to assume that the TBC calls inside and outside the UHF are independent, while respecting the provable security result by using different keys. Note that this situation is not uncommon in TBC-based designs, but in fact appears in the security proof of the theoretical generalization of SCT-2k: GNSIV [14].

FPGA Pipelined Implementation of pLLSIV, LLSIV and SCT-2 In order to demonstrate the differences between LLSIV, pLLSIV and SCT-2k on parallel platforms, we have implemented all three algorithms using a fully pipelined SKINNY implementation that computes one round per pipeline stage. We have synthesized the implementations for Xilinx Kintex-7 FPGA using Vivado. The architecture of the hardware accelerator of LLSIV and pLLSIV is depicted in Figure 5. The two algorithms differ in the number of rounds, which affects the number of pipeline stages for the encryption core and the number of cycles for the decryption circuit. The decryption circuit is a round-based implementation of the SKINNY decryption algorithm. The implementation of SCT-2k differs from that of LLSIV in that it does not need the decryption circuit but the tag is always generated using the encryption pipeline, where only one stage is active at a time. While the architecture requires to call the decryption circuit during verification calls to process the tag, this circuit is not the full SKINNY-128-384 decryption circuit, even for LLSIV. We note that this call to the decryption circuit only uses 0 values for both the domain separator and the counter. Thus, 128 bits of the TBC tweak are fixed to 0 and can be ignored during the implementation, which is a property of SKINNY. This gives us a little cost reduction in practice.

Table 4 shows the resource utilization of the FPGA implementations of different schemes. The LLSIV implementation almost the same number of flip flops and 12.3% more Look Up Tables (LUTs), mainly due to the iterative decryption circuit. However, it needs 39 less cycles for encryption. Note that in this implementation, 1 block needs 1.5 cycles. Thus, the LLSIV implementation can encrypt 26 more blocks (416 bytes) in the same amount of time. pLLSIV is even faster, being able to encrypt 69 more blocks (736 bytes) compared to SCT-2k.

In order to demonstrate the performance gain, Figure 6(Left) shows the number of cycles needed to encrypt different numbers of plaintext blocks. Figure 6(Right) shows the ratio between the number of cycles needed by SCT-2k vs the proposed schemes. It can be seen that when the number of blocks is less than 20, pLLSIV is more than twice faster than SCT-2k, while LLSIV is more than 40% faster.

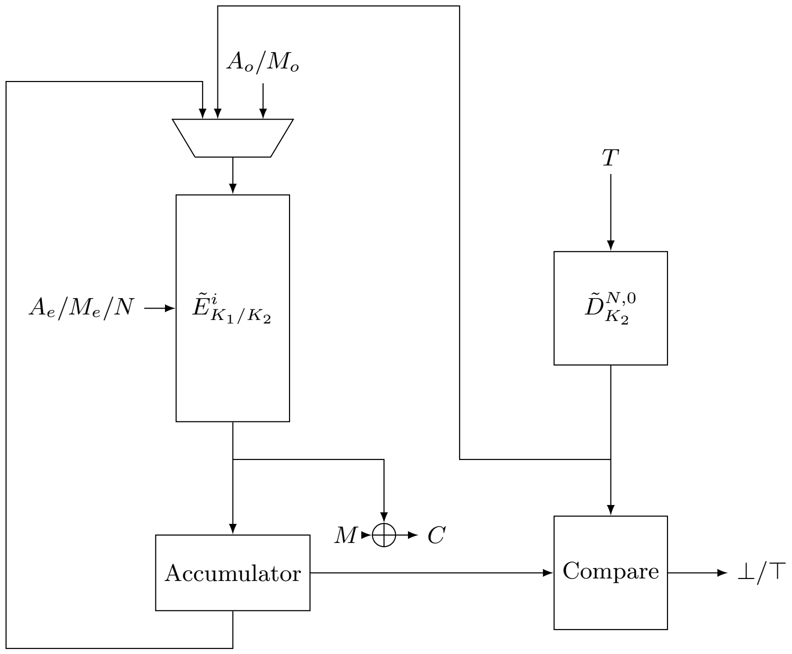

Fig. 5. Simplified architecture of a fully pipelined hardware accelerator for LLSIV and pLLSIV.

Table 4. Synthesis results of the pipelined implementations of SCT-2k, LLSIV and pLLSIV on the Xilinx Kintex-7 FPGA. a and m are the number of 128-bit blocks of associated data and plaintext, respectively. The number of cycles is for the encryption algorithm.

Scheme	**LUTs**	**Flip Flops**	**# of Cycles**
SCT-2k	8230	20581	$118 + a/2 + 3m/2$
LLSIV	9243	20587	$79 + a/2 + 3m/2$
pLLSIV	5392	12907	$49 + a/2 + 3m/2$

4.2 LLSIV-PolyVal-ICE2

In this section, we describe an instantiation of LLSIV based on the PolyVal hash function and the ICE2 TBC with the ideal cipher replaced with AES. The encryption function of LLSIV-PolyVal-ICE2 is depicted in Figure 7. The instance uses two 128-bit uniformly random keys, one for PolyVal and one for ICE2. During encryption, the PolyVal hash function is used to absorb A and M, while in parallel the nonce-based portion of ICE2 (we can refer to it as the KDF) is executed to calculate L and V. Next, the tag and ciphertext are generated in parallel as the calls to AES all encrypt the same hash value and differ only in the exponent i of 2^iL and 2^iV. Thus, similar to the SKINNY based instantiation, the

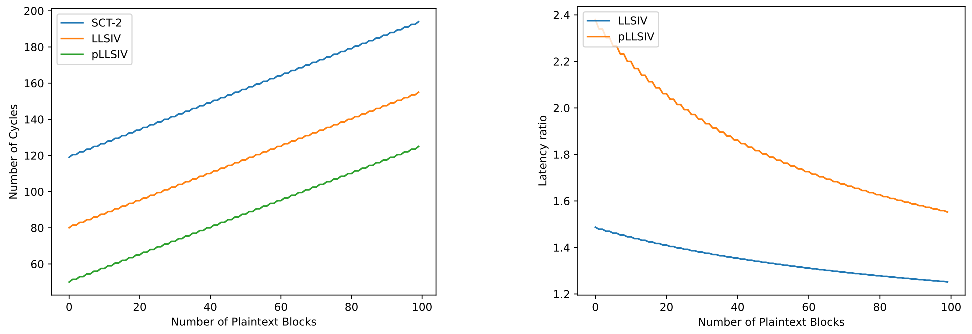

Fig. 6. (Left) The number of cycles needed to encrypt different amounts of plaintext blocks, with at most 2 associated data blocks. (Right) The ratio between the number of cycles needed by SCT-2k and our proposed schemes.

encryption consists of two phases that are fully parallelizable. During decryption, the KDF and tag decryption are executed in parallel in parallel to part of PolyVal that absorbs A. Then, the message is encrypted followed by the remainder of PolyVal. This is again similar to the execution profile of SKINNY based instantiations. This is faster than AES-GCM-SIV on parallel platforms, since the encryption of AES-GCM-SIV require four unparalleizable phases instead of two. First, we have to run the KDF which consists of 4 parallelizable AES calls. Then, PolyVal is evaluated followed by tag generation, followed by encryption. Besides, the KDF of AES-GCM-SIV requires 4 calls, while ours requires only two. Thus, LLSIV-PolyVal-ICE2 is faster than AES-GCM-SIV even on single cores. Note that while AES-GCM-SIV uses a single-key to generate both the hash and encryption subkeys, while LLSIV needs two keys, this is a minor issue, since we can use a separate single-use KDF to extend the master key to two keys. This auxiliary KDF is called only once per key and not for each query, and similar technique is used in AES-GCM. However, we leave the design of such KDF out of scope.

Unlike XH, PolyVal is not parallelizable, so to quantify the gain on pipeline hardware implementation, we need to estimate the performance of PolyVal on FPGA. We implemented an iterative implementation of PolyVal on Kintex-7 FPGA, and our implementation runs at 75 MHz, taking 4 clock cycles per block. Thus, to hash a blocks of A and m blocks or M we need around $4(a+m)$ cycles. In parallel, we need to execute two unparallelizable calls to AES, assuming AES is implemented in a pipelined fashion with one round per stage, and 10 rounds in total. Thus, the first phase of encryption requires $\max(4(a+m),20)$ cycles. The next phase is a simple pipeline implementation of AES that requires $m+9$ cycles. Thus, in total, one query requires $\max(4(a+m),20)+m+9$ cycles.

On the other hand, AES-GCM-SIV encryption consists of 4 phases, as described earlier. The first phase is the KDF which is 4 parallelizable calls to AES, taking 14 cycles. Next, PolyVal takes $4(a+m)$ cycles, followed by once unparallelizable call to AES (11 cycles) and finally the parallelizable encryption $(m+10)$.

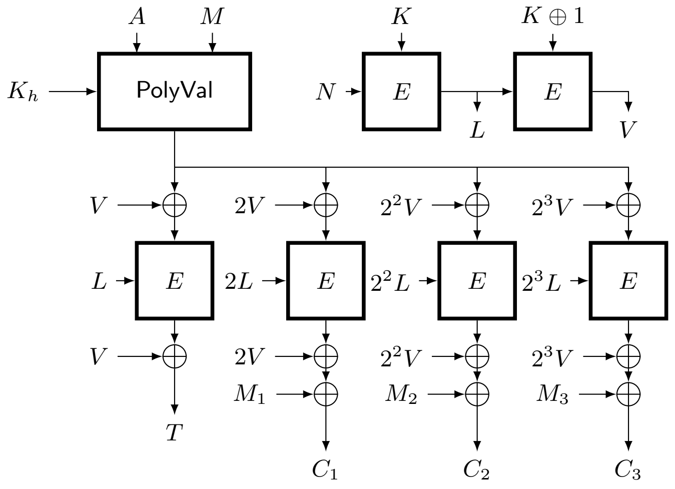

Fig. 7. The encryption function of LLSIV-PolyVal-ICE2 with 3 blocks of plaintext.

In total, it needs $4(a+m)+m+35$ cycles for long messages. If $4(a+m) \geq 20$, then LLSIV-PolyVal-ICE2 takes 25 cycles less than AES-GCM-SIV. 25 cycles are enough to encrypt 5 extra blocks of plaintext ($5 \times 4 + 5$ cycles). While the gain is not as large as the case of SKINNY-based instantiations, it is still significant for latency-critical applications. Besides, the fact that LLSIV-PolyVal-ICE2 is faster than AES-GCM-SIV even on single cores is a big plus: on single cores that process one round of AES per cycle, LLSIV-PolyVal-ICE2 takes 22 cycles less.

The security of LLSIV-PolyVal-ICE2 follows for the straigtforward application of Theorems 3 and 4. It is secure as long as the number of queries is less than $2^{96}/\mu$ when the maximum message length is limited to $2^{38}-1$ bytes. Iwata and Seurin [29] show that AES-GCM-SIV is only secure up to total complexity of 2^{64} in this case. Thus, we consider LLSIV-PolyVal-ICE2 to have similar, but better, numerical security bounds compared to AES-GCM-SIV.

When it comes to the underlying security assumptions of both constructions, we note that LLSIV-PolyVal-ICE2 relies on the single-key security of PolyVal and the ideal cipher model. The latter requires related key security of AES to be sound. AES-GCM-SIV on the other hand relies on the multi-key security of both AES and PolyVal. Thus, AES-GCM-SIV relies on a weaker assumption when it comes to AES, but both schemes cannot rely on the single-key security assumption. However, the ideal cipher model has been established and we believe the trade-off for the improved performance and better bounds is worth it. Due to the ideal cipher assumption, we find it inappropriate to instantiate pLLSIV from LLSIV-PolyVal-ICE2, as reduced-round related key attacks on AES maybe problematic.

FPGA Comparison The synthesis results of AES-GCM-SIV and LLSIV-PolyVal-ICE2 are given in Table 5. AES-GCM-SIV is 25 cycles slower than LLSIV-PolyVal-ICE2 but also about 1200 LUTs smaller. This mainly due to the iterative AES decryption round function used in decryption and the masks used in ICE2.

Table 5. Synthesis results of the pipelined implementations of AES-GCM-SIV and LLSIV-PolyVal-ICE2 on the Xilinx Kintex-7 FPGA. a and m are the number of 128-bit blocks of associated data and plaintext, respectively. The number of cycles is for the encryption algorithm.

Scheme	LUTs	Flip Flops	# of Cycles
AES-GCM-SIV	12780	3017	$4(a+m)+35+m$
LLSIV-PolyVal-ICE2	13965	3401	$4(a+m)+10+m$

5 Low Latency DFV

Minematsu [34] proposed the decryption fast SIV (DFV) framework as a way to optimize the speed of DAE, where the decryption function can be done as a rate-1 function, using an auxiliary tag, and the encryption part of SIV is replaced by a pNAE scheme. He, then, proposed two generic constructions and two dedicated designs. Minematsu [34] discussed several potential methods to optimize his proposed framework to improve their efficiency, and demonstrated that all the considered ideas lead to either insecure constructions or intractable security proofs. However, Minematsu focused mainly on black-box construction where a PRF/MAC is used and its output is used as part of the nonce for the pNAE scheme. Naturally, our proposed technique to construct LLSIV is not part of the ideas considered by Minematsu. In this section, we consider an optimization of the DFV3 scheme by Minematsu that requires one less primitive call. The proposed construction is depicted in Figure 3 and Algorithm 6.

Algorithm 6 The LLDFV Scheme.

```
1: $\mathsf{Enc}_{K_p,K,K_h}(A, M)$
2: $IV_a \leftarrow \mathsf{XH}_1(K_h, A)$
3: $IV_m \leftarrow \mathsf{XH}_2(K_h, M)$
4: $IV_e \leftarrow IV_a \oplus IV_m$
5: $T \leftarrow \tilde{E}^0_K(IV_e)$
6: $(C, V) \leftarrow \mathsf{pNAE.Enc}_{K_p}(IV_a \| IV_e, M)$
7: **return** $(C, T \| V)$

8: $\mathsf{Dec}_{K_p,K,K_h}(A, C, T')$
9: $T, V \xleftarrow{n} T'$
10: $IV_d \leftarrow (\tilde{E}^0_K)^{-1}(T)$
11: $IV_a \leftarrow \mathsf{XH}_1(K_h, A)$
12: $M \leftarrow \mathsf{pNAE.Dec}_{K_p}(IV_a \| IV_d, C, V)$
13: **return** $M$
```

5.1 Security of Low Latency DFV

Instead of using the output of two PRFs as the nonce for a pNAE scheme, we use the output of two universal hash functions. The two hash functions are the XOR-Hash (Algorithm 1), and the same hash function ignoring M. Thus, we know that

$$\mathsf{XH}(A, M) \equiv \mathsf{XH}_1(A) \oplus \mathsf{XH}_2(M)$$

which is a ϵ-AU hash function. However, we also need an assumption on $\mathsf{XH}_1(A)$ on its own and how it interacts with $\mathsf{XH}(A, M)$ when the outputs of both functions are concatenated. An obvious approach is to assume the hash function defined by

$$\mathsf{XH}^{'}(X, i) \equiv \mathsf{XH}_i(X)$$

where the *type* (associated data (1) or plaintext (2)) is part of the input. The *type* input corresponds to the domain separation values in XH. Since the analysis of XH directly applies to $\mathsf{XH}^{'}(X, i)$, we know that $\mathsf{XH}^{'}(X, i)$ is both ϵ-AU and ϵ-AXU. Next, we define the overall UHF:

$$\mathsf{ConcatXH}(A, M) \leftarrow \mathsf{XH}_1(A) \| \mathsf{XH}(A, M).$$

We can show that this function is a (2ϵ)-AU hash function.

Lemma 3. *Given $\mathsf{XH}^{'}(X, i)$ is ϵ-AU and ϵ-AXU, then $\mathsf{ConcatXH}(A, M)$ is (2ϵ)-AU.*

Proof. Given two pairs $(A_1, M_1) \neq (A_2, M_2)$, the proof uses the conditional probability on whether $A_1 = A_2$

$$\Pr[\mathsf{ConcatXH}(A_1, M_1) = \mathsf{ConcatXH}(A_2, M_2) | (A_1, M_1) \neq (A_2, M_2)] \leq$$

$$\Pr[\mathsf{XH}_1(A_1) = \mathsf{XH}_1(A_2) | A_1 \neq A_2] + \Pr[\mathsf{XH}_2(M_1) = \mathsf{XH}_2(M_2) | A_1 = A_2] \leq 2\epsilon.$$

□

Using Lemma 3, we can then show the $\mathtt{d-priv}$ security of LLDFV.

Theorem 3. *Let $\mathbf{A}$ be a (q_e, t)-adversary against the NM privacy of LLDFV as a deterministic AE scheme ($\mu = q_e$). Then, there exists a $(q_e, t^{'})$-adversary $\mathbf{A}^{'}$ and a $(q_e, t^{'})$-adversary $\mathbf{A}^{"}$ for $t^{'} = O(q_e + t)$ and $t^{"} = O(q_e + t)$, such that*

$$\mathrm{Adv}^{\mathtt{d-priv}}_{\mathsf{LLDFV}}(\mathbf{A}) \leq \mathrm{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathbf{A}^{'}) + \mathrm{Adv}^{\mathtt{nr-priv}}_{\mathsf{pNAE}}(\mathbf{A}^{"}) + \frac{0.5 q_e^2}{2^n} + 3\binom{q_e}{2}\epsilon$$

where $\mathsf{XH}_i(X)$ is an ϵ-AU and ϵ-AXU hash function.

Proof. First, $\tilde{E}$ is replaced with a TURP. Then, using the PRP-PRF switching lemma, it is replaced with a uniformly random function R. Next, we observe that as long as the nonce of the underlying pNAE scheme and the input to R are never repeated, then T is uniformly sampled and all the queries to the pNAE scheme are nonce respecting. Finally, the pNAE scheme is replaced with an ideal NAE scheme that outputs random strings of length $|M| + n$. □

Theorem 4. *Let $\mathbf{B}$ be a (q_e, q_d, t)-adversary against the authenticity of LLDFV as a deterministic AE scheme ($\mu = q_e$). Then, there exists a $(q_e + q_d, t^{'})$-adversary $\mathbf{B}^{'}$ and a $(q_e, q_d, t^{'})$-adversary $\mathbf{B}^{"}$ for $t^{'} = O(q_e + q_d + t)$ and $t^{"} = O(q_e + q_d + t)$, such that*

$$\mathrm{Adv}^{\mathtt{d-auth}}_{\mathsf{LLDFV}}(\mathbf{B}) \leq \mathrm{Adv}^{\mathtt{stprp}}_{\tilde{E}}(\mathbf{B}^{'}) + \mathrm{Adv}^{\mathtt{nr-auth}}_{\mathsf{pNAE}}(\mathbf{B}^{"}) + 2\binom{q_e}{2}\epsilon + q_e q_d \epsilon$$

where $\mathsf{XH}_i(X)$ is an ϵ-AU and ϵ-AXU hash function.

Proof. In order to proof this theorem, we construct a series of a hybrid games. Let E_i be the event that the adversary wins in game $\mathbf{Gi}$. $\mathbf{G0}$: The oracles are the real world oracles. $\mathbf{G1}$: First, $\tilde{E}$ is replaced with a TURP.

$$|\Pr[E_0] - \Pr[E_1]| \leq \mathrm{Adv}^{\mathtt{stprp}}_{\tilde{E}}(\mathbf{B}^{'}).$$

$\mathbf{G2}$: The game terminates if during any two encryption queries $(A_i, M_i) \neq (A_j, M_j)$, $\mathsf{ConcatXH}(A_i, M_i) = \mathsf{ConcatXH}(A_j, M_j)$.

$$|\Pr[E_1] - \Pr[E_2]| \leq 2\binom{q_e}{2}\epsilon.$$

$\mathbf{G3}$: The game terminates if there exists a decryption query $(A^\star, C^\star, T^\star)$ and an encryption query (A_i, M_i) such that

$$(A^\star, T^\star) \neq (A_i, \tilde{\pi}(\mathsf{XH}(A_i, M_i)))$$

and

$$\mathsf{ConcatXH}(A_i, M_i) = \mathsf{XH}_1(A^\star)\|\tilde{\omega}(T^\star).$$

Since $\tilde{\pi}$ is bijective, $T^\star \neq T_i$ implies this condition cannot happen. Thus, we only need to look at when $T^\star = T_i$, in which case the condition can only be satisfied if $A^\star \neq A_i$ and $\mathsf{XH}_1(A^\star) = \mathsf{XH}_1(A_i)$. Since there are at more $q_e q_d$ such pairs, then

$$|\Pr[E_2] - \Pr[E_3]| \leq q_e q_d \epsilon.$$

$\mathbf{G4}$: We now consider an adversary $\mathbf{B}^{"}$ that has oracle access to the underlying pNAE scheme, $\tilde{\pi}$, XH_1 and XH_2. $\mathbf{B}^{"}$ simulates the oracles of $\mathbf{G3}$. From $\mathbf{B}$ point of view, games $\mathbf{G3}$ and $\mathbf{G4}$ are indistinguishable.

$$|\Pr[E_3] - \Pr[E_4]| = 0.$$

Besides, if $\mathbf{G4}$ does not terminate, then all the queries made to $\mathsf{pNAE.Enc}$ use unique nonces, and all the queries made to $\mathsf{pNAE.Dec}$ are non-trivial: non-repeating and were not generated by queries to $\mathsf{pNAE.Enc}$.
$\mathbf{G5}$: We replace the pNAE oracles with ideal NAE oracles: all calls to $\mathsf{pNAE.Enc}$ return uniformly random strings and all calls to $\mathsf{pNAE.Dec}$ return $\perp$. Thus,

$$|\Pr[E_4] - \Pr[E_5]| \leq \mathrm{Adv}^{\mathtt{nr-auth}}_{\mathsf{pNAE}}(\mathbf{B}^{"}).$$

Besides, if $\mathbf{G5}$ does not terminate, $\mathbf{B}$ cannot distinguish the oracles from ideal oracles. Thus,

$$\Pr[E_5] = 0.$$

□

5.2 Performance of LLDFV

When instantiating DFV3 using the XOR-Hash function, LLDFV costs one primitive call less. The reduced number of calls makes the scheme faster than DFV3 on any platform (including single-core performance). At the same time, it has the same level of parallelism as LLSIV. In particular, if the underlying pNAE is implemented using the ΘCB3 scheme [33], DFV3 requires two hash functions and $m+3$ TBC calls. On the other hand, LLDFV requires 2 hash functions and $m+2$ TBC calls. On a pipelined architecture of the SKINNY-128-384 TBC, similar to the one used for LLSIV, DFV3 suffers from the same issue as SIV, where there is a call to the TBC during encryption that cannot be parallelized with any other call. Similarly to LLSIV, LLDFV encrypts data in two fully parallelizable phases, where the hash functions can be computed in parallel, while the TBC call used to generate T and the encryption are performed in parallel. Thus, LLDFV requires only 1 extra cycle compared to LLSIV. On the negative side, this implementation of LLDFV requires a pipelined implementation of the decryption circuit. Thus, it requires almost double the area of LLSIV. However, this limitation is inherent from the use of ΘCB3 and is applicable to a similar instantiation of DFV3, as well. In other words, a paralleizable, yet efficient, instantiation of DFV3 already requires both encryption and decryption circuits.

Another alternative is to use an Encrypt-then-MAC scheme as the underlying pNAE scheme. Since we only need the underlying scheme to be secure in the nonce respecting setting, we can use a simple inverse-free scheme, such as counter mode, followed by the Wegman-Carter MAC. Such design would need only encryption calls for the parallelizable functions and would need only an iterative decryption core for the inverse permutation of T. However, this also comes at a similar area cost, since the pipeline depth for encryption function would correspond to two calls instead of one. In other words, we would only need the forward pipelined implementation but its area would almost double.

That being said, in terms of latency, LLDFV offers significantly lower latency in terms of decryption compared to both LLSIV and DFV3, and significantly lower latency than DFV3 in terms of encryption while being only marginally worse than LLSIV.

Last but not least, another limitation of LLDFV is that, unlike LLSIV, is not pruning friendly, since instantiations based on ΘCB3 and Encrypt-then-MAC provide the adversary with access to all the inputs and outputs of the underlying primitive.

5.3 Security Comparison

The security bound of LLDFV is different from DFV3. It is slightly better when the pNAE scheme is only secure upBB, while they are essentially equivalent if the pNAE scheme is secure BBB. Minematsu [34] gave a dedicated design that has the fast decryption property while also being a BBB DAE based on a variant of ΘCB3 and ZMAC. We note that LLDFV is not inherently upBB secure. We can use a different hash function UH with $\epsilon = O(1/2^{2n})$ and use a BBB-secure pNAE.

However, such hash function would lead, mostly likely, to a concatenated hash ConcatUH that has $4n$-bit output, requiring a TBC with a significantly large tweak. For instance, if the pNAE is a variant of ΘCB3, we need a TBC with at least $4.5n$-bit tweak. While this is possible, it may not be very efficient. Since the main goal of this paper is to introduce LLSIV and LLDFV in comparison to SCT-2k and DFV3, respectively, we leave providing an efficient BBB instantiation of LLDFV for a future work.

6 Conclusion

In this paper, we presented three variants of the SIV framework for designing MRAE-secure AEAD. LLSIV is proposed to outperform SCT-2k [14] on parallel platforms. pLLSIV outperfoms LLSIV, SCT-2k, and even online AE schemes based on similar primitives, at the cost of a lower security margin, since it uses a carefully pruned TBC. LLDFV is an optimization of the DFV framework that, not only is more efficient on parallel platforms, but also outperforms DFV3 on any platform as it uses one less TBC call. LLSIV and LLDFV use black-box primitives with compositional security proofs based on hybrid games, while pLLSIV combines the result from LLSIV with the prove-then-prune methodology. We also provide implementation results for a pipelined implementation of LLSIV, pLLSIV and SCT-2k to show the speed up and trade-offs, showing that our new designs on such architecture can encrypt significantly more blocks in the same amount of time, making them excellent for short and medium messages in applications where latency and MRAE security are the primary concerns more than hardware area. We also provided an AES-based instantiation of LLSIV that performs better that AES-GCM-SIV.

Acknowledgement

The author is funded by the Wallenberg-NTU Presidential Postdoctoral Fellowship.

References

1. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process, https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/final-lwc-submission-requirements-august2018.pdf
2. Andreeva, E., Cogliati, B., Lallemand, V., Minier, M., Purnal, A., Roy, A.: Masked iterate-fork-iterate: A new design paradigm for tweakable expanding pseudorandom function. Cryptology ePrint Archive (2022)
3. Andreeva, E., Lallemand, V., Purnal, A., Reyhanitabar, R., Roy, A., Vizár, D.: Forkcipher: a new primitive for authenticated encryption of very short messages. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 153–182. Springer (2019)

4. Andreeva, E., Lallemand, V., Purnal, A., Reyhanitabar, R., Roy, A., Vizár, D.: Forkcipher: a new primitive for authenticated encryption of very short messages. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 153–182. Springer (2019)
5. Avanzi, R.: The qarma block cipher family. almost mds matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Transactions on Symmetric Cryptology pp. 4–44 (2017)
6. Avanzi, R., Banik, S., Dunkelman, O., Eichlseder, M., Ghosh, S., Nageler, M., Regazzoni, F.: The qarmav2 family of tweakable block ciphers. IACR Transactions on Symmetric Cryptology **2023**(3), 25–73 (2023)
7. Banik, S., Bogdanov, A., Luykx, A., Tischhauser, E.: Sundae: small universal deterministic authenticated encryption for the internet of things. IACR Transactions on Symmetric Cryptology pp. 1–35 (2018)
8. Banik, S., Bogdanov, A., Peyrin, T., Sasaki, Y., Sim, S.M., Tischhauser, E., Todo, Y.: Sundae-gift. Submission to Round **1**, 157–161 (2019)
9. Banik, S., Isobe, T., Liu, F., Minematsu, K., Sakamoto, K.: Orthros: a low-latency prf. IACR Transactions on Symmetric Cryptology pp. 37–77 (2021)
10. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The skinny family of block ciphers and its low-latency variant mantis. In: Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36. pp. 123–153. Springer (2016)
11. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 317–330. Springer (2000)
12. Boura, C., David, N., Derbez, P., Leander, G., Naya-Plasencia, M.: Differential meet-in-the-middle cryptanalysis. In: Annual International Cryptology Conference. pp. 240–272. Springer (2023)
13. Chakraborti, A., Datta, N., Jha, A., Mancillas-López, C., Nandi, M., Sasaki, Y.: Estate: A lightweight and low energy authenticated encryption mode. IACR Transactions on Symmetric Cryptology pp. 350–389 (2020)
14. Cogliati, B., Jean, J., Peyrin, T., Seurin, Y.: A long tweak goes a long way: High multi-user security authenticated encryption from tweakable block ciphers. Cryptology ePrint Archive (2022)
15. Cogliati, B., Lee, J., Seurin, Y.: New constructions of macs from (tweakable) block ciphers. IACR Transactions on Symmetric Cryptology pp. 27–58 (2017)
16. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon. Submission to the CAESAR competition: http://ascon. iaik. tugraz. at (2014)
17. Dong, X., Hua, J., Sun, S., Li, Z., Wang, X., Hu, L.: Meet-in-the-middle attacks revisited: key-recovery, collision, and preimage attacks. In: Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part III 41. pp. 278–308. Springer (2021)
18. Dong, X., Qin, L., Sun, S., Wang, X.: Key guessing strategies for linear key-schedule algorithms in rectangle attacks. Springer-Verlag (2022)
19. Dutta, A., Guo, J., List, E.: Forking sums of permutations for optimally secure and highly efficient prfs. Cryptology ePrint Archive (2022)
20. Dworkin, M.J.: Sp 800-38d. recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac (2007)

21. Dworkin, M.J., Barker, E.B., Nechvatal, J.R., Foti, J., Bassham, L.E., Roback, E., Dray Jr, J.F.: Advanced encryption standard (aes) (2001)
22. Gueron, S., Langley, A., Lindell, Y.: Aes-gcm-siv: specification and analysis. Cryptology ePrint Archive (2017)
23. Guo, C., Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Romulus v1.3. Submission to NIST Lightweight Cryptography (2021)
24. Hadipour, H., Bagheri, N., Song, L.: Improved rectangle attacks on skinny and craft. IACR Transactions on Symmetric Cryptology pp. 140–198 (2021)
25. Hadipour, H., Sadeghi, S., Eichlseder, M.: Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 128–157. Springer (2023)
26. Hodjat, A., Verbauwhede, I.: A 21.54 gbits/s fully pipelined aes processor on fpga. In: 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. pp. 308–309. IEEE (2004)
27. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Duel of the titans: the romulus and remus families of lightweight aead algorithms. IACR Transactions on Symmetric Cryptology pp. 43–120 (2020)
28. Iwata, T., Minematsu, K., Peyrin, T., Seurin, Y.: Zmac: a fast tweakable block cipher mode for highly secure message authentication. In: Annual international cryptology conference. pp. 34–65. Springer (2017)
29. Iwata, T., Seurin, Y.: Reconsidering the security bound of aes-gcm-siv. IACR Transactions on Symmetric Cryptology pp. 240–267 (2017)
30. Jean, J., Nikolic, I., Peyrin, T., Seurin, Y.: Deoxys v1. 41 (2016)
31. Jean, J., Nikolić, I., Peyrin, T., Seurin, Y.: The deoxys aead family. Journal of Cryptology **34**(3), 31 (2021)
32. Khairallah, M., Chattopadhyay, A., Peyrin, T.: Looting the luts: Fpga optimization of aes and aes-like ciphers for authenticated encryption. In: Progress in Cryptology–INDOCRYPT 2017: 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings 18. pp. 282–301. Springer (2017)
33. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: Fast Software Encryption: 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers 18. pp. 306–327. Springer (2011)
34. Minematsu, K.: Fast decryption: a new feature of misuse-resistant ae. IACR Transactions on Symmetric Cryptology pp. 87–118 (2020)
35. Naito, Y., Sasaki, Y., Sugawara, T.: Lm-dae: low-memory deterministic authenticated encryption for 128-bit security. IACR Transactions on Symmetric Cryptology pp. 1–38 (2020)
36. Peyrin, T., Seurin, Y.: Counter-in-tweak: authenticated encryption modes for tweakable block ciphers. In: Annual International Cryptology Conference. pp. 33–63. Springer (2016)
37. Qin, L., Dong, X., Wang, X., Jia, K., Liu, Y.: Automated search oriented to key recovery on ciphers with linear key schedule: applications to boomerangs in skinny and forkskinny. IACR Transactions on Symmetric Cryptology pp. 249–291 (2021)
38. Rogaway, P., Bellare, M., Black, J.: Ocb: A block-cipher mode of operation for efficient authenticated encryption. ACM Transactions on Information and System Security (TISSEC) **6**(3), 365–403 (2003)
39. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 373–390. Springer (2006)

40. Saqib, N.A., Rodríguez-Henríquez, F., Diaz-Perez, A.: Aes algorithm implementation-an efficient approach for sequential and pipeline architectures. In: Proceedings of the Fourth Mexican International Conference on Computer Science, 2003. ENC 2003. pp. 126–130. IEEE (2003)
41. Shi, D., Sun, S., Song, L., Hu, L., Yang, Q.: Exploiting non-full key additions: full-fledged automatic demirci-selcuk meet-in-the-middle cryptanalysis of skinny. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 67–97. Springer (2023)
42. Soltani, A., Sharifian, S.: An ultra-high throughput and fully pipelined implementation of aes algorithm on fpga. Microprocessors and Microsystems **39**(7), 480–493 (2015)
43. Song, L., Zhang, N., Yang, Q., Shi, D., Zhao, J., Hu, L., Weng, J.: Optimizing rectangle attacks: a unified and generic framework for key recovery. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 410–440. Springer (2022)
44. Takaki, T., Li, Y., Sakiyama, K., Nashimoto, S., Suzuki, D., Sugawara, T.: An optimized implementation of aes-gcm for fpga acceleration using high-level synthesis. In: 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE). pp. 176–180. IEEE (2020)
45. Tolba, M., Abdelkhalek, A., Youssef, A.M.: Impossible differential cryptanalysis of reduced-round skinny. In: International Conference on Cryptology in Africa. pp. 117–134. Springer (2017)
46. Zhang, Y., Wang, X.: Pipelined implementation of aes encryption based on fpga. In: 2010 IEEE International Conference on Information Theory and Information Security. pp. 170–173 (2010). https://doi.org/10.1109/ICITIS.2010.5688757