École Polytechnique Fédérale de Lausanne

Improving the Security and the Power Consumption of BLE

by Emiljano Gjiriti

Master Thesis

Approved by the Examining Committee:

Prof. Dr. sc. ETH Mathias Payer
Thesis Advisor

Damian Pfammatter
External Expert

Prof.Dr. Daniele Antonioli
Thesis Supervisor

EPFL IC IINFCOM HEXHIVE
BC 160 (Bâtiment BC)
Station 14
CH-1015 Lausanne

June 25, 2021

A journey to the Orion...

To my family, for all their sacrifices and support.

Acknowledgments

I would first like to thank Prof. Daniele Antonioli, my thesis supervisor, for the continuous feedback and the help he has provided me during my thesis. I would also like to thank Prof. Mathias Payer for accepting me in his lab and for giving me the opportunity to work, along with Daniele, in a topic I enjoy. I would also like to thank Damian Vizár for the paper suggestions during my thesis, and in general for helping me become a better researcher and helping me navigate the convoluted world of crypto proofs. I would also like to thank him and my colleagues at CSEM for the help they provided with the experimental measurements. Finally, I would like to thank my family for the support throughout my Master, which has been quite a challenging experience for me, and especially my sister, Erna, who has always been there for me and has helped me overcome a lot of problems with her wise advise and great humour. I couldn't have done this without her support and I'm grateful to have such an amazing sister.

Lausanne, June 25, 2021 Emiljano Gjiriti

Abstract

Bluetooth is one of the most widely used technologies used for communication today with an estimated 5 billion devices using it. Hence, it is of utmost importance to ensure the security of the communication between Bluetooth devices. The current mechanism used for encrypting session information is AES-CCM(Counter with CBC-MAC), which is an authenticated mode of encryption that ensure integrity and confidentiality. In order to ensure the integrity of the messages, it appends a tag to the encrypted messages, which the standard specifies to be 4 bytes. A tag length of such size can however be easily bruteforced with only 2^{32} attempts, which can be quite fast in today's computers. However, just increasing the tag length is not a good solution, because even with the current tag length, most vendors choose to disable encryption in order to save energy, and increasing the tag length would only encourage such behavior, and would thus lead to less safe communications. These situations can be remediated by either modifying CCM to be able to use a larger tag length but without incurring much larger costs for encryption/decryption, or by substituting AES-CCM with more efficient mechanisms such as AES-GCM or one of the latest finalists of the NIST LWC competition. For the latest case, we decided to test Ascon because it is easy to understand and implement, and it was among the highest-performing schemes in the latest Lightweight Cryptography NIST competition. From now on we will refer to AES-CCM and AES-GCM as simply CCM and GCM respectively. This paper is organized as follows:
We start with an introduction to Authenticated Encryption, focusing primarily on the three algorithms that will be benchmarked in this paper: CCM, GCM and Ascon. We will then introduce a simple modification applied to CCM and Ascon to make them more energy efficient while maintaining the same security guarantees. The standard and modified algorithms are tested on a Bluetooth communication setup, so a brief introduction to Bluetooth will be provided, mostly focusing on how the communication between two devices that use such mechanisms is achieved. We will conclude this work by providing benchmarks and comparing the performances of all the previously mentioned methods.

Contents

Acknowledgments **1**

Abstract (English/Français) **2**

1 Background **8**
1.1 Counter with CBC-MAC (CCM) . . . 8
1.2 Galois Counter Mode(GCM) . . . 11
1.2.1 GCM Performance and Security . . . 12
1.2.2 Bluetooth Low Energy . . . 14

2 Ascon **17**
2.1 ASCON . . . 17
2.2 Ascon Security . . . 19
2.3 Security Analysis . . . 21
2.3.1 Confidentiality of the Ascon Scheme . . . 22
2.3.2 Authenticity of the Ascon Scheme . . . 23
2.4 Variable Tag Ascon . . . 25

3 Design **27**
3.1 Variable Tag CCM . . . 27
3.1.1 Design Rationale . . . 28
3.2 Security of vCCM . . . 28

4 Implementation and Evaluation **37**
4.1 Experimental Validation of Energy Efficiency . . . 38

5 Related Work **44**
5.1 Related Work . . . 44
5.2 Future Work . . . 46
5.3 Conclusion . . . 47

Bibliography **48**

Introduction

The standard cryptographic algorithms may often prove to be inefficient in the constrained world of embedded systems. The constraints are multi-dimensional: power constraints, memory constraints, threshold constraints etc. There is an awareness in the security community about the challenges that arise from the ever-growing IoT world that is clearly demonstrated in the increasing number of the NIST Lightweight Cryptography candidates during the last year(51). While these candidates are carefully benchmarked in terms of power and energy consumption, they are not as widely studied as the algorithms that have been existing for much longer(GCM, CCM, OCB etc.). Hence, another possibility, besides the creation of new lightweight crypto schemes, would be the optimization of the existing and the extensively studied algorithms to make them more efficient and as a result, more efficient when used in embedded systems. One of the most widely used authenticated encryption schemes in IoT devices is CCM, which is dominant among other crypto schemes since it is used to encrypt session data in Bluetooth communication. CCM belongs to a larger family of encryption schemes, namely authenticated encryption schemes, which simultaneously ensure the confidentiality and authenticity of the data. Data authenticity is ensured by stretching the ciphertext with a tag of a certain size. Hence, if an adversary wishes to modify the ciphertext, they also have to modify the corresponding tag. However, they cannot do so without knowing the key that was used to generate such tag. The stretch that is appended to the ciphertext is normally in the range of 4 to 16 bytes. In a Desktop environment, the extra bytes would have no visible effect in the performance, but in a simple communication setup between several embedded systems, adding 16 extra bytes for each transmission would have a considerable effect in performance. This effect in performance may be one of the reasons why the tag size in a Bluetooth communication is chosen to only be 4 bytes in size. However, this tag size is relatively small and can be easily bruteforced with a maximum of 2^32 attempts. However, simply increasing the tag size to a hard to bruteforce size(10-16 bytes) is not a practical solution since it would increase the energy consumption, which in many embedded systems is even more important than security. Furthermore, such schemes require a constant variable length for a single key. However, in a simple communication setup, the messages exchanged may not all have the same priority level. For example, sensors in a power plant may exchange some binary data with one-another to indicate whether the radiation level is below a certain threshold. If that is not the case, they may then need to communicate to a central controller to determine how to handle the emergency situation. The data that the sensors exchange with one another has a lower level of priority than the data that they exchange with the central unit, so authenticating such data with a large tag length would be quite inefficient. Furthermore, following the above example,

authenticating a 1-bit data with a 16-byte tag wouldn't be too sensible, especially in a system where the compromise of such data wouldn't have a significant effect. This problem gives rise to the question whether different tag length can be used to encrypt different messages in authenticated encryption schemes. This question is answered by Reyhanitabar, Vaudenay and Vizár in [21], and unfortunately the answer is no. This implies that a different key would need to be used for each tag length, or otherwise the security of the scheme would be violated. On the other hand, storing different keys for each tag length may violate the memory constraints of some embedded systems, since for example in AES, the key size can be 16, 24 or 32 bytes. Figure 1, taken from [22], illustrates the relation between the energy consumption(y-axis), data rate(x-axis) and security level(circle radius). As can be observed from this figure, secure communication protocols with high bandwidth are very inefficient in terms of energy consumption. Meanwhile, energy-efficient protocols such as LoRa offer little security guarantees.

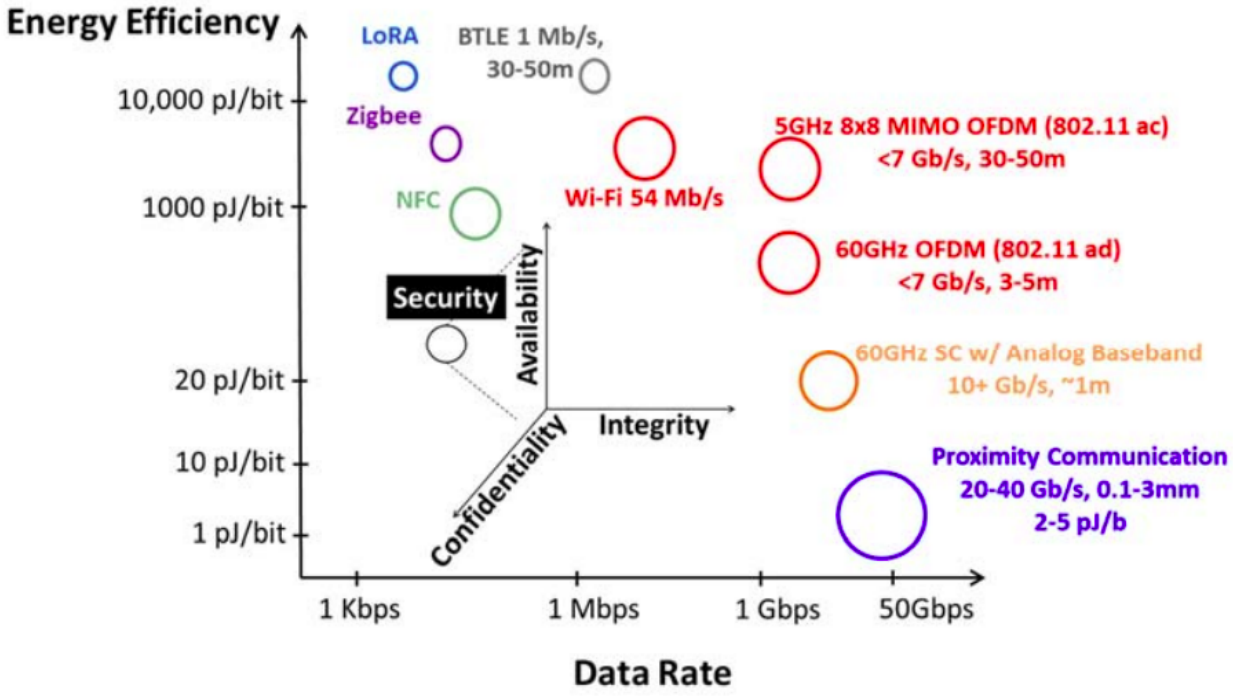
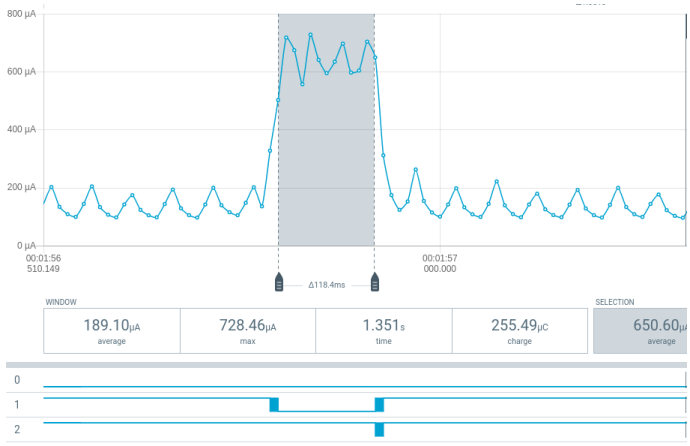
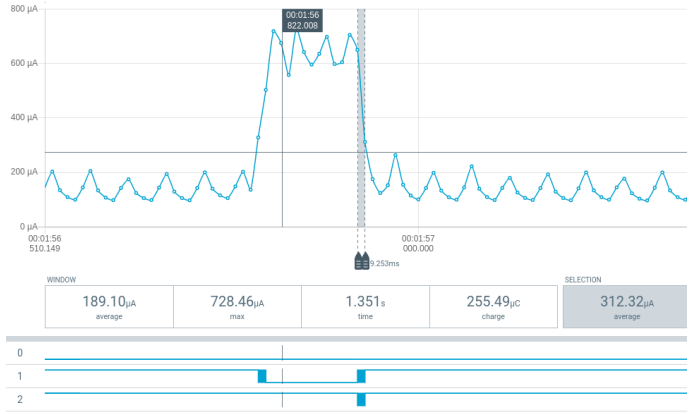


Figure 1 – Relation between the energy efficiency(y-axis), data rate(x-axis) and the security level(circle radius).

Lightweight Cryptography Schemes try to address the above issue by increasing the security of the communication protocols while maintaining a low power consumption. Most of these schemes focus on minimizing and parallelizing the number of operations and thus making the scheme more efficient. However, as illustrated in Figure 2, in a communication setup where two devices frequently exchange encrypted messages, the transmission time is around 10 times the encryption time, so the data exchange will account for most of the energy consumption. Hence, in this scenario, the power consumption could be better optimized by introducing a transformation to the encryption scheme that saves in bandwidth and decreases the transmission time.



(a)



(b)

Figure 2 – Comparison between the transmission time(a) and the encryption time(b)

The Solution to the Problem However, the inefficiencies mentioned in the above paragraph can be overcome by a simple black-box modification to the existing schemes. A similar modification was firstly introduced by Reyhanitabar, Vaudenay and Vizár in [21] to the OCB scheme. However, this scheme is special since it uses a tweakable blockcipher that also requires a tweak as an input besides the key and the plaintext, i.e., $\mathcal{E} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. OCB uses the XEX construction [14] to transform a normal blockcipher to a tweakable blockcipher. Hence, this scheme seems quite specific, so it is not very clear whether the same transformation can be applied to more general schemes that do not use tweakable blockciphers. In [21], it was hinted that the transformation was indeed applicable to a quite large family of cryptographic schemes, namely all the schemes that follow a sponge-like construction, where the tag size is propagated in the calculation of each blockcipher output.

Our Contribution This work provides the following contributions:

- The proof of the existence of a generic construction for variable tag lengths may be quite

challenging and beyond the scope of a Master thesis. A similar and more limited in scope problem that I studied during my thesis, is how the black-box transformation for variable tag authenticated encryption can be implemented in practice, in particular in the CCM and Ascon schemes.

- The CCM scheme was chosen because it is used in Bluetooth communication, and Bluetooth itself is among the most used communication technologies in embedded systems with more than 5 billion devices using it [4]. Furthermore, because of its sponge-like construction, the variable tag transformation in CCM is quite simple and requires no change in the inner workings of the scheme.

- Ascon was chosen because it is one of the most high-performant NIST Lightweight Cryptography finalists in both hardware and software [7, 11]. Furthermore, being a relatively new AE scheme, Ascon's security is not extensively studied. The original paper [8] provides a convoluted security analysis of the primitives underlying the scheme, while an analysis of the scheme as a whole is missing in the literature. This paper seeks to address this issue by providing a simple proof of the security of Ascon.

- We experimentally validate the variable tag transformation by benchmarking its energy benefits in a Bluetooth communication setup between two nRF52840 development boards.

Chapter 1

Background

This chapter gives an introduction to the CCM, GCM and Ascon AE schemes and their security guarantees. As these schemes will be benchmarked in a Bluetooth communication setup, a brief introduction of Bluetooth will also be provided, focusing mostly on how the communication between two parties is achieved.

1.1 Counter with CBC-MAC (CCM)

CTR + CBC-MAC is a nonce-based AE scheme proposed by Whiting, Housley, and Ferguson [6]. It is parameterized by a blockcipher $\mathbf{E} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ with $n = 128$,[1] a tag length $\tau \in \{32, 48, 64, 80, 96, 112, 128\}$, and a nonce length $\nu \in \{56, 64, 72, 80, 88, 96, 104\}$. There is a trade-off between the nonce length ν and the maximal message length $2^{8\cdot(120-\nu)}$. The encryption and the decryption algorithms of $\mathbf{CCM}[\mathbf{E}, \tau, \nu]$ are described in Figure 1.1.

Tag Generation with CBC-MAC. This paragraph will make use of the following notation: We denote by $a \leftarrow\!\$\ S$ the process of uniformly sampling an element from the set S and assigning it to the variable a. We further denote by $|X|$ the length(number of bits) of the variable X and by $X \parallel Y$ the concatenation of the two variable X and Y. Let $\{0,1\}^n$ denote a string of length n and $\{0,1\}^*$ a string of arbitrary length. We also let $\mathsf{left}_l(X)$ denote the process of taking the l leftmost bits of string X for $0 < l < |X|$. The final tag is obtained by truncating a 128-bit tag generated by the plain CBC-MAC algorithm as shown in the upper part of Figure 1.2 to the desired tag length

[1] CCM is only defined for use with 128-bit blockciphers [6]

101: **Algorithm** $\mathcal{E}_K(N, A, M)$
102: $B_0 \leftarrow \mathsf{encB}_0(N, A, M)$
103: $B_1 || \cdots || B_\ell \leftarrow \mathsf{encA}(A) \| \mathsf{encM}(M)$
104: $Y_0 \leftarrow \mathrm{E}_K(B_0)$
105: **for** $i \leftarrow 1$ **to** ℓ **do**
106: $Y_i \leftarrow \mathrm{E}_K(B_i \oplus Y_{i-1})$
107: $T = \mathsf{left}_\tau(Y_\ell)$
108: $M_1 \| M_2 \cdots \| M_m \leftarrow M$ where
109: $|M_m| \leq n$ and $|M_i| = n$ otherwise
110: **for** $i \leftarrow 1$ **to** m **do**
111: $Z_i \leftarrow \mathsf{ctr}(N, i)$
112: $C_i \leftarrow \mathsf{left}_{|M_i|}(\mathrm{E}_K(Z_i)) \oplus M_i$
113: $C \leftarrow C_1 \| C_2 \| \cdots \| C_m$
114: $Z_0 \leftarrow \mathsf{ctr}(N, 0)$
115: $C \leftarrow C \| \mathsf{left}_\tau(\mathrm{E}_K(Z_0)) \oplus T$
116: **return** C

201: **Algorithm** $\mathcal{D}_K(N, A, C)$
202: $C_1 \| C_2 \cdots C_m \| T \leftarrow C$ where
203: $|C_m| \leq n$ and $|C_i| = n$ otherwise
204: and where $|T| = \tau$
205: **for** $i \leftarrow 1$ **to** m **do**
206: $Z_i \leftarrow \mathsf{ctr}(N, i)$
207: $M_i \leftarrow \mathsf{left}_{|C_i|}(\mathrm{E}_K(Z_i)) \oplus C_i$
208: $M = M_1 || M_2 || \cdots || M_m$
209: $B_0 \leftarrow \mathsf{encB}_0(N, A, M)$
210: $B_1 || \cdots || B_\ell \leftarrow \mathsf{encA}(A) \| \mathsf{encM}(M)$
211: $Y_0 \leftarrow \mathrm{E}_K(B_0)$
212: **for** $i \leftarrow 1$ **to** ℓ **do**
213: $Y_i \leftarrow \mathrm{E}_K(B_i \oplus Y_{i-1})$
214: $Z_0 \leftarrow \mathsf{ctr}(N, 0)$
215: $T' = \mathsf{left}_\tau(Y_\ell \oplus \mathrm{E}_K(Z_0))$
216: **if** $T = T'$ **then**
217: **return** M
218: **return** $\perp$

Figure 1.1 – **CCM**$[E, \tau, \nu]$ **mode for AEAD**, with $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a blockcipher. The functions $\mathsf{encB}_0()$, $\mathsf{encA}()$,$\mathsf{encM}()$ and $\mathsf{ctr}()$ are defined in Section 1.1.

Figure 1.2 – Illustration of the inner workings of CCM. The top half depicts the computation of the tag using CBC-MAC and the bottom half depicts the encryption process using the CTR mode. enc(A) is a prefix free encoding of A, while r is an integer such that $|enc(M)| + r$ is divisible by n

τ. The input to CBC-MAC is constructed as $B = \mathsf{encB}_0(N, A, M) \| \mathsf{encA}(A) \| \mathsf{encM}(M)$, with

$$\mathsf{encB}_0(N, A, M) = \mathsf{flags}(A) \| N \| \langle |M| \rangle_{120-\nu} \text{ (outputs a single block)},$$
$$\mathsf{flags}(A) = 0 \| 1_{A=0} \| \langle \tau/16 - 1 \rangle_3 \| \langle 14 - \nu/8 \rangle_3,$$

where $\mathsf{encA} : \mathsf{B}^* \to (\{0,1\}^n)^*$ a prefix-free encoding function that extends the length to a size divisible by n and $\mathsf{encM}(M) = M|0^r$ for the integer $0 \leq r < n$ that makes $|\mathsf{encM}(M)|$ divisible by n. In [12], Jonsson shows that the encoding function $\mathsf{encB}_0(N, A, M) \| \mathsf{encA}(A) \| \mathsf{encM}(M)$ is prefix free, which avoids some trivial forgery attacks. If $l(A) \| A = l(A') \| A'$ for $A \neq A'$, an attacker could just replace A with A' in the corresponding query and trivially perform a forgery. However, in CCM this attack is avoided by construction. When the CBC is finished, the tag is truncated to the desired length and XOR-ed with a permutation generated from the nonce N concatenated with a single bit 0.

CTR Mode Encryption. The ciphertext blocks are computed with CTR mode (lower part of Figure 1.2). The i^{th} input counter block, used to generate the i^{th} key stream block, is computed as $\mathsf{ctr}(N, i) = \mathsf{flags} \| N \| \langle i \rangle_{120-\nu}$, where the flags field is used to encode the message length and is given by $\mathsf{flags} = 0^5 \| \langle 14 - \nu/8 \rangle_3$. Theorem 1 will make use of the following adversarial resources and blockcipher security notions:

Adversarial Resources Denote by q_e^τ and q_d^τ the number of encryption and decryption queries made with a given tag length τ respectively, for $\tau \in \mathcal{I}$. Hence, the total number of queries is $q = \sum_{\tau \in \mathcal{I}} q_e^\tau + q_d^\tau$. Furthermore we denote by σ_i the total number of resources during the i-th query, i.e, $\sigma_i = m_i + a_i$ where m_i, and a_i are the message and authentication data length during the i-th query. Hence, the total number of resources σ, is equal to $\sum_{i=1}^{q}(a_i + m_i)$.

Blockcipher's Security Notions A blockcipher is a function with the following signature:

$$E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$$

i,e it takes as parameter a key $K \in \mathcal{K}$, and maps a bitstring of length n to a permutation of $\{0,1\}^n$. A blockcipher is considered secure if an adversary is unable to distinguish it from a random permutation, i.e.

$$\mathbf{Adv}\,(\mathscr{A}) = \Pr[K \leftarrow\$\, \mathcal{K} : \mathscr{A}^{E_K} \implies 1] - \Pr[\pi \leftarrow\$\, Perm(n) : \mathscr{A}^{\pi} \implies 1]$$

where $Perm(n)$ is the set of all permutations from n-bit strings to n-bit strings. The same notion can also be defined with respect to a random function f instead of a random permutation π. The difference between the two is that the random function is surjective while a random permutation may not be.

The security of the standard CCM scheme was proven by Jonsson in [12] in the following theorem:

Theorem 1 ([12]). *For a blockcipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and any valid values of τ and ν we have*

$$\mathbf{Adv}^{auth}_{CCM[E,\tau,\nu]}(t, q_e, q_d, \sigma_e, \sigma_d) \leq \mathbf{Adv}^{prf}_{\mathbf{E}}(t', \sigma') + \frac{q_d}{2^\tau} + \frac{(2\sigma + 3q)^2}{2^{n+1}}$$

$$\mathbf{Adv}^{priv}_{CCM[E,\tau,\nu]}(t, q_e, \sigma_e) \leq \mathbf{Adv}^{prf}_{\mathbf{E}}(t', \sigma'') + \frac{(2\sigma_e + 3q_e)^2}{2^{n+1}}$$

where $\sigma' \leq 2\sigma + 3q$, $\sigma'' \leq 2\sigma_e + 3q_e$ and $t' \leq t + \gamma \cdot \sigma$ for some "small" constant γ.

Although the above bound seems larger than Jonson's bound, the difference only lies in the way the resources are counted. While Jonson includes the overhead from the CCM computation in the resources, we consider the resources from a user's perspective, who doesn't need to have any knowledge about the internal workings of the algorithm. Hence, if we denote by σ the number of total primitive calls, q the total number of queries, m_i the length of a message block, and a_i the length of an associated block, then the total number of blockcipher call made during the i^{th} query is $m_i + 1$, where the extra one corresponds to the blockcipher call that is used to generate the tag mask, and the total number of calls made during the generation of tag is $a_i + m_i + 2$. We add 2 to $a_i + m_i$ because of the first blockcipher call(which is used to process the block B_0) and the other one may arise because of the length encoding. Hence, the total number of calls to the blockcipher is equal to $\sum_{i=1}^{q}(a_i + m_i + 3) = 2\sigma + 3q$, where we used the fact that $a_i + m_i = \sigma$. Hence, Theorem 1 yields the following **nae** security bound.

Corollary 1.1 ([1]). *For a blockcipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and any valid values of τ and ν*

we have

$$\mathbf{Adv}^{\boldsymbol{nae}}_{\boldsymbol{CCM}[E,\tau,\nu]}(t, q_e, q_d, \sigma_e, \sigma_d) \leq \mathbf{Adv}^{\boldsymbol{prf}}_{\mathbf{E}}(t', \sigma') + \frac{q_d}{2^\tau} + \frac{(2\sigma + 3q)^2}{2^{n+1}} + \frac{(2\sigma_e + 3q_e)^2}{2^{n+1}}$$

where $\sigma' \leq 2\sigma + 3q$ *and* $t' \leq t + \gamma \cdot \sigma$ *for some "small" constant* γ.

1.2 Galois Counter Mode(GCM)

Galois/Counter Mode(GCM) is an authenticated mode of operation for symmetric block ciphers which provides both confidentiality and data integrity. It is proven to be secure and quite efficient, especially in hardware, since its implementation can be pipelined. Because of its efficient implementation, GCM is the state-of-the-art operation mode if the main goal of the communication is to achieve maximal throughput rate. It combines the widely used counter mode with the Galois mode of authentication. The block number is combined with an initialization vector(IV), encrypted using the AES block cipher and XORed with the corresponding plaintext block. Hence, the ciphertext blocks are generated exactly in the same way as in a standard counter mode scheme. This mode of operation is differentiated from the other authenticated encryption schemes that use the CTR mode by the way it generates the tag. The ciphertext blocks that were generated in the previous step are considered as coefficients of a polynomial in the Galois field $GF(2^{128})$, i.e., each element of this field can be considered as a polynomial with coefficients in $\mathbb{Z}_2$ of degree at most 127. To generate the tag, this polynomial is evaluated at the point $H = \mathcal{E}_K(0^{128})$, where 0^{128} is simply a bitstring of all zeroes of length 128 bits. Hence, the final MAC is computed as:

$$HMAC(H, A, C) = X_{m+n+1}$$

where m is the number of 128-bit blocks in A and n i the number of 128-bit blocks in C. This function is evaluated iteratively with each X_i given by

$$X_i = \sum_{j=1}^{i} S_j \cdot H^{i-j+1} = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus S_i) \cdot H & \text{for } i = 1, \ldots, m+n+1 \end{cases}$$

where

$$S_i = \begin{cases} A_i & \text{for } i = 1, \ldots, m-1 \\ A_m^* \parallel 0^{128-v} & \text{for } i = 1 \\ C_{i-m} & \text{for } i = m+1, \ldots, m+n-1 \\ C_n^* \parallel 0^{128-u} & \text{for } i = m+n \\ len(A) \parallel len(C) & \text{for } i = m+n+1 \end{cases}$$

We have to remark here that in the formula for X_i the $\cdot$ operation is not the usual integer multiplication. In $GF(2^{128})$, multiplication is performed by considering each element as a polynomial, multiplying the polynomials in $\mathbb{Z}_2$ and then reducing the final polynomial modulo $x^128+x^7+x^2+x+1$ (or any other irreducible polynomial in $\mathbb{Z}_{128}$. We will provide a simple example to make the above process clearer

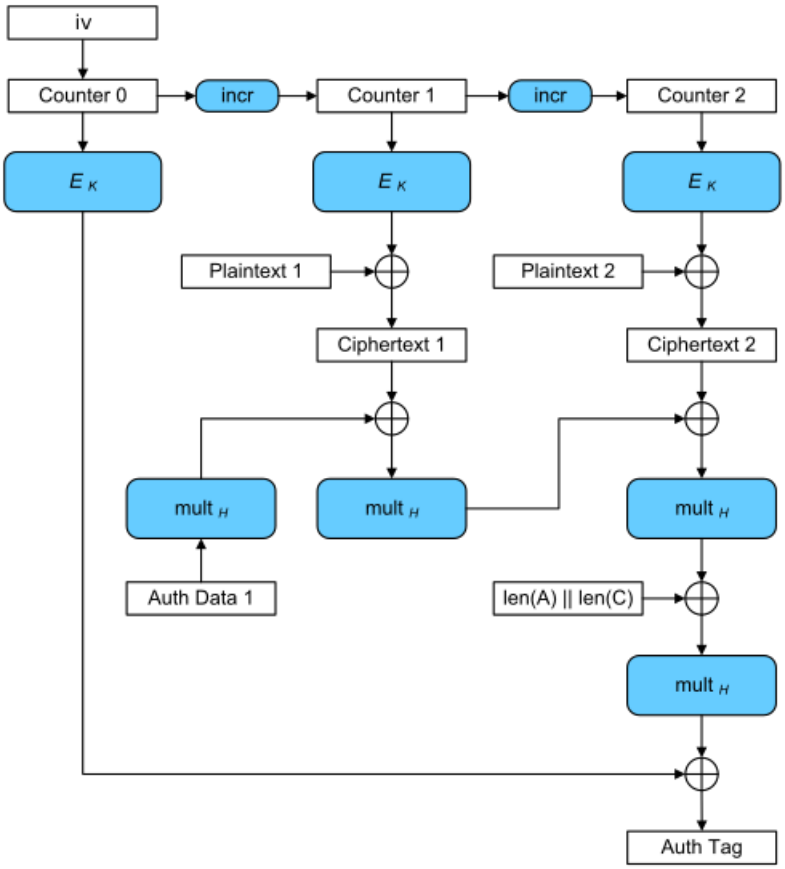


Figure 1.3 – Illustration of GCM AE Scheme

Assume we would like to multiply $a = 0x7$ and $b = 0x7F$. We know that $0x7_{16} = 111_2$ and $0x7F = 1000000_2$. Hence, we can express these numbers as the following polynomials $0x7 \equiv x^2+x+1$ and $0x7F \equiv x^{127}$. After multiplying the polynomials, we obtain

$$x^{127} \cdot (x^2 + x + 1) = x^{129} + x^{128} + x^{127} = x(x^7 + x^2 + x + 2) + x^7 + x^2 + x + 1 + x^{127}$$

Since the polynomials reside in $\mathbb{Z}_2$, we know that $2x^n \equiv 0 \ \forall 0 \leq n \leq 127$. Hence, the above polynomial is equivalent to

$$x^{127} + x^8 + x^3 + x^2 + 1 + x^7 + x^2 + x + 1 + x^{127} \equiv x^8 + x^7 + x^3 + x$$

We can then convert the resulting polynomial to its hexadecimal equivalent

$$x^8 + x^7 + x^3 + x \equiv (110001010)_2 = 0x18A$$

An illustration of the Galois Counter mode of operation is provided in Figure 1.3.

1.2.1 GCM Performance and Security

Performance We decided to also benchmark the GCM performance in this simple communication setup because in [16] it was proven to be much faster than other cryptographic algorithms(OCB

and CWC, especially for small packets, which is the case in our experiments. The criteria used to compare them was **Internet Performance Index**, which evaluates the expected number of bits processed per clock cycle for different packet distributions, where the packet distribution is computed via the following formula [16]:

$$f(s) = \frac{\Pr[S = s]}{\sum_r r \Pr[S = r]}$$

Security GCM is one of the most widely used authenticated encryption schemes because of the high efficiency that results from the parallel encryption of the message blocks. Because of its prevalence, this scheme has been studied in details and proven to be secure. One of the most accessible proofs of the GCM security was given in [18] where the following bound was established for the confidentiality of GCM:

$$\mathbf{Adv}^{\mathbf{vpriv}}_{GCM[Perm(n),\tau]}(\mathscr{A}) \leq \frac{0.5(\sigma + q + 1)^2}{2^n} + \frac{(\sigma + q)(l_N + 1)}{2^N} + \frac{2^{32} q^2 (l_N + 1)}{2^N}$$

and the following bound was established for the authenticity, i.e., resistance to forgery attacks, of GCM:

$$\mathbf{Adv}^{\mathbf{vauth}}_{GCM[Perm(n),\tau]}(\mathscr{A}) \leq \frac{0.5(\sigma + q + q' + 1)^2}{2^n} + \frac{32(q + q')(\sigma + q + 1)(l_N + 1)}{2^n} + \frac{q'(l_A + 1)}{2^\tau}$$

where σ is the total plaintext length, q is the total number of encryption queries, q' is the total number of decryption queries, l_A is the maximum input length, and l_N is the maximum nonce length. Note, that the standard GCM uses the AES as a blockcipher, while the above bounds assume a GCM scheme that uses a random permutation as a blockcipher. Hence, we should also account for the adversary's advantage against the AES blockcipher as follows:

$$\begin{aligned}\mathbf{Adv}^{\mathbf{vpriv}}_{GCM[AES-n,\tau]}(\mathscr{A}) \leq & \frac{0.5(\sigma + q + 1)^2}{2^n} + \frac{(\sigma + q)(l_N + 1)}{2^N} \\ & + \frac{2^{32} q^2 (l_N + 1)}{2^N} + \mathbf{Adv}^{Perm(n)}_{AES-n}(\mathscr{A})\end{aligned}$$

and

$$\begin{aligned}\mathbf{Adv}^{\mathbf{vauth}}_{GCM[Perm(n),\tau]}(\mathscr{A}) \leq & \frac{0.5(\sigma + q + q' + 1)^2}{2^n} + \frac{32(q + q')(\sigma + q + 1)(l_N + 1)}{2^n} \\ & + \frac{q'(l_A + 1)}{2^\tau} + \mathbf{Adv}^{Perm(n)}_{AES-n}(\mathscr{A})\end{aligned}$$

The term $\mathbf{Adv}^{Perm(n)}_{AES-n}$ is not given a specific value since it is an unavoidable cost of using the blockcipher, so it will not affect the security analysis. Furthermore, there are no known attacks against $AES - 128$ that are more efficient than a simple bruteforce attack of exhausting all the 2^{128} key possibilities. Performing such attack would be very inefficient even in today's supercomputers.

1.2.2 Bluetooth Low Energy

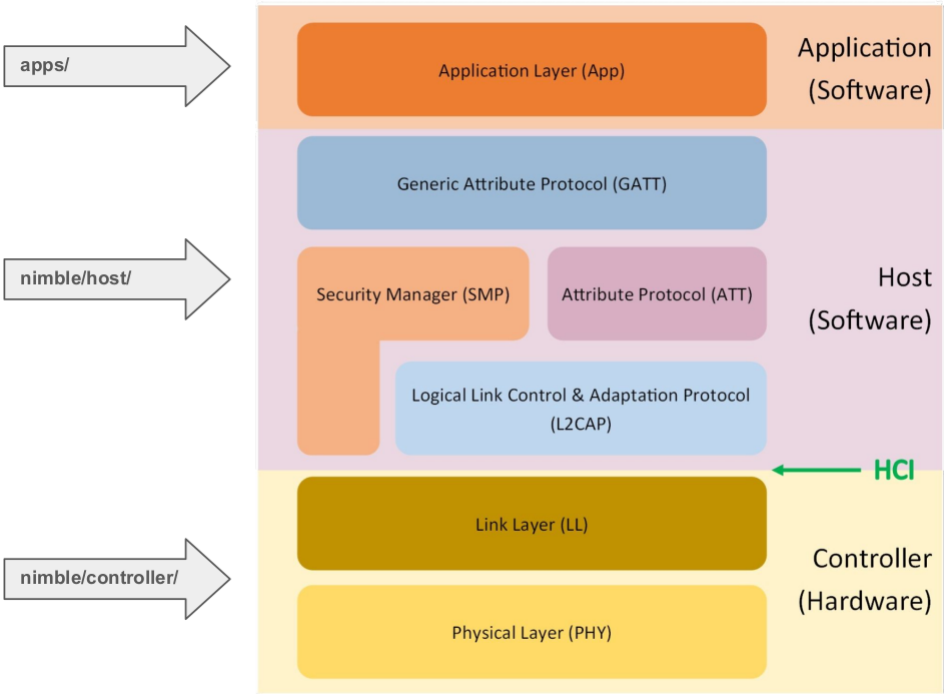


Figure 1.4 – Overview of BLE

Bluetooth Low Energy (BLE) started as part of the Bluetooth 4.0 Core Specification. It's an optimized version of the classic Bluetooth, specifically designed for embedded systems applications, where it has become the main communication standard with several billion devices currently using it.

Bluetooth Devices : Depending on its role, a BLE device may be:

- **Broadcaster**: The device just send advertising packets without expecting any responses or connection requests from the other devices.
- **Observer**: A device that listens to advertising packets but does not establish connections.
- **Central**: A device that listens to the advertising packets and decides whether to establish a connection with the device that sent an advertising packet.
- **Peripheral**: A device that sends advertising packets and accepts a connection request from the central.

Before Connection Establishment Before establishing a connection with another device, a BLE device may either be advertising if it is a peripheral device, or it may be scanning for advertising

packets in case it is a central device. The advertising packets are sent at specific intervals, known as advertising intervals, which by default is set to 687.5 ms, in 40 channels whose centers are 2 MHz apart. 3 of these channels are only used for advertising packets and are known as primary advertising channels, while the other channels, known as secondary advertising channels, are used for data transfer after the connection is established, or in case of a failure of the primary advertising channels. Meanwhile, the central device listens on these channels for advertising packets. A central discovers a peripheral if it is scanning the same channel that the peripheral is advertising on.

After Connection Establishment After establishing a connection, the central and the peripheral need to agree on [3]: the connection interval - the interval at which the two devices transfer data, slave latency - the number of connection events a peripheral can skip without breaking the connection, supervision timeout - if the time between received packets is greater than the supervision amount, a connection is considered lost by the peripheral. After establishing a connection, the two devices need to exchange data, and Generic Attribute Profile(GATT) is an essential part of this process. The Generic Attribute Profile(GATT) establishes how the communication is performed over a BLE connection. GATT provides a reference framework for GATT-based profiles, and each profile is then adapted to the specific use case. However, all profiles must comply with the reference framework provided by GATT in order to function properly. An illustration of the GATT-Based profile hierarchy taken from the core specification is provided in Figure 1.5: From the above illus-

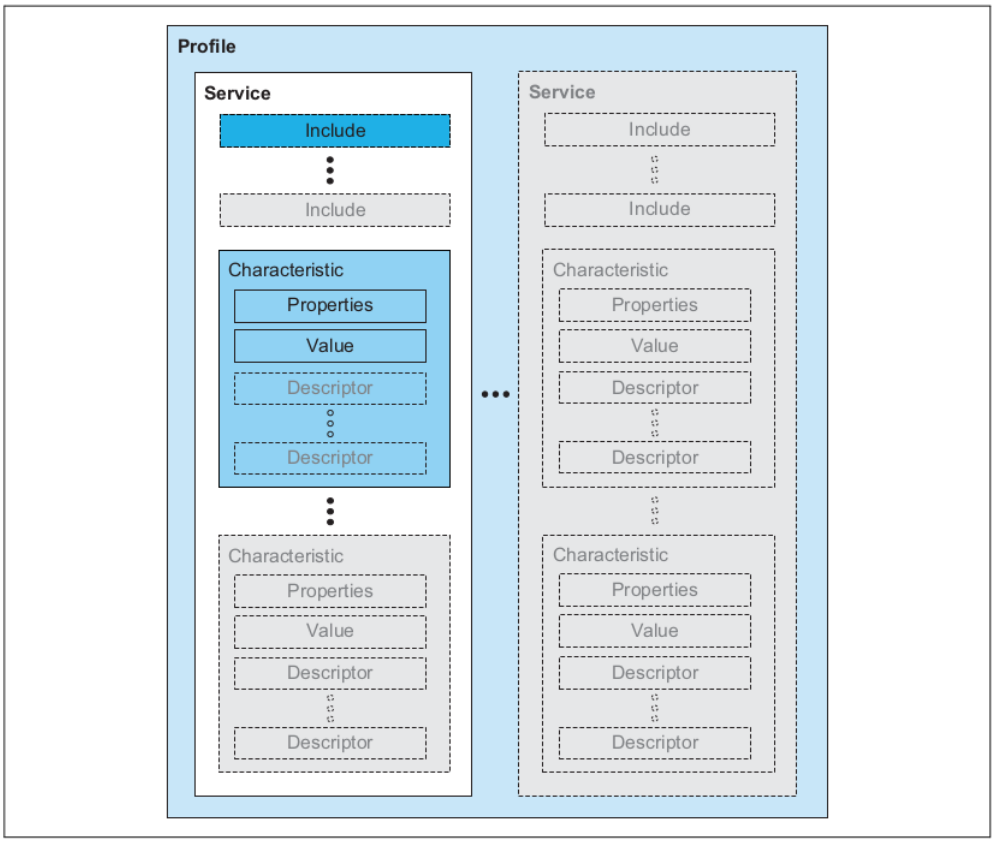


Figure 1.5 – GATT-Based Profile

tration, we observe that a profile contains a service, which consists of characteristics or references to other services. Each characteristic on the other hand contains some value and option descriptors that contain some information about that value. The components encapsulated inside the service will contain the data that will be transmitted, and they are stored as Attributes in the GATT server, where an attribute is the smallest data entity defined by GATT. The devices communicating with one another can be classified into a *server* and a *client*, where attributes are always assumed to be located in the server. The client and server are identified by a MAC address, which is a 6-byte number that uniquely identifies a device among its peers. The standard defines several address types, which are: public, random, static and private device addresses. When the connection is first established, the GATT *client* is not aware of the attributes located in the *server*, so it first needs to issue a service discovery request before being able to read from or write to a characteristic. After the service discovery stage is complete, the client can access a service or a characteristic by using their unique UUID. The UUID is a 128-bit number used in many communication protocols besides Bluetooth, but in the case of Bluetooth they are reduced to 16 or 32 bits since the total payload size supported by the Link Layer is 27 bytes (216 bits). As illustrated in Figure 1.4, the Link Layer is a low-level layer of the Bluetooth protocol that directly interacts with the Physical Layer to establish a link between the devices that are communicating, managing this link and sending data through it.

Furthermore, the server can also determine the level of access of an attribute by setting its permissions' property to: none if the attribute cannot be read or written by a client; readable if the attribute can be read by a client; writable if the attribute can be written by a client; readable and writable if the attribute can both be read and written by a client. The server can also require a certain level of encryption for the connection or a certain level of authentication for the keys.

Chapter 2

Ascon

This chapter seeks to provide an introduction to the Ascon scheme, which is a Lightweight Authenticated Encryption scheme. Furthermore, being a relatively new scheme which was only submitted to NIST in 2019, the security of the scheme is not studied extensively with the only proof provided by the original paper [8] and focusing mostly on the security of the underlying primitives rather than on the security of the scheme as a whole. We seek to address this issue by providing a simple proof of the security of the Ascon scheme and the corresponding variable tag transformation.

2.1 ASCON

ASCON is an authenticated encryption scheme that follows a sponge-like construction and is based on duplex modes. A sponge construction in cryptography is a class of algorithms with a finite state that takes inputs of arbitrary lengths and produces outputs of arbitrary lengths. Algorithms that belong to this class work in two phases:

Phase One : This is called the absorbing stage, where the r-bit input message blocks xor-ed with r bits of the state and interleaved by applying a permutation ρ.

Phase Two : This is called the squeezing stage, where the first r-bits of the state are extracted, interleaved by applying the same permutation p and outputted. The standard ASCON scheme fixes the tag length to 16 bytes, which can be quite inefficient energy-wise when applied to IoT devices. Hence, we benchmark the energy benefits of changing the tag size while using the same key for messages of different priority levels. One of the benefits of choosing a sponge-like lightweight encryption scheme is that the security of the variable-tag transformation can be proven by referring to the work of Reyhanitbar et al in [8], and by slightly modifying their general proof to our specific

case. Ascon is a sponge-based authenticated encryption mode that maintains a state of 320 bits,

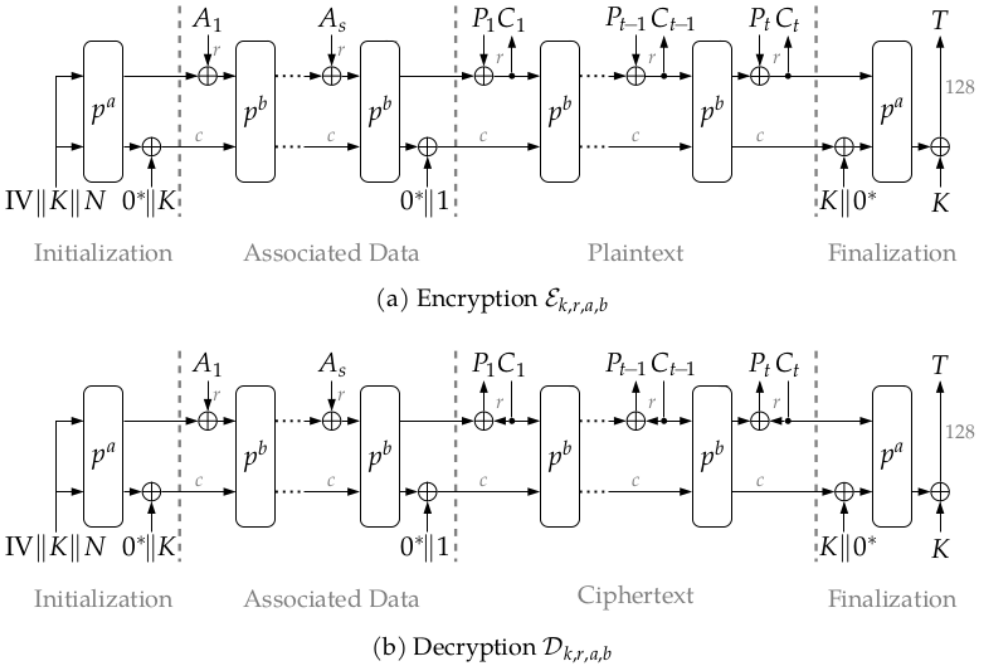


Figure 2.1 – Illustration of the Ascon$[\rho, k, r, a, b]$ scheme with key size k, bitrate r and permutations of a and b rounds.

which is initialized by concatenating an initial value IV with the key K and the nonce N and permuting the resulting state by a permutation function $p^a : \{0,1\}^{320} \to \{0,1\}^{320}$ where a denotes the number of the rounds of the permutation and is provided as a parameter of the scheme. The inner state is concatenated with the associated data or the plaintext/ciphertext blocks in case of an encryption/decryption and uses the resulting state is permuted again through a b-round permutation $p^b : \{0,1\}^{320} \to \{0,1\}^{320}$. The state is separated into an inner part, namely the capacity, denoted by S_c such that $|S_c| = c$, and an outer part, namely the rate, denoted by S_r such that $|S_r| = r$ with $S = S_r \parallel S_c$. The processing of the associated data blocks and the message blocks is streamlined, and the stages are separated by domain separation bits. The choice of Ascon among the other latest NIST Lightweight Cryptography is justified by the following reasons, which are listed in the original paper [8]:

- It is based on the popular and widely studied sponge construction.
- The implementation is quite easy to understand and to modify.
- It performs very well compared to most of the other LWC schemes [20].
- Because of its construction, the cipher is robust to misuse attacks

Figure 2.1 provides an illustration of the encryption and decryption algorithms of Ascon$[\rho, k, r, a, b]$.

From now on, we will simply refer to such a scheme as Ascon without explicitly mentioning the underlying parameters.

2.2 Ascon Security

In the original paper [8], the security analysis is mostly focused on the permutation function, while a general cryptanalysis of the scheme is generally missing in literature. The proof that we provide in this paper is largely based on the proof of the NORX scheme provided in [13]. In the final section of the paper, the authors argue that the same proof can be applied to the NORX scheme if the following conditions are met:

Statement 1 : The first statement taken verbatim from the original paper is as follows: "NORX uses domain separation constant at all rounds, but this is not strictly necessary and other solutions exist. In the privacy and integrity proofs of NORX, and more specifically at the analysis of the state collision caused by a decryption query in Lemma 3, the domain separations are only needed at the transitions between variable-length inputs, such as header to message data or message data to trailer data. This means that the proofs would equally hold if there were simpler transitions at these positions, which is the case for the Ascon scheme([8]).

Remark : In the threat model, we assume that the capacity part is unknown to the attacker. Differently from Ascon, NORX XORs the output of the permutation with the domain separation bits after each round, while Ascon only XORs the output of the permutation at the end of a stage. However, in the security analysis, we assume that the permutation ρ was replaced by a random function f. Hence, XOR-ing a random bitstring with the domain separation bits would not add any extra layer of security to the scheme.

Statement 2 : "The extra permutation evaluations at the initialization and finalization of NORX are not strictly necessary: in the proof we consider the monotone event that no state collides assuming no earlier state collision occurred. For instance, in the analysis of **Dhit** in the proof of Lemma 3, we necessarily have a new input to p at some point, and consequently all next inputs to p are new (except with some probability)"([8]).

Remark : NORX performs several extra steps of initialization and finalization compared to Ascon. In principle, the extra steps may increase the entropy of the inner state, but it may also increase the probability of collision. During the security analysis of the schemes, we analyze the probability of collision after each permutation, assuming that no collision had already occurred. At some stage

of the computation, the inner state will be assigned a new value and the chain of inner states will behave the same in both cases from that point on.

Statement 3 " NORX starts by initializing the state with $\mathbf{init}(K, N) = (K \parallel N \parallel 0^{b-\kappa-\nu}) \oplus c$ for some constant c and then permuting this value. Placing the key and nonce at different positions of the state does not influence the security analysis. The proof would also work if, for instance, the header is preceded with $K \parallel N$ or a properly padded version thereof and the starting state is 0^b."

Remark : In the Ascon scheme the state is initialized by concatenating the initial value, key and the nonce in the following order: $init(IV, K, N) = IV \parallel K \parallel N$, while NORX performs a similar initialization step but the order is $init(Kn, N) = K \parallel N \parallel 0^{b-\kappa-\nu}$. However, immediately after the initialization step, the states are permuted and a random state is therefore obtained, and the only attack that an adversary can perform at this stage is a bruteforce attack on the encryption key. After the first permutation round, the two schemes would behave similarly, so the steps from the security analysis of NORX can be slightly modified to obtain a similar security bound for the Ascon scheme.

Statement 4 "In a similar fashion, there is no problem in defining the tag to be a different τ bits of the final state; for instance the rightmost τ bits.

Remark : In both schemes the truncation of the state to the desired tag length τ is only performed in the last step of the computation. Hence, changing the tag length should not affect the security analysis under the assumption that no collision occurred up to that point.

Statement 5 " Key additions into the capacity part after the first permutation are harmless for the mode security proof. Particularly, as long as these are done at fixed positions, these have the same effect as XORing a domain separation constant".

Remark : This is the only statement from [13] that requires a thorough analysis as it is not immediately obvious why this statement holds. From the construction of both schemes, we notice that Ascon concatenates the capacity part with the key after the first permutation, while NORX uses a domain separation bit of 01 instead. Below we prove that both operations have the same effect on the security analysis. In the Ascon scheme, we note that if κ rightmost bits of the output of the random function after the first permutation is applied are equal to the encryption key, the capacity part will simply be a string of zeroes of size c, so the adversary can obtain the inner state by simply bruteforcing the rate part in 2^r attempts. This is the only bad event that can distinguish between the two games, because otherwise, after the second permutation function the capacity

part will simply be a random bitstring in both games. We denote the occurrence of such event by $key(i) \equiv [y_i]^{\kappa} = K$ where κ is the size in bits of the key and (x_i, y_i) are the inputs and outputs respectively of the i^{th} primitive query, and define $\mathsf{key}\forall_i key(i)$ the union of such event among all the primitive queries. By the fundamental lemma of game playing we obtain [1]:

$$\left|\Pr[\mathscr{A}^{Ascon}] - \Pr[\mathscr{A}^{NORX}]\right| \leq \Pr[\mathscr{A} \text{ sets } \mathsf{key}]$$

In the analysis, we replace the permutation ρ by a random function, so the probability that $key(i)$ occurs is simply $1/2^{\kappa}$. This implies that we easily adapt the security analysis of NORX to work for Ascon and simply add the extra $1/2^{\kappa}$ term in the final bound. For the sake of completeness, we add the security analysis in the next section, since it may be easier than the original one, since NORX uses several branches to process the message blocks, as opposed to Ascon which only uses one branch. Furthermore, as illustrated in Figure 2.2, the Ascon diagram presented in **??** is appended with the auxiliary notation s_i^j, t_i^j, which denote the input and output of the random function f respectively, in order to facilitate the security analysis in the next section.

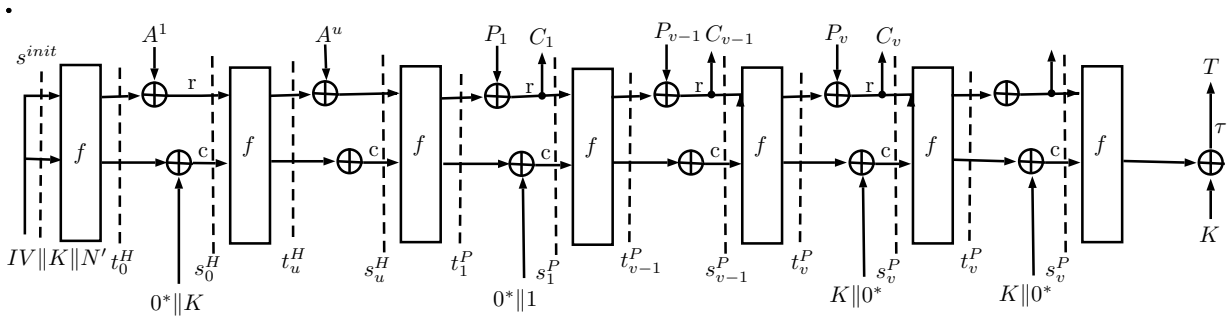


Figure 2.2 – Idealized Ascon scheme with the permutation ρ replaced by a random function f.

The above diagram illustrates an idealized Ascon construction where the permutation ρ is replaced by a random function f. Furthermore, we denote by s_i^H and t_i^H the inputs and outputs respectively to the random function during the stage of the processing of the associated data, and assume there are u such blocks We also denote by s_i^P and t_i^P the inputs and outputs respectively to the random function during the stage of the processing of the input blocks, and assume there are v such blocks.

2.3 Security Analysis

This proof was also provided in [13]. Since we proved in the previous section that the same security analysis also holds for our schemes when the event $key(i)$ does not occur, we can use the same reasoning as in the original paper, though it can be simplified since Ascon is just a simpler construction of NORX with a single branch. To facilitate the analysis, we define several events as

follows:

$$guess(i; j, k) \equiv x_i = s_{j,k}$$

$$coll(j, k; j', k') \equiv parent(s_{j,k}) \neq parent(s_{j',k'}) \wedge s_{j,k} = s_{j',k'}$$

We proceed to prove that if event does not occur, then $(f^{\pm}, \mathcal{E}_K)$ and $(f^{\pm}, \$)$ are indistinguishable, where $\$$ is an idealized random function, i.e., for each encryption, the ciphertext and the tag are assigned a random bitstring.

2.3.1 Confidentiality of the Ascon Scheme

Theorem 2. *Let* $\mathbf{Ascon}[\mathbf{p}, \mathbf{a}, \mathbf{b}, \kappa, \mathbf{r}, \tau]$ *denote the Ascon encryption scheme with key size* κ*, rate* r*, capacity* $c = b - r$*, and tag size* τ*, that uses a permutation of* a *rounds for the initial state and a permutation of* b *rounds for all the other state. Then, this scheme offers the following confidentiality guarantees:*

$$\mathbf{Adv}^{priv}_{\Pi}(q_p, q_\epsilon, \sigma_\epsilon) \leq \frac{q_p \cdot \sigma_\epsilon}{2^b} + \frac{(\sigma_\epsilon)^2}{2^{b+1}} + \frac{\sigma_\epsilon + q_p}{2^\kappa} + \binom{\sigma_\epsilon}{\rho} \cdot (2/2^r)^\rho$$

In the ideal world, we would have $(C_j, A_j) \leftarrow \{0,1\}^{|M|+\tau}$, i.e., each ciphertext block is randomly distributed. If we consider the diagram Figure 2.2, we observe that the j^{th} ciphertext of the i^{th} query is generated by the following formula: $_{i,j} = P_{ij} \oplus [f(s^P_{i,j-1})]^r$. Under the assumption that event has not occurred, each state is the output of a random function and hence XOR-ing a plaintext block P_{ij} which is chosen by the adversary and may be malicious, with a random bitstring will generate a random bistring as in the ideal world. The same logic applies to the tag, which is generated similarly to a ciphertext block and is truncated to τ bits. Having proven that an adversary cannot distinguish between the real-world and the ideal-world scenarios if event has not occurred, we proceed to prove that the probability that such an event occurs is small.

Event Guess.

Let q_ϵ denote the total number of encryption queries and σ_ϵ the total number of primitive calls(calls done offline during primitive queries or online during encryption queries). We first note that if $\exists\ i, j$ such that $x_i = s^{init}_j$, then event $key(i)$ occurs. Hence if we assume that this event has not occurred, then $x_i \neq s^{init}_j\ \forall i, j$. Using a simple probability trick, the following inequality holds:

$$\Pr[guess \vee coll] \leq \Pr[guess \vee coll | \neg key] + \Pr[key]$$

We proved previously that $\Pr[key(i)] = 1/2^\kappa$, for a single query i. The probability of the event for q_p primitive queries is $\frac{q_p}{2^\kappa}$. Let $j_i \in \{1, \ldots, \sigma_\epsilon\}$ denote the number of encryption queries before the j^{th} primitive query for $i \in \{1, \ldots, q_p\}\}$. Assuming that event has not already occurred, the

probability that it occurs during the j^{th} encryption query is:

$$\sum_{i=1}^{q_p}\sum_{j=1}^{j_i} \leq q_p \cdot \sum_{j=1}^{q_\epsilon} \frac{\sigma_{\epsilon,j}}{2^b} \leq \frac{q_p \cdot \sigma_\epsilon}{2^b}$$

Event coll.

This event happens if there's a collision in the input of two primitive calls during the i^{th} and j^{th} decryption query for $i \neq j$, which happens with probability $\frac{\binom{\sigma_\epsilon}{2}}{2^b} \leq \frac{(\sigma_\epsilon)^2}{2^{b+1}}$. Collision with the initial state s_i^{init} should be considered separately as the input of the first primitive call is not a random bitstring like the other primitive calls, but rather a concatenation of the initialization value, key and nonce. Collision with such a state occurs if $\exists j \neq i$ such that $[s_j]^\kappa = K$, i.e., the rightmost κ bits of the state are equal to the bits of the key, where κ is the size in bits of the key. This event occurs with probability $\frac{\sigma_\epsilon}{2^\kappa}$. We also emphasize that differently from the security proof of **vCCM**, here we don't need to worry about collisions because of common prefixes since for a given query i, the initial state is encoded as $s_i^{init} = IV_{k,r,a,b} \parallel K \parallel N$. For different nonces the inner states will differ in the capacity part, so even if the adversary performs two different queries (N, A, M, τ), (N', A', M', τ') such that $[A]^k = [A']^k$ or $A = A'$ and $[M]^k = [M']^k$ for an arbitrary k, then $[s_{i,j}]^r = [s'_{i,j}]^r$. This will result in a collision only if the capacities also match, i.e $[s_{i,j}]^c = [s'_{i,j}]^c$, which occurs with probability $1/2^c$. If we assume that the number of states that collide in the rate part is limited by a threshold ρ, then the union of all such probabilities would be $\frac{q_p \cdot \rho}{2^c}$. Now, we proceed to find the probability that this is actually the case, i.e., that there are at most ρ states that collide in the inner part r. To derive the probability of such collision, we consider a state $s_{j,k-1}$ in the inner state chain of some arbitrary query i. Let $x \in \{0,1\}^r$ be a fixed value. Then one of the states collide with x if $f(s_{j,k-1}) = x$ or $f(s_{j,l-1}) \oplus \alpha = \text{x}$, where α is a predetermined adversarial input in the form of the message blocks or the associated data blocks. Hence, the probability of a collision for a single state is $\frac{2}{2^r}$. Since we're considering the event of collision of ρ states among σ_ϵ primitive calls, the union of all the events yields a probability of $\binom{\sigma_\epsilon}{\rho} \cdot (2/2^r)^\rho$. Summing the bounds for the above events we obtain the following bound:

$$\Pr[\mathsf{event}] \leq \frac{q_p \cdot \sigma_\epsilon}{2^b} + \frac{(\sigma_\epsilon)^2}{2^{b+1}} + \frac{\sigma_\epsilon + q_p}{2^\kappa} + \binom{\sigma_\epsilon}{\rho} \cdot (2/2^r)^\rho$$

2.3.2 Authenticity of the Ascon Scheme

Theorem 3. *Let* $\mathbf{Ascon}[\mathbf{p}, \mathbf{a}, \mathbf{b}, \kappa, \mathbf{r}]$ *denote the Ascon encryption scheme with key size* κ*, rate* r*, capacity* $c = b - r$*, and tag size* τ*, that uses a permutation of* a *rounds for the initial state and a permutation of* b *rounds for all the other states. Then, this scheme offers the following authenticity guarantees:*

$$\mathbf{Adv}_{\Pi}^{auth}(q_p, q_\epsilon, \sigma_\epsilon) \leq \frac{q_p \cdot \sigma_\epsilon}{2^b} + \frac{(\sigma_\epsilon)^2}{2^{b+1}} + \frac{\sigma_\epsilon + q_p}{2^\kappa} + \binom{\sigma_\epsilon}{\rho} \cdot (2/2^r)^\rho + \frac{\sigma_D \cdot q_p}{2^c} + \frac{1}{2^c}$$

where q_p denotes the number of offline primitive calls, σ_ϵ denotes the total number of primitive calls during the encryption queries and σ_D denotes the total number of primitive calls during the decryption queries.
The analysis in this case proceeds similarly to the one in the previous section. However, in this case, an adversary is also allowed to do decryption(forgery) queries, which we denote by $_D$, besides the encryption and primitive queries. If the adversary performs a query (N, A, C, T) such that the decryption algorithm does not return $\perp$, then it has successfully performed a forgery, which we denote by $\mathscr{A}^{forges}$. For the ease of notation, also assume that triggering such an event sets bad-flag to true. We also denote by σ_D the total number of primitive calls during a decryption query, and extend the bad event from the previous sections as follows:

$$bad = guess \vee coll \vee \mathcal{D}guess \vee \mathcal{D}coll$$

where $\mathcal{D}guess$ and $\mathcal{D}coll$ are the equivalent events of $guess$ and $coll$ defined for a decryption query, i.e., $\mathcal{D}coll$ denotes the event that the input of a primitive call during a decryption query collides with the input of a primitive call, and $\mathcal{D}coll$ denotes the event of a collision between inputs of primitive calls in two different queries. As shown in the previous section, when bad-flag is not set, the algorithm behaves identically to the ideal oracle by responding to each query with a random bitstring. Hence, the probability of a collision in this case is 2^τ where τ is the tag length, since the adversary $\mathscr{A}$ can only perform a bruteforce attack. The analysis of the events $guess$ and hit proceeds in the same way as in the previous section. Hence, we only focus on the analysis of the events $\mathcal{D}guess$ and $\mathcal{D}coll$. The $\mathcal{D}guess$ event is triggered when the input to one of the online primitive calls that are made during a decryption query collides with the input of one of the offline primitive calls. As the number of total primitive calls during the decryption queries is σ_D and the total number of offline primitive calls is q_p the probability of the occurrence of such event assuming that the event has not occurred is:

$$\Pr[\mathcal{D}guess|\neg key] = \frac{\sigma_D \cdot q_p}{2^c}$$

The reason for assuming that key has not already occurred is because, as in the previous paragraph, collision with an initial state should be considered separately because of its special construction. This event occurs with probability $1/2^\kappa$ where κ is the size in bits of the key.

Event Dcoll.

$\mathcal{D}coll$ captures the event of the collision of the inner state chains during two encryption queries. The queries may differ in the nonce N, the ciphertext C, the associated data A or the tag size τ, and the analysis differs in each of these cases, so a case by case analysis is needed to achieve a good bound. Assume the current query is (N, A, C) and let $(N_{\delta,j}, A_{\delta,j}), C_{\delta,j}$ be a previous query with the longest common prefix with the current query. Furthermore, we let $\gamma \in \{\mathcal{E}, \mathcal{D}\}$ as we are seeking for collisions in the primitive calls of both encryption and decryption queries. Several cases may arise as follows:

1. $\mathbf{(N,A,C)} = (\mathbf{N}_{\gamma,\mathbf{j}}, \mathbf{A}_{\gamma,\mathbf{j}}, \mathbf{C}_{\gamma,\mathbf{j}}), \tau \neq \tau_{\gamma,\mathbf{j}}$. Since τ is only generated in the last step by the truncation of the inner state, the current query will not generate any new state.
2. $\mathbf{(N,A)} = (\mathbf{N}_{\sigma,\mathbf{j}}, \mathbf{A}_{\sigma,\mathbf{j}}), \mathbf{C} \neq \mathbf{C}_{\sigma,\mathbf{j}}$. We define $l \in \{1, \ldots, min(v, v_{\gamma,j}), \infty\}$ where $1 \leq l \leq min(v, v_{\gamma,j})$ where $v = |C|/r$ and $v_{\gamma,j} = |C_{\gamma,j}|$ is the number of ciphertext blocks, i.e., l denotes the first block where the two input chains of the decryption query differ. Note that l can be ∞ if one of the blocks is a substring of the other. For ease of analysis, we define the internal collision event as follows:

$$\mathsf{coll\text{-}internal}(\gamma, j; \gamma', j'; const) \equiv \mathsf{parent}(s_{\gamma,j}) \neq \mathsf{parent}(s_{\gamma',j'}) \wedge [\mathsf{s}_{\gamma,j}]^c = [\mathsf{s}_{\gamma',j'}]^c$$

Since the output of primitive calls are random bitstrings of size c, we have:

$$\Pr[\mathsf{coll\text{-}internal}] \leq \frac{1}{2^c}$$

Here, we have to distinguish two subcases:

(a) $\mathbf{l} = \infty$. In this case, if we assume without loss of generality that $v < v_{\gamma,j}$, i.e., the input of the current query is a prefix of the input of a previous query, then $s^C_{\gamma,j,v+1} = s^C_v \oplus K \parallel 0*$, where $K \parallel 0*$ are simply the domain separation bits used to separate the message processing and the tag generation stages, will be a new state if coll-internal has not occurred, and the probability of a collision would simply be $1/2^c$.
(b) $\mathbf{l} < \infty$. The same reasoning as above also applies in this case, because $s_l \neq s_{\gamma,j,l}$, so if coll-internal has not occurred, the probability of a collision at some future output of the chain is simpy $1/2^c$.
3. $N = N_{\gamma,j}, A \neq A_{\gamma,j}$ Same reasoning as above applies for this case also. The case $N \neq N_{\gamma,j}$ is trivial because if we assume that the event key has not occurred, then s_i^{init} will be new for each query.

2.4 Variable Tag Ascon

Theorem 4. *Let* $\mathbf{Ascon}[\mathbf{p}, \mathbf{a}, \mathbf{b}, \kappa, \mathbf{r}]$ *denote the Ascon encryption scheme with key size* κ*, rate* r *and capacity* $c = b - r$ *that uses a permutation of* a *rounds for the initial state and a permutation of* b *rounds for all the other state. Then, this scheme offers the following security guarantees:*

$$\begin{aligned}\mathbf{Adv}^{nvae}_{Ascon[p,a,b,\kappa,r]}(q_p, q_\epsilon, \sigma_\epsilon) \leq & \frac{q_p \cdot \sigma_\epsilon}{2^{b-1}} + \frac{(\sigma_\epsilon)^2}{2^b} + \frac{\sigma_\epsilon + q_p}{2^{\kappa-1}} \\ & + \binom{\sigma_\epsilon}{\rho} \cdot (2/2^r)^{\rho-1} + \frac{\sigma_D \cdot q_p}{2^c} + \frac{\sigma_q}{2^c}\end{aligned}$$

In [21], Reyhanitabar, Vaudenay and Vizár briefly mention that including the tag length in the nonce input of sponge-like authenticated encryption schemes is a secure transformation for variable tag lengths. In [17], Vizár et al., provide an algorithm to reduce any call to an AE scheme to a call to a duplex construction, which is a standard and widely used construction is cryptography with

the same security guarantees as a sponge construction [2]. However, the same reduction cannot be applied to Ascon the final inner state before the tag generation is XORed with the key. For the security analysis of the variable tag Ascon, we note that the ideal-world and the real world oracles behave the same except for the case when there is a collision in the input of one of the primitive calls. In the ideal world, we would sample a permutation function $p : \{0,1\}^b \to \{0,1\}^b$ for every different tag length τ. Hence, even if there was a collision in the input of a primitive call for different queries, the primitive would still produce different outputs. In the real-world, however, we use the same primitive for all the tag lengths, so a collision in the input of primitive calls between different queries would result in a collision in the outputs also. For the security analysis of the variable tag Ascon, we use a similar procedure to section 3.2, where we use the notion of the longest common prefix introduced earlier. For the ease of notation, we will also refer to the variable tag Ascon implementation as vAscon. Furthermore, we denote by $s_{i,j}$ the j^{th} block of the i^{th} query, and assume that the current encryption query is (N, A, P), and the query with the longest common prefix with the current query is (N', A', P'). If we denote by f_τ the set of inputs to all the primitive calls during an encryption or decryption query and by $f = \bigcup_{\tau \in \mathcal{I}_T}$ the union of such sets for all the tag lengths, the first event corresponds to $x \in Dom(f) \wedge x \in f_\tau$ where τ is the current tag length and x is an input to a primitive call, while the second event corresponds to $x \in Dom(f) \wedge x \notin Dom(f_\tau)$. Collisions with the initial states should be considered separately, since differently from the other states, this state has no randomness. We assume that the event bad sets the flag bad-flag to true. By induction, we assume that this flag has not previously set and proceed as follows:

Case 1: $j = 0$. By construction $s_i^{init} \neq s_{i'}^{init}\ \forall i \neq i'$. The collision that $s_i^{init} = s_{i',j}$ for $j > 0$ is also 0, since they have different size as we can note from Figure 2.1, that the first permutation differs from the others.

Case 2: $\mathbf{j = \mathsf{llcp_b}(i) + 1}$. The collision event in this case is $P_i \oplus s_{i,j} = P_{i'} \oplus s_{i',j}$. In this case the adversary can just trigger a collision in the rate part by picking $P_{i,j} = P_{i',j}$ since we know that $[s_{i,j}]^r = [s_{i,j'}]^r$. However, $[s_{i,j}]^c$ is still randomly sampled, so the probability of collision in this case is $1/2^c$.

Case3: $\mathbf{j < \mathsf{llcp_b}(i) + 1}$. y induction hypothesis we know that the flag bad-flag has not been previously set and since $\mathsf{llcp}_b(i) + 1 > j$, the state is freshly sampled, so the probability of collision is $1/2^b$. The same analysis could also be applied in case of a decryption query.

If we denote by a_i, m_i the total number of associated data blocks and message blocks respectively during the i^{th} query, for each query we will make $(a_i + m_i + 1)$ calls to the primitive, so the total primitive calls along all the queries would be:

$$\sum_{i=1}^{q_\mathcal{E}+q_\mathcal{D}} (a_i + m_i + 1) = \sigma + q$$

for $\sigma = \sigma_\mathcal{E} + \sigma_\mathcal{D}$ and $q = q_\mathcal{E} + q_\mathcal{D}$. Taking the union of all such events, we obtain:

$$\Pr[\mathscr{A} \text{ sets bad-flag}] \leq \frac{\sigma + q}{2^c}$$

Chapter 3

Design

This chapter seeks to provide an introduction to the variable tag transformation of the CCM scheme, focusing mainly on the ease of its implementation. Furthermore, we also analyze the security of such transformation by following a game-based approach with the two main games defined in Figure 3.6.

3.1 Variable Tag CCM

vCCM is a nonce-based AE scheme parameterized by a blockcipher $\mathbf{E} : \mathcal{K}_\tau \times \{0,1\}^n \to \{0,1\}^n$ with $n = 128$, an ordered set of non-zero tag lengths $\mathcal{I}_T \subseteq \{32, 48, 64, 80, 96, 112, 128\}$, and a nonce length $\nu \in \{56, 64, 72, 80, 88, 96\}$. There is a trade-off between the nonce length ν and the maximal message length of $8 \cdot (2^{(112-\nu)} - 1)$ bits. The encryption and the decryption algorithms of $\mathbf{vCCM}[\mathbf{E}, \mathcal{I}_T, \nu]$ are described in Figure 3.1 and illustrated in Figure 3.2.

.

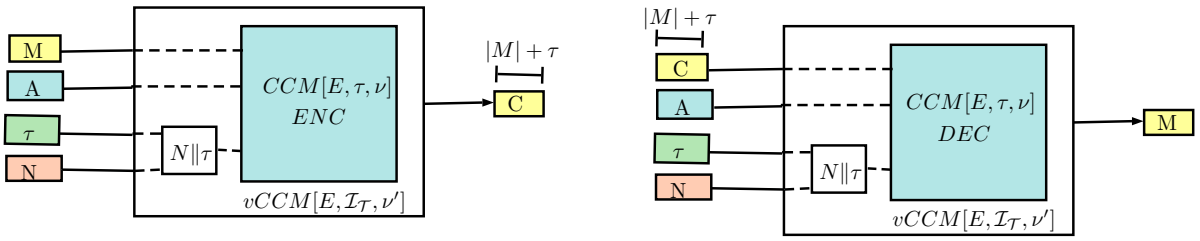


Figure 3.2 – Illustration of the black box transform of **CCM** to **vCCM**. Here, $\nu' = \nu - 8$.

The Transformation.

```
101: Algorithm  E_K(N, A, τ, M)
102:     N' ← N‖⟨τ/8⟩_8
103:     C = CCM[E, τ, ν + 8].E_K(N', A, M)
104:     return C

201: Algorithm  D_K(N, A, τ, C)
202:     N' ← N‖⟨τ/8⟩_8
203:     M = CCM[E, τ, ν + 8].D_K(N', A, C)
204:     return M
```

Figure 3.1 – **vCCM**$[E, \mathcal{I}_T, \nu]$ **mode for AEAD**, with $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a blockcipher.

The nvAE scheme vCCM is obtained as a black-box transformation of CCM, requiring no modifications to the standard CCM. This property has been our primary design goal, as it is key for the ability of the new nvAE scheme to benefit from existing software and hardware implementations, and thus be instantly used at a massive scale.

3.1.1 Design Rationale

Tag length in AD : If the tag length was injected in the associated data and the same message is encrypted with different tag lengths, the change of the tag would only be propagated in the last block of the ciphertext, while the rest of the blocks would remain the same. Hence, a **kess** adversary would be able to easily distinguish between the real and ideal case scenarios with probability $1 - 2^{-n}$, where n is the size in bits of the ciphertext.

Tag length in the nonce : In this case, a change in the tag length would be propagated in the CBC-MAC calculations, because the nonce is encoded in the first block, and in all the encrypted blocks, because the inputs to the block cipher of the CTR mode consist of a concatenation of a flags field, a nonce and a counter.
From the discussion above, it can be clearly seen that injecting the tag length in the nonce is the best option, which justifies the construction depicted in Figure 3.2.

3.2 Security of vCCM

To analyze the security of vCCM, we will make use of the following extension to the security notion of the standard CCM:

Extension to the Standard Notion The standard CCM takes as input a key, nonce, associated data, and message and outputs a ciphertext during encryption as follows:

$$\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C}$$

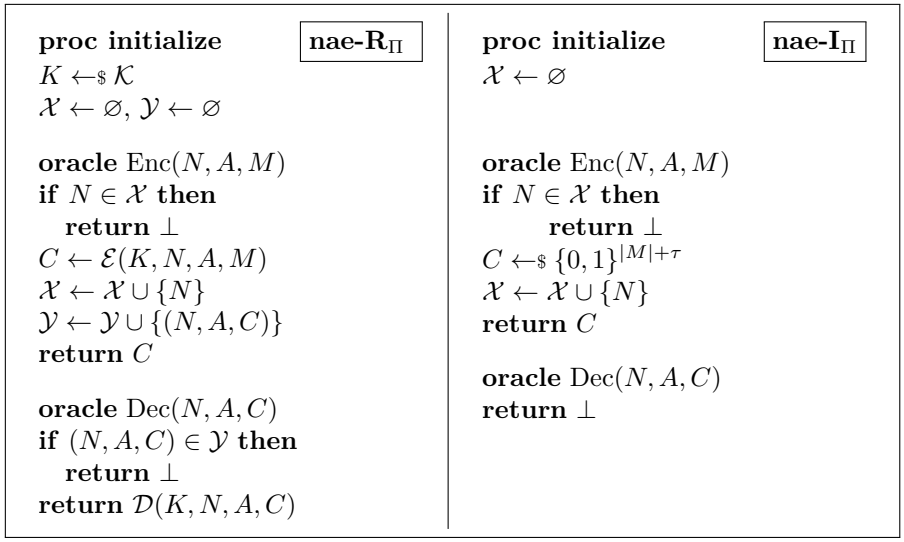


Figure 3.3 – **All-in-one definition** of nAE security for a scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with ciphertext expansion τ.

The decryption function takes the same inputs, except that the ciphertext and the plaintext switch places as follows:

$$\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \rightarrow \mathcal{M}$$

For our variable tag black box transform, we need to add another input to the scheme, $\mathcal{I}_T$, which is the set of all the possible tag lengths for the scheme, i.e.

$$\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{I}_T \times \mathcal{A} \times \mathcal{M} \rightarrow \mathcal{C}$$

and similarly for the decryption

$$\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{I}_T \times \mathcal{A} \times \mathcal{C} \rightarrow \mathcal{M}$$

The security of the standard CCM is defined in [21] as illustrated in Figure 3.3. From this figure we observe that the authors use a game-based approach to define the **nae**−security notion where the adversary communicates with a real-world oracle(shown in the left half of the figure), and an ideal-world oracle(shown in the right half of the figure). The real-world oracle responds to the adversary queries by using the real encryption/decryption oracles, while the ideal world oracle returns a random bitstring of length $|M| + \tau$ in case of an encryption query and the null string in case of a decryption query. The notion of the adversary's advantage seeks to quantify the ability of an adversary to distinguish between the two oracles. The notions of **Kess** and **nvAE Security** introduced in the following section will facilitate the proof of the security of vCCM.

nvAE Security and Kess Notion The notion of the AE security for variable tag lengths is capture by the following two games defined in [21]:

In this case, we observe that differently from Figure 3.3, the games are now parameterized by

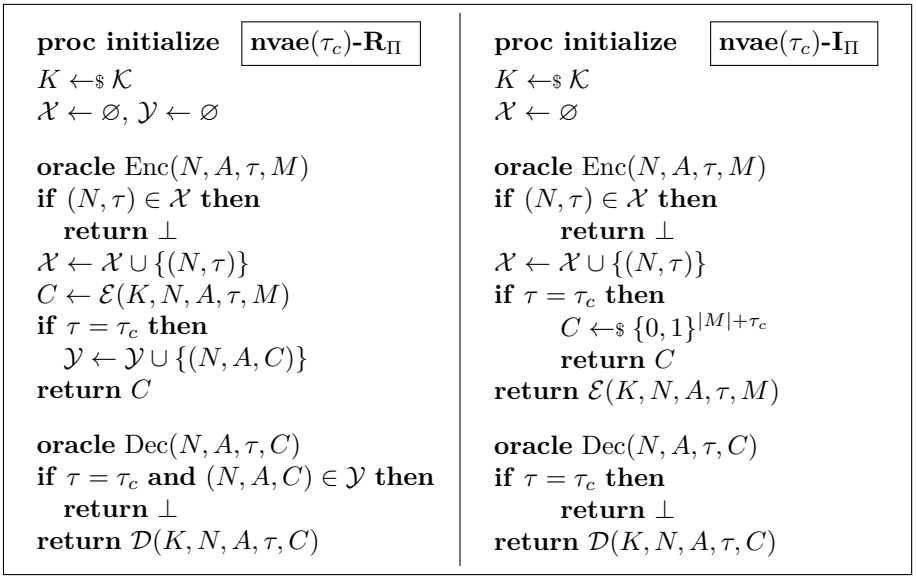


Figure 3.4 – **AE security with variable stretch.** Security games for defining AE security of a nonce-based AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with variable-stretch.

an argument τ. This is done in order to avoid trivial attacks, since our scheme now accepts different τs, and hence the attacker can claim a large advantage over the scheme by simply attacking the shortest tag. However, this definition prevents this trivial attack, since both oracles respond with the same encryption/decryption algorithms for all $\tau \neq \tau_c$. However, the attacker may still use queries with the other tags to gain some information about the blockcipher, which can still be useful when attacking the scheme with $\tau = \tau_c$. In order to facilitate the proof of the **nvae**−security, Reyhanitabar et al. introduce the notion of **kess**− security as illustrated in the games in Figure 3.5: Connecting all the above notions, Reyhanitabar, Vaudenay and Vizár proved in [21] the following theorem:

Theorem 5 (**kess** $\wedge$ **nae** $\Rightarrow$ **nvae**). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a nonce-based AE scheme with variable stretch. We have that*

$$\mathbf{Adv}^{\boldsymbol{nvae}(\tau_c)}_{\Pi}(t, \mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}) \leq \mathbf{Adv}^{\boldsymbol{kess}}_{\Pi}(t', \mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}) + \mathbf{Adv}^{\boldsymbol{nae}}_{\Pi[\tau_c]}(t'', q_e^{\tau_c}, q_d^{\tau_c}, \sigma^{\tau_c}),$$

with $t' = t + O(q)$ and $t'' = t + O(\sigma)$ where $q = \sum_{\tau \in \mathcal{I}_T}(q_e^\tau + q_d^\tau)$ and $\sigma = \sum_{\tau \in \mathcal{I}_T}(\sigma_e^\tau + \sigma_d^\tau)$.

Hence, in order to prove that a **nae**-secure AE-scheme(like CCM) is also **nvae**-secure, i.e., it is secure under the variable tag transformation, we just need to prove that it satisfies the **kess** property as defined above. Intuitively, **kess** seeks to capture the relation between schemes with different tag lengths. In the ideal case, as illustrated in the right-hand side of Figure 3.5, we would have a different key for each tag length. However, this would be quite inefficient, especially in embedded systems where memory could also be a constraint. Hence, in the real world, illustrated on the left-hand side of Figure 3.5, we use the same key for different tag lengths, which in some cases will not be secure as the adversary will be able to distinguish between the real-world and ideal-world oracle with high probability. However, in the analysis of vCCM, we show that this event

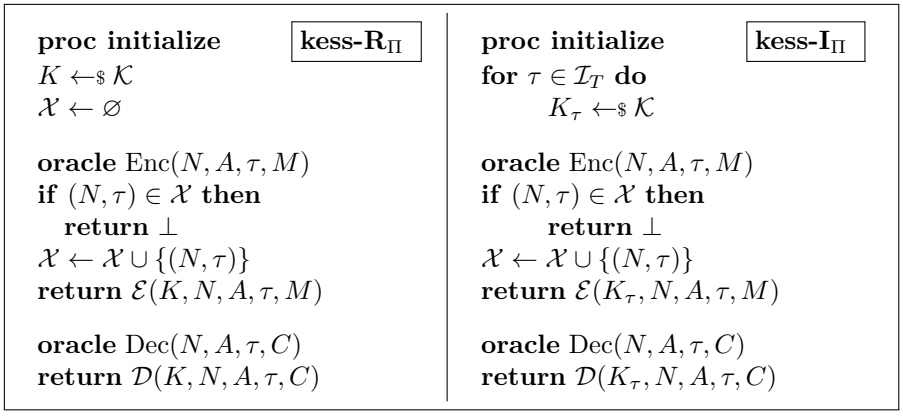


Figure 3.5 – **Key-equivalent separation by stretch.** Games defining **kess** property of a nonce-based AE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with variable stretch. Note that the independent keying for each $\tau \in \mathcal{I}_T$ in game **kess-I**$_\Pi$ can be done by lazy sampling if needed.

happens with very low probability, and hence the scheme is **nvae**−secure.

In this section, we formally state and prove the nvae security of **vCCM** in Theorem 6. The analysis is based on Theorem 5, allowing to reuse the result on nae security of CCM by Jonsson [12] and leaving only the kess property of **vCCM** to be analyzed. We formally state and prove the latter in Lemma 3.2 using the technique of code-based games [1].

Theorem 6. *Let* $\mathrm{E} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ *with* $n = 128$ *be a blockcipher,* $\mathcal{I}_T \subseteq \{32, 48, 64, 80, 96, 112, 128\}$ *and* $\nu \in \{56, 64, 72, 80, 88, 96\}$*. Then the following inequality holds:*

$$\begin{aligned}\mathbf{Adv}^{\boldsymbol{nvae}[\tau_c]}_{\boldsymbol{vCCM}[E,\mathcal{I}_T,\nu]}(t, \mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}_\mathbf{e}, \boldsymbol{\sigma}_\mathbf{d}) \leq & |\mathcal{I}_T| \cdot \mathbf{Adv}^{\boldsymbol{prp}}_{\mathrm{E}}(t', \sigma') + 3 \cdot \frac{(2\sigma + 3q)^2}{2^n} \\ & + \frac{q_d}{2^\tau} + \frac{(2\sigma^{\tau_c} + 3q^{\tau_c})^2}{2^{n+1}} + \frac{(2\sigma_e^{\tau_c} + 3q_e^{\tau_c})^2}{2^{n+1}}\end{aligned}$$

where $\sigma' \leq 2\sigma + 3q$ *and* $t' \leq t + \sigma' \cdot \gamma$ *for a "small" constant* γ*, and the adversarial resources are as defined in* **??**.

Corollary 6.1. *As* $|\mathcal{I}_T| \leq 7$*, we directly have*

$$\begin{aligned}\mathbf{Adv}^{\boldsymbol{nvae}[\tau_c]}_{\boldsymbol{vCCM}[E,\mathcal{I}_T,\nu]}(t, \mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}_\mathbf{e}, \boldsymbol{\sigma}_\mathbf{d}) \leq & 7 \cdot \mathbf{Adv}^{\boldsymbol{prp}}_{\mathrm{E}}(t', \sigma') + 3 \cdot \frac{(2\sigma + 3q)^2}{2^n} \\ & + \frac{q_d}{2^\tau} + \frac{(2\sigma^{\tau_c} + 3q^{\tau_c})^2}{2^{n+1}} + \frac{(2\sigma_e^{\tau_c} + 3q_e^{\tau_c})^2}{2^{n+1}}\end{aligned}$$

Proof. The first step of the proof consists in replacing the block cipher by a random permutation $\pi \leftarrow_\$ \mathrm{Perm}(n)$ in both games **nvae**(τ_c)**-R** and **nvae**(τ_c)**-I** to obtain the following:

$$\mathbf{Adv}^{\mathbf{nvae}(\tau_c)}_{\mathbf{vCCM}[E,\mathcal{I}_T,\nu]}(t, \mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}_\mathbf{e}, \boldsymbol{\sigma}_\mathbf{d}) \leq \mathbf{Adv}^{\mathbf{nvae}(\tau_c)}_{\mathbf{vCCM}[\pi,\mathcal{I}_T,\nu]}(t, \mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}) + 7 \cdot \mathbf{Adv}^{\mathbf{prp}}_{E}(t', \sigma') \ ,$$

where $\sigma' \leq 2\sigma + 3q$ and $t' \leq t + \sigma' \cdot \gamma$. Furthrermore, we note that the second term needs to be multiplied by $|\mathcal{I}_T|$, because in the real word we only use an instance of E, but in the ideal world, we would use one for each tag length.
We can also use the standard $PRP \rightarrow PRF$ switch from [1], to replace the random permutation by a random function $f \leftarrow\!\!\$\ \mathrm{Func}(n)$ and obtain:

$$\mathbf{Adv}^{\mathbf{nvae}(\tau_c)}_{\mathbf{vCCM}[\pi,\mathcal{I}_T,\nu]}(t, \mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}_\mathbf{e}, \boldsymbol{\sigma}_\mathbf{d}) \leq \mathbf{Adv}^{\mathbf{nvae}(\tau_c)}_{\mathbf{vCCM}[f,\mathcal{I}_T,\nu]}(t, \mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}) + 2 \cdot \frac{(2\sigma + 3q)^2}{2^{n+1}}.$$

The random function in the real-world case is not called more than $2\sigma + 3q$ times, and similarly, the $|\mathcal{I}_T|$ permutations in the ideal-world case are not called more than $2\sigma + 3q$ times. Since a random permutation is surjective, while a random function is not, the only bad case occurs in case of a collision in the outputs of the random function. As there are $2\sigma + 3q$ such calls, the probability of collision is $\frac{\binom{2\sigma+3q}{2}}{2^n} \leq \frac{(2\sigma+3q)^2}{2^{n+1}}$. Adding both the ideal and real world cases, we obtain the last term on the right-hand side of the inequality. Next, we use Theorem 5 to get the following inequality

$$\begin{aligned}\mathbf{Adv}^{\mathbf{nvae}(\tau_c)}_{\mathbf{vCCM}[f,\mathcal{I}_T,\nu]}(\mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}_\mathbf{e}, \boldsymbol{\sigma}_\mathbf{d}) \leq\ & \mathbf{Adv}^{\mathbf{kess}}_{\mathbf{vCCM}[f,\mathcal{I}_T,\nu]}(\mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}_\mathbf{e}, \boldsymbol{\sigma}_\mathbf{d}) \\ & + \mathbf{Adv}^{\mathbf{nae}}_{\mathbf{vCCM}[f,\{\tau_c\},\nu]}(q_e^{\tau_c}, q_d^{\tau_c}, \sigma_e^{\tau_c}, \sigma_d^{\tau_c})\ .\end{aligned}$$

From Corollary 1.1 and the fact that $\mathbf{Adv}^{\mathbf{prf}}_f(\mathscr{A}) = 0$, the following inequality holds:

$$\mathbf{Adv}^{\mathbf{nae}}_{\mathbf{vCCM}[f,\{\tau_c\},\nu]}(q_e^{\tau_c}, q_d^{\tau_c}, \sigma_e^{\tau_c}, \sigma_d^{\tau_c}) \leq \frac{q_d}{2^\tau} + \frac{(2\sigma^{\tau_c} + 3q^{\tau_c})^2}{2^{n+1}} + \frac{(2\sigma_e^{\tau_c} + 3q_e^{\tau_c})^2}{2^{n+1}}$$

The proof is finalized by applying Lemma 3.2 to upper bound $\mathbf{Adv}^{\mathbf{kess}}_{\mathbf{vCCM}[f,\mathcal{I}_T,\nu]}(t, \mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}_\mathbf{e}, \boldsymbol{\sigma}_\mathbf{d})$. □

By using the **kess**−security definition in Figure 3.5 and the games in Figure 3.6, we obtain the following bound.

Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a random function $f \leftarrow\!\!\$\ \mathrm{Func}(n)$ with $n = 128$, $\mathcal{I}_T \subseteq \{32, 48, 64, 80, 96, 112, 128\}$ and $\nu \in \{56, 64, 72, 80, 88, 96\}$. Then the following inequality holds:

$$\mathbf{Adv}^{\mathbf{kess}}_{\mathbf{vCCM}[f,\mathcal{I}_T,\nu]}(\mathbf{q_e}, \mathbf{q_d}, \boldsymbol{\sigma}_\mathbf{e}, \boldsymbol{\sigma}_\mathbf{d}) \leq 2 \cdot (2\sigma + 3q)^2/2^n\ .$$

Proof. We use the games defined in Figure 3.6 to analyze the **kess** advantage of the adversary. The algorithm for the games is obtained by using the **vCCM** for the encryption and decryption oracles of the **kess** games, with the blockcipher replaced by a random function as previously discussed.

kess games The if-else blocks are used to ensure that the outputs from the primitive conform to the cases when a single random function is used for all the tag lengths or a different f is used for each tag length. When this rule is violated, this implies that an adversary can easily distinguish

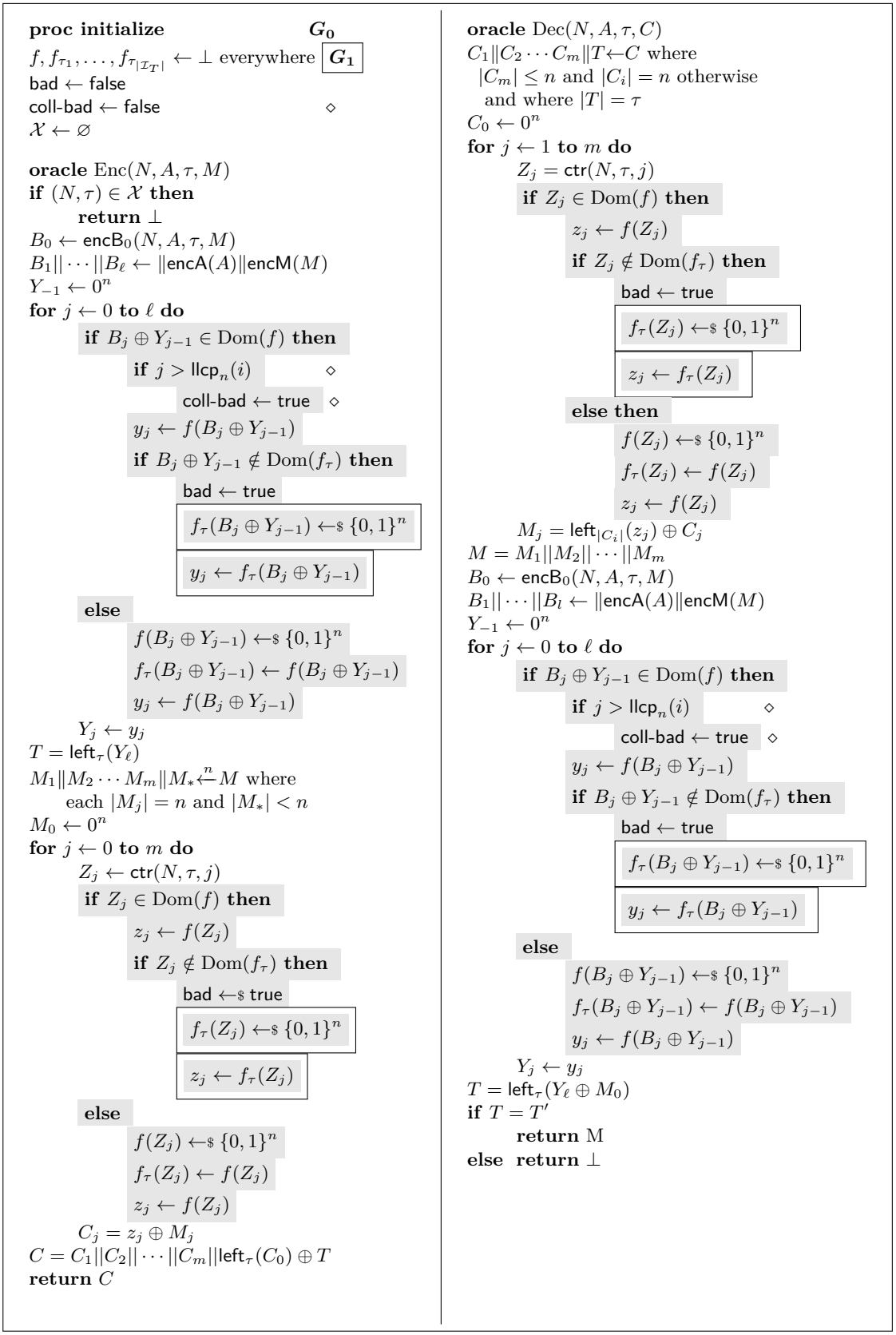


Figure 3.6 – **Games Definition** Adversarial games $\boldsymbol{G_0}$ and $\boldsymbol{G_1}$ for the proof of Lemma 3.2. $\boldsymbol{G_0}$ is obtained by omitting the boxed statements and the statements marked by $\diamond$, while $\boldsymbol{G_1}$ is obtained by including the boxed statements and omitting the statements marked by $\diamond$. Here, we let for a partially defined function f, $\mathrm{Dom}(f)$ denote the set $\{x \in \{0,1\}^n | f(x) \neq \perp\}$. The functions $\mathsf{encB}_0()$ and $\mathsf{ctr}()$ are extended to process τ in the obvious way. For definition of $\mathsf{llcp}_n(i)$ see proof of Lemma 3.2.

between the two games, and hence both games are set the flag bad to true. Above we claimed that our two games simulate the **kess-R** and **kess-I** respectively, but this is not given, so we first need to provide an argument that this is really the case. For $\boldsymbol{G_0}$ this is easy to see because when the boxed statements are removed, the game will just overwrite the image of an already existing point if $B_j \oplus Y_{j-1}$ is already in $\mathrm{Dom}(f)$ or will extend the domain by $B_j \oplus Y_{j-1}$ if this value is not already in the domain.
On the other hand, the statements are included in $\boldsymbol{G_1}$. In this case, if $B_j \oplus Y_{j-1}$ is already in $\mathrm{Dom}(f)$, then the code will check if the value was sampled during a query with the same tag length or with a different one. In the first case, $B_j \oplus Y_{j-1} \in \mathrm{Dom}(f_\tau)$, and the two games will still behave the same so the bad is not set. However, if $B_j \oplus Y_{j-1}$ was sampled with a different tag length, then $B_j \oplus Y_{j-1} \notin \mathrm{Dom}(f_\tau)$. Let's assume that the current tag length is τ' and $B_j \oplus Y_{j-1}$ was sampled during a query with tag length τ. Then, in the ideal world, the value of $f_\tau(B_j \oplus Y_{j-1})$ would just be overwritten by $f_{\tau'}(B_j \oplus Y_{æ-1}) \leftarrow\!\!\$\ \{0,1\}^n$, which is also what happens in our game, when we sample a new value for $f_{\tau'}(B_j \oplus Y_{j-1})$ after setting the bad to true. Hence, we showed that $\boldsymbol{G_0}$ and $\boldsymbol{G_1}$ simulate **kess-R** and **kess-I** respectively, which implies the following equality:

$$\mathbf{Adv}^{\mathbf{kess}}_{\mathbf{vCCM}} = \left|\Pr[\mathscr{A}^{\mathbf{kess\text{-}R}} \Rightarrow 1] - \Pr[\mathscr{A}^{\mathbf{kess\text{-}I}} \Rightarrow 1]\right| = \left|\Pr[\mathscr{A}^{\boldsymbol{G_0}} \Rightarrow 1] - \Pr[\mathscr{A}^{\boldsymbol{G_1}} \Rightarrow 1]\right|$$

Furthermore, by the fundamental lemma of game-playing [1], the following inequality also holds:

$$\Pr[\mathscr{A}^{\boldsymbol{G_0}} \Rightarrow 1] - \Pr[\mathscr{A}^{\boldsymbol{G_1}} \Rightarrow 1] \leq \Pr[\mathscr{A}^{\boldsymbol{G_0}} \text{ sets } \mathsf{bad}] \ ,$$

where the event "sets bad" is defined as the flag bad being true at the end of the experiment.

Assumptions on the experiment For the following collision analysis, we assume that the adversary does not make any repeated queries and for each decryption query (N, A, C), there exists no encryption query (N, A, M) that returned C. As the responses to these queries are trivially known, this assumption causes no loss of generality in the security analysis. In Figure 3.6, we denote by $(N^i, A^i, \tau_i, M^i, C^i)$ denote the nonce, the AD, the stretch, the plaintext and the ciphertext respectively of the i^{th} query. Let $B^i_0, \ldots, B^i_l$ denote the input blocks of CBC-MAC during the i^{th} query and by $Y^0_i, \ldots, Y^i_l$ the corresponding outputs. We also denote by $Z^i_0, \ldots, Z^i_m$ the inputs to the CTR, and by $z^i_0, \ldots, z^i_m$ the corresponding primitive outputs. Then, we have that $C^i_j = M^i_j \oplus z^i_j$ for $1 \geq j \leq m$ and $C^i_0 \oplus T^i = z^i_0 \oplus T^i$. We also introduce the notion of the longest common prefix, which will be widely used in the following proof:

Longest Common Prefix For two arbitrary strings X and Y, let $llcp_n$ denote the number of n-bit blocks that are a prefix of both the strings, i.e., $\mathsf{llcp}_n(X,Y) = \max_{0 \leq i \leq \min(|X|/n, |Y|/n)} \{i \mid \mathsf{left}_{i \cdot n}(X) = \mathsf{left}_{i \cdot n}(Y)\}$ We can also extend the above notion to multiple strings, i.e., $\mathsf{llcp}_n(X_1, \ldots, X_m; Y) = \max\{\mathsf{llcp}_n(X_1, Y), \ldots, \mathsf{llcp}_n(X_m, Y_m)\}$

CBC collisions. Since collision may either happen in the CBC-MAC or between a CBC-MAC and a CTR, we need to study the probability on a case by case basis. Besides the bad flag that was discussed above, we also defined an auxiliary coll-bad, which as can be seen in Figure 3.6,

is set in case of collision of two CBC-MAC inputs that occurs after the index of the largest common prefix. The variable $\mathsf{llcp}_n i$ is used precisely to capture the notion of the largest common prefix between the input blocks of the CBC-MAC in the current query i and the CBC-MAC inputs of the previous $i-1$ queries. Furthermore, we should note that it makes no difference if the query is an encryption or decryption query, because in case of a decryption query, the adversary may still use the notion of the longest common prefix, with the plaintext blocks replaced by the corresponding ciphertext blocks(without the stretch), i.e., $\mathsf{llcp}_n i = \mathsf{llcp}_n(\bar{B}^1, \ldots, \bar{B}^{i-1}; \bar{B}^i)$ where $\bar{B}^r = \mathsf{encB}_0(N^r, A^r, \tau_r, \mathsf{left}_{|C^r|-\tau}(C^r)) \| \mathsf{encA}(A) \| \mathsf{encM}(\mathsf{left}_{|C^r|-\tau}(C^r)) - 1$ for $1 \leq r \leq i$. This construction can be used because in case $(N^i, \tau^i, |C^i|) = (N^j, \tau^j, |C^j|)$ and we use the same associated data in both queries, i.e., $A^i = A^j$ for $i \neq j$, then $\mathsf{encB}_0(N^i, A^i, \tau_i \mathsf{left}_{|C^i|-\tau_i}(C^i)) = \mathsf{encB}_0(N^j, A^j, \tau_j, \mathsf{left}_{|C^j|-\tau_j})$, $\mathsf{encA}(A^i) = \mathsf{encA}(A^j)$ since we assumed that the associated data was the same in both cases, and $\mathsf{encM}(\mathsf{left}_{|C^i|-\tau_i}(C^i)) = \mathsf{encM}(\mathsf{left}_{|C^j|-\tau_j}(C^j)) \iff \mathsf{encM}(M^i) = \mathsf{encM}(M^j)$ since when taking the XOR of the encrypted counter blocks with the message blocks in the CTR mode, the prefix is preserved. Furthermore, using a simple probabilistic trick we also obtain the following inequality:

$$\Pr[\mathscr{A}^{\boldsymbol{G_0}} \text{ sets } \mathsf{bad}] \leq \Pr[\mathscr{A}^{\boldsymbol{G_0}} \text{ sets } \mathsf{coll\text{-}bad}] + \Pr[\mathscr{A}^{\boldsymbol{G_0}} \text{ sets } \mathsf{bad} | \neg \mathscr{A}^{\boldsymbol{G_0}} \text{ sets } \mathsf{coll\text{-}bad}]$$

We can bound $\Pr[\mathscr{A}^{\boldsymbol{G_0}} \text{sets} \mathsf{coll\text{-}bad}]$ by a simple inductive analysis by assuming that when a block is being processed $\mathsf{coll\text{-}bad}$ is not already set to true. We also note that $\mathsf{coll\text{-}bad}$ can be set to true only when sampling y^i_j, i.e., the output of the j^{th} random function call in the i^{th} query. We first proceed with the analysis of the first event, i.e., $\mathscr{A}^{\boldsymbol{G_0}}$ sets $\mathsf{coll\text{-}bad}$. For the ease of analysis, we classify the inputs to the random function into three types: $\mathsf{type1}$ if it has been as the initial CBC-MAC input block B_0, $\mathsf{type2}$ if it has been added during the intermediate CBC-MAC computations, i.e., it is equal to $Y^i_{j-1} \oplus B^{i\prime}_j$ for some $j \leq l$ and $i' < i$, and of $\mathsf{type3}$ if it is a counter block, i.e., it is equal to $\mathsf{ctr}(N^{i'}, \tau_{i'}, j)$ for $i' < i$. An induction analysis requires some base cases, and in this case there are three base cases for the value of j:

Case 1: $j = 0$. Note that when processing the j^{th} block $\mathsf{llcp}_n i$ cannot be greater than j, and in this case $j = 0$ so the probability of collision with a $\mathsf{type1}$ input is 0. The same probability also holds for the event of collision with $\mathsf{type3}$ inputs, because of a domain separation bit in the first byte of B_0. On the other hand, collision with $\mathsf{type2}$ inputs can happen since $Y^{i'}_j$ are randomly distributed variable, so the probability of such collision is $1/2^n$.

Case 2: $j = \mathsf{llcp}_n(i) + 1 > 0$. In this case, there is a collision with a $\mathsf{type1}$ input if $Y^i_{j-1} = B^i_j \oplus B^{i'}_0$ for $i' < i$. By the induction hypothesis, $\mathsf{coll\text{-}bad}$ is still false, and this implies that that Y_{j-1} is independent of all the variables seen so far by the adversary. Hence, the probability of collision is $1/2^n$. Same reasoning also applies for the collision with $\mathsf{type3}$ inputs. Collision in this case implies that $Y^i_{j-1} \oplus B^i_j = Y^{i'}_{j-1} \oplus B^{i'}_j$. If i' is such that $\mathsf{llcp}_n(B^i, B^{i'}) = \mathsf{llcp}_n(i)$, this would imply that $Y^i_{j-1} = Y^{i'}_{j-1}$. Since $j = \mathsf{llcp}_n(i) + 1 \implies \mathsf{llcp}_n(i) = j - 1 \implies B^i_j \neq B^{i'}_j$. Hence, we would have that $B^i_j \neq B^{i'}_j \wedge Y^i_{j-1} Y^{i'}_{j-1}$ but $B^i_j \oplus Y^i_{j-1} = B^{i'}_j \oplus Y^{i'}_{j-1}$ which is clearly a contradiction. Hence, the probability of collision in this case is 0. In case $j \neq \mathsf{llcp}_n(i) + 1$, then the variables Y^i_j and $Y^{i'}_j$ are statistically independent, so the probability of collision is $1/2^n$.

Case 3: $j < \mathsf{llcp}_n(i) + 1$. Collision with $\mathsf{type1}$ variables in this case would imply that $Y^i_{j-1} = B^{i'}_0 \oplus B^i_j$ for some $i' < i$, which by induction hypothesis happens with probability $1/2^n$. Same reasoning also

applies for collision with type3 inputs. Since $j < \mathsf{llcp}_n(i)$, collision with type2 inputs only depends on the values of Y, which by induction hypothesis are randomly distributed, so the probability of collision is $1/2^n$.
From **??**, we know that the total number of calls to the random function is $2\sigma+3q$, so the probability of collision in all the cases is bounded by $\frac{(2\sigma+3q)^2}{2^{n+1}}$.

Other Collisions. From Figure 3.6, we observe that the flag bad can be set to true in four cases: when sampling a value for y^i_j in an encryption query, when sampling a value for y^i_j in a decryption query, when sampling a value for z^i_j in an encryption query, or when sampling a value for z^i_j in a decryption query. Let $y_{bad}(i,j)$ denote the occurrence of the first two events, i.e., bad is set to true when sampling the variable y^i_j during the i^{th} encryption/decryption query, and $z_{bad}(i,j)$ denote the occurrence of the last two events, i.e., bad is set to true when sampling the variable z^i_j during the i^{th} encryption/decryption query. Let E denote the event that the adversary $\mathscr{A}$ sets coll-bad to true. We note that if $1 \le j \le j$ $\Pr[y_{bad}(i,j)|\neg E] = 0$, because coll-bad is never set to true if $j \le \mathsf{llcp}_n(i)$, and if $j > \mathsf{llcp}_n(i)$, we are guaranteed that $B_j \oplus Y_{j-1} \notin Dom(f)$.
To analyze the event $z_{bad}(i,j)$, we proceed with a case by case analysis similar to the previous section:
Collision with type1 inputs occur with probability 0 because of the domain separation bit in the first byte of B_0. The same probability also holds for the collision with type3 inputs $\mathsf{ctr}(N^i,\tau_i,j) \neq \mathsf{ctr}(N^{i'},\tau_{i'},j')$ for $i \neq i'$. In case of collision with type2 inputs, we would have $\mathsf{ctr}(N^i,\tau_i,j) = Y^{i'}_{j'-1} \oplus B^{i'}_{j'}$. This is equivalent to $Y^{i'}_{j'-1} = \mathsf{ctr}(N^i,\tau_i,j) \oplus B^{i'}_{j'}$. Since we're conditioning on coll-bad = false, this implies that $Y^{i'}_{j'-1}$ is statistically independent from the outputs that are already observed by the adversary. Hence, the probability of collision is $1/2^n$.
From the above discussion, and from the fact that the adversary resources are bounded by $2\sigma + 3q$ as discussed in **??**, the following inequality holds:

$$\Pr[\mathscr{A}^{\mathbf{G_0}} \text{ sets } \mathsf{bad}|\neg E] \le \frac{(2\sigma+3q)^2}{2^{n+1}}$$

□

Combining the above bounds with Equation 3.2, we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathbf{nvae}[\tau_c]}_{\mathbf{vCCM}[E,\mathcal{I}_T,\nu]}(t,\mathbf{q_e},\mathbf{q_d},\boldsymbol{\sigma}_\mathbf{e},\boldsymbol{\sigma}_\mathbf{d}) \le & 7\cdot\mathbf{Adv}^{\mathbf{prp}}_{\mathrm{E}}(t',\sigma') + 3\cdot\frac{(2\sigma+3q)^2}{2^n} \\ & + \frac{q_d}{2^\tau} + \frac{(2\sigma^{\tau_c}+3q^{\tau_c})^2}{2^{n+1}} + \frac{(2\sigma_e^{\tau_c}+3q_e^{\tau_c})^2}{2^{n+1}}\end{aligned}$$

which is exactly the bound stated in Theorem 6.

Chapter 4

Implementation and Evaluation

This chapter seeks to describe the experimental setup used to benchmark CCM, GCM and Ascon while justifying the design choices along the way. Furthermore, we focus on providing an experimental validation of the energy efficiency of the variable tag transformations introduced in section 3.1 and section 2.4.

Apache Mynewt and Nimble The experiment described in the next section is implemented on nimble, which is an open-source implementation of Bluetooth Low Energy, built on top of Apache Mynewt. Apache Mynewt is an open-source operating system for embedded systems geared towards applications where power optimization is of high importance, and this is one of the main reasons why we picked such a platform to develop our experiments. Furthermore, it also offers the desired functionalities that are needed for our application, such as precise hardware and software timers and efficient cryptographic libraries like mbedtls and tinycrypt. The program follows the standard central-peripheral architecture, where the peripheral advertises itself, while the central scans for devices and initiates a connection. After the connection is initiated, the central is referred to as the master and the peripheral as the slave. The two devices can now communicate with one another by following the reference framework provided by the Generic Attribute Profile. GATT also establishes the existence of a server, which stores the metadata about all the services and characteristics, and the GATT client which can read or write from the characteristics.

GAP Peripheral/GATT Central First, we note the difference between the GATT and GAP roles, which are unrelated to one another. GAP determines the role of a device before the connection is established, i.e., whether the device will advertise itself in case it is a peripheral, or it will scan for other devices in case it is a central device. GATT on the other hand, determines the role of the device after the connection is established, and it specifies whether the device will be used as a server to store services and characteristics, or whether it will be used as a peripheral. We first need to assign a static address to the peripheral in order to ensure that the central connects with

the correct device. Furthermore, the peripheral will keep advertising and an event handler will be used to detect a change in the status of the connection. The two devices use the Just Works pairing mode because of a lack of input/output capabilities. The GATT central also creates all the necessary services and characteristics for the peripheral to write to. When creating a characteristic, the server can also set a flag to determine whether the data written to the characteristic needs to be encrypted. Since we're disabling the encryption in order to modify the standard **CCM**, we specify no encryption requirements for the characteristic's data. When the peripheral receives the encrypted data, it decrypts it using the same parameters that were used for the encryption by the GATT central, and ensures that the encryption algorithm works correctly by comparing the obtained plaintext with the plaintext from the test vectors.

GAP Central/GATT Peripheral The GAP central scans for advertising packets and connects with the device if its MAC address corresponds to the static address of the peripheral. It also encrypts the data by using one of the three specified encryption modes: CCM, GCM or Ascon determined by a compiler directive. CCM or Ascon can either be used in their standard forms with constant tag lengths, or using our black-box variable tag transform. As discussed in section 1.2, GCM is different from the other two schemes in that it doesn't follow the sponge construction, so our transform cannot be proven secure in such a scheme. Two libraries were used for the implementation of the cryptographic protocols, mbedtls and . They both provide efficient implementations of the tested cryptographic schemes, so the differences in power consumption between the two libraries were negligible.

Crypto Implementations For the CCM and GCM tests, we simply made use of the already implemented functions provided by mbedtls and tinycrypt. For the vCCM implementation, we defined an API that takes the same arguments as the standard CCM, encodes the tag length in the last byte of the nonce and calls the standard CCM encryption/decryption functions provided by mbedtls and tinycrypt. This can be easily done since the CCM standard defines a range of tag lengths that the algorithm supports. However, in case of Ascon, the inner workings of the algorithm had to be modified since the standard only supports a tag length of 16 bytes. In order to support other tag lengths, we needed to substitute the XOR operation in the Ascon S-box implementation illustrated in Figure 4.1 with a truncated XOR, which only keeps τ most significant bytes, where τ is the tag length of the scheme. The truncated XOR is only applied to the x_3 and x_4 parts of the S-box illustrated in Figure 4.1, while the x_1 and x_2 parts are kept unchanged.

4.1 Experimental Validation of Energy Efficiency

In many applications relying on battery-operated low-power devices, (such as wireless sensor networks), energy is among the most critical resources. In this section, we experimentally validate that the flexible trade-off between security level and communication overhead enabled by vCCM and

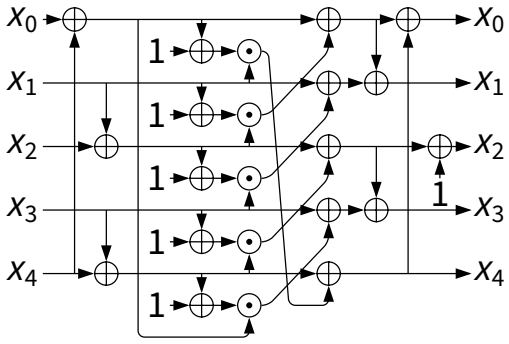


Figure 4.1 – A nonlinear 5-bit substitution layer applied 64-times during a permutation.

vAscon, two nvAE schemes, does indeed translate to significant energy savings. Our experiments and projections are based on a communication scenario where one device always acts as a transmitter and another device as a receiver. The sender alternates between using CCM, GCM, vCCM and vAscon for the encryption and sends the encrypted data using a 2.4 GHz Bluetooth transceiver.

Communication scenario In our scenario, the sender transmits two types of messages, "non-critical" and "critical", periodically. The "non-critical" messages are sent frequently and are assumed to require a lower level of protection. The "critical" messages are sent sporadically and are assumed to require a higher level of security. This setup corresponds to numerous real world scenarios where certain sensors are used to perform some measurements (temperature, leakage etc.), which in this case is a non-critical information and report a shutdown in case of a failure, which is critical information, because the impact of data corruption would be quite significant. For the communication devices, we used two nRF52840 development boards, which interact with a Power Profiler Kit II that is used to measure the power trace during the communication. The block diagram below, which was taken from the Power Profiler 2 user guide [19] illustrates the measurement pipeline and the connections between all the blocks included in the measurement process. The Power Profiler kit was chosen because of its low price and availability. However, the presence of filtering capacitors in nRF52840 development board and the low sampling frequency of the measurement device(10 kHz while the sampling frequency of a low-end oscilloscope is around 1 MHz) resulted in smoothed out current variations and a smaller difference in the power consumption of different tag lengths. Hence, repeating the same measurements with a high precision oscilloscope would in principle yield more precise results and higher gains than presented in Table 4.2. The experimental setup is illustrated in Figure 4.3.

Measurements Several repetitions of the measurements were carried, where each measurement seeks to capture the current consumption of the sender that repeatedly encrypts and sends a payload every 10 seconds. The measurements were done for all the possible (plaintext length, tag length) combination, where the plaintext is 4 bytes long and the tag length takes the values $\{4, 8, 16\}$ bytes. The short plaintext length was chosen so that the tag length dominates the overall

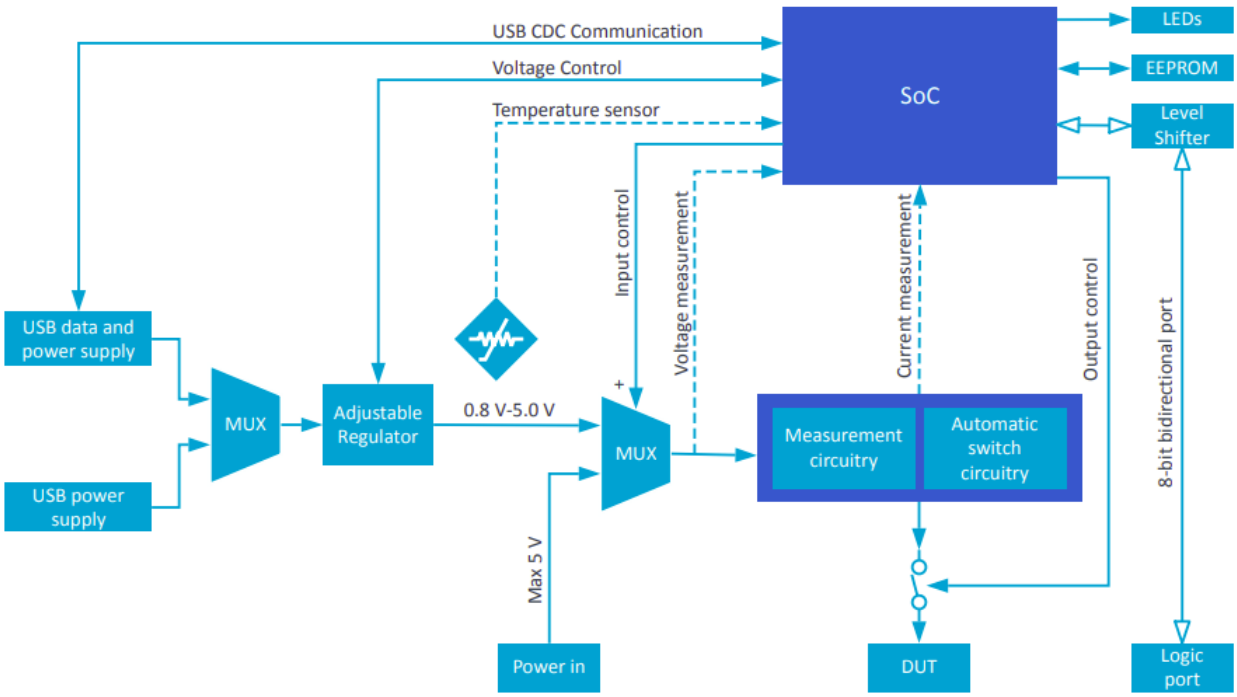


Figure 4.2 – Power Measurements Block Diagram

payload length, which increases the influence of a change in the tag length over the total energy consumption. For each experiment, the power consumption of the standard scheme was compared against the power consumption of the variable-tag transformation of the scheme. The setup for the experiments are summarized in Figure 4.1.

AE Scheme	Message Length(bytes)	Tag Length(bytes)
CCM	4	4,8,16
GCM	4	4,8,16
vCCM	4	4-8,4-16
Ascon	4	16
vAscon	4	4-8,4-16

Table 4.1 – List of the experiments' setup. The x-y values in the second column denote the tag values for the non-priority and priority messages, respectively.

Data Processing The data are exported from the power profiler software nrfconnect361x8664 in the csv format. As the exported data does not contain the values of the digital channels D0-D9, which are used to delimit the occurrence of the transmission events, we had to resort to a statistical approach to isolate the transmission peaks. First, we compute the average current consumption

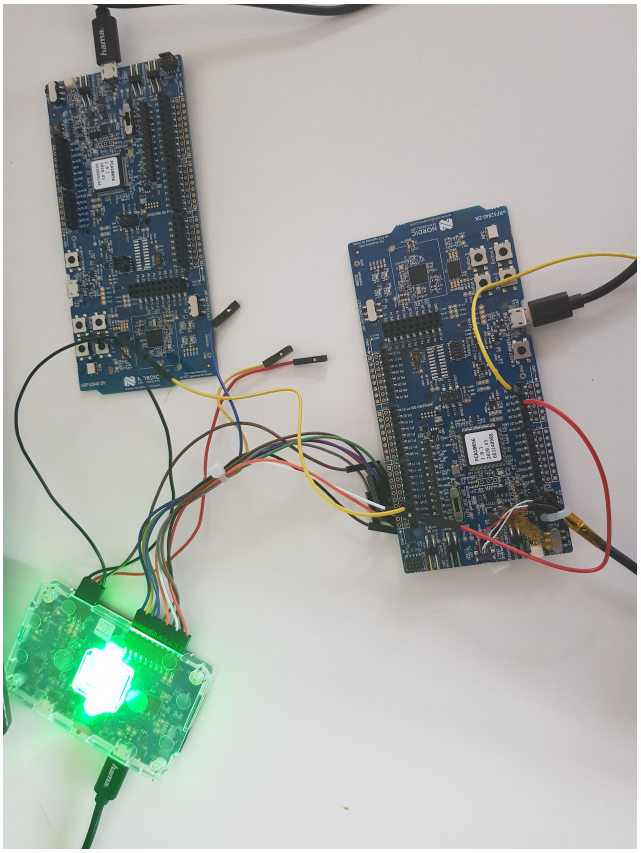


Figure 4.3 – Measurements setup used to measure the energy consumption. Two **nRF52840 DK** are used to exchange data with one another and a power profiler kit is used to measure the energy consumption of the sender

over all the sampled data points using the formula:

$$\mu_E = \frac{1}{|E|} \sum_{i=1}^{|E|} E_i$$

where μ_E denotes the average energy consumption, E_i denote the energy consumption of the i^{th} data point and $|E|$ denotes the number of samples. A sample is considered to be in the transmission interval if its energy consumption is not in the interval $[\mu_E - 4 \cdot \sigma, \mu_E + 4 \cdot \sigma]$.

Results As the major advantage of the variable tag transformation is the bandwidth saving, we first tried to capture the relation between the payload length and the energy consumption in a setup where two devices are sending unencrypted data of length $\{20, 30, 50, 100, 200\}$ bytes via BLE, with the results illustrated in Figure 4.4. Because the power consumption is dominated by the transmission process as illustrated in Figure 2, we would expect similar gaining when using the variable tag transformation. As can be observed from the above figure, the energy

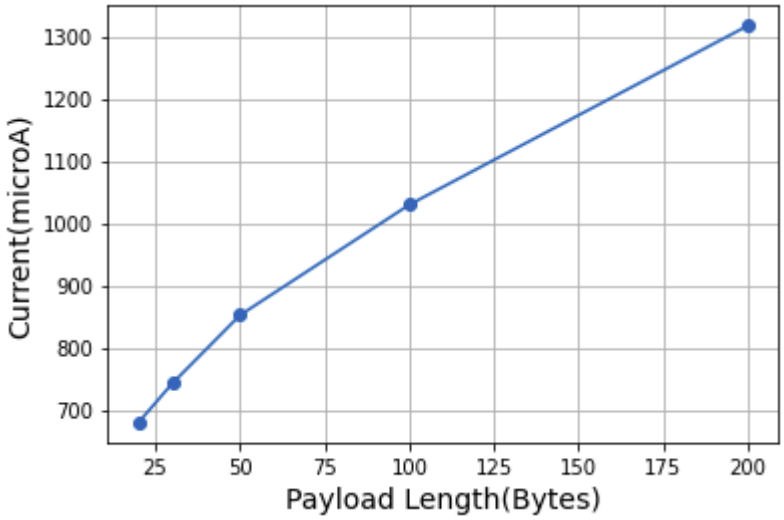


Figure 4.4 – Payload Length(Bytes) vs Current Consumption(microA)

consumption is almost linearly dependent on the payload length. Hence, when we use the variable tag transformation in the 8-byte and 16-byte tag length cases, we expect a saving in bandwidth of 4 bytes and 12 bytes respectively. From Figure 4.4, we observe that the difference in energy consumption is $\approx 10\%$ for a difference in payload length of 10 bytes, and it increases to $\approx 15\%$ for a difference in payload length of 20 bytes. Hence, we expect our savings to be $\approx$10% for 8-byte tags and in the range 10-15% for the 16-byte tags. The final results are displayed in Table 4.2, adn Figures 4.5 and 4.6.

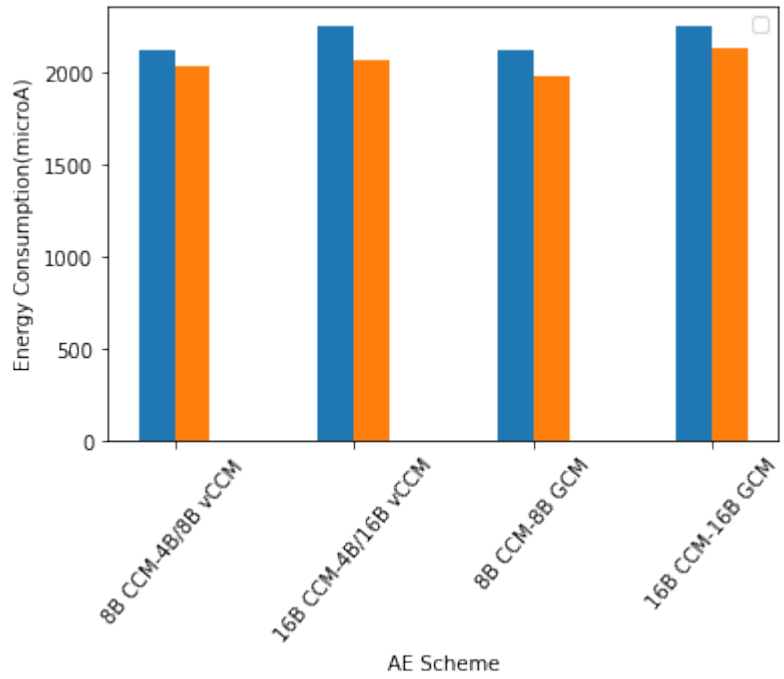


Figure 4.5 – Payload Length(Bytes) vs Current Consumption(microA)

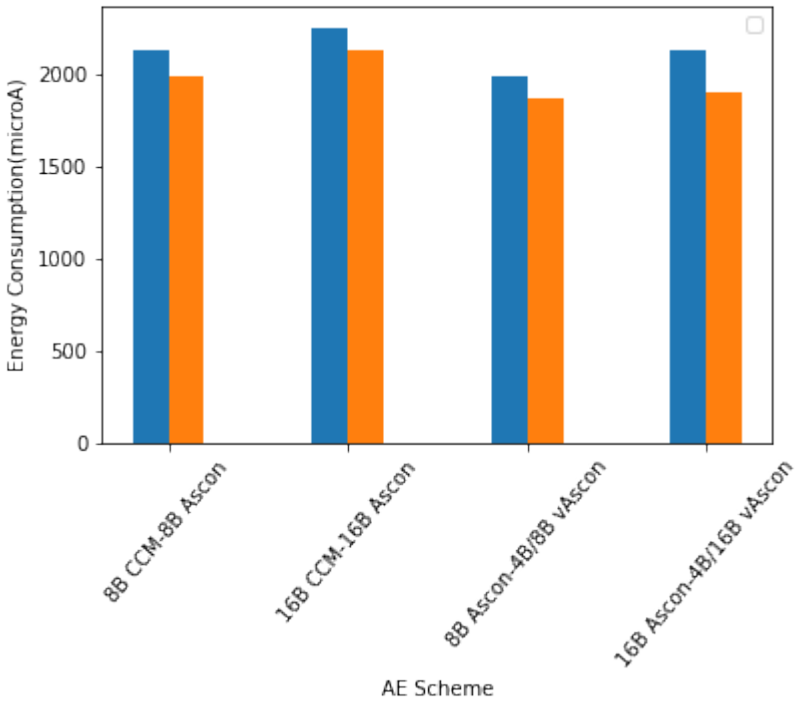


Figure 4.6 – Payload Length(Bytes) vs Current Consumption(microA)

	Energy Savings	Tag Length (Non-Critical/Critical)	Power Consumption(W) x 10^{-6}
(1)	5% less than (3)	4B/8B Vartag CCM	2032.5
(2)	9% less than (4)	4B/16B Vartag CCM	2063.21
(3)	-	8B standard CCM	2125.3
(4)	-	16B standard CCM	2250.3
(5)	6% less than (7)	4B/8B Vartag Ascon	1863.2
(6)	11% less than (8)	4B/16B Vartag Ascon	1893.9
(7)	7% less than (3)	8B standard Ascon	1982.76
(8)	6% less than (4)	16B standard Ascon	2128.98
(9)	2.5% less than (3)	8B GCM	2070.57
(10)	2.3% less than (4)	16B GCM	2198.43

Table 4.2 – Average power consumption

From the table, we observe that we save from 5% to 11% in power consumption using the variable tag transformation. This is much higher than the 2.5% power saving we obtain when using GCM instead of CCM, and even higher than the 7% saving we obtain when using Ascon, which is a lightweight encryption schemes designed specifically for use in embedded systems. These results suggest that in a communication scenario where two IoT devices exchange short messages at high frequencies, implementing the variable tag transformation is a simple and powerful approach to increase the power efficiency of the devices.

Chapter 5

Related Work

5.1 Related Work

Variable-Tag Black-box Transform Similar black-box transformations for existing cryptographic primitives have been studied in several other works. In [21], Reyhanitbar, Vaudenay and Vizar implement a similar transformation to OCB, which is a tweakable blockcipher, by encoding the tag length in the tweak and proving the security of such transformation. Furthermore, they suggest that in sponge-like constructions, encoding the tag length in the nonce input is a secure transformation to enable the use of variable tag lengths in such schemes. Because of the minor modifications that are required to implement vOCB, the authors conclude that the transformation will not significantly impact the performance of OCB. However, despite the conclusion being very reasonable because of the minor modifications that are introduced to the standard OCB, an empirical analysis is missing from the paper. In [17] Mennink et al. provide a security proof of the full-state keyed Sponge and full-state Duplex constructions, which can be used to make many existing sponge-based authenticated encryption schemes more efficient by doing the absorption of the associated data blocks and message blocks concurrently. They further provide an algorithm to reduce a call to an authenticated encryption scheme to a call to a duplexing structure with the proven security bounds. Given such reduction, one can show that each permutation output in a sponge-based AE scheme is equivalent to a PRF image of all the input blocks processed so far. Hence, if the tag value is encoded in the early blocks, the variable tag transformation should in principle be secure.

Benchmarking Lightweight Cryptography Schemes Lightweight Cryptography encapsulates a set of encryption schemes that feature small footprints and low computational complexity, which make them a perfect fit for the constrained world of embedded systems. With the increasing use of IoT devices, NIST received a total of 57 candidates in the first round, only 10 of which managed to get through to the final round. The first round candidates have been benchmarked on

different MCU platforms and architectures by Renner et al., who test the speed, ROM and RAM use, and present the results in [20]. In these benchmarks, the Ascon scheme is ranked among the fastest and most efficient energy-wise, which is the main reason why we decided to benchmark this scheme among the other recent NIST LWC candidates. Similar work has also been done by Hyncica et al. in [11] where they evaluate 15 symmetric primitives using three criteria: throughput, code size and storage utilization.In [7], Dinu et al. also introduce a fair benchmarking framework for lightweight cryptographic systems which offers the possibility of measuring RAM, the code size and speed of 19 optimized cipher implementations. There are also several other papers that focus on providing a benchmark for testing different cryptographic primitives, but most of them focus on the RAM/ROM consumption or the execution time, while the power consumption, despite being a critical factor in these systems, is mostly ignored.

Benchmarking BLE Power Consumption Although Bluetooth Low Energy was introduced as part of the Bluetooth core specification version 4.0 more than a decade ago, there seems to be a scarcity of papers geared towards analyzing its power efficiency. In [9], Fürst et al. benchmark the Bluetooth implementation on a set of chips and smartphones and compare the advertisement packets latency and reception ratio, the received signal strength indication, and the read/write latency for an arbitrary characteristic. In [15], Kindt et al. perform an energy modeling of BLE in all its modes of operations. They also vary several parameters of the protocol and estimate their impact on the energy consumption. Among the parameters that the authors decided to test are: the interframe space, which determines the waiting time of the advertiser after the advertising packet is sent, or the slave latency, which determines the number of connection events a slave may skip without having to wake up etc. In [5], Cho et al. focus solely on the BLE discovery process by analyzing the influence of several parameter settings on the discovery latency and the energy consumption. For their analysis, the authors follow a probabilistic approach, where they consider several scenarios by varying the number of advertisers or scanners and the synchronicity between them. While the energy benefits of BLE are widely studied and documented for several SOCs and implementations, our work is focused on benchmarking an open source implementation of BLE in conjunction with several encryption protocols, which are essential for the secure communication between two or more devices.

Conclusion

5.2 Future Work

Measurements Repetition with High Precision Equipments The Power Profiler II kit was used because of its cheap price and ease of accessibility. However, the device has a sampling rate of only 10 KHz as opposed to the 1 MHz sampling rate of a high precision oscilloscope. Hence, in order to obtain more reliable and better results, the power consumption measurements need to be performed via an oscilloscope. Furthermore, the nRF52840 development board contains some filtering capacitors, which result in a smoother power trace. Hence, the power consumption spikes are not as sharp as they should be, which results in a lower difference in power consumption between different tag length. Performing the measurements in a more optimized board would in principle produce better results.

Security analysis of the NIST LWC Schemes NIST Lightweight Cryptography conference only started a few years ago, so most of the schemes proposed are quite novel and not extensively analyzed in terms of their security guarantees. Hence, a security analysis of these schemes, similar to the proof we provide for the security of Ascon, would be quite beneficial for embedded system engineers who can choose between the available schemes depending on the desired security level.

Porting the Variable Tag Transformation to Other Communication Protocols The variable tag transformation suggested works better for short messages, where the tag length occupies a large part of the payload. Even though the experiments in chapter 4 were conducted with a short message size of only 4 bytes, the BLE has a high overhead, and it appends 8 extra bytes to each packet with 1-2 bytes appended for frequency synchronization, 4 bytes to store the address of the other device and an extra 3 bytes for the cyclic redundancy check. Hence, the overhead is quite large considering that the tag size is 4,8 or 16 bytes, which reduces the impact of tag length in the power consumption. In [10], Gjiriti, Vizár and Reyhanitabar showed that the power savings when using the LoRa protocol, which is a much simpler protocol than BLE, with almost no overhead during packet transmission, were from 10-20%. Hence, in order to better quantify the power savings of the transformation, several other low-overhead protocols(6LowPAN, LPWAN etc.)

could be tested.

5.3 Conclusion

To conclude, we propose the variable tag transformation for the CCM and Ascon authenticated encryption schemes to improve the power consumption and the security of BLE. We further test our transformation in an experimental setup, with two nRF52840 development boards communicating via BLE and alternating between the standard encryption schemes and the schemes endowed with our variable tag transformation. We found a difference in power consumption between 5-11%, which although in absolute terms may not be very significant, it is still worth implementing since the transformation is very easy to implement, and the savings are even greater in communication protocols with a lower overhead, as evidenced by the even larger power savings(around twice higher) in [10], which benchmarked a similar transformation in the much simpler LoRa protocol.

Bibliography

[1] Mihir Bellare and Phillip Rogaway. "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs". In: *EUROCRYPT 2006, Proceedings*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, 2006, pp. 409–426.

[2] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications*. Selected Areas in Cryptography 2011. 2011.

[3] *BLE Tutorial*. https://www.novelbits.io/basics-bluetooth-low-energy/. Accessed: 2021-06-24.

[4] *Bluetooth Market*. https://www.bluetooth.com/wp-content/uploads/2019/03/Bluetooth_Market_Update_2018.pdf. Accessed: 2021-05-20.

[5] Keuchul Cho, Gisu Park, Wooseong Cho, Jihun Seo, and Kijun Han. "Performance analysis of device discovery of Bluetooth Low Energy (BLE) networks". In: *Computer Communications* 81 (2016), pp. 72–85. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2015.10.008. URL: https://www.sciencedirect.com/science/article/pii/S0140366415003886.

[6] *Counter with CBC-MAC (CCM)*. https://www.rfc-editor.org/rfc/pdfrfc/rfc3610.txt.pdf. Accessed: 2021-05-20.

[7] Daniel Dinu, Yann Le Corre, Dmitry Khovratovich, Léo Perrin, Johann Großschädl, and Alex Biryukov. "Triathlon of Lightweight Block Ciphers for the Internet of Things". In: IACR. 2015.

[8] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *Ascon v1.2*. NIST Lightweight Cryptography. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/ascon-spec-round2.pdf. 2019.

[9] Jonathan Fürst, Kaifei Chen, Hyung-sin Kim, and Philippe Bonnet. "Evaluating Bluetooth Low Energy for IoT". In: Apr. 2018. DOI: 10.1109/CPSBench.2018.00007.

[10] Emiljano Gjiriti, Reza Reyhanitabar, and Damian Vizár. "Power Yoga: Variable-Stretch Security of CCM for Energy-Efficient Lightweight IoT". In: 2021, pp. 446–468. DOI: https://doi.org/10.46586/tosc.v2021.i2.446-46.

[11] Ondrej Hyncica, Pavel Kucera, Petr Honzik, and Petr Fiedler. “Performance Evaluation of Symmetric Cryptography in Embedded Systems”. In: IEEE. 2011.

[12] Jakob Jonsson. “On the Security of CTR + CBC-MAC”. In: *SAC 2002*. Ed. by Kaisa Nyberg and Howard M. Heys. Vol. 2595. LNCS. Springer, 2002, pp. 76–93.

[13] Philipp Jovanovic, Atul Luykx, and Bart Mennink. *Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes*. Advances in Cryptology – ASIACRYPT 2014. 214.

[14] Khoongming Khoo, Serge Vaudenay, Thomas Peyrin, and Siang Meng Sim. “Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. LNCS. 2004, pp. 16–31.

[15] Philipp H. Kindt, Daniel Yunge, Robert Diemer, and SAMARJIT CHAKRABORTY. “Energy Modeling for the Bluetooth Low Energy Protocol”. In: 2020. DOI: https://doi.org/10.1145/3379339.

[16] David A. McGrew and John Viega. “The Security and Performance of the Galois/Counter Mode (GCM) of Operation”. In: *INDOCRYPT 2004*. LNCS. 2004, pp. 343–355.

[17] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. *Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption*. Advances in Cryptology – ASIACRYPT 2015. 2015.

[18] Yuichi Niwa, Keisuke Ohashi, Kazuhiko Minematsu, and Tetsu Iwata. *GCM Security Bounds Reconsidered*. Cryptology ePrint Archive, Report 2015/214. https://eprint.iacr.org/2015/214.pdf. 2015.

[19] *Power Profiler Kit 2 User Guide*. https://infocenter.nordicsemi.com/pdf/PPK_2_User_Guide_20201201.pdf. Accessed: 2021-06-23.

[20] Sebastian Renner, Enrico Pozzobon, and Jurgen Mottok. “Benchmarking Software Implementation of 1st Round Candidates of the NIST LWC Project on Microcontrollers”. In: NIST. 2019.

[21] Reza Reyhanitabar, Serge Vaudenay, and Damian Vizár. “Authenticated Encryption with Variable Stretch”. In: *ASIACRYPT 2016, Proceedings, Part I*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 396–425.

[22] Shreyas Sen, Jinkyu Koo, and Saurabh Bagchi. “TRIFECTA: Security, Energy Efficiency, and Communication Capacity Comparison for Wireless IoT Devices”. In: *IEEE Internet Computing* 22.1 (2018), pp. 74–81. DOI: 10.1109/MIC.2018.011581520.