SimpleENC and SimpleENCsmall – an Authenticated Encryption Mode for the Lightweight Setting[⋆]

Shay Gueron[1] and Yehuda Lindell[2]

[1] University of Haifa, Israel, and Amazon, USA
[2] Bar Ilan University, Israel, and Unbound Tech Ltd., Israel

Abstract. Block cipher modes of operation provide a way to securely encrypt using a block cipher, and different modes of operation achieve different tradeoffs of security, performance and simplicity. In this paper, we present a new authenticated encryption scheme that is designed for the lightweight cryptography setting, but can be used in standard settings as well. Our mode of encryption is extremely simple, requiring only a single block cipher primitive (in forward direction) and minimal padding, and supports streaming (online encryption). In addition, our mode achieves very strong security bounds, and can even provide good security when the block size is just 64 bits. As such, it is highly suitable for lightweight settings, where the lifetime of the key and/or overall amount encrypted may be high. Our new scheme can be seen as an improved version of CCM that supports streaming, and provides much better bounds.

1 Introduction

1.1 Background and the Challenge

Block ciphers are a basic building block in encryption. Modes of operation are ways of using block ciphers in order to obtain secure encryption, and have been studied for decades. Nevertheless, new computing settings and threats make the design of new and better modes of operation a very active field of research. For just one example, NIST has recently initiated a competition for a mode of operation that is suited for lightweight ciphers [18]. This has unique challenges due to the need that the mode be both *simple* and provide high security in multiple settings.

The gold standard for encryption security is *authenticated encryption with additional data* (AEAD), ensuring both privacy and integrity. In this paper, we construct an AEAD mode of operation that is suitable for the lightweight setting (and others).

The challenge – many messages or long messages with both unique nonce or random IV.[3] In many settings, the nonce can be guaranteed to

[⋆] This paper contains the full security analysis for the round-1 candidate to the NIST Lightweight Cryptography standardization process named “SIMPLE”; see [19].

[3] Throughout, we use the term *nonce* for a value that is guaranteed to be unique, and use the term IV when it is randomly chosen.

be unique (e.g., by storing the previously used nonce and incrementing it for each encryption). In such cases, better bounds can be achieved than when the IV is chosen randomly and so repeats at the birthday bound. However, devices may be stateless, making it impossible to ensure a unique nonce. In such cases, random IVs must be used. As such, a good mode of operation should be able to work *both* for unique nonces and random IVs, and should provide better bounds in the former case. In order to see why this is not always trivial, consider the counter mode of operation. In this mode, the ith block is encrypted by masking it with the output of the block cipher applied to an encoding of the nonce and the index i. This encoding must ensure that the input to the block cipher is unique each time (with very high probability). As such, the standard way of doing this is to make the nonce smaller than the block size, and to concatenate the binary representation of i in the remaining bits. For example, with a 128-bit block, the nonce can be 96 bits, and up to 2^{32} blocks can be encrypted, since 32 bits remain for the encoding of the block counters. As long as the nonces are unique, all inputs to the block cipher are also unique. We stress that one *cannot* take a nonce of 128-bits long, and simply increment it for each block (mod 2^{128}), since then unique nonces may result in overlapping inputs to the block cipher. For example, if the unique nonces are achieved by simply incrementing by one (and so in consecutive encryptions we have N and $N+1$), then the same input will be used to the block cipher, as $N+(i+1) \bmod 2^{128} = (N+1)+i \bmod 2^{128}$, resulting in security being completely broken. In contrast, if only random IVs are used, then with high probability such an overlap will not occur unless a large number of blocks (near the birthday bound) are encrypted.

As a result of the above, a counter-based mode of operation that must support nonce-based encryption is limited to the size of the nonce which cannot be too large, and this limits the number of encryptions when used with a random IV. For example, the standard nonce sizes for AES for CTR, AES-GCM and so on is 96 bits (to enable encrypting messages of up to length 2^{32} blocks). However, this means that when random IVs are used, an IV repeats with probability 2^{-32} already after encrypting just 2^{32} messages. The NIST recommendation for these modes is therefore to encrypt at most 2^{32} messages, which is a severe limitation in practice today.

Thus, there is a significant challenge in constructing a single scheme that meets the following requirements simultaneously:

1. It can be used both with unique nonces and random IVs (where uniqueness guarantees nothing about the format of the nonces, but just that they are different).
2. It can be used to encrypt many blocks in any configuration: from a small number of very large messages to a large number of small messages, and everything in between.

Simplicity and lightweight. Additional challenges arise since in order to make the mode of operation suitable for the lightweight setting, it must be very simple, in the following sense:

1. Only a single primitive should be required (e.g., a block cipher only, and in forward mode only).
2. The entire message need not be held in memory or processed more than once. Thus, streaming (online computation) needs to be supported. Preferably, the length of the overall message need not be known at the onset.
3. The code for implementing the mode should be small. Preferably, the code for decryption should be very similar to the code for encryption, reducing the code base.

1.2 Design Goals for our Lightweight Mode of Operation

NIST recently initiated a competition for standardizing a lightweight encryption method [18]. The competition covers both the underlying (stream or block) cipher, and the method of using the cipher to encrypt; i.e., the mode of operation. We consider only the mode of operation here, that can use any block cipher E with block size of 128 (and even 64) bits. The desired properties that are targeted are:

1. *Security:* The top priority design goal is the security achieved by the scheme. The mode is intended to rely only on the standard (and most basic) assumption on E, that modern block ciphers should be pseudorandom permutations when the keys are selected uniformly at random from the key space. In particular, we do not require additional properties or assumptions, like related-key security.
2. *Flexibility:* The mode should allow for encrypting a large number of blocks while preserving a high security margin. To be concrete, considering a 128-bit block, the goal is to be able to encrypt up to 2^{50} bytes (equivalently, 2^{46} blocks) in *any* configuration of number of messages and message length. Specifically (and per the requirements of the NIST call [18, Sec. 3.1]), security should be maintained when 2^{50} single-byte encryptions are carried out, or when encrypting a message of size 2^{50} bytes, and everything in between.
3. *Long lifetime for keys:* The maximal number of messages that can be encrypted with a single key is a significant concern in lightweight scenarios, where devices (communicating with a server) are deployed "in the field" and it could be extremely difficult to rotate keys. It is likely that messages emitted from the device are (very) short, but over time, a large number of such messages need to sent.
4. *Random nonces:* The mode should support a mode where uniquely-chosen nonces or randomly-chosen nonces are used. In particular, one should not require random nonces, but must allow them.
5. *Simplicity and frugality:* The mode should be simple to describe and also to implement, e.g., not involving a length block, or not requiring complex padding or multiple conditional branches. The mode should use only a small number of cryptographic primitives, preferably one.
6. *Online (streaming):* The mode should be "online", meaning that the lengths of the additional authentication data and/or message should not be needed at the onset.

1.3 The SimpleENC and SimpleENCsmall Modes of Operation

We present a new mode of operation (with variants) that is designed to meet the requirements described in Section 1.2, and can work with any block cipher. We have two main modes: one for a large block size (e.g., 128 bits) and one for a small block size (e.g., 64 bits). Unlike other existing modes, our mode achieves good security even for a small block size. Our mode of operation is extremely simple and can be used both for the lightweight and general setting, as it achieves a very high level of security. It can be seen as an improvement over AES-CCM, that is simpler, provides much better security bounds, and can be run in online mode.

Background – CCM. Before proceeding to describe our mode, we observe that CBC-MAC and Counter-Mode encryption are obvious choices for *simple* authentication and encryption. Indeed, the known authenticated-encryption scheme CCM [7,16,17] leverages exactly these primitives to gain its simplicity. Unfortunately, raw CBC-MAC is known to be insecure for messages of arbitrary length, unless it operates over a prefix-free set of inputs (or otherwise enhanced like by encrypting the tag with an additional key). Due to this, CCM uses a relatively complex padding to achieve the prefix-free property. For this reason, CCM is not an online mode and the lengths of the messages are required in advance. In addition, CCM cannot support the conflicting requirements of simultaneously supporting a few long messages and many short messages with unique and random nonces. On the one hand, one cannot safely use short randomized nonces because of the high probability that nonces would repeat. For example, in order to leave enough bits (in a block of size 128) to account for a message of length 2^{50} bytes (2^{46} blocks), the nonce can have at most 82 bits. However, this means that with 2^{-32} security, at most 2^{11} different messages can be encrypted.

Our construction paradigm. We base our construction on CBC-MAC and Counter-Mode encryption, due to their simplicity. However, we overcome the aforementioned limitations by having a long nonce as input, but deriving keys and a shorter nonce in each encryption. This is inspired by the continual key-derivation method of [9] but is different since we also derive a nonce. Since we derive both a nonce and encryption key, security is maintained as long as there is no simultaneous collision of the derived key and derived nonce. In addition, by XORing the derived nonce to the first block of the message in the CBC-MAC computation, we obtain effective prefix-freeness, enabling us to use plain CBC-MAC which is much simpler. In some sense, our mode of operation "breaks all the rules" of CBC-MAC:

1. We apply CBC-MAC to the plaintext and not the ciphertext (enabling the encryption and MAC portions to run independently),
2. We do not require any padding to ensure prefix-freeness, and
3. We do not encrypt the resulting MAC-tag.

Ordinarily, this would be completely insecure. However, by a small but novel modification, the above is secure. Namely, we XOR the (derived) nonce to the *first block* of the message before MACing it. This very simple change results in prefix-freeness with very high probability, as long as the nonce does not repeat. It also means that the tag need not be encrypted since the same message encrypted twice with different nonces will have different pseudorandom tags.

A high-level description of `SimpleENC`. The mode `SimpleENC` is designed for use with a "large" block size (at least 128-bits long). Denoting by n the size of the block and by κ this size of the key, we first consider the case that $\kappa = n$. The nonce size is $(n-\tau)$ for some $\tau \geq 2$, such that the strings $[N \parallel 0], [N \parallel 1], [N \parallel 2]$ fit into one block. Upon input (N, A, M) and key K, where N is a nonce, A is additional authentication data, and M is the message, the output C and *tag* are produced in three (logical) steps, namely *Derive*, *Encrypt* and *Authenticate*, as follows:[4]

1. *Derive:* Derive $2\kappa + n = 3n$ random bits to define K_E and K_M of length κ each, and two half blocks N_1, N_2 of length $n/2$ each. The derivation invokes E over $[N \parallel 0], [N \parallel 1], [N \parallel 2]$ using key K, in order to derive $3n$ bits.
2. *Encrypt:* Encrypt M in counter mode with the key K_E and initial counter N_1; let C be the result.
3. *Authenticate:* Compute CBC-MAC on the message $N_2 \tilde{\oplus} A \| M$ with the key K_M (where $N_2 \tilde{\oplus} A$ denotes the operation of XORing N_2 with the first block of A, and leave the rest of the blocks unchanged); let Tag be the result.[5]
4. *Output:* Output (C, Tag).

A variant of `SimpleENC`, that we call `SimpleENC`$'$ in the sequel, works for the case that $\kappa = 2n$ (e.g., for a block-size of 128 and a key of size 256, as with AES-256). The only difference is in the *derive* step since $2\kappa + n = 5n$, and thus 5 blocks of pseudorandomness are derived by invoking E over $[N \parallel 0], \ldots, [N \parallel 4]$ using key K. This also means that the nonce size is at most $(n - \tau)$ for some $\tau \geq 3$. From here on, we take $\tau = 8$ so that all operations are on bytes.

A high-level description of `SimpleENCsmall`. The mode `SimpleENCsmall` is designed for use with a "small" block size (e.g., of size 64-bits). For a typical case, consider the case that $\kappa = 2n$. Then, `SimpleENCsmall` works in the same way as `SimpleENC`$'$, except that instead of deriving the keys and nonces directly with the block cipher (pseudorandom permutation), we derive them using CENC [12,13].[6] CENC is an efficient way of constructing a pseudorandom function from a block cipher that provides security beyond the birthday bound. It is parameterised by a parameter w, and works by XORing the output of an additional block for each w blocks output. CENC has strong bounds, achieving security of $\frac{w\cdot\ell\cdot q}{2^n}$

[4] The Derive-Encrypt-Authenticate description is a logical description of *Simple*, which is useful for the simplicity of description and for analysis. An implementation can (and should) interleave the encryption and the authentication steps in order to, among other things, avoid reading the input from memory twice.

[5] Formally, A and M are each padded with 10^* first in order to achieve unambiguity.

[6] We remark that `SimpleENCsmall` in the NIST lightweight cryptography round-1 candidate [19] uses CENC also in the encryption phase, and not just for key derivation. However, we observe that this is not necessary and counter-mode encryption suffices.

where ℓ equals the number of blocks encrypted per message and q the number of messages (rather than $\frac{(q\cdot\ell)^2}{2^{n+1}}$ for a direct use of a permutation). This overcomes the birthday barrier on the number of different encryptions (since this is the number of different derivations), and allows us to achieve good bounds.

Efficiency and simplicity. The nonce-based derivation precedes encryption and authentication, and is an overhead paid toward key lifetime enhancement. In most cases, this overhead is small and insignificant, and is offset by the fact that length-encoding, encryption of the tag, and so on are not needed.

We observe that both counter mode and plain CBC-MAC are very simple to implement, and avoiding length encoding allows the mode to be an online (streaming) mode. In addition, decryption is almost identical to encryption, as can be seen in the formal description in Appendix B.

The choice of the block ciphers. `SimpleENC` and `SimpleENCsmall` can be used with any block cipher desired. Some examples of note are `GIFT128` [1], `Speck128` [2], and `AES128` (all with 128-bit block sizes for `SimpleENC`), and `GIFT64`, `PRESENT` [6] and `Speck64` (with a 64-bit block size for `SimpleENCsmall`). The choice of AES is obvious, albeit not necessarily for the lightweight setting, as this is a well established cipher used in multiple standards. However, CBC-MAC does not enable pipelining, so this mode would not be the best choice on modern processors which have out-of-order capabilities and AES-NI support.

1.4 Security Bounds

The precise security bounds are provided within the paper below. Here, we present tables that show the bounds achieved for different configurations of: *number of encrypted messages* denoted q_E, *maximum size of an encrypted messages* denoted ℓ_{max}, and *number of decryption queries by the adversary* denoted q_D.

`SimpleENC`. In Table 1, we provide bounds for the case of $n = 128$. As can be seen, it is possible to go *well beyond the birthday bound* and still maintain a security margin of 2^{-32}. Furthermore, whereas standard schemes like CCM fail with probability 1/2 at 2^{64} blocks encrypted, `SimpleENC` can encrypt well beyond that number of blocks and still achieve security of 2^{-32}. This is due to the nonce-based derivation method that we use.

n	q_E	ℓ_{max}	q_D	Total Blocks Encrypted	Total Bytes Encrypted	Security Bound
128	2^{46}	2^{24}	2^{62}	2^{70}	2^{74}	2^{-33}
128	2^{40}	2^{28}	2^{32}	2^{68}	2^{72}	2^{-33}
128	2^{32}	2^{32}	2^{48}	2^{64}	2^{68}	2^{-33}
128	2^{16}	2^{40}	2^{32}	2^{56}	2^{60}	2^{-33}
128	2^{8}	2^{44}	2^{42}	2^{52}	2^{56}	2^{-32}
128	2^{1}	2^{48}	2^{20}	2^{49}	2^{53}	2^{-32}

Table 1. Security bounds for `SimpleENC`

The bounds in Table 1 refer to the case that unique nonces are guaranteed. When considering *random IVs* that may repeat with some probability, the only difference is an additional term of ${q_E}^2/2^{n-7}$, which in the case of $n = 128$ means that for security of 2^{-32} it must hold that $q_E \leq 2^{44}$. Thus, only the first row of the table is affected (and only mildly).

`SimpleENCsmall`. In Table 2, we provide bounds for the case of $n = 64$. Most modes of operation fail miserably at this block size due to the birthday bound. In contrast, `SimpleENCsmall` provides very good bounds, even when encrypting a high volume of messages. Note that when encrypting many messages, the lengths of the encrypted messages must still be quite small. Fortunately, the typical use of a small-block block cipher is for weak devices that encrypt at small volumes. These devices often have a long life-time and thus may encrypt many messages overall, making these types of parameters ideal. Having said the above, it is also possible to encrypt a smaller number of large messages and maintain security.

n	q_E	ℓ_{max}	q_D	Total Blocks Encrypted	Total Bytes Encrypted	Security Bound
64	2^{26}	2^3	2^{20}	2^{29}	2^{32}	2^{-32}
64	2^{24}	2^4	2^{20}	2^{28}	2^{31}	2^{-32}
64	2^{20}	2^6	2^{20}	2^{26}	2^{29}	2^{-32}
64	2^{14}	2^9	2^{10}	2^{23}	2^{26}	2^{-32}
64	2^{12}	2^{10}	2^{10}	2^{22}	2^{25}	2^{-32}
64	2^8	2^{12}	2^{10}	2^{20}	2^{23}	2^{-32}
64	2^{32}	2^4	2^{24}	2^{36}	2^{39}	2^{-24}
64	2^{28}	2^6	2^{23}	2^{34}	2^{37}	2^{-24}
64	2^{14}	2^{13}	2^{21}	2^{27}	2^{30}	2^{-24}

Table 2. Security bounds for `SimpleENCsmall`.

We stress that the bounds in Table 2 assume that unique nonces are used. However, if *random IVs* are used then an additional term of ${q_E}^2/2^{n-8}$ must be considered, which in the case of $n = 64$ means that security of 2^{-32} cannot be achieved. Lower security bounds can be achieved by limiting q_E to a small number. Nevertheless, `SimpleENCsmall` with a block size of 64 bits only, should typically used in unique-nonce mode (e.g., by a device holding minimal state in the form of a counter that is incremented with every encryption).

2 Description of `SimpleENC`, `SimpleENC`′ and `SimpleENCsmall`

The `SimpleENC`, `SimpleENC`′ and `SimpleENCsmall` modes of encryption can be used with a unique or randomly-chosen nonce. The descriptions below will thus refer to a nonce, and in the analysis of security we will show separate bounds for the case that the nonce is guaranteed to be unique and the case that it is randomly chosen. We provide a high-level description of the modes here, and refer to Appendices A, B and C for the exact specifications.

2.1 The Building Blocks CBCMAC-IV, CTRENC, and CENC

CBCMAC-IV. Let X be a nonempty sequence of $x > 0$ blocks, $X = X_1, \ldots, X_x$, let R be a block, and let K be a key. Define the following sequence of blocks: $T_0 = R$ and $T_i = E(K, X_i \oplus T_{i-1})$ for $i = 1, \ldots, x$. The last block T_x is also referred to as an authentication tag (for X). The CBCMAC-IV of X under the key K and the IV R is denoted by $\texttt{CBCMAC-IV}(K, R, X)$ and is defined to be the block T_x. The input X is also referred to as a "message".

Remark 2.1. CBCMAC-IV is defined only for sequences (messages) of full blocks, consisting of at least one block.

Remark 2.2. The standard basic CBC-MAC is a special case of CBCMAC-IV where $R = 0^n$, i.e., $\texttt{CBCMAC-IV}(K, 0^n, X)$. Alternatively, CBCMAC-IV of the message X can be viewed as the standard CBC-MAC applied to $X' = (X_1 \oplus R), X_2, \ldots, X_x$. In other words, $\texttt{CBCMAC-IV}(K, R, X)$ =CBC-MAC(K, X').

CTRENC. Let $0 < \delta \leq n$ be an integer, let IV be a string of $(n - \delta)$ bits, and let K be a key. Let M be a nonempty string of bits such that $|M| \leq n \cdot 2^\delta$. Let $i_{[\#\delta]}$ denote the δ-bit representation of the index i. Denote $|M| \pmod n = r$. Parse $M = M_1, \ldots, M_m$ as a sequence of $m = \lceil |M|/n \rceil$ blocks (possibly appending $(n - r)$ 0 bits to M, when $r > 0$, to complete to M to an integer number of blocks). Observe that $1 \leq m \leq 2^\delta$. The (counter mode) CTR encryption of M under the key K with the IV IV is denoted by $\texttt{CTRENC}(K, \texttt{IV}, M)$ and is defined as the ciphertext C computed as follows.

```
for $j = 1, \ldots, m$
    $Ctr_j = \texttt{IV} \parallel (j-1)_{[\#\delta]}$
    $C_j = M_j \oplus E(K, Ctr_j)$
if $r > 0$ then $C_m^* = Truncate(C_m, r)$
$C = C_m^* \parallel C_{m-1} \parallel \ldots \parallel C_1$
```

By definition, $|C| = |M|$. (As a degenerate case, the ciphertext for an empty string is also an empty string.)

Remark 2.3. The blocks Ctr_j are called counter blocks. They are well defined for $j = 1, \ldots, m$ due to the constraint $1 \leq m \leq 2^\delta$ and the running counter $(j-1)_{[\#\delta]}$. The choice of δ implies limits on the longest possible length of the message M, although an implementation can choose to allocate δ bits for counter, but independently restrict message lengths to less than $\sim 2^\delta$ blocks. When $\delta = n$ the IV is an empty string and the counter blocks have the form $(j-1)_{[\#\mathbf{n}]}$.

CENC derivation. We use CENC in order to carry out key derivation, and therefore limit our description to our specific use. Let $0 < \delta < n$ be an integer, let IV be a string of $(n - \delta)$ bits, and let K be a key. The CENC derivation of $c \cdot n$ bits under the key K with the IV IV is denoted by $\texttt{CENC}(K, \texttt{IV}, c)$ and is defined as the output computed as follows.

```
$D_0 = E(K, \texttt{IV} \parallel (0)_{[\#\delta]})$
```

```
  for $j = 1, \ldots, c$
      $Ctr_j = \texttt{IV} \parallel (j)_{[\#\delta]}$
      $D_j = E(K, Ctr_j) \oplus D_0$
  Output $D = D_1 \parallel \ldots \parallel D_c$
```

The CENC encryption mode is defined in [12] and its improved security bounds are shown in [13] (to be a corollary from a theorem proved in [14]).

2.2 SimpleENC – encryption with $\kappa = n$ for large n

SimpleENC = $\texttt{SimpleENC}_K(N, A, M)$ is an AEAD scheme (mode of operation) that encrypts and authenticates a header A and message M, with a nonce N and key K. It uses the building blocks CBCMAC-IV and CTRENC that are described above. This scheme is defined for keys of length κ and block size $n = \kappa$.

Key generation. SimpleENC uses a single key K of length κ.

Encryption. Let A be authentication data, M the plaintext, and N a nonce of length $n - 8$. The message is encrypted in SimpleENC in three steps:

1. *Key derivation:* The block cipher is applied to $N\|0$, $N\|1$ and $N\|2$ in order to obtain K_E, K_{MAC} and $[N_2, N_1]$, all of length n (note that N_1, N_2 are of length $n/2$ each). We remark that N can be of length $n-2$ to support these 3 derivations, but we set it to be $n - 8$ so that everything is defined over bytes.
2. *Encryption:* Encrypt the message M using CTRENC with key K_E and initialization vector $IV = N_1$.
3. *Authentication:* Compute Tag by applying CBCMAC-IV with key K_{MAC} and initialization vector $IV = N_2$ to A and M. In order to prevent ambiguity, the actual string input to CBCMAC-IV is $\texttt{pad10}^*(A) \parallel \texttt{pad10}^*(M)$, where $\texttt{pad10}^*(x)$ denotes the operation of padding x to be block aligned by add a single 1 and then zeroes (and adding a full block if x is already aligned). Recall that $IV = N_2$ is XORed into the first block of this string.
4. *Output:* Output $Tag\|C$.

Decryption. Let A be authentication data, C the ciphertext, N a nonce of length $n - 8$, and Tag the MAC tag. The ciphertext is decrypted in SimpleENC in three steps:

1. *Key derivation:* The block cipher is applied to $N\|0$, $N\|1$ and $N\|2$ in order to obtain K_E, K_{MAC} and $[N_2, N_1]$, all of length n (note that N_1, N_2 are of length $n/2$ each).
2. *Decryption:* Decrypt the ciphertext C using CTRENC with key K_E and initialization vector $IV = N_1$. Let the result be M.
3. *Authentication verification:* Compute Tag' by applying CBCMAC-IV with key K_{MAC} and initialization vector $IV = N_2$ to A and M, with padding as in encryption.
4. *Output:* Output M if $Tag' = Tag$; otherwise, output $\perp$.

Simplicity. Observe that decryption is almost identical to encryption since *decryption* of C in `CTRENC` is exactly the same code as *encryption* of M in `CTRENC`; likewise, the computation of `CBCMAC-IV` is identical. Thus, the only difference is in the output step. In addition, `SimpleENC` only requires computation of the block cipher in the "forward" direction (encryption). Finally, only very trivial 10^* padding is required; in particular, no length padding is needed, and it enables encryption in an online mode (streaming).

Performance. The input for encryption is (N, A, M). The following computation counts the number of blocks in the padded A and M combination.

– *For the encryption of M:* M (possibly padded with 0 bits to the next boundary of a multiple of n) is parsed as m blocks, where $m = \lceil |M|/n \rceil$.
– *For the authentication of* $X = \mathtt{pad10}^*(A) \parallel \mathtt{pad10}^*(M)$: X consists of x blocks where $x = a + m'$, and a is the number of blocks in $\mathtt{pad10}^*(A)$ and m' is the number of blocks in $\mathtt{pad10}^*(M)$. The value of a is $a = 1 + \lfloor |A|/n \rfloor$ (e.g., $a = 1$ if $A = \perp$) and the value of m' is $m' = 1 + \lfloor |M|/n \rfloor$.

The performance of `SimpleENC` is measured in terms of the number of invocations of the block cipher E for processing A and M. The key derivation step requires 3 invocations of E. The `CTRENC` encryption requires m invocations of E. The `CBCMAC-IV` authentication requires $x = a + m'$ invocations of E. Thus, the total number of invocations of E is

$$\mathtt{TotalECalls}(\mathtt{SimpleENC}) = 3 + a + m' + m = 5 + \left\lfloor \frac{|A|}{n} \right\rfloor + 2 \cdot \left\lfloor \frac{|M|}{n} \right\rfloor .$$

2.3 `SimpleENC`′ – encryption with $\kappa = 2n$ for large n

This mode is a variant of `SimpleENC` for the case that n is large, but the key is larger. For example, it can be used with AES256 where $n = 128$ and $\kappa = 256$. `SimpleENC`′ is identical to `SimpleENC` except that the key derivation step must generate more key material. Thus, the only difference in both encryption and decryption is as follows:

1. *Key derivation:* The block cipher is applied to $N\|0$, $N\|1, \ldots, N\|4$ in order to obtain K_E, K_{MAC} of length $2n$ each, and $[N_2, N_1]$ of length n (note that N_1, N_2 are of length $n/2$ each).

Performance. The performance of `SimpleENC`′ differs from `SimpleENC` by the fact that the key derivation step requires 5 invocations of E instead of 3. Thus, the total number of invocations of E is

$$\mathtt{TotalECalls}(\mathtt{SimpleENC}') = 5 + a + m' + m = 7 + \left\lfloor \frac{|A|}{n} \right\rfloor + 2 \cdot \left\lfloor \frac{|M|}{n} \right\rfloor .$$

2.4 SimpleENCsmall – encryption with $\kappa = 2n$ for small n

SimpleENCsmall is a variant of SimpleENC that can be used for block ciphers with a small block size, like 64 bits. In most modes of operations, encryption with such small blocks is insecure when encrypting even a small number of small messages. This fact was demonstrated in practice in the highly effective Sweet32 attack on 3DES in TLS [5]. SimpleENCsmall is designed to provide good security even in this case.

$\texttt{SimpleENCsmall} = \texttt{SimpleENCsmall}_K(N, A, M)$ is an AEAD scheme (mode of operation) that encrypts and authenticates a header A and message M, with a nonce N and key K. It uses the building blocks CBCMAC-IV and CTRENC that are described above. This scheme is defined for keys of length κ and block size $\kappa = 2n$.

We provide a high-level description of the mode here, and refer to Appendix C for the exact specification. Observe that since the block length here is short, we use CENC for derivation and encryption in order to get good bounds.

Key generation. SimpleENCsmall uses a single key K of length κ.

Encryption. Let A be authentication data, M the plaintext, and N a nonce of length $n - 8$. The message is encrypted in SimpleENCsmall in three steps:

1. *Key derivation:* The block cipher is applied to $N\|0, N\|1, \ldots, N\|5$ in order to obtain 6 blocks of length $\kappa/2$. Then, CENC is used to obtain 5 output blocks, by XORing the result of $E(K, N\|0)$ to all other results. Finally, keys K_E and K_{MAC} of length $n = 2\kappa$ each are defined using 4 of the output blocks, and two nonces $[N_2, N_1]$ of length $n/2$ each are defined using the fifth output block. and $[N_2, N_1]$. We remark that N can be of length $n - 3$ to support these 3 derivations, but we set it to be $n - 8$ so that everything is defined over bytes.
2. *Encryption:* Encrypt the message M using CTRENC with key K_E and initialization vector $IV = N_1$.
3. *Authentication:* Compute Tag by applying CBCMAC-IV with key K_{MAC} and initialization vector $IV = N_2$ to A and M. In order to ensure no ambiguity, the actual string input to CBCMAC-IV is $\texttt{pad10}^*(A) \| \texttt{pad10}^*(M)$. Recall that $IV = N_2$ is XORed into the first block of this string.
4. *Output:* Output $Tag\|C$.

Decryption. Let A be authentication data, C the ciphertext, N a nonce of length $n-8$, and Tag the MAC tag. The ciphertext is decrypted in SimpleENCsmall in three steps:

1. *Key derivation:* Exactly as in encryption.
2. *Decryption:* Decrypt the ciphertext C using CTRENC with key K_E and initialization vector $IV = N_1$. Let the result be M.
3. *Authentication verification:* Compute Tag' by applying CBCMAC-IV with key K_{MAC} and initialization vector $IV = N_2$ to A and M, with padding as in encryption.

4. *Output:* Output M if $Tag' = Tag$; otherwise, output $\perp$.

Simplicity. `SimpleENCsmall` is almost as simple as `SimpleENC`, with the only difference being that `CENC` is used for the derivation. Since we use `CENC` with a *single* additional block only, this is very easily implemented (requiring only to XOR each output of the block cipher with the derived blocks).

Performance. The key derivation step using `CENC` requires 6 invocations of E, `CTRENC` encryption requires m invocations of E, and `CBCMAC-IV` authentication requires $x = a + m'$ invocations of E. Thus, the total number of invocations of E is

$$\texttt{TotalECalls}(\texttt{SimpleENCsmall}) = 6 + a + m' + m = 8 + \left\lfloor \frac{|A|}{n} \right\rfloor + 2 \cdot \left\lfloor \frac{|M|}{n} \right\rfloor .$$

Thus, `SimpleENCsmall` provides good security even for small-block block ciphers, at a cost that is almost that of `SimpleENC`.

3 Definitions of Security

Before proceeding to formally prove our bounds, we provide definitions for chosen-plaintext security (CPA) and authenticated encryption (AE). We differentiate between nonce-based security, where the nonce is provided by the adversary but the adversary is required to use a *unique nonce* in each encryption query, and IV-based security where a *random IV* is chosen as part of the encryption process (and not chosen by the adversary). We distinguish these cases by calling the former a nonce and the latter an IV.

CPA-secure nonce-based encryption (nE). Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a nonce-based encryption scheme. Encryption is a deterministic function receiving a key K, nonce N and message M, and is denoted $C = \mathsf{Enc}_K(N, M)$. Consider the following oracles:

- *Oracle* Enc_K*:* upon input (N, M), it computes $C = \mathsf{Enc}_K(N, M)$. The output is C.
- *Oracle* $\$_K$*:* upon input (N, M), it computes $C = \mathsf{Enc}_K(N, M)$. The output is a random string of length $|C|$.

The CPA-advantage of an adversary $\mathcal{A}$ against a nonce-based encryption scheme is defined to be:

$$\mathsf{Adv}^{\mathsf{nCPA}}_{\Pi}(\mathcal{A}) = \left| \Pr_K \left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot)} = 1 \right] - \Pr_K \left[\mathcal{A}^{\$_K(\cdot,\cdot)} = 1 \right] \right|$$

where $\mathcal{A}$ may not make two queries $(N, M), (N, M')$ to its oracle with the same nonce N.

CPA-secure IV-based encryption (ivE). Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IV-based encryption scheme, and let the space of IVs be $\mathcal{IV}$. Encryption involves choosing $IV \leftarrow \mathcal{IV}$ uniformly at random, and then computing the deterministic function $\mathsf{Enc}_K(IV, M)$. Consider the following oracles:

- *Oracle* Enc_K*:* upon input M, it chooses $IV \leftarrow \mathcal{IV}$ at random and computes $C = \mathsf{Enc}_K(IV, M)$. The output is $(IV \| C)$.
- *Oracle* $\$_K$*:* upon input M, it chooses $IV \leftarrow \mathcal{IV}$ at random and computes $C = \mathsf{Enc}_K(IV, M)$. The output is a random string of length $|(IV \| C)|$.

The advantage of an adversary $\mathcal{A}$ against an IV-based encryption scheme is defined to be:

$$\mathsf{Adv}_{\Pi}^{\mathsf{ivCPA}}(\mathcal{A}) = \left| \Pr_K \left[\mathcal{A}^{\mathsf{Enc}_K(\cdot)} = 1 \right] - \Pr_K \left[\mathcal{A}^{\$_K(\cdot)} = 1 \right] \right|$$

Secure nonce-based authenticated encryption (nAE). Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a nonce-based encryption scheme. Encryption is a deterministic function receiving a key K, nonce N, associated data A and plaintext message M, and is denoted $C = \mathsf{Enc}_K(N, A, M)$. We denote decryption by $\mathsf{Dec}_K(N, A, C)$. Consider the following oracles:

- *Oracle* $\$_K$*:* upon input (N, A, M), it computes $C = \mathsf{Enc}_K(N, A, M)$. If $C = \perp$ then the output is $\perp$; otherwise, the output is a random string of length $|C|$.
- *Oracle* $\perp$*:* upon any input, returns $\perp$.

The advantage of an adversary $\mathcal{A}$ against a nonce-based authenticated encryption scheme is defined to be:

$$\mathsf{Adv}_{\Pi}^{\mathsf{nAE}}(\mathcal{A}) = \left| \Pr_K \left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot), \mathsf{Dec}_K(\cdot,\cdot,\cdot)} = 1 \right] - \Pr_K \left[\mathcal{A}^{\$_K(\cdot,\cdot,\cdot), \perp(\cdot,\cdot,\cdot)} = 1 \right] \right|$$

where $\mathcal{A}$ may not make two *encryption* queries $(N, M, C), (N, M', C')$ with the same first component (nonce), and may not make any *decryption* query for a value (N, A, C) that was obtained as output from some encryption query.

Secure random-IV based authenticated encryption (ivAE). Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IV-based encryption scheme, and let the space of IVs be $\mathcal{IV}$. Encryption involves choosing $IV \leftarrow \mathcal{IV}$ uniformly at random, and then computing the deterministic function $\mathsf{Enc}_K(IV, A, M)$. We denote decryption by $\mathsf{Dec}_K(IV, A, C)$. Consider the following oracles:

- *Oracle* $\$_K$*:* upon input (A, M), it chooses $IV \leftarrow \mathcal{IV}$ uniformly at random and computes $C = \mathsf{Enc}_K(IV, A, M)$. If $C = \perp$ then the output is $\perp$; otherwise, the output is a random string of length $|(IV \| C)|$.
- *Oracle* $\perp$*:* upon any input, returns $\perp$.

The advantage of an adversary $\mathcal{A}$ against an IV-based authenticated encryption scheme is defined to be:

$$\mathsf{Adv}_{\Pi}^{\mathsf{ivAE}}(\mathcal{A}) = \left| \Pr_K \left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot), \mathsf{Dec}_K(\cdot,\cdot,\cdot)} = 1 \right] - \Pr_K \left[\mathcal{A}^{\$_K(\cdot,\cdot), \perp(\cdot,\cdot,\cdot)} = 1 \right] \right|$$

where $\mathcal{A}$ may not make any *decryption* query for a value (N, A, C) that was obtained as output from some query to Enc.

4 Authenticity Bounds

Our entire security analysis models the block cipher (when used with a random key) as a random permutation. This is standard, and the only difference necessary is to add the difference between a random permutation and a block cipher.

4.1 The Basic Forgery Bound for `CBCMAC-IV`

A multi-key and multi-query MAC game. In our setting, since the MAC keys are derived using a pseudorandom function from the nonce, we actually have a multi-key scenario. Thus, in order to prove our authenticity bound for `SimpleENC`, we begin by defining a multi-key and multi-query MAC game for our purpose. We denote the MAC computation algorithm by Mac and the verify algorithm by Vrfy (thus $\mathsf{Vrfy}_k(m,t) = 1$ if t is a valid MAC-tag on m with key k, and otherwise it equals 0).

We define an oracle Mac that receives a message m, chooses a fresh random key k, stores (i,k) where i is the next unused index (initially $i = 0$), and returns $t = \mathsf{Mac}_k(m)$. In addition, we define an oracle Vrfy that receives a triple (i,m,t) and checks if some (i,k) has been stored. If yes, and t was not returned as the response from a previous query $\mathsf{Mac}_k(m)$, it returns $\mathsf{Vrfy}_k(m,t)$; if no, it chooses a new k, stores (i,k), and returns $\mathsf{Vrfy}_k(m,t)$. We consider an experiment where an adversary is given access to both the Mac and Vrfy oracles, and succeeds if it asks some query to Vrfy that returns 1. We remark that since Mac and Vrfy have access to the same set of keys, they are not really different oracles; rather they are formally modeled as a single oracle with two different types of queries. The MAC forging experiment that we consider for $\mathcal{A}$ with MAC scheme $\Pi = (\mathsf{Mac}, \mathsf{Vrfy})$ is as follows:

Experiment $\mathsf{MultiForge}_{\Pi}(\mathcal{A})$:
1. Execute $\mathcal{A}^{\mathsf{Mac}(\cdot),\mathsf{Vrfy}(\cdot,\cdot,\cdot)}(1^n)$, where the oracles are defined as above.
2. Output 1 if and only if there exists a query to Vrfy that returned 1.

We say that an adversary $\mathcal{A}$ is a $(Q_M, Q_V, \vec{\ell})$-adversary if it makes at most Q_M queries to the Mac oracle and at most Q_V queries to the Vrfy oracle, and the ith query to the Mac oracle is of length ℓ_i blocks where $\vec{\ell} = (\ell_1, \ldots, \ell_{Q_M})$. Observe that the ith query to the MAC oracle uses the ith key, and thus an independent key is used each time. Therefore, the same key may be used more than once, but only with the probability that the same random key is chosen more than once amongst q keys. We now bound the adversarial advantage in $\mathsf{MultiForge}$.

Lemma 4.1. *Let $\Pi = (\mathsf{Mac}, \mathsf{Vrfy})$ be* `CBCMAC-IV` *as defined in* `SimpleENC` *and* `SimpleENCsmall` *with a random permutation. Then, for every $(Q_M, Q_V, \vec{\ell})$-adversary $\mathcal{A}$, it holds that*

$$\Pr\left[\mathsf{MultiForge}_{\Pi}(\mathcal{A}) = 1\right] \leq Q_V \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{Q_M}^2}{2^{n+1}},$$

where $\ell_{\max}$ *is the maximum value in* $\{\ell_1, \ldots, \ell_{Q_M}\}$.

Proof. We begin by modifying $\mathsf{MultiForge}$ to $\mathsf{MultiForge}'$ with the only difference being that if the Mac oracle chooses a key that has already been used before, then the experiment halts and outputs 1 (i.e., the adversary wins). Since the probability that two keys are equal is at most $\frac{{Q_M}^2}{2^{n+1}}$, it follows that

$$\Pr\left[\mathsf{MultiForge}_{\Pi}(\mathcal{A}) = 1\right] \leq \Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1\right] + \frac{{Q_M}^2}{2^{n+1}}.$$

Now, in $\mathsf{MultiForge}'$, each CBC-MAC key is used for tagging at most one message. In particular, the i'th key is used to generate a MAC-tag on a *single* message of length ℓ_i. By [4], for any single key, the advantage of the adversary is at most

$$\frac{12 \cdot \ell \cdot Q^2}{2^n} + \frac{64 \cdot \ell^4 \cdot Q^2}{2^{2n}}$$

where Q is the number of queries, and ℓ the length of the longest query. Thus, intuitively, in our setting where $Q = 1$ per key the probability that any single query to Vrfy of the form (i, m, t) returns 1 is at most

$$\frac{12 \cdot \ell_i}{2^n} + \frac{64 \cdot \ell_i^4}{2^{2n}}.$$

Since $\mathcal{A}$ makes at most Q_V queries to Vrfy, we therefore claim that

$$\Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1\right] \leq Q_V \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) \tag{1}$$

where $\ell_{\max}$ is the maximum value in $\{\ell_1, \ldots, \ell_{Q_M}\}$. Note that these are the lengths of the queries to the Mac oracle.

We prove Eq. (1) by reducing the security to a single CBC-MAC forgery. In order to do so, we bound the forging probability for every possible distribution over the verification queries. Specifically, let $\vec{q} = (q_1, \ldots, q_{Q_M+Q_V})$ be a vector of integers where q_i is the (non-negative) number of *verification queries* made to the ith MAC key. Observe that each MAC query and verification query can potentially be to a new key, and thus there are at most $Q_M + Q_V$ different keys used throughout the game. However, since there are Q_V verification queries, we have that $\sum_{i=1}^{Q_M+Q_V} q_i = Q_V$. We denote by $\mathsf{query}(\vec{q})$ the event that in the execution of $\mathsf{MultiForge}'$, $\mathcal{A}$'s queries to the verification oracle are exactly as in $\vec{q}$. We have:

$$\Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1\right] = \sum_{\vec{q}} \Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1 \mid \mathsf{query}(\vec{q})\right] \cdot \Pr[\mathsf{query}(\vec{q})] \tag{2}$$

where the sum is over all possible integer vectors $\vec{q}$ such that $\sum_{i=1}^{Q_M+Q_V} q_i = Q_V$ (technically, this is the set of all the partitions of the integer $Q_M + Q_V$ with non-negative summands). We now argue that for *every* $\vec{q}$,

$$\Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1 \mid \mathsf{query}(\vec{q})\right] \leq Q_V \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right). \tag{3}$$

In order to see this, fix $\vec{q}$, and let $\mathcal{A}_i$ be an adversary who attacks CBC-MAC for a single key. $\mathcal{A}_i$ works by choosing all MAC keys k_j for $j \neq i$ (formally for $j \in \{1, \ldots, Q_M + Q_V\} \setminus \{i\}$). Then, $\mathcal{A}_i$ invokes $\mathcal{A}$ and simulates the Mac and Vrfy oracles for all keys k_j with $j \neq i$ by using the keys it chose. In contrast, for every Mac or Vrfy oracle query to the ith key, $\mathcal{A}_i$ sends the query externally to its own oracle. If the queries made by $\mathcal{A}$ are not exactly $\vec{q}$, then $\mathcal{A}_i$ aborts. Denote by $\mathsf{forge}(\mathcal{A}_i)$ the probability that $\mathcal{A}_i$ succeeds in forging a tag in one of the verification queries. Then, since in $\mathsf{MultiForge}'$ there is at most *one* Mac query to the ith key, and the longest query is $\ell_{\max}$, we have

$$\Pr\left[\mathsf{forge}(\mathcal{A}_i)\right] \leq q_i \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right).$$

Observe that this argument holds since we have conditioned over $\vec{q}$, and we only consider $\mathcal{A}_i$'s success when the query vector is $\vec{q}$. Denote by $\mathsf{forge}_{\mathcal{A}}(i)$ the event that $\mathcal{A}$ forges a verification query for the ith key. Then,

$$\Pr\left[\mathsf{forge}(\mathcal{A}_i)\right] = \Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1 \wedge \mathsf{forge}_{\mathcal{A}}(i) \mid \mathsf{query}(\vec{q})\right]$$

Thus,

$$\begin{aligned}
&\Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1 \mid \mathsf{query}(\vec{q})\right] \\
&\quad\leq \sum_{i=1}^{Q_M+Q_V} \Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1 \wedge \mathsf{forge}_{\mathcal{A}}(i) \mid \mathsf{query}(\vec{q})\right] \\
&\quad= \sum_{i=1}^{Q_M+Q_V} \Pr\left[\mathsf{forge}(\mathcal{A}_i)\right] \\
&\quad\leq \sum_{i=1}^{Q_M+Q_V} q_i \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) \\
&\quad= Q_V \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right),
\end{aligned}$$

where the last equality is because $\sum_{i=1}^{Q_M+Q_V} q_i = Q_V$. This proves Eq. (3). Combining this with Eq. (2), we have:

$$\begin{aligned}
&\Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1\right] \\
&\quad= \sum_{\vec{q}} \Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1 \mid \mathsf{query}(\vec{q})\right] \cdot \Pr[\mathsf{query}(\vec{q})] \\
&\quad\leq \sum_{\vec{q}} Q_V \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) \cdot \Pr[\mathsf{query}(\vec{q})] \\
&\quad= Q_V \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) \cdot \sum_{\vec{q}} \Pr[\mathsf{query}(\vec{q})] \\
&\quad= Q_V \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right).
\end{aligned}$$

We conclude that

$$\Pr\left[\mathsf{MultiForge}_{\Pi}(\mathcal{A})=1\right] \leq Q_V \cdot \left(\frac{12\cdot \ell_{\max}}{2^n} + \frac{64\cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{Q_M}^2}{2^{n+1}},$$

thereby completing the proof. ■

4.2 An Improved Forgery Bound for `CBCMAC-IV`

The bound proven in Section 4.1 is strong enough for a large block, since $\frac{{Q_M}^2}{2^{n+1}}$ is small (e.g., for $n = 128$, we can achieve 2^{-32} security with $Q_M \leq 2^{48}$). However, for smaller blocks, this is too weak and we need to prove a different bound. Intuitively, there is one important element that we did not utilize at all in the analysis in Section 4.1; namely, the fact that a random nonce of length $n/2$ is XORed to the first block in the `CBCMAC-IV` computation. This is important since the analysis of Section 4.1 assumes that each message is MACed with a different key, whereas CBC-MAC is secure when the same key is used to MAC multiple messages as long as these messages are prefix-free. Since the first block of the messages are XORed with a random nonce of length $n/2$ this ensures prefix-freeness with high probability, unless the nonce repeats. Thus, `CBCMAC-IV` actually provides much better bounds, and remains secure as long as the *same* combination of derived key and derived nonce does not repeat. We do this in the following lemma.

Lemma 4.2. *Let $\Pi_1 = (\mathsf{Mac}, \mathsf{Vrfy})$ be* `CBCMAC-IV` *as defined in* `SimpleENC` *and let Π_2 be as in* `SimpleENCsmall`*, with a random permutation. Then, for every $(Q_M, Q_V, \vec{\ell})$-adversary $\mathcal{A}$ with* `SimpleENC`*, it holds that*

$$\Pr\left[\mathsf{MultiForge}_{\Pi}(\mathcal{A}_1)=1\right] \leq Q_V \cdot \left(\frac{48\cdot \ell_{\max}}{2^n} + \frac{256\cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{Q_M}^3}{6\cdot 2^{3n}},$$

where $\ell_{\max}$ is the maximum value in $\{\ell_1, \dots, \ell_{Q_M}\}$. Likewise, when considering `SimpleENCsmall` *it holds that*

$$\Pr\left[\mathsf{MultiForge}_{\Pi}(\mathcal{A}_2)=1\right] \leq Q_V \cdot \left(\frac{48\cdot \ell_{\max}}{2^n} + \frac{256\cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{Q_M}^3}{6\cdot 2^{5n}}.$$

Proof. In the proof of Lemma 4.1, we bounded the cheating probability by first bounding the probability that a derived key is obtained more than once. This immediately adds a bound of ${Q_M}^2/2^{n+1}$, which is a birthday bound. However, observe that if a derived key is obtained multiple times, but the *prefix* of the message is different each time, then CBC-MAC is still secure. In `SimpleENC`, the random (half) nonce N_2 is XORed to the first block of the message, guaranteeing prefix freeness with high probability. Thus, we can actually bound the probability of forging the CBC-MAC even if the key repeats multiple times. By [15, Theorem

2.2], the probability of r collisions on q queries over a domain of size 2^S is at most $\frac{q^r}{r!\cdot 2^{(r-1)\cdot S}}$. Thus, the probability that a key/nonce pair repeats 3 times or more is upper bound by $\frac{q^3}{6\cdot 2^{2S}}$. Note that the nonce N_2 is XORed into the first block, but since N_2 is random the probability that two first blocks repeat is the same probability of a collision as above. (Note that $S = 1.5n$ for `SimpleENC` and $S = 2.5n$ for `SimpleENCsmall`.)

Returning to the proof of Lemma 4.1, we define MultiForge to be exactly the same as before except that now we also choose a random N_2 and XOR it into the first block of a queried message. Next, we define MultiForge′ to be the same as MultiForge except that if there exists a key/nonce-N_2 pair that repeats (i.e., is chosen) three or more times, then the adversary loses. By the above, we have

$$\Pr\left[\mathsf{MultiForge}_{\Pi}(\mathcal{A}) = 1\right] \leq \Pr\left[\mathsf{MultiForge}'_{\Pi}(\mathcal{A}) = 1\right] + \frac{{Q_M}^3}{6\cdot 2^{2S}}.$$

Now, in MultiForge′, each CBC-MAC key is used at most *twice*, and prefix freeness is guaranteed. By [4], for any single key, the advantage of the adversary is at most

$$\frac{12\cdot \ell_{\max}\cdot Q^2}{2^n} + \frac{64\cdot {\ell_{\max}}^4\cdot Q^2}{2^{2n}}$$

but here $Q \leq 2$ and thus the advantage of the adversary for any single key is at most

$$\frac{48\cdot \ell_{\max}}{2^n} + \frac{256\cdot {\ell_{\max}}^4}{2^{2n}}.$$

In the same way as in the proof of Lemma 4.1, we multiply this by Q_V to obtain the forging probability over all keys. We therefore conclude that

$$\Pr\left[\mathsf{MultiForge}_{\Pi}(\mathcal{A}) = 1\right] \leq Q_V\cdot\left(\frac{48\cdot \ell_{\max}}{2^n} + \frac{256\cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{Q_M}^3}{6\cdot 2^{2S}}.$$

Using the fact that $S = 1.5n$ for `SimpleENC` and $S = 2.5n$ for `SimpleENCsmall`, the proof is completed. ■

4.3 The Authenticity Bound for `SimpleENC`

Unique nonce case. We begin by analyzing `SimpleENC` in the nonce setting, where the adversary must use a unique nonce in every encryption query (such an adversary is called *nonce respecting*).

Theorem 4.3. *Consider the nonce-version of* `SimpleENC`*. Then, for every nonce-respecting $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$-adversary $\mathcal{A}$, the probability that $\mathcal{A}$ makes a query to the decryption oracle that does not return $\perp$, denoted* $\mathsf{ValidDec}_{\mathcal{A}}$*, is at most*

$$\Pr\left[\mathsf{ValidDec}_{\mathcal{A}}\right] \leq \min\left\{\begin{array}{l} q_D\cdot\left(\frac{12\cdot\ell_{\max}}{2^n} + \frac{64\cdot{\ell_{\max}}^4}{2^{2n}}\right) + \frac{10\cdot{q_E}^2}{2^{n+1}}\ , \\ q_D\cdot\left(\frac{48\cdot\ell_{\max}}{2^n} + \frac{256\cdot{\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6\cdot 2^{3n}} + \frac{9\cdot{q_E}^2}{2^{n+1}} \end{array}\right\}.$$

where $\ell_{\mathsf{max}} = \max\{\ell_1,\ldots,\ell_{q_E}\}$ for $\vec{\ell}_E = (\ell_1,\ldots,\ell_{q_E})$.

Proof. Let $\mathcal{A}$ be a $(q_E, q_D, n, \vec{\ell}_E, \vec{\ell}_D)$-adversary and let ℓ_{max} be the maximum length in $\vec{\ell}_E$. Without loss of generality, we assume that $\mathcal{A}$ does not query its decryption oracle with any ciphertext received from its encryption oracle. We bound the probability that $\mathcal{A}$ makes a decryption query that returns a value that is not $\perp$. Let $\mathsf{validDec}_{\mathcal{A}}$ denote the event that $\mathcal{A}$ makes such a query; we bound $\mathsf{ValidDec}_{\mathcal{A}}$ by showing that it is possible to forge a MAC in the experiment $\mathsf{MultiForge}$ defined above with probability that is related to the probability that $\mathsf{ValidDec}_{\mathcal{A}}$ occurs. In the sequel, we denote by $\mathsf{pad}(N_2, A, M)$ the construction of the MAC message as specified in `SimpleENC`. Specifically, N_2 is XORed with the first block of A, and `pad10`* is applied to the result; then, $\mathsf{pad0}^*(M)$ is concatenated to the result.

We construct an adversary $\mathcal{A}'$ for $\mathsf{MultiForge}$ that uses $\mathcal{A}$ in order to forge. Adversary $\mathcal{A}'$ works as follows:

1. $\mathcal{A}'$ invokes $\mathcal{A}$, and initializes $i = 0$ and $i' = q_E$.
2. For every encryption query (N, A, M) made by $\mathcal{A}$, adversary $\mathcal{A}'$ chooses a random K_E and $[N_2 \| N_1]$ (note that in this case $\mathcal{A}$ is nonce respecting so N never repeats, meaning that the derivation is new each time). Then, $\mathcal{A}'$ queries its MAC oracle to obtain a tag T on $\mathsf{pad}(N_2, A, M)$, and locally encrypts M in counter mode with nonce N_1 and key K_E. $\mathcal{A}$ stores the association (i, N, K_E, N_2, N_1), sets $i = i + 1$, and returns (N, A, C, T) where C is the resulting ciphertext and T the tag received back.
3. For all decryption queries (N, A, C, T) made by $\mathcal{A}$, if N is a new nonce then $\mathcal{A}'$ chooses a random K_E and $[N_2 \| N_1]$, stores the new association (i', N, K_E, N_2, N_1) and sets $i^* = i' = i'{+}1$; else, $\mathcal{A}'$ retrieves (i, N, K_E, N_2, N_1) already stored and sets $i^* = i$. Then, $\mathcal{A}'$ decrypts C with the key K_E and nonce N_1, to obtain M, and queries its Vrfy oracle with $(i^*, \mathsf{pad}(N_2, A, M), T)$.

We define the above by game_0, and the probability that $\mathcal{A}'$ successfully forges by $\mathsf{AdvMAC}^0_{\Pi}(\mathcal{A}')$. To analyze this probability, we define a new game, denoted game_1.

Game game_1: This game is identical to game_0 except that the experiment chooses a random permutation π. Then, for every nonce N it derives K_E, K_M, $[N_2 \| N_1]$ as in `SimpleENC` using π. Then, it hands K_E, N_2, N_1 to the adversary $\mathcal{A}'$, and uses K_M in the MAC query. The adversary $\mathcal{A}'$ works in the same way as in game_0, using these keys and nonce values. Let $\mathsf{AdvMAC}^1_{\Pi}(\mathcal{A}')$ denote the probability that $\mathcal{A}'$ successfully forges in this game.

We claim that the probability that $\mathcal{A}'$ outputs a valid forgery in game_1 is at most $9 \cdot {q_E}^2/2^{n+1}$ far from the probability that $\mathcal{A}'$ outputs a valid forgery in game_0. In order to see this, observe that the only difference is that all keys and nonces are chosen uniformly at random in game_0, and are derived using a pseudorandom permutation in game_1. Since choosing all values uniformly at random is equivalent to using a random function, we have that the difference between the games is the difference between a random function and a random permutation. The number of queries made to this function is equal to $3 \cdot q_E$, the number of encryption queries made by $\mathcal{A}$ (this is only encryption queries since the decryption queries are all replied with $\perp$). Thus,

$$\left|\mathsf{AdvMAC}^0_{\Pi}(\mathcal{A}') - \mathsf{AdvMAC}^1_{\Pi}(\mathcal{A}')\right| \le \frac{9 \cdot {q_E}^2}{2^{n+1}}.$$

Finally, observe that the probability that $\mathcal{A}$ outputs a valid forgery in game_1 is *exactly* the probability that a $(q_E, q_D, n, \vec{\ell})$-adversary succeeds in $\mathsf{MultiForge}$. Thus, by Lemma 4.1, we have

$$\begin{aligned}\Pr\left[\mathsf{ValidDec}_{\mathcal{A}}\right] &\le q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^2}{2^{n+1}} + \frac{9 \cdot {q_E}^2}{2^{n+1}} \\ &= q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{10 \cdot {q_E}^2}{2^{n+1}}\end{aligned}$$

and by Lemma 4.2, we have

$$\Pr\left[\mathsf{ValidDec}_{\mathcal{A}}\right] \le q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{3n}} + \frac{9 \cdot {q_E}^2}{2^{n+1}}.$$

This completes the proof. ■

Random IV case. In the case of a random IV, the only difference between this case and the previous one is if the IV repeats. Since the IV is of length $n - 8$, this makes a difference of at most $\frac{{q_E}^2}{2^{n-7}}$ (i.e., it is the standard birthday bound). Thus, we just need to add this to each of the bounds in Theorem 4.3, and we have:

Theorem 4.4. *Consider the random-IV version of* `SimpleENC`. *Then, for every* $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$*-adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *makes a query to the decryption oracle that does not return* $\perp$*, denoted* $\mathsf{ValidDec}_{\mathcal{A}}$*, is at most*

$$\Pr\left[\mathsf{ValidDec}_{\mathcal{A}}\right] \le \min\left\{\begin{array}{l} q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{10 \cdot {q_E}^2}{2^{n+1}} + \frac{{q_E}^2}{2^{n-7}}\ , \\ q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{3n}} + \frac{9 \cdot {q_E}^2}{2^{n+1}} + \frac{{q_E}^2}{2^{n-7}} \end{array}\right\}.$$

where $\ell_{\mathsf{max}} = \max\{\ell_1, \ldots, \ell_{q_E}\}$ *for* $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$*.*

4.4 The Authenticity Bound for `SimpleENC′`

The only difference between `SimpleENC` and `SimpleENC′` is that in the latter, the `CTRENC` and `CBCMAC-IV` keys are of length $2n$ and thus are derived using two applications of the random permutation each. This makes a difference of additional multiples of $\frac{{q_E}^2}{2^{n+1}}$ over the bounds proven in Theorems 4.3 and 4.4 for the nonce-based and random-IV settings, respectively. In particular, instead of 3 queries there are 5 queries, and thus the $\frac{9 \cdot {q_E}^2}{2^{n+1}}$ term should be replaced with $\frac{25 \cdot {q_E}^2}{2^{n+1}}$ throughout (note that the final term $\frac{10 \cdot {q_E}^2}{2^{n+1}}$ in the bounds is changed to $\frac{26 \cdot {q_E}^2}{2^{n+1}}$).

Theorem 4.5. *Consider the nonce-version of* SimpleENC′. *Then, for every nonce-respecting* $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$*-adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *makes a query to the decryption oracle that does not return* $\perp$*, denoted* $\mathsf{ValidDec}_{\mathcal{A}}$*, is at most*

$$\Pr\left[\mathsf{ValidDec}_{\mathcal{A}}\right] \leq \min \left\{ \begin{array}{l} q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{26 \cdot {q_E}^2}{2^{n+1}} \ , \\ q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{3n}} + \frac{25 \cdot {q_E}^2}{2^{n+1}} \end{array} \right\} .$$

where $\ell_{\mathsf{max}} = \max\{\ell_1, \ldots, \ell_{q_E}\}$ *for* $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$*.*

4.5 The Authenticity Bound for SimpleENCsmall

The difference between SimpleENC and SimpleENCsmall with respect to integrity (the CBCMAC-IV part) is only is only in the derivation of the MAC keys. In SimpleENC the derivation uses a random permutation, whereas in SimpleENCsmall the derivation uses CENC which is closer to a random function. The reason for this change is that for SimpleENCsmall we assume that n is too small to allow a standard birthday bound. Thus, we only include the possibility for $\mathsf{ValidDec}_{\mathcal{A}}$ that does not have the $\frac{{q_E}^2}{2^{n+1}}$ term.

The authenticity bound – unique nonce case. We first prove the authenticity bound for SimpleENCsmall for the unique nonce setting.

Theorem 4.6. *Consider the nonce-version of* SimpleENCsmall. *Then, for every nonce-respecting* $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$*-adversary* $\mathcal{A}$ *with* $5 \cdot q_E < 2^n/67$*, the probability that* $\mathcal{A}$ *makes a query to the decryption oracle that does not return* $\perp$*, denoted* $\mathsf{ValidDec}_{\mathcal{A}}$*, is at most*

$$\Pr\left[\mathsf{ValidDec}_{\mathcal{A}}\right] \leq q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{5n}} + \frac{25 \cdot q_E}{2^n},$$

where $\ell_{\mathsf{max}} = \max\{\ell_1, \ldots, \ell_{q_E}\}$ *for* $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$*.*

Proof. The proof of this bound is the same as that of Theorem 4.3 with the exception of the analysis of how game_1 differs from game_0 (and that we take the bound for SimpleENCsmall from Lemma 4.2, and not for SimpleENC). In the proof of Theorem 4.3 the difference between game_0 and game_1 was a pure birthday bound, whereas here we utilize the bounds of CENC. Namely, here, the only difference between game_0 and game_1 is that all keys and nonces are chosen uniformly at random in game_0, and are derived using CENC and a pseudorandom permutation in game_1. Since choosing all values uniformly at random is equivalent to using a random function, we have that the difference between the games is the difference between a random function and CENC with a random permutation. The number of queries made to this function is equal to q_E, the number of encryption queries made by $\mathcal{A}$ (this is only encryption queries since the decryption queries are all replied with $\perp$). Furthermore, we apply CENC with $w = 5$. The CENC bounds of [13] state that at long as $w \cdot q \leq 2^n/67$ then CENC with a random permutation can be distinguished from a random function with probability at most $\frac{\ell \cdot w \cdot q}{2^n}$, where q equals the number of queries and ℓ is the number of blocks. In our usage for the key derivation, $\ell = w = 5$ and so as long as $5 \cdot q_E < \frac{2^n}{67}$, the CENC derivations can be distinguished from random with probability at most $\frac{25 \cdot q_E}{2^n}$. Thus,

$$\left|\mathsf{AdvMAC}^0_{\Pi}(\mathcal{A}') - \mathsf{AdvMAC}^1_{\Pi}(\mathcal{A}')\right| \leq \frac{25 \cdot q_E}{2^n}.$$

Finally, observe that the probability that $\mathcal{A}$ outputs a valid forgery in game_1 is *exactly* the probability that a $(q_E, q_D, n, \vec{\ell})$-adversary succeeds in $\mathsf{MultiForge}$. Thus, by Lemma 4.2, we have

$$\Pr\left[\mathsf{ValidDec}_{\mathcal{A}}\right] \leq q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{5n}} + \frac{25 \cdot q_E}{2^n}.$$

This completes the proof. ■

Random IV case. As with `SimpleENC`, in the case of a random IV, the only difference between this case and the previous one is if the IV repeats. Since the IV is of length $n - 8$, this makes a difference of at most $\frac{{q_E}^2}{2^{n-7}}$ (i.e., it is the standard birthday bound). Thus, we just need to add this to the bound in Theorem 4.6, and we have:

Theorem 4.7. *Consider the random-IV version of* `SimpleENCsmall`*. Then, for every* $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$*-adversary* $\mathcal{A}$ *with* $5 \cdot q_E < 2^n/67$*, the probability that* $\mathcal{A}$ *makes a query to the decryption oracle that does not return* $\perp$*, denoted* $\mathsf{ValidDec}_{\mathcal{A}}$*, is at most*

$$\Pr\left[\mathsf{ValidDec}_{\mathcal{A}}\right] \leq q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{5n}} + \frac{25 \cdot q_E}{2^n} + \frac{{q_E}^2}{2^{n-7}}$$

where $\ell_{\mathsf{max}} = \max\{\ell_1, \ldots, \ell_{q_E}\}$ *for* $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$*.*

Remark: It is important to note that in the random IV case, `SimpleENCsmall` can only be used to encrypt *a small number of possibly long messages*. In particular, consider the case of $n = 64$ and a desired security margin of 2^{-32}. Then, due to the term of $\frac{{q_E}^2}{2^{n-7}}$, it is possible to encrypt at most $2^{12.5}$ different messages.

5 Authenticated-Encryption Security Bounds

In this section, we analyze the security of the three schemes. We use the authenticity bounds of Section 4 in order to prove the authenticated-encryption security (AEAD) of `SimpleENC`, `SimpleENC′` and `SimpleENCsmall`.

5.1 Authenticated-Encryption Security for `SimpleENC`

We use the authenticity bound of Section 4.3 in order to bound the advantage of the adversary in the authenticated-encryption setting. We begin with the nonce-respecting setting, where the nonce in each encryption is *guaranteed* to be unique, and then provide the bounds for the random-IV setting.

The nonce-respecting setting. We have the following theorem:

Theorem 5.1. *Denote by Π the nonce-version of* `SimpleENC`*. Then, for any nonce-respecting $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$-adversary $\mathcal{A}$, we have:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{nAE}}_{\Pi}(\mathcal{A}) \leq \frac{10 \cdot {q_E}^2}{2^{n+1}} &+ \sum_{i=1}^{q_E} \frac{{\ell_i}^2}{2^{n+1}} \\ &+ \min\left\{\begin{array}{l} q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{10 \cdot {q_E}^2}{2^{n+1}}\ , \\ q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{3n}} + \frac{9 \cdot {q_E}^2}{2^{n+1}} \end{array}\right\},\end{aligned}$$

where $\ell_{\mathsf{max}} = \max\{\ell_1, \ldots, \ell_{q_E}\}$ for $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$.

Proof. Let $\mathcal{A}$ be a $(q_E, q_D, n, \vec{\ell}_E, \vec{\ell}_D)$-adversary. Our aim is to bound:

$$\left|\Pr_K\left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot),\mathsf{Dec}_K(\cdot,\cdot,\cdot)} = 1\right] - \Pr_K\left[\mathcal{A}^{\$_K(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} = 1\right]\right|.$$

By Theorem 4.3,

$$\begin{aligned}&\left|\Pr_K\left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot),\mathsf{Dec}_K(\cdot,\cdot,\cdot)} = 1\right] - \Pr_K\left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} = 1\right]\right| \\ &\quad \leq \min\left\{\begin{array}{l} q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{10 \cdot {q_E}^2}{2^{n+1}}\ , \\ q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{3n}} + \frac{9 \cdot {q_E}^2}{2^{n+1}} \end{array}\right\}. \qquad (4)\end{aligned}$$

Based on this, we can construct a CPA-adversary $\mathcal{A}'$ who invokes $\mathcal{A}$ and forwards all encryption queries by $\mathcal{A}$ to its own encryption oracle and back. In contrast, it replies to all decryption queries with $\perp$. At the end of the execution, $\mathcal{A}'$ outputs whatever $\mathcal{A}$ outputs. It is immediate that

$$\Pr_K\left[\mathcal{A}'^{\mathsf{Enc}_K(\cdot,\cdot,\cdot)} = 1\right] = \Pr_K\left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} = 1\right]. \qquad (5)$$

Next, we bound:

$$\left|\Pr_K\left[\mathcal{A}'^{\mathsf{Enc}_K(\cdot,\cdot,\cdot)} = 1\right] - \Pr_K\left[\mathcal{A}'^{\$_K(\cdot,\cdot,\cdot)} = 1\right]\right|.$$

We construct a nonce-based CPA-adversary $\mathcal{A}''$ who attacks an encryption scheme $\Pi' = \mathsf{Enc}'$ that is defined to be counter-mode using a fresh random key for every encryption (this cannot be decrypted in principle, but this is not important here). For every encryption query (N, A, M) made by $\mathcal{A}'$, adversary $\mathcal{A}''$ works by choosing random K_M and N_1, N_2, queries its encryption oracle with (N_1, M) and gets back C, and computes the tag T using N_2 and K_M.[7] Then, $\mathcal{A}''$ returns (N, C, T) to $\mathcal{A}'$. At the end of the execution, $\mathcal{A}''$ outputs whatever $\mathcal{A}'$ outputs. The only difference between $\mathcal{A}'$'s view in its execution with oracle Enc and in the execution with $\mathcal{A}''$, is that K_E, K_M, N_1, N_2 are all random with $\mathcal{A}''$ whereas they are derived using a random permutation in the execution with oracle Enc. Thus, using the birthday bound for three derivations per query as in the proof of Theorem 4.3, we have

$$\left|\Pr_K\left[\mathcal{A}''^{\mathsf{Enc}'_K(\cdot,\cdot,\cdot)} = 1\right] - \Pr_K\left[\mathcal{A}'^{\mathsf{Enc}_K(\cdot,\cdot,\cdot)} = 1\right]\right| \leq \frac{9 \cdot {q_E}^2}{2^{n+1}}. \tag{6}$$

Next, observe that since Enc' chooses a fresh random key each time, the output of the i'th encryption can be distinguished from random with probability at most $\frac{{\ell_i}^2}{2^{n+1}}$, using the standard PRP-PRF switching lemma (where $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$), assuming that no key repeats. Thus, we have that

$$\left|\Pr_K\left[\mathcal{A}''^{\mathsf{Enc}_K(\cdot,\cdot,\cdot)} = 1\right] - \Pr_K\left[\mathcal{A}''^{\$_K(\cdot,\cdot,\cdot)} = 1\right]\right| \leq \sum_{i=1}^{q_E} \frac{{\ell_i}^2}{2^{n+1}} + \frac{{q_E}^2}{2^{n+1}}, \tag{7}$$

where the additional $\frac{{q_E}^2}{2^{n+1}}$ is to ensure that no key repeats. We use the continual key derivation (and that no key repeats) to obtain that the bound on $\texttt{CTRENC}$ is $\sum_{i=1}^{q_E} \frac{{\ell_i}^2}{2^{n+1}}$, which is much smaller than the standard bound of $\frac{(\sum_{i=1}^{q_E} \ell_i)^2}{2^{n+1}}$ for counter mode with a single key throughout.

Finally, observe that there exists an adversary $\mathcal{A}'''$ who invokes $\mathcal{A}''$ and forwards all encryption queries and replies to $\mathcal{A}'''$, and responds to all decryption queries with $\perp$, such that

$$\Pr_K\left[\mathcal{A}'''^{\$_K(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} = 1\right] = \Pr_K\left[\mathcal{A}''^{\$_K(\cdot,\cdot,\cdot)} = 1\right]. \tag{8}$$

Combining Equations (4)–(8), we have:

$$\begin{aligned} &\left|\Pr_K\left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot),\mathsf{Dec}_K(\cdot,\cdot,\cdot)} = 1\right] - \Pr_K\left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} = 1\right]\right| \\ &\leq \frac{10 \cdot {q_E}^2}{2^{n+1}} + \sum_{i=1}^{q_E} \frac{{\ell_i}^2}{2^{n+1}} + \min\left\{ \begin{array}{l} q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{10 \cdot {q_E}^2}{2^{n+1}}\ , \\ q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{3n}} + \frac{9 \cdot {q_E}^2}{2^{n+1}} \end{array} \right\}, \end{aligned}$$

thereby completing the proof. ■

Remark 5.2 (Tightness of the bound.). Before proceeding, we observe that the factor of $\frac{10 \cdot {q_E}^2}{2^{n+1}}$ appears twice; once in the authenticity bound and a second

[7] Note that although $\mathcal{A}'$ does not have a decryption oracle, it still interacts with $\Pi = \texttt{SimpleENC}$ and thus the CBC-MAC part must be computed for it.

time when proving the encryption. Thus, we "pay" for the key derivation twice, and this is actually not necessary. A tighter bound could be achieved with a less modular proof. However, since this makes only a difference of "one bit" of security, we prefer the modular proof.

The random-IV setting. The only difference between the random-IV setting and the nonce-based setting is that a random $(n-8)$-bit IV may repeat with probability ${q_E}^2/2^{n-7}$. Thus, all bounds for the nonce-respecting settings immediately translate to the random-IV setting, by adding an additional ${q_E}^2/2^{n-7}$ to the bound (and using the authenticity bound of Theorem 4.4 instead). This results in an overall addition of $2 \cdot {q_E}^2/2^{n-7} = {q_E}^2/2^{n-8}$.

Theorem 5.3. *Denote by Π the random-IV version of* `SimpleENC`. *Then, for any $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$-adversary $\mathcal{A}$, we have:*

$$\begin{aligned}\mathsf{Adv}_{\Pi}^{\mathsf{nAE}}(\mathcal{A}) \leq \frac{{q_E}^2}{2^{n-8}} + \frac{10 \cdot {q_E}^2}{2^{n+1}} + \sum_{i=1}^{q_E} \frac{{\ell_i}^2}{2^{n+1}} \\ + \min\left\{\begin{array}{l} q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{10 \cdot {q_E}^2}{2^{n+1}}, \\ q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{3n}} + \frac{9 \cdot {q_E}^2}{2^{n+1}} \end{array}\right\},\end{aligned}$$

where $\ell_{\mathsf{max}} = \max\{\ell_1, \ldots, \ell_{q_E}\}$ for $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$.

5.2 Authenticated-Encryption Security for `SimpleENC′`

As we have described in Section 4.4, the only difference between `SimpleENC` and `SimpleENC′` is that additional derivations are needed. In particular, two additional derivations are needed and this increases the factor of $\frac{9 \cdot {q_E}^2}{2^{n+1}}$ to $\frac{25 \cdot {q_E}^2}{2^{n+1}}$. Thus, we obtain comparable bounds to Theorems 5.1 and 5.3 by increasing these terms, respectively.

Theorem 5.4. *Denote by Π the nonce-version of* `SimpleENC′`. *Then, for any nonce-respecting $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$-adversary $\mathcal{A}$, we have:*

$$\begin{aligned}\mathsf{Adv}_{\Pi}^{\mathsf{nAE}}(\mathcal{A}) \leq \frac{26 \cdot {q_E}^2}{2^{n+1}} + \sum_{i=1}^{q_E} \frac{{\ell_i}^2}{2^{n+1}} \\ + \min\left\{\begin{array}{l} q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{26 \cdot {q_E}^2}{2^{n+1}}, \\ q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{3n}} + \frac{25 \cdot {q_E}^2}{2^{n+1}} \end{array}\right\},\end{aligned}$$

where $\ell_{\mathsf{max}} = \max\{\ell_1, \ldots, \ell_{q_E}\}$ for $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$.

Theorem 5.5. *Denote by Π the random-IV version of* `SimpleENC′`*. Then, for any $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$-adversary $\mathcal{A}$, we have:*

$$\mathsf{Adv}_{\Pi}^{\mathsf{nAE}}(\mathcal{A}) \leq \frac{{q_E}^2}{2^{n-8}} + \frac{26 \cdot {q_E}^2}{2^{n+1}} + \sum_{i=1}^{q_E} \frac{{\ell_i}^2}{2^{n+1}}$$
$$+ \min \left\{ \begin{array}{l} q_D \cdot \left(\frac{12 \cdot \ell_{\max}}{2^n} + \frac{64 \cdot {\ell_{\max}}^4}{2^{2n}} \right) + \frac{26 \cdot {q_E}^2}{2^{n+1}} \ , \\ q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}} \right) + \frac{{q_E}^3}{6 \cdot 2^{3n}} + \frac{25 \cdot {q_E}^2}{2^{n+1}} \end{array} \right\},$$

where $\ell_{\mathsf{max}} = \max\{\ell_1, \ldots, \ell_{q_E}\}$ for $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$.

5.3 Authenticated-Encryption Security for `SimpleENCsmall`

In the case of `SimpleENCsmall`, the key-derivation and is carried out using `CENC`. This provides better security bounds, as we will now show.

Theorem 5.6. *Denote by Π the nonce-version of* `SimpleENCsmall`*. Then, for any nonce-respecting $(q_E, q_D, \vec{\ell}_E, \vec{\ell}_D)$-adversary $\mathcal{A}$ such that $5 \cdot q_E < 2^n/67$ and ${\ell_{\max}}^2 < 2^n/67$, we have:*

$$\mathsf{Adv}_{\Pi}^{\mathsf{nAE}}(\mathcal{A}) < \frac{{q_E}^2}{2^{2.5n+1}} + \frac{q_E \cdot {\ell_{max}}^2}{2^n}$$
$$+ q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}} \right) + \frac{{q_E}^3}{3 \cdot 2^{4n}} + \frac{50 \cdot q_E}{2^n}$$

where $\ell_{\mathsf{max}} = \max\{\ell_1, \ldots, \ell_{q_E}\}$ for $\vec{\ell}_E = (\ell_1, \ldots, \ell_{q_E})$.

Proof. The outline of the proof is similar to that of Theorem 5.1, with the exception that "losing" in the case that an encryption key repeats yields a poor bound due to the small block size. Thus, as in the proof of Lemma 4.2, we utilize the fact that security is preserved unless the same encryption key and derived nonce N_2 repeat *simultaneously*. We now show the analogous bounds for each of Equations (4) to (7), with the modification as described:

1. Eq. (4) is replaced using the authenticity bound for `SimpleENCsmall` from Theorem 4.6. Thus, we have
$$\left| \Pr_K \left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot), \mathsf{Dec}_K(\cdot,\cdot,\cdot)} = 1 \right] - \Pr_K \left[\mathcal{A}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot), \perp(\cdot,\cdot,\cdot)} = 1 \right] \right|$$
$$\leq q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}} \right) + \frac{{q_E}^3}{6 \cdot 2^{5n}} + \frac{25 \cdot q_E}{2^n}$$
2. Eq. (5) remains unchanged.
3. Eq. (6) is changed due to the fact that the derivation uses `CENC`. As in the proof of Theorem 4.6, using the bounds of [13] we have that as long as $5 \cdot q_E < 2^n/67$, then
$$\left| \Pr_K \left[{\mathcal{A}''}^{\mathsf{Enc}'_K(\cdot,\cdot,\cdot)} = 1 \right] - \Pr_K \left[{\mathcal{A}'}^{\mathsf{Enc}_K(\cdot,\cdot,\cdot)} = 1 \right] \right| \leq \frac{25 \cdot q_E}{2^n}.$$

4. We change Eq. (7) and prove a better bound for a small block based on the fact that even if a key repeats, security is still preserved as long as a different nonce is used for that (repeated) key. Note that the key/nonce pair is of length $2.5n$ here, and thus the probability that a key/nonce-N_1 repeats is less than $\frac{{q_E}^2}{2^{2.5n+1}}$. Next, note that by [15], the probability that a key (of length $2n$) repeats three or more times is at most $\frac{{q_E}^3}{6 \cdot 2^{4n}}$. Assuming this does not happen, each key is used at most twice, and so the standard bound on `CTRENC` for each key will be at most $\frac{(2\cdot\ell)^2}{2^n}$. However, for each key that is repeated, the number of terms in the sum over all queries q_E is reduced by one; in the "worst case", all keys repeat twice. Thus, the term $\sum_{i=1}^{q_E} \frac{{\ell_i}^2}{2^{n+1}}$ in Eq. (7) can be upper bounded by $\frac{q_E}{2} \cdot \frac{(2\cdot\ell_{max})^2}{2^{n+1}} = \frac{q_E \cdot {\ell_{max}}^2}{2^n}$, where ℓ_{max} is the largest message encrypted. We therefore have that Eq. (7) is replaced with:
$$\left|\Pr_K\left[\mathcal{A}''^{\mathsf{Enc}_K(\cdot,\cdot,\cdot)} = 1\right] - \Pr_K\left[\mathcal{A}''^{\$_K(\cdot,\cdot,\cdot)} = 1\right]\right| \leq \frac{q_E \cdot {\ell_{max}}^2}{2^n} + \frac{{q_E}^2}{2^{2.5n+1}} + \frac{{q_E}^3}{6 \cdot 2^{4n}}.$$
5. Eq. (8) remains unchanged.

Combining all of the above, we have that

$$\begin{aligned}
\mathsf{Adv}_{\Pi}^{\mathsf{nAE}}(\mathcal{A}) \leq{} & \frac{25 \cdot q_E}{2^n} + \frac{q_E \cdot {\ell_{max}}^2}{2^n} + \frac{{q_E}^2}{2^{2.5n+1}} + \frac{{q_E}^3}{6 \cdot 2^{4n}} \\
& + q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{6 \cdot 2^{5n}} + \frac{25 \cdot q_E}{2^n} \\
< {} & \frac{{q_E}^2}{2^{2.5n+1}} + \frac{q_E \cdot {\ell_{max}}^2}{2^n} \\
& + q_D \cdot \left(\frac{48 \cdot \ell_{\max}}{2^n} + \frac{256 \cdot {\ell_{\max}}^4}{2^{2n}}\right) + \frac{{q_E}^3}{3 \cdot 2^{4n}} + \frac{50 \cdot q_E}{2^n},
\end{aligned}$$

where the last inequality is obtained by consolidating terms, and replacing $\frac{{q_E}^3}{6\cdot 2^{5n}}$ with $\frac{{q_E}^3}{6\cdot 2^{4n}}$. This completes the proof. ■

Remark 5.7. As in Remark 5.2, a tighter bound can be obtained via a less modular proof. However, the difference is not significant.

5.4 Concrete Security Bounds

In Tables 1 and 2, we present concrete bounds for different choices of n, q_E, q_D and $\vec{\ell}_E$, showing that our new modes can support a very large number of encryptions, whether these are due to many small messages or few large messages.

Acknowledgments

This research was supported by the Israel Science Foundation (grant No. 1018/16), and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Directorate in the Prime

Minister's Office. The first author was also supported by the Center for Cyber Law & Policy at the University of Haifa, in conjunction with the Israel National Cyber Directorate in the Prime Minister's Office; the Ministry of Science and Technology, Israel, and the Department of Science and Technology, Government of India.

References

1. S. Banik, S.K Pandey, T. Peyrin, Y. Sasaki, S.M. Sim, Y. Todo. GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption. In *CHES 2017*, Springer (LNCS 10529), pages 321–345, 2017.
2. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers. SIMON and SPECK: Block Ciphers for the Internet of Things. *IACR Cryptology ePrint Archive*, report #2015/585, 2015.
3. M. Bellare and R. Impagliazzo. A Tool For Obtaining Tighter Security Analyses Of Pseudorandom Function Based Constructions, With Applications To PRP To PRF Conversion. *IACR Cryptology ePrint Archive*, report #1999/024, 1999.
4. M. Bellare, K. Pietrzak and P. Rogaway. Improved Security Analyses for CBC MACs. In *CRYPTO 2005*, Springer (LNCS 3621), pages 527–545, 2005.
5. K. Bhargavan and G. Leurent. On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN. In *ACM CCS*, pages 456–467, 2016.
6. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *CHES 2007*, Springer (LNCS 4727), pages 450–466, 2007.
7. M. Dworking. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38c.pdf
8. P. Fouque, G. Martinet, F. Valette and S. Zimmer. On the Security of the CCM Encryption Mode and of a Slight Variant. In *ACNS 2008*, Springer (LNCS 5037), pages 411–428, 2008.
9. S. Gueron and Y. Lindell. Better Bounds for Block Cipher Modes of Operation via Nonce-Based Key Derivation. In *ACM CCS 2017*, pages 1019–1036, 2017.
10. S. Gueron, A. Langley and Y. Lindell. AES-GCM-SIV: Specification and Analysis. *IACR Cryptology ePrint Archive*, report 2017/168, 2017.
11. S. Gueron, A. Langley, Y. Lindell. AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption. IETF RFC 8452, https://www.rfc-editor.org/info/rfc8452.
12. T. Iwata. New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In *FSE 2006*, Springer (LNCS 4047), pages 310–327, 2006.
13. T. Iwata, B. Mennink and D. Vizár. CENC is Optimally Secure. *IACR Cryptology ePrint Archive*, report 2016/487, 2016.
14. J. Patarin. On Linear Systems of Equations with Distinct Variables and Small Block Size. In *Information Security and Cryptology*, Springer (LNCS 3935), pages 299–321, 2005.
15. K. Suzuki, D. Tonien, K. Kurosawa and K. Toyota. Birthday Paradox for Multi-collisions. Proceedings of the *9th International Conference on Information Security and Cryptology*, Springer (LNCS 4296), pages 29–40, 2006.

16. D. Whiting, R. Housley and N. Ferguson. IEEE 802.11-02/001r2: AES Encryption & Authentication Using CTR Mode & CBC-MAC. March 2002.
17. D. Whiting, R. Housley and N. Ferguson. Counter with CBC-MAC (CCM), AES Mode of Operation. Contribution to NIST, May 2002. Available from http://csrc.nist.gov/encryption/modes/proposedmodes/
18. -, Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf
19. -, NIST Lightweight Cryptography, Round 1 Candidates. https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-1-Candidates

A Preliminaries for the Full Specification

Strings of bits. This document deals with strings of bits (strings for short). The length of a string S is denoted by $|S|$, where $S > 0$ for all nonempty strings. The empty string is denoted by the symbol $\perp$, which is also used as a signal for failure. The symbol $\oplus$ denotes the bit-wise XOR operation and the symbol $\|$ denotes concatenation of strings. By convention, strings are written in "Little EndiaN' orientation. If S is a string of length $|S| = L$ its bits are written as $(S =)\ s_{L-1}s_{L-2}\dots s_0$ such that the least significant bit (s_0) is in the rightmost position and the most significant bit (s_{L-1}) is in the leftmost position. A string of k repeated zero bits is denoted by 0^k. The right-shift of the string $S = s_{L-1}\dots, s_0$ by θ positions $(\theta \leq L)$ is the string $0^\theta \| s_{L-1}\dots s_\theta$ and is denoted by $S \gg \theta$.

Encoding of integers as strings. For integers p, k such that $0 \leq p < 2^k$, $p_{[\#\mathbf{k}]}$ denotes the k-bit (string) binary representation of the integer p. For example $19_{[\#\mathbf{8}]} = 00010011$.

Blocks and block ciphers. Hereafter, E is used for denoting a block cipher with block size of n bits and key size of $\kappa \geq n$ bits. For all the cases discussed here, n and κ are powers of 2, and either $\kappa = n$ or $\kappa = 2n$. A string of n bits is called a block. The encryption of the plaintext block P under the key K is denoted by $E(K, P)$. Let $B = b_{n-1}\dots b_0$ be a block. Then the notation $[B2, B1] = B$ indicates that $B2$ and $B1$ are strings of $n/2$ bits defined by $B2 = B_{n-1}\dots B_{n/2}$, $B1 = B_{n/2-1}\dots B_0$. They are also called "half blocks". For an integer $1 \leq r < n$ the truncation of B to r bits is the string of r bits $b_{r-1}\dots b_0$, and is denoted by $Truncate(B, r)$.

Parsing a string of bits as blocks. Let $V = p_{v-1}p_{v-2}\dots p_0$ be a nonempty string of v bits where v is divisible by n, i.e., $v = \xi \times n$ for some positive integer ξ. Then, V can be parsed as a sequence of ξ blocks. It is written as $\bar{V}_{\xi-1} \| \dots \| \bar{V}_0$. Alternatively, V can be represented as a list (sequence of blocks) $V_1, \dots, V_\xi$ (with indexes increasing from left to right), where $\bar{V}_j = V_{j+1} = p_{jn+n-1}p_{jn+n-2}\dots p_{jn}$, $j = 0, \dots, \xi - 1$.

The pad10*( ) padding. Let Y be a string of bits. The mandatory padding of Y is the string $\texttt{pad10}^*(Y)$ generated by first appending the bit 1 and then zero padding the result.

Remark A.1. $\texttt{pad10}^*(S)$ padding always modifies the string S (hence the label "mandatory"). It is always a nonempty string, and in particular, consists of at least one block. If $|S|$ is divisible by n, then one block is added to S in order to generate $\texttt{pad10}^*(S)$.

B Specification of the AEAD scheme SimpleENC

$\texttt{SimpleENC} = \texttt{SimpleENC}_K(N, A, M)$ is an AEAD scheme that operates over a block cipher E for which $\kappa = n$. The scheme encrypts and authenticates a header A, a message M with a nonce N, under the key K. Note that M and/or A may be the empty strings.

Parameters. The parameters that define SimpleENC are A_{max}, M_{max}, D_{max}, τ, δ, n, κ (with $\kappa = n$) as follows. The maximal allowed lengths for the header and for the message in any single encryption are denoted by A_{max} and M_{max}, respectively. To be considered legitimate, the input strings A, M must satisfy $0 \leq |A| \leq A_{max}$, $0 \leq |M| \leq M_{max}$. The nonce N has length $|N| = (n - \tau)$, where $2 \leq \tau < n$. It is assumed that $M_{max} \leq n \cdot 2^{\delta} - 1$, and for simplicity the value $\delta = n/2$ is fixed.

For convenience, D_{max} is used hereafter in order to denote the maximal number of bits that can be processed with a given key.

Structure. SimpleENC can be viewed as a three step construction: **(a)** nonce based derivation based on CTR mode that produces an encryption key, an authentication key, and two half nonces; **(b)** encryption (in CTRENC mode) of M; **(c)** authentication of $X = \texttt{pad10}^*(A) \parallel \texttt{pad0}^*(M)$ using CBCMAC-IV.

B.1 Nonce based derivation (for $\kappa = n$)

The following derivation function $\texttt{Derive}(K, N)$ is defined.

```
Derive$(K, N)$
    Input: $K$, $N$
    Parameter: $\tau$
    (a.) $K_E = E(K, N \parallel 0_{[\#\tau]})$
    (b.) $K_{MAC} = E(K, N \parallel 1_{[\#\tau]})$
    (c.) $[N2, N1] = E(K, N \parallel 2_{[\#\tau]})$
    Output: $[N2, N1], K_E, K_{MAC}$
```

Algorithm Derive.

B.2 The SimpleENC AEAD scheme

The encryption and decryption flows of SimpleENC are illustrated in Algorithm 1 and Algorithm 2, respectively.

```
$\texttt{SimpleENC}_K(N, A, M)$
    Input: $N$, $A$, $M$
    1. $([N2, N1], K_E, K_{MAC}) = \texttt{Derive}(K, N)$
    2. if $M = \perp$ then $C = \perp$
    else
        $\texttt{IV} = N1$
        $C = \texttt{CTRENC}(K_E, \texttt{IV}, M)$
    end if
    3. $X = \texttt{pad10}^*(A) \parallel \texttt{pad0}^*(M)$
        $R = N2 \parallel 0^{n/2}$
        $Tag = \texttt{CBCMAC-IV}(K_{MAC}, R, X)$
    Output: $Tag \parallel C$
```

Algorithm 1. SimpleENC **- encryption flow.**

```
$\texttt{SimpleENC}_K(N, Tag, A, C)$
    Input: $N$, $A$, $C$, $Tag$
    1. $([N2, N1], K_E, K_{MAC}) = \texttt{Derive}(K, N)$
    2. if $C = \perp$ then $M = \perp$
    else
        $\texttt{IV} = N1$
        $M = \texttt{CTRENC}(K_E, \texttt{IV}, C)$
    end if
    3. $X = \texttt{pad10}^*(A) \parallel \texttt{pad0}^*(M)$
        $R = N2 \parallel 0^{n/2}$
        $Tag' = \texttt{CBCMAC-IV}(K_{MAC}, R, X)$
         if $Tag' = Tag$ then
        $S = M$
        else
            $S = \perp$
        end if
    Output: $S$
```

Algorithm 2. SimpleENC **- decryption flow.**

Remark B.1. Note that the decryption flow is almost identical to the encryption flow. This is advantageous in the lightweight setting, as it reduces the size of the code.

Parameters choice for concrete instantiations. The selected parameters are $\kappa = n = 128$, $\delta = n/2 = 64$, $\tau = 8$, $M_{max} = A_{max} = 2^{53} - 1$, $D_{max} = 2^{53} - 1$. The lengths of all the inputs and outputs are required to be divisible by 8, so that they can be viewed as strings of bytes.

Remark B.2. Although $\tau = 2$ suffices for the derivation, the value $\tau = 8$ is chosen so that all lengths are divisible by 8 (and considered as bytes).

Random nonces. The nonce length is 120 bits. A uniform random selection of nonces, used across q encryptions has nonce collision probability of (at most) $q^2/2^{121}$. For a limit of $q \leq 2^{46}$ encryptions, this probability is at most 2^{-29} which seems a sufficient margin for practical usage.

C Specification of the AEAD scheme `SimpleENCsmall`

$\texttt{SimpleENCsmall} = \texttt{SimpleENCsmall}_K(N, A, M)$ is an AEAD scheme that operates with a block cipher E for which $\kappa = 2n$. The scheme encrypts and authenticates a header A, a message M with a nonce N, under the key K. As above, M and/or A may be the empty strings. It is called "small" since it is primarily defined in practice for a 64-bit block, which if used in a naive way provides low security.

Parameters. The parameters that define `SimpleENCsmall` are A_{max}, M_{max}, D_{max}, τ, δ, n, κ (with $\kappa = n$) as follows. The maximal allowed lengths for the header and for the message are denoted by A_{max} and M_{max}, respectively. To be considered legitimate, the input strings A, M must satisfy $0 \leq |A| \leq A_{max}$, $0 \leq |M| \leq M_{max}$. The nonce N has length $|N| = (n - \tau)$, where $2 \leq \tau < n$. The maximal number of bits that can be processed with a given key is denoted by D_{max}. It is assumed that $M_{max} \leq n \cdot 2^{\delta} - 1$, and for simplicity the value $\delta = n/2$ is fixed.

Structure. `SimpleENCsmall` can be viewed as a three steps construction: **(a)** nonce based derivation based on XORP/CENC that produces an encryption key, an authentication key, and two half nonces; **(b)** encryption (in `CTRENC` mode) of M; **(c)** authentication of $X = \texttt{pad10}^*(A) \parallel \texttt{pad0}^*(M)$ using `CBCMAC-IV`.

C.1 Nonce based derivation (for $\kappa = 2n$)

The following derivation function $\texttt{DeriveDouble}(K, N)$, is defined.

```
DeriveDouble$(K, N)$
  Input: $K$, $N$
  Parameter: $\tau$
  $T0 = E(K, N \parallel 0_{[\#\tau]})$
  $T1 = E(K, N \parallel 1_{[\#\tau]}) \oplus T0$
  $T2 = E(K, N \parallel 2_{[\#\tau]}) \oplus T0$
  $T3 = E(K, N \parallel 3_{[\#\tau]}) \oplus T0$
  $T4 = E(K, N \parallel 4_{[\#\tau]}) \oplus T0$
  $T5 = E(K, N \parallel 5_{[\#\tau]}) \oplus T0$
  (a.) $K_E = T2 \parallel T1$
  (b.) $K_{MAC} = T4 \parallel T3$
  (c.) $[N2, N1] = T5$
  Output: $[N2, N1], K_E, K_{MAC}$
```

Algorithm `DeriveDouble`.

C.2 The `SimpleENCsmall` AEAD scheme

The encryption and decryption for `SimpleENCsmall` are illustrated in Algorithm 3 and Algorithm 4, respectively.

```
SimpleENCsmall$_K(N, A, M)$
  Input: $N$, $A$, $M$
  1. $([N2, N1], K_E, K_{MAC})$ = DeriveDouble$(K, N)$
  2. if $M = \perp$ then $C = \perp$
     else
         IV $= N1$
         $C$ = CTRENC$(K_E,$ IV$, M)$
     end if
  3. $X$ = pad10$^*(A) \parallel$ pad0$^*(M)$
     $R = N2 \parallel 0^{n/2}$
     $Tag$ = CBCMAC-IV$(K_{MAC}, R, X)$
  Output: $Tag \parallel C$
```

Algorithm 3. `SimpleENCsmall` - encryption flow.

```
$\texttt{SimpleENCsmall}_K(N, Tag, A, C)$
  **Input:** $N$, $A$, $C$, $Tag$
  1. $([N2, N1], K_E, K_{MAC}) = \texttt{DeriveDouble}(K, N)$
  2. if $C = \perp$ then $M = \perp$
    else
        $\texttt{IV} = N1$
        $M = \texttt{CTRENC}(K_E, \texttt{IV}, C)$
    end if
  3. $X = \texttt{pad10}^*(A) \parallel \texttt{pad0}^*(M)$
    $R = N2 \parallel 0^{n/2}$
    $Tag' = \texttt{CBCMAC-IV}(K_{MAC}, R, X)$
     if $Tag' = Tag$ then
        $S = M$
    else
        $S = \perp$
    end if
  **Output:** $S$
```

Algorithm 4. `SimpleENCsmall` **- decryption flow.**

Remark C.1. Note that the decryption flow is almost identical to the encryption flow. This is advantageous in the lightweight setting, as it reduces the size of the code.