IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2024, No. 2, pp. 545–587. DOI:10.46586/tches.v2024.i2.545-587

Gleeok: A Family of Low-Latency PRFs and its Applications to Authenticated Encryption

Ravi Anand[1], Subhadeep Banik[2], Andrea Caforio[3], Tatsuya Ishikawa[1], Takanori Isobe[1], Fukang Liu[4], Kazuhiko Minematsu[5,6], Mostafizar Rahman[1] and Kosei Sakamoto[7]

[1] University of Hyogo, Kobe, Japan.
ravianandsps@gmail.com,takanori.isobe@ai.u-hyogo.ac.jp,mrahman454@gmail.com
[2] University of Lugano, Lugano, Switzerland.
subhadeep.banik@usi.ch
[3] EPFL, Lausanne, Switzerland.
[4] Tokyo Institute of Technology, Tokyo, Japan
liu.f.ad@m.titech.ac.jp
[5] NEC, Kawasaki, Japan.
k-minematsu@nec.com
[6] Yokohama National University, Yokohama, Japan.
[7] Mitsubishi Electric Corporation, Kamakura, Japan.
sakamoto.kosei@dc.mitsubishielectric.co.jp

Abstract. In this paper, we propose a new family of low-latency pseudorandom functions (PRFs), dubbed Gleeok.
Gleeok utilizes three 128-bit branches to achieve a 256-bit key size while maintaining low latency. The first two branches are specifically designed to defend against statistical attacks, especially for differential attacks, while the third branch provides resilience against algebraic attacks. This unique design enables Gleeok to offer ultra-low latency while supporting 256-bit keys, setting it apart from existing ciphers dedicated to low-latency requirements. In addition, we propose wide-block variants having three 256-bit branches. We also present an application of Gleeok to short-input authenticated encryption which is crucial for memory encryption and various real-time communication applications. Furthermore, we present comprehensive hardware implementation results that establish the capabilities of Gleeok and demonstrate its competitiveness against related schemes in the literature. In particular, Gleeok achieves a minimum latency of roughly 360 ps with the NanGate 15 nm cell library and is thus on par with related low-latency schemes that only feature 128-bit keys while maintaining minimal overhead when equipped in an authenticated mode of operation.[1]

Keywords: Ultra-low latency · Pseudorandom function · Authenticated encryption · Short input · Memory encryption · Beyond 5G (B5G)

1 Introduction

Need for Ultra-low Latency Encryption. In lightweight cryptography, the latency of cryptographic primitives has received significant attention due to the increasing need for protecting real-time communication in practical applications. This trend has led to active research on low-latency cryptographic primitives. One prominent application

[1]For the sake of transparency and reproducibility, the full source code of all investigated schemes is available in the form of a public repository at https://github.com/qantik/gleeok.

Received: 2023-10-15 Accepted: 2023-12-15 Published: 2024-03-12

of low-latency primitive is encryption or authentication of computer memory. Memory encryption/authentication is a core component of various security mechanisms/applications such as Trusted Execution Environment (TEE) or Pointer authentication. TEE is a mechanism for processors that ensures the secure execution of programs against unauthorized entities. It often employs a memory encryption scheme as it has to access a main memory (typically a DRAM) put outside the physically protected area. For example, Intel SGX uses a dedicated memory encryption scheme called MEE [Gue16a, Gue16a]. A typical memory encryption scheme (as MEE) consists of MACs/hash functions and Authenticated Encryptions (AEs) composed in a tree for providing replay protection in addition to confidentiality and authenticity [Gue16a, HJ06, RCPS07, IMO+22]. This type of memory encryption has been extensively studied from the system architecture viewpoint, e.g., [YEP+06, SNR+18a, TSB18a, Ava22]. The latency of the cryptographic core inside a memory encryption scheme directly impacts the memory read/write latency, hence it is a critical factor in determining the overall TEE performance. Pointer authentication [Qua17] is a security feature of processors that adds integrity to the pointers while executing a program. ARM deployed Pointer authentication to ARMv8-A processors, called PAC (pointer authentication code), which is essentially a set of special instructions for computing MAC for pointers using QARMA [Ava17].

Another application of low latency encryption schemes will be in the 5G and the Beyond 5G (B5G) mobile communication network. In the latest release by 3GPP for 5G [gpp] it is stated that:

> One of the requirements of 5G/IMT-2020 is ultra-low latency communication with only 1 ms end-to-end latency. To achieve this, it is important that the 256-bit algorithms have as low latency as possible. As the traffic is typically encrypted and decrypted several times, the latency of the 256-bit algorithms will be added several times to the end-to-end latency.

B5G technology is anticipated to enhance the features of 5G, focusing on ultra-low latency and higher data rates. This is particularly important in applications where a delay of even milliseconds can have serious consequences. This includes autonomous vehicles, which require real-time security for vehicle-to-vehicle and vehicle-to-infrastructure communication. Tele-medicine applications, such as remote surgery and diagnostics, depend on low-latency encryption for precise interactions. Immersive VR and AR experiences necessitate encryption with minimal latency to ensure seamless interactions. One may refer to [HP22, JFZ+20, APJ+20]. One can also refer to the white paper for Beyond 5G/6G by NICT [nic] in which they offer a comprehensive perspective on the applications of ultra-low latency encryption in the society envisioned during the B5G era.

Need for 256-bit Key in Latency-Critical Applications. The most of existing low-latency cryptographic primitives have at most a 128-bit key since a longer key generally implies a larger latency. However, one can find a general trend in many real-world applications of (symmetric-key) cryptography that favors longer, typically 256-bit keys. For example, the 3GPP standardization organization has highlighted the significance of increasing the security level to a 256-bit key. One of the reasons for doing so is in light of the potential impact of quantum computing, especially Grover's algorithm [Gro96].

This entails designing a scheme that supports a 256-bit key, which typically provides the desired classical security level, while also ensuring a quantum security level of 128 bits, specifically in relation to Grover's attack. Similar arguments can be found in SNOW-V [EJMY19], ZUC-256 [Tea18] and also by Ericsson [eri]. Our (not fully substantiated) guess is that industries generally prefer a longer-key primitive if performance is satisfactory. For example, many encrypted storage HW/SW products, such as USB drives or cloud storage,

adopt XTS with AES-256. Some concrete examples are Kingston Ironkey series[2], WinMagic SecureDoc[3], and Microsoft Azure Storage encryption[4]. Since the latency requirement is more stringent for memory encryption, most of the current memory encryption schemes adopt a 128-bit key. However, we think the trend toward requiring longer keys will spill over into applications where latency is critical.

Need for an Ultra-low Latency Authenticated Encryption (AE) Scheme. The focus of the cryptographic primitives designed for low-latency, such as PRINCE [BCG+12], QARMA [Ava17], Mantis [BJK+16] and Orthros [BIL+21], is primarily on the latency of the primitive itself, and their potential application in designing low-latency authenticated encryption (AE) schemes has not been deeply explored. However, the development of ultra-low latency AE schemes holds great potential for various applications in the B5G era, as it enables ultra-low latency communications. Exploring the application of low-latency encryption schemes for AE is therefore of significant importance. Authenticated encryption is not only needed for integrity protection but also to provide acceptable confidentiality. An AE scheme with low latency characteristics can facilitate real-time communication and support a wide range of applications in the B5G era.

Besides, as described above, common TEEs including Intel's SGX [Gue16b] employ a memory encryption scheme to protect main computer memory. In a memory encryption scheme for TEE, an AE scheme typically applies to each memory unit, where its latency is critical.

Motivation. Based on the discussion above, we believe it is evident that there is a need for a new scheme that can simultaneously fulfill the requirements of ultra-low latency and support 256-bit keys. To the best of our knowledge, there does not exist an encryption scheme dedicated to providing low latency that satisfies these specific requirements. This fact motivates us to address this need by introducing a new family of low-latency PRF called Gleeok[5]. The primary objective of Gleeok is to offer ultra-low latency, while also supporting 256-bit keys.

Existing Low-latency Primitives. To the best of our knowledge, PRINCE [BCG+12] is the first block cipher that specifically aims for a low-latency design. It follows a similar structure to AES but with a MixColumns-like mapping using a branch number of 4 instead of 5. The key distinction between PRINCE and other ciphers, including AES, is its symmetric design around a linear layer in the middle. PRINCE is a 64-bit block cipher that achieves a significant reduction in the number of rounds while maintaining a moderate level of complexity per round. PRINCEv2 [BEK+20] refines its design to enhance security. Following the design approach of PRINCE, Avanzi proposed QARMA [Ava17], a family of low-latency tweakable block ciphers, which was recently redesigned to QARMAv2 [ABD+23] to improve its security bounds and allow for longer tweaks, while keeping similar latency and area. Mantis [BJK+16] and BipBip [BDD+23] are examples of a low-latency tweakable block cipher and SPEEDY [LMMR21] is a family of ultra low-latency block cipher. Banik et al. proposed Orthros [BIL+21] which is a 128-bit block pseudorandom function (PRF) specifically designed with a primary focus on minimizing latency in fully unrolled circuits. It utilizes a sum of two SPN-type keyed permutations, with round functions based on Midori [BBI+15]. Orthros operates on a 128-bit key and a 128-bit block size. In terms of hardware implementation, Orthros claims to achieve the lowest latency compared to

[2] https://www.kingston.com/en/usb-flash-drives/encrypted
[3] https://winmagic.com/en/products/full-disk-encryption-for-windows/
[4] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption
[5] Gleeok [Glioʊk] is a dragon monster with three heads in *The Legend of Zelda* series.

other state-of-the-art low-latency primitives. Note that these schemes (except QARMAv2) however do not support keys of size 256.

Our Contributions

We propose a family of ultra-low latency PRFs, dubbed Gleeok, providing ultra-low latency while supporting a 256-bit key. Gleeok provides 256- and 128-bit key recovery security against classical and Grover's attacks, respectively. We also propose an AE scheme based on Gleeok which aims at low latency operation for short inputs, which is important for example, memory encryption.

In Section 2, we provide the specification of Gleeok. Section 3 describes its design rationale. Gleeok draws inspiration from Orthros, which is a composition of multiple branches acting as independent block ciphers. One aspect that sets Gleeok apart is that it supports a 256-bit key (Orthros has a 128-bit key) while providing sub-nano class latency. Gleeok is comprised of three branches, with the first two branches specifically designed to defend against statistical attacks, especially for differential attacks, which is the most powerful attack on low-latency ciphers [LMMR21, BDBN23, DEKM16, BJK+16]. On the other hand, the third branch is dedicated to algebraic attacks. The key idea of Gleeok is to synthesize these two contrasting properties of the underlying permutations. The new design of Gleeok is realized through a fine-grained security evaluation utilizing SAT-based tools, which offer powerful capabilities for exploring the design space and identifying optimal parameters for the underlying components. By incorporating bit-level properties and conducting in-depth evaluations, Gleeok achieves significant enhancements in the security against existing attacks, especially for differential attacks, without incurring any delay overhead.

Section 4 studies the security of Gleeok and shows that Gleeok is secure against several classes of classical attacks. Also, since Gleeok supports 256-bit key, we can claim security of 128-bit against Grover's key search attack[6].

Section 5 proposes a low-latency AE scheme based on Gleeok of 128-bit block size. From the application perspective, it is dedicated to short inputs up to 512 bits, which perfectly fits in the current memory encryption for TEEs. Following the idea of SGX's AE scheme [GM16, Gue16b], it uses a simple algebraic universal hash called IP_{64} (for Inner Product hashing over a 64-bit field) in addition to Gleeok. It has a 128-bit tag and achieves 128-bit security against classical attacks in terms of data complexity, assuming the computational security of Gleeok. We call this AE Gleeok-128–IP_{64}.

Section 6 provides a comprehensive suite of hardware implementation results that establish the B5G capabilities of Gleeok-128 and Gleeok-256, Gleeok-128–IP_{64} as well as Gleeok-128–GCM (GCM using Gleeok as its cryptographic core) and demonstrate their competitiveness against related schemes in the literature. Several noteworthy observations can be made from the analysis in this section. Gleeok-128 in its full-round version is on a par with other low latency schemes while the reduced-round variant for size-restricted inputs exhibits a considerable advantage and can be considered the most latency-efficient PRF to date.

We also leverage the payoffs of Gleeok-128 as the core module of Gleeok-128–IP_{64}, paving the way for a low latency Encrypt-then-MAC algorithm that especially in the decryption setting achieves a low latency overhead that results in throughput rates beyond 1 Tbits/s. The versatility of Gleeok-128 is further extended to general-purpose authenticated encryption where we devise an efficient low-latency GCM-based AEAD circuit termed Gleeok-128–GCM that eclipses AES-based equivalents by more than twofold. An abridged synthesis comparison is given in Table 1. We conclude in Section 7.

[6] Bonnetain et al. [BSS22] showed a possibility of beyond quadratic speedup using a quantum computer, hence faster than Grover, however, this requires specific structure for the target and it is not known how this result could be extended/generalized to other designs.

Table 1: Latency (ps) and throughput (Gbits/s) comparison between the Gleeok-based schemes and related schemes from the literature for the NanGate 15 nm cell library. The † symbol denotes round-reduced variants.

PRF			AE (IP_{64})			AE (GCM)		
Scheme	Key	Latency	Scheme	Latency	Max TP	Scheme	Latency	Max TP
Gleeok-128	256	358.52	Encryption			Encryption		
Gleeok-128†	256	298.46	Gleeok-128	647.21	791.09	Gleeok-128	456.91	140.07
Gleeok-256	256	521.43	AES-128	1169.66	437.73	AES-128	953.92	67.09
Gleeok-256†	256	438.44	AES-256	1393.28	367.48	AES-256	1224.13	52.28
Orthros	128	351.55	Decryption			Decryption		
QARMA-128	128	640.00	Gleeok-128	395.22	1295.48	Gleeok-128	449.61	284.69
Midori	128	603.78	AES-128	936.81	546.54	AES-128	950.42	134.68
PRINCE	128	371.62	AES-256	1211.07	422.76	AES-256	1215.91	105.27

2 Specification

Gleeok includes two variants of pseudorandom function (PRF) called Gleeok-128 and Gleeok-256, both of which are based on three SPN-based keyed permutations as illustrated in Figure 1. Both Gleeok-128 and Gleeok-256 have a 256-bit key and a block size being 128 and 256 bits, respectively. In both variants, a 128/256-bit plaintext M is first copied into three internal states X^1, X^2, and X^3, each of which is given as the input to each 128/256-bit keyed permutations Branch1, Branch2, and Branch3, respectively. A 128/256-bit ciphertext C is generated by XORing all outputs of Branch1, Branch2, and Branch3. We give a more detailed specification of Gleeok-128 and Gleeok-256 in the following sections.

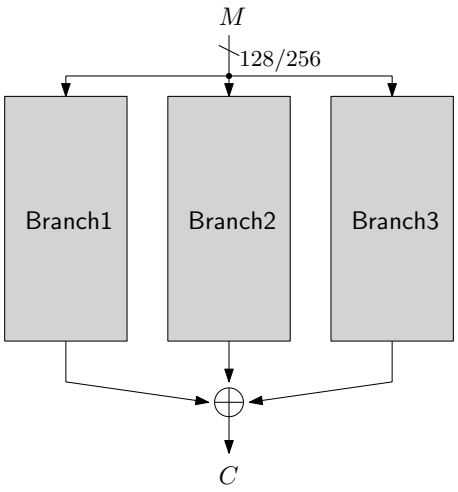


Figure 1: Overview of Gleeok.

2.1 Underlying Keyed Permutations

The underlying keyed permutations are based on a substitution-permutation network with a block size of 128 bits and 256 bits in Gleeok-128 and Gleeok-256, respectively. We give two variants of Gleeok-128 and Gleeok-256 depending on the security claim: Gleeok-128-r and Gleeok-256-r where r denotes the number of rounds for the underlying keyed permutation. For Gleeok-128 and Gleeok-256, we have Gleeok-128-(12,10) and Gleeok-256-(16,12), respectively. The whitening key is applied before the first round, and the linear

layer in the last round is removed. Note that the permutation π_{init} is applied to the internal state before the first round only in Branch2 in Gleeok-256.

In the following explanations of the detailed round function, we denote each operation and the internal states with indexes $i \in \{1, 2, 3\}$ and $\{s, l\}$ to identify the branch number and variants of Gleeok where s and l denote operations in Gleeok-128 and Gleeok-256, respectively. The internal state $X^{\{(s,l),i\}}$ can be expressed in bit level such as $X^{\{s,i\}} = (x_0^i||x_1^i||\dots||x_{127}^i)$ and $X^{\{l,i\}} = (x_0^i||x_1^i||\dots||x_{255}^i)$.

2.1.1 Round Functions

The round function of each branch consists of a parallel application of S-boxes S, a permutation π, a 3 XOR operation θ, a key addition RK_{xor}, and a constant addition RC_{xor} as follows:

$$R = RC_{xor} \circ RK_{xor} \circ \pi \circ \theta \circ S,$$

where R denotes the round function of each branch. Figure 2 shows the overview of the round function.

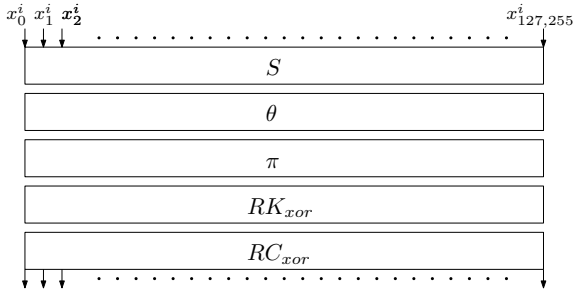


Figure 2: Overview of the round function.

S-box (S). In Branch1 and Branch2, the 3-bit S-box S_3 and the 5-bit S-box S_5 are alternately applied to 128/256-bit internal state, namely, the internal state is updated as follows:

$$\begin{aligned} X^{\{s,i\}} \leftarrow & (S_3(x_0^i||x_1^i||x_2^i)||S_5(x_3^i||x_4^i||x_5^i||x_6^i||x_7^i)||S_3(x_8^i||x_9^i||x_{10}^i)|| \\ & \dots||S_3(x_{120}^i||x_{121}^i||x_{122}^i)||S_5(x_{123}^i||x_{124}^i||x_{125}^i||x_{126}^i||x_{127}^i)), \\ X^{\{l,i\}} \leftarrow & (S_3(x_0^i||x_1^i||x_2^i)||S_5(x_3^i||x_4^i||x_5^i||x_6^i||x_7^i)||S_3(x_8^i||x_9^i||x_{10}^i)|| \\ & \dots||S_3(x_{248}^i||x_{249}^i||x_{250}^i)||S_5(x_{251}^i||x_{252}^i||x_{253}^i||x_{254}^i||x_{255}^i)). \end{aligned}$$

In Branch 3, the 4-bit S-box is applied to the 128/256-bit internal state, namely, the internal state is updated as follows:

$$\begin{aligned} X^{\{s,i\}} \leftarrow & (S_4(x_0^i||x_1^i||x_2^i||x_3^i)||S_4(x_4^i||x_5^i||x_6^i||x_7^i)||\dots||S_4(x_{124}^i||x_{125}^i||x_{126}^i||x_{127}^i)), \\ X^{\{l,i\}} \leftarrow & (S_4(x_0^i||x_1^i||x_2^i||x_3^i)||S_4(x_4^i||x_5^i||x_6^i||x_7^i)||\dots||S_4(x_{252}^i||x_{253}^i||x_{254}^i||x_{256}^i)). \end{aligned}$$

We show S_3, S_4, and S_5 in Tables 2a, 2b, and 2c, respectively.

3 XOR (θ). Each output bit of θ can be obtained by $x_i \leftarrow x_{i+t_0^{\{(s,l),j\}}} + x_{i+t_1^{\{(s,l),j\}}} + x_{i+t_2^{\{(s,l),j\}}}$ where j denotes the branch number. We shows the parameters of t_0, t_1, and t_2 in Table 3.

Table 2: S-boxes used in Branch1, Branch2, Branch3. All S-boxes are given in hexadecimal.

(a) S-box S_3.

x	0	1	2	3	4	5	6	7
$S_3(x)$	0	5	3	2	6	1	4	7

(b) S-box S_4.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S_4(x)$	1	0	2	4	3	8	6	d	9	a	b	e	f	c	7	5

(c) S-box S_5.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S_5(x)$	0	5	a	b	14	11	16	17	9	c	3	2	d	8	f	e
x	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$S_5(x)$	12	15	18	1b	6	1	4	7	1a	1d	10	13	1e	19	1c	1f

Table 3: The parameters of θ.

Gleeok-128	$t_0^{\{s,j\}}$	$t_1^{\{s,j\}}$	$t_2^{\{s,j\}}$	Gleeok-256	$t_0^{\{l,j\}}$	$t_1^{\{l,j\}}$	$t_2^{\{l,j\}}$
Branch1	12	31	86	Branch1	5	93	172
Branch2	4	23	78	Branch2	2	90	179
Branch3	7	15	23	Branch3	0	4	8

Permutation (π). The permutation π is a bit-wise permutation as can be given by $x_i \leftarrow x_{p_j^{(s,l)} \cdot i \bmod (128,256)}$ where j denotes the branch number. We show the parameters of each branch in Table 4.

Table 4: The Parameters of π.

Gleeok-128	p_j^s	Gleeok-256	p_j^l
Branch1	117	Branch1	85
Branch2	117	Branch2	85
Branch3	11	Branch3	39

Key Addition (RK_{xor}). The r-th round internal states in Branch1, Branch2, and Branch3 are xored with the corresponding round key $RK_r^{\{(s,l),i\}}$ where $i \in \{1,2,3\}$.

Constant Addition (RC_{xor}). The r-th round internal states in Branch1, Branch2, and Branch3 are xored with the corresponding round constant $RC_r^{\{(s,l),i\}}$ where $i \in \{1,2,3\}$. The round constants are given as follows:

$$RC_r^{\{s,i\}} = \mathrm{MSB}_{128}((\pi - 3) \lll (r \times 128) + (i \times 12 \times 128)),$$
$$RC_r^{\{s,i\}} = \mathrm{MSB}_{128}((\pi - 3) \lll (r \times 256) + (i \times 16 \times 256) + (3 \times 12 \times 128)),$$

where MSB_{128} extracts the most significant 128 bits. For example, since the fraction part of π is expressed as `0x243f6a8885a308d313198a2e03707344 a4093822299f31d0082efa98`..., $RC_1^{\{s,1\}}$ and $RC_2^{\{s,1\}}$ are `0x243f6a8885a308d313198a2e03707344` and `0xa4093822299f31d0082efa98ec4e6c89`, respectively.

Detailed Procedure of Each Branch. Algorithm 1 shows the detailed procedure of each branch for in Gleeok-128 and Gleeok-256.

Algorithm 1 Procedure of each branch. R depends on the variant of Gleeok-128 and Gleeok-256.

function Branch$i(K, X^{\{s,i\}})$ in Gleeok-128
  $(RK_0^i, RK_1^i, \cdots, RK_R^i) \leftarrow \mathsf{KSF}i_s(K)$
  $X^{\{s,i\}} \leftarrow X^{\{s,i\}} \oplus RK_0^i$
  for $r = 1$ to $R-1$ **do**
    $X^{\{s,i\}} \leftarrow S(X^{\{s,i\}})$
    $X^{\{s,i\}} \leftarrow \theta(X^{\{s,i\}})$
    $X^{\{s,i\}} \leftarrow X^{\{s,i\}} \oplus RK_r^i$
    $X^{\{s,i\}} \leftarrow X^{\{s,i\}} \oplus RC_r^i$
  end for
  $X^{\{s,i\}} \leftarrow S(X^{\{s,i\}})$
  $X^{\{s,i\}} \leftarrow X^{\{s,i\}} \oplus RK_R^i$
  $X^{\{s,i\}} \leftarrow X^{\{s,i\}} \oplus RC_R^i$
return $X^{\{l,i\}}$
end function

function Branch$i(K, X^{\{l,i\}})$ in Gleeok-256
  if $i = 2$ **then**
    $(x_0^*||x_1^*||\ldots||x_{256}^*) \leftarrow X^{\{l,2\}}$
    for $j = 0$ to 255 **do**
      $x_j^2 \leftarrow x_{41\cdot j \bmod 256}^*$
    end for
  end if
  $(RK_0^i, RK_2^i, \cdots, RK_R^i) \leftarrow \mathsf{KSF}i_s(K)$
  $X^{\{l,i\}} \leftarrow X^{\{l,i\}} \oplus RK_0^i$
  for $r = 1$ to $R-1$ **do**
    $X^{\{l,i\}} \leftarrow S(X^{\{l,i\}})$
    $X^{\{l,i\}} \leftarrow \theta(X^{\{l,i\}})$
    $X^{\{l,i\}} \leftarrow X^{\{l,i\}} \oplus RK_r^i$
    $X^{\{l,i\}} \leftarrow X^{\{l,i\}} \oplus RC_r^i$
  end for
  $X^{\{l,i\}} \leftarrow S(X^{\{l,i\}})$
  $X^{\{l,i\}} \leftarrow X^{\{l,i\}} \oplus RK_R^i$
  $X^{\{l,i\}} \leftarrow X^{\{l,i\}} \oplus RC_R^i$
return $X^{\{l,i\}}$
end function

2.2 Key Scheduling

2.2.1 Key Loading

Gleeok-128. The 256-bit key K is divided into two 128-bit keys in different ways for each branch. For Branch1, K is divided into two 128-bit keys as, i.e., $K = K_0||K_1$ where $K_0 = (k_0||k_1||\cdots||k_{127})$ and $K_1 = (k_{128}||k_{129}||\cdots||k_{255})$. Note that k_i denotes a key bit. For Branch2 and Branch3, K_0 and K_1 are set as $\{K_0 = (k_{128}||k_{129}||\cdots||k_{255}),\ K_1 = (k_0||k_1||\cdots||k_{127})\}$ and $\{K_0 = (k_{64}||k_{65}||\cdots||k_{191}),\ K_1 = (k_0||k_1||\cdots||k_{63}||k_{192}||k_{193}||\cdots||k_{255})\}$, respectively. Both K_0 and K_1 are used to generate the round keys for Branch1, Branch2, and Branch3 with the key scheduling function $\mathsf{KSF1}_s$, $\mathsf{KSF2}_s$, and $\mathsf{KSF3}_s$, respectively. The details of the key scheduling function will be described later.

Gleeok-256. The 256-bit key K is loaded into the input of key scheduling function $\mathsf{KSF1}_l$, $\mathsf{KSF2}_l$, and $\mathsf{KSF3}_l$ to generate the round keys for Branch1, Branch2, and Branch3, respectively.

2.2.2 Key Scheduling Function

We adopt the permutation-based key scheduling function for both Gleeok-128 and Gleeok-256, i.e., the round keys are updated by a permutation. In the key scheduling functions of Gleeok-128, the divided 128-bit keys K_1 and K_2 are alternately applied to the permutations and generate the round keys, while the 256-bit key K in the key scheduling function of Gleeok-256 is applied to the permutations in each round and generate the round keys. The permutations used in the key scheduling function can be expressed with the same formula as π, i.e., $k_i \leftarrow k_{pk_j^{(s,l)} \cdot i \bmod (128,256)}$ where j denotes the branch number. Table 5 and Algorithm 2 show the parameters of permutations and the detailed procedure of $\mathsf{KSF}i_{(s,l)}$ where $i \in \{1, 2, 3\}$, respectively.

Table 5: The permutations used in the key scheduling functions.

Gleeok-128	pk_i^s	Gleeok-256	pk_i^l
Branch1	29	Branch1	57
Branch2	51	Branch2	177
Branch3	107	Branch3	239

Algorithm 2 Procedure of $\mathsf{KSF}i_{(s,l)}$. R depends on the variant of Gleeok-128 and Gleeok-256.

function $\mathsf{KSF}i_s(K)$
  $(K_0||K_1) \leftarrow K$
  for $r = 0$ to R **do**
    $(k_0||k_1||\cdots||k_{127}) \leftarrow K_{r \bmod 2}$
    for $j = 0$ to 127 **do**
      $k_j^* \leftarrow k_{pk_i^s \cdot j \bmod 128}$
    end for
    $K_{r \bmod 2} \leftarrow (k_0^*||k_1^*||\cdots||k_{127}^*)$
    $RK_r^i \leftarrow K_{r \bmod 2}$
  end for
return $(RK_0^i, RK_1^i, \cdots, RK_R^i)$
end function

function $\mathsf{KSF}i_l(K)$
  for $r = 0$ to R **do**
    $(k_0||k_1||\cdots||k_{255}) \leftarrow K$
    for $j = 0$ to 255 **do**
      $k_j^* \leftarrow k_{pk_i^s \cdot j \bmod 256}$
    end for
    $K \leftarrow (k_0^*||k_1^*||\cdots||k_{255}^*)$
    $RK_r^i \leftarrow K$
  end for
return $(RK_0^i, RK_1^i, \cdots, RK_R^i)$
end function

2.3 Data Processing

The 128/256-bit ciphertext is generated by an XOR of the output of Branch1, Branch2, and Branch3. Algorithm 3 shows the entire algorithm of Gleeok-128, and Gleeok-256.

Algorithm 3 Processing algorithm of Gleeok, which is same in Gleeok-128 and Gleeok-256.

function GLEEOK(K, M)
  for $i = 1$ to 3 **do**
    $X^{\{(s,l),i\}} \leftarrow M$
    $Y_i^{(s,l)} \leftarrow \mathsf{Branch}i(K, X^{\{(s,l),i\}})$
  end for
  $C \leftarrow Y_1^{(s,l)} \oplus Y_2^{(s,l)} \oplus Y_3^{(s,l)}$
return C
end function

Security Claim.

Gleeok-128-12 and Gleeok-256-16 claims 256-bit security in the single-key setting. We also propose the reduced-round variants, Gleeok-128-10 and Gleeok-256-12, claiming 256-bit security for key recovery, when available data is limited to 2^{64} and 2^{128}, respectively. All variants of Gleeok-128 and Gleeok-256 do not claim any security in the related-key and known/chosen-key settings.

3 Design Rationale

3.1 General Construction

Gleeok is inspired by the concept of Orthros [BIL+21], namely the sum of multiple branches that behave as independent block ciphers. Orthros employs two branches, enabling parallel implementation and reducing latency. Additionally, this multi-branch structure makes it challenging to apply known attack methods against block ciphers. As demonstrated by Orthros's analysis, this approach can achieve a sufficient level of security with a smaller number of rounds compared to designing a stand-alone block cipher.

Three Branches for Supporting 256-bit Key. Gleeok further pushes this approach by employing three branches to increase the key size to 256 bits while keeping lower latency. The three-branch structure increases the difficulty of performing backward computation from the output by requiring the guessing of at least two outputs of the permutations.

This makes it more challenging to mount a key recovery phase in the last rounds, as it would necessitate guessing a 256-bit value.

Moreover, the three-branch structure exhibits significantly improved differential/linear characteristics with fewer rounds compared to the two-branch structure, even when considering clustering effects [TISI23]. Additionally, constructing distinguishers for three different branches simultaneously poses a significant challenge for the adversary, leading us to believe that the pseudo-random functions (PRFs) are secure even in a relatively small number of rounds.

Enhancing Multi-User Security. Moreover, Gleeok-128 could be interpreted as an instantiation of SoP3-2 proposed by Wonseok, Hwigyeom, Jooyoung and Yeongmin at Asiacrypt 2022 [CKLL22]. To our knowledge, our Gleeok-128 is the first such instantiation with dedicated branches. The original SoP3-2 takes an independent key for each branch, instead, we use a single key with three different algorithms. If we view each Branch in Gleeok-128 as a tweakable block cipher (TBC) taking a tweak, where each branch takes one of the three possible values, a hybrid argument shows quite strong multi-user security bound (see Table 1 of [CKLL22]). In particular, it has a multi-user bound stronger than that of 2-branch designs – namely the sum of two permutations (SoP2) [DHT17, CKLL22]. In terms of the number of users (μ) and the number of maximum queries to one user ($q_{\max}$), SoP2 has the bound $\sqrt{\mu q_{\max}/2^n}$, while SoP3-2 has $\sqrt{\mu} q_{\max}^2/2^{2.5n}$ [CKLL22], where n denotes the block size in bits.

More concretely, let $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be an n-bit block keyed function with key $K \in \mathcal{K}$. Let $\mathbf{Adv}_E^{\texttt{mu-prf}}(\mu, q_{\max}, t)$ denote the maximal multi-user pseudorandom function (mu-prf) advantage for adversaries with μ users and time complexity t, where each user is queried at most $q_{\max}$ times. Here, each user takes an independent and random key over $\mathcal{K}$ and the adversary tries to distinguish so-called the real world from the ideal world. Each query consists of user index $i \in \{1, \ldots, \mu\}$ and the input $X \in \{0,1\}^n$, and the response in the real world is $E_{K_i}(X)$ for independent and random key K_i, whereas, in the ideal world, the response is $\mathsf{R}_i(X)$ for n-bit uniform random function[7] R_i which is independent for each i. The output of Gleeok-128 with key $K \in \mathcal{K} = \{0,1\}^{256}$ and input $X \in \{0,1\}^{128}$ is $\mathsf{Gleeok\text{-}128}(K, X) = \mathsf{Branch1}(K, X) \oplus \mathsf{Branch2}(K, X) \oplus \mathsf{Branch3}(K, X)$. We interpret the three branches as a TBC $\mathsf{Branch} : \mathcal{K} \times \{1,2,3\} \times \{0,1\}^n \to \{0,1\}^n$, namely the second argument is the tweak which specifies the branch we use. Let $\mathsf{P} : \{1,2,3\} \times \{0,1\}^n \to \{0,1\}^n$ be the ideal n-bit block TBC of tweak space $\{1,2,3\}$ and let $f[\mathsf{P}]$ be the random function defined as $f[\mathsf{P}](X) = \mathsf{P}(1,X) \oplus \mathsf{P}(2,X) \oplus \mathsf{P}(3,X)$. Note that $f[\mathsf{Branch}]$ is equivalent to Gleeok-128. We have

$$\mathbf{Adv}^{\texttt{mu-prf}}_{\mathsf{Gleeok\text{-}128}}(\mu, q_{\max}, t) \le \mathbf{Adv}^{\texttt{mu-tprp}}_{\mathsf{Branch}}(\mu, 3q_{\max}, t') + \mathbf{Adv}^{\texttt{mu-prf}}_{f[\mathsf{P}]}(\mu, q_{\max}) \tag{1}$$

$$\le \mathbf{Adv}^{\texttt{mu-tprp}}_{\mathsf{Branch}}(\mu, 3q_{\max}, t') + \frac{\sqrt{\mu} q_{\max}^2}{2^{2.5n}} \tag{2}$$

for $t' = t + O(\mu q_{\max})$, where the first term of the right-hand side of Eq. (1) denotes the multi-user distinguishing advantage of Branch from the ideal TBC and the second term denotes the multi-user distinguishing advantage of $f[\mathsf{P}]$ from the n-bit uniform random function. We evaluate the latter without restricting time complexity, so we drop the time complexity from the argument. The first inequality follows from the basic hybrid argument. The second one follows the fact that $f[\mathsf{P}]$ is equivalent to SoP3-2 (as ideal TBC implements an independent random permutation for each tweak) and the mu-prf bound of SoP3-2 from [CKLL22].

The bound of Eq. (2) tells that Gleeok-128 has 128-bit multi-user security with 128-bit

[7] An n-bit uniform random function is a random function that is sampled uniformly random overall functions of n-bit domain and range.

query complexity per user. This is much stronger than Orthros which has 128-bit single-user security with 64-bit multi-user security coming from the trivial collision between keys.

We remark that the above analysis relies on the computational assumption that each branch behaves as an independent and secure block cipher. Our security analysis supports that there is no distinguisher in each branch. However, this does not directly indicate the assumption we used here. Note that an analysis based on a similar assumption has been done for Orthros [BIL+21, Section 3.1]. We have not found anything that invalidates the assumption for Gleeok-128 and leave further analysis open.

Wideblock Variant. Gleeok-256 is a scaled-up version of Gleeok-128 in terms of input/output sizes. The extended input size helps build various modes with 128-bit security in an efficient manner, at the cost of increased latency. For Gleeok-256, the bound of SoP3-2 is vacuous (as its key length is equal to the block size), however, Gleeok-256 simply holds 128-bit multi-user security via a proof based on the key collision.

3.2 Two Types of Keyed Permutations

Gleeok is comprised of three branches, each serving a specific purpose. The first two branches specifically are designed to secure against statistical attacks, especially for differential attacks, while the third branch is dedicated to addressing algebraic attacks.

Permutation with Strong Resistance against Differential Attacks. Low-latency primitives often have a limited number of rounds, which can result in insufficient growth in differential probability. Notably, several primitives such as Mantis [DEKM16, BJK+16] and SPEEDY [LMMR21, BDBN23] have been broken by differential cryptanalysis. Moreover, the best-known attack to the first low-latency primitive PRINCE [BCG+12] is also differential cryptanalysis [CFG+14]. Therefore, it is essential to ensure sufficient security against differential cryptanalysis while minimizing the delay in low-latency primitives.

To address this issue, we leverage two strategies: adopting a 3-branch-based construction and designing an underlying keyed permutation with strong resistance against differential cryptanalysis by bit-level optimization. By employing these strategies, we aim to achieve sufficient security against differential cryptanalysis with a small number of rounds in underlying permutations.

Permutation with Strong Resistance against Algebraic Attacks. On the other hand, XORing the outputs of multiple keyed permutations does not enhance the security against algebraic-degree-based cryptanalysis because the degree of a multiple-branch-based construction is determined by the highest degree among the underlying keyed permutations. Therefore, it is important to ensure resistance against algebraic-degree-based cryptanalysis in addition to differential cryptanalysis when designing low-latency primitives with a minimal number of rounds.

Combined Design. As the algebraic degree of a 3-branch-based construction is determined by the highest algebraic degree among the three underlying keyed permutations, it is reasonable to include one keyed permutation with strong resistance against algebraic-degree-based cryptanalysis and two keyed permutations with strong resistance against differential cryptanalysis in a 3-branch-based construction. This combination allows us to achieve a balanced approach, addressing both algebraic-degree-based and differential cryptanalysis threats effectively within the construction.

3.3 Design of Keyed Permutations

In order to minimize the latency of an entire construction, the underlying keyed permutations also should have a small delay. Considering that Feistel-based constructions only encrypt half of the block in each round, it is more suitable for low-latency design to employ an SPN-based construction such as PRINCE, QARMA, and SPEEDY. Hence, we employ SPN-based keyed permutations as underlying components of Gleeok. In this section, we explain how to design round functions that have strong resistance against differential and algebraic cryptanalysis while maintaining a small delay.

3.3.1 Design in a Nutshell

A common approach to designing an SPN-based primitive is to first choose a suitable S-box and then adjust a linear layer, generally including a matrix and a permutation, with focus on implementation and security.

Limitation of Design by Active S-boxes Estimation. To guarantee security against differential and linear attacks, it is conventionally important to obtain the lower bounds for the number of active S-boxes. Specifically, we first find n_s-bit wise truncated differential/linear characteristics attaining the minimum number of active S-boxes, then the differential/linear probability is bounded by product of the number of active S-boxes and maximum differential/linear probability of S-box, where n_s is the size of S-box.

In this approach, the selection of the linear layer is made independently without directly considering the bit-level properties of the S-box, such as the Difference Distribution Table (DDT) and Linear Approximation Table (LAT). Instead, the focus is on only the maximum differential/linear probability of the S-boxes. This approach simplifies the design process and allows for efficient security evaluation against differential and linear attacks within a practical time. However, it may not fully leverage the specific characteristics of the S-boxes and may not overlook the potential for optimizing the combination of S-boxes and linear layers in terms of both security and performance for specific cryptographic purposes.

Bit-level Optimization for Differential/Linear Attacks. Our design approach is based on the hypothesis that there is an unexplored design space that can be leveraged by considering the bit-level properties of the S-boxes when selecting the linear layers. It should be mentioned that the recent development of SAT-based automatic tools, as proposed by Sun et al. [SWW21], can provide us with the capability to perform bit-level evaluations for differential and linear attacks. In particular, these tools allow us to determine the tightest security bounds by identifying the optimal differential characteristics.

Thus, by utilizing these tools, we can leverage the bit-level characteristics of the S-boxes when selecting linear layers, enabling us to perform further bit-level optimization of the round function design to enhance security against differential and linear attacks while maintaining low-latency performance. In the following sections, we first provide a detailed rationale for the design of the linear and non-linear layers independently, and then we present the best combination of the linear and non-linear layers based on the optimal differential characteristics.

Difference from Existing Approaches. In the design of PRESENT and GIFT [BKL+07, BPP+17], the selection of the linear layer is based on the bit-level properties of the S-boxes, specifically by considering the local behaviors around a set of S-boxes, known as the BOGI strategy. However, these designs employ linear layers that consist only of bit permutations, which is not the best choice from a low-latency perspective. Incorporating XOR operations in linear layers to achieve low latency while considering the heuristic properties becomes complex and challenging. To address this issue, our evaluation focuses

on the differential/linear characteristic probability of the entire function by SAT-aided tool, rather than examining local behaviors. By analyzing these probabilities of the entire function, we are able to examine the optimality of the choices of nolinear and linear layers for the entire round functions, rather than the local optimality. Indeed, a similar approach identified variants of GIFT achieving better resistance against differential attack [SPWW22]. Furthermore, we explore various combinations of S-boxes during the design process to find the most suitable configuration for our desired properties.

3.3.2 Linear Layer

We employ Subterranean-based linear layer [DMMR20], consisting of the θ operation (3-input XOR) and the π operation (a bit permutation). Since the θ operation is performed by almost the same delay as that of the almost MDS matrix used in Midori and Orthros during encryption, Subterranean-based linear layer can be expected to realize almost the same delay as that of Midori and Orthros whose linear layers consist of an almost MDS matrix and a permutation. On the other hand, the advantage of employing Subterranean-based linear layer compared to that of Midori and Orthros, is the increased flexibility in choosing the parameters of the θ operation, along with the π operation, to consider the bit-level properties of the S-boxes and enhance security against specific attack vectors. More specifically, we explore the parameters of the θ and π operations as follows:

$$\theta: \ x_i \leftarrow x_{i+t_0} + x_{i+t_1} + x_{i+t_3},$$
$$\pi: \ x_i \leftarrow x_{p \cdot i \bmod (128,256)},$$

where $i \in \{0, 1, 2, \cdots, 127(\text{or } 255)\}$. Note that the π operation must be a bit permutation, i.e., the parameter i must be chosen among the values whose greatest common divisor (gcd) with 128/256 is 1. Therefore, we have 64/128 and $\binom{128}{3}/\binom{256}{3}$ candidates for the 128-/256-bit variants of the π and θ operations, respectively.

In many designs of symmetric cryptography, the parameters in a linear layer are chosen by a manner of some security properties, such as diffusion property. As with other designs, we also use diffusion property as a criterion of a good linear layer. Since the diffusion property is dependent on not only a linear layer but also an S-box, we will give the specific parameters of a linear layer with a combination of an S-box in Sect. 3.4.

3.3.3 S-Box

We carefully select the S-box based on its delay and security properties. In terms of security, we aim for S-boxes that satisfy both the maximal differential probability and the squared maximum absolute correlation being 2^{-2}. To estimate the path delay of S-boxes, we utilize a metric called *depth* [BBI+15].

Definition 1. Depth: The depth is defined as the sum of the sequential path delays of basic operations, namely AND, OR, NAND, NOR, and NOT.

Following assumption in [BBI+15], we assume the depth of XOR, AND/OR, NAND/NOR, and NOT being 2, 1.5, 1, and 0.5, respectively.

4-bit Low-latency Sbox. Banik et al. already gave a 4-bit S-box with the smallest gate Equivalents (GE) and depth as a non-linear layer of Orthros [BIL+21], whose GE and depth are 20 and 3.5, respectively. Hence, we reuse Orthros's S-box as one of the candidates in the nonlinear layer of Gleeok.

3- and 5-bit Low-latency Sbox. In addition, we focus on the χ function used in Subterranean since it is the well-known-analyzed function in numerous works and its depth is also 3.5, which can be expressed as follows:

$$\chi: \; x_i \leftarrow s_i + (s_{i+1} + 1)s_{i+2}.$$

Considering the balance of security property and depth, we employ the 3- and 5-bit variants of the χ operations, both of which can be viewed as a 3- and 5-bit S-box shown in Table 2a and 2c, respectively. Note that both the 3- and 5-bit variants of the χ operation have the maximal differential probability and the squared maximum absolute correlation of 2^{-2} and 2^{-2}, respectively. A security and hardware comparison of the Gleeok S-box to related variants from the literature is given in Table 6. For the specific construction of the non-linear layer, it will be given in Sect. 3.4.

Remarks. Applying a large S-box, such as a 6-bit S-box [LMMR21, BDD+23], emerges as one potential strategy in designing low-latency ciphers. BipBip/Speedy S-boxes instead of S_3/S_5 for branches 1 and 2 could potentially lead to a reduction of the number of rounds. However, obtaining tight security bounds against differential attacks for 128/256-bit block ciphers having such large S-boxes is still a computationally demanding task. On the other hand, for our S-boxes up to 5 bits, we can derive tight bounds for differential characteristics across a wide class of candidates within a practical time. In addition, delays and area of our 3 to 5-bit S-boxes are significantly smaller than those of 6-bit S-box [BDD+23] as shown in Table 6. Taking advantage of these facts, we focus on smaller S-boxes for our purpose of designing a permutation with strong resistance against differential attacks[8].

3.4 Exploring the Best Combination of the Linear Layer and S-box

Our objective for the underlying keyed permutation is to minimize the number of rounds in order to achieve a small delay. However, achieving strong resistance against differential cryptanalysis while keeping the number of rounds low is a challenging task, as demonstrated by several cryptanalysis results on low-latency primitives [BCG+12, Ava17, BIL+21, LMMR21]. As mentioned in Sect. 3.3.1, we take into consideration the properties of the S-boxes when determining the parameters of the linear layer including XOR operations to achieve this challenging goal.

3.4.1 Approach to Identify Best Combinations

To efficiently identify the best combinations of the linear layer in Sect. 3.3.2 and the S-boxes in Sect. 3.3.3 in terms of delay and security, we employ a two-step evaluation process. First, we evaluate the diffusion property to identify the candidates with the best diffusion property. Then we evaluate the security of the remaining candidates against differential cryptanalysis to determine the best combinations. The detailed procedure for the evaluation of the diffusion property and security against differential cryptanalysis will be given later in this section.

For a non-linear layer, we need to consider not only the specification of an S-box but also the way in which they are applied in parallel. Taking into account the security property of each S-box and our initial evaluations for several combinations of S-boxes, it appears favorable to construct the non-linear layer by (1) only S_4, (2) (3) two types of combination of S_3 and S_5. Thus, we set the non-linear layer in the following three options in the 128-bit variant of keyed permutations:

$$\phi_0 \leftarrow (S_4^0||S_4^1||S_4^2||\cdots||S_4^{30}||S_4^{31}),$$

[8] We used a LUT-based synthesis approach to derive the numbers in Table 6 which is described in more detail in Section 6. Note that additional micro-architectural optimisations may further increase the efficiency of the investigated S-box circuits.

Table 6: Hardware and cryptanalytic characteristics of the S-boxes of Gleeok and related schemes. $\mathcal{A}$ and $\mathcal{L}$ denote area-optimised and delay-optimised circuits respectively, see Section 6. Note that Gleeok S_4 and Orthros S_4 refer to the same S-box.

Scheme	S-Box	Width	Area (GE)		Delay (ps)		DP	C^2	Degree	Full Diffusion
			$\mathcal{A}$	$\mathcal{L}$	$\mathcal{A}$	$\mathcal{L}$				
NanGate 15 nm										
Gleeok	S_3	3	11.25	21.75	12.78	6.61	2^{-2}	2^{-2}	2	-
	S_4	4	16.50	30.50	8.85	5.18	2^{-2}	2^{-2}	3	✓
	S_5	5	18.50	46.00	11.82	6.61	2^{-2}	2^{-2}	2	-
BipBip	S	6	53.75	91.75	20.34	13.70	2^{-4}	2^{-4}	2	-
SPEEDY	S	6	41.49	76.25	17.45	7.43	2^{-3}	$2^{-2.83}$	5	✓
Orthros	S_4	4	16.50	30.50	8.85	5.18	2^{-2}	2^{-2}	3	✓
Midori	Sb_1	4	16.25	30.50	9.85	7.37	2^{-2}	2^{-2}	3	✓
QARMA	σ_0	4	15.25	48.75	11.19	5.68	2^{-2}	2^{-2}	3	-
	σ_1	4	18.00	24.25	15.07	9.66	2^{-2}	2^{-2}	3	✓
	σ_2	4	22.00	53.00	18.69	6.75	2^{-2}	2^{-2}	3	✓
PRINCE	S	4	16.75	30.25	14.00	6.05	2^{-2}	2^{-2}	3	✓
TSMC 28 nm										
Gleeok	S_3	3	10.33	52.33	52.57	24.80	2^{-2}	2^{-2}	2	-
	S_4	4	13.00	55.33	48.13	20.98	2^{-2}	2^{-2}	3	✓
	S_5	5	17.67	90.67	52.57	24.80	2^{-2}	2^{-2}	2	-
BipBip	S	6	50.00	230.33	112.09	49.78	2^{-4}	2^{-4}	2	-
SPEEDY	S	6	34.66	191.33	106.31	29.10	2^{-3}	$2^{-2.83}$	5	✓
Orthros	S_4	4	13.00	55.33	48.13	20.98	2^{-2}	2^{-2}	3	✓
Midori	Sb_1	4	14.33	61.67	62.38	29.32	2^{-2}	2^{-2}	3	✓
QARMA	σ_0	4	13.00	76.33	83.57	21.03	2^{-2}	2^{-2}	3	-
	σ_1	4	15.33	56.33	78.45	35.41	2^{-2}	2^{-2}	3	✓
	σ_2	4	21.00	106.00	81.45	26.48	2^{-2}	2^{-2}	3	✓
PRINCE	S	4	16.00	86.33	90.35	25.05	2^{-2}	2^{-2}	3	✓

$$\phi_1 \leftarrow (S_3^0||S_3^1||S_3^2||\cdots||S_3^{40}||S_5^{41}),$$
$$\phi_2 \leftarrow (S_3^0||S_5^1||S_3^2||\cdots||S_3^{30}||S_5^{31}).$$

In the 256-bit variant, the non-linear layer is applied to the S-boxes in the same manner as in the 128-bit variant.

For ϕ_0, our aim is to enhance security against algebraic-degree-based attacks, taking advantage of the fact that S_4 has the highest degree among all our S-boxes (See Table 6). For ϕ_1 and ϕ_2, we expect to enhance the security against differential cryptanalysis by leveraging a good property of the χ operation. With this in mind, we proceed to explore the best parameters in the linear layers in combination with the above non-linear layer ϕ_0, ϕ_1, and ϕ_2. Hereafter, we elaborate on the detailed procedure of each evaluation to find the best combination of the linear and non-linear layers.

3.4.2 Screening by Diffusion Property

We start the evaluation by screening the candidates based on the number of rounds required to achieve full diffusion, where each input bit can influence all output bits. We examine the diffusion property of each layer, analyzing the number of rounds needed for the full diffusion. For example, we denote 2.5-round full diffusion if each input bit can influence all output bits after 2 rounds and the non-linear layer. Table 7 shows the upper bound for

the number of bits that each input bit can influence, as well as the number of candidates that achieve full diffusion within the corresponding number of rounds, as determined by our evaluation.

Table 7: The number of candidates that achieve full diffusion in corresponding rounds and the upper bound for the number of bits that each input bit can influence after each operation.

Upper bounds for the number of influenced bits								Our results					
Rounds	Operation	128 bits			256 bits			128 bits			256 bits		
		ϕ_0	ϕ_1	ϕ_2	ϕ_0	ϕ_1	ϕ_2	ϕ_0	ϕ_1	ϕ_2	ϕ_0	ϕ_1	ϕ_2
0	input	1	1	1	1	1	1	-	-	-	-	-	-
1	S-box	4	3	3	4	3	3	-	-	-	-	-	-
1.5	θ and π	12	9	9	12	9	9	-	-	-	-	-	-
2	S-box	48	27	27	48	27	27	-	-	-	-	-	-
2.5	θ and π	**128**	81	81	144	81	81	**4352**	-	-	-	-	-
3	S-box	**128**	**128**	**128**	**256**	243	192	15296512	**398**	**352**	**13685504**	-	-
3.5	θ and π	**128**	**128**	**128**	**256**	**256**	**256**	19451328	6032870	7681408	252385280	**39801**	**25856**

As can be seen in Table 7, there are still a large number of candidates remaining after the screening based on the minimum number of rounds required for full diffusion for several S-boxes. To further narrow down candidates, we also look at the minimum number of bits influenced by each input bit before 0.5 or 1 round of the full-diffusion round in several S-boxes. For example, if the 0-th to 126-th input bits can influence 81 bits and the 127-th input bit can influence 74 bits at 0.5 rounds before the full-diffusion round, the minimum number of bits that each input bit can influence is estimated as 74. Among the remaining candidates, we focus on the ones with the most powerful diffusion property, meaning the candidates that have the highest minimum number of bits influenced by each input bit before 0.5 or 1 round of full diffusion, as explained in the following,

128-bit Variant.

ϕ_0: We evaluate the minimum number of influenced bits at 0.5 rounds before the full-diffusion round for the 4352 candidates. Based on this evaluation, we identified 4096 candidates that passed this screening.

ϕ_1: We evaluate the minimum number of influenced bits at 1 round before the full-diffusion round for the 398 candidates. From this evaluation, we identified 16 candidates that passed this screening.

ϕ_2: We perform the same evaluation for ϕ_2 as we did for ϕ_0 and identify 16 candidates that pass the screening. However, these 16 candidates are considered too weak against differential cryptanalysis according to the screening process for differential attacks. To obtain additional candidates, we conducted the same evaluation for 7681408 candidates that achieved full diffusion after 3.5 rounds. From this evaluation, we identified 1408 candidates that passed the screening.

256-bit Variant.

ϕ_0, ϕ_1: We perform the same evaluation for ϕ_0 and ϕ_1 as we did for ϕ_0 in the 128-bit variant. From this evaluation, we identify 438986 candidates for ϕ_0 and 21580 candidates for ϕ_1 that pass the screening.

ϕ_2: We evaluate them using the same approach as ϕ_0 in the 128-bit variant, resulting in 1664 candidates. However, we find that all of these candidates are too weak against differential cryptanalysis according to the screening process for differential attacks.

Therefore, we proceed to evaluate all 25,856 candidates that achieve full diffusion after 3.5 rounds in the next screening step.

As a result, we eventually obtained 4096/16/16(1408) and 438986/21580/1664(25856) candidates for the 128-bit and 256-bit variants, respectively. These candidates will proceed to the evaluation of security against differential cryptanalysis.

3.4.3 Screening by Security against Differential Cryptanalysis

After evaluating the diffusion property of the candidates, we proceed to select the round function in each branch based on their security against differential cryptanalysis. It is worth noting that our evaluation aims to find the optimal differential characteristics rather than determining the lower bound for the number of active S-boxes. As this process is computationally expensive compared to the active-S-box-based evaluation, to efficiently identify promising candidates, we iteratively narrow down candidates round by round in each of the non-linear layers ϕ_0, ϕ_1, and ϕ_2. We summarize our evaluation as follows:

128-bit Variant.

ϕ_0: We evaluate the 3-round optimal differential characteristics for all 4096 candidates. As a result, we identified 512 out of 4096 candidates with the optimal differential characteristic of weight 20, which are the highest ones.

ϕ_1: We evaluate the 4-round optimal differential characteristics for all 16 candidates. As a result, we identify the 4-round optimal differential characteristic of weight 42, which are the highest ones.

ϕ_2: We first evaluate the 4-round optimal differential characteristics for 16 candidates. However, all of them have too weak resistance against differential cryptanalysis, having the optimal differential characteristics with the weight being less than 16 at 4 rounds. Therefore, we then evaluate the 4-round optimal differential characteristics for 1408 candidates described in the explanation of the diffusion property. As a result, we identify 16 out of 1408 candidates with the optimal differential characteristic of weight 50, which are the highest ones.

256-bit Variant.

ϕ_0 Since it is impossible to evaluate security against differential cryptanalysis for 438986 remaining candidates, we randomly choose 1 candidate and evaluate the optimal differential characteristics in each round. We would like to emphasize that ϕ_0 mainly aims at obtaining strong resistance against not differential cryptanalysis but algebraic-degree-based attacks.

ϕ_1 We evaluate the 3-round optimal differential characteristics for all 21580 candidates. As a result, we identify 7184 candidates with the optimal differential characteristic of weight 26 which are the highest ones. Then, we attempt to find the 4-round optimal differential characteristic with a weight of more than 57 for these candidates. However, we cannot find any candidate with a 4-round optimal differential characteristic of weight greater than 57.

ϕ_2 We first evaluate 1,664 candidates that achieve full diffusion after 3.5 rounds and have the maximum number of influenced bits after 3 rounds. However, we only find the optimal differential characteristic with a weight of 20 as the best one. Therefore, we proceed to evaluate the 3-round optimal differential characteristics for 25,856 candidates that achieve full diffusion after 3.5 rounds. As a result, we identified 1,728

candidates with the 3-round optimal differential characteristics of weight 26. Next, we evaluate the 4-round optimal differential characteristics for these 1,728 candidates in search of a candidate with a weight greater than 57.

3.4.4 Identifying Best Combinations

After the above evaluation, we decide the candidates of ϕ_0 and ϕ_2 as the round function of each branch. More specifically, for Gleeok-128, we randomly choose 2 out of 16 candidates in ϕ_2 as the round functions of Branch1 and Branch2, aiming at enhancing security against differential cryptanalysis. As the round function in Branch3, we choose the candidate of ϕ_0 with the 8-round optimal differential characteristic of the weight 64 for enhancing security against algebraic-degree-based attacks. Notably, Branch3 is capable of achieving strong resistance against integral cryptanalysis, which contributes to the overall security of the construction (as discussed in Sect. 4.3). For Gleeok-256, we choose 2 out of 1728 candidates as the round function of Branch1 and Branch2, specifically choosing those with 4-round optimal differential characteristics of weight more than 65 and 64. As the round function in Branch 3, we choose the candidate of ϕ_0 with 8-round optimal differential characteristics of the weight 2^{-88} for the same reason as Branch3 in the 128-bit variant.

As an interesting observation, for the 128-bit variant, ϕ_2 outperforms ϕ_1 in terms of the weight of the optimal differential characteristics at the same number of rounds, despite having fewer S-boxes compared to ϕ_1. This highlights the importance of considering the combination of the linear layer and the non-linear layer in the design process.

Low-Latency Round Function with Strong Resistance against Differential Attack. We show the comparison of the weight of the optimal differential characteristics for the other low-latency primitives with Branch1 and Branch2 in Table 8. The round functions of Gleeok demonstrate significantly stronger resistance against differential cryptanalysis compared to other primitives while minimizing delay. This accomplishment aligns with our primary design objective.

Table 8: Weight of the optimal differential characteristics and critical path delay of the round functions for the NanGate 15 nm cell and TSMC 28 nm cell libraries.

Cipher	Rounds					Delay (ps)	
	1	2	3	4	5	NanGate 15 nm	TSMC 28 nm
Gleeok-128 B1	2	8	**24**	**50**	$\geq$ **67**	18.95	82.01
Gleeok-128 B2	2	8	**24**	**50**	$\geq$ **66**	18.86	81.52
Gleeok-256 B1	2	8	**26**	$\geq$ **66**	-	21.71	83.14
Gleeok-256 B2	2	8	**26**	$\geq$ **65**	-	21.30	88.19
Orthros B1	2	8	14	19	29	20.21	82.13
Orthros B2	2	8	13	19	26	19.97	82.31
Midori-128	2	8	14	32	49	23.31	85.07
QARMA-64	-	-	-	32	40	21.70	89.37
QARMA-128	-	-	-	32	43	21.76	88.38
PRINCE	-	-	-	32	39	20.71	83.88
BipBip	-	-	20	32	36	26.39	108.28

3.4.5 Other Design Choices

Applying π_{init} for Branch 2 in Gleeok-256. From our initial analysis, we discovered that Gleeok-256 could easily have a clustering effect due to the large state size and the same parameter of π operation in Branch1 and Branch2. In order to mitigate this potential problem, we apply π_{init} to Branch2 in Gleeok-256, which can contribute to reducing the similarity between Branch1 and Branch2. Note that applying a bit permutation does not influence the delay.

Decision for the Number of Rounds on the Round-Reduced Variants of Gleeok. We choose the number of rounds for the round-reduced variants of Gleeok, namely Gleeok-128-10 and Gleeok-256-12, based on the security analysis against differential/linear cryptanalysis. According to Table 9 in Sect. 4.1, the required number of rounds with the probabilities less than 2^{-64} and 2^{-128} for Gleeok-128 and Gleeok-256 is 3 and 4 rounds, respectively. These are fewer by 2 and 4 rounds compared to those required for 2^{-128} and 2^{-256} for Gleeok-128 and Gleeok-256, respectively. Hence, we decide on 10 and 12 rounds for the round-reduced variants of Gleeok-128 and Gleeok-256, respectively.

4 Security Evaluation

4.1 Differential and Linear Cryptanalysis

One of the most popular ways to show the resistance against differential [BS91] or linear cryptanalysis [MY93] is to evaluate the lower bound for the number of active S-boxes. In our evaluation, we conduct a more in-depth evaluation, i.e., exploring the optimal differential/linear characteristics that can reveal a more detailed resistance against differential/linear cryptanalysis than an active S-box-based evaluation. To find the optimal characteristics on as long rounds as possible, we employ an SAT-based automatic search method integrated with Matsui's bounding conditions proposed by Sun et al. [SWW21] that is the state-of-the-art method.

Table 9 shows the probability of the optimal differential/linear characteristics for each branch in Gleeok-128 and Gleeok-256. Gleeok-128 and Gleeok-256 have the optimal differential/linear characteristics with the probability of less than $2^{-64}/2^{-128}$ and $2^{128}/2^{256}$ over 3/5 and 4/8 rounds. Therefore, we expect that all variants of Gleeok-128 and Gleeok-256 have a resistance against differential/linear cryptanalysis.

Clustering Effect. As reported by Taka et al. [TISI23], the multi-branch-based construction can be more susceptible to the differential clustering effect compared to single-branch-based constructions. Given the increased degree of freedom in differential transitions with the addition of branches, it is important to examine the clustering effect in Gleeok.

In our evaluation, we specifically investigate the clustering effect on the best-found differential characteristics. For Gleeok-128, we initially evaluate the clustering effect on the 4-round differential characteristic with weight 125. However, no significant clustering effect was observed on this characteristic. It was only when we examined the 4-round differential characteristic with weight 155 that we finally observed a clustering effect, resulting in a reduction in probability from 2^{-155} to $2^{-139.96}$. This weak clustering effect could come from the evaluation of the small number of rounds, making it difficult to achieve the same output difference from the same input difference with different internal transitions.

This phenomenon can be also observed on Gleeok-256. We evaluate the clustering effect on the 4-round differential characteristic with weight 138 but cannot observe the clustering effect at all. It might be possible that a stronger clustering effect may be observed with a larger number of rounds, however an underlying characteristic will have enough weight to render the consideration of the clustering effect negligible.

Table 9: The probability of optimal differential/linear characteristics. DP and C denote the differential probability and the absolute correlation, respectively. The upper bound of DP/C^2 by the product of the DP/C^2 in multiple rounds is written in gray.

Variant	Branch	DP/C^2	Rounds							
			1	2	3	4	5	6	7	8
Gleeok-128	Branch1	DP	2^{-2}	2^{-8}	2^{-24}	2^{-50}	$\leq 2^{-67}$	2^{-69}	2^{-75}	2^{-100}
		C^2	2^{-2}	2^{-8}	2^{-22}	2^{-44}	$\leq 2^{-64}$	2^{-66}	2^{-72}	2^{-88}
	Branch2	DP	2^{-2}	2^{-8}	2^{-24}	2^{-50}	$\leq 2^{-66}$	2^{-68}	2^{-74}	2^{-100}
		C^2	2^{-2}	2^{-8}	2^{-22}	2^{-44}	$\leq 2^{-64}$	2^{-66}	2^{-72}	2^{-88}
	Branch3	DP	2^{-2}	2^{-8}	2^{-20}	2^{-32}	2^{-36}	2^{-48}	2^{-52}	2^{-64}
		C^2	2^{-2}	2^{-8}	2^{-20}	2^{-30}	2^{-36}	2^{-48}	2^{-52}	2^{-64}
	Gleeok-128	DP	2^{-6}	2^{-24}	2^{-70}	$\leq 2^{-126}$	2^{-132}	2^{-148}	2^{-196}	2^{-252}
		C^2	2^{-6}	2^{-24}	2^{-66}	$\leq 2^{-124}$	2^{-130}	2^{-148}	2^{-190}	2^{-248}
Gleeok-256	Branch1	DP	2^{-2}	2^{-8}	2^{-26}	$\leq 2^{-66}$	2^{-68}	2^{-74}	2^{-92}	2^{-132}
		C^2	2^{-2}	2^{-8}	2^{-26}	$\leq 2^{-66}$	2^{-68}	2^{-74}	2^{-92}	2^{-132}
	Branch2	DP	2^{-2}	2^{-8}	2^{-26}	$\leq 2^{-65}$	2^{-67}	2^{-73}	2^{-91}	2^{-130}
		C^2	2^{-2}	2^{-8}	2^{-26}	2^{-64}	$\leq 2^{-78}$	2^{-80}	2^{-90}	2^{-128}
	Branch3	DP	2^{-2}	2^{-8}	2^{-20}	2^{-39}	2^{-61}	$\leq 2^{-72}$	2^{-74}	2^{-81}
		C^2	2^{-2}	2^{-8}	2^{-18}	2^{-38}	2^{-52}	2^{-64}	2^{-76}	2^{-88}
	Gleeok-256	DP	2^{-6}	2^{-24}	2^{-76}	$\leq 2^{-139}$	2^{-145}	2^{-173}	2^{-215}	2^{-278}
		C^2	2^{-6}	2^{-24}	2^{-72}	$\leq 2^{-136}$	2^{-142}	2^{-160}	2^{-208}	2^{-272}

4.2 Impossible Differential Cryptanalysis

Since the SAT model to evaluate the optimal differential characteristics in Sect. 4.1 captures only valid characteristics, it can be also applied to the evaluation for finding the longest impossible differences with some modifications. To find the longest impossible differences, what we need to change is to remove the objective function from the SAT model, namely, clauses to restrict the total weight in a model will be removed. Instead of these clauses, we add the clauses to fix the input and output differences. A similar technique is often used in the evaluation by an MILP [ST17]. In the evaluation by an MILP, it is computationally infeasible to check the existence of impossible differences in all pairs of input and output differences. Therefore, only the pairs having one active bit in both input and output differences will be evaluated to estimate the existence of the impossible differences in each round. We follow this strategy to find the longest impossible differences. As a result, we find that the longest impossible differences of Branch1/Branch2/Branch3 in Gleeok-128 and Gleeok-256 are the 3/3/3- and 4/4/4-round ones, respectively. Additionally, the longest impossible differences in the entire Gleeok-128 and Gleeok-256 are found on 2 and 2 rounds. Since there are 2 and 2 rounds left as a security margin for Gleeok-128 and Gleeok-256, respectively, we believe that all variants of Gleeok-128 and Gleeok-128 have a resistance against impossible differential cryptanalysis.

4.3 Integral Cryptanalysis

We employ division property [Tod15] to find the longest integral distinguishers. One of the popular ways to explore division property is to make use of solver-aided automatic search tools, such as an MILP-based tool proposed by Xiang et al. [XZBL16]. Since Xiang et al.'s method can be naturally converted to an SAT-based method, we search for the longest integral distinguishers by Xiang et al.'s method with an SAT. As the number of rounds for Gleeok-128 and Gleeok-256 depends on a data complexity that the attacker can exploit, we attempt to find the longest integral distinguisher with a data complexity of

$2^{64}/2^{127}$ and $2^{128}/2^{255}$ for Gleeok-128 and Gleeok-256, respectively. In the evaluation of integral distinguishers with a data complexity of 2^{64} and 2^{128}, we assign 64 and 128 active bits to the input and evaluate some patterns of these inputs because it is impossible to explore all input patterns.

Table 10 shows the results of our evaluation. In Table 10, Branch1/Branch2/Branch3 in Gleeok-128 and Gleeok-256 have integral dinstinguishers up to 6/6/5 and 6/6/5 rounds, respectively. As described in Sect. 3.2, the highest degree of the entire Gleeok-128 and Gleeok-256 depends on a branch with the strongest resistance against degree-based attack. Therefore, the maximum number of rounds having the integral distinguishers for Gleeok-128 and Gleeok-256 are bounded by Branch3. In fact, the entire Gleeok-128 and Gleeok-256 have the integral distinguisher up to 5 and 5 rounds, respectively, which is the same number of rounds for that of Branch3. Hence, we expect that all variants of Gleeok-128 and Gleeok-256 can resist integral cryptanalysis since there are still enough security margins.

Table 10: The integral distinguishers in each round for each branch in Gleeok-128 and Gleeok-256.

Variant	Branch	Data complexity															
		$2^{64}/2^{128}$						$2^{127}/2^{255}$									
		Rounds						Rounds									
		4	5	6	7	8	9	5	6	7	8	9	10	11	12	13	14
Gleeok-128	Branch1	✓	✓	?	?	×	×	✓	✓	?	?	?	?	?	?	?	×
	Branch2	✓	✓	?	?	×	×	✓	✓	?	?	?	?	?	?	?	×
	Branch3	✓	×	×	×	×	×	✓	?	?	×	×	×	×	×	×	×
	Gleeok-128	✓	×	×	×	×	×	✓	?	?	×	×	×	×	×	×	×
Gleeok-256	Branch1	✓	✓	?	?	?	×	✓	✓	?	?	?	?	?	×	×	×
	Branch2	✓	✓	?	?	?	×	✓	✓	?	?	?	?	?	×	×	×
	Branch3	✓	✓	×	×	×	×	✓	?	?	?	×	×	×	×	×	×
	Gleeok-256	✓	✓	×	×	×	×	✓	?	?	?	×	×	×	×	×	×

✓:There exist the integral distinguishers

×:There do not exist integral distinguishers

?: Evaluation does not finish in practical time.

4.4 Truncated Differential Attack

Truncated differential cryptanalysis, originally introduced by Knudsen in [Knu94], offers a different perspective on analyzing cryptographic primitives. Instead of focusing on the precise values of inputs and outputs, this approach considers patterns of differences, whether they are zero or non-zero. By studying these difference patterns, the vulnerabilities of the primitives can be analyzed.

Consider the output differences with respect to the input difference `110` in the difference distribution table (DDT) of the 3-bit S-box. It can be observed that the first two bits of the output difference are always complementary to each other. This primarily stems from the dependency between the `AND` operations with a common input in the χ function [BVAPD11]. This particular property can be exploited to construct distinguishers for the underlying permutations. Consider an input difference for Branch1 in Gleeok-128 whose only first two bits are active ($110\cdots0$). In the S-box output, the first two bits should be either (0,1) or (1,0) (the remaining bits should be zero) with probability $\frac{1}{2}$. This truncation property can be further extended for one more round and two disjoint sets, A and B can be formed as shown in Table 11.

After two rounds, it is expected that all the bit differences corresponding to the positions in one of these sets should be zero with probability $\frac{1}{2}$. After three rounds, such sets do

not exist due to the internal round operations. These properties can be extended by prepending a differential that satisfies the input difference of the truncated differential. For Branch1, such differentials exist for 2 rounds with probability 2^{-92}. Hence, it seems that a 4-round distinguisher can be constructed with probability 2^{-93}. Note that, as the size of the two sets is small, the success probability of this distinguisher will be too low (to distinguish it from a random permutation). Thus, we believe this distinguisher will be effective for at most three rounds for Branch1. Branch2 also exhibits a similar kind of resistance to such truncated differential attacks. In Branch3, this property extends for one more round which makes at most 4 rounds vulnerable to such attacks. The underlying permutations of Gleeok-256 also exhibits similar resistance against these types of truncated differential attacks.

Table 11: Sets corresponding to truncated differential

$$A = \{2, 7, 9, 11, 14, 16, 19, 21, 30, 32, 37, 42, 44, 46, 49, 51, 55, 56, 60, 65, 67, 72, 77, 79, 81, 86, 90, 91, 95, 100, 102, 104, 107, 109, 112, 114, 116, 121, 125\}$$

$$B = \{0, 1, 5, 10, 12, 17, 22, 26, 31, 35, 36, 40, 45, 47, 52, 57, 61, 66, 70, 75, 80, 82, 87, 92, 94, 103, 105, 110, 115, 117, 119, 122, 124\}$$

4.5 Boomerang/Rectangle Attacks

The boomerang attack [Wag99] is a variant of the differential attack that was introduced by Wagner. It involves subdividing a cipher, denoted as E, into two parts: E_1 and E_0. This subdivision allows for the penetration of a larger number of rounds, which would otherwise not be vulnerable to differential attacks. Initially, two independent differential trails for E_0 and E_1 are combined to construct the boomerang trail. But later on, several incompatibilities regarding the independence of the two trails are identified [Mur11] and subsequently several frameworks are proposed considering a E_m layer in between the two layers [BK09, DKS10, DKS14]. Introduction of the *boomerang connectivity table* (BCT) marks a significant advancement in this regard that combines E_0 and E_1 with an one round E_m [CHP+18].

Here, also we have employed a SAT-based analysis tool to automatically search for optimal boomerang characteristics with BCT-effect. It is important to note that our focus here is on analyzing the security of the underlying pseudorandom permutations (PRP), *i.e.*, Branch1/Branch2/Branch3 in Gleeok-128 and Gleeok-256, rather than the PRF as a whole. This is due to the fact that the PRF lacks a decryption oracle, making it impossible to launch a boomerang attack directly on the PRF itself. The boomerang characteristics for Branch1 and Branch2 is quite similar for both Gleeok-128 and Gleeok-256. Table 12 and Table 13 list the boomerang characteristics for Gleeok-128 and Gleeok-256 respectively considering a varied numbers of rounds for E_0 and E_1. In the tables, r_{E_0} and r_{E_1} denotes the total number of rounds in E_0 and E_1 trail respectively. As one round E_m is considered, thus the total number of rounds of the boomerang distinguisher is $(r_{E_0} + r_{E_1} - 1)$. Note that, '?' denotes that the solver has not stopped.

The Rectangle attack, introduced in [BDK01, KKS00], modifies the setting of the boomerang attack from adaptive chosen plaintext/ciphertext to chosen plaintext, albeit with a reduced probability of finding quartets. The attack's notion is quite similar to that of the boomerang attack, where two shorter trails are combined to form a longer trail over a larger number of rounds. In general, boomerang distinguisher with probability p for a n-bit primitive can be converted to rectangle one if $\sqrt{p} \geq 2^{\frac{-n+1}{2}}$, which guarantees a much lower likelihood of finding a quartet compared to the boomerang attack. Thus only some of the boomerang distinguishers in Table 12 and Table 13 can be modified to rectangle

Table 12: Maximum probability of boomerang characteristics for the underlying PRPs in Gleeok-128. The entry x in the table corresponds to the probability 2^{-x}.

(a) Probability for Branch1/Branch2

r_{E_1} \ r_{E_0}	2	3	4
2	9	25	58
3	21	39	73?
4	53	71	100?

(b) Probability for Branch3

r_{E_1} \ r_{E_0}	2	3	4
2	8	20	44
3	20	32	56
4	44	56	80

Table 13: Maximum probability of boomerang characteristics for the underlying PRPs in Gleeok-256. The entry x in the table corresponds to the probability 2^{-x}.

(a) Probability for Branch1/Branch2

r_{E_1} \ r_{E_0}	2	3	4
2	9	26	64
3	21	42	80?
4	57	80?	105?

(b) Probability for Branch3

r_{E_1} \ r_{E_0}	2	3	4
2	9	23	48
3	20	34	62
4	44	60	86

ones. In particular, distinguishers whose probability is more than $2^{-63.5}$ and $2^{-127.5}$ for Gleeok-128 and Gleeok-256 respectively, can be modified to mount rectangle attack.

4.6 Meet-in-the-Middle Attacks

In the meet-in-the-middle attack, the adversary needs to determine matching intermediate states from the plaintexts and the corresponding ciphertexts by making guesses about the involved round keys. The final XOR operation of Gleeok-128 is composed of three 128-bit states. Hence, to mount a meet-in-the-middle attack on Gleeok-128, the adversary is required to guess two out of three 128-bit output of the branches which will make the attack less efficient than the brute force attack. If the strategy involves starting from the intermediate states (as in [SA09]), in such cases also the adversary is required to guess the corresponding intermediate 128-bit values of the other two branches.

In similar way, mounting meet-in-the-middle attack on Gleeok-256 is also less efficient than the brute force attack. Thus, we believe that both Gleeok-128 and Gleeok-256 are secure against meet-in-the middle attacks.

4.7 Invariant Subspace Attack

In [BCLR17], Beierle et al. demonstrated the possibility of launching an invariant subspace attack on a block cipher. This attack relies on finding a non-trivial invariant for the substitution layer. This invariant is required to have two properties: being invariant under the linear layer matrix L used by the cipher and its linear space should contain all the differences between the round keys. It is shown for the designs consisting of simple key scheduling function (like round keys are generated by adding some round constants to the master key), all the differences between the round keys can be easily computed as the differences of the round constants and these differences must belong to the linear space of the invariant. In [BCLR17], the authors calculated $W_L(D)$, where D represents the set of all round constant differences. $W_L(D)$ represents the smallest L-invariant subspace that contains D. For a n-bit block cipher, if the dimension of $W_L(D)$ is at least $(n-1)$ and the S-box does not have any linear component, then there is no non-trivial invariant on

the S-box layer. If the dimension of $W_L(D)$ is less than $(n-1)$ (and close to n), then the existence of any non-trivial invariant can be detected by further analysing the properties of the S-box layer.

The round keys used in the underlying permutation of both Gleeok-128 and Gleeok-256 is not derived by directly adding some round constants to the master key. Instead a linear permutation is used to update the round keys. Hence, it can be concluded that the key scheduling function is not simplistic in nature and the set D can not be constructed in a straightforward way. Thus, we believe that any non-trivial invariant subspace can not be found by directly applying the approaches of [BCLR17].

4.8 Slide Attacks

The slide attack was introduced by Biryukov and Wagner [BW99] and exploits block ciphers with identical round functions. Consider a r-round block cipher E with identical round function F (*i.e.,* the round keys are also identical), such that $E = F \circ \cdots \circ F = F^r$. In this attack, a "slid pair" is constructed, consisting of $(P, E(P))$ and $(P', E(P'))$, where $P' = F(P)$. If F is a weak permutation, there is a high probability that the relation $E(P') = F(E(P))$ holds. The slide attack can also be mounted if the block cipher utilizes a key scheduling function instead of identical subkeys. If the key scheduling function has a periodicity of p, it is possible to combine p rounds as F and mount the slide attack. In general, it can be concluded that if the cipher is composed of R rounds and $R \leq p$, then it is not possible to mount slide attack.

In the context of analyzing the security of the underlying permutations against the slide attack, it is necessary to analyze the periodicity of the key scheduling functions. Both Gleeok-128 and Gleeok-256 employ permutation-based key scheduling functions, which requires assessing the periodicity of these functions. Of particular interest to us is the multiplicative order of the linear permutation (as it fixes the periodicity) responsible for updating the round keys. Upon observation, we found that for all three branches of Gleeok-128, the multiplicative order of the linear permutation is 32, which is more than the total number of rounds of these branches. This effectively eliminates the possibility of mounting a slide attack. In the case of Gleeok-256, the minimum multiplicative order (16) is observed in Branch 3, which is also more than the total number of rounds in the underlying permutation. Consequently, any potential for a slide attack is effectively eliminated.

4.9 Security Against Key Recovery Attacks

Key recovery attacks on permutations are typically carried out by leveraging underlying distinguishers. The attack process involves considering an underlying distinguisher and then prepending or/and appending a certain number of rounds to make guesses about the related key bits that satisfy the distinguisher. In the case of both Gleeok-128 and Gleeok-256, which consist of three branches, the adversary needs to construct three separate distinguishers for each branch using the same set of plaintexts. Constructing distinguishers for three different branches simultaneously presents a significant challenge, leading us to believe that the PRFs are secure against key recovery attacks. Besides, it requires guessing two outputs of the permutations to perform the backward computation from the output. This makes attaching a key recovery phase in the last rounds more difficult.

Furthermore, the key scheduling function of Gleeok-128 employs distinct subsets of bits from the 256-bit master key to initialize K_1 and K_2 for each of the three branches. Let's consider the 256-bit master key as $k_0||\cdots||k_{255}$. Specifically, for Branch1, Branch2, and Branch3, K_1 utilizes $k_0||\cdots||k_{127}$, $k_{128}||\cdots||k_{255}$, and $k_{64}||\cdots||k_{191}$ respectively. Consequently, even if an adversary attempts a key recovery attack based on 32 bits of the distinguisher, they would need to simultaneously guess a total of $3 \times 32 = 96$ bits. Indeed, our evaluation finds that the best attack on Gleeok-128 and Gleeok-256 are 5-round integral

distinguishing attacks for Gleeok-128 and Gleeok-256, rather than key recovery attacks. As 256/512-bit guessing for Gleeok-128 and Gleeok-256 is required to compute the last rounds from the ciphertext, respectively, developing a method to effectively incorporate a key recovery phase utilizing a distinguisher in such a scheme is non-trivial task and remains an open problem.

4.10 Grover's Attack

A quantum adversary can leverage Grover's algorithm [Gro96] to perform an exhaustive key search using a limited number of plaintext-ciphertext pairs. For Gleeok, this requires $2^{256/2} = 2^{128}$ iterations. As a result, Gleeok offers 128-bit key recovery security against Grover's attack. We note that Gleeok specifically provides 128-bit key recovery security against quantum adversaries restricted to classical online queries only and does not claim security against quantum adversaries with access to quantum online queries.

4.11 Other Attacks

The security of Gleeok is also analyzed against the yoyo attack [BBD+98], exchange attack [BR19] and mixture differential attack [Gra18]. Note that, all these attacks are mounted either on word-oriented ciphers or by using decryption queries. As Gleeok is a bit-based cipher and does not have a decryption oracle, we can conclude that these attacks can not be mounted on both variants of Gleeok.

5 Application to Low-latency Authenticated Encryption

An interesting application of Gleeok is authenticated encryption (AE), which simultaneously achieves confidentiality and authenticity. AE has been one of the central topics in symmetric-key cryptography. In particular, we aim to design a low-latency AE dedicated to short inputs based on Gleeok. This could be particularly useful in memory encryption, such as Intel SGX's memory encryption engine (MEE) [Gue16b, Gue16a], which encrypts the main memory of computers to realize a secure program execution using a small physically protected area inside the CPU (also known as Trusted Execution Environment, TEE). In typical memory encryption schemes including MEE, the entire memory is divided into short units (512 bits), and each unit is encrypted by an AE scheme. Note that the full memory encryption scheme is more complex, it implements what is called an authentication tree [HJ05] consisting of MACs/hash functions and AES. We focus on AE dedicated to short, fixed-length inputs that perfectly fit the memory encryption described above. Another potential application of low-latency AE is the forthcoming Beyond 5G/6G network as mentioned in Section 1. This feature is expected to realize various applications needing (near) real-time remote control such as remote surgery and remote car control. Our study will be useful in designing AEs in these applications as well. For the basics of nonce-based AE, we refer to e.g. [BN08, Rog04] for the syntax and the security notions.

Mode consideration. OCB [KR11] is a quite efficient, parallelizable AE scheme and has a low latency if the underlying primitive is, however, it requires an invertible primitive while Gleeok is not invertible. OTR [Min14] has mostly equivalent features to OCB and it is inverse-free, however, its 2-round Feistel structure incurs some latency overhead, hence we consider it is not an ideal solution if latency is the primary goal.

We consider decryption latency to be more critical than encryption latency in many applications. For example, if we use memory encryption schemes (e.g.) in a TEE, encryption latency affects the latency of memory write operation, while decryption latency affects the latency of memory read operation. The latter is much harder to hide than

the former in general. These observations suggest that the encrypt-then-MAC (EtM) composition is the best option since decryption and verification of the ciphertext can be done in parallel. Indeed, Intel SGX's MEE adopted a variant of AES-GCM– it can be seen as an optimized variant of EtM – that is tailored to short inputs and is suitable for low-latency implementation. GCM uses GHASH (a polynomial hash function) for its MAC part, however it incurs delay due to the structure. MEE's AE instead adopted a simpler hashing, which we call inner product (IP) hashing. Concretely, we fix the message length as a multiple of n (the block size of Gleeok we use). Suppose a positive integer s which is a factor of n and let $w = n/s$. IP hashing over $\mathrm{GF}(2^s)$, which we call IP_s, takes a key $K = (K_1, \ldots, K_{wm}) \in \mathsf{GF}(2^s)^{wm}$ and a message $M = (M[1], \ldots, M[m]) \in (\{0,1\}^n)^m$ is defined as

$$\mathsf{IP}_s(K, M) = \sum_{i=1,\ldots,m, j=1,\ldots,w} K_{(i-1)\cdot w+j} \cdot M[i,j],$$

where multiplications and additions are over $\mathrm{GF}(2^s)$ and $M[i] = (M[i,1] \| \ldots \| M[i,w])$[9]. Each $M[i,j]$ is considered as an element of $\mathrm{GF}(2^s)$ in the conventional way. The key length is as long as $|M|$, hence only practical for short inputs. Note that for fixed-input length, the length annotation as in GCM is not needed to ensure security.

To demonstrate the utility of Gleeok, we follow the approach of MEE. Using Gleeok-128, we compose the counter (CTR) mode (more precisely, the same as the 96-bit nonce version of GCTR, the one underlying GCM) and a concatenation of two independent instances of IP_s for $s = n/2 = 64$ in the EtM composition. Namely first, we perform GCTR encryption to obtain the ciphertext and apply two IP_s functions to the ciphertext and concatenate two 64-bit outputs to have 128-bit hash value. This value is finally xored with $\mathsf{Gleeok\text{-}128}(K_e, N\|0^{32})$ for encryption key K_e and 96-bit nonce N, just in the same manner as GCM. Keys for CTR and IP hash functions are independent for simplicity, but the latter can be derived and precomputed from Gleeok-128 using an appropriate domain separation as with GCM. Let $\mathsf{Gleeok\text{-}128\text{–}IP}_{64}$ be the resulting nonce-based AE with 128-bit tag. To be a bit more concrete, for a 96-bit nonce N and a plaintext M, where $|M| = mn$ for a fixed m, we first compute $C = M \oplus \mathsf{GCTR}[\mathsf{Gleeok\text{-}128}_{K_e}](N, m, 1)$ (GCTR using Gleeok-128 as the primitive to have mn-bit keystream, using the internal 32-bit counter starting from 1). The authentication tag is $T = (T[1]\|T[2]) \oplus \mathsf{GCTR}[\mathsf{Gleeok\text{-}128}_{K_e}](N, 1, 0)$, where $T[i] = \mathsf{IP}_s(K_H^i, C)$ and the hash key is $K_H = (K_H^1, K_H^2)$, each mn bits. The encryption output is the tuple (C, T). Considering memory encryption as a primary application, we do not consider associated data as the nonce can contain required additional information (e.g. the leaf address of the authentication tree). Not only MEE, the modern memory encryption schemes such as [TSB18b, SNR+18b] use a fixed 512-bit plaintext ($m = 3$) because it fits in the cache line of common CPUs. Thus, supporting this fixed length is important.

Security. The provable security bound of $\mathsf{Gleeok\text{-}128\text{–}IP}_{64}$ against nonce-respecting adversary is quite easy to derive. Observe that IP_s for m-block input is $1/2^s$(-almost) XOR uniform, namely, $\mathsf{IP}_s(K, X) \oplus \mathsf{IP}_s(K, X')$ for any $X \neq X'$ is uniformly random when the key K is random (note that the input length does not affect the bias because of the key being long as input). This also implies that two parallel applications of IP_s yield a 128-bit universal hash function with $1/2^{2s}$(-almost) XOR uniformity. Consider an adversary $\mathcal{A}$ against $\mathsf{Gleeok\text{-}128\text{–}IP}_{64}$ who uses q_e encryption queries, q_d decryption queries. The privacy (confidentiality) bound of $\mathsf{Gleeok\text{-}128\text{–}IP}_{64}$ against $\mathcal{A}$ is at most the (computational) PRF advantage of Gleeok-128 and the authenticity (integrity) bound is at most $q_d/2^{2s}$ plus the PRF advantage of Gleeok-128. As $2s = n = 128$, $\mathsf{Gleeok\text{-}128\text{–}IP}_{64}$ has 128-bit security in terms of data complexity. The proof is quite straightforward since $\mathsf{Gleeok\text{-}128\text{–}IP}_{64}$ is a generic EtM composition, hence we omit it here. AES-GCM has

[9] This should be written as $\mathsf{IP}_{s,n,m}$; we assume n and m are fixed and understood from the context.

64-bit security [IOM12, NOMI15] due to the use of 128-bit block cipher. Hence, we also double bit security. We stress that ours is dedicated to short inputs, while AES-GCM is a general-purpose AE.

Section 6 describes our implementations of Gleeok-128-IP$_{64}$ in detail, its performance figures, together with a comparison with AES-based counterparts. We also benchmark the generic-purpose AE variant by replacing AES in AES-GCM by Gleeok-128; see Sect. 6.3.

6 Hardware Synthesis

A low-latency hardware-based PRF circuit exhibits a critical path length that allows for high clock frequencies which, in turn, guarantee adequate throughput rates. In integrated circuit technology, the critical path denotes the signal path with the largest delay and is a function of the circuit depth, the utilised logic gates, and their underlying physical properties governed by the selected cell library. Naturally, a construction can exhibit a small critical path and still require several clock cycles for a single computation. This is a trademark, for example, of lightweight block ciphers such as GIFT [BPP+17], SKINNY [BJK+16] and Midori [BBI+15] whose low latency round function circuits need to be invoked in a repeated fashion to compute one encryption. In order to truly achieve a low latency hardware-oriented cipher, the corresponding circuit needs to execute its computation in as few clock cycles as possible by means of unrolling the round function, i.e., by calculating multiple round function invocations in the same clock cycle through replicating and serially chaining the round functions circuits. This design strategy is at the core of the PRINCE [BCG+12] block cipher whose circuit offers low latency guarantees in the fully unrolled setting in which one encryption is executed in a single clock cycle. This means that the PRINCE circuit is completely combinatorial without any storage elements in the form of flip-flops that hold the intermediate cipher state. In the PRF realm, Orthros [BIL+21] is a PRINCE-inspired fully unrolled pseudorandom function that offered further latency gains through careful choice of the substitution and linear layers as part of the round function. The circuits presented in this work follow in the lines of both PRINCE and Orthros.

Modus Operandi. We use the *Synopsys Design Compiler* to generate netlists of our designs using the fast experimental NanGate 15 nm and the industry-grade TSMC 28 nm cell libraries. The exact synthesis options are explained in more detail in the following sections. The obtained netlist fragments are then analysed for correctness and passed through a testbench that records the switching activity from which we then derive the power and energy consumption by means of the *Synopsys Power Compiler*. Note that the HDL sources for related schemes such as Orthros are publicly available.[10]

6.1 PRF Circuits

Of late there have been many unique approaches to construct S-box circuits to minimize circuit latency [LMMR21, BDD+23]. For example in the SPEEDY family of block ciphers, the authors correctly argue that the lower the total capacitive load seen by the output of a gate, the faster the delay of signal transitions through it. As a result by using specific standard cells from the NanGate libraries, the authors tailor-make the structure of both the S-box and the linear layer, by including a network of buffers and adopting a topology that ensures that the capacitive load across the circuit is well-balanced.

However, for the experiments in this work, we used a lookup-table-based approach, in which the description of the S-box is passed in table form to the synthesizer. This design choice is due to the following reasons.

[10] https://github.com/subhadeep-banik/orthros.

- The LUT-based synthesis approach applied to the 3/4/5-bit S-boxes utilized in Gleeok already yields convincing speed figures as to construct a latency-competitive PRF supporting 256-bit keys. The fact that LUT-based synthesis leads to fast circuits has already been observed and utilized to achieve latency-efficient circuits for Orthros [BIL+21] and for the small-dimension block cipher BipBip [BDD+23]. Note that the LUT-based approach does not preclude further optimisations in the same vein as exemplified for SPEEDY nor does it prove that comparably smaller-width S-boxes of Gleeok are more suitable in the creation of latency-efficient than wider S-boxes as the 6-bit component in SPEEDY.[11]
- Although the SPEEDY approach of designing and synthesizing S-boxes leads to more optimized fully unrolled encryption circuits, it comes with a drastic power consumption overhead of factor (10-100x) which we wanted to avoid in the design of Gleeok.
- The LUT-based approach, as demonstrated with Orthros and BipBip, and the SPEEDY technique generalize well across a wide range of cell libraries and thus we are optimistic that the presented synthesis figures in this work translate well to other foundries.
- We compare Gleeok against a wide selection of related fully-unrolled, low-latency ciphers. As such, a fair comparison can only be established when all schemes are synthesised in the same manner without gate-level optimisation of one scheme at the expense of others. LUT-based compilation facilitates this goal.

We remark that our final target is the synthesis of the entire block cipher which involves simultaneous optimization of multiple iterations of round function modules connected serially. After extensive experimentation, we have come to the conclusion that more than the S-box architecture (especially when the size of the S-box is limited to 4-bits), the factor that most affects the final circuit is the compilation options presented to the synthesizer. We used 2 compile options:

F1 We compile each S-box separately using the `compile_ultra` directive and then re-compile the entire block cipher (again using `compile_ultra`), asking it not to alter the individual S-box architecture (usually done by using the `set_dont_touch` directive).

F2 We compile the entire block cipher architecture using the `compile_ultra` directive, without synthesizing any sub-component separately.

We repeated the synthesis for the entire Gleeok-128, Orthros, QARMA-128, Midori-128, PRINCE circuits for both the LUT and algebraic representations of the S-boxes, using both compilation options **F1, F2**. Following the synthesis workflow in Orthros [BIL+21], we first obtain a circuit with minimal area and then we restrict the latency incrementally till we obtain the circuit with optimal delay. Fig 3 shows the area/latency trends for both the NanGate 15 nm and TSMC 28 nm libraries. We observed that irrespective of the S-box architecture, the data point optimal with respect to latency ($\mathcal{L}$) was likely to be obtained for option **F2** whereas the data point optimal regarding circuit area ($\mathcal{A}$) was likely to be obtained for option **F1** for both the libraries. Figure 3 captures the result for the Gleeok-128 circuit.

In Table 14 (for the NanGate 15 nm cell library) and in Table 17 (for the TSMC 28 nm cell library), we compare our fully unrolled combinatorial pseudorandom functions against related low latency schemes in the literature in order to establish the competitiveness of

[11] A synthesis approach somewhere halfway between the LUT and SPEEDY techniques consists in mapping the algebraic S-box expressions directly to a circuit by restricting the choice of logic gates. The consequence of this are slower circuits compared to the LUT and SPEEDY equivalents as the synthesizing tool is not able anymore to use the full capabilities of a cell library such as components with high drive strengths.

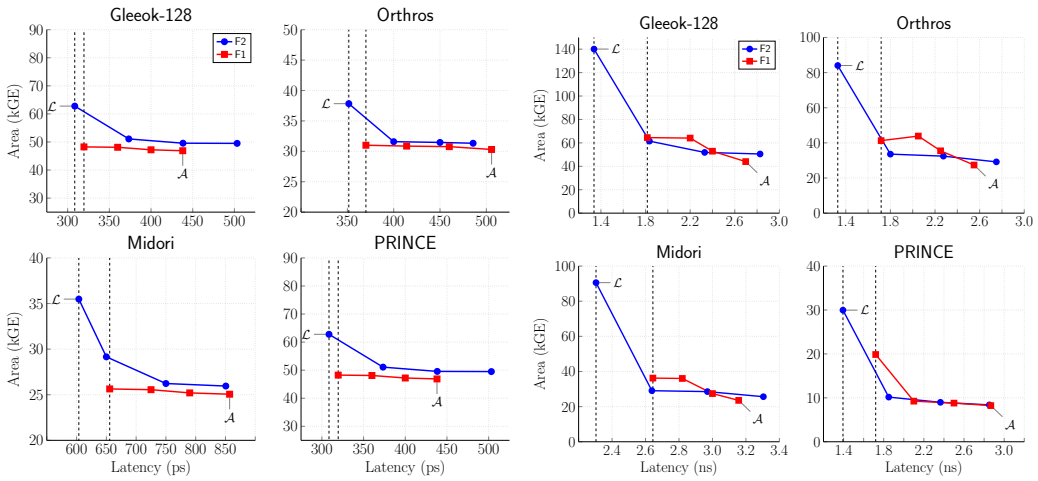


Figure 3: Area/latency comparison of Gleeok-128 with compilation options **F1, F2**

our designs. Both latency-optimised circuits $\mathcal{L}$ (following **F2**) and area-optimised circuits $\mathcal{A}$ (following **F1**) are tabulated.

6.2 IP_{64} Circuits

The number of parallel Gleeok-128 cores in Gleeok-128–IP_{64} is parameterised by the message block length and given by $m+1$ alongside two IP_{64} modules containing each $2 \cdot m$ GF(2^{64}) multipliers. These multipliers are implemented as fully combinatorial Karatsuba circuits that recursively multiply bits over increasingly smaller subfields. At its core, a Karatsuba multiplier circuit is an array of 2-bit multipliers over GF(2^2) passed as LUTs to the circuit compiler. The corresponding 4-bit outputs can then be used to compute multiplications over GF(2^4) using only linear XOR gates, which then are used to linearly compute GF(2^8) multiplications. Proceeding in this manner, it is possible to obtain multiplication circuits for any number of bits. The nature of Gleeok-128–IP_{64} allows for the encryption and authentication of m 128-bit blocks in a single clock cycle thus precluding the need for any storage elements. A schematic depiction of such a circuit is shown in Figure 4a. In this arrangement, the critical path of the circuit comprises both the Gleeok-128 core and the subsequent IP_{64} module leading to an elevated latency compared to the bare Gleeok-128 circuit.

However, a potential application for a Gleeok-128–IP_{64} circuit lies in memory decryption where latency is a crucial metric. In the decryption setting, the Gleeok-128 cores are disconnected from the latency-heavy GF(2^{64}) multipliers as the ciphertext is directly fed into the IP_{64} components. Since Gleeok-128 already ranks as one of the most latency-efficient schemes, its deployment in Gleeok-128–IP_{64} leads to a low-latency Decrypt-then-MAC construction with minimal latency overhead. A schema of the decryption circuit is given in Figure 4b. In Table 15, we compare Gleeok-128–IP_{64} to the same scheme only having replaced the core module with fully unrolled AES-128 and AES-256 circuits for the NanGate 15 cell library (see Table 18, for the TSMC 28 nm measurements). There are a handful of ways of implementing the AES round function components with respect to different optimisation criteria. In terms of latency, the most sensible approach lies in the T-table approach where SubBytes, ShitRows and MixColumns are encoded as 32-bit table look-ups. This synthesis strategy usually leads to the most latency-efficient circuits at the expense of silicon area. Hence, we also followed the T-table implementation technique in the design of the AES–128–IP_{64} and AES–256–IP_{64} circuits. As in Section 6.1, we

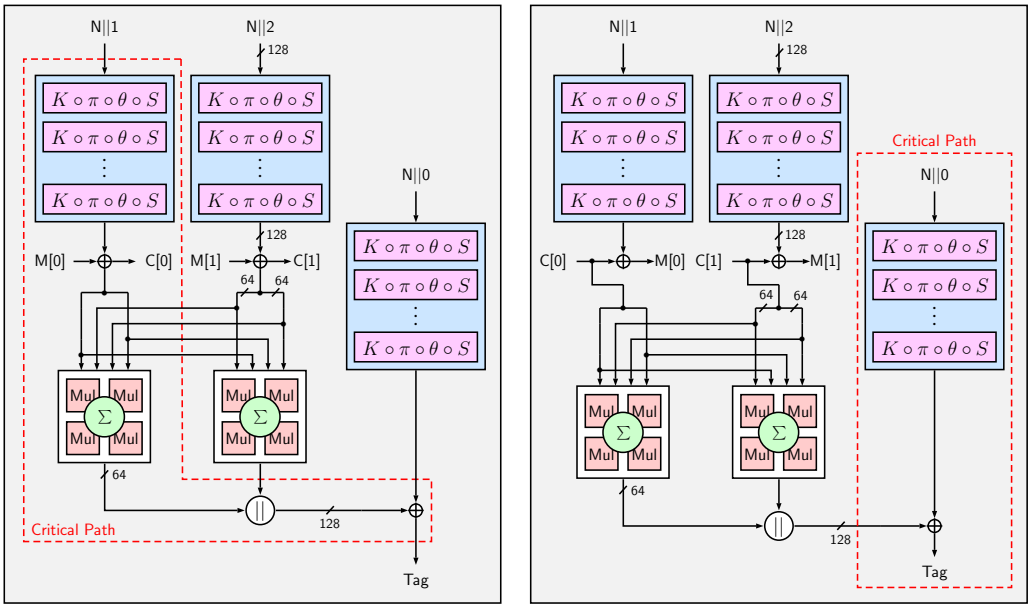
Table 14: Area-optimised and latency-optimised synthesis comparison of the investigated schemes for the NanGate 15 nm cell library at a clock frequency of 10 MHz. The maximum throughput rate is calculated as $\frac{\text{Block}}{\text{Latency}}$ where $\frac{1}{\text{Latency}}$ is the maximum clock frequency permitted by the critical path. For reference, the table also includes round-reduced variants of Gleeok-128 and Gleeok-256 for size-restricted input sizes.

Scheme	Rounds	Key/Block	Area	Latency	Max TP	Power	Energy
		Bits/Bits	GE	ps	Gbits/s	mW	nJ/Block
Gleeok-128 $\mathcal{A}$	12	256/128	46879	488.04	262.27	1.819	0.182
Gleeok-128 $\mathcal{L}$	12	256/128	62784	358.52	357.02	2.519	0.252
Gleeok-128 $\mathcal{A}$	10	256/128	38885	408.89	313.04	1.275	0.128
Gleeok-128 $\mathcal{L}$	10	256/128	49629	298.46	428.87	1.435	0.144
Gleeok-256 $\mathcal{A}$	16	256/256	124493	683.63	374.42	6.700	0.670
Gleeok-256 $\mathcal{L}$	16	256/256	173704	521.43	490.96	8.589	0.859
Gleeok-256 $\mathcal{A}$	12	256/256	93241	514.32	497.74	3.761	0.376
Gleeok-256 $\mathcal{L}$	12	256/256	125342	438.44	583.89	4.521	0.452
Orthros $\mathcal{A}$	12	128/128	30303	505.67	253.13	2.295	0.230
Orthros $\mathcal{L}$	12	128/128	37838	351.55	339.10	2.637	0.264
QARMA-128 $\mathcal{A}$	24	128/128	27564	832.33	153.79	3.689	0.369
QARMA-128 $\mathcal{L}$	24	128/128	45115	640.00	186.27	5.611	0.561
Midori $\mathcal{A}$	20	128/128	25056	856.98	149.36	3.531	0.353
Midori $\mathcal{L}$	20	128/128	35484	603.78	197.44	4.466	0.447
PRINCE $\mathcal{A}$	12	128/64	8321	516.55	123.89	0.633	0.063
PRINCE $\mathcal{L}$	12	128/64	11891	371.62	199.51	0.894	0.089

use synthesis modes **F1** and **F2** to obtain area-optimised and delay-optimised circuits respectively for Gleeok-128-IP$_{64}$. For AES-128-IP$_{64}$and AES-256-IP$_{64}$mode **F2** achieved both the most area and delay efficient circuits. We further remark that Gleeok-128-IP$_{64}$ exhibits a more than twofold latency advantage over the AES-based circuits reaching a maximum throughput rate of more than 1 Tbit/s for $m = 3$.

6.3 GCM Circuits

Gleeok-128 can be further be integrated into a GCM algorithm in the context of general-purpose authenticated encryption where a Gleeok-128 core replaces the AES-128 or AES-256 modules in order to achieve a low-latency variant. The sequential nature of the GHASH function in GCM stands orthogonal to parallelisation akin to Gleeok-128-IP$_{64}$ as described in Section 6.2. Therefore, we opt for a stateful register-based construction that separates the Gleeok-128 core from the Karatsuba GF(2^{128}) multiplication component by buffering the ciphertext in a register for one clock cycle. The total latency of this GCM encryption circuit is then $2 + a + 2 \cdot m + 1$ cycles where the first two cycles generate the hash and tag keys respectively, followed by the processing of a 128-bit associated data blocks and m 128-bit message blocks that are each processed in two clock cycles. Finally, the tag is generated in an additional cycle. The corresponding circuit is depicted in Figure 5a. In the decryption circuit, the Gleeok-128 core and the GHASH function can be executed concurrently and thus each ciphertext block can be absorbed in one clock cycle giving a total latency of $2 + a + m + 1$ cycles. The Gleeok-128-GCM decryption circuit is shown in Figure 5b. The competitiveness of Gleeok-128-GCM is demonstrated in Table 16 for the NanGate 15 nm cell library and in Table 19 for the TSMC 28 nm cell library, where we distinguish short and long input. A short input consists of 1 AD block (128 bits) and



(a) Encryption **(b)** Decryption

Figure 4: Gleeok-128–IP$_{64}$ encryption (a) and decryption (b) circuits for $m = 2$.

2 message blocks (256 bits) whereas a long input contains 2 AD blocks (256 bits) and 10000 message blocks (1.28 Mbits). The synthesis modes are identical to the ones used in Section 6.2.

6.4 Protected Circuits

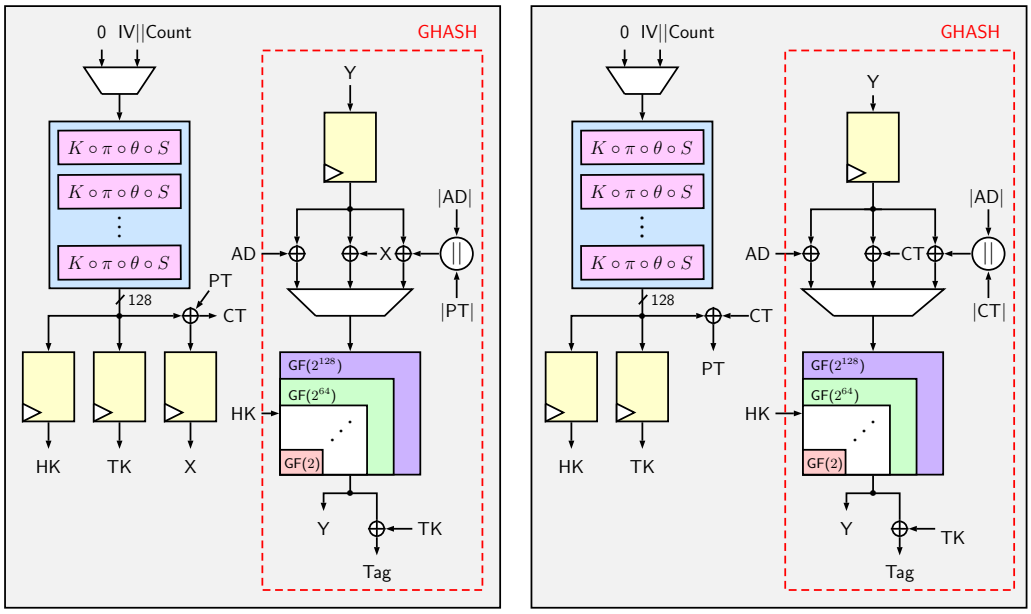
Implemented in an unprotected manner, fully unrolled, low-latency circuits are susceptible to the same power analysis attacks that plague related round-based variants with the added complexity of finding an appropriate scheme for complex algebraic expression induced by serially chained round function modules. In fact, the efficiency of common masking techniques that operate in the glitch-extended d-probing model such as Threshold Implementations (TI) [NRR06] and generic Low-Latency Masking (GLM) schemes [GIB18] is closely tied to the algebraic degree of the to-be-masked functions. As such, a masking scheme that adequately protects single-cycle cryptographic circuits has yet to be proposed, which tethers the focus of research onto masked round-based circuits that compute one round function in one clock cycle. Note that in this setting, it is not possible to decompose a round function into simpler components separated by a layer of registers as this would increase the number cycles required to compute one round function, hence the complete sharing of the round function has to be achieved in a single stage. This restriction has been investigated for the PRINCE block cipher in a work by Müller et al. [MMM21] in which the authors proposed first-order ($d = 1$) secure TI and GLM circuits. In the following, we discuss the construction of a single-stage first-order secure Threshold Implementation of Gleeok.

The basis of a first-order secure TI circuits lies in the splitting of a t-degree Boolean function $F(x_0, x_1, \ldots, x_{n-1}) = y$ into at least $t + 1$ component functions $F_i = y_i$ such that $F = \sum_{i=0}^{t} F_i$. This can be facilitated by sharing each input variables into at least $t + 1$ independent shares such that $x_i = \sum_{j=0}^{t} x_{i,j}$ and then making sure that each component function F_i is independent of at least one input share e.g., F_0 does not take any $x_{i,0}$ as input. In the TI terminology, this property is denoted as non-completeness and is usually

Table 15: Synthesis of 12-round Gleeok-128-IP$_{64}$, AES-128-IP$_{64}$, AES-256-IP$_{64}$ for the NanGate 15 nm cell library at a clock frequency of 10 MHz for different block sizes.

Scheme	Mode	Area	Latency	Max TP	Power	Energy
		GE	ps	Gbits/s	mW	nJ/Block
			$m = 1$			
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Enc	120654	860.70	148.72	5.875	0.588
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Dec	118263	576.62	221.98	4.297	0.430
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Enc	136072	609.42	210.04	5.512	0.551
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Dec	129126	384.32	333.06	3.836	0.384
AES-128-IP$_{64}$ $\mathcal{A}$	Enc	453637	1959.44	65.33	51.397	5.140
AES-128-IP$_{64}$ $\mathcal{A}$	Dec	446596	1651.62	77.50	41.718	4.172
AES-128-IP$_{64}$ $\mathcal{L}$	Enc	484893	1122.78	114.01	51.433	5.143
AES-128-IP$_{64}$ $\mathcal{L}$	Dec	480609	959.05	133.47	43.146	4.315
AES-256-IP$_{64}$ $\mathcal{A}$	Enc	618012	2387.02	53.62	92.788	9.279
AES-256-IP$_{64}$ $\mathcal{A}$	Dec	610955	2099.59	60.96	70.368	7.037
AES-256-IP$_{64}$ $\mathcal{L}$	Enc	691052	1217.39	105.14	135.906	13.591
AES-256-IP$_{64}$ $\mathcal{L}$	Dec	710846	1157.17	110.61	86.408	8.641
			$m = 2$			
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Enc	187700	916.55	279.31	9.709	0.971
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Dec	175439	623.60	410.52	6.825	0.683
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Enc	203957	621.24	412.08	7.920	0.792
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Dec	189012	383.50	667.55	5.425	0.543
AES-128-IP$_{64}$ $\mathcal{A}$	Enc	689252	2064.11	124.03	79.845	7.985
AES-128-IP$_{64}$ $\mathcal{A}$	Dec	680320	1791.72	142.88	63.280	6.328
AES-128-IP$_{64}$ $\mathcal{L}$	Enc	762850	1104.29	231.82	84.268	8.427
AES-128-IP$_{64}$ $\mathcal{L}$	Dec	759303	928.06	275.84	69.512	6.951
AES-256-IP$_{64}$ $\mathcal{A}$	Enc	952222	2460.86	104.03	144.418	14.442
AES-256-IP$_{64}$ $\mathcal{A}$	Dec	941117	2182.20	117.31	121.360	12.136
AES-256-IP$_{64}$ $\mathcal{L}$	Enc	1027892	1443.30	177.37	135.906	13.591
AES-256-IP$_{64}$ $\mathcal{L}$	Dec	1022608	1235.66	207.18	121.119	12.112
			$m = 3$			
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Enc	318220	875.79	584.62	32.691	3.269
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Dec	297248	607.67	842.56	23.664	2.366
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Enc	334231	647.21	791.09	12.935	1.294
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Dec	314772	395.22	1295.48	8.958	0.896
AES-128-IP$_{64}$ $\mathcal{A}$	Enc	1161156	2221.98	230.43	131.220	13.122
AES-128-IP$_{64}$ $\mathcal{A}$	Dec	1138181	1864.85	274.55	101.322	10.132
AES-128-IP$_{64}$ $\mathcal{L}$	Enc	1281414	1169.66	437.73	136.981	13.698
AES-128-IP$_{64}$ $\mathcal{L}$	Dec	1268915	936.81	546.54	111.257	11.126
AES-256-IP$_{64}$ $\mathcal{A}$	Enc	1601596	2580.70	198.40	237.932	23.793
AES-256-IP$_{64}$ $\mathcal{A}$	Dec	1577468	2290.58	223.52	203.403	20.340
AES-256-IP$_{64}$ $\mathcal{L}$	Enc	1740494	1393.28	367.48	234.981	23.498
AES-256-IP$_{64}$ $\mathcal{L}$	Dec	1721029	1211.07	422.76	205.772	20.577

straightforward to obtain for both linear and non-linear functions. The real challenge, however, comes in ensuring that the sharing $F_0, F_1 \ldots, F_t$ is uniform, in other words for



(a) Encryption **(b)** Decryption

Figure 5: Gleeok-128-GCM encryption (a) and decryption (b) circuits.

Table 16: Synthesis comparison of the investigated of the 12-round Gleeok-128-GCM circuits with AES-based variants for the NanGate 15 nm cell library at a clock frequency of 10 MHz. Recall that the running time is $3 + a + 2 \cdot m$ cycles and $3 + a + 1 \cdot m$ cycles for the encryption and decryption circuits respectively with $a = 1$, $m = 2$ for short inputs and $a = 2$, $m = 10000$ for long inputs. The maximum throughput is calculated for inputs with asymptotically large message sizes.

Scheme	Mode	Area	Latency	Max TP	Power	Energy	
		GE	ps	Gbits/s	mW	nJ/Short	nJ/Long
Gleeok-128-GCM $\mathcal{A}$	Enc	83474	545.35	117.36	1.504	1.203	3008.752
Gleeok-128-GCM $\mathcal{A}$	Dec	81764	537.93	237.95	1.480	0.988	1380.740
Gleeok-128-GCM $\mathcal{L}$	Enc	99938	456.91	140.07	1.561	1.249	3122.781
Gleeok-128-GCM $\mathcal{L}$	Dec	98235	449.61	284.69	1.759	1.055	1759.880
AES-128-GCM $\mathcal{A}$	Enc	247315	1380.64	46.35	4.322	3.458	8646.161
AES-128-GCM $\mathcal{A}$	Dec	245614	1374.43	93.13	5.012	3.007	5014.506
AES-128-GCM $\mathcal{L}$	Enc	272007	953.92	67.09	3.737	2.990	7475.869
AES-128-GCM $\mathcal{L}$	Dec	271072	950.42	134.68	4.335	2.601	4337.168
AES-256-GCM $\mathcal{A}$	Enc	333330	1871.83	34.19	7.146	5.717	14295.573
AES-256-GCM $\mathcal{A}$	Dec	331625	1863.85	68.68	5.877	3.526	5879.939
AES-256-GCM $\mathcal{L}$	Enc	376990	1224.13	52.28	6.404	5.123	12811.202
AES-256-GCM $\mathcal{L}$	Dec	375286	1215.91	105.27	5.403	3.242	5405.702

each input value $x_0, x_1, \ldots x_{n-1}$, each output masking $y_0, y_1, \ldots y_{n-1}$ is equally likely to occur. Non-completeness usually does not imply uniformity and even with the addition of correction terms a uniform sharing may not be obtainable for every function without increasing the number of shares or the introduction of additional randomness.
The crucial functions that require attention regarding their split into non-complete and uniform components are the three non-linear S-boxes S_3, S_4 and S_5 of Gleeok.

- S_3, S_5: Both functions are composed of a cyclic application of the 3-bit Keccak χ transformation, more specifically

$$S_3(x_0,x_1,x_2)=\begin{cases}x_0+(x_1+1)x_2\\x_1+(x_2+1)x_0\\x_2+(x_0+1)x_1\end{cases}\quad S_5(x_0,x_1,x_2,x_3,x_4)=\begin{cases}x_0+(x_1+1)x_2\\x_1+(x_2+1)x_3\\x_2+(x_3+1)x_4\\x_3+(x_4+1)x_0\\x_4+(x_0+1)x_1\end{cases}$$

  χ is a quadratic function and thus a Threshold Implementation using 3 shares is in line theoretically, but practically, only non-uniform non-complete decompositions appear feasible without injecting fresh randomness. As a remedy, Bilgin et al. proposed a 4-share circuit in [BDN+14] which we will reuse in our protected Gleeok implementation.

- S_4: Gleeok and Orthros share the same 4-bit S-box of cubic degree composed of the following individual Boolean functions:

$$S_4(x_0,x_1,x_2,x_3)=\begin{cases}x_3x_1x_0+x_3x_1+x_2x_1x_0+x_0\\x_3x_2x_1+x_3x_2+x_2x_1x_0+x_2x_1+x_1x_0\\x_3x_2x_1+x_3x_2+x_3x_1x_0+x_3x_1+x_3x_0+x_2x_1+x_2+x_1\\x_3x_2x_1+x_3x_2x_0+x_3x_2+x_3+x_2x_0+x_2+1\end{cases}$$

  Again, similarly to S_3 and S_5, a four-share decomposition is in line with the TI methodology, as the algebraic degree of the coordinate function is 3, but appears unachievable in practice. S_4 is in the cubic equivalence class $\mathcal{C}_{163}$ and thus can be decomposed in into a pair of of 4-bit permutations $S_4 = S_4' \circ A$ where A is an affine transformation using the TI tool developed in [BNN+12] such that

$$S_4' = \texttt{102483D6EBA975FC},\ A = \texttt{01235476BA98EFCD}.$$

  S_4' can now be now uniformly shared using the direct sharing approach which, in turn, results in a non-complete, uniform 5-share Threshold Implementation for S_4 as proven in [BNN+12].

Using the proposed 4-share TI for S_3, S_5 and the 5-share TI for S_5, we can sketch the circuit as shown in Figure 6. The implementation is straightforward, each branch is equipped with its own shared state register that feeds the round function which means that the register size $(4+4+5)n$ bits is where $n \in \{128, 256\}$ denotes the block size. Note that each branch of Gleeok has to be masked independently which incurs a total of $(3+3+4)n$ bits of initial randomness where $n \in \{128, 256\}$ denotes the block size.

7 Conclusion

In this paper, we propose a family of ultra-low latency PRFs and an authenticated encryption Gleeok providing ultra-low latency while supporting a 256-bit key. Gleeok provides 256- and 128-bit key recovery security against classical and Grover's attacks, respectively. We also propose Gleeok-128-IP$_{64}$ a nonce-based AE with 128-bit tag. We provide a comprehensive suite of hardware implementation results that establish the B5G capabilities of Gleeok-128 and Gleeok-256, Gleeok-128-IP$_{64}$ as well as Gleeok-128-GCM and demonstrate their competitiveness against related schemes in the literature.

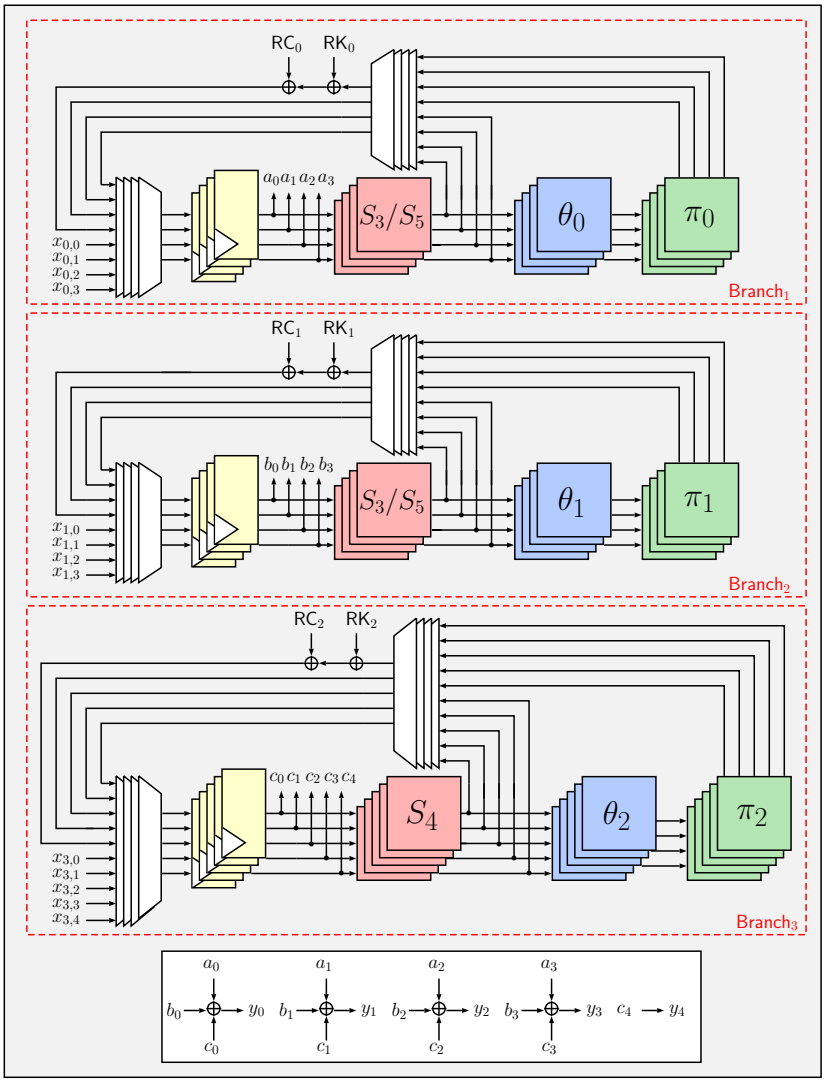


Figure 6: Round-based first-order Threshold Implementation circuit of Gleeok where branch 1 and 2 use a 4-share while branch 3 is protected with 5 shares.

Acknowledgements

We are grateful to the anonymous reviewers and the shepherd for their detailed comments and suggestions, which improved the quality and presentation of this work. This result is obtained from the commissioned research (JPJ012368C05801) by the National Institute of Information and Communications Technology (NICT), Japan.

References

[ABD+23] Roberto Avanzi, Subhadeep Banik, Orr Dunkelman, Maria Eichlseder, Shibam Ghosh, Marcel Nageler, and Francesco Regazzoni. The qarmav2 family of tweakable block ciphers. *IACR Transactions on Symmetric Cryptology*, pages 25–73, 2023.

[APJ+20] Renato B Abreu, Guillermo Pocovi, Thomas H Jacobsen, Marco Centenaro, Klaus I Pedersen, and Troels E Kolding. Scheduling enhancements and performance evaluation of downlink 5g time-sensitive communications. *IEEE Access*, 8:128106–128115, 2020.

[Ava17] Roberto Avanzi. 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 Trans. Symmetric Cryptol.*, 2017(1):4–44, 2017.

[Ava22] Roberto Avanzi. Cryptographic protection of random access memory: How inconspicuous can hardening against the most powerful adversaries be? In *CCSW@CCS*, page 1. ACM, 2022.

[BBD+98] Eli Biham, Alex Biryukov, Orr Dunkelman, Eran Richardson, and Adi Shamir. Initial observations on skipjack: Cryptanalysis of skipjack-3xor. In Stafford E. Tavares and Henk Meijer, editors, *Selected Areas in Cryptography '98, SAC'98, Kingston, Ontario, Canada, August 17-18, 1998, Proceedings*, volume 1556 of *Lecture Notes in Computer Science*, pages 362–376. Springer, 1998.

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 208–225. Springer, Heidelberg, December 2012.

[BCLR17] Christof Beierle, Anne Canteaut, Gregor Leander, and Yann Rotella. Proving resistance against invariant attacks: How to choose the round constants. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 647–678. Springer, 2017.

[BDBN23] Christina Boura, Nicolas David, Rachelle Heim Boissier, and María Naya-Plasencia. Better steady than speedy: Full break of SPEEDY-7-192. In *EUROCRYPT (4)*, volume 14007 of *Lecture Notes in Computer Science*, pages 36–66. Springer, 2023.

[BDD+23] Yanis Belkheyar, Joan Daemen, Christoph Dobraunig, Santosh Ghosh, and Shahram Rasoolzadeh. Bipbip: A low-latency tweakable block cipher with small dimensions. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):326–368, 2023.

[BDK01] Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. In Birgit Pfitzmann, editor, *Advances in Cryptology*

- *EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2001.

[BDN+14] Begül Bilgin, Joan Daemen, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Gilles Van Assche. Efficient and first-order dpa resistant implementations of keccak. In *Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers 12*, pages 187–199. Springer, 2014.

[BEK+20] Dusan Bozilov, Maria Eichlseder, Miroslav Knezevic, Baptiste Lambin, Gregor Leander, Thorben Moos, Ventzislav Nikov, Shahram Rasoolzadeh, Yosuke Todo, and Friedrich Wiemer. PRINCEv2 - More Security for (Almost) No Overhead. In *SAC*, volume 12804 of *Lecture Notes in Computer Science*, pages 483–511. Springer, 2020.

[BIL+21] Subhadeep Banik, Takanori Isobe, Fukang Liu, Kazuhiko Minematsu, and Kosei Sakamoto. Orthros: A low-latency PRF. *IACR Trans. Symm. Cryptol.*, 2021(1):37–77, 2021.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 123–153. Springer, Heidelberg, August 2016.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *J. Cryptol.*, 21(4):469–491, 2008.

[BNN+12] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. Threshold implementations of all 3 ×3 and 4 ×4 s-boxes. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2012.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi

Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 321–345. Springer, Heidelberg, September 2017.

[BR19] Navid Ghaedi Bardeh and Sondre Rønjom. The exchange attack: How to distinguish six rounds of AES with 2ˆ88.2 chosen plaintexts. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 347–370. Springer, 2019.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4:3–72, 1991.

[BSS22] Xavier Bonnetain, André Schrottenloher, and Ferdinand Sibleyras. Beyond quadratic speedups in quantum attacks on symmetric schemes. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 315–344. Springer, Heidelberg, May / June 2022.

[BVAPD11] Guido Bertoni, Gilles Van Assche, Michaël Peeters, and Joan Daemen. Keccak reference, 2011.

[BW99] Alex Biryukov and David A. Wagner. Slide attacks. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 1999.

[CFG+14] Anne Canteaut, Thomas Fuhr, Henri Gilbert, María Naya-Plasencia, and Jean-René Reinhard. Multiple differential cryptanalysis of round-reduced PRINCE. In *FSE*, volume 8540 of *Lecture Notes in Computer Science*, pages 591–610. Springer, 2014.

[CHP+18] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang Connectivity Table: A New Cryptanalysis Tool. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 683–714. Springer, 2018.

[CKLL22] Wonseok Choi, Hwigyeom Kim, Jooyoung Lee, and Yeongmin Lee. Multi-user Security of the Sum of Truncated Random Permutations. In *ASIACRYPT (2)*, volume 13792 of *Lecture Notes in Computer Science*, pages 682–710. Springer, 2022.

[DEKM16] Christoph Dobraunig, Maria Eichlseder, Daniel Kales, and Florian Mendel. Practical key-recovery attack on MANTIS5. *IACR Trans. Symmetric Cryptol.*, 2016(2):248–260, 2016.

[DHT17] Wei Dai, Viet Tung Hoang, and Stefano Tessaro. Information-Theoretic Indistinguishability via the Chi-Squared Method. In *CRYPTO (3)*, volume 10403 of *Lecture Notes in Computer Science*, pages 497–523. Springer, 2017.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual*

Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings, volume 6223 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2010.

[DKS14] Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. *J. Cryptol.*, 27(4):824–849, 2014.

[DMMR20] Joan Daemen, Pedro Maat Costa Massolino, Alireza Mehrdad, and Yann Rotella. The subterranean 2.0 cipher suite. *IACR Transactions on Symmetric Cryptology*, pages 262–294, 2020.

[EJMY19] Patrik Ekdahl, Thomas Johansson, Alexander Maximov, and Jing Yang. A new SNOW stream cipher called SNOW-V. *IACR Trans. Symmetric Cryptol.*, 2019(3):1–42, 2019.

[eri] https://www.ericsson.com/en/blog/2020/6/encryption-in-virtualized-5g-environments.

[GIB18] Hannes Gross, Rinat Iusupov, and Roderick Bloem. Generic low-latency masking in hardware. *IACR TCHES*, 2018(2):1–21, 2018. https://tches.iacr.org/index.php/TCHES/article/view/871.

[GM16] Shay Gueron and Nicky Mouha. Simpira v2: A Family of Efficient Permutations Using the AES Round Function. *IACR Cryptol. ePrint Arch.*, page 122, 2016.

[gpp] https://www.3gpp.org/ftp/Specs/archive/33_series/33.841/.

[Gra18] Lorenzo Grassi. Mixture differential cryptanalysis: a new approach to distinguishers and attacks on round-reduced AES. *IACR Trans. Symmetric Cryptol.*, 2018(2):133–160, 2018.

[Gro96] Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.

[Gue16a] Shay Gueron. A Memory Encryption Engine Suitable for General Purpose Processors. *IACR Cryptol. ePrint Arch.*, page 204, 2016.

[Gue16b] Shay Gueron. Memory Encryption for General-Purpose Processors. *IEEE Secur. Priv.*, 14(6):54–62, 2016.

[HJ05] William Eric Hall and Charanjit S. Jutla. Parallelizable Authentication Trees. In *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2005.

[HJ06] W. Eric Hall and Charanjit S. Jutla. Parallelizable authentication trees. In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 95–109. Springer, Heidelberg, August 2006.

[HP22] Paulo Sergio Rufino Henrique and Ramjee Prasad. *6G: The Road to the Future Wireless Technologies 2030*. CRC Press, 2022.

[IMO+22] Akiko Inoue, Kazuhiko Minematsu, Maya Oda, Rei Ueno, and Naofumi Homma. ELM: A low-latency and scalable memory encryption scheme. *IEEE Trans. Inf. Forensics Secur.*, 17:2628–2643, 2022.

[IOM12] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and Repairing GCM Security Proofs. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 31–49. Springer, 2012.

[JFZ+20] Zhiyuan Jiang, Siyu Fu, Sheng Zhou, Zhisheng Niu, Shunqing Zhang, and Shugong Xu. Ai-assisted low information latency wireless networking. *IEEE Wireless Communications*, 27(1):108–115, 2020.

[KKS00] John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2000.

[Knu94] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 1994.

[KR11] Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In *FSE*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, 2011.

[LMMR21] Gregor Leander, Thorben Moos, Amir Moradi, and Shahram Rasoolzadeh. The SPEEDY family of block ciphers engineering an ultra low-latency cipher from gate level for secure processor architectures. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):510–545, 2021.

[Min14] Kazuhiko Minematsu. Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2014.

[MMM21] Nicolai Müller, Thorben Moos, and Amir Moradi. Low-latency hardware masking of PRINCE. In Shivam Bhasin and Fabrizio De Santis, editors, *Constructive Side-Channel Analysis and Secure Design - 12th International Workshop, COSADE 2021, Lugano, Switzerland, October 25-27, 2021, Proceedings*, volume 12910 of *Lecture Notes in Computer Science*, pages 148–167. Springer, 2021.

[Mur11] Sean Murphy. The Return of the Cryptographic Boomerang. *IEEE Trans. Inf. Theory*, 57(4):2517–2521, 2011.

[MY93] Mitsuru Matsui and Atsuhiro Yamagishi. A new method for known plaintext attack of feal cipher. In *Advances in Cryptology—EUROCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques Balatonfüred, Hungary, May 24–28, 1992 Proceedings 11*, pages 81–91. Springer, 1993.

[nic] https://beyond5g.nict.go.jp/images/download/NICT_B5G6G_WhitePaperEN_v3_0.pdf.

[NOMI15] Yuichi Niwa, Keisuke Ohashi, Kazuhiko Minematsu, and Tetsu Iwata. GCM Security Bounds Reconsidered. In *FSE*, volume 9054 of *Lecture Notes in Computer Science*, pages 385–407. Springer, 2015.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS 06*, volume 4307 of *LNCS*, pages 529–545. Springer, Heidelberg, December 2006.

[Qua17] Qualcomm. Pointer Authentication on ARMv8.3. https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/pointer-auth-v7.pdf, 2017.

[RCPS07] Brian Rogers, Siddhartha Chhabra, Milos Prvulovic, and Yan Solihin. Using Address Independent Seed Encryption and Bonsai Merkle Trees to Make Secure Processors OS- and Performance-Friendly. In *MICRO*, pages 183–196. IEEE Computer Society, 2007.

[Rog04] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, Heidelberg, December 2004.

[SA09] Yu Sasaki and Kazumaro Aoki. Finding preimages in full md5 faster than exhaustive search. In *Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28*, pages 134–152. Springer, 2009.

[SNR+18a] Gururaj Saileshwar, Prashant J. Nair, Prakash Ramrakhyani, Wendy Elsasser, José A. Joao, and Moinuddin K. Qureshi. Morphable Counters: Enabling Compact Integrity Trees For Low-Overhead Secure Memories. In *MICRO*, pages 416–427. IEEE Computer Society, 2018.

[SNR+18b] Gururaj Saileshwar, Prashant J. Nair, Prakash Ramrakhyani, Wendy Elsasser, José A. Joao, and Moinuddin K. Qureshi. Morphable Counters: Enabling Compact Integrity Trees For Low-Overhead Secure Memories. In *MICRO*, pages 416–427. IEEE Computer Society, 2018.

[SPWW22] Ling Sun, Bart Preneel, Wei Wang, and Meiqin Wang. A greater gift: Strengthening gift against statistical cryptanalysis. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 115–144. Springer, 2022.

[ST17] Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In *EUROCRYPT (3)*, volume 10212 of *Lecture Notes in Computer Science*, pages 185–215, 2017.

[SWW21] Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the SAT method. *IACR Trans. Symmetric Cryptol.*, 2021(1):269–315, 2021.

[Tea18] The ZUC Design Team. The zuc-256 stream cipher. .http://www.is.cas.cn/ztzl2016/zouchongzhi/201801/W020180126529970733243.pdf, 2018.

[TISI23] Kazuma Taka, Tatsuya Ishikawa, Kosei Sakamoto, and Takanori Isobe. An efficient strategy to construct a better differential on multiple-branch-based designs: Application to orthros. In *CT-RSA*, volume 13871 of *Lecture Notes in Computer Science*, pages 277–304. Springer, 2023.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 287–314. Springer, 2015.

[TSB18a] Meysam Taassori, Ali Shafiee, and Rajeev Balasubramonian. VAULT: Reducing Paging Overheads in SGX with Efficient Integrity Verification Structures. In *ASPLOS*, pages 665–678. ACM, 2018.

[TSB18b] Meysam Taassori, Ali Shafiee, and Rajeev Balasubramonian. VAULT: Reducing Paging Overheads in SGX with Efficient Integrity Verification Structures. In *ASPLOS*, pages 665–678. ACM, 2018.

[Wag99] David A. Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 1999.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 648–678, 2016.

[YEP+06] Chenyu Yan, Daniel Englender, Milos Prvulovic, Brian Rogers, and Yan Solihin. Improving Cost, Performance, and Security of Memory Encryption and Authentication. In *ISCA*, pages 179–190. IEEE Computer Society, 2006.

A Auxiliary Hardware Measurements

Table 17: Area-optimised and latency-optimised synthesis comparison of the investigated schemes for the TSMC 28 nm cell library at a clock frequency of 10 MHz. The maximum throughput rate is calculated as $\frac{\text{Block}}{\text{Latency}}$ where $\frac{1}{\text{Latency}}$ is the maximum clock frequency permitted by the critical path. For reference, the table also includes round-reduced variants of Gleeok-128 and Gleeok-256 for size-restricted input sizes.

Scheme	Rounds	Key/Block	Area	Latency	Max TP	Power	Energy
		Bits/Bits	GE	ns	Gbits/s	mW	nJ/Block
Gleeok-128 $\mathcal{A}$	12	256/128	44011	2.698	47.44	1.401	0.140
Gleeok-128 $\mathcal{L}$	12	256/128	140088	1.338	95.67	3.792	0.379
Gleeok-128 $\mathcal{A}$	10	256/128	36506	2.236	57.25	0.940	0.094
Gleeok-128 $\mathcal{L}$	10	256/128	109675	1.121	114.18	2.443	0.244
Gleeok-256 $\mathcal{A}$	16	256/256	116924	3.787	67.59	5.001	0.500
Gleeok-256 $\mathcal{L}$	16	256/256	269121	2.185	117.16	11.474	1.147
Gleeok-256 $\mathcal{A}$	12	256/256	87574	2.806	91.23	2.738	0.274
Gleeok-256 $\mathcal{L}$	12	256/256	199066	1.730	147.98	6.146	0.615
Orthros $\mathcal{A}$	12	128/128	27391	2.554	50.12	1.424	0.142
Orthros $\mathcal{L}$	12	128/128	84030	1.328	96.39	4.305	0.431
QARMA-128 $\mathcal{A}$	24	128/128	27894	4.582	27.94	2.525	0.253
QARMA-128 $\mathcal{L}$	24	128/128	87583	2.43	52.78	8.002	0.800
Midori $\mathcal{A}$	20	128/128	23551	4.392	29.14	2.533	0.253
Midori $\mathcal{L}$	20	128/128	90570	2.260	56.64	8.770	0.877
PRINCE $\mathcal{A}$	12	128/64	8233	2.867	22.32	0.377	0.038
PRINCE $\mathcal{L}$	12	128/64	29965	1.394	45.91	1.571	0.157

Table 18: TSMC 28 nm comparison of 12-round Gleeok-128-IP$_{64}$ and AES-128-IP$_{64}$.

Scheme	Mode	Area	Latency	Max TP	Power	Energy
		GE	ns	Gbits/s	mW	nJ/Block
			$m = 1$			
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Enc	123778	4.380	29.22	2.850	0.285
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Dec	123769	2.766	46.27	2.823	0.282
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Enc	304195	1.773	72.19	5.512	0.551
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Dec	310194	1.365	93.77	3.836	0.384
AES-128-IP$_{64}$ $\mathcal{A}$	Enc	433991	10.985	11.652	51.397	5.140
AES-128-IP$_{64}$ $\mathcal{A}$	Dec	428665	9.468	13.519	41.718	4.172
AES-128-IP$_{64}$ $\mathcal{L}$	Enc	502831	4.602	27.814	51.433	5.143
AES-128-IP$_{64}$ $\mathcal{L}$	Dec	500498	3.332	39.445	43.146	4.315
			$m = 2$			
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Enc	203762	4.425	57.850	4.702	0.470
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Dec	203760	2.808	91.161	4.675	0.468
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Enc	510660	1.820	140.66	7.920	0.792
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Dec	498742	1.500	170.67	5.425	0.543
AES-128-IP$_{64}$ $\mathcal{A}$	Enc	669112	11.023	23.22	79.845	7.985
AES-128-IP$_{64}$ $\mathcal{A}$	Dec	665587	9.509	26.92	63.280	6.328
AES-128-IP$_{64}$ $\mathcal{L}$	Enc	972300	4.684	54.65	84.268	8.427
AES-128-IP$_{64}$ $\mathcal{L}$	Dec	970197	3.277	78.12	69.512	6.951
			$m = 3$			
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Enc	363731	4.496	113.879	15.704	1.570
Gleeok-128-IP$_{64}$ $\mathcal{A}$	Dec	363731	2.880	177.777	14.357	1.436
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Enc	883903	1.881	272.196	12.935	1.294
Gleeok-128-IP$_{64}$ $\mathcal{L}$	Dec	880199	1.450	353.103	8.958	0.896
AES-128-IP$_{64}$ $\mathcal{A}$	Enc	1139425	11.150	45.919	131.220	13.122
AES-128-IP$_{64}$ $\mathcal{A}$	Dec	1098642	9.584	53.422	101.322	10.132
AES-128-IP$_{64}$ $\mathcal{L}$	Enc	1679123	3.804	134.595	136.981	13.698
AES-128-IP$_{64}$ $\mathcal{L}$	Dec	1614014	3.332	153.661	111.257	11.126

Table 19: TSMC 28 nm comparison of 12-round Gleeok-128-GCM and AES-based variants.

Scheme	Mode	Area	Latency	Max TP	Power	Energy	
		GE	ns	Gbits/s	mW	nJ/Short	nJ/Long
Gleeok-128-GCM $\mathcal{A}$	Enc	76860	3.13	20.43	0.841	0.673	1682.421
Gleeok-128-GCM $\mathcal{A}$	Dec	75489	3.09	41.42	0.830	0.498	830.415
Gleeok-128-GCM $\mathcal{L}$	Enc	212331	1.97	32.49	2.691	2.152	5383.346
Gleeok-128-GCM $\mathcal{L}$	Dec	210827	1.92	66.67	1.759	1.055	1759.880
AES-128-GCM $\mathcal{A}$	Enc	231826	6.43	9.95	1.977	1.582	3954.989
AES-128-GCM $\mathcal{A}$	Dec	230462	6.39	20.03	2.368	1.421	2369.184
AES-128-GCM $\mathcal{L}$	Enc	380316	3.64	17.58	3.644	2.915	7289.822
AES-128-GCM $\mathcal{L}$	Dec	378812	3.59	35.65	4.337	2.602	4339.169
AES-256-GCM $\mathcal{A}$	Enc	313851	8.75	7.315	3.372	2.698	6745.686
AES-256-GCM $\mathcal{A}$	Dec	311851	8.71	14.70	2.787	1.672	2788.394
AES-256-GCM $\mathcal{L}$	Enc	561629	4.64	13.79	6.968	5.584	13963.490
AES-256-GCM $\mathcal{L}$	Dec	560122	4.60	27.83	5.938	3.563	5940.969