Better Bounds for Block Cipher Modes of Operation via Nonce-Based Key Derivation

Shay Gueron
University of Haifa and Amazon Web Services
shay@math.haifa.ac.il

Yehuda Lindell
Bar-Ilan University
lindell@biu.ac.il

ABSTRACT

Block cipher modes of operation provide a way to securely encrypt using a block cipher. The main factors in analyzing modes of operation are the *level of security* achieved (chosen-plaintext security, authenticated encryption, nonce-misuse resistance, and so on) and *performance*. When measuring the security level of a mode of operation, it does not suffice to consider asymptotics, and a concrete analysis is necessary. This is especially the case today, when encryption rates can be very high, and so birthday bounds may be approached or even reached.

In this paper, we show that key-derivation at every encryption significantly improves the security bounds in many cases. We present a new key-derivation method that utilizes a *truncated block cipher*, and show that this is far better than standard block-cipher based key derivation. We prove that by using our key derivation method, we obtain greatly improved bounds for many modes of operation, with a result that the lifetime of a key can be significantly extended. We demonstrate this for AES-CTR (CPA-security), AES-GCM (authenticated encryption) and AES-GCM-SIV (nonce-misuse resistance). Finally, we demonstrate that when using modern hardware with AES instructions (AES-NI), the performance penalty of deriving keys at each encryption is insignificant for most uses.

1 INTRODUCTION

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, the construction of nonce-misuse resistant modes of operation, that remain secure even if a nonce repeats, is one consideration in the recent CAESAR competition.

One issue that has recently become a concern is the block size of block ciphers and the ramification that this has on security. Specifically, when a block cipher with block size n is used to encrypt $2^{n/2}$ blocks, then birthday collisions occur with high probability, potentially resulting in a security breach. Although the threat due to such collisions is often thought to be theoretical in nature, it was recently shown that real attacks can be carried out when 3DES is used in TLS, because of the small block size [5]. Specifically, the block size of 3DES is 64 bits, and thus collisions occur at just 2^{32} blocks, or 32GB of data, which can be transferred in under an hour on a fast Internet connection, and in seconds within a data center.

At first sight, this problem of birthday collisions is a problem that is only of relevance for 3DES. Modern block ciphers, like AES, have a block size of 128, and the birthday bound is thus 2^{64}, which corresponds to a whopping 1 million Petabytes of data. However, in reality, birthday collisions *are* a concern, even for AES or other 128-bit block ciphers. This is because the standard NIST recommendation is to stop using a key when the probability of some leakage exceeds 2^{-32}. Thus, after encrypting 2^{48} blocks, keys must be changed. Furthermore, in many popular modes of operation, birthday bounds are actually reached far earlier. Two important examples are counter (CTR) mode and AES-GCM, when using random IVs. In both of these case, the standard implementation used a 96-bit IV, and thus collisions occur in the IV with probability 2^{-32} after encrypting only 2^{32} different messages (with fresh random IVs). Therefore, the number of messages that can be encrypted with a single key is actually quite low.

A real example – QUIC. QUIC [18] is a new transport protocol that is designed to improve the performance of connection-oriented web applications that are currently using TCP, while providing security protection that is comparable to that of TLS. QUIC encrypts "source-address tokens", with the property that a cluster of servers can recognize them in the future, but without clients being able to forge them. Simply adding a MAC would suffice, but for future-proofing they should also be confidential. All servers can share a fairly long-lived secret key, but the servers need to be able to create these tokens quickly, and independently. Since a central allocation system for nonces is not operationally viable, random selection of nonces is the only possibility. AES-GCM's limit of 2^{32} random nonces (per key) suggests that, even if the system rotated these secret keys *daily*, it could not issue more than about 50K tokens per second. However, in order to process DDoS attacks the system may need to be able to issue of hundreds of millions of tokens per second. A similar problem arises in TLS with session tickets [2]. Although the demands are significantly reduced in this context, a limit of 50K tickets per second is still insufficient for many sites, and thus plain AES-GCM is unsuitable for this as well.

1.1 Our Results

We introduce a generic technique for significantly extending the lifetime of a key. The idea is very simple: *first derive a per-message key by applying a key-derivation function with the master-key and nonce, and then use the per-message key to encrypt the message.* Intuitively, this ensures that no single key is used too much, and so many more blocks can be encrypted. Furthermore, it requires only minimal changes to existing schemes, which is important for deployment. However, implementing this idea has two major challenges:

(1) On the one hand, key derivation based on a hash function would yield good bounds but is very slow. On the other hand, standard key derivation based on AES in counter mode yields poor bounds. In particular, collisions would occur with probability 2^{-32} after 2^{48} derivations.
(2) Even if one were to overcome the key derivation bound using a block cipher, this would require running an AES key expansion

for every message. Standard methods of key expansion, e.g., using the AES-NI `aeskeygenassist` instruction, are very slow.

We address the above challenges and show how to use continual key derivation in an efficient way, and with very good bounds.

Continual key derivation. We indeed use AES-based key derivation. However, in contrast to the standard counter-mode based key derivation [4], we use a *truncated* block cipher. For example, in order to derive a 128-bit key, two AES encryptions are computed and the key is taken to be the concatenation of the first half of each output block; likewise, to derive a 256-bit key, four AES encryptions are used. The reason that this key derivation is preferable is due to the fact that the "best" key derivation utilizes a pseudorandom *function*, whereas block ciphers are pseudorandom *permutations*. Thus, as the number of derivations approaches the birthday bound, the keys derived can no longer be assumed to be random. However, a truncated block cipher is no longer a permutation, and the more it is truncated the closer it behaves to a pseudorandom function. Using this method and AES, we can derive Q keys (as is needed, for example, for AES256-GCM) that can be distinguished from random with advantage at most $O\left(\frac{Q}{2^{96}}\right)$. In contrast, the standard counter-based method can be distinguished from random with advantage $O\left(\frac{Q^2}{2^{128}}\right)$. Thus, in order to maintain a 2^{-32} upper bound on the advantage, we are able to derive up to 2^{64} keys, instead of up to 2^{48} keys using the standard counter-based method. (Note that if we were to truncate even more, the bounds would be even better.)

Having overcome the problem of the key derivation bound, we come to the second issue of performance. First, using truncated AES may seem to actually further harm performance since more AES encryptions are needed (double, to be exact). However, on modern processors with an AES-NI instruction set, AES encryptions are fully pipelined and so the difference in cost between 2 AES operations and 4 is negligible. Next, as we mentioned above, the key expansion operation is very expensive, even using AES-NI. We therefore use a method described in [14] for computing key expansion via the AES-NI *round function*, and pipelining this together with encryptions. As we show, the result has very little overhead (in percentages), with the exception of very small messages. In all cases, in objective terms, the overhead is relatively small. For example, key expansion of a 128-bit key using `aeskeygenassist` takes 111 cycles, our optimized key expansion takes 48 cycles, and our key expansion interleaved with 2 AES encryptions takes 58 cycles.

Finally, we provide a very detailed analysis of our method for general modes of operation, and apply it to AES-CTR, AES-GCM [15, 16] and AES-GCM-SIV [12, 13]. Since our analysis is general, it can be applied to other schemes as well, and we hope will therefore be useful beyond these specific examples. We compare the bounds that we achieve to the bounds of the basic modes without key generation, and show that the lifetime of keys can be greatly extended using our technique. To illustrate this, with a standard 96-bit nonce, AES-CTR and AES-GCM can be used to encrypt at most 2^{48} blocks (e.g., 2^{32} messages of length 2^{16} each), while keeping the adversarial advantage below 2^{-32}. In contrast, using our key derivation, the same modes can be used to encrypt 2^{64} messages of length 2^{16} each, with an adversarial advantage of at most 2^{-32}.

Encryption with a nonce vs a random IV. Throughout this paper, we refer to a nonce as a non-repeating value that is unique but not necessarily random. This is in contrast to an IV that is considered random. It is well known that for modes of operation for which nonce uniqueness suffices (like CTR), if nonce uniqueness can be guaranteed, then this is preferable to choosing a random IV. This is because random IVs collide at the birthday bound, and possibly earlier if the device encrypting has poor entropy. It is interesting to note that our key-derivation method does *not* increase the number of messages that one can encrypt, when a random IV is used. This is due to the fact that IV collisions still happen with the same probability, and when they collide the same key and nonce is used even with key derivation. Nevertheless, in these cases, our method does enable one to encrypt longer messages (and so overall many more blocks). However, in the unique nonce setting, where we assume that the parties can guarantee uniqueness (e.g., by keeping the current counter as state), our method does enable encrypting far more messages, as described above.

Nonce-misuse resistance. The most popular authenticated-encryption mode of operation today, AES-GCM, is seeing widespread use due to its attractive performance, which is enhanced by AES and polynomial multiplication instructions that are now part of many modern processor architectures. However, it suffers catastrophic failures of confidentiality and integrity if two distinct messages happen to be encrypted, under the same key, with the same nonce. While the requirements for authenticated encryption specify that the pair of (key, nonce) shall only ever be used once, and thus prohibit such failures, there are cases where, in practice, guaranteed uniqueness of nonces is a concern. This was shown recently for when AES-GCM is used with random nonces in TLS [7].

Nonce-misuse resistant authenticated encryption schemes [19] do not suffer from this problem. For this class of authenticated encryption, encrypting two messages with the same nonce only discloses whether the messages were equal or not. This is the minimum amount of information that a deterministic algorithm can leak in this situation. In [11], an authenticated-encryption mode called GCM-SIV was introduced; this scheme is based on the same paradigm as the SIV mode by [19], but has far enhanced performance due to the use of the same building blocks as AES-GCM. However, GCM-SIV's bounds are not optimal, and in particular, cannot be used to encrypt more than 2^{32} messages, as in AES-GCM. This makes GCM-SIV unsuitable for QUIC, with the requirements as above. Using our key derivation method, our analysis shows that it is possible to encrypt up to approximately 2^{64} messages of length 2^{16} (for one example of parameters). This makes it suitable for settings like QUIC and others. *We stress that when using a nonce-misuse resistance scheme, it is possible to encrypt essentially as many messages when using a random IV as when using nonces that are guaranteed to be unique, even when not using very good randomness. This is because IV repetitions do not harm security (beyond leaking that the same message was possibly encrypted).*

Consider now the use of GCM-SIV with our key derivation to generate tokens in QUIC. In this case, encryption must be carried out using a random IV (since different servers share the same key but not joint state), and must be able to generate millions of tokens

per second. At the rate of 1 million per second, the original GCM-SIV of [11] would require rotating keys every hour. In contrast, even at the rate of 1 billion per second, GCM-SIV with key derivation would reach the bounds after only 500 years of use.

Impact. AES-GCM-SIV has been proposed as a CFRG standard [13]. There are a number of minor differences between the original GCM-SIV in [11] and the proposed standard, but the main difference is the use of our proposed key derivation technique. An original analysis of this technique appeared in [12], specifically for GCM-SIV. However, the analysis here provides far better bounds, and is general and thus can be applied to other modes like CTR and AES-GCM. AES-GCM-SIV, utilizing our key-derivation technique, has already been integrated into BoringSSL [1] (Google's fork of OpenSSL), and Google is also already using it in QUIC [18].

1.2 Related Work

Bellare and Abdalla [3] suggested a re-keying mechanism to increase the lifetime of a key. They provide security analyses for different re-keying mechanisms, and show that re-keying indeed improves the security margins and therefore extends the lifetime of the master key. Our method is different in the mechanism, and also in the key derivation itself. Specifically, [3] consider a scenario where keys are changed periodically using an external counter. This requires storing state, and coordination between different machines using the same key. In contrast, we use the nonce to derive a key, develop a general result on the security benefits of this, and apply it to a number of different schemes. Our results are very different. For one, we obtain that when using schemes that must be nonce respecting, our method enables encrypting longer messages but not more messages (overall more blocks). In contrast, when using nonce misuse-respecting schemes, our method enables encrypting many more messages with a random IV. Thus, our method enables parties to encrypt using a random IV rather than with a unique nonce (which is preferable since state is not needed), and obtain excellent bounds even when the source of entropy is not perfect.

2 DEFINITIONS OF SECURITY

2.1 Preliminaries

Within our proofs, we will use the following theorem from [21] that bounds the multi-collision probability of randomly chosen items.

Theorem 2.1 (Theorem 2 of [21].). *Let $2 \le r \le q \le A$. Let q balls be thrown, one by one (independently) at random, into A bins. Let $MultiColl(A, q, r)$ denote the event (called an r multi-collision) that there exists at least one bin that contains at least r balls. Then,*

$$\Pr[MultiColl(A, q, r)] \le \frac{q^r}{r! \cdot A^{r-1}}. \quad (1)$$

2.2 Key-Derivation Functions (KDF)

We define the notion of KDF security, which is actually just the definition of a pseudorandom function; we use the term KDF since this is its use in our scheme (we will separately define what we need with respect to block cipher security). Let $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^m$ be a keyed function, with keys of length κ, input of length n and output of length m (note that in some cases it will hold that $m = \kappa$, but not always; this will depend on the concrete encryption scheme being considered). Then:

Experiment ExptKDF$_{\mathcal{A},F}$:

(1) Choose a random $b \leftarrow \{0,1\}$ and key $k \leftarrow \{0,1\}^\kappa$, and function f chosen randomly from the set of all functions from $\{0,1\}^n$ to $\{0,1\}^m$.
(2) If $b = 0$ then set $O(x) = f(x)$; if $b = 1$ then set $O(x) = F_k(x)$.
(3) Obtain $b' \leftarrow \mathcal{A}^{O(\cdot)}(\lambda)$, where λ denotes the empty input.
(4) Output 1 if and only if $b' = b$.

We say that $\mathcal{A}$ is a (t, Q)-adversary if it runs in at most t steps and makes at most Q queries to O. We define the advantage of $\mathcal{A}$ by

$$\mathsf{AdvKDF}_{\mathcal{A},F} = 2 \cdot \mathrm{Prob}\left[\mathsf{ExptKDF}_{\mathcal{A},F} = 1\right] - 1.$$

2.3 Multiple-Instance Block Cipher Security

We now define an experiment for the purpose of formalizing the assumption on the underlying block cipher. Clearly, the most basic assumption on a block cipher is simply that it is a pseudorandom permutation (or function). However, we wish to consider the case that an adversary interacts with Q *different instances* of the block cipher with different keys. It is well known that there is a general reduction to the single key case. However, this reduction results in a degradation of Q in the distinguishing success, as well as a blow-up of Q in the running time of the adversary. Specifically, via a standard hybrid argument, one can show that if there exists an adversary running in time t that distinguishes a series of Q block cipher instances with independent keys from Q random permutations (resp., functions) with probability ϵ, then there exists an adversary running in time $t \cdot Q$ that distinguishes a single block cipher instance from a single random permutation (resp., function) with probability ϵ/Q. This reduction is very wasteful since it both increases the running time and reduces the distinguishing capability by Q. To see why this is problematic, for AES-128, assume that we wish to claim that no adversary running in time 2^{48} and querying $Q = 2^{48}$ different instances can succeed with probability greater than 2^{-32} (which is the standard security margin used in practice). That is, we start with an adversary with $t = Q = 2^{48}$ and $\epsilon = 2^{-32}$. Using the generic reduction in an attempt to prove *by contradiction*, we would conclude that if such an adversary could be constructed, then one could construct an adversary attacking a single instance of AES-128 that runs in time $t \cdot Q = 2^{96}$ and succeeds with probability greater than $\epsilon/Q = 2^{-80}$. Such an adversary *clearly exists*, and thus there is no contradiction. Therefore, we cannot conclude that our (reasonable) assumption holds. We stress that this is a logical failure and not a security failure. Needless to say, an adversary running in time 2^{96} is not reasonable. However, we are trying to verify a multi-key claim on AES via reduction to the single key case, and it is this *reduction* that is meaningless. (The reduction states that X implies Y, and is supposed to prove that X is false since Y is false. However, the resulting Y is clearly true, and thus this implies nothing about X.)

As such, in this section, we will provide a *direct formalization* of multi-instance security and state the accepted assumption regarding the security of AES.

Block cipher – single instance. A block cipher $E = E_k$ with block size n and key size κ is a family of permutations over $\{0,1\}^n$, indexed by the key $k \in \{0,1\}^\kappa$. Denote by $\mathsf{AdvPRP}_E(t,B)$ the advantage of distinguishing E (with a key chosen uniformly at random) from a random permutation over $\{0,1\}^n$, for all adversaries running in at most t steps and making at most B queries to the oracle $O(\cdot) = E_k(\cdot)$. Note that each query is a single block, and thus B denotes the number of blocks queried to the oracle.

Specifying the multi-instance adversary. Informally, we wish to consider adversaries that interact with Q different instances of the block cipher (each with an independent key). In order to analyze security, we need to consider two parameters: how many times each instance was queried, and the maximum number of different instances that the same block was queried to. We now formalize this.

We consider an adversary $\mathcal{A}$ given access to a series of block cipher instances. Formally, the adversary is given a single oracle, and defining that the adversary's oracle query includes an *index* saying which of the instances in the series it is querying, together with the actual input to that instance. We assume that the adversary makes distinct queries to every instance (i.e., $\mathcal{A}$ does not make superfluous queries), and we denote by B_i the number of queries made by $\mathcal{A}$ to the ith instance. Let Q be the number of different instances and let $\vec{B} = (B_1, \ldots, B_Q)$. Let $\boldsymbol{Q}_i$ be the set of queries made to the ith instance (and thus $|\boldsymbol{Q}_i| = B_i$). Then, the frequency of an input x is the number of sets $\boldsymbol{Q}_i$ such that $x \in \boldsymbol{Q}_i$, and the maximum frequency is the maximum over the frequencies of all inputs. We say that $\mathcal{A}$ is a $(t, Q, \vec{B}, \mu)$-adversary if:

- $\mathcal{A}$ runs in at most t steps,
- $\mathcal{A}$ interacts with at most Q instances,
- $\mathcal{A}$ queries the ith instance at most B_i times, and
- The maximum query frequency is μ.

Defining multi-instance security. The most straightforward way of defining the security of a block cipher in the multi-instance setting is simply as a standard pseudorandom function experiment, but with a series of oracles, as above. (Note that we are interested in the distance of the block cipher from a pseudorandom *function* and not from a permutation.) A first attempt at the definition is as follows:

Experiment $\mathsf{ExptPRF}^Q_{\mathcal{A},E}$ – first attempt:

(1) Choose a random $b \leftarrow \{0,1\}$, random keys $k_1, \ldots, k_Q \leftarrow \{0,1\}^\kappa$ and random functions $f_1, \ldots, f_Q$ from $\{0,1\}^n$ to $\{0,1\}^n$.
(2) If $b = 0$ then set $O(i,x) = f_i(x)$; if $b = 1$ then set $O(i,x) = E_{k_i}(x)$.
(3) Obtain $b' \leftarrow \mathcal{A}^{O(\cdot,\cdot)}(\lambda)$.
(4) Output 1 if and only if $b' = b$.

Although natural as a definition, this actually is not sufficient in the case that Q is very large. This is because in such a case, an adversary can carry out the following attack *which has nothing to do with the security of the block cipher*: for every $i = 1, \ldots, Q$, query $O(i, 0^n)$ and $O(i, 1^n)$. If there exist distinct i, j such that $O(i,0^n) = O(j,0^n)$ and $O(i,1^n) = O(j,1^n)$, then output 1; else, output 0. Now, if $b = 1$, then the probability that $k_i = k_j$ for some i, j is $\frac{Q^2}{2^{n+1}}$. In contrast, if $b = 0$, then the probability that $f_i(0^n) = f_j(0^n)$ *and* $f_i(1^n) = f_j(1^n)$ for some i, j is $\frac{Q^2}{2^{2n}}$. Thus, if $Q = 2^{n/2}$, then it follows that an adversary can distinguish with very high probability. Although this attack works, it actually does not represent any real attack on the block cipher. Rather, if $k_i = k_j$ then this just means that effectively the ith instance is queried $B_i + B_j$ times and it makes no difference (there may be settings where it does makes a difference, but this is not the case here). We therefore modify the experiment so that if $k_i = k_j$ then $f_i = f_j$ as well. We have:

Experiment $\mathsf{ExptPRF}^Q_{\mathcal{A},E}$:

(1) Choose a random $b \leftarrow \{0,1\}$, random keys $k_1, \ldots, k_Q \leftarrow \{0,1\}^\kappa$, and random functions $f_1, \ldots, f_Q$ from $\{0,1\}^n$ to $\{0,1\}^n$, under the constraint that if $k_i = k_j$ then $f_i = f_j$.
(2) If $b = 0$ then set $O(i,x) = f_i(x)$; if $b = 1$ then set $O(i,x) = E_{k_i}(x)$.
(3) Obtain $b' \leftarrow \mathcal{A}^{O(\cdot,\cdot)}(1^n)$
(4) Output 1 if and only if $b' = b$.

As above, we define the advantage of the adversary to be

$$\mathsf{AdvPRF}^Q_{\mathcal{A},E} = 2 \cdot \mathrm{Prob}\left[\mathsf{ExptPRF}^Q_{\mathcal{A},E} = 1\right] - 1.$$

An analogous definition regarding distinguishing E from a random *permutation* in a multikey setting can be formalized. We denote the experiment and the associated advantage by $\mathsf{ExptPRP}^Q_{\mathcal{A},E}$ and $\mathsf{AdvPRP}^Q_{\mathcal{A},E}$, respectively.

Multi-instance security of ideal ciphers. We follow [17, Theorem 2] that discusses the ideal block cipher multi-key security. They show that if E is an ideal cipher, then for every $(t, Q, \vec{B}, \mu)$-adversary trying to distinguish E from a random *permutation*, it holds that $\mathsf{AdvPRP}^Q_{\mathcal{A},E} \leq \frac{Q^2 + 2Q \cdot T_E}{2^{\kappa+1}}$, where T_E is the number of queries that the adversary makes to the cipher with adversarially chosen keys.[1]

We extend their result to consider pseudorandom functions. In addition, we provide a stronger result with better bounds, which is based on the fact that in the analysis of [17], the adversary "wins" as soon as two keys in two different instances are the same. However, in our game, two keys in two different instances does not result in the adversary succeeding, as discussed above. This enables us to obtain a better bound, as in the following theorem:

Theorem 2.2. *Let E be an ideal cipher with key of length k and domain $\{0,1\}^n$. Then, for every $(t, Q, \vec{B}, \mu)$-adversary making at most T_E queries to E with adversarially chosen keys, it holds that*

$$\mathsf{AdvPRF}^Q_{\mathcal{A},E} \leq \min\left\{\frac{Q^3}{6 \cdot 2^{2\kappa}} + \frac{Q \cdot B\mathrm{max}^2}{2^n},\ \frac{Q^2}{2^{\kappa+1}} + \frac{\sum_{i=1}^Q (B_i)^2}{2^{n+1}}\right\} + \frac{\mu \cdot T_E}{2^\kappa}$$

where $B\mathrm{max} = \max\{B_1, \ldots, B_Q\}$.

[1] As shown in [6], an adversary can compute E on T_E different keys with some fixed input x (independently of the oracle), and can then query x to all of the function instances. If the adversary computed E on one of the actual keys, then it will distinguish. The probability that an actual key was queried in the first phase is $T_E/2^\kappa$ and thus overall this attack succeeds with probability $\frac{Q \cdot T_E}{2^\kappa}$.

The proof of Theorem 2.2 appears in Appendix A. For the parameters that we are interested in, it holds that $\frac{Q^3}{6\cdot 2^{2\kappa}} + \frac{Q\cdot B\max^2}{2^n} < \frac{Q^2}{2^{\kappa+1}} + \frac{\sum_{i=1}^{Q}(B_i)^2}{2^{n+1}}$. Thus, in the sequel, we will use the bound

$$\mathsf{AdvPRF}^{Q}_{\mathcal{A},E} \le \frac{Q^3}{6\cdot 2^{2\kappa}} + \frac{Q\cdot B\max^2}{2^n} + \frac{\mu\cdot T_E}{2^{\kappa}} \tag{2}$$

Assumption on AES. We make the following assumption on *AES* in the multi key scenario.

Assumption 1. *AES behaves like the ideal cipher E in the multi-instance setting. Thus, Eq. (2) holds for AES, with the addition of* $Q\cdot \mathsf{AdvPRP}_{AES}(t, 2B\max)$*, which is assumed to be extremely small, even for very large t and L.*

This is merely an assumption that AES (with a uniform random key) meets its design goals to be indistinguishable from a random permutation even after viewing a large amount of outputs. This also reflects the current state-of-the-art cryptanalysis of AES. We note that AES is weaker than an ideal cipher in the context of *related-key attacks*, which is outside the scope of its design goals. However, in the setting that we consider here, no weakness is known. The addition of $Q\cdot \mathsf{AdvPRP}_{AES}(t, 2B\max)$ is due to the fact that in our analysis in the proof of Theorem 2.2, at most $2B\max$ queries are made per AES instance.

2.4 Encryption Security

Black-box encryption game. Our main theorem can be applied to many different settings: CPA security, authenticated encryption, nonce misuse-resistant authenticated encryption, and more. In order to be able to apply the theorem in a general way, we begin by defining the notion of a black-box encryption game. Informally, this is an encryption experiment that can be defined via an oracle given to the adversary, and can be run using only black-box access to the block cipher. As we will show, all standard encryption games are of this type.

Formally, consider the following experiment outline. Let $\mathcal{O}$ be an oracle, let $\mathcal{A}$ be an adversary, and let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. We define:

Experiment $\mathbf{ExptEnc}_{\mathcal{A},\Pi,\mathcal{O}}$**:**

(1) Choose a random $b \leftarrow \{0,1\}$ and a random key $k \leftarrow \mathsf{Gen}(1^\kappa)$.
(2) Compute $b' \leftarrow \mathcal{A}^{\mathcal{O}(b,k,\cdot)}(1^\kappa)$.
(3) Output 1 if and only if $b' = b$.

Definition 2.3. An encryption experiment is called black box if it is of the form ExptEnc with a specified (possibly stateful) oracle $\mathcal{O}$, and $\mathcal{O}$ can be computed using black-box access to a function or series of functions.

By instantiating the oracle appropriately, it is possible to define standard eavesdropping adversaries, CPA-security (via an LR oracle), CCA-security, nonce-misuse resistance and more. See Appendix B for these specifications and the formal definition of security using them. We say that $\mathcal{A}$ is a $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$-nonce adversary, with $\vec{N}_E = (N_E^1, \ldots, N_E^Q)$, $\vec{N}_D = (N_D^1, \ldots, N_D^Q)$ and $\vec{B} = (B_1, \ldots, B_Q)$, if it runs in at most t steps, queries its encryption and decryption oracle with at most Q different nonces, queries the ith nonce with at most N_E^i encryption queries and at most N_D^i decryption queries, the number of blocks processed with the ith nonce in both encryption and decryption queries is B_i, the longest AAD is less than 2^a blocks and the longest message is less than 2^m blocks. We say that an adversary is a $(t, Q, N_E, \vec{N}_D, \vec{B}, a, m)$-IV adversary if it is as above, except that the number of encryption queries overall is N_E (the IV is chosen randomly so the adversary cannot influence the number of queries with the ith nonce).

2.5 Encryption Security with Many Random Functions

In order to prove security of the modes presented here, we will refer to an encryption experiment, denoted ExptEncRF, which is the same encryption experiment as above, except that the block cipher operations used in encryption and decryption are replaced by random functions (over the same domain as the block cipher). Furthermore, we consider multiple random functions via an indexed oracle, as in experiment ExptPRF.

The aim of this notion is to enable an analysis of the mode of operation via random functions, and relate this to the original construction. We stress that our comparison is of a block cipher to a random function and not to a random permutation. However, we will still be able to use this to obtain very good bounds, via the use of key derivation.

For an encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ that uses a block cipher as its underlying primitive, we denote by $(\mathsf{Gen}_f, \mathsf{Enc}_f, \mathsf{Dec}_f)$ an analogous scheme where Gen_f chooses a random function over the same domain and range as the block cipher, and each block cipher invocation in encryption and decryption is replaced by a call to the random function f. We note that this is only defined for modes of operation that do not invert the block cipher during decryption (thus, it is defined for CTR but not for CBC). The experiment is formally defined given an adversary $\mathcal{A}$, encryption scheme Π and oracle $\mathcal{O}$ for the appropriate security notion. As for the multi-key PRF experiment, we use an indexed oracle to choose which key is used. However, we wish to model the fact that a different function is used for each different nonce. Thus, for an oracle $\mathcal{O}(b, k, X)$, where X may be a vector of inputs, we define $\tilde{\mathcal{O}}(b, X)$ to return a response as would be computed by $\mathcal{O}(b, \cdots)$, but where the block cipher computation with key k is replaced with one of the random functions $f_1, \ldots, f_Q$. The choice of the random function is made based on the nonce/IV; specifically, a different function is used for each different nonce (or different IV). We define:

Experiment $\mathbf{ExptEncRF}^{Q}_{\mathcal{A},\Pi,\mathcal{O}}$**:**

(1) Choose a random $b \leftarrow \{0,1\}$, random keys $k_1, \ldots, k_Q \leftarrow \{0,1\}^\kappa$, and random functions $f_1, \ldots, f_Q$ under the constraint that if $k_i = k_j$ then $f_i = f_j$.
(2) Obtain $b' \leftarrow \mathcal{A}^{\tilde{\mathcal{O}}(\cdots)}(1^\kappa)$, where $\tilde{\mathcal{O}}(\cdots)$ is defined as above.
(3) Output 1 if and only if $b' = b$.

We define the advantage of the adversary to be

$$\mathsf{AdvEncRF}^{Q}_{\mathcal{A},\Pi,\mathcal{O}} = 2\cdot \mathrm{Prob}\left[\mathsf{ExptEncRF}^{Q}_{\mathcal{A},\Pi,\mathcal{O}} = 1\right] - 1.$$

3 THE MAIN THEOREM

In this section we prove our main theorem, that provides the security bounds when applying key derivation at every encryption.The theorem is very generic and can be used for different notions of security and different schemes to obtain different concrete bounds, as will be shown below. In Section 4, we show how to implement an efficient KDF with very good bounds.

3.1 Nonce-Based Encryption

We begin by considering the case of nonce-based encryption. Recall that $\mathcal{A}$ is a $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$-nonce adversary if it runs in at most t steps, queries its encryption and decryption oracle with at most Q different nonces, queries the ith nonce with at most N_E^i encryption queries and at most N_D^i decryption queries, the number of blocks processed with the ith nonce in both encryption and decryption queries is B_i, the longest AAD is less than 2^a blocks and the longest message is less than 2^m blocks.

Theorem 3.1 (nonce-based encryption schemes). *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a nonce-based encryption scheme using block cipher* E*, let* Π' *be the scheme obtained by applying the key derivation function* F *to the nonce in order to derive the key for encrypting the message, and let* Π'' *be the same as* Π' *except that a truly random function is used instead of* F*.*

Consider any black-box encryption game with oracle O*. Then, for every* $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$*-nonce adversary* $\mathcal{A}$*, there exists an* $(O(t), Q)$*-adversary* $\mathcal{A}_1$ *for* F*, and an* $(O(t), Q, \vec{B}, \mu)$*-adversary* $\mathcal{A}_2$ *for* E *where* μ *depends on* Π*, such that*

$$\mathsf{AdvEnc}_{\Pi',\mathcal{A},O} = \frac{1}{2} \cdot \mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{1}{2} \cdot \mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E} + \mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},O}.$$

Proof. Let $\mathcal{A}$ be a $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$-nonce-adversary in $\mathsf{ExptEnc}$ with scheme Π'. We claim that there exists an $(O(t), Q)$-adversary $\mathcal{A}_1$ for $\mathsf{ExptKDF}$ and F_1 such that

$$\mathsf{AdvEnc}_{\Pi',\mathcal{A},O} = \frac{1}{2} \cdot \mathsf{AdvKDF}_{\mathcal{A}_1,F_1} + \mathsf{AdvEnc}_{\Pi'',\mathcal{A},O}. \quad (3)$$

In order to see this, observe that the only difference between Π' and Π'' is whether the KDF is F or a truly random function. We therefore construct an adversary $\mathcal{A}_1$ that attempts to distinguish F from random, using $\mathcal{A}$.

Adversary $\mathcal{A}_1$ works in $\mathsf{ExptKDF}$ and attempts to distinguish F from random. $\mathcal{A}_1$ invokes $\mathcal{A}$ and simulates an execution of $\mathsf{ExptEnc}_{\Pi',\mathcal{A},O}$ with $\mathcal{A}$. Specifically, upon receiving any encryption or decryption oracle query, $\mathcal{A}_1$ calls its own oracle with the nonce as input and uses the result as the key to carry out the encryption or decryption, respectively. At the end of the experiment, $\mathcal{A}_1$ outputs 1 if and only if $\mathcal{A}$ outputs $b' = b$. It is clear that if $\mathcal{A}_1$'s oracle is F then it perfectly simulates $\mathsf{ExptEnc}_{\Pi',\mathcal{A},O}$, whereas if $\mathcal{A}_1$'s oracle is a truly random function then it perfectly simulates $\mathsf{ExptEnc}_{\Pi'',\mathcal{A},O}$. Let $\mathsf{ExptKDF}^0_{\mathcal{A}_1,F}$ denote an execution of $\mathsf{ExptKDF}_{\mathcal{A}_1,F}$ where $b = 0$, and likewise $\mathsf{ExptKDF}^1_{\mathcal{A}_1,F}$ where $b = 1$. Then,

$$\begin{aligned}
&\mathsf{Prob}[\mathsf{ExptKDF}_{\mathcal{A}_1,F} = 1] \\
&= \frac{1}{2}\mathsf{Prob}[\mathsf{ExptKDF}^0_{\mathcal{A}_1,F} = 1] + \frac{1}{2}\mathsf{Prob}[\mathsf{ExptKDF}^1_{\mathcal{A}_1,F} = 1] \\
&= \frac{1}{2}\mathsf{Prob}[\mathsf{ExptEnc}_{\Pi'',\mathcal{A},O} = 0] + \frac{1}{2}\mathsf{Prob}[\mathsf{ExptEnc}_{\Pi',\mathcal{A},O} = 1]
\end{aligned}$$

where the last equality is due to the fact that we wish to analyze when $\mathcal{A}_1$ outputs $b' = b$ (when $b = 0$ we have that $b' = b$ if $\mathcal{A}$ is incorrect in $\mathsf{ExptEnc}_{\Pi'',\mathcal{A},O}$, whereas when $b = 1$ we have that $b' = b$ if $\mathcal{A}$ is correct in $\mathsf{ExptEnc}_{\Pi',\mathcal{A},O}$). Using the fact that

$$\mathsf{Prob}[\mathsf{ExptEnc}_{\Pi'',\mathcal{A},O} = 0] = 1 - \mathsf{Prob}[\mathsf{ExptEnc}_{\Pi'',\mathcal{A},O} = 1]$$

we conclude that

$$\begin{aligned}
\mathsf{Prob}[\mathsf{ExptKDF}_{\mathcal{A}_1,F} = 1] \;=\; &\frac{1}{2} + \frac{1}{2}\mathsf{Prob}[\mathsf{ExptEnc}_{\Pi',\mathcal{A},O} = 1] \\
&- \frac{1}{2}\mathsf{Prob}[\mathsf{ExptEnc}_{\Pi'',\mathcal{A},O} = 1].
\end{aligned}$$

Thus,

$$\begin{aligned}
&\mathsf{AdvKDF}_{\mathcal{A}_1,F} = 2 \cdot \mathsf{Prob}[\mathsf{ExptKDF}_{\mathcal{A}_1,F} = 1] - 1 \\
&= \mathsf{Prob}[\mathsf{ExptEnc}_{\Pi',\mathcal{A},O} = 1] - \mathsf{Prob}[\mathsf{ExptEnc}_{\Pi'',\mathcal{A},O} = 1] \\
&= 2 \cdot \mathsf{AdvEnc}_{\Pi',\mathcal{A},O} + 1 - 2 \cdot \mathsf{AdvEnc}_{\Pi'',\mathcal{A},O} - 1 \\
&= 2 \cdot \mathsf{AdvEnc}_{\Pi',\mathcal{A},O} - 2 \cdot \mathsf{AdvEnc}_{\Pi'',\mathcal{A},O}
\end{aligned}$$

and so $\mathsf{AdvEnc}_{\Pi',\mathcal{A},O} = \frac{1}{2} \cdot \mathsf{AdvKDF}_{\mathcal{A}_1,F} + \mathsf{AdvEnc}_{\Pi'',\mathcal{A},O}$.

Noting that $\mathcal{A}_1$ runs in essentially the same time as $\mathcal{A}$ and queries its oracle once for every different nonce, we have that $\mathcal{A}_1$ runs at most $O(t)$ steps and makes at most Q oracle queries. This concludes the proof of Eq. (3).

Next, we claim that for every $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m))$-adversary $\mathcal{A}$ for $\mathsf{ExptEnc}$, there exists an $(O(t), Q, \vec{B}, \mu)$-adversary $\mathcal{A}_2$ for E in $\mathsf{ExptPRF}^Q$, such that

$$\mathsf{AdvEnc}_{\Pi'',\mathcal{A},O} = \frac{1}{2} \cdot \mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E} + \mathsf{AdvEncRF}^{Q}_{\mathcal{A},\Pi'',O}. \quad (4)$$

In order to see this, note that if we replace the block cipher E in Π'' with a truly random function, then by definition the resulting experiment is exactly that of $\mathsf{ExptEncRF}^Q_{\mathcal{A},\Pi'',O}$. Thus, we construct an adversary $\mathcal{A}_2$ who simulates the experiment with scheme Π'' for $\mathcal{A}$, and using its oracle to compute the encryption/decryption responses. Observe that in order to simulate the entire experiment, $\mathcal{A}_2$ needs to call its oracle for every block encrypted/decrypted, and needs to call a different function for each nonce. Thus, it calls its ith oracle B_i times. At the end of the experiment, $\mathcal{A}_2$ outputs 1 if and only if $\mathcal{A}$ outputs $b' = b$.

As above, if $\mathcal{A}_2$ receives a series of random functions for an oracle then it simulates $\mathsf{ExptEncRF}^Q_{\mathcal{A},\Pi'',O}$, and if $\mathcal{A}_2$ receives a series of pseudorandom functions E for an oracle then it simulates $\mathsf{ExptEnc}_{\Pi'',\mathcal{A},O}$. Thus, denoting by $\mathsf{ExptPRF}^{Q,b}$ the experiment where the bit b is chosen, we have:

$$\begin{aligned}
&\mathsf{Prob}[\mathsf{ExptPRF}^Q_{\mathcal{A}_2,E} = 1] \\
&= \frac{1}{2}\mathsf{Prob}[\mathsf{ExptPRF}^{Q,0}_{\mathcal{A}_2,E} = 1] + \frac{1}{2}\mathsf{Prob}[\mathsf{ExptPRF}^{Q,1}_{\mathcal{A}_2,E} = 1] \\
&= \frac{1}{2}\mathsf{Prob}[\mathsf{ExptEncRF}_{\Pi'',\mathcal{A},O} = 0] + \frac{1}{2}\mathsf{Prob}[\mathsf{ExptEnc}_{\Pi'',\mathcal{A},O} = 1]
\end{aligned}$$

where the last equality is due to the fact that we wish to analyze when $\mathcal{A}_2$ outputs $b' = b$, as above. Using the same manipulation as above, we have that Eq. (4) holds.

Combining Equations (3) and (4), we conclude that

$$\mathsf{AdvEnc}_{\Pi',\mathcal{A},O} = \frac{1}{2}\cdot \mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{1}{2}\cdot \mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E} + \mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},O},$$

completing the proof. □

3.2 Security for Random-IV Encryption

In the case of encryption using a random IV, the adversary does not input nonces. Furthermore, since an IV can repeat and does so with high probability when the birthday bound is approached, the analysis is different to that of a "nonce-respecting" adversary for which unique nonces are guaranteed. We treat this by bounding the number of times that an IV will repeat and then using this as the upper bound on the number of times each random function is used in ExptEncRF. Towards this, we use the bound in Theorem 2.1, that states that the probability that at least one IV of length ℓ bits repeats at least 4 times out of q randomly selected IVs is less than $\frac{q^4}{24\cdot 2^{3\ell}}$.

We now apply this by bounding the number of times the ith random function is used by 2^{m+2}, where $2^m - 1$ is the maximum length of any message encrypted or decrypted in the encryption experiment. The fact that no more than 3 collisions are observed is taken into account by adding $\frac{q^4}{24\cdot 2^{3\ell}}$ to the bound. Furthermore, we take 2^{m+2} as an upper bound on the number of blocks encrypted using any single nonce since we "allow" 3-way collisions and so the same IV can be used to encrypt three messages, each of length less than 2^m.)

Given the above, it follows that for every encryption scheme Π with IV of length ℓ bits and every $(t, Q, N_E, \vec{N}_D, \vec{B}, a, m)$-IV adversary $\mathcal{A}$, there exists a $(t, Q', \vec{N}_E, \vec{N}_D, \vec{B}', a, m)$-nonce adversary $\mathcal{A}'$ such that

$$\mathsf{AdvEnc}_{\Pi,\mathcal{A},O} \le \mathsf{AdvEnc}_{\Pi,\mathcal{A}',O} + \frac{{N_E}^4}{24\cdot 2^{3\ell}}. \tag{5}$$

where $Q' = Q + N_E$, and for every i it holds that $N^i_E \le 3$ and $B'_i \le B_i + 2^{m+2}$. (Recall that in the random-IV game, Q only refers to the nonces in decryption queries; therefore the number of nonces overall in the nonce-game with $\mathcal{A}'$ is at most $Q + N_E$. Also, $B'_i \le B_i + 2^{m+2}$ since the number of blocks encrypted or decrypted with the ith nonce is equal to the number decrypted by $\mathcal{A}$ plus an additional 3 encryptions at most.) The adversary $\mathcal{A}'$ just invokes $\mathcal{A}$ and chooses the nonces used in encryption at random. As long as there is no 4-way collision on an IV, it follows that $\mathcal{A}'$ distinguishes successfully whenever $\mathcal{A}$ does. Thus, $\mathcal{A}'$'s advantage is *at least* that of $\mathcal{A}$, minus the probability of a 4-way collision. Eq. (5) follows.

Combining Theorem 3.1 with Eq. (5), we have:

Theorem 3.2 (random-IV-based encryption schemes). *Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a random-IV-based encryption scheme using block cipher E and an IV of length ℓ bits, let Π' be the scheme obtained by applying the key derivation function F to the nonce in order to derive the key for encrypting the message, and let Π'' be the same as Π' except that a truly random function is used instead of F.*

Consider any black-box encryption game with oracle O. Then, for every $(t, Q, N_E, \vec{N}_D, \vec{B}, a, m)$-IV adversary $\mathcal{A}$, there exists an $(O(t), Q')$-adversary $\mathcal{A}_1$ for F, and an $(O(t), Q', \vec{B}', \mu)$-adversary $\mathcal{A}_2$ for E where μ depends on Π, and a $(t, Q', \vec{N}_E, \vec{N}_D, \vec{B}', a, m)$-nonce adversary $\mathcal{A}_3$ for Π'' such that

$$\mathsf{AdvEnc}_{\Pi',\mathcal{A},O} = \frac{1}{2}\cdot \mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{1}{2}\cdot \mathsf{AdvPRF}^{Q'}_{\mathcal{A}_2,E} + \mathsf{AdvEncRF}^{Q'}_{\Pi'',\mathcal{A}_3,O} + \frac{{N_E}^4}{24\cdot 2^{3\ell}},$$

where $Q' = Q + N_E$, and for every i it holds that $N^i_E \le 3$ and $B'_i \le B_i + 2^{m+2}$.

A version with no IV collisions. Most encryption schemes, like CTR and AES-GCM, fail as soon as an IV repeats at all. It is therefore useful to repeat Theorem 3.2 for the case that an IV does not repeat at all. In this case, we consider $\mathcal{A}_3$ who is *nonce-respecting* and add $\frac{q^2}{2^{\ell+1}}$ since this is the probability that some (2-way) collision will happen in the IV. We therefore have:

Theorem 3.3 (random-IV encryption – no collision version). *Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a random-IV-based encryption scheme using block cipher E and an IV of length ℓ bits, let Π' be the scheme obtained by applying the key derivation function F to the nonce in order to derive the key for encrypting the message, and let Π'' be the same as Π' except that a truly random function is used instead of F.*

Consider any black-box encryption game with oracle O. Then, for every $(t, Q, N_E, \vec{N}_D, \vec{B}, a, m)$-IV adversary $\mathcal{A}$, there exists an $(O(t), Q')$-adversary $\mathcal{A}_1$ for F, and an $(O(t), Q', \vec{B}', \mu)$-adversary $\mathcal{A}_2$ for E where μ depends on Π, and a $(t, Q', \vec{N}_E, \vec{N}_D, \vec{B}', a, m)$-nonce respecting adversary $\mathcal{A}_3$ for Π'' such that

$$\mathsf{AdvEnc}_{\Pi',\mathcal{A},O} = \frac{1}{2}\cdot \mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{1}{2}\cdot \mathsf{AdvPRF}^{Q'}_{\mathcal{A}_2,E} + \mathsf{AdvEncRF}^{Q'}_{\Pi'',\mathcal{A}_3,O} + \frac{{N_E}^2}{2^{\ell+1}},$$

where $Q' = Q + N_E$, and for every i it holds that $B'_i \le B_i + 2^m$.

4 KEY DERIVATION WITH GOOD BOUNDS

We describe a new Key Derivation Function (KDF) here, which we call DeriveKey. It is efficient and simple to implement, and obtains very good bounds. The KDF works by truncating outputs of a pseudorandom permutation. Concretely, we apply the AES pseudorandom permutation (using a "master" key) to the input nonce and an index, and truncate each 128-bit output to 64 bits. Thus, a 128-bit key is derived by applying AES twice, and a 256-bit key is derived by applying AES four times. Different amount of key material bits are derived in a similar way, depending on the application. Note also that the number of truncated bits can be changed, according to the desired tradeoff between the performance and the security bounds.

We use a nonce of 96-bits, for convenience, since this is standard practice for existing AES-GCM interfaces. As an example, consider AES-GCM [15] that uses DeriveKey. The KDF is used for deriving an AES encryption key of length 128 or 256 (for encryption with AES128 or AES256, respectively), and a GHASH hash key of length 128. These 256 or 384 key material bits can be obtained by 4 or 6

AES invocations. The algorithm is formally described in Fig. 1 (for this amount of key bits).

DeriveKey(K, N)

```
Context: encryption-keylength (= 128 or 256)
if encryption-keylength = 128 AES is AES128, else AES is AES256
Key: K
Input: N (96 bits)
If encryption-keylength =128 then repeats = 4, else repeats = 6
for i from 0 to repeats-1 do
  Tj = AES (K, N [95:0] || IntToString32 (i))
end
K1 = T1 [63:0] || T0 [63:0]
If keylength=128 then
   K2 = T3 [63:0] || T2 [63:0]
else
   K2 = T5 [63:0] || T4 [63:0] || T3 [63:0] || T2 [63:0]
end
Output: K1 (128 bits), K2 (128 or 256 bits)
```

Figure 1: DeriveKey uses the KDF ("master") key K to derive two new keys: K1 (128 bits) and K2 (128 or 256 bits).

Intuitively, truncating the output of AES is advantageous since it lowers the distinguishing probability of AES from a pseudorandom *function* (versus permutation). Specifically, using a random permutation has the disadvantage that derived keys are distinguishable from random at around the birthday bound. In contrast, a random function suffers from no such limitation, and thus a pseudorandom function (versus permutation) is advantageous in this sense. By truncating the pseudorandom permutation, the result is no longer a permutation. As we will see below, the more a random permutation is truncated, the closer it becomes to a random function.

The following lemma is proven in [9], which explores the problem of distinguishing the truncation of a randomly chosen permutation from a random function. The upper bound on the distinguishing advantage (originally due to [20]), is simplified in [9] to the easy-to-use form

$$\mathbf{Adv}_{n,m}(\tilde{q}) \leq \min\left(\frac{\tilde{q}^2}{2^{n+1}}, \frac{\tilde{q}}{2^{\frac{m+n}{2}}}, 1\right) \tag{6}$$

where $\tilde{q} \leq \frac{3}{4} \cdot 2^n$ is the number of queries made by the distinguisher, and where the (randomly chosen) permutation over n bits is truncated to $n-m$ bits (for some $1 \leq m < n$). We comment that [10] have recently proved that this bound is essentially tight. Plugging in $n = 128$ and $m = 64$ as in our DeriveKey procedure, and recalling that $\mathsf{AdvPRP}_{AES}(t, B)$ denotes the probability of distinguishing AES from a random permutation with B queries to the block cipher, we have:

Lemma 4.1 (The DeriveKey advantage). *For every (t, Q)-adversary $\mathcal{A}$ for* DeriveKey *(obtaining Q pairs of keys (K_1, K_2) of overall length 256 and 384 bits, respectively), it holds that*

$$\mathsf{AdvKDF}_{\mathcal{A},\mathrm{DeriveKey}} \leq \mathsf{AdvPRP}_{AES}(t, 6Q) + \min\left\{\frac{36Q^2}{2^{129}}, \frac{6Q}{2^{96}}, 1\right\}.$$

Proof. To obtain one pair of keys (K_1 and K_2), DeriveKey computes at most 6 AES operations using the KDF key (when K_2 is 128-bits long, only 4 AES operations are required). Thus, an adversary $\mathcal{A}$ making at most $Q \leq \frac{3}{24} \cdot 2^{128}$ queries to DeriveKey and obtaining Q pairs of keys (K_1, K_2) can be simulated by an adversary $\mathcal{A}'$ making at most $\tilde{q} \leq \frac{3}{4} \cdot 2^{128}$ queries to AES. If a truly random permutation were used instead of AES, then plugging the above into Eq. (6), with $\tilde{q} = 6Q$, $n = 128$ and $m = 64$, we derive that the distinguishing advantage between the result of the KDF and a truly random function is

$$\min\left\{\frac{36Q^2}{2^{129}}, \frac{6Q}{2^{96}}, 1\right\},$$

since

$$\frac{\tilde{q}^2}{2^{n+1}} = \frac{36Q^2}{2^{129}} \quad \text{and} \quad \frac{\tilde{q}}{2^{\frac{m+n}{2}}} = \frac{6Q}{2^{96}}$$

and the bound on the number of queries $\tilde{q} \leq \frac{3}{4} \cdot 2^n$ is equivalent to $6Q \leq \frac{3}{4} \cdot 2^n$, implying that $Q \leq \frac{3}{24} \cdot 2^n$. Finally, since AES is a pseudorandom permutation and not a truly random permutation, the bound includes $\mathsf{AdvPRP}_{AES}(t, 6Q)$, which is the maximum advantage over all adversaries in distinguishing (a single instance of) AES from a random *permutation*, when running in t steps and using $6Q$ queries. This completes the proof. □

Actual DeriveKey bounds. The crucial point to observe in the bound in Lemma 4.1 is that the advantage is the *minimum* of $O(Q/2^{96})$ and $O(Q^2/2^{192})$. Thus, the birthday bound of $Q \approx 2^{64}$ for distinguishing AES from a pseudorandom function is not the smallest upper bound here. Rather, at $Q = 2^{64}$, the distinguishing advantage is only $\frac{6}{2^{32}}$ since the linear term of $\frac{6Q}{2^{96}}$ is much smaller for large Q. Thus, it is possible to derive far more keys than by using counter-mode.[2] It is worth noting that for small values of Q, the quadratic term is smaller; however, the minimum is so small in these cases that this is irrelevant. In conclusion, using the NIST bounds for AES-GCM that allow for 2^{-32} advantage, we are still able to derive approximately 2^{64} different keys. We remark that naive key derivation that utilizes standard AES without truncation would provide an adversarial advantage of $O(Q^2/2^{129})$, which is much higher than our method for large values of Q.

Efficiency. In order to compute DeriveKey, the number of AES invocations is 4 with encryption key-length = 128, and 6 with encryption key-length = 256. Importantly, these AES computations are parallelizable. Furthermore, the AES key schedule for the master key K can be pre-computed and cached, and so we can ignore the key expansion overhead. Thus, on a modern CPU with AES instructions (AES-NI) with throughput of 1 cycle and latency of 4 cycles (e.g., the Intel processor, microarchitecture codename Skylake), DeriveKey consumes ~ 50 cycles in the first case, and ~ 65 in the second case. To see why, consider the first case with 4 independent AES computations. These are computed by 4×10 AES rounds, executed via 40 AESENC instruction invocations, that consume 40+4 cycles, plus a few cycles for aligning the data. Such overheads are inconsequential in most situations.

The performance of key-derived schemes. Let Π be a nonce based encryption scheme. From the performance viewpoint, the associated key-derived scheme Π′ seems to pay only the extra cost of computing the KDF over the nonce (prior to running Π with the derived keys). However, there is an implicit additional performance

[2] If keys are derived by running $AES_K(i)$ for $i = 1, 2, \ldots$, then after 2^{64} key derivations, the derived keys cannot be argued to be indistinguishable from random. This is because a truly random key derivation mechanism would provide some colliding keys, whereas the result of AES in counter mode would never collide.

overhead of computing the key schedule for the derived key. This is due to the fact that in Π it is possible to pre-compute the round keys once, but in Π' the round keys cannot be pre-computed and. Therefore, the cost of the key schedule must be added. Specifically, if Π uses AES as the block cipher, then the performance of Π' includes the cost of an additional AES key expansion, on top of the cost of the KDF itself. When the encryption is carried out on modern processors with fully pipelined AES-NI, then the key expansion instruction `aeskeygenassist` should not be used, since it is not efficient. Rather, as described in [14], key expansion can be carried out using the AES-NI round functions (and additional shifts), and this can be interleaved with the encryption of the first few blocks. This reduces the overall latency of the key expansion and encryption. Table 1 shows the performance of AES key expansion alone using `aeskeygenassist` (note that beyond it being expensive, it cannot even be pipelined, because `aeskeygenassist` does not have throughput 1), compared with our optimized method of AES key expansion using the AES-NI round function, compared with our key expansion interleaved with the encryption of a few blocks. The maximum performance advantage that this technique offers is at $x = 4$ blocks and thus we show the performance in numbers of cycles up to this point. As we see, the additional overhead of the key expansion is negligible when the messages are not extremely short. Specifically, the number of cycles per byte of AES-128 encryption *together with key expansion* using our method, is 3.15 for 1 block, 1.8 for 2 blocks, 1.37 for 3 blocks and 1.19 for 4 blocks.

Key length	Key expansion – `aeskeygenassist`	Optimized key expansion	Key expansion interleaved with x blocks			
			$x = 1$	$x = 2$	$x = 3$	$x = 4$
128	111	48	50	58	66	76
256	146	80	85	99	-	-

Table 1: Optimized code performance i*n cycles* of AES key expansion and key expansion interleaved with the encryption of a few blocks, run on Intel microarchitecture codename Skylake.

5 BOUNDS FOR COUNTER-BASED MODES

In this section, we analyze the bounds obtained in CTR mode and AES-GCM, using our key derivation method.

5.1 CTR Encryption with Unique Nonces

We begin by considering the CPA-security of CTR mode with unique nonces (i.e., with a nonce-respecting adversary). We first provide the bound for basic CTR without key derivation, and then provide the bound using our key-derivation proposal. We consider the standard version of CTR with an IV of length $\ell < n$, and where messages of length at most $2^{n-\ell}$ can be encrypted with each IV. The counter is taken to be the IV concatenated with the block number (from 0 to $2^{n-\ell}-1$). We consider a $(t, Q, \vec{N}_E, \vec{B}, m)$-adversary $\mathcal{A}$ (in this setting there are no decryption queries and no AAD; thus $\vec{N}_D$ and a are not referenced). For this scheme, $m \leq 2^{n-\ell}$ and its actual value makes no difference beyond that. Since $\mathcal{A}$ is nonce-respecting, we have that $\vec{N}_E$ is the vector of all ones. Let B_i be the length (in blocks) of the plaintext encrypted using the ith nonce.

Basic CTR. The overall number of blocks encrypted with E is $\sum_{i=1}^{Q} B_i$, and these are all encrypted under a single key. It is well known that the distinguishing probability in this case is upper bounded $\frac{\left(\sum_{i=1}^{Q} B_i\right)^2}{2^{n+1}}$.[3] Concretely for a 128-bit block cipher like AES, when the overall number of blocks encrypted reaches 2^{48}, the security level is only 2^{-32}. Furthermore, if 2^{64} blocks are encrypted, then security is broken with very high probability; this could happen if 2^{48} plaintext each of length 2^{16} were encrypted.

CTR with key derivation. We apply Theorem 3.1 and thus we need to analyze $\mathcal{A}$'s advantage when interacting in an encryption experiment where E is replace with a truly random *function*. We consider CPA security here, and thus we consider an LR-oracle experiment; see Appendix B. Let LR denote the LR-oracle in the experiment; we therefore need to bound $\mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},LR}$. In this experiment, all the counters are encrypted using a truly random function. Now, since all counters are guaranteed to be unique (because $\mathcal{A}$ is nonce-respecting and so $N_E^i = 1$ for all i), the output pad is random and so encryption is essentially a one-time pad. Thus, $\mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},LR} = 0$ for all parameter settings. By Theorem 3.1, we therefore conclude that for every $(t, Q, \vec{N}_E, \vec{B}, m)$-nonce respective adversary $\mathcal{A}$, there exists an $(O(t), Q)$-adversary $\mathcal{A}_1$ for F and an $(O(t), Q, \vec{B}, \mu)$-adversary $\mathcal{A}_2$ for E for μ to be specified, such that

$$\begin{aligned}\mathsf{AdvEnc}^{\mathrm{cpa}}_{\Pi',\mathcal{A},LR} &= \frac{1}{2}\cdot\mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{1}{2}\cdot\mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E} \\ &\quad + \mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},LR} \\ &= \frac{1}{2}\cdot\mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{1}{2}\cdot\mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E}.\end{aligned}$$

Observe that in this setting, for every two different nonces the inputs to the block ciphers are all different. Thus, the same input is never used for two different keys, and the maximum query frequency μ equals 1. For simplicity, we take the first term in the bound in Eq. (2) of Theorem 2.2. Letting $B\max = \max\{B_i\}$, we have that each key is used $B\max$ times, and so

$$\mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E} \leq \frac{Q^3}{6\cdot 2^{2\kappa}} + \frac{Q\cdot B\max^2}{2^n} + \frac{T_E}{2^\kappa}.$$

Applying Lemma 4.1 for AdvKDF, we conclude that

$$\begin{aligned}\mathsf{AdvEnc}_{\Pi',\mathcal{A},LR} \leq \frac{1}{2}\cdot\Bigg(&\mathsf{AdvPRP}_{AES}(t, 6Q) + \min\left\{\frac{36Q^2}{2^{129}}, \frac{6Q}{2^{96}}, 1\right\} \\ &+ \frac{Q^3}{6\cdot 2^{2\kappa}} + \frac{Q\cdot B\max^2}{2^n} + \frac{T_E}{2^\kappa}\Bigg). \qquad (7)\end{aligned}$$

Consider now the case of $Q = 2^{48}$ and $B\max = 2^{16}$, as above, meaning that at most 2^{16} blocks are encrypted per nonce. First, note that the advantage of distinguishing AES from a random *permutation* (AdvPRP_{AES}) for such a Q is extremely small. Next, for such a Q, the term $6Q/2^{96}$ is minimum (at less than 2^{-45}). Now, regarding AdvPRF, with $\kappa = n = 128$ (as in AES-128) and reasonable values of T_E, the dominant term in the advantage is $\frac{Q\cdot B\max^2}{2^n}$. For these parameters, we have that it equals $\frac{2^{48}\cdot 2^{32}}{2^{128}} = 2^{-48}$. Thus, overall,

[3] This bound is tight since an adversary in an IND-CPA game can distinguish between encryptions of blocks that are all 0 versus encryptions of random blocks. This is due to the fact that encryptions of 0 will all have distinct ciphertexts, whereas encryptions of random blocks will result in a collision between two ciphertext blocks at the birthday bound.

the advantage is approximately 2^{-46} (since all advantages are multiplied by 1/2). In basic counter mode, this is broken with high probability, whereas here this is well within the range of being secure.

Encryption limits. Given that the dominant terms are $\frac{6Q}{2^{96}}$ and $\frac{Q \cdot B\max^2}{2^n}$, for AES-128 we have that one can encrypt almost $Q = 2^{64}$ messages of length $B\max = 2^{16}$ blocks and still remain within the NIST-recommended limit of 2^{-32} advantage. This is because

$$\frac{1}{2} \cdot \frac{6Q}{2^{96}} = \frac{3}{2^{32}} \quad \text{and} \quad \frac{Q \cdot B\max^2}{2^n} = \frac{2^{64} \cdot \left(2^{16}\right)^2}{2^{128}} = 2^{-32}.$$

This is *way beyond* the birthday bound. Alternative choices of parameters for longer messages yield that for $Q = 2^{32}$ messages of length $B\max = 2^{32}$ blocks or $Q = 2^{48}$ messages of length $B\max = 2^{24}$ blocks, the advantage is 2^{-32}, because in both cases $\frac{Q \cdot B\max^2}{2^n} = 2^{-32}$.

Remark 1. *We observe that many more blocks can be encrypted when encrypting many smaller messages than when encrypting fewer large messages. This is due to the fact that the key derivation is per message, and thus is more effective when more messages are encrypted. Of course, this is only true due to our truncated key derivation method; were we to use standard CTR key derivation, we would not be able to encrypt more than 2^{48} messages under any circumstances.*

5.2 CTR Encryption with Random IVs

We now proceed to CTR with random nonces. As in Section 5.1, we consider IVs of length ℓ. Let B_i denote the length (in blocks) of the ith message encrypted.

Basic CTR. An adversary can distinguish if an IV repeats *or* if there is a collision in the blocks (as in the unique nonce case). Thus, we have that the distinguishing probability is $\frac{{N_E}^2}{2^{\ell+1}} + \frac{\left(\sum_{i=1}^{Q} B_i\right)^2}{2^{n+1}}$, where N_E is the number of encryption queries. When taking $\ell = 96$ as is standard practice for AES-CTR, this means that at $N_E = 2^{48}$ messages the scheme is broken with very high probability. As a result, NIST recommends that CTR with random IVs not be used for more that $N_E = 2^{32}$ messages (guaranteeing that the distinguishing probability is below 2^{-32}). Thus, CTR with a random IV cannot be used to encrypt many messages. However, note that if only 2^{32} messages are encrypted, but each is of length 2^{32}, then the distinguishing probability is still very high due to the $\frac{\left(\sum_{i=1}^{Q} B_i\right)^2}{2^{n+1}}$ factor. Thus, basic CTR cannot be used to encrypt many messages *or* many blocks. Specifically, in order to maintain a distinguishing probability below 2^{-32} one must encrypt at most 2^{32} messages and at most 2^{48} blocks overall.

CTR with key derivation. We apply Theorem 3.3 and so

$$\begin{aligned}\mathsf{AdvEnc}_{\Pi',\mathcal{A},LR} \;<\; & \frac{1}{2} \cdot \mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{Q^3}{6 \cdot 2^{2\kappa}} + \frac{Q^2}{2^{\ell+1}} \\ & + \frac{1}{2} \cdot \mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E} + \mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A}_3,LR}\end{aligned}$$

where $\mathcal{A}_3$ in the AdvEncRF experiment is a nonce-respecting making $Q = N_E$ encryption queries, and for every i it holds that $B'_i \leq B_i + 2^m$. (Note that we consider CPA security here only and so there are no decryption queries, meaning that $Q = N_E$.)

We begin by bounding $\mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A}_3,LR}$. In this game, as long as all IVs are unique, the adversary's advantage is 0 as in Section 5.1. Thus, $\mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A}_3,LR} = 0$; recall that $\mathcal{A}_3$ here is already nonce respecting. Using the bound on AdvPRF from Section 5.1, we have that

$$\begin{aligned}\mathsf{AdvEnc}_{\Pi',\mathcal{A},LR} < & \frac{1}{2} \cdot \mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{Q^3}{6 \cdot 2^{2\kappa}} \\ & + \frac{1}{2} \cdot \left(\frac{Q^3}{6 \cdot 2^{2\kappa}} + \frac{Q \cdot B\max^2}{2^n} + \frac{T_E}{2^\kappa} \right) + \frac{Q^2}{2^{\ell+1}}.\end{aligned}$$

(The maximum query frequency here μ equals 1 here, for the same reason as in Section 5.1. Of course, here it is possible that the same *IV* is used twice. However, this already results in the adversary winning the game, and in any case just results in the same input to the same key and not the same input to different keys.)

The dominant terms here are $\frac{Q \cdot B\max^2}{2^n}$ and $\frac{Q^2}{2^{\ell+1}}$. The crucial difference between here and the basic CTR case is that we have replaced $\frac{\left(\sum_{i=1}^{Q} B_i\right)^2}{2^{n+1}}$ with $\frac{Q \cdot B\max^2}{2^n}$. Concretely, consider the example of $Q = 2^{32}$ messages, each of length $B\max = 2^{32}$. In basic CTR this is insecure. In contrast, here we have $\frac{Q \cdot B\max^2}{2^n} = 2^{-32}$ and so this is still secure. (Using a similar analysis as in Section 5.1, the advantage in AdvKDF is very small and so can be ignored.)

Remark 2. *It is interesting to note that when using our method for CTR with a random IV, we do not gain anything in the* number of messages *being encrypted. However, we do gain significantly with the overall number of blocks. Thus, it is possible to encrypt* long *messages using our key derivation method, and security is maintained. In contrast, when considering CTR with unique nonces, we also gain significantly with respect to the number of messages encrypted.*

5.3 AES-GCM with Unique Nonces

AES-GCM is an authenticated-encryption mode of operation; it uses counter mode, with the addition of an authenticator. We analyze this mode in Appendix C and show that we obtain the same bounds as for CTR with unique nonces. In particular, although an additional term of $\frac{(2^a+2^m) \cdot \sum_{i=1}^{Q} N_D^i}{2^n}$ is added due to the fact that the setting allows decryption queries, this is insignificant. Thus, it is possible to encrypt $Q = 2^{64}$ messages of length $B\max = 2^{16}$, $Q = 2^{48}$ messages of length 2^{24}, or $Q = 2^{32}$ messages of length 2^{32}, and still remain within the NIST-recommended limit of 2^{-32} advantage. See Appendix C for the full analysis.

5.4 AES-GCM with Random IVs

The analysis here is similar to the that of CTR mode with random IV's (Section 5.2), with the analysis of AES-GCM (Section 5.3). The results are analogous, and lead to the same conclusions.

6 AES-GCM-SIV – BETTER NONCE-MISUSE RESISTANCE

GCM-SIV is a very fast nonce-misuse resistant AEAD mode of operation that was presented in [11]. (Hereafter, GCM-SIV refers to

the two-key variant of [11], with AES as the block cipher.) GCM-SIV has different performance characteristics for encryption and decryption. For encryption, it is slower than AES-GCM, because achieving full nonce-misuse resistance requires, by definition, two (*serialized*) passes over the data. Nevertheless, since GCM-SIV can use the same CPU instructions that accelerate AES-GCM, optimized implementations run GCM-SIV (for 128-bit keys) at less than one cycle per byte on modern processors (roughly 2/3 of the speed of nonce-respecting AES-GCM). On the other hand, GCM-SIV decryption runs at almost the same speed as AES-GCM.

Informally, GCM-SIV uses a universal-hash key ($K1$) and an encryption key ($K2$), applies a universal hash function (GHASH) with $K1$ to the encoded *AAD* (additional authentication data) and *MSG* (plaintext to be encrypted), and generates an authentication tag by AES-encrypting the hash value, XOR-ed with the nonce, under $K2$. Finally, the plaintext *MSG* is encrypted with AES in CTR mode, using $K2$, and with an initial counter derived from the authentication tag. This strategy means that the initial counter (effective nonce for the CTR encryption) is pseudorandom for every different nonce/message pair. Thus, even if the actual nonce repeats, the effective nonce used to mask the encryption is different for different messages. The security bounds of GCM-SIV were proven in [11, Theorem 4.3], as follows:

Theorem 6.1 (Theorem 4.3 of [11] (2-Key GCM-SIV)). *The GCM-SIV mode of operation is a nonce-misuse resistant authenticated encryption scheme. Furthermore, for every adversary $\mathcal{A}$ attacking the GCM-SIV construction, making q_E encryption queries and q_D decryption queries of maximum plaintext length m blocks with $m < 2^{32}$,[4] there exists an adversary $\mathcal{A}'$ making Q' queries to distinguish F from a random function, such that*

$$\mathsf{AdvEnc}_{\mathcal{A},\Pi,\mathsf{nmrAE}} < 2\cdot\mathsf{AdvPRF}^1_{\mathcal{A}',E} + \frac{(M+1)\cdot(q_E+q_D)^2}{2^{n-1}} + \frac{{q_E}^2}{2^{n-m-2}} \tag{8}$$

where $t(\mathcal{A}') \le 6\cdot t(\mathcal{A})$ and $Q' \le 2(q_E+q_D)+L$, the value L is the overall number of blocks encrypted or decrypted, and M is an upper bound on the length of all encryption and decryption queries including the length of the message plus the AAD.

Safety margins for using GCM-SIV, and the implied limit on the lifetime of a key. The dominant term in the bound will typically be $\mathsf{AdvPRF}^1_{\mathcal{A}',E}$. This is due to the fact that after encrypting or decrypting L blocks, the advantage of the adversary is $L^2/2^n$, Note also that the term ${q_E}^2/2^{n-m-2}$ represents the probability of their being a collision in a counter input to AES, which then would leak plaintext material. In order provide a recommendation on the maximal number of GCM-SIV encryptions (with the same key), it is useful to refer to NIST's guidelines [8] for AES-GCM with a random 96-bit IV (or any IV whose bitlength is not 96), which faces an analogous situation. The NIST requirement is that the probability of an IV collision should not exceed 2^{-32}, and this is translated in [8] to limiting the allowed number of encryptions with AES-GCM using a random IV to 2^{32}.

With the same rationale, due to the $L^2/2^n$ advantage, at most 2^{48} blocks overall can be encrypted or decrypted in order to remain within the recommended NIST bounds [8]. Furthermore, the $Q^2/2^{n-m-2}$ bound places a limit of at most 2^{39} messages of length 2^{16} blocks, or 2^{42} messages of length at most 2^{10} blocks.

Better bounds. We show how our key derivation method yields far better bounds on the use of GCM-SIV. This method has already been incorporated into the standard proposal for CFRG for GCM-SIV [13]; however, the concrete bounds analyzed in [12] for this proposal are significantly inferior to those that we provide here.

6.1 AES-GCM-SIV with Nonce-Misuse

Let $\mathcal{A}$ be a $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$-nonce adversary. Recall that $\vec{N}_E$ and $\vec{N}_D$ are vectors where N^i_E and N^i_D denote the number of times $\mathcal{A}$ queries its encryption and decryption oracle, respectively, with the ith nonce. In addition, $\vec{B} = (B_1, \ldots, B_Q)$ is such that B_i blocks overall are encrypted or decrypted using the ith nonce. For the AES-GCM-SIV scheme, this includes the message blocks and the one additional block used to mask the hash result. Finally, a and m are such that the longest AAD is less than 2^a blocks and the longest message is less than 2^m blocks. Since we are now considering the nonce-misuse case, $\vec{N}_E$ can be an arbitrary vector; this is unlike the nonce-respecting case where $N^i_E = 1$ for all i.

By Theorem 3.1, for every $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$-nonce adversary $\mathcal{A}$, there exists an $(O(t), Q)$-adversary $\mathcal{A}_1$ for F, and an $(O(t), Q, \vec{B}, \mu)$-adversary $\mathcal{A}_2$ for E with μ that depends on Π, such that

$$\begin{aligned}\mathsf{AdvEnc}_{\Pi',\mathcal{A},\mathsf{nmrAE}} &= \frac{1}{2}\cdot\mathsf{AdvKDF}_{\mathcal{A}_1,F} \\ &\quad + \frac{1}{2}\cdot\mathsf{AdvPRF}^Q_{\mathcal{A}_2,E} + \mathsf{AdvEncRF}^Q_{\Pi'',\mathcal{A},\mathsf{nmrAE}}.\end{aligned} \tag{9}$$

By what we have seen already, we can bound

$$\mathsf{AdvKDF} \le \mathsf{AdvPRP}_{AES}(t, 6Q) + \min\left\{\frac{36Q^2}{2^{129}}, \frac{6Q}{2^{96}}, 1\right\}.$$

Now, by the standard assumption on AES, for all reasonable values of t and Q, we can ignore the $\mathsf{AdvPRP}_{AES}(t, 6Q)$ portion. Furthermore, observe that $\frac{36Q^2}{2^{129}} < \frac{6Q}{2^{96}}$ if and only if $Q < 2^{33}/6$. Since we are interested in large values of Q (for small values of Q the bounds are very good in any case), we take the term $\frac{6Q}{2^{96}}$. Thus, we have

$$\mathsf{AdvKDF} \le \frac{6Q}{2^{96}}. \tag{10}$$

Regarding AdvPRF, by Assumption 1, we have

$$\mathsf{AdvPRF}^Q_{\mathcal{A}_2,E} \le \frac{Q^3}{6\cdot 2^{2\kappa}} + \frac{Q\cdot B\mathrm{max}^2}{2^n} + \frac{\mu\cdot T_E}{2^\kappa}.$$

where $B\mathrm{max} = \max\{B_i\}$. It thus remains to bound μ, which is an upper bound on the number of times that the same block is encrypted with AES under different keys. In the specification of GCM-SIV [11], the counter/nonce is generated by applying AES to the XOR of the tag T (which is the result of a universal hash on the message) and the nonce N. In this case, using key derivation, two keys K_1 and K_2 are derived. The key K_1 is used to encrypt the message, and the key K_2 is used to compute the authentication tag

[4] In actuality, GCM-SIV as described in [11] uses a fixed $m = 32$ that is not dependent on the lengths of the messages encrypted by the adversary. Thus, the dominant term is $Q^2/2^{94}$ even if only short messages are encrypted. This was changed in [12, Section 3] in the GCM-SIV$^+$ scheme by setting the derived IV to be of length $n - 1 = 127$ bits instead of $n - m = 96$ bits. Nevertheless, this modification is not related to the topic of this paper, and we therefore assume this as the baseline.

as $TAG = AES_{K_2}(T \oplus N)$. This tag TAG is also used to derive the initial counter. Thus, $\mu > 1$ only if there exist multiple different nonces (resulting in multiple different derived keys) and messages, resulting in the same TAG. (To be exact, $\mu > 1$ if the same counter is input in any position, and so if the tags are close then this may happen.) For simplicity, we bound the probability that the most significant 94 bits of the tag are the same, since if these are different then the input to AES in encryption is always different. However, this event is just the probability of a multicollision on different inputs. Using Theorem 2.1, the probability of a multicollision of 5 or more is upper bound by $\frac{Q^5}{120\cdot(2^{94})^4} \approx \frac{Q^5}{2^{383}}$. Thus, even for $Q = 2^{64}$, this probability is negligible at 2^{-63}. We can therefore take $\mu = 5$ as a very conservative bound. Thus, we can write

$$\mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E} \leq \frac{Q^3}{6\cdot 2^{2\kappa}} + \frac{Q\cdot B\mathsf{max}^2}{2^n} + \frac{5\cdot T_E}{2^\kappa}. \tag{11}$$

It remains to analyze $\mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},\mathsf{nmrAE}}$, which is the advantage of the adversary in the nonce-misuse resistant setting, when a random function is used to encrypt all messages. In this setting, for every different nonce, a random GHASH function is chosen,[5] as well as random functions for counter-based encryption. For the sake of our analysis, the following informal description of encryption—replacing AES with a truly random function—suffices (see [11] for a full specification):

(1) The universal hash function is applied to the additional authenticated data (AAD) and plaintext message; denote the output by T.
(2) The random function is applied to $T \oplus N$ (prefixed by a single zero), where N is the input nonce; denote the output by TAG.
(3) The initial counter is set to be TAG, prefixed by a single one, and CTR-encryption is used from the counter. (The prefixing with zero and one is used to ensure that the same input is never provided to the random function to derive TAG in the previous step, and to encrypt in this step.)

As in AES-GCM in Section C, the probability that an adversary receives back a non-$\perp$ result in a singe decryption query is at most $\frac{2^a+2^m}{2^n}$, since this is the maximum degree of the polynomial in the universal hash function. Since there are $\sum_{i=1}^{Q} N_D^i$ decryption queries overall, the probability that a decryption query returns $\perp$ is upper bound by $\frac{2^a+2^m}{2^n}\cdot\sum_{i=1}^{Q} N_D^i$.

Assume now that all decryption queries return $\perp$. We claim that in this case, the advantage of the adversary in the experiment is 0, *unless* the same counter value is input twice for the *same nonce* and for *different messages*. In order to see why, observe that if different nonces are always input for different messages, then all values are encrypted using truly random and independent one-time pads. In particular, even if the same counter value appears for different nonces, nothing is revealed since independent random functions are used to encrypt when there are different nonces. (Recall that if the same nonce is used with the same message, then the same result is returned, and this adds no advantage to the adversary in the nonce-misuse resistance setting.)

[5] In actuality, AES-GCM-SIV in [12] uses a universal hash function called $POLYVAL$. This is essentially the same as GHASH except that the order of the bytes in the 16-byte blocks is reversed to make it more efficiently computable, and the order of the bits inside the bytes is reversed to make it compatible with the standard definitions.

We thus need to bound the probability that the same nonce is used for different messages. Now, the probability that this occurs is simply the probability of a collision of the initial counter. Recall that the maximum-length of any encrypted message is $2^m - 1$ blocks. If the $n-1-m$-most significant bits of the initial counter differ, then different counters are always input to the random function (note that by the prefixing of the counter by 1, it is of length $n-1$ and not length n). Since N_E^i is the number of messages encrypted with a single nonce, such a collision occurs in the ith nonce with probability at most 0

$$\frac{\binom{N_E^i}{2}}{2^{n-1-m}} < \frac{(N_E^i)^2}{2^{n-m-2}}.$$

Applying a union bound and including the advantage from decryption queries, we have that this advantage is upper bound by

$$\mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},\mathsf{nmrAE}} \leq \frac{(2^a+2^m)\cdot\sum_{i=1}^{Q} N_D^i}{2^n} + \frac{\sum_{i=1}^{Q}(N_E^i)^2}{2^{n-m-2}}. \tag{12}$$

Combining Equations (9)–(12), and noting that $n = 128$ here, we have the following theorem:

Theorem 6.2 (AES-GCM-SIV bounds). *Let Π' be the AES-GCM-SIV scheme and assume that Assumption 1 holds. Then, for every $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$-nonce adversary $\mathcal{A}$, it holds that:*

$$\mathsf{AdvEnc}_{\Pi',\mathcal{A},\mathsf{nmrAE}} \leq \frac{3Q}{2^{96}} + \frac{Q^3}{3\cdot 2^{2\kappa}} + \frac{Q\cdot B\mathsf{max}^2}{2^{129}} + \frac{5T_E}{2^{\kappa+1}} + \frac{(2^a+2^m)\cdot\sum_{i=1}^{Q} N_D^i}{2^{128}} + \frac{\sum_{i=1}^{Q}(N_E^i)^2}{2^{126-m}} \tag{13}$$

where $B\mathsf{max} = \max\{B_i\}_{i=1}^{Q}$ and κ is the key length.

Simplified bounds. Consider a simplified setting where AAD is not used in the application (and thus AAD is not processed even in decryption queries). In such a case, $(2^a+2^m)\cdot\sum_{i=1}^{Q} N_D^i < \frac{Q\cdot B\mathsf{max}^2}{2}$ (since $B\mathsf{max}$ includes all use of the ith nonce in both encryption and decryption queries, and so the AAD must be huge for this inequality to not hold). Thus $\frac{Q\cdot B\mathsf{max}^2}{2^{n+1}}$ dominates the term $\frac{(2^a+2^m)\cdot\sum_{i=1}^{Q} N_D^i}{2^{128}}$. Next, observe that for $Q \leq 2^{64}$ and AES with key-size 128 or above, it holds that the $\frac{3Q}{2^{96}}$ term dominates $\frac{Q^3}{3\cdot 2^{2\kappa}}$ and so the latter can be removed. In addition, for all "reasonable" adversaries, we have that $\frac{5T_E}{2^{\kappa+1}}$ is very small, and so can be ignored.

Finally, in typical applications, we can assume that $(2^a+2^m)\cdot\sum_{i=1}^{Q} N_D^i < \frac{Q\cdot B\mathsf{max}^2}{2}$ and thus the term $\frac{(2^a+2^m)\cdot\sum_{i=1}^{Q} N_D^i}{2^{128}}$ can also be ignored. This gives us the following very simple bound.

Corollary 6.3. *Let Π' be the AES-GCM-SIV scheme, and assume that Assumption 1 holds, that $Q \leq 2^{64}$, and that $(2^a+2^m)\cdot\sum_{i=1}^{Q} N_D^i < \frac{Q\cdot B\mathsf{max}^2}{2}$. Then, for every $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, 0, m)$-nonce adversary $\mathcal{A}$:*

$$\mathsf{AdvEnc}_{\Pi',\mathcal{A},\mathsf{nmrAE}} \leq \frac{3Q}{2^{96}} + \frac{Q\cdot B\mathsf{max}^2}{2^{129}} + \frac{\sum_{i=1}^{Q}(N_E^i)^2}{2^{126-m}}.$$

As we will see below, in many cases this is dominated by $Q\cdot B\mathsf{max}^2/2^{129}$.

Interpreting the bounds. Before considering nonce-misuse resistance, consider the case of $Q = 2^{56}$ and $m = B\max = 2^{16}$ (in this case, each nonce is used once). As with AES-GCM, the original GCM-SIV is not secure since by Eq. (8), the advantage is $\frac{Q^2}{2^{n-m-2}}$. With the above numbers, the advantage is $\frac{2^{112}}{2^{110}} > 1$. In contrast, in Eq. (13), the dominant term is $\frac{Q \cdot B\max^2}{2^{n+1}}$, which for these parameters equals $\frac{2^{88}}{2^{129}} = 2^{-41}$. This difference is very significant.

We stress that a similar analysis appears in [12]. However, the analysis there concludes that if there exists an adversary $\mathcal{A}$ for AES-GCM-SIV then there exists an adversary $\mathcal{A}_2$ that distinguishes AES from a random function, where $\mathcal{A}_2$ *running time is* Q *times the running time of* $\mathcal{A}$. In contrast, in our result, $\mathcal{A}_2$'s running-time is in the same order as $\mathcal{A}$. For the parameters we are considering with $Q = 2^{48}$, the analysis of [12] is meaningless. In particular, let the running-time of $\mathcal{A}$ be $T = 2^{48}$. Then, we would have no contradiction, since an adversary running in time $Q \cdot T = 2^{96}$ can easily distinguish AES from a random function. Nevertheless, this is an artifact of the *analysis* in [12] and not the actual bounds, as we have shown here.

We now proceed to consider nonce misuse. In this case, we now need to consider the additional advantage from the term $\frac{\sum_{i=1}^{Q}(B_i)^2}{2^{n-m-2}}$; recall that B_i is the number of times that the same nonce is reused. It follows that the advantage is below the NIST recommendation of 2^{-32} as long as $\sum_{i=1}^{Q}(B_i)^2 < 2^{94-m}$ (plugging in $n = 128$). Concretely, if short messages are encrypted (say, $m \leq 2^{16}$), then this is achieved even for $Q = 2^{45}$ and $B_i = 2^{10}$, or for $Q = 2^{32}$ and $B_i = 2^{15}$. For longer messages of length 2^{30}, this is achieved for example with $Q = 2^{25}$ and $B_i = 2^6$. In addition to the above, we need to consider the term $\frac{Q \cdot B\max^2}{2^{n+1}}$, where $B\max$ is the maximum numbers of blocks encrypted with a single nonce. For the values of Q, B_i and m above, we have that $B\max = B_i \cdot m$ (since we are considering that all messages and nonce misuse is of the same maximal length). Thus, for $m = 2^{16}$, $Q = 2^{45}$, and $B_i = 2^{10}$ we have that the probability is $\frac{2^{45} \cdot (2^{10+16})^2}{2^{129}} = 2^{-32}$. Likewise, for $m = 2^{16}$, $Q = 2^{32}$ and $B_i = 2^{15}$, we have that the probability is $\frac{2^{32} \cdot (2^{15+16})^2}{2^{129}} = 2^{-35}$. For the setting of longer messages with the parameters above, this term is also below 2^{-32}, as required. By inspection, one can verify that all other terms are dominated by the above two.

Encryption limits. In this case, the dominant terms for large Q are $\frac{3Q}{2^{96}}$, $\frac{Q \cdot B\max^2}{2^{n+1}}$ and $\frac{\sum_{i=1}^{Q}(B_i)^2}{2^{n-m-2}}$. As long as nonce-misuse is not too high, we have that $\frac{\sum_{i=1}^{Q}(B_i)^2}{2^{n-m-2}}$ is small. Thus, we once again obtain that one can encrypt almost 2^{64} messages with the same nonce being used for up to $B\max = 2^{16}$ blocks, or 2^{48} messages with the same nonce being used for up to 2^{24} blocks, or 2^{32} messages with the same nonce being used for up to $B\max = 2^{32}$ blocks, and still remain within the NIST-recommended limit of 2^{-32} advantage. Observe that when nonce misuse is low, it is possible to support the encryption of much longer messages, because the $B\max^2$ factor is the overall number of blocks with each nonce (and this factor is quadratic).

Nonce-misuse vs no-nonce. The advantage of deploying nonce-misuse resistant schemes is that such schemes do not break when nonces repeat. However, some have interpreted such schemes as being secure when no nonce (or a fixed nonce) is used. Of course, in such a case, repeating plaintexts are detected. Nevertheless, in some applications, messages are guaranteed to not repeat, in which case it would seem safe to always use the same fixed nonce.

Our analysis shows that although the above has some truth to it, repeatedly using the same nonce can impact the security bounds. For example, consider the case of 2^{32} messages of length 2^{32}, all encrypted using the same nonce. In this case, $Q = 1$ and $B\max = 2^{64}$, and thus the term $\frac{Q \cdot B\max^2}{2^{129}} = \frac{1}{2}$ and so there is no security guarantee. This must be the case since encrypting 2^{64} blocks with the same key can never be secure (by the birthday paradox). Thus, when encrypting massive amounts of data, one should not purposefully reuse the same nonce all the time.

6.2 AES-GCM-SIV with Random IVs

In order to derive the advantage in the case of a random IV, the only difference is due to the probability that nonces repeat. Even given the case of a poor source of entropy, the probability that a randomly-chosen IV will repeat very many times is extremely low. In this case, we apply Theorem 3.2, and obtain a *fundamentally different* result from CTR and AES-GCM with a random IV. In particular, AES-GCM-SIV can be used with a random IV to encrypt essentially the same number of blocks as when used with (hopefully unique) nonces. Specifically, Theorem 3.2 yields a bound that is the same as for the case of a nonce-adversary, with the addition of $\frac{{N_E}^4}{24 \cdot 2^{3\ell}}$. Considering the case of a 96-bit IV, we have that this equals $\frac{{N_E}^4}{24 \cdot 2^{288}}$. This value is smaller than $\frac{3Q}{2^{96}}$ for any $N_E \leq 2^{64}$, and so can be ignored. Thus, whereas AES-GCM with a random IV is limited to just 2^{32} encryptions, with AES-GCM-SIV it is even possible to encrypt $Q = 2^{64}$ messages of length 2^{12} each, or $Q = 2^{48}$ messages of length 2^{20} each; see Table 2 below. Before providing the bound, we note that since $N_E^i \leq 3$ for every i in this case, we can also ignore the $\frac{\sum_{i=1}^{Q}(N_E^i)^2}{2^{126-m}}$ term in Theorem 6.2 and Corollary 6.3. We provide the simpler version of the bound, as in Corollary **??**, and conclude:

Theorem 6.4. *Let* Π' *be the AES-GCM-SIV scheme, and assume that Assumption 1 holds, that* $Q \leq 2^{64}$*, and that* $(2^a + 2^m) \cdot \sum_{i=1}^{Q} N_D^i < \frac{Q \cdot B\max^2}{2}$*. Then, for every* $(t, Q, N_E, \vec{N}_D, \vec{B}, 0, m)$*-IV adversary* $\mathcal{A}$*:*

$$\mathsf{AdvEnc}_{\Pi', \mathcal{A}, \mathsf{nmrAE}} \leq \frac{3Q}{2^{96}} + \frac{Q \cdot B\max^2}{2^{129}}.$$

6.3 Summary Parameters

In Table 2, we show what parameters can be used for AES-GCM-SIV, within the NIST error bound of 2^{-32}.

Observe that in almost all cases, the term $\frac{Q \cdot B\max^2}{2^{n+1}}$ dominates; the only exception is for extremely short messages. It is also worthwhile noting that while a bad event that happens with probability $\frac{Q \cdot B\max^2}{2^{n+1}}$ enables distinguishing the encryption game from the random game, it is not a catastrophic event as in the case of a bad event that occurs with probability $\sum_{i=1}^{Q} \frac{(N_E^i)^2}{2^{n-m-2}}$. This is because the latter

Scheme	Q	N_E^i	2^m	$\frac{Q \cdot Bmax^2}{2^{129}}$	$\sum_{i=1}^{Q} \frac{(N_E^i)^2}{2^{126-m}}$
AES-GCM-SIV (nonce)	2^{45}	2^{10}	2^{16}	2^{-32}	2^{-45}
	2^{32}	2^{15}	2^{16}	2^{-35}	2^{-48}
	2^{25}	2^{6}	2^{30}	2^{-32}	2^{-59}
	2^{32}	1	2^{30}	2^{-35}	2^{-62}
	1	2^{31}	2^{16}	2^{-34}	2^{-47}
	2^{42}	2^{8}	2^{16}	2^{-39}	2^{-52}
	2^{64}	2^{10}	2^{3}	2^{-39}	2^{-39}
	2^{64}	2^{15}	1	2^{-33}	2^{-31}
	2^{64}	2^{8}	2^{8}	2^{-33}	2^{-38}
	2^{48}	2^{10}	2^{14}	2^{-33}	2^{-44}
	2^{48}	2^{8}	2^{16}	2^{-33}	2^{-46}
	2^{64}	2^{10}	2^{10}	2^{-25}	2^{-32}
	2^{48}	2^{10}	2^{16}	2^{-29}	2^{-42}
	2^{32}	2^{10}	2^{24}	2^{-29}	2^{-50}
AES-GCM-SIV (random IV)	2^{64}	-	2^{12}	2^{-35}	-
	2^{48}	-	2^{20}	2^{-35}	-

Table 2: Example parameters and security bounds for dominant terms (exponent rounded to nearest integer). Recall that Q is the number of difference nonces in encryption and decryption queries, N_E^i is the number of messages encrypted per nonce (we assume all are equal), and $2^m - 1$ is the maximum message length. Observe that $Bmax = (N_E^i + N_D^i) \cdot 2^m$ when all messages are of maximum length. Bounds that are below acceptable are colored in red.

bad event reveals complete plaintext and breaks the authenticator for all future messages.

Note that for short messages, AES-GCM-SIV provides very impressive bounds. For example, in the QUIC protocol, the messages are very short, at approximately 100 bytes which translates to 8 blocks. QUIC uses a random IV and so it is possible to encrypt 2^{64} messages safely, even if they are much longer (2^{12} blocks). When considering the nonce setting and nonce misuse, then even if nonces are misused up to 1,024 times, it is well within the security bounds to encrypt even 2^{64} messages. Thus, if QUIC is used in a setting with a poor pseudorandom generator, it is still possible to encrypt at extremely high volumes. We remark that in QUIC there is no AAD at all. Thus, the assumption in Theorem 6.4 holds trivially.

7 PERFORMANCE

This section discusses some performance results, measured on the Skylake processor (with Hyperthreading and Turbo disabled), as shown in Table 3. The results cover short ,medium and long messages for Π= CTR, AES-GCM and GCM-SIV (encryption) and their respective Π′ variant, using DeriveKey. Note that GCM-SIV+ with DeriveKey, is called AES-GCM-SIV (see [12]). In all of these cases, the key is of 128 bits. Obviously the relative overhead incurred by DeriveKey (see Section 4) is higher for (very) short messages, as illustrated messages of 64 bytes (4 blocks). This is unavoidable. For CTR, where there are no other computations except encryption, the relative overhead is the highest, and for the other modes it is reduced is by the other operations. However, already for medium messages, and certainly for long ones, the relative overhead decreases monotinically, down to a few percents. Derivation of 256-bit keys has a similar performance effect.

Given the objectively fast rates obtained, we argue that in most applications, the gain in security margins is worth the additional overhead. This is especially true for applications that encrypt masses of data because **(a)** in this case the percentage overhead is very small, and **(b)** the better security margins are of importance.

The AES-GCM code that we measured is the OpenSSL (1.0.2k) implementation, using the OpenSSL speed utility;[6] the results were converted to cycles and cycles per-byte (C/B). Note that this utility does not include the Init step. As a result (due to the structure of the OpenSSL code), this also means that the encryption of the mask first counter block (`10000000000000000000000000000000`) is not measured.[7] Therefore, to make a consistent comparison, the OpenSSL results needed to be adjusted by adding the cost of one encryption. We used a very generous estimation as follows: for the 128 bit case, we added 45 cycles, and for the 256 bit case, we added 60 cycles. These adjustments have negligible impact for long messages, but are noticeable for short ones.

Message length (bytes)	Original (C/B)	Incl. Derivation (C/B)	Relative overhead
	CTR	**w/ DeriveKey**	
64	0.69	2.04	2.96
1,024	0.63	0.71	1.13
4,096	0.63	0.65	1.03
8,192	0.63	0.64	1.02
	AES-GCM	**w/ DeriveKey**	
64	2.96	3.95	1.34
1,024	0.84	0.90	1.07
4,096	0.68	0.70	1.02
8,192	0.66	0.67	1.01
	GCM-SIV	**w/ DeriveKey**	
64	4.53	5.67	1.25
1,024	1.25	1.37	1.10
4,096	1.01	1.04	1.03
8,192	0.97	0.98	1.01

Table 3: Performance (throughput in *cycles per byte* on a Skylake processor) of CTR, AES-GCM, and AES-GCM-SIV (128-bit key) with and without DeriveKey, for short, medium and long messages. The table shows (rightmost column) the relative overhead due to the derivation. See explanation and discussion in the text.

ACKNOWLEDGMENTS

We thank Adam Langley for many helpful discussions regarding AES-GCM-SIV and the key derivation technique, and we thank Tetsu Iwata and Yannick Seurin for pointing out some typos and small errors in an earlier manuscript.

[6] For example, `openssl speed -evp aes-128-gcm`, and `openssl speed -decrypt -evp aes-256-gcm`.

[7] Technically, the speed utility measures AES-GCM with a fixed key and repeating nonces, which does not really represent a legitimate usage of the cipher, rather a performance characteristic.

REFERENCES

[1] BoringSSL, https://boringssl.googlesource.com/boringssl/
[2] RFC5077: Transport Layer Security (TLS) Session Resumption without Server-Side State, https://tools.ietf.org/html/rfc5077#section-4
[3] A. Abdalla and M. Bellare. Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques. In *ASIACRYPT 2000*, Springer (LNCS 1976), pages 546–559, 2000.
[4] E. Barker and J. Kelsey. Recommendation for Random Number Generation Using Deterministic Random Bit Generators, NIST Special Publication 800-90A. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
[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] E. Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Information Processing Letters*, 84(3):117–124, 2002.
[7] H. Bock, A. Zauner, S. Devlin, J. Somorovsky and P. Jovanovic. Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS. In the *10th USENIX Workshop on Offensive Technologies* (WOOT 16), 2016.
[8] M. Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication. *Federal Information Processing Standard Publication* FIPS 800-38D, 2006. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
[9] S. Gilboa and S. Gueron. How many queries are needed to distinguish a truncated random permutation from a random function?, Journal of Cryptology (2017). doi:10.1007/s00145-017-9253-0
[10] S. Gilboa and S. Gueron. The Advantage of Truncated Permutations. Manuscript, 2016. https://arxiv.org/abs/1610.02518.
[11] S. Gueron, Y. Lindell. GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. *22nd ACM CCS*, pages 109–119, 2015.
[12] S. Gueron, A. Langley, Y. Lindell. AES-GCM-SIV: Specification and Analysis, Cryptology ePrint Archive, Report 2 017/168, 2017. http://eprint.iacr.org/2017/168.
[13] S. Gueron, A. Langley, Y. Lindell. https://tools.ietf.org/html/draft-irtf-cfrg-gcmsiv
[14] S. Gueron, Y. Lindell, A. Nof and B. Pinkas. Fast Garbling of Circuits Under Standard Assumptions. *22nd ACM CCS*, pages 567–578, 2015.
[15] D.A. McGrew and J. Viega The Galois/Counter Mode of Operation (GCM). http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf
[16] D.A. McGrew and J. Viega The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In *INDOCRYPT 2004*, Springer (LNCS 3348), pages 343–355, 2004.
[17] N. Mouha, A. Luykx. Multi-key Security: The Even-Mansour Construction Revisited. Advances in Cryptology – CRYPTO 2015, Proceedings Part I, pp. 209–223 (2015).
[18] QUIC, a multiplexed stream transport over UDP. https://www.chromium.org/quic.
[19] P. Rogaway and T. Shrimpton. Deterministic Authenticated Encryption: A Provable-Security Treatment of the Key-Wrap Problem. In *EUROCRYPT 2006*, Springer (LNCS 4004), pages 373–390, 2006.
[20] A. J. Stam, Distance between sampling with and without replacement, Statist. Neerlandica **32** (1978), no. 2, 81–91.
[21] 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.

A PROOF OF THEOREM 2.2

THEOREM A.1 (THEOREM 2.2 – RESTATED). *Let E be an ideal cipher with key of length k and domain $\{0,1\}^n$. Then, for every $(t, Q, \vec{B}, \mu)$-adversary making at most T_E queries to E with adversarially chosen keys, it holds that*

$$\mathsf{AdvPRF}^{Q}_{\mathcal{A},E} \leq \min\left\{\frac{Q^3}{6\cdot 2^{2\kappa}} + \frac{Q\cdot B\mathrm{max}^2}{2^n},\ \frac{Q^2}{2^{\kappa+1}} + \frac{\sum_{i=1}^{Q}(B_i)^2}{2^{n+1}}\right\} + \frac{\mu\cdot T_E}{2^\kappa} \tag{14}$$

where $B\mathrm{max} = \max\{B_1,\ldots,B_Q\}$.

PROOF. We use the terminology and setup of [17]. The proof extends the considerations in [17, Theorem 2]. Let $\mathcal{A}$ be an adversary in $\mathrm{ExptPRF}^{Q}$, who makes queries to the oracle $O(i,x)$ of the experiment, as well as queries to the ideal cipher E. Note that the queries to the ideal cipher are of the form (k,x), and the value returned is $E_k(x)$.

Let Λ be the event that either:

(1) There is a 3-collision in the keys $k_1,\ldots,k_Q$ chosen in $\mathrm{ExptPRF}^{Q}$; i.e., there are at least three indexes $i,j,\ell \in [Q]$ for which $k_i = k_j = k_\ell$, or
(2) The adversary made a query to E (or its inverse) with a chosen key $\bar{k}$, such that $\bar{k}$ is one of $k_1,\ldots,k_Q$.

We first claim that the probability that Λ happens is bounded from above by

$$\mathrm{Prob}[\Lambda] \leq \frac{Q^3}{6\cdot 2^{2\kappa}} + \frac{\mu\cdot T_E}{2^\kappa}. \tag{15}$$

This follows from the fact that by Theorem 2.1, the probability that there is a 3-collision is at most $\frac{Q^3}{6\cdot 2^{2\kappa}}$. Furthermore, as shown in [17], the probability that the adversary makes a query as in condition (2) is at most $\frac{\mu\cdot T_E}{2^\kappa}$.

Assuming now that Λ does not happen. In this case, we can divide the list $k_1,\ldots,k_Q$ to two categories: s_1 keys that appear only once, and s_2 (distinct) pairs of colliding keys that appear twice; observe that $s_1 + 2s_2 = Q$ (since Λ does not occur, there are no 3-collisions, so these are all possibly categories).

Each pair $k_i, k_j = k'$ in the second category can be viewed as a single key k' which was used for *at most* $B_i + B_j \leq 2B\mathrm{max}$ distinct queries. Accordingly, there are, effectively, $Q' = s_1 + s_2 \leq Q$ keys.

Consequently, given that Λ does not happen, the queries to E_K (with Q' keys) plus the chosen key queries to the ideal cipher E, are identical to queries to Q' random permutations plus the ideal cipher queries. It therefore follows that

$$\mathsf{AdvPRP}^{Q}_{\mathcal{A},E} = \mathrm{Prob}[\Lambda].$$

Accounting for the standard advantage for distinguishing a random function from a random permutation, which is $\frac{\ell^2}{2^{n+1}}$ for ℓ oracle queries to a single key, we obtain

$$\mathsf{AdvPRF}^{Q}_{\mathcal{A},E} \leq \mathrm{Prob}(\Lambda) + s_1\cdot\frac{B\mathrm{max}^2}{2^{n+1}} + s_2\cdot\frac{(2B\mathrm{max})^2}{2^{n+1}}.$$

This follows because the maximum number of queries to the instances of keys that appear only once is $B\mathrm{max}$, and the maximum number to the queries appearing twice is $2B\mathrm{max}$. Eq. (17) is maximized (over all possible choices of s_1, s_2) when $s_1 = 0$ and $s_2 = Q/2$ (although this event is highly unlikely). Thus,

$$\begin{aligned}\mathsf{AdvPRP}^{Q}_{\mathcal{A},E} &\leq \mathrm{Prob}(\Lambda) + \frac{Q}{2}\cdot\frac{(2B\mathrm{max})^2}{2^{n+1}}\\ &= \mathrm{Prob}(\Lambda) + \frac{Q\cdot B\mathrm{max}^2}{2^n} \qquad (16)\\ &\leq \frac{Q^3}{6\cdot 2^{2\kappa}} + \frac{\mu\cdot T_E}{2^\kappa} + \frac{Q\cdot B\mathrm{max}^2}{2^n}. \qquad (17)\end{aligned}$$

Finally, observe that the event where there are no key collisions (i.e., $s_1 = Q, s_2 = 0$) occurs with probability $\frac{Q^2}{2^{\kappa+1}}$. In this case, the distinguishing probability between a random function and random permutation is $\frac{(B_i)^2}{2^{n+1}}$, in the ith instance. Using the same arguments as above, and a union bound over the difference between a random

function and permutation for each instance, we have that

$$\mathsf{AdvPRF}^{Q}_{\mathcal{A},E} \leq \frac{Q^2}{2^{\kappa+1}} + \frac{\sum_{i=1}^{Q}(B_i)^2}{2^{n+1}} + \frac{\mu \cdot T_E}{2^{\kappa}}. \quad (18)$$

We can choose the minimum between the bounds in Eq. (17) and Eq. (18), and this completes the proof. □

Remark 3. *Observe that in the case that all B_i's are approximately the same, the bound of Eq. (17) is almost certainly smaller than the bound in Eq. (18). However, if B_i is very small for most values of i, and only larger for a few, then the bound of Eq. (18) may be significantly smaller. Thus, we include both bounds in the theorem statement.*

B DEFINING ENCRYPTION SECURITY IN OUR MODEL

In order to show how standard notions of security are formulated this way, we give some examples of oracles:

- *Eavesdropping adversary for a single message (*EAV*):* Oracle O receives (b, k, m_0, m_1) and outputs $\mathsf{Enc}_k(m_b)$ if $|m_0| = |m_1|$, and $\perp$ otherwise. After being called once, O halts and answers no more queries. (Note that although O receives four input values, $\mathcal{A}$ provides only m_0, m_1.)
- *LR-oracle security (*CPA*):* O is exactly the same as in the previous item, except that it does not halt and answers an unlimited number of queries. There are two versions of this: nCPA for nonce based encryption, and ivCPA for random-IV based encryption.
- *CCA security (*CCA*):* Define O to carry out both encryption and decryption. Formally, define $O(b, k, \mathsf{Enc}, m_0, m_1) = \mathsf{Enc}_k(m_b)$ if $|m_0| = |m_1|$, and $\perp$ otherwise. Furthermore, define the decryption oracle $O(b, k, \mathsf{Dec}, c, \lambda) = \mathsf{Dec}_k(c)$ if c was not returned in a previous call to O, and $\perp$ otherwise.
- *Nonce-respecting authenticated encryption (*nAE*):* First denote by $\mathsf{Enc}_k(IV, a, m)$ an encryption of additional authentication data a and message m using nonce IV. Then, for $b = 0$, define the oracle $O(0, k, \mathsf{Enc}, IV, a, m) = \mathsf{Enc}_k(IV, a, m)$ if IV has not been used in a previous encryption query, and $\perp$ otherwise. Furthermore, define $O(0, k, \mathsf{Dec}, c, \lambda, \lambda) = \mathsf{Dec}_k(c)$ for c that was not previously returned by an encryption query, and $\perp$ otherwise. For $b = 1$, define $O(1, k, \mathsf{Enc}, IV, a, m)$ by first computing $c = \mathsf{Enc}_k(IV, a, m)$; if $c = \perp$ then output $\perp$, else output a random string of length $|c|$. Furthermore, define $O(1, k, \mathsf{Dec}, c, \lambda, \lambda)$ to always return $\perp$.
- *Nonce-misuse resistant authenticated encryption (*nmrAE*):* This is *identical* to the previous formulation, with the exception that Enc is a deterministic function of (IV, a, m), and thus if the same (IV, a, m) is queried to Enc when $b = 1$ then the same random string is returned as in the last query of (IV, a, m). This expresses the fact that the only thing revealed in such a case is that the same value was encrypted.
- *IV-based formulations:* We will also consider authenticated encryption and misuse-resistant authenticated encryption with random IVs. This is the same as for nonce-based encryption as above, except that the adversary does not choose the nonce but the IV is randomly chosen. For authenticated encryption, denoted ivAE, if the same IV is chosen by the oracle in two different encryption queries, then the bit b is returned to the adversary (signifying that the adversary "won"). For misuse-resistant authenticated encryption, denoted ivmrAE, the only difference is that for $b = 1$ if the same IV is chosen for the same (a, m) from a previous query, then the same random string is returned.

As can be seen, all standard definitions can be formulated in this way. This formulation is slightly cumbersome since a single oracle is defined instead of separate ones for the case of $b = 0$ and $b = 1$. However, this formulation allows us to prove a single theorem that can be applied to all such definitions. For sake of exposition, we will refer to queries to the encryption and decryption oracles, with the understanding that this refers to queries to O that contain parameter Enc or Dec, respectively. We denote the oracles by nCPA, ivCPA, nAE, ivAE, nmrAE, ivmrAE for the appropriate level of security.

For the sake of our analysis, we need to separately consider adversaries for the nonce and IV settings. This is due to the fact that we need to specify the level at which nonces repeat. Therefore, we will say that $\mathcal{A}$ is a $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$-nonce adversary, with $\vec{N}_E = (N_E^1, \ldots, N_E^Q)$, $\vec{N}_D = (N_D^1, \ldots, N_D^Q)$, and $\vec{B} = (B_1, \ldots, B_Q)$, if:

- $\mathcal{A}$ runs in at most t steps,
- $\mathcal{A}$ queries its encryption or decryption oracle with Q different nonces overall
- $\mathcal{A}$ queries the ith nonce for N_E^i encryption queries and N_D^i decryption queries
- The number of blocks encrypted with the ith nonce in encryption and decryption queries is B_i
- The longest additional authentication data (AAD) in an encryption or decryption query is less than 2^a blocks
- The longest message in an encryption or decryption query (not including AAD) is less than 2^m blocks.

For the random-IV setting, we will say that $\mathcal{A}$ is a $(t, Q, N_E, \vec{N}_D, \vec{B}, a, m)$-IV adversary if it as above, with the difference that encryption queries are random and so the number of different nonces is not specified; rather N_E queries overall are made. Note that in this case, Q is the number of different nonces queried by $\mathcal{A}$ in calls to the *decryption oracle*.

Finally, for $O \in \{\mathsf{nCPA}, \mathsf{ivCPA}, \mathsf{nAE}, \mathsf{ivAE}, \mathsf{nmrAE}, \mathsf{ivmrAE}\}$, we define the advantage of the adversary in game ExptEnc to be

$$\mathsf{AdvEnc}_{\mathcal{A},\Pi,O} = 2 \cdot \mathrm{Prob}\left[\mathsf{ExptEnc}_{\mathcal{A},\Pi,O} = 1\right] - 1.$$

We say that a $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$-nonce adversary is nonce-respecting if for all i it holds that $N_E^i = 1$, and we say that it is nonce-disrespecting if there exists an i for which $N_E^i > 1$.

C AES-GCM WITH UNIQUE NONCES

In this section, we consider the concrete case of AES-GCM with a standard 96-bit IV. Here, the encrypted messages have length at most $2^{32} - 2$ blocks (with each nonce). We consider a nonce-respecting adversary here, and thus assume that all the nonces are unique. We consider a $(t, Q, \vec{N}_E, \vec{N}_D, \vec{B}, a, m)$-adversary $\mathcal{A}$. The analysis is very similar to that of the CTR case, because AES-GCM encryption is based on CTR mode. The difference is that the GHASH authenticator, which is an almost XOR universal family of hash functions, is applied to the ciphertext, and one additional encryption

per message is used, to encrypt (mask) the GHASH value. Thus, due to its polynomial construction, the probability of finding a successful forgery in a single decryption attempt is upper bounded by $\frac{2^a+2^m}{2^n}$, where 2^a is the maximum length of the AAD and 2^m-1 is the maximum message length. Since the number of decryption queries equals $\sum_{i=1}^{Q} N_D^i$, a union bound yields that the forgery probability is upper bounded by $\frac{(\sum_{i=1}^{Q} N_D^i)\cdot(2^a+2^m)}{2^n}$.

Basic AES-GCM. The overall number of blocks encrypted with E is $\sum_{i=1}^{Q}(B_i+1) = Q + \sum_{i=1}^{Q} B_i$, and these are all encrypted under a single key. The distinguishing probability in this case is upper bounded by $\frac{\left(\sum_{i=1}^{Q}(B_i+1)\right)^2}{2^{n+1}} + \frac{(\sum_{i=1}^{Q} N_D^i)\cdot(2^a+2^m)}{2^n}$ (as described above, the second term reflects the probability that GHASH values collide). Concretely, for a 128-bit block cipher like AES, when $Q + \sum_{i=1}^{Q} B_i$ reaches 2^{48}, the security level is only (roughly) 2^{-32}. Furthermore, if 2^{64} blocks are encrypted, then security is broken with very high probability; this could happen if 2^{48} plaintext each of length 2^{16} were encrypted. Observe that this is essentially the same as CTR mode, since the additional advantage of $\frac{(\sum_{i=1}^{Q} N_D^i)\cdot(2^a+2^m)}{2^n}$ gained by the decryption queries is typically dominated by the other term.

AES-GCM with key derivation. We apply Theorem 3.1 and thus it remains to analyze $\mathcal{A}$'s advantage when interacting in an encryption experiment where E is replaced with a truly random *function*. Note that in the AES-GCM standard, the GHASH key is generated by applying E to the block of all-zeroes, and thus here the key is the output of the truly random function on 0^{128}. Since the GCM specification prevents inputting the block of all zeroes at any other time, this implies that the GHASH key is random.

Consider the nonce-respecting authenticated-encryption experiment described in Section 2.4). In this experiment, the adversary receives an oracle $O = \mathsf{nAE}$ that either decrypts and encrypts according to the scheme, or that outputs a random string for encryption and returns $\perp$ for every decryption (decryption requests of ciphertexts that were obtained from previous encryption queries are decrypted correctly). We need to bound $\mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},\mathsf{nAE}}$. In this experiment, all the counters are encrypted using a truly random function. Since all the counters are unique, the advantage of the adversary due to encryption queries is 0 (all strings output by the oracle in both cases are random). Thus, the only advantage is due to a decryption query that may return something other than $\perp$ from a decryption query (in such a case, the adversary will know that it received a real decryption oracle and so can distinguish). Thus, from the above, we have $\mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},O} = \frac{(\sum_{i=1}^{Q} N_D^i)\cdot(2^a+2^m)}{2^n}$ for all parameter settings. By Theorem 3.1, we conclude that there exists an $(O(t),Q)$-adversary $\mathcal{A}_1$ for F and an $(O(t),Q,\vec{B},\mu)$-adversary $\mathcal{A}_2$ for E, such that

$$\begin{aligned}\mathsf{AdvEnc}_{\Pi',\mathcal{A},\mathsf{nAE}} &= \frac{1}{2}\cdot\mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{1}{2}\cdot\mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E} \\ &\quad + \mathsf{AdvEncRF}^{Q}_{\Pi'',\mathcal{A},\mathsf{nAE}} \\ &= \frac{1}{2}\cdot\mathsf{AdvKDF}_{\mathcal{A}_1,F} + \frac{1}{2}\cdot\mathsf{AdvPRF}^{Q}_{\mathcal{A}_2,E} \\ &\quad + \frac{Q\cdot(B\mathsf{max}+1)}{2^n}.\end{aligned}$$

As in the case of counter mode, since all nonces are unique, we have that $\mu = 1$. This is exactly the same as in CTR mode, with the addition of $\frac{(\sum_{i=1}^{Q} N_D^i)\cdot(2^a+2^m)}{2^n}$. We can therefore use the analysis in the bound in Eq. (7) and obtain:

$$\begin{aligned}&\mathsf{AdvEnc}_{\Pi',\mathcal{A},\mathsf{nAE}} \\ &\leq \frac{1}{2}\cdot\left(\mathsf{AdvPRP}^{6Q}_{\mathcal{A}',AES} + \min\left\{\frac{36Q^2}{2^{129}}, \frac{6Q}{2^{96}}, 1\right\} + \frac{Q^3}{6\cdot 2^{2\kappa}}\right. \\ &\qquad \left. + \frac{Q\cdot B\mathsf{max}^2}{2^n} + \frac{T_E}{2^\kappa}\right) + \frac{(\sum_{i=1}^{Q} N_D^i)\cdot(2^a+2^m)}{2^n}.\end{aligned}$$

Consider now the case of $\kappa = n = 128$ (as in AES-128), and $Q = 2^{48}$ and $B\mathsf{max} = 2^{16}$, as above, meaning that at most 2^{16} blocks are encrypted per nonce. Then, for reasonable values of T_E, we have that the dominant term in the advantage is $\frac{Q\cdot B\mathsf{max}^2}{2^n}$. For these parameters, we have that it equals $\frac{2^{48}\cdot 2^{32}}{2^{128}} = 2^{-48}$. This term is small even for large Q, and for $Q = 2^{48}$ we get $6\cdot 2^{-48}$, which does not change the security margin (in any significant way). Consquently, unlike basic AES-GCM mode, this is well within the range of being secure.

Encryption limits. The additional term of $\frac{(\sum_{i=1}^{Q} N_D^i)\cdot(2^a+2^m)}{2^n}$ that has been added here over CTR-mode encryption is insignificant. Thus, we obtain the same encryption limits and can encrypt $Q = 2^{64}$ messages of length $B\mathsf{max} = 2^{16}$, $Q = 2^{48}$ messages of length 2^{24}, or $Q = 2^{32}$ messages of length 2^{32}, and still remain within the NIST-recommended limit of 2^{-32} advantage.