Enabling Memory Encryption Using Ascon Derivatives

Sergej Deutsch[1], Christoph Dobraunig[1], David Durham[1], Simon Johnson[1], Krystian Matusiewicz[1], Carlos Rozas[1], and Alexander Tereschenko[1]

[1]Intel Corporation, {firstname.lastname}@intel.com

Draft version 0.5

Abstract

Keeping data stored in memory confidential is a tough challenge when assuming an adversary with physical access to it. First, we have fine-grained access to typically 256 to 512 bits at once, so nonces or tags of the typical sizes used for authenticated encryption would lead to a sizeable overhead. Second, we have stringent performance requirements, including low latency that directly impacts CPU performance and, more generally, strict area and energy constraints, especially for the high bandwidth of memory interfaces. Currently, memory encryption solutions typically rely on the AES block cipher. In this paper, we explore ways of using Ascon as a building block for memory encryption. Our analysis shows that Ascon-based constructions achieve better performance than AES-based HCTR2 and can also compete with XTS-AES while providing larger block sizes.

1 Introduction

Encrypting memory is a challenging problem for several reasons. First, when memory is paired with a modern CPU, we typically have rather fine-grained access to 256-bit or 512-bit cache lines of data. Hence, the straightforward usage of authenticated encryption schemes with, e.g., a modest 64-bit tag and a 32-bit nonce would already lead to a sizeable memory and performance overhead. Second, the implementation characteristics of the ciphers used become increasingly important. The high memory bandwidth of modern CPU and GPU systems requires the use of multiple hardware cryptographic engines, driving the need for solutions requiring low area and energy. In addition, for CPUs, the latency of memory decryption is important, since it has a sizeable impact on performance. Simplified, if a read from memory occurs, the data from memory has to arrive at the CPU for it to continue the execution of a certain program.

Currently, one of the most widely deployed solutions for memory encryption is XTS-AES [20, 50]. Essentially, XTS-AES is a series of tweakable block cipher calls, where typically just the address is taken as the tweak. This solution has the benefit that it works without ciphertext expansion. Additionally, the memory controller typically knows the address for reads from memory before the data arrives. Hence, the tweak can be precomputed, and the added latency from decryption is roughly just one call to the AES block cipher. On the downside, the encryption is deterministic and the granularity is only 128 bits, which has been exploited in recent attacks [7, 51]. Adding a nonce to XTS-AES is conceptually possible, but would roughly double the read latency. A 128-bit block size also adds further downsides combined with XTS-AES, since it puts the birthday bound at roughly 2^{64} encrypted blocks.

Recently, the National Institute for Standards and Technology (NIST) has launched the development of cryptographic accordions [47]. In a nutshell, cryptographic accordions should act like a strong tweakable wide pseudorandom permutation (STWPRP) or a tweakable block cipher, where the block size can be chosen arbitrarily. NIST wants to focus on the construction

HCTR2 [9] instantiated using AES. Compared to XTS-AES, a solution based on HCTR2 would allow for larger encrypted blocks, e.g., 256-bit or 512-bit ones. On the downside, HCTR2 always has a critical path of roughly 2 AES computations plus 2 finite field multiplications, whether or not a nonce is included in the tweak.

In this paper, we study ways to use pseudorandom functions derived from Ascon-AEAD [17, 52] and Ascon-PRF [18] as a building block in the docked double decker [26]. On a high level, the docked double decker, as we instantiate it, has a similar cost structure to HCTR2. Per encrypted 128-bit block, we need 1 call to a strong cryptographic function and approximately 2 calls to a finite field multiplication. For both modes, the critical path passes through 2 strong cryptographic functions and 2 finite field multiplications. However, for the docked double decker, we do not need to rely on block ciphers. Hence, we get the following advantages.

- **Use Ascon's permutation instead of AES.** As shown in Table 4, Ascon has better hardware implementation characteristics compared to AES with respect to latency and area. As a result, we obtain cryptographic accordions with performance that, in contrast to HCTR2 with AES, lends itself to latency-demanding applications like memory encryption. Interestingly, the resulting solutions even offer performance that competes with AES-128-based [10] instances of XTS-AES.
- **No inverse needed.** XTS-AES and HCTR2 need AES for encryption and the inverse of AES for decryption. In contrast, the docked double decker does not require the inverse of Ascon's permutation for decryption. Hence, this provides the potential to share encryption and decryption circuits.
- **Beyond-birthday-bound security if tweaks are not repeated too often.** As detailed in Section 6, the docked double decker allows us to provide beyond-birthday-bound security in cases where the tweak is not repeated too often.
- **Larger block size.** This advantage over XTS-AES is shared with HCTR2. Having larger blocks allows software to put additional secrets and/or randomness next to sensitive values. This further complicates codebook attacks.
- **Inclusion of a nonce.** In contrast to XTS-AES, our solution allows the inclusion of a nonce without affecting the latency of decryption. Hence, the same circuitry could be reused while a nonce is provided for very sensitive memory locations.
- **Integration of side-channel countermeasures.** Ascon uses an S-box with a low algebraic degree. Such S-boxes lend themselves to side-channel countermeasures like masking [23].

Related Work. Constructions converting PRFs into (strong) pseudo-random permutations were already described by Luby and Rackoff in 1985 and 1988 [37, 38], where they showed that a 3-round Feistel network with 3 different (pseudo) random functions gives a PRP and a 4-round Feistel with 4 different (pseudo) random functions results in an SPRP. Later Ohnishi (see [55]), and Patarin [48] demonstrated that only 2 (pseudo) random functions are necessary to instantiate (S)PRPs. Later, Lucks [39] showed that the condition for the first random function in a 3-round LR construction can be relaxed. He argued that a function can be used, where the probability that two chosen inputs give a certain output difference is small. Naor and Reingold showed that the first and last round of a Luby-Rackoff construction can be replaced by pairwise independent permutations [43] and in their extended version [44] that the first and last random function can be replaced by ϵ-AXU (almost-xor-universal) functions. Iwata and Kurosawa [31] then showed that random functions in the middle layer can be the same if the function of the first layer is an AXU hash function and the function of the last layer is a uniform and AXU hash function.

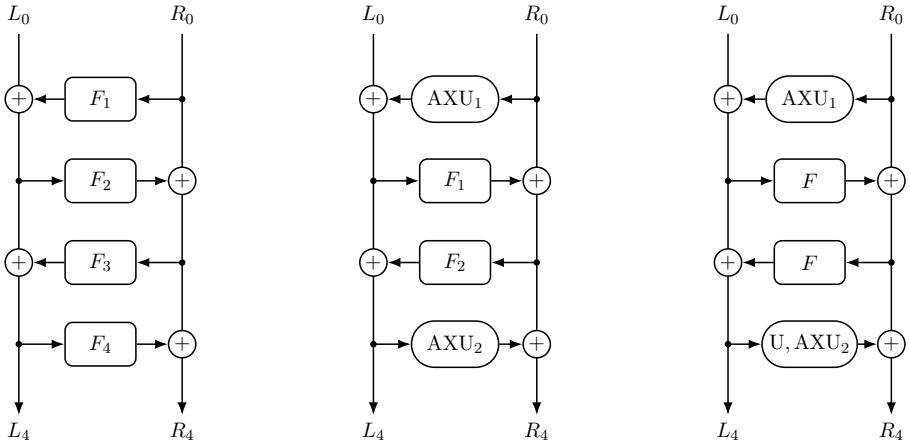

Figure 1: Evolution of SPRPs trading "heavy" PRFs for more lightweight alternatives: 4-round Luby-Rackoff, Naor-Reingold and Iwata-Kurosawa constructions.

Later, enciphering modes also emerged, that essentially act as (S)PRP, but allow for a variable block size. Early examples include candidates that follow an encrypt-mix-encrypt structure, which essentially consists of two block cipher-based encryption modes with linear mixing in between [27–29]. As another category, we have many schemes following the hash-encrypt-hash paradigm [4, 5, 54]. While algorithms of the hash-encrypt-hash category already quite resemble Naor and Reingold's principle, many of them, including HCTR2 [9], require a block cipher and its inverse for instantiation. In contrast, the docked double decker [26] just requires PRFs.

Our work can be seen as a complement to MATTER [1]. While MATTER aims to design tweakable block ciphers [35] making direct use of Ascon's permutation in Feistel networks, we aim to instantiate tweakable block ciphers and also accordion cipher modes [46] using components matching or staying close to the specifications of Ascon-AEAD and Ascon-PRF.

Outline. First, we show the used notation in Section 2. After that, we give some background on Ascon's permutation and the modes that use it in Section 3. In Section 4, we discuss how PRFs can be used to get STPRPs. Section 5 defines the STPRPs we derive from existing Ascon functionality and Section 6 discusses their security. A comparison with our constructions and constructions based on AES can be found in Section 7.

2 Notation

Table 1: Notation used throughout this paper

Notation	Definition
K, K_i	Secret key K of k bits
$L,\ J$	Secret key $L,\ J$ of n bits
N	128-bit Nonce
A	Associated data
A_i	i-th block of associated data A
P	Plaintext
P_i	i-th block of plaintext P
C	Ciphertext
C_i	i-th block of ciphertext C
Ω	ω-bit authentication tag

Notation	Definition
$Const$	Constant value
I	Input data
I_i	i-th block of input data I
O	Output data
O_i	i-th block of output data O
T	Tweak T of t bits
p, p^a	Permutation p^a consisting of a rounds of p
$x\|\|y$	Concatenation of bitstrings x and y
$x \oplus y$	Bitwise XOR of bitstring x and y
$x \odot y$	Multiplication of field elements in $GF(2^n)$
$\lfloor x \rfloor_a$	Returning the most significant a bits of bitstring x

3 Background on Ascon's Cryptographic Capabilities

3.1 Introduction to Ascon

Ascon first appeared as a family of authenticated encryption schemes entering the CAESAR competition in 2014 [12]. CAESAR was a public competition searching for authenticated encryption schemes, where Ascon v1.2 [14], containing Ascon-128 and Ascon-128a, was selected as the first choice for lightweight applications (resource-constrained environments) in 2019 [3]. In the same year, Ascon v1.2 amended by the addition of hash functions and extendable output functions [15] entered NIST's lightweight cryptography standardization process [45]. Throughout the competition, research efforts also led to the development of pseudo-random functions derived from the Ascon design [16, 18]. Finally, the authenticated encryption scheme Ascon-AEAD and hash functions were standardized in NIST SP 800-232 [52] in 2025. Ascon is also part of NIST's transition to post-quantum cryptography standards report NIST IR 8547 [42].

3.2 Authenticated Encryption with Ascon (Ascon-AEAD)

3.2.1 Working Principle of Ascon-AEAD

Ascon-AEAD is a nonce-based authenticated encryption scheme with associated data (nonce-based AEAD). As shown in Figure 2, Ascon-AEAD uses either just a 128-bit key K_1, or a 256-bit key split into 128-bit halves K_1 and K_2. Ascon can authenticate associated data A of arbitrary length and authenticate and encrypt plaintext P of arbitrary length to ciphertext C. For confidentiality, Ascon-AEAD uses a 128-bit nonce N. For authenticity, an up to 128-bit tag Ω is generated.

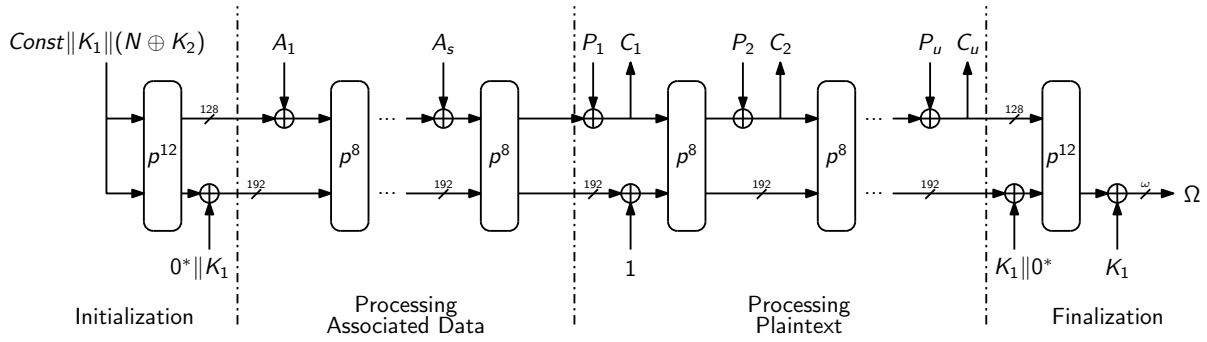

Figure 2: Ascon-AEAD: Nonce-based authenticated encryption with associated data.

3.2.2 Security of Ascon-AEAD

Security claims in NIST SP 800-232. For a nonce-respecting adversary, NIST essentially claims 128-bit security regarding confidentiality and authenticity as long as less than 2^{54} bytes are processed, which matches the submission requirements of the competition. Note that 2^{68} bytes were the original claim in the submission document [15].

In addition, there are security claims under a nonce-misuse scenario. In essence, it is claimed that Ascon-AEAD keeps its authenticity properties if nonce reuse stays under 2^8. In addition, if each pair of nonce and associated data stays unique, also confidentiality is preserved if nonce reuse stays under 2^8.

Security proofs for Ascon-AEAD. Typically, proofs of security in symmetric cryptography are made for idealized primitives. Hence, they can be seen as a kind of soundness argument for the mode of operation, showing that the higher-level construction does not introduce any weaknesses by itself. The whole construction is secure as long as the underlying primitives are secure. In the case of Ascon-AEAD, we can recommend the work of Lefevre and Mennink [32]. An overview of the results under various security models is given in Figure 3, where the size of the internal state $b = 320$ and the size of the inner part of the state that is kept secret $c = 192$ in the case of Ascon-AEAD.

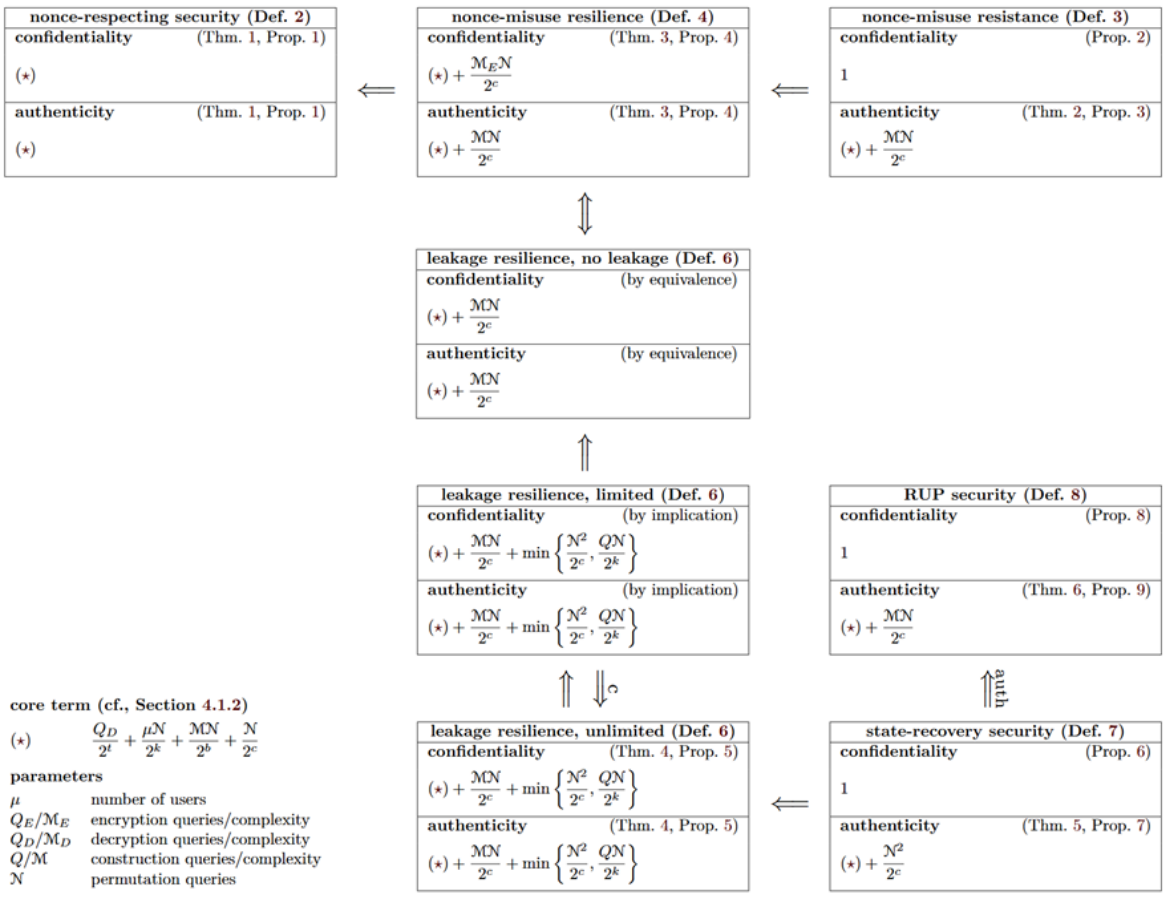

Figure 3: High-level overview of the security bounds of the mode of Ascon-AEAD under various security models. The results are given in big $\mathcal{O}$ notation. Taken from "SoK: Security of the Ascon Modes" in IACR ToSC 2025 (1) by Charlotte Lefevre and Bart Mennink licensed under CC BY 4.0.

Cryptanalytic results on Ascon. In symmetric cryptography, it is common to analyze simplified or weakened variants of the cryptographic scheme. This is typically done by reducing the number of rounds of the underlying primitive. For example, in the case of Ascon, one can reduce the number of rounds of the initialization from 12 to 7, which is the number of rounds the best attacks on Ascon cover. The remaining rounds to 12 are called the security margin. In the case of Ascon, this margin is 5 rounds, or 40% of the total rounds. For comparison, the best attacks in the single key setting reach 7 rounds for AES-128 [11] and 10 rounds for AES-256 [36], leading to a security margin of 30% and 28% respectively.

Overview of cryptanalytic results in a nonce-respecting scenario. In Table 2, we see a summary of the best attacks on Ascon-AEAD in a nonce-respecting scenario (meaning that the nonce N is chosen to be unique for every encryption). What we can see here is that the attacks with the highest number of rounds are targeting the Initialization phase of Ascon. Here, the number of rounds covered is 7 and it did not change since 2017 with improvements on the key space covered and attack complexity.

Table 2: Excerpt of best attacks applicable to round-reduced Ascon-AEAD in a nonce-respecting scenario.

Target	Rounds	Type	Time	Data	Reference
Initialization	7/12	Key Recovery	2^{77}	2^{77}	[34]
Initialization	7/12	Key Recovery	$2^{95.9}$	2^{64}	[49]
Initialization	7/12	Key Recovery	$2^{72.4}$	2^{70}	[30]
Data Processing	3/8	Forgery	2^{116}	2^{116}	[22]
Finalization	3/12	Forgery	2^{19}	2^{19}	[22]
Finalization	4/12	Forgery	2^{101}	2^{101}	[13]

Overview of cryptanalytic results in a nonce-misuse scenario. In Table 3, we have an excerpt of the best attacks on Ascon-AEAD in a nonce-misuse scenario. For the attacks shown in the table, this essentially means that the same nonce is used throughout the whole attack. As we can see, misusing the nonce increases the attack surface during the processing of data and finalization. For Data Processing, similar attacks as on the Initialization become possible, however, they become harder due to the key being diffused to 320 bits after the Initialization. The key recovery attack on 7 out of 8 rounds requires a huge data complexity, and hence, has limited impact on its practical security.

Table 3: Excerpt of best attacks applicable to round-reduced Ascon-AEAD in a nonce-misuse scenario.

Target	Rounds	Type	Time	Data	Reference
Data Processing	6/8	State Recovery	$2^{39.6}$	$2^{39.6}$	[2]
Data Processing	7/8	Key Recovery	2^{117}	2^{117}	[6]
Finalization	6/12	Forgery	2^{33}	2^{33}	[33]

3.2.3 Pruning Ascon-AEAD to Get Two 256-Bit Input PRFs

We can prune Ascon-AEAD to get two different PRFs with 256-bit input although they use the same secret key. For the first PRF, we just take the nonce N and one associated data block A_1

as input to the PRF and use the keystream used to encrypt the first ciphertext block C_1 as PRF output. It leads to the construction shown in Figure 4.

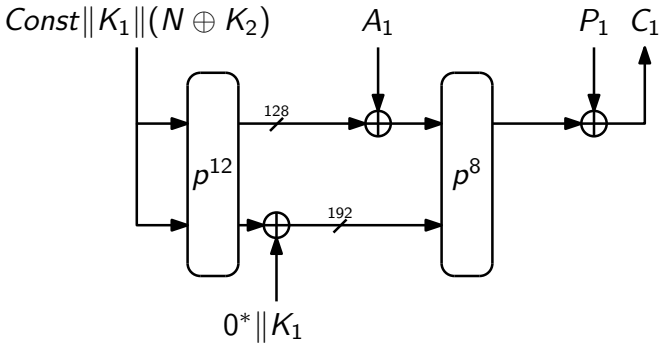

Figure 4: Option 1 to get a PRF by pruning Ascon-AEAD.

The second option is to use the nonce N and the first plaintext block P_1 as input to the PRF. We do not use any associated data blocks. The output of the PRF is the keystream used to encrypt the second ciphertext block C_2. The first ciphertext block C_1 is discarded. The result is shown in Figure 5.

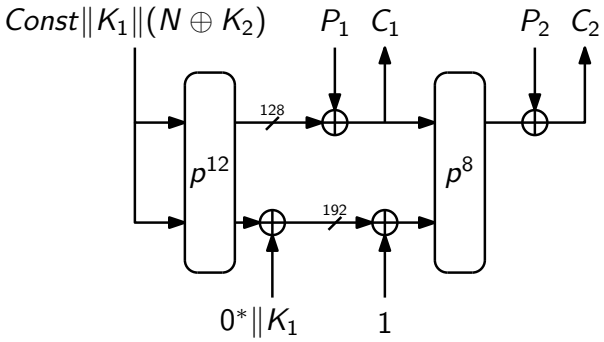

Figure 5: Option 2 to get a PRF by pruning Ascon-AEAD.

We essentially can make a PRF by pruning Ascon, since we know from the proofs and analysis that it is a good AEAD producing a pseudo-random keystream to encrypt its plaintext blocks. By making use of the domain separation between associated data and plaintext, we can craft two PRFs. However, we are considering a general-purpose PRF. This means that the nonce N could repeat significantly more than 2^8 times. Hence, we are in a nonce-misuse scenario and the analysis in this scenario outlined in Table 3 becomes relevant for its security.

3.3 A Pseudo-Random Function Using Ascon's Permutation (Ascon-PRF)

3.3.1 Working Principle of Ascon-PRF

Ascon-PRF [18] is a simple construction as shown in Figure 6. First, it expands a k-bit key K into a 320-bit secret equivalent key K_D. From then on, the input messages are absorbed in 256-bit blocks I_i. After the last input block has been absorbed, the output is generated by squeezing 128-bit blocks O_i.

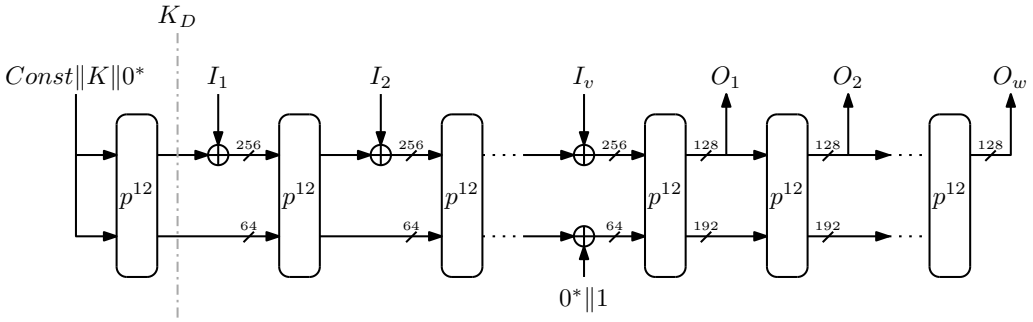

Figure 6: Ascon-PRF for arbitrary-length input and output

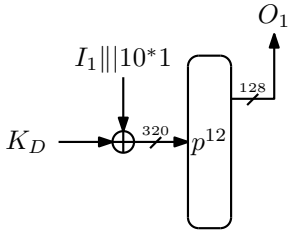

Figure 7: Ascon-PRF for one input and output block

3.3.2 Security of Ascon-PRF

From a provable perspective the situation of Ascon-PRF is more straightforward compared to the AEAD. Bart Mennink [41] has written an overview on the security of the keyed duplex including a security bound for Ascon-PRF. Following Corollary 1 of [41] and using the notation of Figure 3 while neglecting small constants using big $\mathcal{O}$ notation, we get roughly

$$Adv^{PRF}_{Ascon-PRF}(\mathcal{A}) \lesssim \frac{\mathcal{N}+\mathcal{M}}{2^c} + \frac{\mathcal{M}^2+\mathcal{N}^2+\mathcal{N}\cdot\mathcal{M}}{2^b} + \frac{\mathcal{M}^2}{2^{\min\{c+k,b\}}} + \frac{\mu\mathcal{N}}{2^k} + \frac{\mu^2}{2^k} \tag{1}$$

or

$$Adv^{PRF}_{Ascon-PRF}(\mathcal{A}) \lesssim \frac{\mathcal{N}}{2^c} + \frac{\mathcal{M}^2}{2^b-\mathcal{M}} + \frac{\mathcal{M}^2}{2^b} + \frac{\mathcal{M}^2}{2^{\min\{c+k,b\}}} + \frac{\mu\mathcal{N}}{2^k} + \frac{\mu^2}{2^k} \tag{2}$$

if $\mathcal{N}+\mathcal{M} \leq 0.1\cdot 2^c$. Please note that the first bound contains a term $\frac{2\nu^{\mathcal{M}}_{r,c}(\mathcal{N}+\mathcal{M})}{2^c}$ and the second bound $\frac{2\nu^{\mathcal{M}}_{r,c}(\mathcal{N}+1)}{2^c}$. So, $\nu^{\mathcal{M}}_{r,c}$ depends on $\mathcal{M}$. However, if we assume a maximum data complexity $\mathcal{M}$ of 2^{127}, $\nu^{\mathcal{M}}_{128,192} \leq 58$ (see Appendix A of [41]). Hence, $\nu^{\mathcal{M}}_{r,c}$ is a small constant in the simplification of the bounds.

As discussed in the design document [18], the attacks that matter for Ascon-PRF are the attacks on the initialization in the nonce-respecting setting shown in Table 2 and even more so the attacks in the nonce-misuse setting shown in Table 3. In the transition from absorbing to squeezing, we have a highly similar attack situation as during the nonce-misuse attack for data processing. Except, Ascon-PRF uses 12 rounds for the permutation everywhere and the rate during absorbing is 256 bits instead of 128 bits. Hence, we expect attacks on the PRF to cover a similar number of rounds as the attacks shown in Table 3. Even if we very pessimistically assume that 2 more rounds can be attacked compared to Ascon-AEAD, we still would have a 25% security margin.

4 Strong Tweakable Pseudo-Random Permutations from PRFs

4.1 Docked Double Decker

The docked double decker (DDD) [26] is an accordion cipher mode [46]. Accordion cipher modes are also often referred to as strong tweakable wide pseudo-random permutations (STWPRP), having the characteristic that they can instantiate blocks of arbitrary length and — dependent on the used PRF F_K — also arbitrary tweak length. In the case of DDD, the block length has to be at least $2n$ bits. In Figure 8, we show a simplification of DDD that uses only one pseudo-random function (F_K) [19].

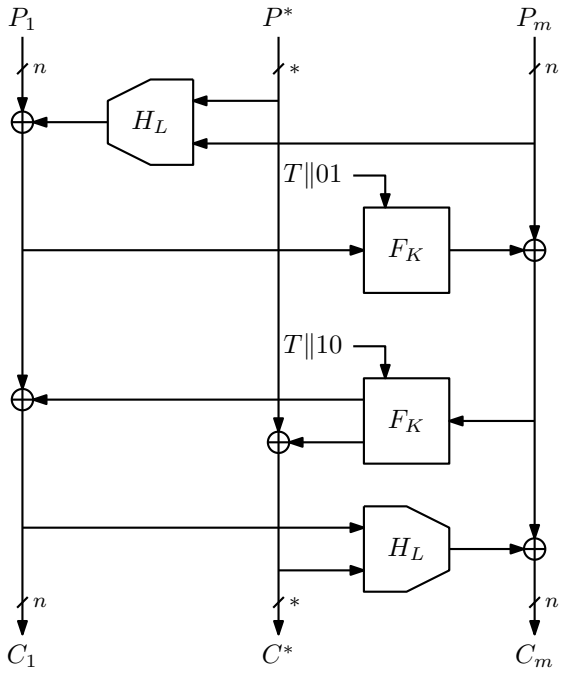

Figure 8: Docked double decker to get an STPRP for arbitrary block size (minimum block size is $2n$ bits).

As shown in Figure 8, we use the docked double decker with two hash functions H_L at the top and the bottom. As stated in [19], POLYVAL [24] seems to be a viable choice.

4.2 Docked Double Decker on $2n$ Bits

Essentially, when we use the docked double decker [26] fixed to a $2n$-bit block size as shown in Figure 9, it becomes an STPRP similar to variants of the Luby-Rackoff [37] or Naor-Reingold [44] construction. The fix to a $2n$-bit block size allows us to use a finite field multiplication over $\mathrm{GF}(2^n)$ as ϵ-AXU function. From, e.g., [24, 31], we know that the finite field multiplication over $\mathrm{GF}(2^n)$ is ϵ-AXU, with $\epsilon = 2^{-n}$.

Following Gunsing et al. [26, Theorem 1] again considering big $\mathcal{O}$ notation, we get

$$Adv^{\pm STPRP}_{DDD2n}(\mathcal{A}) \lesssim Adv^{PRF}_{F}(\mathcal{A}') + \sum_{T\in\{0,1\}^t} \binom{\mathcal{M}_T}{2} \left(\frac{1}{2^n} + \frac{1}{2^{2n}}\right). \tag{3}$$

Here, $\mathcal{M}_T$ is the number of queries made per tweak T and adversary $\mathcal{A}'$ has a total query complexity of $2 \cdot \sum_{T\in\{0,1\}^t}(\mathcal{M}_T)$. Please note that already Naor and Reingold [44] give bounds with $\frac{\mathcal{M}^2}{2^n} + \frac{\mathcal{M}^2}{2^{2n}}$ for similar constructions as Figure 9, but no (fixed) tweak.

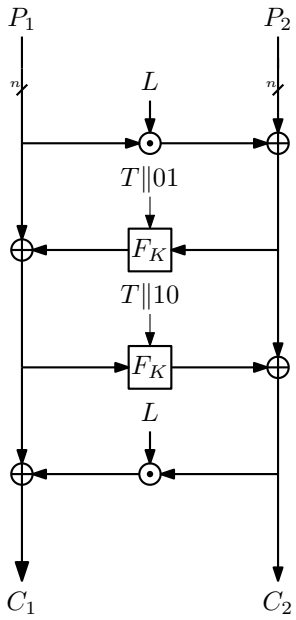

Figure 9: Docked double decker to get an STPRP for $2n$-bit block size.

5 Strong Tweakable Pseudo-Random Permutations from Ascon

5.1 STPRP Using the 256-bit Docked Double Decker

5.1.1 256-bit DDD Based on Ascon-AEAD (Ascon-DDD-P8)

Figure 10 shows the docked double decker of Figure 9 instantiated with a PRF derived from Ascon-AEAD shown in Figure 4. This results in an instance similar to the Luby-Rackoff [37] or Naor-Reingold [44] constructions. L represents a 128-bit key used in the finite field multiplication and K_1 and K_2 are the 128-bit halves of the 256-bit key $K_1||K_2$. T is a 128-bit tweak that can be used to add context like addresses, versioning information, nonces, etc. $K_1||K_2$ and T are used to precompute the 320-bit state S_1, which is then used in the processing of plaintext and ciphertext.

5.1.2 256-bit DDD Based on Ascon-PRF (Ascon-DDD-P12)

In Figure 11 we see the docked double decker from Figure 9 instantiated with Ascon-PRF for one block from Figure 7. Here, L is a 128-bit secret key uniformly chosen and used in the Galois field multiplication at the beginning and at the end of the docked double decker.

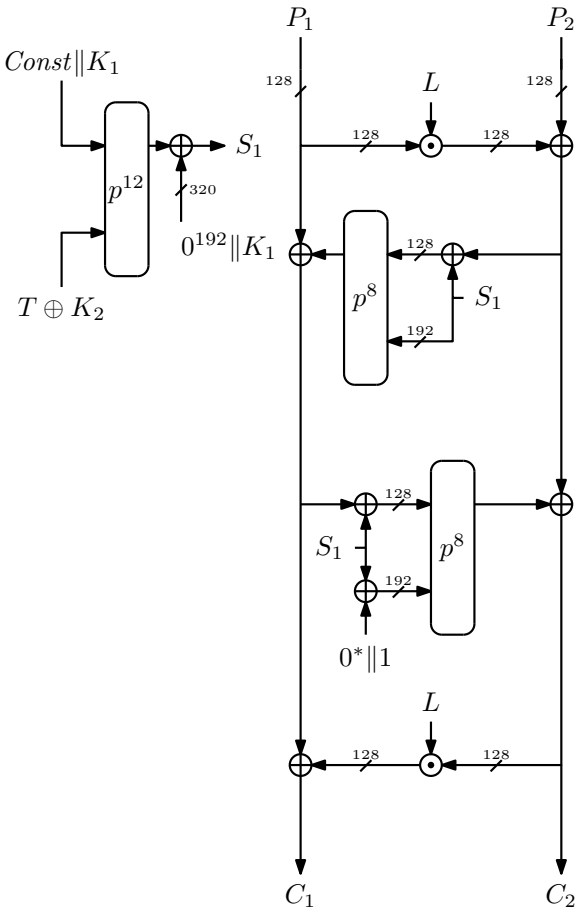

Figure 10: 256-bit docked double decker instantiated with PRF derived from Ascon-AEAD

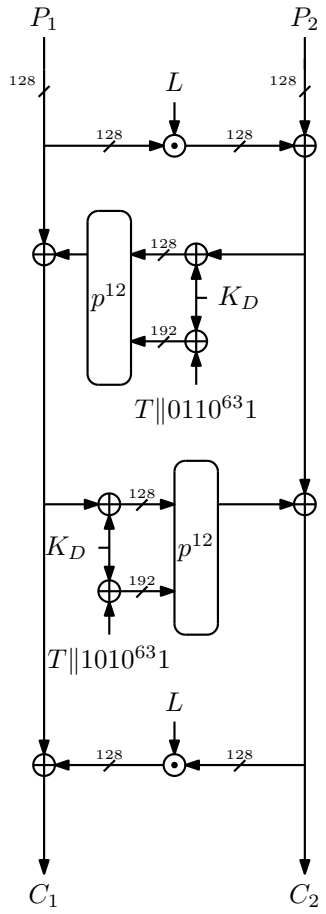

Figure 11: 256-bit docked double decker instantiated with Ascon-PRF

The one-block variant of Ascon-PRF uses the 320-bit equivalent key K_D. Its 256-bit input is formed by 128 bits coming from one Feistel branch, a 125-bit tweak T, either 01 or 10 domain separator, followed by the padding bit 1. In Figure 11, we also show the processing of all 320 bits, hence the last 64 bits ($0^{63}1$) before squeezing starts are also shown. The 128-bit output of the PRF is XORed with the other Feistel branch that was not the input. The 125-bit tweak can be freely used to absorb additional information, like addresses or a nonce.

5.1.3 Possible Ways to Choose the Key L

There are several ways to derive the secret key L besides drawing from a random 128-bit uniform distribution.

- Derive L using the PRF. For example, compute $L = \lfloor p^{12}\left((0^{254}\|11\|0^{63}1) \oplus K_D\right)\rfloor_{128}$.
- Make L address-dependent by deriving it using the PRF, e.g.,

$$L_{\mathsf{Addr}} = \left\lfloor p^{12}\left(\left(0^*\|\mathsf{Addr}\|11\|0^{63}1\right) \oplus K_D\right)\right\rfloor_{128}.$$

5.2 Further Variants of Ascon-based STPRPs

For simplicity, we focus on variants instantiated with Ascon-PRF in this section.

5.2.1 Ascon-Based STPRP with Arbitrary Block Size Using the DDD

So far, we have looked mostly at the docked double decker reduced to a strict balanced Feistel Network, so that we end up with essentially a Luby-Rackoff or Naor-Reingold construction.

However, by reducing the tweak T by 29 bits to 96 bits, we effectively can embed a counter that lets us instantiate an accordion cipher mode [46] that can process up to 2^{33}-byte blocks. In Figure 12, we give an example encrypting 512-bit blocks.

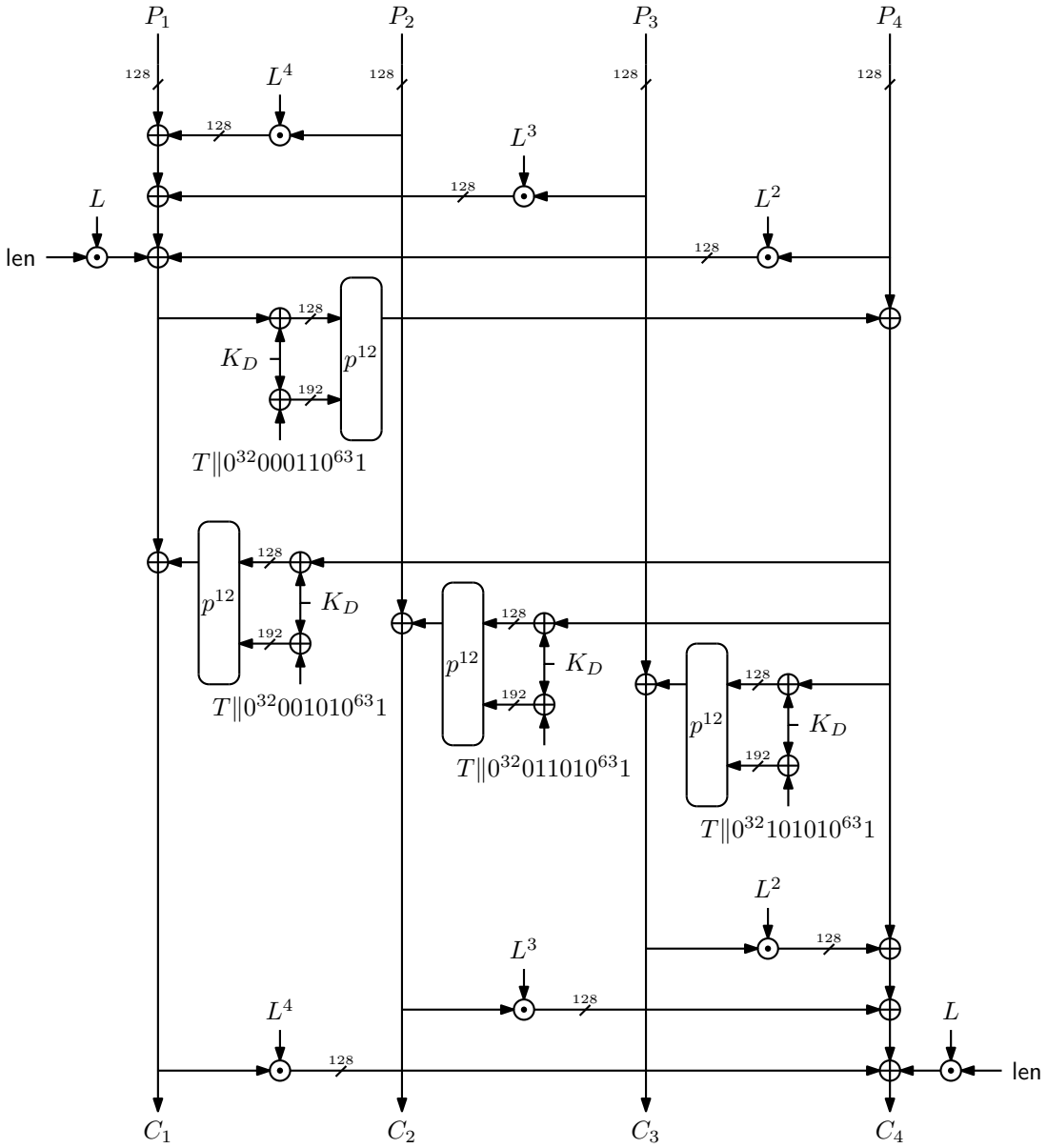

Figure 12: Example of the Ascon-PRF-based DDD instantiated for 512-bit blocks

5.2.2 Instance with Increased Outer Lanes for Increased Data Complexity

Essentially, the data complexity that the docked double decker tolerates is dictated by the birthday bound in the bitsize n of the outer lanes per tweak. So, one way to increase the tolerable data complexity of the mode is to increase the width of the outer two lanes. This, in turn, means that the PRFs have to absorb more data, which we can trade off by using smaller tweaks. For example, if we allow for 64-bit tweaks and 14-bit counters, each PRF could absorb 176 bits of data input. In Figure 13, we give an example encrypting 512-bit blocks.

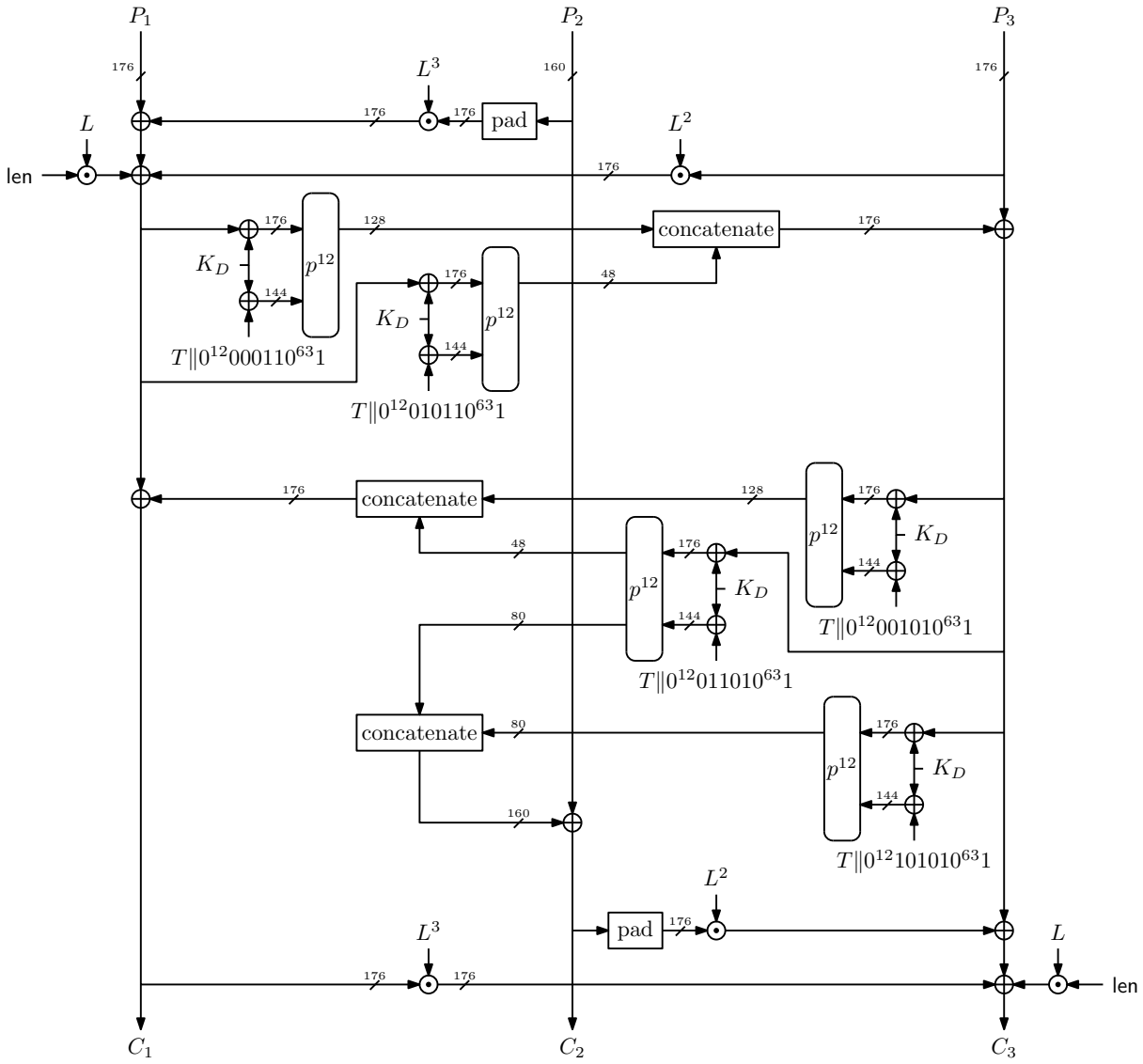

Figure 13: 512-bit docked double decker with 176-bit outer lanes instantiated with Ascon-PRF. Increasing the width of the outer lanes allows for higher data complexity.

Note that if larger tweaks are needed, additional blocks serving as the tweak can be prepended to Ascon-PRF.

6 On the Security of the Ascon-Based STPRPs

6.1 Security Claim for the Ascon-Based STPRPs

First, we want to state that the bounds presented in Table 3 and Equations (1), (2), and (3) take priority and we do not claim security outside of those bounds. Especially Equation (3) has to be considered since it essentially is limited by the birthday bound per tweak.

Apart from that, we are confident in the security claims made by the designers of Ascon that claim 128-bit security for up to 2^{64} processed blocks per key [17, 18]. For the case of Ascon-PRF-based constructions using a 256-bit key K, we think that the vast cryptanalytic results on Ascon and generic security bounds can warrant security of up to ≈ 188 bits for up to 2^{80} processed blocks per key assuming that the total adversarial data complexity for all keys stays within reasonable bounds (e.g., less than 2^{110} blocks).

6.2 Interpretation of the Security Claim and Bounds

In order to illustrate the meaning of security bounds we can consider "unlucky" and high-data-complexity scenarios where we model boundary conditions for the memory size and the number of operations performed.

As an example for the 2^{80} data limit consider a simple example of a memory system that has 2^{60} addresses spanning a single block where each address is written at most 2^{20} times, but we have a single outlier that is written 2^{50} times. From the security bound of the DDD of Section 5.1.2, where a block is 256 bits, we have

$$\begin{aligned} Adv^{\pm STPRP}_{DDD2n}(\mathcal{A}) &\lesssim Adv^{PRF}_{Ascon-PRF}(\mathcal{A}') + 2^{60}2^{39}(2^{-128}+2^{-256}) + 2^{99}(2^{-128}+2^{-256}) \\ &\lesssim Adv^{PRF}_{Ascon-PRF}(\mathcal{A}') + 2^{-28} \end{aligned}$$

If we consider the construction of Section 5.2.2 with 512-bit blocks, we get

$$\begin{aligned} Adv^{\pm STPRP}_{DDD2n}(\mathcal{A}) &\lesssim Adv^{PRF}_{Ascon-PRF}(\mathcal{A}') + 2^{60}2^{39}(2^{-176}+2^{-352}) + 2^{99}(2^{-176}+2^{-352}) \\ &\lesssim Adv^{PRF}_{Ascon-PRF}(\mathcal{A}') + 2^{-76} \end{aligned}$$

In this case, we see that we can tolerate even larger outliers than 2^{50}, e.g., 2^{70} for a single address and keep the advantage $\lesssim 2^{-37}$.

For the PRF in the above cases, we just consider the total data complexity over all keys with 2^{110} blocks and get

$$Adv^{PRF}_{Ascon-PRF}(\mathcal{A}') \lesssim \frac{\mathcal{N}}{2^{188}} + \frac{\mu\mathcal{N}}{2^{256}} + \frac{\mu^2}{2^{256}} + 2^{-98}$$

Note that for both, HCTR2 and XTS, the advantage of an attacker reaches ≤ 1 for such high data complexities.

A more complex example is shown in Figure 14. Here we have a Gaussian distribution where the peak is at 2^{19} writes per address for 2^{60} addresses and we have outliers for up to 2^{49} writes to a single address.

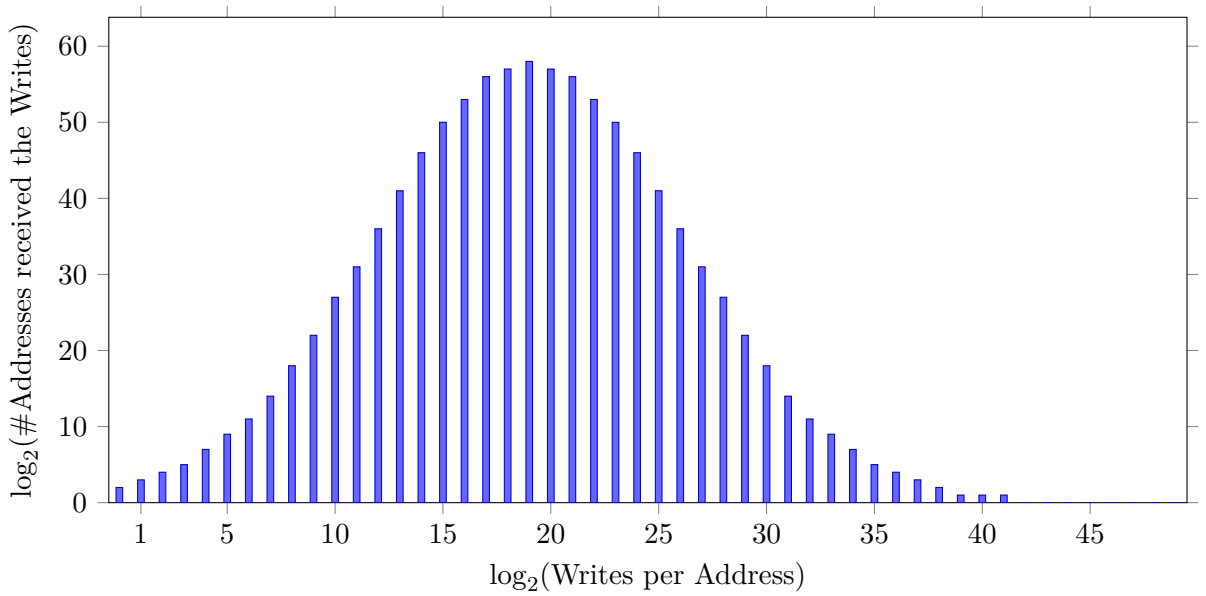

Figure 14: Example of write distribution. On the x-axis, we see the $\log_2$ of the number of writes a single address receives. On the y-axis, we see the $\log_2$ of the number of addresses that receive the number of writes per address indicated on the x-axis.

For the write pattern in Figure 14 considering the scheme of Section 5.1.2, we have encrypted a total of approximately 14 yobibyte while the attacker advantage is still $\lesssim 2^{-28}$. To sum up, we see that the schemes can deal with very high data complexities, as long as writes to single addresses do not get too excessive.

6.3 Generic Attack on 256-Bit Docked Double Decker Using Ascon-PRF

Clearly, generic attacks on the docked double decker are possible. As an example, we show a generic attack on a specific instantiation of docked double decker using Ascon PRF and finite field multiplications in $\mathrm{GF}(2^n)$. Concurrent work [21] provides more insights into this type of attacks on the DDD and HCTR2.

6.3.1 The Attack

The goal of the attack is to find a differential structure as shown in Figure 15. We assume that the tweak T is constant. On the left side, we do not want to have a difference indicated by $P'_1 \oplus P''_1 = 0$ and on the right side we are fine with any difference indicated by $P'_2 \oplus P''_2 = \Delta_1$. Since the input of the finite field multiplication in the first round has no difference, the difference Δ_1 is preserved deterministically. Hence, we have a 0 difference on the left side and Δ_1 on the right side. In the second round, we require a probabilistic transition since the input to the PRF here is Δ_1, but we want to have 0 difference on the output of the PRF. Assuming an ideal PRF, this transition holds with a probability of 2^{-128}. Then, the transition of the third round is deterministic, since we have a 0 difference at the input of the PRF. For the fourth round, we are fine with any transition of Δ_1 to Δ_2.

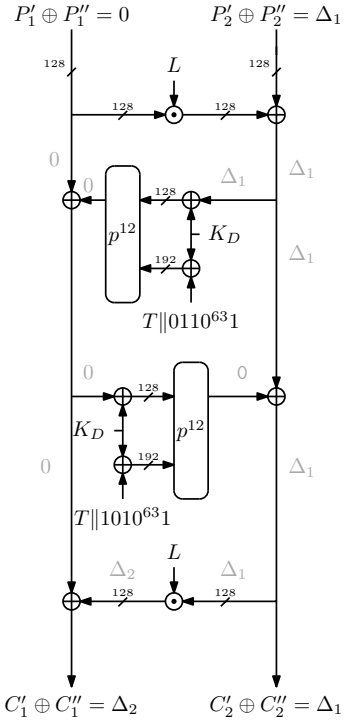

Figure 15: Difference propagation for DDD with Ascon-PRF

Once we have a pair of plaintexts following this structure, we can recover the secret key L. We know $C'_2 \cdot L \oplus C''_2 \cdot L = \Delta_2 = C'_1 \oplus C''_1$ and can solve for L. Once we know L, we also know the inputs I_1 and the outputs O_1 of Ascon-PRF. Since O_1 is 128 bits and the inner part of the 320-bit state of Ascon that is unknown is 192 bits, we can recover it with 2^{192} guesses. Once we know the whole state, we can compute backwards to the secret key K used by Ascon-PRF.

Due to the birthday paradox, we can find collisions in the PRF of the second round of Figure 15 with a data complexity of 2^{64}. One way of doing this is using an ordered list or hash table. This table stores P_1, P_2, C_1, C_2 and is indexed by $P_2 \oplus C_2$. Once we get two entries with the same index we have $P_2' \oplus C_2' = P_2'' \oplus C_2''$ and thus $P_2' \oplus P_2'' = C_2' \oplus C_2'' = \Delta_1$ and hence, a candidate for a collision in the second-round PRF. Note that we do not expect many false positives. We expect that memoryless collision finding is also applicable.

To sum up, the attack works in time roughly 2^{192} and has a data complexity of 2^{64} under a single tweak. We want to note that we only found a working attack when the tweak difference is 0.

6.3.2 Trading Higher Data Complexity for Lower Time Complexity

For the key recovery part of the attack, it is possible to trade a higher data complexity for lower time complexity. Assume we have already recovered L using 2^{64} data. Then, assuming we have more data, we can hope to see multi-collisions at the output of Ascon-PRF. The data complexity for finding a ζ-collision is $\mathcal{M} = \left(\zeta! \cdot 2^{(b-c)(\zeta-1)}\right)^{1/\zeta}$ [53]. For example, the data complexity for a 16-collision is already approximately 2^{122}. Still, one then would need a time complexity of 2^{188} to recover the rest of the truncated state, and hence the key. The trade-off that gives the lowest time complexity is when the data complexity and time complexity are both 2^{160}. However, we assume that it is unrealistic to collect that much data.

6.3.3 An Attack on HCTR2 for Comparison

For HCTR2 [9], the hash key can be recovered with a similar complexity. Let us consider a two-block encryption with HCTR2 having two 128-bit blocks and a matching hash function as shown in Figure 16. Since we assume constant tweaks throughout the attack, we simplified the contribution from the hash function to an unknown Q and extracted the multiplication with the hash key L of the varying part of the hash function. With R, we denote an additional secret key. For the attack, we want to have a difference indicated by $P_1' \oplus P_1'' = 0$ on the left side and on the right side we are fine with any difference indicated by $P_2' \oplus P_2'' = \Delta_1$. This difference then becomes some difference Δ_2 after the first round. Then, we assume that the blockcipher E_K preserves the transition Δ_2 to Δ_2. Hence, the difference to the second block cipher and so the difference at its output is 0. The probability for this to happen is 2^{-128}. Finally, the second finite field multiplication changes Δ_1 to Δ_3 in general. Although the difference at the input of the finite field multiplication is the same as in the first round, the individual values forming the pairs are likely not the same. Then we get $C_1' \oplus C_1'' = \Delta_4$ and $C_2' \oplus C_2'' = \Delta_1$. From that, we know $C_2' \cdot L \oplus C_2'' \cdot L = \Delta_3$, $P_2' \cdot L \oplus P_2'' \cdot L = \Delta_2$, and $\Delta_2 \oplus \Delta_3 = \Delta_4$. So we have $C_2' \cdot L \oplus C_2'' \cdot L \oplus P_2' \cdot L \oplus P_2'' \cdot L = \Delta_4 = C_1' \oplus C_1''$ and can solve for L. Note that the same arguments as for the attack on DDD can be made for the data complexity due to the birthday paradox. Hence, we expect the attack to succeed after collecting 2^{64} data.

Note that this attack also applies when different tweaks are used. For instance, just considering tweak length of one block, we have $Q = \mathsf{bin}\,(2\,|T| + 2)\,L^3 \oplus T \cdot L^2$ and hence $Q' = (2\,|T'| + 2)\,L^3 \oplus T' \cdot L^2$ and $Q'' = (2\,|T''| + 2)\,L^3 \oplus T'' \cdot L^2$. However, since both are added for the top and the bottom and stay the same, the contributions cancel out. In particular, we have $C_2' \cdot L \oplus C_2'' \cdot L \oplus Q' \oplus Q'' = C_1' \oplus C_1'' \oplus \Delta_2$, and $P_2' \cdot L \oplus P_2'' \cdot L \oplus Q' \oplus Q'' = \Delta_2$. Hence, still $C_2' \cdot L \oplus C_2'' \cdot L \oplus P_2' \cdot L \oplus P_2'' \cdot L = C_1' \oplus C_1''$.

7 Implementation

We evaluate fully unrolled, combinational 256-bit datapaths for two Ascon-DDD instantiations: Ascon-DDD-P12, based on Ascon-PRF (Section 5.1.2, Figure 11), and Ascon-DDD-P8,

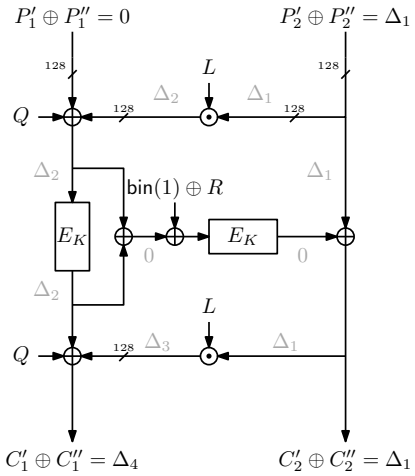

Figure 16: Difference propagation for HCTR2

based on a PRF derived from Ascon-AEAD (Section 5.1.1, Figure 10). We compare them with XTS-AES-128 [20] and HCTR2-AES-128 [9], each processing 256 data bits per operation.

7.1 Evaluation Model

We report encryption (write) and decryption (read) separately, under the following evaluation model for memory encryption, where the tweak is derived from the address:

(a) *Per-key values* (the AES key schedules and HCTR2's hash key L, its square L^2, additional key R, and length-encoding term $\mathsf{bin}(2|T|+2) \cdot L^3$; see Section 6.3) are precomputed once per key. Their logic contributes area but no latency.

(b) *Writes:* the tweak arrives together with the data, so all tweak processing is on the write latency path; for HCTR2, $T \cdot L^2$ is computed in parallel with $P_2 \cdot L$.

(c) *Reads:* the tweak (address) is known before the data returns from memory. Any value that depends only on the keys and the tweak is therefore computed during the memory access; we assume its latency is hidden by the memory access and exclude it from the read latency. The read datapath takes this value from a register; the logic computing it is included in the area, the register is not. We apply this uniformly: XTS precomputes the encrypted tweak T_0 and αT_0, Ascon-DDD-P8 precomputes the state S_1 (Section 5.1.1), and HCTR2 precomputes $Q = \mathsf{bin}(2|T|+2) \cdot L^3 \oplus T \cdot L^2$ (Section 6.3). Ascon-DDD-P12 has no tweak-only computation, so its read path is unchanged.

7.2 Hardware Architecture

Ascon-DDD. Both datapaths use two GF(2^{128}) multipliers, two unrolled permutations (p^{12} for P12, p^8 for P8), and the connecting XOR network; P8 additionally contains an unrolled p^{12} computing S_1. No key expansion is required. The P12 tweak is 125 bits; all other designs use 128-bit tweaks.

AES engines and S-boxes. Each AES-128 engine is a fully unrolled ten-round datapath. We compare two S-box implementations. The "LUT" S-box is a 256-entry table that is synthesized into standard-cell logic. The GF(2^4) S-box is the combined forward/inverse circuit of Maximov and Ekdahl [40], fixed to one direction per engine.

XTS-AES-128. One XTS-AES-128 instance processes a 128-bit block, so to match the 256-bit input of Ascon-DDD we use two instances with shared key expansion and tweak generation. The shared tweak engine encrypts the data-unit number under the tweak key to obtain T_0 and precomputes αT_0 for the second block, where multiplication by the primitive element α of $\mathrm{GF}(2^{128})$ is a shift with conditional reduction XOR.

HCTR2-AES-128. The two hash evaluations are computed in the distributed form $Q \oplus P_2 \cdot L$ and $Q \oplus C_2 \cdot L$ (cf. Figure 16). The datapath uses three $\mathrm{GF}(2^{128})$ multipliers (one for $T \cdot L^2$ and one for each hash evaluation) and two AES engines: two encryption engines on writes, and one decryption and one encryption (XCTR) engine on reads. On writes, $T \cdot L^2$ and $P_2 \cdot L$ are computed in parallel, so both directions have the same critical-path structure: a $\mathrm{GF}(2^{128})$ multiplication, two AES evaluations, and a second $\mathrm{GF}(2^{128})$ multiplication. On reads, however, the first AES evaluation uses a decryption engine. The per-key values L, L^2, R, and $\mathsf{bin}(2|T|+2) \cdot L^3$ are computed once per key by dedicated logic, separate from the datapath: an AES-128 key-expansion unit, an AES-128 encryption engine, a POLYVAL key-conversion unit, and a POLYVAL multiplier, which computes L^2 and $\mathsf{bin}(2|T|+2) \cdot L^3$ sequentially.

$\mathrm{GF}(2^{128})$ multipliers. All $\mathrm{GF}(2^{128})$ multipliers, including those for HCTR2's POLYVAL in its GHASH-isomorphic form [25], use a 128-bit carry-less multiplier followed by reduction modulo $x^{128} + x^7 + x^2 + x + 1$.

7.3 Synthesis Results

We use Synopsys Design Compiler and the ASAP7 [8] `asap7sc7p5t` SLVT standard-cell library for synthesis. All configurations use the same latency-optimizing flow with cross-module optimization, with the design flattened before the final incremental compile. The AES key schedules and the HCTR2 per-key logic contribute area but no latency. We report combinational area; registers are excluded. Table 4 summarizes the synthesis results.

Table 4: Synthesis results for 256-bit datapaths under the evaluation model of Section 7.1.

	Read		Write	
Design	**Area (μm^2)**	**Latency (ps)**	**Area (μm^2)**	**Latency (ps)**
Ascon-DDD-P12	12,102	2,172	12,050	2,180
Ascon-DDD-P8	12,401	1,570	12,485	2,311
XTS-AES-128 ($\mathrm{GF}(2^4)$ S-box)	15,410	2,523	14,451	4,502
XTS-AES-128 (LUT S-box)	25,713	1,856	26,893	2,951
HCTR2-AES-128 ($\mathrm{GF}(2^4)$ S-box)	22,211	5,087	21,420	4,840
HCTR2-AES-128 (LUT S-box)	32,442	3,677	32,128	3,349

Ascon-DDD vs. AES-based references. Compared with the speed-optimized XTS-AES-128 (LUT S-box), Ascon-DDD-P8 has 15% lower read latency, while Ascon-DDD-P12 has 17% higher read latency. Compared with the area-optimized XTS-AES-128 ($\mathrm{GF}(2^4)$ S-box), P12 has 14% lower read latency. Both Ascon-DDD variants are smaller than all AES-based references in both directions; relative to the corresponding XTS configurations, P12 is 1.2–1.3× smaller and P8 is 2.1–2.2× smaller. Both Ascon-DDD variants also have lower latency and area than the corresponding HCTR2 configurations: P12 is 1.8× smaller and has 55–57% lower latency in both directions than HCTR2 ($\mathrm{GF}(2^4)$ S-box), while P8 is 2.6× smaller and has 57% lower read and 31% lower write latency than HCTR2 (LUT S-box). On writes, P12 has 52% lower latency than XTS-AES-128 ($\mathrm{GF}(2^4)$ S-box), and P8 has 22% lower latency than XTS-AES-128 (LUT

S-box). The P12 advantage reflects the time XTS spends encrypting the tweak before processing the data.

Ascon-DDD-P8 vs. Ascon-DDD-P12. On reads, where the S_1 precomputation is hidden, P8 traverses fewer permutation rounds and has 28% lower latency than P12. The additional p^{12} for S_1, however, makes P8 slightly larger, and on writes, where S_1 cannot be precomputed, P8 has the higher latency.

8 Conclusion

In this paper, we have instantiated cryptographic accordions, in particular the docked double decker, with functions derived from Ascon. We compare them with a cryptographic accordion based on HCTR2 and AES-128, but also with the simpler XTS mode using AES-128. In our hardware evaluation, both Ascon-based constructions achieve lower area and latency than the AES-128-based HCTR2 implementations we tested. Furthermore, the proposed constructions can even outperform simpler modes like XTS-AES-128, in all but one case, where Ascon-DDD-P12 has 17% higher read latency than XTS-AES-128 with the LUT S-box. However, this implementation uses only 47% of the area of that XTS-AES-128 configuration.

From a security perspective, we show that the proposed constructions are able to deal with much higher data complexities compared to the AES-based alternatives. To sum up, our proposed constructions provide a viable alternative to AES-based constructions for cryptographic accordions. In particular, they are even suitable for very demanding environments like memory encryption. Hence, we believe that they can have a positive impact on real-world security.

Bibliography

[1] Roberto Avanzi, Orr Dunkelman, and Kazuhiko Minematsu. *MATTER: A Wide-Block Tweakable Block Cipher*. Cryptology ePrint Archive, Paper 2024/1186. 2024. URL: https://eprint.iacr.org/2024/1186.

[2] Jules Baudrin, Anne Canteaut, and Léo Perrin. "Practical Cube Attack against Nonce-Misused Ascon". In: *IACR Transactions on Symmetric Cryptology* 2022.4 (2022), pp. 120–144. DOI: 10.46586/TOSC.V2022.I4.120-144. URL: https://doi.org/10.46586/tosc.v2022.i4.120-144.

[3] CAESAR. *Final Portfolio*. Available at https://competitions.cr.yp.to/caesar-submissions.html. 2019.

[4] Debrup Chakraborty and Palash Sarkar. "A New Mode of Encryption Providing a Tweakable Strong Pseudo-random Permutation". In: *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*. Ed. by Matthew J. B. Robshaw. Vol. 4047. Lecture Notes in Computer Science. Springer, 2006, pp. 293–309. DOI: 10.1007/11799313_19. URL: https://doi.org/10.1007/11799313_19.

[5] Debrup Chakraborty and Palash Sarkar. "HCH: A New Tweakable Enciphering Scheme Using the Hash-Encrypt-Hash Approach". In: *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*. Ed. by Rana Barua and Tanja Lange. Vol. 4329. Lecture Notes in Computer Science. Springer, 2006, pp. 287–302. DOI: 10.1007/11941378_21. URL: https://doi.org/10.1007/11941378_21.

[6] Donghoon Chang, Deukjo Hong, Jinkeon Kang, and Meltem Sönmez Turan. "Resistance of Ascon Family Against Conditional Cube Attacks in Nonce-Misuse Setting". In: *IEEE Access* 11 (2023), pp. 4501–4516. DOI: 10.1109/ACCESS.2022.3223991. URL: https://doi.org/10.1109/ACCESS.2022.3223991.

[7] Jalen Chuang, Alex Seto, Nicolas Berrios, Stephan van Schaik, Christina Garman, and Daniel Genkin. "TEE.fail: Breaking Trusted Execution Environments via DDR5 Memory Bus Interposition". In: *47th IEEE Symposium on Security and Privacy (IEEE S&P '26)*. IEEE Computer Society, 2026. URL: https://tee.fail.

[8] Lawrence T. Clark, Vinay Vashishtha, Lucian Shifren, et al. "ASAP7: A 7-nm FinFET Predictive Process Design Kit". In: *Microelectronics Journal* 53 (2016), pp. 105–115. DOI: 10.1016/j.mejo.2016.04.006.

[9] Paul Crowley, Nathan Huckleberry, and Eric Biggers. *Length-preserving encryption with HCTR2*. Cryptology ePrint Archive, Paper 2021/1441. 2021. URL: https://eprint.iacr.org/2021/1441.

[10] Joan Daemen and Vincent Rijmen. *The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition*. Information Security and Cryptography. Springer, 2020. ISBN: 978-3-662-60768-8. DOI: 10.1007/978-3-662-60769-5. URL: https://doi.org/10.1007/978-3-662-60769-5.

[11] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. "Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting". In: *Advances in Cryptology - EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. LNCS. Springer, 2013, pp. 371–387. DOI: 10.1007/978-3-642-38348-9_23. URL: https://doi.org/10.1007/978-3-642-38348-9_23.

[12] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *Ascon v1: Submission to the CAESAR Competition*. 2014.

[13] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Cryptanalysis of Ascon". In: *Topics in Cryptology - CT-RSA 2015*. Ed. by Kaisa Nyberg. LNCS. Springer, 2015, pp. 371–387. DOI: 10.1007/978-3-319-16715-2_20. URL: https://doi.org/10.1007/978-3-319-16715-2_20.

[14] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *Ascon v1.2: Submission to the CAESAR Competition*. 2016.

[15] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *Ascon v1.2: Submission to NIST*. 2019.

[16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *Ascon PRF, MAC, and Short-Input MAC*. Cryptology ePrint Archive, Paper 2021/1574. 2021. DOI: 10.1007/978-3-031-58868-6_15. URL: https://eprint.iacr.org/2021/1574.

[17] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon v1.2: Lightweight Authenticated Encryption and Hashing". In: *Journal of Cryptology* 34.3 (2021), p. 33. DOI: 10.1007/S00145-021-09398-9. URL: https://doi.org/10.1007/s00145-021-09398-9.

[18] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon MAC, PRF, and Short-Input PRF - Lightweight, Fast, and Efficient Pseudorandom Functions". In: *Topics in Cryptology - CT-RSA 2024*. Ed. by Elisabeth Oswald. LNCS. Springer, 2024, pp. 381–403. DOI: 10.1007/978-3-031-58868-6_15. URL: https://doi.org/10.1007/978-3-031-58868-6_15.

[19] Christoph Dobraunig, Krystian Matusiewicz, Bart Mennink, and Alexander Tereschenko. "Efficient Instances of Docked Double Decker with AES, and Application to Authenticated Encryption". In: *Advances in Cryptology - EUROCRYPT 2025*. Ed. by Serge Fehr and Pierre-Alain Fouque. LNCS. Springer, 2025, pp. 62–92. DOI: 10.1007/978-3-031-91107-1_3. URL: https://doi.org/10.1007/978-3-031-91107-1_3.

[20] Morris Dworkin. *NIST SP 800-38E: Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices*. Tech. rep. NIST, 2010.

[21] Jonathan Fuchs. *Structural Tightness of Quadratic Multi-Query Bounds for Universal Hashing: Applications to Accordion Modes*. Cryptology ePrint Archive, Paper 2026/314. 2026. URL: https://eprint.iacr.org/2026/314.

[22] David Gérault, Thomas Peyrin, and Quan Quan Tan. "Exploring Differential-Based Distinguishers and Forgeries for ASCON". In: *IACR Transactions on Symmetric Cryptology* 2021.3 (2021), pp. 102–136. DOI: 10.46586/TOSC.V2021.I3.102-136. URL: https://doi.org/10.46586/tosc.v2021.i3.102-136.

[23] Hannes Gross, Rinat Iusupov, and Roderick Bloem. "Generic Low-Latency Masking in Hardware". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018.2 (May 2018), pp. 1–21. DOI: 10.13154/tches.v2018.i2.1-21. URL: https://tches.iacr.org/index.php/TCHES/article/view/871.

[24] Shay Gueron, Adam Langley, and Yehuda Lindell. *AES-GCM-SIV: Specification and Analysis*. Cryptology ePrint Archive, Paper 2017/168. 2017. URL: https://eprint.iacr.org/2017/168.

[25] Shay Gueron, Adam Langley, and Yehuda Lindell. *AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption*. RFC 8452. IETF, 2019.

[26] Aldo Gunsing, Joan Daemen, and Bart Mennink. "Deck-Based Wide Block Cipher Modes and an Exposition of the Blinded Keyed Hashing Model". In: *IACR Transactions on Symmetric Cryptology* 2019.4 (2019), pp. 1–22. DOI: 10.13154/TOSC.V2019.I4.1-22. URL: https://doi.org/10.13154/tosc.v2019.i4.1-22.

[27] Shai Halevi. "EME*: Extending EME to Handle Arbitrary-Length Messages with Associated Data". In: *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*. Ed. by Anne Canteaut and Kapalee Viswanathan. Vol. 3348. Lecture Notes in Computer Science. Springer, 2004, pp. 315–327. DOI: 10.1007/978-3-540-30556-9_25. URL: https://doi.org/10.1007/978-3-540-30556-9_25.

[28] Shai Halevi and Phillip Rogaway. "A Tweakable Enciphering Mode". In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 482–499. DOI: 10.1007/978-3-540-45146-4_28. URL: https://doi.org/10.1007/978-3-540-45146-4_28.

[29] Shai Halevi and Phillip Rogaway. "A Parallelizable Enciphering Mode". In: *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*. Ed. by Tatsuaki Okamoto. Vol. 2964. Lecture Notes in Computer Science. Springer, 2004, pp. 292–304. DOI: 10.1007/978-3-540-24660-2_23. URL: https://doi.org/10.1007/978-3-540-24660-2_23.

[30] Kai Hu. "Improved Conditional Cube Attacks on Ascon AEADs in Nonce-Respecting Settings with a Break-Fix Strategy". In: *IACR Transactions on Symmetric Cryptology* 2024.2 (2024), pp. 118–140. DOI: 10.46586/TOSC.V2024.I2.118-140. URL: https://doi.org/10.46586/tosc.v2024.i2.118-140.

[31] Tetsu Iwata and Kaoru Kurosawa. "On the Universal Hash Functions in Luby-Rackoff Cipher". In: *Information Security and Cryptology - ICISC 2002*. Ed. by Pil Joong Lee and Chae Hoon Lim. LNCS. Springer, 2002, pp. 226–236. DOI: 10.1007/3-540-36552-4_16. URL: https://doi.org/10.1007/3-540-36552-4_16.

[32] Charlotte Lefevre and Bart Mennink. "SoK: Security of the Ascon Modes". In: *IACR Transactions on Symmetric Cryptology* 2025.1 (2025), pp. 138–210. DOI: 10.46586/TOSC.V2025.I1.138-210. URL: https://doi.org/10.46586/tosc.v2025.i1.138-210.

[33] Yanbin Li, Guoyan Zhang, Wei Wang, and Meiqin Wang. "Cryptanalysis of round-reduced ASCON". In: *Science China Information Sciences* 60.3 (2017), p. 38102. DOI: 10.1007/S11432-016-0283-3. URL: https://doi.org/10.1007/s11432-016-0283-3.

[34] Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. "Conditional Cube Attack on Round-Reduced ASCON". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 175–202. DOI: 10.13154/TOSC.V2017.I1.175-202. URL: https://doi.org/10.13154/tosc.v2017.i1.175-202.

[35] Moses D. Liskov, Ronald L. Rivest, and David A. Wagner. "Tweakable Block Ciphers". In: *Advances in Cryptology - CRYPTO 2002*. Ed. by Moti Yung. LNCS. Springer, 2002, pp. 31–46. DOI: 10.1007/3-540-45708-9_3. URL: https://doi.org/10.1007/3-540-45708-9_3.

[36] Jiqiang Lu and Wenchang Zhou. "Improved meet-in-the-middle attack on 10 rounds of the AES-256 block cipher". In: *Designs, Codes and Cryptography* 92.4 (2024), pp. 957–973. DOI: 10.1007/S10623-023-01323-4. URL: https://doi.org/10.1007/s10623-023-01323-4.

[37] Michael Luby and Charles Rackoff. "How to Construct Pseudo-Random Permutations from Pseudo-Random Functions (Abstract)". In: *Advances in Cryptology - CRYPTO '85*. Ed. by Hugh C. Williams. LNCS. Springer, 1985, p. 447. DOI: 10.1007/3-540-39799-X_34. URL: https://doi.org/10.1007/3-540-39799-X_34.

[38] Michael Luby and Charles Rackoff. "How to Construct Pseudorandom Permutations from Pseudorandom Functions". In: *SIAM Journal on Computing* 17.2 (1988), pp. 373–386. DOI: 10.1137/0217022. URL: https://doi.org/10.1137/0217022.

[39] Stefan Lucks. "Faster Luby-Rackoff Ciphers". In: *Fast Software Encryption - FSE 1996*. Ed. by Dieter Gollmann. LNCS. Springer, 1996, pp. 189–203. DOI: 10.1007/3-540-60865-6_53. URL: https://doi.org/10.1007/3-540-60865-6_53.

[40] Alexander Maximov and Patrik Ekdahl. "New Circuit Minimization Techniques for Smaller and Faster AES SBoxes". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019.4 (2019), pp. 91–125. DOI: 10.13154/tches.v2019.i4.91-125.

[41] Bart Mennink. "Understanding the Duplex and Its Security". In: *IACR Transactions on Symmetric Cryptology* 2023.2 (2023), pp. 1–46. DOI: 10.46586/TOSC.V2023.I2.1-46. URL: https://doi.org/10.46586/tosc.v2023.i2.1-46.

[42] Dustin Moody, Ray Perlner, Andrew Regenscheid, Angela Robinson, and David Cooper. *NIST IR 8547: Transition to Post-Quantum Cryptography Standards*. Tech. rep. NIST, 2026. URL: https://doi.org/10.6028/NIST.IR.8547.

[43] Moni Naor and Omer Reingold. "On the Construction of Pseudo-Random Permutations: Luby-Rackoff Revisited (Extended Abstract)". In: *ACM Symposium on the Theory of Computing 1997*. Ed. by Frank Thomson Leighton and Peter W. Shor. ACM, 1997, pp. 189–199. DOI: 10.1145/258533.258581. URL: https://doi.org/10.1145/258533.258581.

[44] Moni Naor and Omer Reingold. "On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited". In: 12.1 (1999), pp. 29–66. DOI: 10.1007/PL00003817. URL: https://doi.org/10.1007/PL00003817.

[45] National Institute of Standards and Technology. *Lightweight Cryptography*. Available at https://csrc.nist.gov/projects/lightweight-cryptography. 2017.

[46] National Institute of Standards and Technology. *NIST SP 800-197A: PRE-DRAFT Call for Comments: NIST Launches Development of Cryptographic Accordions*. Tech. rep. Available at https://csrc.nist.gov/pubs/sp/800/197/a/iprd. NIST, 2025.

[47] NIST. *NIST Launches Development of Cryptographic Accordions*. 2025. URL: https://csrc.nist.gov/pubs/sp/800/197/a/iprd.

[48] Jacques Patarin. "How to Construct Pseudorandom and Super Pseudorandom Permutations from one Single Pseudorandom Function". In: *Advances in Cryptology - EUROCRYPT '92*. Ed. by Rainer A. Rueppel. LNCS. Springer, 1992, pp. 256–266. DOI: 10.1007/3-540-47555-9_22. URL: https://doi.org/10.1007/3-540-47555-9_22.

[49] Shuo Peng, Kai Hu, Jiahui He, and Meiqin Wang. "Improved Key Recovery Attacks of Ascon". In: *Topics in Cryptology - CT-RSA 2025*. Ed. by Arpita Patra. LNCS. Springer, 2025, pp. 123–146. DOI: 10.1007/978-3-031-88661-4_6. URL: https://doi.org/10.1007/978-3-031-88661-4_6.

[50] Phillip Rogaway. "Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC". In: *Advances in Cryptology - ASIACRYPT 2004*. Ed. by Pil Joong Lee. LNCS. Springer, 2004, pp. 16–31. DOI: 10.1007/978-3-540-30539-2_2. URL: https://doi.org/10.1007/978-3-540-30539-2_2.

[51] Alex Seto, Oytun Kuday Duran, Samy Amer, et al. "WireTap: Breaking Server SGX via DRAM Bus Interposition". In: *2025 SIGSAC Conference on Computer and Communications Security (CCS '25)*. Association for Computing Machinery, 2025. URL: https://wiretap.fail.

[52] Meltem Sönmez Turan, Kerry McKay, Jinkeon Kang, John Kelsey, and Donghoon Chang. *NIST SP 800-232: Ascon-Based Lightweight Cryptography Standards for Constrained Devices*. Tech. rep. NIST, 2025. URL: https://doi.org/10.6028/NIST.SP.800-232.

[53] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. "Birthday Paradox for Multi-collisions". In: *Information Security and Cryptology - ICISC 2006*. Ed. by Min Surp Rhee and Byoungcheon Lee. LNCS. Springer, 2006, pp. 29–40. DOI: 10.1007/11927587_5. URL: https://doi.org/10.1007/11927587_5.

[54] Peng Wang, Dengguo Feng, and Wenling Wu. "HCTR: A Variable-Input-Length Enciphering Mode". In: *Information Security and Cryptology, First SKLOIS Conference, CISC 2005, Beijing, China, December 15-17, 2005, Proceedings*. Ed. by Dengguo Feng, Dongdai Lin, and Moti Yung. Vol. 3822. Lecture Notes in Computer Science. Springer, 2005, pp. 175–188. DOI: 10.1007/11599548_15. URL: https://doi.org/10.1007/11599548_15.

[55] Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai. "On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses". In: *Advances in Cryptology - CRYPTO '89*. Ed. by Gilles Brassard. LNCS. Springer, 1989, pp. 461–480. DOI: 10.1007/0-387-34805-0_42. URL: https://doi.org/10.1007/0-387-34805-0_42.