CCM-SIV: Single-PRF Nonce-Misuse-Resistant Authenticated Encryption

Patrick Kresmer and Alexander Zeh
Research and Development Center
Infineon Technologies AG, Munich, Germany
{patrick.kresmer, alexander.zeh}@infineon.com

Abstract. We propose a new nonce-misuse-resistant authenticated encryption scheme, which instantiates the SIV paradigm of Rogaway and Shrimpton. In contrast to the GCM-SIV approach proposed by Gueron and Lindell, we do only use a single type of cryptographic primitive, which can be advantageous in restricted embedded devices. Furthermore, we use three independent and fixed subkeys derived from a single master key. Similar to the CCM mode, our scheme uses a combination of the CTR mode for the symmetric encryption and a MAC based on the CBC construction and is therefore called CCM-SIV. We provide a detailed security proof for our scheme. Furthermore, we outline its extension to a nonce-based key derivation as the AES-GCM-SIV approach.

Keywords: AEAD, AES-GCM, AES-GCM-SIV, AES-CCM, Nonce

1 Introduction

To efficiently achieve authenticity, integrity and confidentiality on the communication channels, authenticated encryption with associated data (AEAD) schemes are used. A famous and widely deployed AEAD system is the Galois-Counter Mode-of-Operation of the AES block cipher (AES-GCM, [15]). The security of the scheme heavily relies on the uniqueness of the nonces, since they are used as a “randomizing” input for the AES-GCM encryption. If they happen to be not unique, the security of AES-GCM is completely broken, since the plaintext and secret hashing keys can be recovered (see, e.g., [2, 12]).

GCM-SIV [7] is based on the same building blocks as AES-GCM, but it combines them through the SIV composition method [19]. The SIV paradigm endows the resulting scheme with “nonce-misuse-resistance”. A nonce-based key derivation for GCM-SIV [7] was proposed by Gueron et al. [14] (and denoted as AES-GCM-SIV) and by Iwata and Minematsu in [10]. The original security bounds of GCM-SIV were re-considered by Iwata and Seurin [11] and led to an update of [6] in July 2017 and [8].

AES Counter with CBC-MAC (AES-CCM, [4]) is, as AES-GCM, a specified cipher suite for IPsec [9], TLS [1] and relies also on the uniqueness of the nonces. AES-CCM combines the CBC-MAC and the well-known counter (CTR) mode and uses a single type of PRF (in contrast to AES-GCM where a second
function for the Galois field multiplication is required). Clearly, operations of AES-GCM can be parallelized and, e.g., Intel’s PCLMULQDQ instruction [5] allows impressive fast realizations.

Our goal is to deploy nonce-misuse resistant AEAD schemes in restricted embedded systems with fault-tolerant requirements [17], where operating elements are often realized dual (or triple) modular redundant. In this context, it is beneficial that AES-CCM (resp. CCM-SIV) does not require a second cryptographic primitive. A reference implementation of CCM-SIV can be found on gitlab [13].

Contribution. We propose a new nonce-misuse-resistant authenticated encryption scheme called CCM-SIV based on the CTR mode of operation and a dedicated CBC-based MAC. Our scheme only uses a single type of cryptographic primitive as the main building block and derives three independent subkeys from a single masterkey by a deterministic pseudo random number generator (PRNG).

We provide a detailed security proof of our scheme. The following main theorem summarizes our result:

**Theorem 1.** [Security of CCM-SIV.] Let $\tilde{\Pi}$ denote the nonce-misuse-resistant authenticated encryption (nAE) scheme CCM-SIV defined by Algorithm 8 and $\mathcal{A}$ be a probabilistic polynomial time (PPT) adversary, who wins the nAE game defined in Figure 11 with non-negligible advantage. Then there exists a wrapping PPT adversary $\mathcal{B}$ playing the PRF game, who calls $\mathcal{A}$ as a subroutine and is able to distinguish the underlying PRF $E$ from a truly random function with non-negligible advantage. Particular, it holds that

$$\text{adv}^\text{nAE}_{\tilde{\Pi}}(\mathcal{A}) \leq 2 \cdot \text{adv}^\text{PRF}_E(\mathcal{B}) + \epsilon \cdot \frac{(q_e + q_d)^2}{2} + \frac{q_e^2 \cdot p_{\text{max}}}{2} + \frac{q_d^2}{2^x},$$

where $\epsilon$ is the security bound of the underlying $\epsilon$-almost XOR universal hash function family used in the tag generation, $q_e$ the number of the (polynomially bounded) encryption queries of $\mathcal{A}$, $q_d$ the number of the (polynomially bounded) decryption queries of $\mathcal{A}$ and $x$ the block size of $E$. In the PRF security game, $\mathcal{B}$ sends his encryption queries to the oracle, which is either the PRF $E_K$ with $K \leftarrow K$ or a truly random function.

In Section 2 we introduce the notation used throughout this paper and we recall the SIV paradigm. Section 3 provides the reader with a detailed description of the CCM-SIV key generation, authenticated encryption and authenticated decryption. The security analysis of our new nonce-misuse resistant AEAD scheme CCM-SIV represents the main part of this paper and is done in four steps in Section 4. Section 6 gives a short summary and conclusion.

## 2 Notation

For a number $x \in \mathbb{N}^+$, the set $\{0, 1\}^x$ denotes the set of all bit strings of size $x$ and $\{0, 1\}^*$ denotes the countable set of all possible bit strings. The script letters $\mathcal{A}, \mathcal{B}, \mathcal{D}$ are used for algorithms, whereas all other script letters such as $X$ are
used to denote sets, e.g. $\mathcal{X} = \{0,1\}^x$. Upper-case letters such as $X$ are used for a single data block, for example $X \in \mathcal{X}$, where $\mathcal{X} = \{0,1\}^x$ and $x$ is the block size. Upper-case letters such as $E,F,H$ are used for functions. $E_{K}$ denotes the AES block cipher under key $K$. Messages, that consist of multiple blocks are denoted with bold letters, e.g. $X = (X_1, X_2, \ldots, X_m)$ for an $m$-block message, where each $X_1, X_2, \ldots, X_m \in \mathcal{X}$ and $X \in \mathcal{X}^m$. Bold letters are also used for elements in $\{0,1\}^*$. The symbol $\mathcal{X}^{\leq m_{\text{max}}}$ denotes the set of all messages, that consists of at most $m_{\text{max}}$ data blocks. Lower-case letters are used for different purposes such as indices, size values or functions. For a data block $X \in \mathcal{X}$ and $\mathcal{X} = \{0,1\}^x$, $|X|$ denotes the bit-size $x$ of $X$, whereas $|\mathcal{X}|$ denotes the number of all possible values for $X$ (i.e. $|\mathcal{X}| = 2^x$). The vector symbol $\vec{X}$ is used for vector-valued messages, which may contain several different elements of different lengths, e.g. $\vec{X} = (A,B,C)$ with $A,B \in \{0,1\}^a$ and $C \in \mathcal{X}$. The concatenation of two bit strings $X,Y \in \{0,1\}^a$ is written as $X \parallel Y$. $\Pr[\text{event | cond}]$ denotes the probability that event occurs conditioned to the case, that cond has occurred. $X \overset{\text{U}}{\leftarrow} \mathcal{X}$ means, that $X$ is chosen uniformly and independently from the set $\mathcal{X}$. We use “truly random” as a synonym for uniformly distributed and independent values. $\text{Funs}_{\vec{a},\vec{b}}$ denotes the set of all possible functions mapping $\{0,1\}^a \times \{0,1\}^b \rightarrow \{0,1\}^c$. The adversary’s advantage $\text{adv}_{\Pi}^G (A)$ denotes the absolute difference of the probability that an adversary $A$ wins a security game $G$ and the probability that an adversary $A$ loses a security game $G$. In the security game $G$, the adversary $A$ communicates with an oracle $O$, that is either the scheme $\Pi$ itself (pseudo-random oracle) or its idealized mathematical model (truly random oracle). Winning the game means, the adversary guesses the correct oracle version. $\Pr[A^F \Rightarrow \text{PR}]$ denotes the probability, that an adversary $A$ guesses “pseudo-random” after talking to a function $F$, whereas $\Pr[A^{\text{Funs}_{\vec{a},\vec{b}}} \Rightarrow \text{R}]$ denotes the probability, that she guesses “random” after talking to a uniformly chosen function out of the set $\text{Funs}_{\vec{a},\vec{b}}$.

The SIV construction [18], respectively construction A4 of Namprempre et al. [16] is a general paradigm of combining a secure IV-based symmetric encryption scheme with a vector-input PRF to obtain a secure nonce-based and nonce-misuse-resistant authenticated encryption scheme. Both primitives have two distinct and independent keys $K_1$ and $K_2$. The main idea is to use the authentication tag $T$ as the IV for the IV-based symmetric encryption. Since the MAC consists of a PRF, its output is uniformly and independently distributed as long as the inputs to the PRF are all distinct. This means, the tag looks like a randomly chosen value and hence it meets the requirements for an IV. Figure 1(a) and 1(b) illustrate the SIV encryption and decryption. For the calculation of the tag $T$, the inputs for the PRF are the additional data $A$, the plaintext $P$ and the nonce $N$. Clearly, $A$ and $P$ must be inputs to the PRF, because we want to provide authenticity/integrity for both. The nonce must also be an input to the PRF. Even if the same message $(A,P)$ is sent twice, the ciphertext-tag pair must be distinct. This obviously cannot be the case, if the nonce requirement is violated. Since the PRF and the symmetric encryption scheme are deterministic functions, if the same triple $(A,P,N)$ is encrypted twice, then the tag, the IV
and the ciphertext are also the same. Consequently, when the same nonce is used twice, then an attacker learns, if the same message was encrypted or not, but nothing beyond that.

An inherent drawback of this construction is, that the symmetric encryption has to wait for the IV, before it can start the encryption. The SIV encryption is inherently sequential: First, the PRF has to be computed and after that, the encryption of the plaintext can start. This means, the plaintext has to be temporarily stored somewhere or loaded twice. During the SIV decryption (Figure 1(b)), the decryption of the ciphertext \( C \) can immediately start. The required IV is the received tag \( T \). As soon as an block of ciphertext is decrypted, the corresponding plaintext block is fed into the PRF. The output of the PRF is compared to the received tag \( T \). If they are not equal, the invalidity symbol \( \perp \) is outputted instead of the plaintext.

3 CCM-SIV

The name “CCM-SIV” suggests, that the scheme is a straightforward SIV-variant of the CCM mode-of-operation [20]. However, the only building blocks that CCM-SIV has in common with standard CCM is, that it also uses a combination of CTR mode for symmetric encryption and a MAC based on the CBC construction. The CCM-SIV scheme, which is proven to be a secure nonce-misuse-resistant authenticated encryption scheme in Section 4, is shown in Figure 2 and 3. It consists of a combination of a special PRF based on CBC and the CTR mode for the IV-based encryption/decryption. The CCM-SIV authen-
Fig. 2: CCM-SIV encryption. The tag $T$ is calculated using a dedicated vector-input PRF based on the CBC construction over an encoded input consisting of the padded additional data $(A_1, \ldots, A_a)$, the padded plaintext $(P_1, \ldots, P_p)$ and a length block. The tag is then used as the synthetic IV for the CTR mode encryption of the plaintext.

ticated encryption (Figure 2) takes as input three 128 bit keys $K_1, K_2, K_3$, a blocks of additional data $A = (A_1, \ldots, A_a)$, $p$ blocks of plaintext data $P = (P_1, \ldots, P_p)$, a 128 bit nonce $N$ and outputs the corresponding $p$ ciphertext blocks $C = (C_1, \ldots, C_p)$ and a 128 bit message authentication tag $T$. The CCM-SIV authenticated decryption (Figure 3) takes as input again three 128 bit keys $K_1, K_2, K_3$, $a$ blocks of additional data $A = (A_1, \ldots, A_a)$, $p$ blocks of ciphertext $C = (C_1, \ldots, C_p)$, a 128 bit nonce $N$, a 128 bit message authentication tag $T$ and outputs either the corresponding $p$ blocks of plaintext $P = (P_1, \ldots, P_p)$ for a valid message, or the invalidity symbol $\perp$ for invalid messages. The three required keys $K_1, K_2, K_3$ for the underlying cryptographic primitives can be derived from a single master key $K$ with a deterministic Pseudo Random Number Generator (PRNG) based on the AES block cipher [3, Section 4.4.4]. The key generation is shown in the first function of Algorithm 1.

The constant values 0, 1, 2 (encoded as 128 bit unsigned integers) are encrypted using the master key $K$, to obtain the three required keys. In contrast to AES-GCM-SIV, these subkeys are fixed during the lifetime of a master key $K$. The key derivation has to be done only once after the negotiation of the master key $K$. If $K$ is chosen uniformly at random, then $E_K$ is a secure PRF. As long as the inputs to $E_K$ are distinct, the corresponding outputs are uniformly distributed and independent. This is the case for constant values 0, 1, 2.
Fig. 3: CCM-SIV decryption. First, the ciphertext \((C_1, \ldots, C_p)\) is decrypted by using the received tag \(T\) as the IV for CTR mode. The MAC is then calculated the same was as in the CCM-SIV encryption. The calculated tag is compared to the received tag \(T\) to check for validity.

Algorithm 1 CCM-SIV

\begin{align*}
1: \text{function } & \text{KeyGeneration( )} \\
2: & K \leftarrow \mathcal{K} \\
3: & K_1 \leftarrow E_K(0) \\
4: & K_2 \leftarrow E_K(1) \\
5: & K_3 \leftarrow E_K(2) \\
6: & \text{return } K = (K_1, K_2, K_3) \\
7: \text{function } & \text{Encryption}_K(A, P, N) \\
8: & T \leftarrow E_{K_1}(\text{CBC}_{K_1}(e(A, P)) \oplus N) \\
9: & C \leftarrow \text{CTR}_{K_2}(P) \\
10: & \text{return } (C, T) \\
11: \text{function } & \text{Decryption}_K(A, C, N, T) \\
12: & P \leftarrow \text{CTR}_{K_1}(C) \\
13: & T' \leftarrow E_{K_2}(\text{CBC}_{K_1}(e(A, P)) \oplus N) \\
14: & \text{if } T = T' \text{ then} \\
15: & \quad \text{return } P \\
16: & \text{else} \\
17: & \quad \text{return } \bot
\end{align*}
Authentication. The authentication is done by first applying an encoding function \( e \) on the input \((A, P)\). This function appends \( r \) zeros to the last block \( A_a \) of the additional data and \( s \) zeros to the last block \( P_p \) of the plaintext. If any of these are already a multiple of the block size, then no zeros are appended. Similarly to AES-GCM-SIV, the encoding function adds another block called the length block. This block contains the concatenation of the original bit-length \(|A|\) with the original bit-length \(|P|\), both encoded as 64-bit unsigned integers yielding a full 128-bit block. To obtain the message tag \( T \), the set of blocks \( A_1, \ldots, (A_a \parallel 0^r), P_1, \ldots, (P_p \parallel 0^s), (|A| \parallel |P|) \) is applied to the CBC construction using key \( K_1 \). The output of the CBC construction is XORed with the nonce \( N \) and finally encrypted with a single block cipher invocation using key \( K_2 \).

Encryption/Decryption. The encryption/decryption is done in CTR mode by using the tag \( T \) as the IV. Here, the first counter block \( I_1 \) is simply initialized with the tag \( T \). Only the least significant 32 bits are used as the counter field. This block (i.e., the IV) is simply initialized with the 128-bit tag \( T \). For every subsequent counter block \( I_2, I_3, \ldots \), the least significant 32 bits are incremented using \( \text{mod}2^{32} \) arithmetic. The maximum plaintext length for the CTR mode is therefore \( 2^{32} \) blocks, because then the 32-bit counter will wrap around, causing the two-time-pad problem. Note, that if the last plaintext block \( P_p \) is smaller than the blocksize, not the full keystream \( E_K(I_p) \) is needed to encrypt/decrypt \( P_p \) resp. \( C_p \). In the figures, the truncation of \( E_K(I_p) \) to \(|P_p|\) bits is omitted for reasons of simplicity. The latter two functions in Algorithm 1 describe the CCM-SIV authenticated encryption and authenticated decryption. These two functions are an instantiation of the general SIV paradigm shown in Figures 1(a) and 1(b) from the beginning of this section. Even though the CCM-SIV scheme is a constructed mixture of standard CCM, the SIV paradigm and the AES-GCM-SIV scheme - which are all proven secure - a proper security proof needs to be done for this mixture.

4 The Security of CCM-SIV

Based on the assumption, that the raw CBC-construction described by Algorithm 2 is a secure PRF for prefix-free messages [3, Section 6.4.1] and that the CTR mode-of-operation is a secure IV-based symmetric encryption scheme [7], the security proof of CCM-SIV is done in four steps:

1. The prefix-free PRF family CBC-E is an \( \epsilon \)-almost XOR universal hash function family (\( \epsilon \)-AXU) for messages, that are a multiple of the block size \( x \).
   Since we cannot guarantee prefix-free messages, a less conservative model than a PRF is introduced here. It is shown, that the XOR-distance between two outputs of the CBC construction is almost truly random.
2. An injective encoding on the input yields an \( \epsilon \)-AXU family for additional data and plaintext of any bit size.

---

\(^1\)The first counter block is denoted as \( I_1 \), because there is no \( I_0 \) needed for the CTR encryption of the MAC, as it is the case for standard CCM.
Since messages are not always multiples of the block size, this step provides a solution to handle messages of any bit-length. Furthermore, vector-valued messages are introduced here.

3. The composition of this bit-wise $\epsilon$-AXU, a XOR operation and a single block PRF yields a secure vector-input PRF over the additional data, the plaintext and the nonce.

This step returns to the original notion of a PRF by composing several building blocks. Now, we have a PRF for vector inputs, that can be of any bit-size.

4. Combining the CTR mode-of-operation with the vector-input PRF from Step 3 yields a secure nonce-misuse-resistant authenticated encryption scheme.

In this last step, the vector-input PRF from Step 3 and the CTR mode-of-operation are combined according to the SIV paradigm. It is proven, that this composition provides confidentiality, authenticity and integrity, even if the same nonce repeats.

4.1 Step 1: The raw CBC construction is an $\epsilon$-AXU

In this step we prove, that the CBC-$E$ construction is an $\epsilon$-almost XOR universal hash function family for messages, that are a multiple of the block size $x$ of the underlying PRF $E$.

\begin{algorithm}
\begin{algorithmic}[1]
\Require $K \xleftarrow{} \mathcal{K}$
\Function{$\text{CBC-}E_K(\mathbf{X})$}{X consists of $m$ blocks.}
\State $Y \leftarrow 0$
\For{$i \leftarrow 1$ to $m$}
\State $Y \leftarrow E_K(Y \oplus X_i)$
\EndFor
\State \Return $Y$
\EndFunction
\end{algorithmic}
\caption{CBC-$E$ construction.}
\end{algorithm}

The proof builds upon the fact, that the raw CBC construction CBC-$E$ described by Algorithm 2 is a secure PRF family for prefix-free inputs, that are a multiple of the block size $x$ [3, Section 6.4.1]. The set of all input messages is called prefix-free, if no message is a proper prefix of another message:

\textbf{Definition 1.} [Proper Prefix] A message $\mathbf{X} = (X_1, \ldots, X_m)$ is a proper prefix of a message $\mathbf{X'} = (X'_1, \ldots, X'_m)$ with $1 \leq m < m' \leq m_{\text{max}}$, if for every $1 \leq i \leq m$ it holds that $X_i = X'_i$.

Following Boneh [3, Section 6.4.1], Definition 2 states that CBC-$E_K$ is indistinguishable from a truly random function, if the messages are prefix-free and $K$ is chosen uniformly from the key space $\mathcal{K}$. The related security game PF-PRF works as follows: The PPT adversary $A$ queries her oracle $O$ with at most $q$ prefix-free messages $\mathbf{X}_1, \ldots, \mathbf{X}_q \in \mathcal{X}^{\leq m_{\text{max}}}$, where the oracle is either CBC-$E_K$ with $K \xleftarrow{} \mathcal{K}$ or a truly random function with the same output size $x$. $A$ wins the game, if she is able to distinguish both oracle versions with non-negligible
probability. The wrapping PPT algorithm \( \mathcal{B} \) plays the PRF game attacking the underlying PRF \( E_K \), where he queries at most \( m_{\text{max}} \) message blocks for every \( \mathcal{A} \)-query. \( \mathcal{B} \)'s oracle is either the PRF \( E_K \) with a freshly chosen \( K \leftarrow \mathcal{K} \) or a truly random function with the same output size \( x \). \( \mathcal{B} \) wins the PRF game, if he is able to distinguish both oracle versions with non-negligible probability:

**Definition 2.** [CBC-\( E \)'s security as a PRF] Let \( \mathcal{A} \) be a PPT algorithm, who is able to distinguish CBC-\( E \) from a random function in the PF-PRF game with non-negligible advantage. Then there exists a PPT adversary \( \mathcal{B} \), who distinguishes \( E \) from a random function in the PRF game with non-negligible advantage. Particularly, it holds that

\[
\text{adv}^{\text{PF-PRF}}_{\text{CBC-} E} (\mathcal{A}) \leq \text{adv}^{\text{PRF}}_E (\mathcal{B}) + \frac{q^2 m_{\text{max}}^2}{2^x}.
\]

Based on the assumption, that the AES block cipher \( E \) is secure, such an \( \mathcal{A} \) cannot exist, since this would contradict (1).

Given a secure PRF family for prefix-free inputs that are a multiple of the block size \( x \) such as CBC-\( E \), we now show that this has a property called **difference unpredictability**, **uniform difference property** or simply **\( \epsilon \)-Almost XOR Universality**.

**Definition 3.** [\( \epsilon \)-Almost XOR Universal Hash Function Family.] An \( \epsilon \)-Almost XOR Universal Hash Function Family (\( \epsilon \)-AXU) is a function family \( H : \mathcal{K} \times \{0, 1\}^* \rightarrow X \) with \( H_K : \{0, 1\}^* \rightarrow X \mid \forall K \in \mathcal{K} \), \( X = \{0, 1\}^x \), \( \mathcal{K} = \{0, 1\}^K \), such that for all \( X, X' \in \{0, 1\}^* \), \( X \neq X' \), all \( H_K \) and all \( \delta \in X \) it holds that

\[
\Pr [H_K(X) \oplus H_K(X') = \delta] \leq \epsilon
\]

with \( 0 < \epsilon < 1 \). Or in other words, if the probability, that the XOR-difference of two hash values is \( \delta \), is at most \( \epsilon \).

The related XUHF game defined in Figure 4 works as follows: In the beginning, the challenger chooses a key \( K \) uniformly from the key space \( \mathcal{K} \) and then sets the oracle to \( H_K \). The adversary sends two distinct messages \( X_1, X_2 \in X \leq m_{\text{max}} \) together with a guess \( \delta \in X \) for the XOR-difference of the hash values. The challenger checks, if \( H_K(X_1) \oplus H_K(X_2) = \delta \) and sets the flag win in this case. Obviously, \( \mathcal{A} \) wins the game if \( \text{win} = 1 \) and loses the game if \( \text{win} = 0 \). \( \mathcal{A} \)'s advantage in this game is defined to be \( \Pr[\text{win} = 1] \) or short \( \Pr[\text{win}] \). Upper-bounding this advantage then yields a measure for the security as an \( \epsilon \)-AXU:

**Definition 4.** [\( \epsilon \)-AXU security.] Let \( H \) be a function family as in Definition 3. Then \( H \) is a computationally secure \( \epsilon \)-Almost XOR UHF family (\( \epsilon \)-AXU), if there exists a negligible \( \epsilon \), such that for all PPT adversaries \( \mathcal{A} \) playing the XUHF game defined in Figure 4 it holds that

\[
\text{adv}^{\text{XUHF}}_H (\mathcal{A}) \leq \epsilon.
\]

The following Lemma proves, that CBC-\( E \) is also an \( \epsilon \)-AXU family:
Fig. 4: The challenger chooses a function family instance $H_K$ by uniformly choosing a key $K$ from the set $K$. Then the adversary $A$ outputs two distinct messages $X_1, X_2$ and a guess $\delta$ for the XOR-difference. The challenger evaluates $H_K$ on $X_1, X_2$ and sets the flag win to 1, if $\delta$ was right.

**Lemma 1.** [CBC-E’s security as an ϵ-AXU.] Let $A$ be a PPT adversary, who wins the XUHF game defined in Figure 4 with non-negligible advantage. Then there exists a wrapping PPT adversary $B$, who distinguishes CBC-E from a random function in the PF-PRF game with non-negligible advantage. Particularly, it holds that

$$\text{adv}^{\text{XUHF}}_{\text{CBC-E}}(A) \leq \text{adv}^{\text{PF-PRF}}_{\text{CBC-E}}(B) + \frac{1}{2^\epsilon}.$$

**Proof.** The proof works by describing a wrapping adversary $B$, which tries to win the PF-PRF game by making use of a hypothetical adversary $A$, which is playing the XUHF game. Figure 5 shows the framework for $B$. Adversary $A$

Fig. 5: The wrapping algorithm $B$ tries to win the PF-PRF game on the outside by using the XUHF-adversary $A$ as a subroutine. $B$ sets the flag guess $b'$ to either PR or R, depending on whether $O(\hat{X}_1)$ and $O(\hat{X}_2)$ collide or not.
is playing the XUHF game and therefore outputs two distinct messages $X_1, X_2$ together with an expected XOR-difference $\delta$. Adversary $B$’s goal is to distinguish between CBC-E and a random choice out of $\text{Func}_{\text{mix}}$. For this, $B$ may do the following: First, he appends $\delta$ to $X_1$ and a zero block to $X_2$. With this, the modified messages $X_1 = X_1 || \delta$ and $X_2 = X_2 || 0^x$ will cause a detectable collision in $O(X_1)$ and $O(X_2)$, if $O = \text{CBC-E}_K$ and $\delta$ was correctly guessed by $A$. This is because $\delta = \text{CBC-E}_K(X_1) \oplus \text{CBC-E}_K(X_2)$ and therefore

$$
\text{CBC-E}_K(X_1) \oplus \text{CBC-E}_K(X_2) = E_K(\text{CBC-E}_K(X_1) \oplus \delta) \oplus E_K(\text{CBC-E}_K(X_2) \oplus 0^x) = E_K(\text{CBC-E}_K(X_2)) \oplus E_K(\text{CBC-E}_K(X_2) \oplus 0^x) = 0
$$

The problem with this is, that $B$ cannot simply pass $X_1 || \delta$ and $X_2 || 0^x$ to his challenger, since this message pair might not be prefix-free. So $B$ has to make the pair prefix-free first. For this, we can make use of the extension property of CBC-E [3, Section 7.2.2]:

**Definition 5.** [Extension Property.] A PRF Family $F : K \times \mathcal{X}^{\leq m_{\text{max}}} \rightarrow \mathcal{X}$ with $F = \{F_K : \mathcal{X}^{\leq m_{\text{max}}} \rightarrow \mathcal{X} \mid \forall K \in \mathcal{K}\}$, $\mathcal{X} = \{0,1\}^x$ and $K = \{0,1\}^k$ is “extendable” or has an “extension property”, if

$$
F_K(X) = F_K(X') \Rightarrow F_K(X||A) = F_K(X'||A)
$$

for all $X, X' \in \mathcal{X}^{\leq m_{\text{max}}}$, $X \neq X'$, all $A \in \mathcal{X}$ and all $K \in \mathcal{K}$.

Obviously, CBC-E is extendable: Consider two distinct messages $X$ and $X'$ with $\text{CBC-E}_K(X) = \text{CBC-E}_K(X')$ and some $A \in \mathcal{X}$. Then

$$
\text{CBC-E}_K(X||A) = \text{CBC-E}_K(\text{CBC-E}_K(X) \oplus A) = \text{CBC-E}_K(\text{CBC-E}_K(X') \oplus A) = \text{CBC-E}_K(X'||A).
$$

To make the two queries $X_1 || \delta$ and $X_2 || 0^x$ prefix-free, adversary $B$ uses the extension property in the following way (the pseudo code for the wrapping adversary $B$ is given in Algorithm 3).

<table>
<thead>
<tr>
<th>Algorithm 3 Pseudo Code for $B$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: receive $X_1, X_2, \delta$ from $A$</td>
</tr>
<tr>
<td>2: choose any $A$ s.t. $A \notin X_1, X_2$ and $A \neq \delta$</td>
</tr>
<tr>
<td>3: $X_1 \leftarrow X_1</td>
</tr>
<tr>
<td>4: $X_2 \leftarrow X_2</td>
</tr>
<tr>
<td>5: send $X_1, X_2$ to challenger</td>
</tr>
<tr>
<td>6: if $O(X_1) = O(X_2)$ then</td>
</tr>
<tr>
<td>7: return $b' \leftarrow \text{PR}$</td>
</tr>
<tr>
<td>8: else</td>
</tr>
<tr>
<td>9: return $b' \leftarrow \text{R}$</td>
</tr>
</tbody>
</table>

11
Upon receiving queries $X_1, X_2$ from $A$ and appending $\delta$ and $0^x$, $B$ chooses a block $A$ that is not an element of $X_1$ or $X_2$ and is neither $\delta$ nor $0^x$. Such an $A$ must exist in an asymptotic setting, because the PPT adversary $A$ outputs at most a polynomially bounded number of blocks and the number of possible block values grows exponentially in $x$. This $A$ is appended to both messages, yielding a prefix-free pair $X_1 = X_1||\delta||A$ and $X_2 = X_2||0^x||A$. Adversary $B$ then queries his challenger with this prefix-free pair. If the two messages $X_1||\delta$ and $X_2||0^x$ would have caused a collision $\text{CBC-E}_K(X_1||\delta) = \text{CBC-E}_K(X_2||0^x)$, then due to the extension property of $\text{CBC-E}$, the extended messages $X_1 = X_1||\delta||A$ and $\hat{X}_2 = X_2||0^x||A$ must also cause a collision $\text{CBC-E}_K(X_1) = \text{CBC-E}_K(\hat{X}_2)$. Once he has sent the modified messages $X_1, \hat{X}_2$ to his challenger and received $O(X_1), O(\hat{X}_2)$, $B$ checks if $O(X_1)$ collides with $O(\hat{X}_2)$. If this is the case, he outputs $\text{PR}$. Otherwise, he outputs $\text{B}$. Since we are assuming that $A$ guesses the right $\delta$ for $\text{CBC-E}_K(X_1) \oplus \text{CBC-E}_K(X_2)$ with non-negligible probability, the constructed adversary $B$ is then also able to distinguish $\text{CBC-E}$ from a random function with non-negligible probability because of the following collision:

$$
\text{E}_K(\text{E}_K(\text{CBC-E}_K(X_1) \oplus \delta) \oplus A)
= \text{E}_K(\text{E}_K(\text{CBC-E}_K(X_1) \oplus \text{CBC-E}_K(X_1) \oplus \text{CBC-E}_K(X_2)) \oplus A)
= \text{E}_K(\text{E}_K(\text{CBC-E}_K(X_2) \oplus 0) \oplus A)
$$

(2)

To relate the advantages of $A$ and $B$, first observe that

$$
\text{adv}_{\text{CBC-E}}^{\text{PP-PRF}}(B) = \left| \text{Pr}[B_{\text{Funs}_{m \rightarrow x}} \Rightarrow \text{R}] - \text{Pr}[B_{\text{CBC-E}} \Rightarrow \text{R}] \right|
= 1 - \text{Pr}[B_{\text{Funs}_{m \rightarrow x}} \Rightarrow \text{PR}] - \left( 1 - \text{Pr}[B_{\text{CBC-E}} \Rightarrow \text{PR}] \right)
\geq \text{Pr}[B_{\text{CBC-E}} \Rightarrow \text{PR}] - \text{Pr}[B_{\text{Funs}_{m \rightarrow x}} \Rightarrow \text{PR}].
$$

(3)

If $B$ is interacting with a uniform choice out of $\text{Funs}_{m \rightarrow x}$, the values $O(X_1)$ and $O(\hat{X}_2)$ are uniformly distributed in $\mathcal{X}$ and therefore

$$
O(X_1) = O(\hat{X}_2)
$$

holds with probability $1/|\mathcal{X}|$ and thus

$$
\text{Pr}[B_{\text{Funs}_{m \rightarrow x}} \Rightarrow \text{PR}] = \frac{1}{2^x}.
$$

(4)

$A$’s guessed difference $\delta$ is correct with probability $\text{adv}_{H}^{\text{XUHF}}(A)$. If $B$ is interacting with $\text{CBC-E}$, a correct $\delta$ causes the collision in (2). Hence $B$ outputs $\text{PR}$ with at least this probability:

$$
\text{Pr}[B_{\text{CBC-E}} \Rightarrow \text{PR}] \geq \text{adv}_{H}^{\text{XUHF}}(A).
$$

(5)

Combining (3), (4) and (5) concludes the proof:

$$
\text{adv}_{\text{CBC-E}}^{\text{PP-PRF}}(B) \geq \text{adv}_{H}^{\text{XUHF}}(A) - \frac{1}{2^x}.
$$

□
4.2 Step 2: Injective encoding for bit-wise vector-valued inputs

The function family $\text{CBC-E}^* : K \times \mathcal{X}^{\leq m_{\text{max}}} \to \mathcal{X}$ was proved to be a secure $\epsilon$-AXU family for messages, that are a multiple of the block size $x$ and at most $m_{\text{max}}$ blocks long. This step constructs a secure $\epsilon$-AXU family $\text{CBC-E}^* : K \times \mathcal{X} \times \mathcal{P} \to \mathcal{X}$, that takes as input an additional data part $A \in \mathcal{A}$ and a plaintext $P \in \mathcal{P}$, where $\mathcal{A} = \mathcal{P} = \{0, 1\}^*$, but with the following restriction on the maximum length of $A$ and $P$:

$$\left\lceil \frac{|A|}{x} \right\rceil + \left\lceil \frac{|P|}{x} \right\rceil + 1 \leq m_{\text{max}}.$$  (6)

Without loss of generality, additionally assume that $|A|, |P| \leq 2^{x/2}$ bit, so that both the bit-length of $A$ and the bit-length of $P$ can be expressed as $x/2$ bit integer values.

**Algorithm 4** Injective Encoding Function $e$.

1: function $e(A, P)$
2: $r \leftarrow x - |A_a|$
3: $s \leftarrow x - |P_p|$
4: $L \leftarrow |A| \parallel |P| : L \in \{0, 1\}^*$
5: $X \leftarrow A \parallel 0^r \parallel P \parallel 0^s \parallel L$
6: return $X = (X_1, \ldots, X_m)$

Fix an input tuple $(A, P)$ according to the restrictions described above and let $K \leftarrow \mathcal{K}$, then $\text{CBC-E}^*_K(A, P)$ is constructed as follows: $\text{CBC-E}^*_K(A, P) = \text{CBC-E}_K(e(A, P))$, with an injective encoding function $e : \{0, 1\}^* \times \{0, 1\}^* \to \mathcal{X}^{\leq m_{\text{max}}}$. The encoding function $e$ is defined by Algorithm 4 and $\text{CBC-E}^*$ is illustrated in Figure 6. The additional data $A$ consists of $a$ blocks $A_1, \ldots, A_a$,

![Fig. 6: The CBC-E* function is the composition of the injective encoding function e and the standard CBC construction. The additional data and the plaintext are padded with zeros and a length block containing the bit lengths of A and P is appended to the data, before applying the CBC construction.](image)

where the last block $A_a$ might be incomplete, i.e. $|A_a| \leq x$. The same holds for $P = (P_1, \ldots, P_p)$ with $|P_p| \leq x$. The injective encoding function $e$ fills $A_a$ with $r$ zeros, such that $|(A_a \parallel 0^r)| = x$ and fills $P_p$ with $s$ zeros, such that $|(P_p \parallel 0^s)| = x$. These two parts are concatenated together with a length block.
$L$ that contains the bit-length of $A$ and the bit-length of $P$, both encoded as $x/2$ bit values. The final output $X$ then contains $m$ blocks with

$$m = a + p + 1 \leq m_{\text{max}}$$

and hence is a proper input for CBC-E. The following corollary states, that CBC-$E^*$ is also a secure $\epsilon$-AXU family.

**Corollary 1.** [CBC-$E^*$’s security as an $\epsilon$-AXU family.] If the CBC-E function family shown in Algorithm 2 is a secure $\epsilon$-AXU family for messages that are a multiple of the block size $x$ and that are at most $m_{\text{max}}$ blocks long, then the composition CBC-$E^*(A, P) = \text{CBC-E}(e(A, P))$ is also a secure $\epsilon$-AXU family for distinct messages $(A, P)$ that fulfills (6).

**Proof.** First, let us verify that $e$ is indeed an injective function, i.e. it never maps two distinct messages $(A, P), (A', P')$ to the same output $X = X'$. Consider two distinct messages $(A, P), (A', P')$ and their corresponding output $X$ and $X'$. We assume that $X = X'$. Then it must hold that $L = L'$, since these are the last blocks in $X$ resp. $X'$. Hence $|A| = |A'|$ and $|P| = |P'|$. From this follows, that $r = s'$ and $s = s'$. The only parts of $X$ and $X'$, that are left, are $A$ and $P$ itself. Therefore, they must be equal if $X = X'$. This contradicts the assumption, that $(A, P), (A', P')$ are distinct. Therefore, $e$ is an injective function, which preserves the distinctness between two messages. Furthermore, $e$ never outputs messages that are longer than $m_{\text{max}}$ blocks due to its restriction on $A$ and $P$. By Lemma 1, applying the CBC-E construction on distinct inputs with a maximum length of $m_{\text{max}}$ blocks yields a secure $\epsilon$-AXU family. \qed

### 4.3 Step 3: Construction of a PRF for MAC calculation

For the construction of a PRF over a nonce $N \in \mathcal{X}$, additional data $A \in \{0, 1\}^*$ and plaintext $P \in \{0, 1\}^*$, this step follows the same approach as Gueron and Lindell used for GCM-SIV [7]. There, the PRF is constructed by the composition $F_{K_2}(H_{K_1}(e(A, P)) \oplus N)$, where $F_{K_2} : \mathcal{X} \rightarrow \mathcal{X}$ is a PRF instance using an encryption key $K_2 \in \mathcal{K}$, $H_{K_1} : \mathcal{X}^{\leq m_{\text{max}}} \rightarrow \mathcal{X}$ is an $\epsilon$-AXU instance using a hashing key $K_1 \leftarrow \mathcal{K}$ and $e : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathcal{X}^{\leq m_{\text{max}}}$ is the injective encoding function from the previous step. For CCM-SIV, the $\epsilon$-AXU family $H$ is CBC-E*$ and the PRF $F$ is the AES block cipher $E$. The composition $E_{K_2}(\text{CBC-E}_{K_1}(e(A, P)) \oplus N)$ is denoted as $F_{K_2}^a$, and illustrated in Figure 7. Formally, $F^* : \mathcal{K}^2 \times \mathcal{A} \times \mathcal{P} \times \mathcal{N} \rightarrow \mathcal{T}$ with $A = \mathcal{P} = \{0, 1\}^*$, $\mathcal{N} = \mathcal{T} = \mathcal{K} = \mathcal{X}$ and $F^* = \{F^*_{K_1, K_2} : \mathcal{A} \times \mathcal{P} \times \mathcal{N} \rightarrow \mathcal{T} | \forall K_1, K_2 \in \mathcal{K}\}$. The message $X = (X_1, \ldots, X_m)$ in Figure 7 is the output of the injective encoding function $e$ and contains the additional data $A$, the plaintext $P$ and the length block $L$. For a blocks of additional data $A$ and $p$ blocks of plaintext $P$, it holds that $m = a + p + 1 \leq m_{\text{max}}$. The output of CBC-E$_{K_1}$ is XORed with the nonce $N$ and then encrypted by $E_{K_2}$ to obtain the tag $T$. To analyze the security of this composition as a vector-input PRF for bit-wise inputs, Figure 8 defines the
Fig. 7: The secure PRF family $F^*$ over the nonce $N$ and a message $(X_1,\ldots,X_m)$ is accomplished by a composition of CBC-$E_{K_1}$ as an $\epsilon$-AXU, a XOR operation and a PRF $E_{K_2}$.

Fig. 8: The oracle is either the PRF $F^*_{K_1,K_2}$ with $K_1,K_2$ chosen uniformly and independent or a uniform choice out of $\text{Funs}_{*,*,z\to z}$. The adversary $A$ sends at most $q$ vector valued queries $\vec{Q}_1,\ldots,\vec{Q}_q$ to the oracle and outputs a flag guess $b'$ at the end.
related security game PRF*. In this game, the adversary $A$ outputs $q$ distinct vector valued queries $\vec{Q}_1, \ldots, \vec{Q}_q$, of which each $\vec{Q}_i$ with $1 \leq i \leq q$ is of the form $\vec{Q}_i = (A_i, P_i, N_i)$. The challenger sets the oracle to either a PRF instance $F_{K_1, K_2}^*$ with $K_1, K_2$ chosen uniformly and independently from the key space, or to a truly random function. The symbol $\text{Funs}_{*,*,x \rightarrow x}$ denotes the set of all functions mapping $\{0, 1\}^* \times \{0, 1\}^* \times X \rightarrow X$. $A$’s advantage is defined the same way as in the PRF game:

$$\text{adv}_{F^*}^{\text{PRF}}(A) = \left| \Pr[A_{\text{Funs}_{*,*,x \rightarrow x}} \Rightarrow R] - \Pr[A_{F^*} \Rightarrow R] \right|.$$  

If this term can be upper-bounded by a negligible function for all PPT adversaries $A$, then $F^*$ is considered to be a secure vector-input PRF over the additional data $A$, the plaintext $P$ and the nonce $N$.

**Lemma 2.** $[F^*$’s security as a PRF.] Let $F^*$ be the composition of an $\epsilon$-AXU family and a PRF family $E$ as described above, and let $A$ be a PPT adversary, who wins the PRF* game defined in Figure 8 with non-negligible advantage. Then there exists a wrapping PPT adversary $B$, who distinguishes $E$ from a random function in the PRF game. Particularly, it holds that

$$\text{adv}_{F^*}^{\text{PRF}}(A) \leq \text{adv}_{E}^{\text{PRF}}(B) + \epsilon \cdot \frac{q^2}{2}.$$  

Lemma 2 states, that an efficient adversary $A$ breaking $F^*$ cannot exist, because otherwise we would be able to construct an efficient adversary $B$ breaking $E$ and this would be a contradiction to the assumption, that AES is secure.

**Proof** (Proof of Lemma 2.). Figure 9 shows the framework for $B$ and Algorithm 5 his pseudo code. The adversary $B$ chooses the key $K_1$ for CBC-$E$ by himself. When he receives the $i$th query $\vec{Q}_i$ from $A$, he encodes $A_i, P_i$ using the injective encoding function $e$, applies CBC-$E_{K_1}$, XORs the resulting hash value with the nonce $N_i$ and sends the result to his PRF challenger.

**Algorithm 5** Pseudo Code for $B$.

1: $K_1 \leftarrow \mathcal{K}$

2: for all queries $\vec{Q}_i$ from $A$ with $1 \leq i \leq q$ do
3: $X_i \leftarrow \text{CBC-}E_{K_1}(e(A_i, P_i)) \oplus N_i$
4: send $X_i$ to challenger
5: send $O(X_i)$ to $A$
6: return same $b'$ as $A$

Depending on $b$, the challenger either applies $E_{K_2}$ or a truly random function and sends back $O(X_i)$. $B$ then simply passes $O(X_i)$ to $A$. At the end, $B$ outputs the same flag $b'$ as $A$. By definition, $B$’s advantage in the PRF game and $A$’s advantage in the PRF* game are

$$\text{adv}_{E}^{\text{PRF}}(B) = \left| \Pr[B_{\text{Funs}_{*,*,x \rightarrow x}} \Rightarrow R] - \Pr[B_{E} \Rightarrow R] \right|,$$  

$$\text{adv}_{F^*}^{\text{PRF}}(A) = \left| \Pr[A_{\text{Funs}_{*,*,x \rightarrow x}} \Rightarrow R] - \Pr[A_{F^*} \Rightarrow R] \right|.$$
If $b = \text{PR}$, $B$ perfectly simulates the function $F_{K_1,K_2}^*$, because in this case

$$O(X_i) = E_{K_2} \left( \text{CBC-} E_{K_1}(e(A_i,P_i)) \oplus N_i \right) \overset{\text{def}}{=} F_{K_1,K_2}^*$$

with $K_1,K_2$ chosen uniformly and independently from $\mathcal{K}$. Since $B$ outputs the same flag $b'$ as $A$, it holds that

$$\Pr [B^E \Rightarrow R] = \Pr [A^{F^*} \Rightarrow R]. \quad (9)$$

Subtracting (8) and (7), applying the triangle inequality and using (9) yields

$$\text{adv}_{E}^{F_{PRF}^*}(A) - \text{adv}_{E}^{F_{PRF}}(B) \leq \left| \Pr [A^{\text{Funs}_{x,x} \Rightarrow R}] - \Pr [A^{F^*} \Rightarrow R] \right| - \left| \Pr [B^{\text{Funs}_{x,x} \Rightarrow R}] - \Pr [B^E \Rightarrow R] \right| \leq \left| \Pr [A^{\text{Funs}_{x,x} \Rightarrow R}] - \Pr [B^{\text{Funs}_{x,x} \Rightarrow R}] \right|. \quad (10)$$

If $b = \text{R}$, $B$ perfectly simulates a uniform choice out of $\text{Funs}_{x,x}^{*,*} \Rightarrow x$, if and only if $O(X_1), \ldots, O(X_q)$ are all independent and uniformly distributed. This is only the case, if the inputs $X_1, \ldots, X_q$ for the function $O \overset{\text{def}}{=} \text{Funs}_{x,x}$ are all distinct. Let coll denote the event, that there exists at least one pair of indices $i,j$ with $i \neq j$ and $i,j \in \{1,\ldots,q\}$ such that

$$X_i = X_j \iff \text{CBC-} E_{K_1}(e(A_i,P_i)) \oplus N_i = \text{CBC-} E_{K_1}(e(A_j,P_j)) \oplus N_j.$$ 

If coll does not occur, then all inputs to $O \overset{\text{def}}{=} \text{Funs}_{x,x}$ are distinct and $B$ perfectly simulates a uniform choice out of $\text{Funs}_{x,x}^{*,*}$. This is, because the output
distribution of $O \xleftarrow{} \text{Funs}_{x \rightarrow x}$ is uniformly distributed. Using this conditional probability space, the absolute difference in (10) can be rewritten as

\[
\left| \Pr [A \text{Funs}_{x \rightarrow x} \Rightarrow R] - \Pr [B \text{Funs}_{x \rightarrow x} \Rightarrow R] \right| \\
= \Pr [A \text{Funs}_{x \rightarrow x} \Rightarrow R \mid \text{coll}] \Pr [\text{coll}] + \Pr [A \text{Funs}_{x \rightarrow x} \Rightarrow R \mid \text{coll}] \Pr [\text{coll}] \\
- \Pr [B \text{Funs}_{x \rightarrow x} \Rightarrow R \mid \text{coll}] \Pr [\text{coll}] - \Pr [B \text{Funs}_{x \rightarrow x} \Rightarrow R \mid \text{coll}] \Pr [\text{coll}] \\
= \Pr [\text{coll}] \left( \Pr [A \text{Funs}_{x \rightarrow x} \Rightarrow R \mid \text{coll}] - \Pr [B \text{Funs}_{x \rightarrow x} \Rightarrow R \mid \text{coll}] \right) \\
\leq \Pr [\text{coll}].
\]

The last inequality is because the absolute difference between two probabilities is at most 1. Combining (10) and (11) yields

\[
\text{adv}_{PrF}^F (A) - \text{adv}_{PrF}^E (B) \leq \Pr [\text{coll}]. \tag{12}
\]

It remains to upper-bound $\Pr [\text{coll}]$. For this, we are interested in the probability, that any two distinct $A$ queries $\vec{Q}_i \neq \vec{Q}_j$ result in two same $B$ queries $X_i = X_j$. We have to consider two cases:

1. $(A_i, P_i) = (A_j, P_j)$ and $N_i \neq N_j$: The probability that any two distinct queries collide can be written as

\[
\Pr [\text{CBC-E}_{K_1} (e(A_i, P_i)) \oplus N_i = \text{CBC-E}_{K_1} (e(A_j, P_j)) \oplus N_j] \\
= \Pr [\text{CBC-E}_{K_1} (e(A_i, P_i)) \oplus \text{CBC-E}_{K_1} (e(A_j, P_j)) = N_i \oplus N_j] \\
= 0,
\]

because $N_i \oplus N_j \neq 0$ if $N_i \neq N_j$.

2. $(A_i, P_i) \neq (A_j, P_j)$ and $N_i = N_j$: Since CBC-E is an $\epsilon$-AXU family, the probability that any two distinct queries collide can be written as

\[
\Pr [\text{CBC-E}_{K_1} (e(A_i, P_i)) \oplus N_i = \text{CBC-E}_{K_1} (e(A_j, P_j)) \oplus N_j] \\
= \Pr [\text{CBC-E}_{K_1} (e(A_i, P_i)) \oplus \text{CBC-E}_{K_1} (e(A_j, P_j)) = N_i \oplus N_j] \\
= \Pr [\text{CBC-E}_{K_1} (e(A_i, P_i)) \oplus \text{CBC-E}_{K_1} (e(A_j, P_j)) = 0] \\
\leq \epsilon,
\]

because the probability that the XOR-difference between two outputs of an $\epsilon$-AXU family takes a specific value is at most $\epsilon$.

Note, that there are less than $q^2/2$ possible pairs $\vec{Q}_i, \vec{Q}_j$. By the union bound it follows that

\[
\Pr [\text{coll}] \leq \epsilon \cdot \frac{q^2}{2}.
\]
Applying this to (12) proves the lemma:
\[
\text{adv}^{\text{PRF}^*}_{E^*}(A) \leq \text{adv}^{\text{PRF}}_E(B) + \epsilon \cdot \frac{q^2}{2}.
\]

\[\square\]

### 4.4 Step 4: Combining the PRF with an ivE according to SIV

The previous step established a secure vector-valued PRF $F^*$ over the additional data, the plaintext and the nonce. This PRF is used to provide authenticity and integrity in CCM-SIV. In this step, we prove that combining this PRF with a secure IV-based symmetric encryption scheme for confidentiality yields a secure nonce-misuse-resistant authenticated encryption scheme.

**Algorithm 6** CTR-$E$ Encryption.

1: function $\text{CTR-}E^I_{K_3}(P)$
2: \hspace{1em} $I_1 \leftarrow I$
3: \hspace{1em} for $i \leftarrow 1$ to $p$ do
4: \hspace{2em} $C_i \leftarrow E^I_{K_3}(I_i \oplus P_i)$
5: \hspace{2em} $I_{i+1} \leftarrow \text{inc}_{32}(I_i)$ \hspace{0.5em} $\triangleright$ Increments least significant 32 bit of $I_i$ (modulo $2^{32}$).
6: \hspace{1em} return $C = (C_1, \ldots, C_p)$

CCM-SIV uses the CTR mode-of-operation of the block cipher $E_K$ (cf. Algorithm 6) as the IV-based symmetric encryption scheme $\Pi$. Formally, $\Pi$ is defined to be a set of three PPT algorithms (Gen, Enc, Dec) shown in Algorithm 7.

**Algorithm 7** ivE scheme $\Pi$

1: function $\text{Gen}( )$
2: \hspace{1em} return $K_3 \leftarrow K$
3: function $\text{Enc}_{K_3}(P)$
4: \hspace{1em} $I \leftarrow I$
5: \hspace{1em} $C \leftarrow \text{CTR-}E^I_{K_3}(P)$
6: \hspace{1em} return $(C, I)$
7: function $\text{Dec}_{K_3}(C, I)$
8: \hspace{1em} $P \leftarrow \text{CTR-}E^I_{K_3}(C)$
9: \hspace{1em} return $P$

The security game ivE for $\Pi$ works as follows: The adversary $A$ queries the oracle with at most $q$ plaintext queries $P_1, \ldots, P_q \in X \subseteq p_{\text{max}}$, where $p_{\text{max}}$ is the maximum number of message blocks for a single message. For CCM-SIV $p_{\text{max}} = 2^{32}$. The oracle is either the encryption function $\text{Enc}_{K_3}$ with $K_3 \leftarrow K$ or a truly random function with the same output size. $A$ wins the game, if she distinguishes both oracle versions with non-negligible probability. The security of $\Pi$ can be reduced to the security of the underlying PRF $E$ [7].
Definition 6. [Π’s security in the ivE game.] Let Π be the ivE scheme defined by Algorithm 7 and A be an adversary, who wins the ivE game with non-negligible advantage. Then there exists a wrapping PPT adversary B, who distinguishes E from a random function in the PRF game with non-negligible advantage. Particularly, it holds that

\[ \text{adv}^{\text{ivE}}_{\Pi}(A) \leq \text{adv}^{\text{PRF}}_{E}(B) + \frac{q^2 \cdot \text{max}}{2 \cdot 2^x}. \]

Based on Definition 6 and Lemma 2, we construct a secure nonce-based authenticated encryption (nAE) scheme \( \tilde{\Pi} \) through a composition of Π resp. CTR-E and \( F^* \) by using the 3-key instantiation of the SIV paradigm [18], also called construction A4 by Namprempre et al. [16]. The nAE scheme \( \tilde{\Pi} \) is a triple (Gen, Enc, Dec) of three PPT algorithms. The key generation algorithm Gen generates a triple key \( K = (K_1, K_2, K_3) \), where each element \( K_1, K_2, K_3 \) is chosen uniformly from \( K \). The authenticated encryption algorithm Enc takes as input the key \( K \), the additional data \( A \in \{0,1\}^* \), the plaintext \( P \in \{0,1\}^* \) and a nonce \( N \in X \) and outputs the corresponding ciphertext \( C \in \{0,1\}^* \) of the same length as \( P \) and a message tag \( T \in X \). The authenticated decryption algorithm Dec takes as inputs the key \( K \), the additional data \( A \in \{0,1\}^* \), the ciphertext \( C \in \{0,1\}^* \), a nonce \( N \in X \) and a tag \( T \in X \). Without loss of generality, we assume that \( P \)'s and \( C \)'s sizes are a multiple of the block size \( x \).

Algorithm 8 gives the pseudo codes for \( \tilde{\Pi} \).

**Algorithm 8** CCM-SIV scheme \( \tilde{\Pi} \) (formal description)

1: function Gen()
2: \( \text{return} \ K \leftarrow K^3 \)
3: function Enc\(_K\)(A, P, N)
4: \( T \leftarrow F^*_{K_1,K_2}(A, P, N) \)
5: \( C \leftarrow \text{CTR-E}_{K_3}(P) \)
6: \( \text{return} \ (C, T) \)
7: function Dec\(_K\)(A, C, N, T)
8: \( P \leftarrow \text{CTR-E}_{K_3}^T(C) \)
9: \( T' \leftarrow F^*_{K_1,K_2}(A, P, N) \)
10: if \( T = T' \) then
11: \( \text{return} \ P \)
12: else
13: \( \text{return} \bot \)

Figure 10(a) and 10(b) provide a graphical representation of Enc and Dec.

To analyze the security of \( \tilde{\Pi} \), we let A play the nAE security game defined in Figure 11. In the beginning, the challenger again chooses a flag \( b \) uniformly from the set \( \{R, PR\} \). But this time, the challenger establishes two types of oracles, the authenticated encryption oracle \( O_e \) and the authenticated decryption oracle \( O_d \). In the pseudo-random case, the challenger runs the key generation Gen to
generate the required keys $(K_1, K_2, K_3) = K$. After that, he sets the encryption oracle to $\tilde{\text{Enc}}$ and the decryption oracle to $\tilde{\text{Dec}}$. In the random version of the game, the encryption oracle $O_e$ and the decryption oracle $O_d$ should represent the idealized versions of $\tilde{\text{Enc}}$ resp. $\tilde{\text{Dec}}$. For $\tilde{\text{Enc}}$, the idealized version is a function, that takes as input the triple $(A, P, N)$ and outputs a truly random ciphertext-tag pair $(C, T)$, where $C$ is of the same length as $P$. In other words, the encryption oracle $O_e$ should be a random choice out of $\text{Funs}_{\ast, \ast \rightarrow \ast, \ast}$. To avoid ambiguities with the idealized version of the ivE scheme $\Pi$, the idealized version of $\tilde{\text{Enc}}$ is denoted as the function $\$(\cdot)$, such that $\$(\cdot) \leftarrow \text{Funs}_{\ast, \ast \rightarrow \ast, \ast}$.

The idealized version of $\tilde{\text{Dec}}$ is the function $\perp(\cdot)$, which rejects every decryption query by outputting the invalidity symbol $\perp$. Throughout the game, the adversary sends two types of queries. She is allowed to send up to $q_e$ encryption queries $\overrightarrow{Q}_1^i, \ldots, \overrightarrow{Q}_{q_e}^i$ to the encryption oracle $O_e$, where each encryption query $\overrightarrow{Q}_i^i$ with $1 \leq i \leq q_e$ is of the form $\overrightarrow{Q}_i^i = (A_i, P_i, N_i)$ and she is allowed to send up to $q_d$ decryption queries $\overleftarrow{Q}_1^j, \ldots, \overleftarrow{Q}_{q_d}^j$ to the decryption oracle $O_d$, where each decryption query $\overleftarrow{Q}_j^j$ with $1 \leq j \leq q_d$ is of the form $\overleftarrow{Q}_j^j = (A_j, C_j, N_j, T_j)$. The encryption and decryption queries may be sent in any order by the adversary. At the end, $A$ outputs a flag guess $b'$. It is assumed, that $A$’s queries are all distinct. This is sound, because $A$ learns nothing by querying the same message twice, because she receives the same output twice. Note, that repeating nonces are explicitly allowed in the encryption queries, as long as the triples $(A_i, P_i, N_i)$ are all distinct. This is, because we want to construct a nonce-misuse-resistant
Fig. 11: In the pseudo-random case, the encryption oracle $O_e$ is the authenticated encryption scheme $\tilde{\text{Enc}}$ and the decryption oracle $O_d$ is the authenticated decryption scheme $\tilde{\text{Dec}}$. In the truly random case, the encryption oracle always outputs a random ciphertext-tag pair and the decryption oracle always outputs $\bot$. The adversary $A$ sends at most $q_e$ encryption queries and at most $q_d$ decryption queries to the corresponding oracles. At the end, the adversary outputs a flag guess $b'$.

authenticated encryption scheme. Another restriction is, that $A$ is not allowed to send a decryption query $(A_j, C_j, N_j, T_j)$, where $C_j, T_j$ were obtained by a prior encryption query $(A_i, P_i, N_i)$ with $A_i = A_j$ and $N_i = N_j$. This would lead to a trivial win, because $A$ is then able to distinguish $\bot$ from $\tilde{\text{Dec}}$. This is also sound, because it does not make sense for an attacker in a real world scenario to decrypt a message, when he already knows the result. The advantage of $A$ breaking $\tilde{\Pi}$ in the nAE game is then defines as

$$\text{adv}^{\text{nAE}}_{\tilde{\Pi}}(A) = \left| \Pr[A, \bot \Rightarrow \text{R}] - \Pr[A, \tilde{\text{Enc}}, \tilde{\text{Dec}} \Rightarrow \text{R}] \right|.$$ 

This means, if $A$ is not able to distinguish $\tilde{\text{Enc}}$ from $\$ or $\tilde{\text{Dec}}$ from $\bot$ (with non-negligible probability), then the scheme is considered to be secure. This notion properly covers the security requirements for an authenticated encryption scheme. The indistinguishably of a ciphertext-tag pair from random values guarantees confidentiality. The inability of $A$ to come up with a decryption query, that decrypts to a valid $P$ instead of $\bot$ guarantees unforgeability and hence authenticity and integrity. Furthermore, the nonce-misuse-resistance property is taken into account by the game, because the adversary is allowed to repeat nonces (i.e. nonce-disrespecting adversaries), as long as the triples $(A_i, P_i, N_i)$ are distinct. The following main lemma relates the insecurity of the CCM-SIV scheme $\tilde{\Pi}$ to the insecurities of the underlying building blocks $\tilde{\Pi}$ and $F^*$. Based on the proof of this lemma, the proof of the main theorem (Theorem 1) will follow immediately.
Lemma 3. [˜Π’s security in the nAE game.] Let ˜Π be the nAE scheme defined by Algorithm 8 and A be an adversary, who wins the nAE game defined in Figure 11 with non-negligible advantage. Then there exists a wrapping PPT adversary B who distinguishes F* from a random function in the PRF* game defined in Figure 8 with non-negligible advantage and a wrapping PPT adversary D who distinguishes Π from its idealized version in the ivE game with non-negligible advantage. Particular, it holds that
\[
\text{adv}_{\text{nAE}}(\tilde{\Pi}) \leq \text{adv}_{\text{PRF}^*}(B) + \text{adv}_{\text{ivE}}(D) + \frac{q_d}{2^r}.
\] (13)

Lemma 3 states, that ˜Π must be a secure nAE scheme, because otherwise we would be able to break F* and Π. This would be a contradiction to Lemma 2 and Definition 6, because if \text{adv}_{\text{nAE}}(\tilde{\Pi}) is non-negligible, either \text{adv}_{\text{PRF}^*}(B) or \text{adv}_{\text{ivE}}(D) or both must be also non-negligible to fulfill (13).

Proof (Proof of Lemma 3.). The proof contains two reductions. The first reduction B is given in Algorithm 9 and the related framework is illustrated in Figure 12.

**Algorithm 9** Pseudo Code for B.

1: \(K_3 \leftarrow \mathcal{K}\)

2: **for all** queries \(\vec{Q}^e_i = (A, P, N)\) from A with \(1 \leq i \leq q_e\) **do**
3: \(\text{send } \vec{Q} = \vec{Q}^e_i\) to challenger
4: \(T \leftarrow \mathcal{O}(\vec{Q})\)
5: \(C \leftarrow \text{CTR-E}_{K_3}(P)\)
6: \(\text{send } (C, T)\) to A

7: **for all** queries \(\vec{Q}^d_j = (A, C, N, T)\) from A with \(1 \leq j \leq q_d\) **do**
8: \(P \leftarrow \text{CTR-E}_{K_3}(C)\)
9: \(\text{send } \vec{Q} = (A, P, N)\) to challenger
10: \(T' \leftarrow \mathcal{O}(\vec{Q})\)
11: **if** \(T = T'\) **then**
12: \(\text{send } P\) to A
13: **else**
14: \(\text{send } \perp\) to A
15: **return** same \(b'\) as A

The adversary A is playing the nAE game. She may output up to \(q_e\) encryption queries and up to \(q_d\) decryption queries in any order. After an encryption query, she expects the corresponding ciphertext-tag pair, and after a decryption query, she expects either the corresponding plaintext or the symbol \(\perp\). The wrapping algorithm B has two handlers for these two types of A-queries to simulate the authenticated encryption Enc and the authenticated decryption Dec. Since B is playing the PRF* game, his goal is to distinguish between \(F^*\) and a random choice out of \(\text{Funs}_{\ast, \ast, x \rightarrow z}\). For this, B outputs the same flag guess \(b'\) as A at the
Fig. 12: The wrapping algorithm $B$ tries to win the $\text{PRF}^*$ game on the outside by using the $\text{nAE}$-adversary $A$ as a subroutine. $B$ outputs the same flag guess $b'$ as $A$.

end. In the beginning, $B$ chooses the encryption key $K_3$ uniformly from the set $\mathcal{K}$. Upon receiving an encryption query $Q^e_i$ from $A$, he does the following: First, he passes the query $Q^e_i$ directly to his challenger and uses the output $O (Q^e_i)$ as the tag $T$ (as a replacement for the $F^*$ evaluation). Then he runs the CTR-$E$ encryption using the key $K_3$ and the tag $T$ as IV. The resulting ciphertext $C$ together with $T$ is sent back to $A$. Upon receiving a decryption query $Q^d_j$ from $A$, $B$ does the following: First, he runs the CTR-$E$ decryption using the key $K_3$ and the tag $T$ as the IV, which is part of the decryption query, to obtain the unverified plaintext $P$. Then he sends the triple $(A, P, N)$ to his challenger and uses the output as $T'$ (as a replacement for the $F^*$ evaluation). If $T = T'$, the unverified plaintext $P$ is considered to be valid and sent back to $A$. If $T \neq T'$, he sends back the symbol $\perp$. By definition, the advantages of $A$ and $B$ are

$$\text{adv}^{\text{nAE}}(A) = \left| \Pr \left[ A^{\text{Enc}, \perp} \Rightarrow R \right] - \Pr \left[ A^{\text{Enc}, \text{Dec}} \Rightarrow R \right] \right|.$$  \hspace{1cm} (14)

$$\text{adv}^{\text{PRF}^*}(B) = \left| \Pr \left[ B^{\text{Funs}^*, \ldots, x \rightarrow x} \Rightarrow R \right] - \Pr \left[ B^{F^*} \Rightarrow R \right] \right|.$$  \hspace{1cm} (15)

If $B$’s challenger chooses $b = \text{PR}$, then $B$ perfectly simulates $\widehat{\text{Enc}}$ and $\widehat{\text{Dec}}$, because $O = F^*$ in Algorithm 9. Since $B$ outputs the same $b'$ as $A$, it holds that

$$\Pr \left[ A^{\widehat{\text{Enc}}, \widehat{\text{Dec}}} \Rightarrow R \right] = \Pr \left[ B^{F^*} \Rightarrow R \right].$$  \hspace{1cm} (16)

Subtracting (15) from (14), applying the triangle inequality and making use of (16) yields

$$\text{adv}^{\text{nAE}}(A) - \text{adv}^{\text{PRF}^*}(B) \leq \left| \Pr \left[ A^{\text{Enc}, \perp} \Rightarrow R \right] - \Pr \left[ B^{\text{Funs}^*, \ldots, x \rightarrow x} \Rightarrow R \right] \right|.$$  \hspace{1cm} (17)
The second reduction $D$ is trying to break the ivE-security of $\Pi$ by running $A$ as a subroutine.

**Algorithm 10** Pseudo Code for $D$.

1: for all queries $\vec{Q}^e_i = (A, P, N)$ from $A$ with $1 \leq i \leq q_e$ do
2: send $P_i = P$ to challenger
3: $(C, T) \leftarrow \mathcal{O}(P_i)$
4: send $(C, T)$ to $A$

5: for all queries $\vec{Q}^d_j = (A, C, N, T)$ from $A$ with $1 \leq j \leq q_d$ do
6: send $\bot$ to $A$
7: return same $b'$ as $A$

The pseudo code for $D$ is given in Algorithm 10 and the corresponding framework is illustrated in Figure 13. The adversary $D$ plays the ivE game with his challenger and again consists of two handlers for the two different types of $A$-queries. Upon receiving an encryption query $\vec{Q}^e_i$ from $A$, $D$ sends the plaintext $P_i$ to his challenger as a replacement for $\tilde{\text{Enc}}$ and gets back the corresponding ciphertext-IV pair $(C_i, I_i)$. $D$ simply passes this pair to $A$ and thereby misuses the IV $I_i$ as the tag $T$. Upon receiving a decryption query $\vec{Q}^d_j$ from $A$, $D$ always answers with $\bot$. At the end, $D$ outputs the same $b'$ as $A$. The advantage of $D$ in the ivE game is defined as

$$\text{adv}_{\text{ivE}}^{\Pi}(D) = \Pr[D^{\text{Funs}_{px \rightarrow px, x}} \Rightarrow R] - \Pr[D^{\text{Enc}} \Rightarrow R].$$

(18)

Now consider the conditional probability space $b = R$. By looking at Algorithm 10 and Figure 13, one can see that $A$’s encryption queries are answered by a truly...
random function $O \leftarrow \text{Funs}_{px \rightarrow px,x}$ and $A$'s decryption queries are always answered with $\perp$. Hence, $D$ perfectly simulates $\$ , $\perp$. Therefore, it holds that
\[
\Pr[A^\$ , \perp \Rightarrow R] = \Pr[D^{\text{Funs}_{px \rightarrow px,x}} \Rightarrow R]. \tag{19}
\]
Subtracting (18) from (17), applying the triangle equality and making use of (19) yields
\[
\text{adv}_n^{\text{AE}}(A) - \text{adv}_n^{\text{PRF}}(B) - \text{adv}_n^{\text{ivE}}(D) 
\leq \left| \Pr[\text{D}^{\text{Enc}} \Rightarrow R] - \Pr[\text{B}^{\text{Funs}_{*,*,x \rightarrow x}} \Rightarrow R] \right|. \tag{20}
\]
It remains to bound the quantity on the right side of (20). For this, we have to analyze, under which circumstances $D^{\text{Enc}}$ behaves different from $B^{\text{Funs}_{*,*,x \rightarrow x}}$. Let us first compare the encryption query handlers in Algorithm 10 and 9. Since $D$ is interacting with $\text{Enc}$, an encryption query $\vec{Q}^e_i = (A, P, N)$ from $A$ is answered with $(C, T) \leftarrow \text{Enc}_{K_3}(P)$. Since $B$ is interacting with $\text{Funs}_{*,*,x \rightarrow x}$, an encryption query $\vec{Q}^e_i = (A, P, N)$ from $A$ is answered with $(C, T)$, where $T$ is chosen uniformly by $O \leftarrow \text{Funs}_{*,*,x \rightarrow x}$ and $C \leftarrow \text{CTR-E}_{T}^{K_3}(P)$. But this is exactly $\text{Enc}_{K_3}(P)$. Therefore, both $D$ and $B$ answer encryption queries from $A$ with the same probability distribution. Now we have to compare the decryption handlers in Algorithm 10 and 9. $D$ answers all decryption queries $\vec{Q}^d_j$ from $A$ with $\perp$, whereas $B$ answers with the plaintext $P$ instead of $\perp$, if the two tags $T$ and $T'$ accidentally match up. $T$ is chosen by the adversary $A$, whereas $T'$ is calculated by $O \leftarrow \text{Funs}_{*,*,x \rightarrow x}$. Let match denote the event, that at least one such a match occurs during the decryption queries. Then conditioned to the case $\text{match}$, $D$ always outputs $\perp$ and hence $D^{\text{Enc}}$ behaves the same as $B^{\text{Funs}_{*,*,x \rightarrow x}}$. With this, we can rewrite the right side of (20):
\[
\left| \Pr[\text{D}^{\text{Enc}} \Rightarrow R] - \Pr[\text{B}^{\text{Funs}_{*,*,x \rightarrow x}} \Rightarrow R] \right|
= \left| \Pr[\text{D}^{\text{Enc}} \Rightarrow R \mid \text{match}] \Pr[\text{match}] + \Pr[\text{D}^{\text{Enc}} \Rightarrow R \mid \neg \text{match}] \Pr[\neg \text{match}] - \Pr[\text{B}^{\text{Funs}_{*,*,x \rightarrow x}} \Rightarrow R \mid \text{match}] \Pr[\text{match}] - \Pr[\text{B}^{\text{Funs}_{*,*,x \rightarrow x}} \Rightarrow R \mid \neg \text{match}] \Pr[\neg \text{match}] \right|
= \Pr[\text{match}] \left| \Pr[\text{D}^{\text{Enc}} \Rightarrow R \mid \text{match}] - \Pr[\text{B}^{\text{Funs}_{*,*,x \rightarrow x}} \Rightarrow R \mid \text{match}] \right|
\leq \Pr[\text{match}] . \tag{21}
\]
The tag $T'$ is uniformly distributed and hence takes the value $T$ with probability $1/2^x$. For $q_d$ distinct decryption queries the union bound gives
\[
\Pr[\text{match}] \leq \frac{q_d}{2^x} . \tag{22}
\]
Combining (20), (21) and (22) proves the lemma:

\[ \text{adv}^{n\text{AE}}_{\tilde{H}}(A) - \text{adv}^{\text{PRF}}_{F^*}(B) - \text{adv}^{\text{ivE}}_{\tilde{H}}(D) \leq \frac{qd}{2^x}. \]

Lemma 3 gives an upper bound for the nAE-security of \( \tilde{H} \) by relating it with the PRF*- and ivE-security of the underlying building blocks \( F^* \) and \( \Pi \). The main theorem of the paper (Theorem 1) gives an upper bound for the nAE-security of \( \tilde{H} \) by relating it with the PRF-security of the underlying PRF \( E \). We now have everything together to do the proof for Theorem 1 in a straightforward way:

**Proof (Proof of Theorem 1).** Let \( B \) be an adversary incorporating the PRF-adversary from Lemma 2 and the PRF-adversary from Definition 6. Then substituting the advantages in Lemma 3 with the bounds in Lemma 2 and Definition 6 yields

\[ \text{adv}^{n\text{AE}}_{\tilde{H}}(A) \leq 2 \cdot \text{adv}^{\text{PRF}}_{E}(B) + \epsilon \cdot \frac{(q_e + q_d)^2}{2} + \frac{q_e^2 \cdot p_{\text{max}}}{2 \cdot 2^x} + \frac{qd}{2^x}. \] (23)

The maximum number of allowed plaintext blocks \( p_{\text{max}} \) in Theorem 1 is dependent on the bit length of the counter field in the CTR mode encryption. For a \( c \) bit counter (in case of CCM-SIV \( c = 32 \) bit), \( p_{\text{max}} = 2^c \). So the dominating term in (23) is

\[ \frac{q_e^2 \cdot p_{\text{max}}}{2 \cdot 2^x} = \frac{q_e^2}{2 \cdot 2^{x-c}}. \]

5  **Differences to GCM-SIV and Nonce-based Key Derivation**

Our main theorem (Theorem 1) differs from [7, Theorem 4.2] of the 3-key instantiation of the GCM-SIV scheme, where \( p_{\text{max}} = 2^k - 1 \), the bit length of the counter is represented as \( k \) (instead of \( c \)) and the block size is represented as \( n \) (instead of \( x \)), by a factor \( 1/2 \) in two terms. This is due to the fact that we upper bound \( \sum_{i=0}^{2^k-1} i \) by \( q^2/2 \) instead of \( q^2 \).

A modification to a 2-key variant of CCM-SIV is straightforward. Furthermore, an extension to a nonce-based key derivation as proposed in [6, Fig. 2] and [11, Section 5] to reduce the dominating term \( q_e^2 \cdot p_{\text{max}} / 2 \cdot 2^x \) in Theorem 1 to \( QR^2 \cdot p_{\text{max}} / 2 \cdot 2^x \) (using the PRF-advantage), where \( Q \) is the number of distinct nonces used throughout all encryption queries and \( R \) is the maximal number of repetitions of any nonce in all encryption queries, is realizable.
6 Conclusion

We introduced a new nonce-misuse-resistant authenticated encryption scheme called CCM-SIV, which only uses a single type of PRF as the underlying cryptographic primitive (AES). This makes the scheme compatible with most existing AES software libraries and hardware accelerators. Since we use three independent subkeys, which can be derived from a single master key by a deterministic pseudo random number generator, no strange internal collisions on the inputs of the AES block cipher can occur between the MAC generation and the symmetric encryption parts. Therefore, no modifications have to be applied to circumvent possible internal collisions. This makes the scheme cryptographically strong and easy to implement in both hardware and software.

References