Birthday Paradox Calculations and Approximation

Joshua E. Hill
InfoGard Laboratories

2015-March-23 *v1.00*

1 Birthday Problem

In the *birthday problem*, we have a group of n randomly selected people. If we assume that birthdays are uniformly distributed among the 365 days of the year[1] (which we'll later call "buckets"), then what is the chance that any people in the group share a birthday with each other?

Directly calculating this probability is obnoxious, but we can calculate the probability of this event using the probability of the complement of this event; namely, we calculate the probability of no two people sharing the same birthday (which is the complement of the situation we care about) and subtract this probability from 1.[2] This (under the appropriate convention for the binomial coefficient) is

$$1 - \prod_{k=0}^{n-1} \frac{365-k}{365} = 1 - \frac{\binom{365}{n} \cdot n!}{365^n}.$$

It's fairly clear that if you have 366 people in the room, then (by the pigeonhole principle) you are guaranteed to have at least one shared birthday (and, indeed, the above formula captures this behavior!) The surprising thing is that the likelihood grows quite quickly: for $n = 47$, the probability of a shared birthday is slightly greater than 95%, and $n = 23$ is the first to make a shared birthday more likely than not. Figure 1 shows how quickly this probability grows.

2 The Birthday Paradox in Cryptography

In cryptography, we are often interested in the instance where some random process (which is either a deterministic process that we are modeling as random, or a literal random process) happens to produce the same output as some prior output, which in this setting is called a *collision*. This is another application of the classic *birthday paradox*, but generally with more possible "buckets".

[1] We'll ignore the vulgarities of the actual calendar such as leap years, leap seconds, various conventions for the calendar, etc. We further assume that cows are spherical, and of uniform density.

[2] Making use of the laws of noncontradiction and excluded middle.

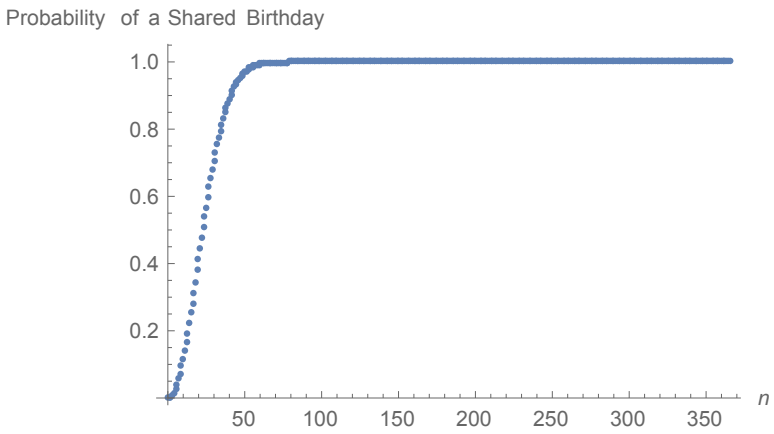

Figure 1: Probability of a Shared Birthday

If we consider the case where there are m buckets and n outputs, we can slightly abstract the above calculation and find that the probability of at least one collision is[3]

$$\begin{aligned}\Pr(\text{collision}) &= 1 - \prod_{k=0}^{n-1} \frac{m-k}{m} \\ &= 1 - \prod_{k=0}^{n-1} \left(1 - \frac{k}{m}\right). \end{aligned} \tag{1}$$

If we need to actually do calculations, then large quantities for n or m make the above calculation impractical, but this gives us a hint as to how to proceed.

3 Approximation

We develop a standard approximation for the above by first remembering that

$$e^{-x} = \sum_{j\geq 0} \frac{(-1)^j x^j}{j!} = 1 - x + \sum_{j\geq 2} \frac{(-1)^j x^j}{j!}. \tag{2}$$

For ease of notation, we'll denote

$$\varepsilon_x = \sum_{j\geq 2} \frac{(-1)^j x^j}{j!},$$

which (by the above) tells us that $\varepsilon_x = e^{-x} + x - 1$.

Some calculus shows us that ε_x is non-negative, and is an increasing function when $x > 0$.

One way of interpreting Equation (2) is that

$$1 - x = e^{-x} - \varepsilon_x, \tag{3}$$

[3]The analogous closed form $1 - \frac{\binom{m}{n}\cdot n!}{m^n}$ is also true, but it is generally impractical to make calculations based on this form of the equation.

and so for values of x very close to 0 we see that $\varepsilon_x \approx 0$, so we can approximate $1 - x \approx e^{-x}$. If we apply this approximation to Equation (1), we arrive at

$$\begin{aligned}
\Pr(\text{collision}) &\approx 1 - \prod_{k=0}^{n-1} e^{-\frac{k}{m}} \\
&= 1 - \exp\left(-\frac{0 + 1 + \cdots + n - 1}{m}\right) \\
&= 1 - \exp\left(-\frac{(n-1)(n)}{2m}\right). \qquad (4)
\end{aligned}$$

For ease of notation, we'll denote

$$A = \exp\left(-\frac{(n-1)(n)}{2m}\right),$$

which is our approximation of the probability of there not being a collision.

We were operating under the assumption that our approximation is reasonable, but we'd like some better idea as to how good the approximation is for actual values of m and n. For this, denote $x_k = k/m$:

$$\begin{aligned}
\varepsilon &= |(1 - A) - \Pr(\text{collision})| \\
&= |\Pr(\text{no collision}) - A| \\
&= \left|\prod_{k=0}^{n-1} (1 - x_k) - A\right| \\
&= \left|\prod_{k=0}^{n-1} \left(e^{-x_k} - \varepsilon_{x_k}\right) - A\right| \\
&= A\left|\left(\prod_{k=0}^{n-1} \left(1 - e^{x_k}\varepsilon_{x_k}\right) - 1\right)\right| \\
&= A\left(1 - \prod_{k=0}^{n-1} \left(e^{x_k} - x_k e^{x_k}\right)\right) \qquad (5) \\
&\leq A\left(1 - \left(e^{n/m} - \frac{n}{m}e^{n/m}\right)^n\right) \\
&= A\left(1 - \underbrace{\exp\left(\frac{n^2}{m}\right)\left(1 - \frac{n}{m}\right)^n}_{M}\right) \\
&= \exp\left(-\frac{(n-1)(n)}{2m}\right)\left(1 - \exp\left(\frac{n^2}{m}\right)\left(1 - \frac{n}{m}\right)^n\right) \qquad (6)
\end{aligned}$$

The step yielding Equation (5) is based on some calculus, and the behavior of the terms being multiplied together, given the possible values of x_k. The sign change in Equation (5) tells us that the $\Pr(\text{collision}) \geq 1 - A$; in order to properly bound the probability of collision, we need to include the contribution of ε.

Together, Equations (4) and (6) allow us to bound the probability of a collision given arbitrary input parameters, namely

$$\begin{aligned}
\Pr(\text{collision}) &= 1 - A + \varepsilon \\
&\leq 1 - A + A(1 - M) \\
&\leq 1 - AM \\
&= 1 - \exp\left(-\frac{(n-1)(n)}{2m}\right)\exp\left(\frac{n^2}{m}\right)\left(1 - \frac{n}{m}\right)^n \\
&= 1 - \exp\left(\frac{n(n+1)}{2m}\right)\left(1 - \frac{n}{m}\right)^n
\end{aligned}$$

We could bound the probability of collision as being less than p by bounding

$$\Pr(\text{collision}) \leq 1 - \exp\left(\frac{n(n+1)}{2m}\right)\left(1 - \frac{n}{m}\right)^n < p,$$

which is true if

$$1 - p < \exp\left(\frac{n(n+1)}{2m}\right)\left(1 - \frac{n}{m}\right)^n. \tag{7}$$

Taking the log of both sides, we find

$$\log(1 - p) < \left(\frac{n(n+1)}{2m}\right) + n\log\left(1 - \frac{n}{m}\right). \tag{8}$$

Equation (8) is readily verified for various parameter settings using a computer algebra system.

4 Collision of Blocks

In some circumstances, we have the case where collisions are in some sense "binned", that each choice could collide with some fixed number of other choices. Conceptually, mapping a random output through any non-injective function could elicit such a situation, but we also encounter this situation in more natural settings. In this case, we'll describe the bin size as b. Happily, this proceeds in much the same way as the above. First note that in terms of exact calculation, we have

$$\begin{aligned}
\Pr(\text{collision}) &= 1 - \prod_{k=0}^{n-1}\frac{m - bk}{m} \\
&= 1 - \prod_{k=0}^{n-1}\left(1 - \frac{bk}{m}\right) &(9)\\
&= 1 - \prod_{k=0}^{n-1}\left(1 - \frac{k}{(m/b)}\right). &(10)
\end{aligned}$$

Equation (9) gives the equivalent formulation as Equation (1), but perhaps Equation (10) gives a better intuitive notion of what is going on. This shows us that in the case that we are concerned with block

collisions, we have exactly the same situation as the non-block case, but the number of buckets is the ratio[4] m/b.

As the exact calculation takes this form, we can naturally use the same approximation that we have already developed by replacing the term m with the ratio m/b.

5 A Note on Conditional Probability

We occasionally encounter situations where we have already done the work necessary to calculate the probability of a birthday collision after n outputs into two collections of buckets (where each collection has a possibly different number of buckets) and we then want to consider a setting where these are combinatorially combined. For example, imagine that we have two random processes, the first of which outputs a 16-bit value and the second of which outputs an 8-bit value. Using the above development, it's easy to calculate the probability of collision for each after 2^6 outputs (indeed, this can be accomplished with no need to approximate, given a modern computer). One can directly do the same for a 24-bit value.

Table 1: Independent Events: A Non-Example

	$(n = 2^6)$
m	Pr(collision)
2^8	$p_8 = 0.99982$
2^{16}	$p_{16} = 0.030303$
2^{24}	$p_{24} = 0.000120156$

We know that the axiom of conditional probability tells us that if we have two events A and B, then

$$\Pr(A \cap B) = \Pr(A|B) \cdot \Pr(B),$$

which, in the case that A and B are independent, gives the very useful

$$\Pr(A \cap B) = \Pr(A) \cdot \Pr(B) \qquad \text{(when } A \text{ and } B \text{ are independent.)}$$

Observe that $p_{24} \neq p_8 \cdot p_{16}$, so the probability of collisions are *not independent*. The reason for this is somewhat clearer once one considers the example of two single-bit quantities.

After three outputs, each single-bit quantity is guaranteed to collide with some past output (there are, after all, only two possible buckets!), but a single two-bit quantity only has a 62.5% chance of colliding after 3 outputs. We need more than both collisions happening for *some* output, they must collide on the *same* output.

The conclusion we should draw from this is that we can't simply multiply chances of collision together and expect to get something sensible out, because these events are not independent.

[4] We'll try to avoid including the time measurement "fortnights" in any calculations featuring this ratio of "buckets per bin".

6 Motivating Examples

In each of the following examples, we use Equation (8) to find the maximal integer value of ℓ so that if, for a given value of m, after $n = 2^\ell$ outputs, the probability for a collision is less than some given maximum probability bound for a collision, p.

6.1 CBC or OFB Modes With AES and DES

In CBC mode, information about the underlying plaintext leaks in the event that the cipher outputs a previously output value. In OFB mode, such a collision causes the complete compromise of all data encrypted after that collision.

If we model DES and AES as random maps, then we can directly apply the material we have just developed to find out how many ciphertexts can be output before the probability of producing a collision becomes higher than some bound. For DES (which has a 64-bit block size), we let $m = 2^{64}$ and for AES (which has a 128-bit block size), we let $m = 2^{128}$.

Table 2: Maximum Number of Encryptions for DES and AES in CBC or OFB mode

Algorithm	p			
	2^{-1}	2^{-20}	2^{-32}	2^{-40}
DES	2^{32}	2^{22}	2^{16}	2^{12}
AES	2^{64}	2^{54}	2^{48}	2^{44}

6.2 IPSec with GCM

RFC 4106 provides a way of using AES GCM mode within IPSec's Encapsulating Security Payload (ESP), so long as key agreement is handled using "an automated key management system", commonly IKE or IKEV2. For this section, we assume that IKE or IKEV2 are being used.

In NIST Special Publication 800-38D, there must be a chance of less than 2^{-32} that any particular GCM IV is re-used with a particular key. For all calculations that follow in this section, we thus set $p = 2^{-32}$, unless otherwise explicitly stated.

In this use, the GCM IV is always a 12-octet (96-bit) value. The first part of the GCM IV is the "salt", a 4-octet value that is created using the same IKE KDF process that produces the session keys for the security association; for this analysis, the salt and the session key are modeled as randomly distributed. We refer to each set of independent IKE (or IKEV2) security associations as a *key context*; each key context has a particular (fixed) key and salt. The last part of the GCM IV is an 8-octet ESP IV, which is set in one of three ways:

1. A counter with a fixed starting point.
2. A counter with a randomly generated starting point.
3. A randomly generated value for each message.

For completeness, we also provide an analysis for the case where four of the octets of the ESP IV are fixed to some ID associated with the sender (e.g., the sender's IP) in order to comply with NIST FIPS 140-2 IG A.5 (dated 1/15/2015).

6.2.1 Fixed Starting Point

With a fixed starting point, one necessarily collides in the event that the key and salt collide, so the size of the colliding bitstring is the key length plus 32 bits for the salt. As such, we have the results in Table 3.

Table 3: Allowable Key Contexts (Fixed Starting Point Bounds)

Key Length	Key Contexts
128	2^{64}
192	2^{96}
256	2^{128}

In this case, the size of the counter does not affect the collision bounds but does limit the number of possible messages within a single key context. As such, encoding the device's ID into the ESP IV doesn't affect the number of allowable key contexts that can be used for the given collision probability bound, but does make it so that the number of allowable key contexts applies only on a per-module basis. The resulting number of messages per key context limits here are at most 2^{64} messages for implementations that use the full 8-octet ESP IV as a counter, and 2^{32} messages for implementations using 4 of the ESP IV octets to encode an ID (leaving the remaining 4 octets to encode the counter.)

6.2.2 Randomly Generated Starting Point

With a randomly generated starting point, you get the benefit of being able to guarantee no collision of the ESP IV within a fixed key context, along with some added resistance to collision provided by the random starting index. To establish how much added collision resistance we get, we need to first limit the number of messages that can be encrypted within a particular key context; this is a tunable parameter, so we examine a few reasonable settings for the number of messages per key context, which we'll call T.

When you choose a starting point for the ESP IV, x, you not only exclude all the ESP IV selections up to $x + T$ (which could be used within this key context) but also those starting after $x - T$ (as selecting one of these could eventually cause a collision). We thus exclude at most $2T$ ESP IV selections for each choice (thus $b = 2T$). As a special case, if there is effectively no limit on the number of messages, then $T = 2^{64}$ in the case where 8 octets are used for the counter, and $T = 2^{32}$ in the instance where 4 octets are used for the counter. (In this "no limit" case, we have made it so that when a key and salt collision happen, then there will be a full GCM IV collision).

In order for the tuple (Key, GCM IV) to collide, we must have a simultaneous collision of the key and salt, then additionally encounter a block collision. As a sample calculation of the number of buckets for a

128-bit key and a maximum of 2^{32} messages per key context, we have

$$m = \overbrace{2^{128}}^{\text{AES key}} \cdot \overbrace{2^{32}}^{\text{Salt}} \cdot \frac{\overbrace{2^{64}}^{\text{(Binned) Counter}}}{2 \cdot 2^{32}} = 2^{128+32+31} = 2^{191}.$$

Table 4: Allowable Key Contexts (Random Starting Point Bounds)

Key Length	Max Encrypts per Key Context (T)			
	2^{16}	2^{32}	2^{48}	2^{64}
128	2^{88}	2^{80}	2^{72}	2^{64}
192	2^{120}	2^{112}	2^{104}	2^{96}
256	2^{152}	2^{144}	2^{136}	2^{128}

If we effectively do not limit the number of messages per key context (and allow the full 2^{64} distinct encryptions), this approach is equivalent to the case where we have a fixed starting point (this column has the same results as in Table 3).

If, in response to NIST FIPS 140-2 IG A.5, we fixed 4 octets of our ESP IV by setting them to a device-specific identifier (e.g. the device IP), then there are only thirty-two bits of counter left to be randomly assigned. As such, we surely couldn't encrypt more than 2^{32} messages, and we are left with the bounds described in Table 5.

Table 5: Allowable Key Contexts (Random Starting Point Bounds, Compliance with IG A.5)

Key Length	Max Encrypts per Key Context (T)	
	2^{16}	2^{32}
128	2^{72}	2^{64}
192	2^{104}	2^{96}
256	2^{136}	2^{128}

This modification results in a reduction of both the allowable key contexts, and the total messages that can be encrypted within a single key context. The advantage of this approach is that it makes each GCM IV device specific.

6.2.3 Randomly Generated Per-Message ESP IVs

In IPsec, the first four octets of the GCM IV are fixed within a key context, so there are only eight octets available to be randomly set within a fixed key context. In order to maintain the desired per-key-context collision bound of 2^{-32}, we must restrict to at most 2^{16} messages within a fixed key context.

In the event that a key-context collision has occurred (*i.e.*, a particular key and salt combination have been used again), the total collision probability would surely rise above our bound, so we are then again left with the situation addressed already in Table 3.

6.2.4 Attacker Model

An attacker who wants to be prepared to attack a system using GCM in this way will be forced to dedicate some amount of computational and storage resources in order to notice that a collision has occurred. There are two basic approaches: an attacker can monitor within a particular key context, or an attacker can monitor across multiple key contexts which (in the case where the address is not encoded into the GCM IV) could take place over any device capable of communicating using GCM with IPsec.

Attackers monitoring for collisions within a fixed key context need to store all the data within that particular key context. As the only approach outlined above susceptible to this style of attack is the random per-message ESP IV approach, the attacker would have to store at most 2^{16} messages, each of which requires storage of at least $9 \approx 2^3$ octets (a 8-octet ESP IV and a 1-octet ciphertext must both be stored) and at most roughly 2^{32} octets (for IPv6) or 2^{16} octets (for IPv4). Within a key context, duplication of the sent ESP IV implies a full (Key, GCM IV) collision, so an attacker just needs to compare the ESP IV on at most 2^{16} messages.

For an attacker to have a 2^{-32} chance of success, they must then be willing to store 2^{19} to 2^{48} octets (roughly 0.5 MB to 280 TB), with a practical maximum being approximately 4 GB (for attackers not dealing with IPv6 implementations that support jumbograms). Looking for collisions within this data set requires at most 2^{16} operations, so this is an overall modest effort.

Attackers monitoring for collisions across multiple key contexts have a harder time. They can again watch for collisions within the 8-octet ESP IV, but this reveals relatively little. In practice, comparison of these ESP IVs may require substantial computation (given the scale of the numbers involved), but for the purpose of this analysis, we assume that establishing if a particular ESP IV has been seen before is a computation that can occur in $\mathcal{O}(1)$ operations (e.g., using the RAM model, a table lookup or optimal hash table lookup). We also assume that the attacker can establish if there has indeed been a (Key, GCM IV) collision in $\mathcal{O}(1)$ operations once the ESP IV matches. Importantly, the number of operations an attacker needs to perform is never less than the number of messages gathered for an attack, as each additional message requires some processing to check if the ESP IV has re-occurred, and if so then to check if this is an example of a (Key, GCM IV) collision.

For example, if an attacker monitors 2^{64} distinct key contexts, each with 2^{16} messages in them, then the attack has a work factor of 2^{80} computations. These results are summarized in Table 6. All the attacks summarized in Table 6 have a probability of success of 2^{-32}; for comparison, the key exhaustion attacks with this probability of success have a computational work factor of 2^{96} for a 128-bit AES key, 2^{160} for a 192-bit AES key, and 2^{224} for a 256-bit AES key.

A more reasonable estimation of the effort required for an attack would be the the attacker's expected resource use for a successful attack; for this, we examine the case of a probability bound of $p = \frac{1}{2}$. These results are summarized in Table 7.

All the attacks summarized in Table 7 have a probability of success of 2^{-1}; for comparison, the key exhaustion attacks with this probability of success have a computational work factor of 2^{127} for a 128-bit AES key, 2^{191} for a 192-bit AES key, and 2^{255} for a 256-bit AES key.

For Tables 6 and 7, attacks with no computational advantage over the relevant key exhaustion attacks are greyed out. Within these tables, the stated computational bounds can be converted to storage bounds (in octets) by adding a storage factor to each exponent; this storage factor is between 3 (associated with

attacks on data streams containing only the smallest possible messages) to 32 (associated with attacks on data streams containing only the largest possible messages).[5]

Table 6: Attack Computational Cost ($p = 2^{-32}$)

Key Length	Max Encrypts per Key Context			
	2^{16}	2^{32}	2^{48}	2^{64}
Fixed Starting Point				
128	2^{80}	2^{96}	2^{112}	2^{128}
192	2^{112}	2^{128}	2^{144}	2^{160}
256	2^{144}	2^{160}	2^{176}	2^{192}
Random Starting Point				
128	2^{104}	2^{112}	2^{120}	2^{128}
192	2^{136}	2^{144}	2^{152}	2^{160}
256	2^{168}	2^{176}	2^{184}	2^{192}
Random Starting Point (A.5)				
128	2^{88}	2^{96}	N/A	N/A
192	2^{120}	2^{128}	N/A	N/A
256	2^{152}	2^{160}	N/A	N/A
Randomly Generated Per-Message ESP IV				
128	2^{80}	N/A	N/A	N/A
192	2^{112}	N/A	N/A	N/A
256	2^{144}	N/A	N/A	N/A

If an attacker is interested in finding a collision for one particular key context (and can thus discard data when it is found not to be related to the desired key context), then the amount of storage is dramatically reduced, but so is the attacker's chance of success. In order to benefit from a birthday style collision, the attacker must be able to in some sense retain all the data they have seen, which (as seen above) imposes a massive storage and computational cost.

6.2.5 Standards Details

Some of the above apparent options run afoul of various requirements or guidance. For what follows, we adopt the language of NIST SP800-38D. This document provides for two ways of constructing the GCM IV, a *deterministic construction* (specified in Section 8.2.1) and a *RBG-based Construction* (specified in Section 8.2.2).

Deterministic Construction

Setting a fixed starting point for the ESP IV must be considered a deterministic construction of the GCM IV. Setting a randomly generated starting point for the ESP IV and then incrementing from there may also be considered a deterministic construction of the GCM IV. In this setting, the salt can be considered

[5] In fact, the average cost per octet is best for the attacker when each message is largest!

Table 7: Attack Expected Computational Cost ($p = 2^{-1}$)

Key Length	Max Encrypts per Key Context			
	2^{16}	2^{32}	2^{48}	2^{64}
Fixed Starting Point				
128	2^{96}	2^{112}	2^{128}	2^{144}
192	2^{128}	2^{144}	2^{160}	2^{176}
256	2^{160}	2^{176}	2^{192}	2^{208}
Random Starting Point				
128	2^{119}	2^{127}	2^{135}	2^{144}
192	2^{151}	2^{159}	2^{167}	2^{176}
256	2^{183}	2^{191}	2^{199}	2^{208}
Random Starting Point (A.5)				
128	2^{103}	2^{112}	N/A	N/A
192	2^{135}	2^{144}	N/A	N/A
256	2^{167}	2^{176}	N/A	N/A
Randomly Generated Per-Message ESP IV				
128	2^{96}	N/A	N/A	N/A
192	2^{128}	N/A	N/A	N/A
256	2^{160}	N/A	N/A	N/A

as part of the fixed field. The salt does not uniquely identify the device, so to satisfy the requirements of NIST FIPS 140-2 IG A.5, an additional 32 bits of the ESP IV must be fixed to some identifying information (e.g., the device IP). This results in the invocation field being at most 32 bits.[6]

The above analysis suggests that requiring this fixed identifier within the ESP IV does not substantially affect the difficulty of launching an attack from a computational perspective (indeed, it makes it worse in the randomly-set case); it does make it so that an attacker must gather a ridiculous amount of data from one target, but this doesn't seem wildly less practical than an attacker gathering an even more ridiculous amount of data from a possibly vast number of targets. Fundamentally, the attacker must store all this data somewhere, and the amount of data involved is several orders of magnitude beyond anything available to all of humanity (the minimum storage for any of these attacks is on the order of 1 yottabytes, or 10^{24} bytes), to say nothing of single (even well funded) attackers.

RBG-based Construction

Starting with a randomly generated ESP IV value (and then incrementing from there) and setting the ESP IV randomly for each message can both *almost* be considered a "RBG-based Construction" of the GCM IV. In this construction, the salt value must be considered part of the *free field*, as it is not generated using an "approved RBG with a sufficient security strength", does not get incremented along with the rest of the random field as described in Paragraph 2 of Section 8.2.2, and is fundamentally associated with the

[6] Note that SP800-38D does *not* appear to require encoding the address in case of IPSEC with IKE, as use of IKE in IPSEC assures that (as in paragraph 4 of Section 8.2.1) "a fresh key is limited to a single session of a communication protocol", with probability bound vastly better than 2^{-32}.

key context, not the particular message. The remaining 64-bits of the GCM IV (that is, the ESP IV) is thus categorized as the *random field*. Unfortunately, even if this random field is generated by an approved RBG, it is not sufficiently large to meet the requirements of NIST SP800-38D Section 8.2.2 (the random field is required to be at least 96 bits by Paragraph 1 of Section 8.2.2).

If the IKE KDF is considered suitably strong to be allowed to produce the session keys used in the IPsec security associations, then it seems reasonable to consider its output as random-looking enough to make up a portion of the random field. Similarly, though fixing a 4-octet portion of the random field does make the discussed 2^{32} message maximum the wrong bound for per-message randomly generated random fields, we saw above that a message maximum of 2^{16} yields the correct collision bound. In the instance that random field is initially set randomly and then incremented, this maximum number of messages has less significance.