On Weak Keys and Forgery Attacks against Polynomial-based MAC Schemes

Gordon Procter and Carlos Cid

Information Security Group,
Royal Holloway, University of London
{gordon.procter.2011,carlos.cid}@rhul.ac.uk

Abstract. Universal hash functions are commonly used primitives for fast and secure message authentication in the form of Message Authentication Codes (MACs) or Authenticated Encryption with Associated Data (AEAD) schemes. These schemes are widely used and standardised, the most well known being McGrew and Viega's Galois/Counter Mode (GCM). In this paper we identify some properties of hash functions based on polynomial evaluation that arise from the underlying algebraic structure. As a result we are able to describe a general forgery attack, of which Saarinen's cycling attack from FSE 2012 is a special case. Our attack removes the requirement for long messages and applies regardless of the field in which the hash function is evaluated. Furthermore we provide a common description of all published attacks against GCM, by showing that the existing attacks are the result of these algebraic properties of the polynomial-based hash function. We also greatly expand the number of known weak GCM keys and show that almost every subset of the keyspace is a weak key class. Finally, we demonstrate that these algebraic properties and corresponding attacks are highly relevant to GCM/2^+, a variant of GCM designed to increase the efficiency in software.

Keywords: Universal Hashing, MAC, Galois/Counter Mode, Cycling Attacks, Weak Keys.

1 Introduction

The study of information-theoretic message authentication codes and universal hashing was initiated by Gilbert et al. [18] and Carter and Wegman [12,13,44,45]. Universal hash functions can be used to construct message authentication codes in both the information-theoretically secure and computationally secure settings (see [10,45]). Simmons [39] provides a general summary of the theory of unconditionally secure message authentication. Bernstein [3,5] provides a thorough description of the genealogy and more recent literature of unconditionally secure message authentication, including a description of the contributions of Bierbrauer et al. [6], den Boer [14], and Taylor [42] to polynomial-based hashing. Bernstein [4] also gives an interesting overview of the security of universal hash function based MACs in the computationally secure setting. Shoup [38] describes several methods for realising universal hash function families that are related to polynomials including the evaluation hash [6,14,42] which is a variant of the division hash or cryptographic CRC of Krawczyk [25] (itself a variant of Rabin's fingerprinting codes [33]).

In this paper, we focus on message authentication codes constructed from universal hash functions that are realised by polynomial evaluation. These are widely used and standardised; for examples see [5,15,21,24,26,37]. McGrew and Viega's Galois/Counter Mode (GCM) [30] is the most widely deployed polynomial-based scheme. The algorithm is generally assumed to be secure, with a small number of papers containing attacks against the authentication component via the universal hash function: Ferguson's attack against truncated GCM tags [17], demonstrating that the security of short tags is significantly lower than would be expected; Joux's 'forbidden attack' [23], illustrating the brittleness of GCM under nonce reuse; Handschuh and Preneel's extension to Joux's attack [20]; and Saarinen's cycling attacks [36], which highlight a weakness due to the underlying algebraic structure of a hash function based on polynomial evaluation.

Both Handschuh and Preneel [20] and Saarinen [36] have described classes of weak keys for polynomial evaluation based universal hash functions, with Saarinen particularly focusing on GCM.

Contributions. A motivation of this work was the observation that all existing attacks against GCM are algebraic in nature, and in fact seem to exploit a fundamental underlying algebraic structure of the polynomial-based hash function. The contributions of this paper are to identify and study some of the properties of hash functions based on polynomial evaluation that are the result of this underlying algebraic structure. As a result, we are able to describe a general forgery attack, of which Saarinen's cycling attack is a special case; our attack can, however, be used with short messages, applies regardless of the field in which the hash is evaluated, and facilitates length extension attacks against GCM. Furthermore, we provide a common description of all published attacks against GCM by showing that the existing attacks are the result of these algebraic properties of the polynomial-based hash function. We also greatly expand the number of known weak GCM keys, and show that almost every subset of the keyspace is a weak key class. Finally, we demonstrate that these algebraic properties and corresponding attacks are highly relevant to GCM/2^+, a variant of GCM designed to increase the efficiency in software. We note that the attacks presented in this paper do not in any way contradict the security bounds for GCM given by McGrew and Viega [31]. However, the algebraic properties (and related attacks) discussed in this paper appear to be an inherent feature of polynomial-based authentication schemes and therefore should be considered in the security assessment of new schemes and extensions of existing ones. Additionally, we consider the consequences of a related property on another polynomial based scheme in Section 9.

A preliminary version of this paper was presented at Fast Software Encryption 2013. This paper includes additional explanation of our results and further details on their relationship to the previously known results; Section 9 has been added and we extend our results on weak keys to include all two element subsets of the keyspace in a specific case.

Structure. This paper is structured as follows. In Section 2 we introduce the notation that will be used throughout this paper and provide a brief description of the syntax and security of message authentication codes. In Section 3 we give a basic overview of four schemes that use hash functions based on polynomial evaluation for message authentication, including GCM and SGCM. In Section 4, we briefly describe the existing results on the security of polynomial-based MACs. In Section 5 we describe the main technique used in this paper for the cryptanalysis of polynomial-based authentication schemes and discuss some features of the resulting attack that make it more interesting than cycling attacks. Section 6 explains the relationship between the properties described in this paper and the known results on the security of polynomial-based MACs. In Section 7 we show that there are many more weak key classes for hash functions based on polynomial evaluation than have previously been described and suggest a method to realise a key recovery attack against polynomial-based hash function schemes. In Section 8 we apply the attacks described in this paper to GCM/2^+. In 9 we identify similar techniques that can be applied to Square Hash, another universal hash function family based on a different form of polynomial evaluation. Section 10 contains a discussion of the consequences of this paper.

2 Preliminaries

2.1 Notation

We consider a message M parsed as $M_1||\dots||M_m$, where each M_i is n bits long and $||$ represents concatenation of strings. In the syntax of authenticated encryption with associated data [34], this message consists of associated data $A \in \mathcal{A}$ that is authenticated but not encrypted and plaintext $P \in \mathcal{P}$ that will be encrypted and authenticated.

A family of hash functions will be denoted $\mathcal{H} = \{h_H : \{0,1\}^\star \to \{0,1\}^n \mid H \in \mathcal{K}_\mathcal{H}\}$ with each hash function h_H indexed by a key $H \in \mathcal{K}_\mathcal{H}$. A block cipher E is a family of permutations on $\{0,1\}^n$, with each permutation indexed by a key $k \in \mathcal{K}_E$. The application of a block cipher to input $x \in \{0,1\}^n$ using key k will be denoted by $E_k(x)$. A nonce will be denoted by N.

A finite field will be denoted by $\mathbb{K}$ unless the order of the field has particular relevance, in which case it will be denoted by $\mathbb{F}_{p^r}$ with $|\mathbb{F}_{p^r}| = p^r$. The multiplicative group of a field $\mathbb{K}$ will be denoted by $\mathbb{K}^\star$.

2.2 Universal hash functions

A family of hash functions is said to be *ϵ–almost XOR universal* if for every $M, M' \in \{0,1\}^\star$ with $M \neq M'$ and for every $c \in \{0,1\}^n$, $\Pr_{H \in \mathcal{K}_\mathcal{H}}[h_H(M) \oplus h_H(M') = c] < \epsilon$. Throughout this paper ϵ–almost XOR universal will be abbreviated to ϵ–AXU. This condition was introduced by Krawczyk [25] under the name *ϵ–OTP–Secure* as it is a necessary and sufficient condition for unconditional MAC security when the output of the hash function is encrypted with the one time pad in a field of characteristic 2. In this paper we will generally refer to ϵ–AXU hash function families; however any remark made that requires an ϵ–AXU hash function family in characteristic 2 will also hold for an ϵ–almost strongly universal [41] or ϵ–almost $\varDelta$ universal [40] hash function family in any finite field.

A polynomial based hash function family is a common way to realise an ϵ–AXU hash function family. Shoup [38] describes several examples of this type of construction; the main example of interest to this paper is the evaluation hash. In the case of the evaluation hash the message M determines a polynomial $g_M = \sum_{i=1}^{m} M_i x^i \in \mathbb{K}[x]$, where $M = M_1 || \ldots || M_m$ with each $M_i \in \mathbb{K}$. The hash key is an element $H \in \mathbb{K}$ and we define the hash function by $h_H(M) = g_M(H)$.

There are several methods for turning a universal hash function into a message authentication code (see [10,45] for early examples). The two most common methods are $E_k(N) + h_H(M)$ and $E_k(h_H(M))$.

2.3 Syntax

We will follow Black et al. [7] for a description of the syntax of nonce-based message authentication schemes. A message authentication scheme is a pair of algorithms, Gen and MAC, with four associated sets: $\mathcal{K}$, the set of possible keys; $\mathcal{M}$, the message space; $\mathcal{N}$, the set of nonces and $\mathcal{T}$, the set of possible authentication tags.

The key generation algorithm Gen takes as input the security parameter and probabilistically outputs the shared key $k \in \mathcal{K}$. The algorithm MAC takes as input a key $k \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, and a message $M \in \mathcal{M}$ and outputs a tag $T \in \mathcal{T}$. The authenticity of a tuple (N, M, T) is verified by computing $\mathsf{MAC}(k, N, M)$: if $T = \mathsf{MAC}(k, N, M)$ then the tag is valid, otherwise it is invalid.

2.4 Security

An adversary attacking a message authentication scheme is given access to two oracles: a tag generation oracle $\mathcal{S}$ and a verification oracle $\mathcal{V}$. At the beginning of the experiment Gen is run to obtain k, then $\mathcal{S}$ takes queries (N, M) and returns $\mathsf{MAC}(k, N, M)$. The verification oracle takes queries (N, M, T) and returns 1 if $T = \mathsf{MAC}(k, N, M)$ or 0 otherwise. An adversary is said to successfully forge an authentication tag if they can produce a verification query (N, M, T) so that $\mathcal{V}$ returns 1 when (N, M) was not previously queried to $\mathcal{S}$.

A common restriction of this security notion is to nonce-respecting adversaries where, although the adversary can control the nonce, they never query $\mathcal{S}$ for (N, M') if they have previously queried $\mathcal{S}$ for (N, M).

For polynomial-based MACs, McGrew and Viega [31], Ferguson [17], and Handschuh and Preneel [20] all assert that the probability of creating a valid (non-truncated) tag having seen a single valid (message, tag) pair is approximately $m/|\mathbb{K}|$ where the polynomial is evaluated in $\mathbb{K}$ and m is the length of message that the construction operates on. It is worth emphasising that in this context, m is the *maximum* permissible message length. This is included in the original paper [31] but is not made explicitly clear in the later papers [17,20]. In this paper we will demonstrate the importance of this distinction via a method of forging GCM tags using a longer message than the one that was given in the valid (message, tag) pair from the tag generation oracle.

Throughout this paper we will focus on GCM for concreteness however the majority of the comments apply equally to any other hash function based on polynomial evaluation. Most of the results in this paper apply equally to both common constructions of MACs from universal hash functions, either $T = E_k(N) + h_H(M)$ or $T = E_k(h_H(M))$, as our results are based on collisions in the hash function. Where necessary it will be made clear that a remark is dependent on one of these general constructions or the specific structure of GCM.

2.5 Weak Keys

For any cryptographic algorithm, a relevant question for its security assessment is whether it contains *weak keys*. Handschuh and Preneel [20, Sect. 3.1] give the following definition of weak keys:

> In symmetric cryptology, a class of keys $[\mathcal{D}]$ is called a *weak key class* if for the members of that class the algorithm behaves in an unexpected way and if it is easy to detect whether a particular unknown key belongs to this class. For a MAC algorithm, the unexpected behavior can be that the forgery probability for this key is substantially larger than average. Moreover, if a weak key class $[\mathcal{D}]$ is of size C, one requires that identifying that a key belongs to this class requires testing fewer than C keys by exhaustive search and fewer than C verification queries.

Handschuh and Preneel [20] and Saarinen [36] have identified weak key classes for GCM and other polynomial-based hashes; we discuss and extend these classes in Section 7.

3 Polynomial-based Authentication Schemes

We present below a brief description of some of the main authentication schemes based on polynomial evaluation hash functions that are of relevance to our work. We will also discuss Square Hash in Section 9; Square Hash is another family of universal hash functions also based on polynomials.

3.1 Galois/Counter Mode

Galois/Counter Mode (GCM) is an AEAD scheme submitted to NIST by McGrew and Viega in 2004, with the specification slightly revised in 2005 [30] (although the revision contained *'no normative changes [from the 2004 specification]'*). GCM combines counter mode encryption with a polynomial evaluation based MAC following the Encrypt–then–MAC paradigm, although the authentication key is derived from the block cipher key.

AES–GCM encryption takes as input: a key k, an initialisation vector IV (the nonce), plaintext $P = P_1||\dots||P_p$ and additional data $A = A_1||\dots||A_a$. The key is 128, 192 or 256 bits long, the IV should preferably be 96 bits long although any length is supported (see [22]), and for each i, $|P_i| = |A_i| = 128$ except for perhaps a partial final block. With this input, AES–GCM returns a ciphertext $C = C_1||\dots||C_p$ (the same length as the plaintext) and an authentication tag T.

The plaintext is encrypted using AES in counter mode, under key k with counter value starting at CTR_1. If the IV is 96 bits long the initial counter value (CTR_0) is $\mathsf{IV}||0^{31}1$, otherwise it is a polynomial evaluation based hash of IV after zero padding (using the hash key described below). For each i, $\mathsf{CTR}_i = \mathsf{inc}(\mathsf{CTR}_{i-1})$, where $\mathsf{inc}(\cdot)$ increments the last 32 bits of its argument (modulo 2^{32}).

The authentication tag is computed from a polynomial evaluation hash (in $\mathbb{F}_{2^{128}}$). The message M is parsed as 128-bit blocks (with partial final blocks zero padded) and each block is interpreted as an element of $\mathbb{F}_{2^{128}}$. The first block M_1 encodes the length of the (unpadded) plaintext and additional data and will be referred to as the 'length field' throughout this paper. This is followed by blocks of additional data $M_2, \dots, M_{a+1} = A_a, \dots, A_1$ and then the encrypted plaintext $M_{a+2}, \dots, M_{a+p+1} = C_p, \dots, C_1$. Note that in this description the labelling of the blocks M_i are reversed from those given in the original GCM specification as this gives a neater description of the polynomial used in evaluating the hash function. The hash key H is derived from the block cipher key: $H = E_k(0^{128})$. The hash function is then computed as $h_H(M) = \sum_{i=1}^{a+p+1} M_i H^i$ (where all operations are in $\mathbb{F}_{2^{128}}$). The authentication tag is given by:

$$T_M = E_k(\mathsf{CTR}_0) \oplus h_H(M).$$

3.2 Sophie Germain Counter Mode

In 2012, Saarinen [36] observed cycling attacks against GCM and other polynomial MACs and hashes. Following this Saarinen proposed SGCM [35] as a variant of GCM; SGCM differs from GCM only by the choice of field in which the hash is computed. SGCM uses $\mathbb{F}_q$, where $q = 2^{128} + 12451$, rather than $\mathbb{F}_{2^{128}}$, as $\mathbb{F}_q^\star$ has significantly fewer subgroups than $\mathbb{F}_{2^{128}}^\star$. It was claimed that SGCM offers increased resistance to cycling attacks as a result of this change.

3.3 GCM with Short Multiplications

In 2012 Aoki and Yasuda [1] proposed GCM/2^+, a variant of GCM that evaluates the hash function using 'short' multiplications in $\mathbb{F}_{2^{64}}$ rather than multiplications in $\mathbb{F}_{2^{128}}$. The motivation for this change is to increase the efficiency in software, where multiplications in $\mathbb{F}_{2^{128}}$ are significantly more expensive than those in $\mathbb{F}_{2^{64}}$. This change has significant implications for the security of the scheme, which we will discuss in Section 8.

3.4 Poly1305–AES

Bernstein proposed Poly1305–AES in 2005 [5][1]. Poly1305–AES takes as input two 128-bit keys, one for AES and one for the hash (with some specific bits set to zero); a 128-bit nonce; and a message (a byte string). The output of Poly1305–AES is a 128-bit authentication tag.

The hash of a message is computed by evaluating a polynomial at the secret key (in $\mathbb{F}_{2^{130}-5}$) and encrypting this by adding (also in $\mathbb{F}_{2^{130}-5}$) the output of $\mathsf{AES}_k(N)$ before reducing modulo 2^{128}.

[1] There is a preliminary version from 2004 on his website: `http://cr.yp.to/mac.html`

4 The Security of Polynomial-based Authentication Schemes

In this section, we briefly describe the main existing results on the security of polynomial-based MACs. Because GCM is the most prominent example of a message authentication scheme based on polynomial evaluation, most of these results were originally described in terms of GCM, however they can also be applied to more general polynomial-based schemes.

We will show in Section 6 that these results can be described as special cases of the properties discussed in Section 5.

4.1 Ferguson's Short Tag Attack

Ferguson's attack against GCM when short tags are used [17] begins by observing that, because the output of the hash function is encrypted additively, if the authentication tag is less than 128 bits long then a full collision in the hash function is not required as a collision in the leading bits of the hash function will still cause a collision in the authentication tag. The second observation is that, because the polynomial hash is evaluated in a field of characteristic 2, squaring is a linear operation. So, if message blocks M_{2^i} (for some i) are altered by an adversary, the effect on the authentication tag is a linear function of H. In particular, it is possible for the adversary to alter these message blocks in a way that guarantees particular bits of the authentication tag will not change. This means that the effective length of the authentication tag is reduced and forgeries become more likely.

Ferguson also notes that once a forgery has been observed, the adversary gains information about H. With each successive forgery, more information about H is derived and the adversary is able to use this information when manipulating the M_{2^i} and increase the number of bits of the authentication tag that are guaranteed not to change. Eventually the adversary will recover the entire hash key.

4.2 Joux's Forbidden Attack

Joux's 'forbidden attack' against GCM [23] requires two messages, M and M', that are authenticated with the same (key, IV) pair. Reusing the (key, IV) pair in GCM has the effect of reusing H, k and N. This allows the adversary to conclude that the XOR of the authentication tags is the hash of the XOR of the messages:

$$\begin{aligned} T_M \oplus T_{M'} &= (h_H(M) \oplus E_k(N)) \oplus (h_H(M') \oplus E_k(N)) \\ &= h_H(M) \oplus h_H(M') \\ &= h_H(M \oplus M') \end{aligned}$$

As the hash is a polynomial evaluated at H and the adversary knows T_M, $T_{M'}$, and both messages, they are able to derive a polynomial that is known to have a root at H, namely $h_H(M \oplus M') \oplus T_M \oplus T_{M'}$. Joux suggests that by collecting pairs of messages authenticated with the same IV, an adversary could compute the GCD of these polynomials and eventually recover the key. This attack is prevented if we only consider nonce-respecting adversaries.

4.3 Handschuh and Preneel

Handschuh and Preneel [20] describe methods to verify a guess for and to recover the hash key.

The method for verifying a guess $H^\star$ for the hash key requires an authentication tag on a message of at least two blocks. Suppose that a valid authentication tag is known for $M_1||M_2$, then pick any M_2' and compute $M_1' = M_1 + (M_2 - M_2')H^\star$. Then the authentication tag for $M_1||M_2$ is valid for $M_1'||M_2'$ if $H = H^\star$.

The key recovery attack extends the one described by Joux, as it does not require nonce reuse. Given a valid authentication tag on a message M, the adversary performs a verification query using the same tag but a different message M'; this message is chosen so that the polynomial defined by $M - M'$ has many distinct roots. A successful forgery implies that the hash key is one of the roots of this polynomial and so a binary search can be conducted on those roots in order to recover the hash key. The key recovery method was initially identified by Black and Cochran [8,9] but extended and generalised by Handschuh and Preneel. This attack is described as infeasible by Handschuh and Preneel in the case of GCM, due to the blocksize of 128 bits, however we will show that it is precisely as feasible as Saarinen's cycling attacks, which are described below.

Handschuh and Preneel [20, Sect. 3.1] also identify 0 as a weak authentication key for GCM and other similar constructions as $h_0(M) = 0$ for every message M.

4.4 Saarinen's Cycling Attacks

In 2012, Saarinen [36] proposed cycling attacks against GCM and other polynomial-based MACs and hashes. The key observation is that if a hash key H lies in a subgroup of order t, then $H^t = 1 \in \mathbb{K}$ and (for any i, j) message blocks M_i and M_{i+jt} can be swapped without changing the value of the hash.

For example (ignoring GCM's length encoding), if $H^4 = H$ then blocks M_1 and M_4 can be swapped without changing the value of the hash:

$$\begin{aligned} h_H(M_1||M_2||M_3||M_4) &= M_1 \cdot H \oplus M_2 \cdot H^2 \oplus M_3 \cdot H^3 \oplus M_4 \cdot H^4 \\ &= M_4 \cdot H \oplus M_2 \cdot H^2 \oplus M_3 \cdot H^3 \oplus M_1 \cdot H^4 \\ &= h_H(M_4||M_2||M_3||M_1). \end{aligned}$$

The attack is carried out by obtaining a valid tag for a message and performing a verification query using the same tag with the message formed by simply swapping the position of two message blocks. Saarinen observes that any t that divides $2^{128} - 1$ can be used and that swapping M_i and M_{i+t} will give a successful forgery with probability at least $\frac{t+1}{2^{128}}$.

Saarinen [36] also describes 'targeted bit forgery attacks' in which, instead of whole blocks, only some bits are swapped (subject to the condition that $M_i \oplus M_{i+t}$ remains constant). This method permits the adversary more control over the forged message and the corresponding plaintext; we discuss this further in Section 5.2.

Saarinen [36] also demonstrated that there are many more weak keys than those described by Handschuh and Preneel by showing that small-order subgroups of $\mathbb{K}^\star$ are weak key classes. The forgery technique described above is successful if the authentication key is an element of a small-order subgroup with order dividing the distance between the swapped message blocks. This gives a method for identifying whether the authentication key is in that class using one valid (message, tag) pair and a single verification query; these classes of weak keys meet Handschuh and Preneel's definition of weak keys (given in Section 2.5).

The observation of this technique and the large number of weak key classes was the motivation for proposing SGCM (described in Section 3.2).

5 The Algebraic Structure of Polynomial-based Authentication Schemes

This section describes the main observation that allows us to give a general forgery attack against polynomial-based MAC schemes. We describe a malleability property of polynomial-based MAC schemes and use this to identify a length-extension attack against GCM. Finally, we discuss possible methods to generate polynomials with the required properties.

5.1 A Generalised Forgery Attack

Let $\mathcal{H}$ be a family of hash functions $\mathcal{H} = \{h_H : \{0,1\}^\star \to \{0,1\}^n \mid H \in \mathcal{K}_\mathcal{H}\}$ based on polynomial evaluation and let M be an input string. Let $h_H(M) = g_M(H)$, where $g_M(x) = \sum_{i=1}^{m} M_i x^i \in \mathbb{K}[x]$ and $H \in \mathbb{K}$. Now let $q(x) = \sum_{i=1}^{r} q_i x^i \in \mathbb{K}[x]$ be a polynomial with constant term zero, such that $q(H) = 0$. Then it follows that

$$h_H(M) = g_M(H) = g_M(H) + q(H) = g_{M+Q}(H) = h_H(M+Q),$$

where $Q = q_1||q_2||\dots||q_r$ and the addition $M + Q$ is done block-wise (the shorter message is zero-padded if required). Thus, given a polynomial $q(x)$ satisfying these properties, it is straightforward to construct collisions for the hash function. Every polynomial in the ideal generated by $x^2 - Hx$ has these properties because every element of this ideal has roots at 0 and H. The ideal generated by $f(x)$ is defined as:

$$\langle f(x)\rangle = \{r(x) \cdot f(x) | r(x) \in \mathbb{K}[x]\} \subseteq \mathbb{K}[x].$$

In particular, the ideal $\langle x^2 - Hx\rangle$ contains precisely the polynomials in $\mathbb{K}[x]$ that have $x^2 - Hx$ as a factor.

Collisions in the hash function correspond to MAC forgeries by substituting the original message for the one that yields a collision in the hash function. These forgeries arise from collisions in the hash function and hence the messages can be substituted without any dependence on the method or key used to encrypt the output of the hash function. This method allows an adversary to create forgeries when he has seen a tuple of (nonce, message, tag) by only modifying the message.

We remark that $g_M(x)$ is defined to have a zero constant term, if this were not the case and the hash of a message was encrypted additively (i.e. $T = E_k(N) + h_H(M)$) it would be possible to flip bits in the first message block and flip the same bits in the authentication tag to create a valid forgery. Because of this, the polynomial $q(x)$ that is used to forge via a full hash collision will always have x as a factor.

One of the main contributions of this paper is to observe that by working with polynomials in the ideal $\langle x^2 - Hx\rangle$, it is straightforward to produce forgeries for polynomial evaluation based authentication schemes. For example, Saarinen's cycling attacks [36] are realised by working with particular polynomials, namely $x^t - x$ (for more detail, see Sections 4.4 and 6.4). The forgery is successful if $(x - H)|(x^t - x)$ and therefore if $x^t - x \in \langle x^2 - Hx\rangle$. However, the forgery will be successful if *any* polynomial in this ideal is used to mount a similar attack. Furthermore, use of these polynomials also makes it possible to test for membership of large subsets of the keyspace with a single valid (message, tag) pair and a single verification query (see Section 7).

We also note that it is possible to extend the set of polynomials that are suitable for use as a forgery polynomial in the case where the authentication tag is created by additively encrypting the output of the hash function. This result has also been described by Zhu, Tan, and Gong [46], who refer to the earlier version of this paper; we give a more algebraic treatment by casting the result in terms of quotient rings of $\mathbb{K}[x]$. It is possible to predict additive relations between the output of the hash function on two different messages and, in this case, these differences are preserved by the additive encryption. This allows an adversary to manipulate the value of the authentication tag accordingly. If the output of the hash function is encrypted using a block cipher then these relations are not preserved and so a full collision is required, as described above.

In this more general setting, we consider the cannonical homomorphism into the quotient ring:

$$\begin{aligned}\phi :&\mathbb{K}[x] \to \mathbb{K}[x]/I\\ &f(x) \mapsto \bar{f}(x) = f(x) + I\end{aligned}$$

where $I = \langle x - H\rangle$.

We observe that it is possible to pick a cannonical representative of each coset; by the remainder theorem $\bar{f}(x) = f(H) + I$. This homomorphism partitions $\mathbb{K}[x]$ into $|\mathbb{K}|$ equivalence classes, with $f(x) \sim g(x)$ precisely when $f(H) = g(H)$.

Now, let $q(x) = q_0 + q_1 x + \dots q_r x^r \in \mathbb{K}[x]$ and $\widetilde{Q} = q_1 || \dots || q_r$, so $\widetilde{Q}$ is the concatenation of non-constant coefficients of $q(x)$.

Note that

$$\begin{aligned}\phi\left(q_0 + g_{M+\widetilde{Q}}(x)\right) &= \phi\left(q_0 + \sum_{i=1}^{r}(M_i + q_i)x^i\right)\\ &= \phi\left(\sum_{i=1}^{r} M_i x^i\right) + \phi\left(q_0 + \sum_{i=1}^{r} q_i x^i\right)\\ &= \phi\left(g_M(x)\right) + \phi\left(q(x)\right)\end{aligned}$$

Also

$$\begin{aligned}\phi\left(q_0 + g_{M+\widetilde{Q}}(x)\right) &= \phi\left(q_0 + (M_1 + q_1)x + \dots + (M_r + q_r)x^r\right)\\ &= \phi\left(q_0\right) + \phi\left(\sum_{i=1}^{r}(M_i + q_i)x^i\right)\end{aligned}$$

So

$$\phi(g_M(x)) + \phi(q(x)) = \phi(q_0) + \phi\left(\sum_{i=1}^{r}(M_i + q_i)x^i\right)$$

which is equivalent to the following statements:

$$\begin{aligned}g_M(x) + q(x) + I &= q_0 + \sum_{i=1}^{r}(M_i + q_i)x^i + I\\ \exists p(x) \in I \text{ s.t. } g_M(x) + q(x) &= q_0 + \sum_{i=1}^{r}(M_i + q_i)x^i + p(x)\\ g_M(H) + q(H) &= q_0 + \sum_{i=1}^{r}(M_i + q_i)H^i + \underbrace{p(H)}_{=0}\\ h_H(M) + (q(H) - q_0) &= h_H(M + \widetilde{Q})\end{aligned}$$

This means that if H is a root of $q(x)$ then forging with $\widetilde{Q}$ creates a predictable difference between the hash outputs. Because $T_M = E_k(N) + h_H(M)$, these predictable differences between the hash outputs will also be present between the authentication tags.

The description given at the beginning of this section is a special case in which $q(H) = q_0 = 0$.

5.2 Malleability

Saarinen [36] also described 'targeted bit forgeries' against GCM where, rather than swapping the full blocks M_i and M_{i+jt}, corresponding bits in each ciphertext block are flipped. This is a special case of the general attack, by using a multiple of $q(x)$.

If $q(H) = 0$, then $\alpha \cdot q(H) = 0$ for any $\alpha \in \mathbb{K}$ and

$$\begin{aligned} T_M &= E_k(N) + h_H(M) \\ &= E_k(N) + M_1 \cdot H + \cdots + M_m \cdot H^m \\ &= E_k(N) + (M_1 + \alpha q_1) \cdot H + \cdots + (M_m + \alpha q_m) \cdot H^m \\ &= T_{M+\alpha Q} \end{aligned}$$

where $T_{M+\alpha Q}$ is the authentication tag for the message $M_1 \oplus \alpha \cdot q_1 || \ldots || M_m \oplus \alpha \cdot q_m$ (recall that M contains the associated data, encrypted plaintext and the length of both).

If the plaintext is encrypted using a stream cipher (or a block cipher in counter mode) flipping bits in the ciphertext causes the same bits in the plaintext to be flipped. This allows us to predict relations between the original plaintext and the forged plaintext (as $C_i \oplus \alpha q_i$ decrypts to $P_i \oplus \alpha q_i$). Because α can be chosen so as to set $C_i \oplus \alpha q_i$ equal to any value chosen by the adversary (for a single i), an adversary can choose a differential between the original message and the forged message (in a single block).

If further control over the underlying plaintext is required, several forgery polynomials could be used. For example, using two forgery polynomials q_1 and q_2 an adversary can choose constants α_1 and α_2 and create the forgery $M \oplus \alpha_1 q_1 \oplus \alpha_2 q_2$. In the best case, using t polynomials permits the adversary control over t message blocks. The cost of this extra malleability is that the forgery is only successful if the authentication key is a root of the greatest common divisor of the two polynomials. This can be extended to give as much control over the plaintext as required, but for every extra malleable block the success probability is reduced by at least $\frac{1}{|\mathcal{K}_{\mathcal{H}}|}$.

If the plaintext was encrypted using a block cipher (not in counter mode) then an adversary would not have this fine control over the plaintext, but would still be able to manipulate the ciphertext in this way.

This property also permits an adversary to create as many forgeries as there are non-zero elements in the field (see [8,9,29] for further discussion of multiple forgeries).

5.3 Length Extension

In the GCM specification, the last block input to the hash function (corresponding to the term $M_1 \cdot H$ in the MAC calculation) describes the length of the plaintext and additional data. The general attack described in Section 5.2 allows an adversary to manipulate the length field (even though it does not explicitly appear in the sent message). If an adversary is given a valid tag for a message, then the content of the length field is known as it correctly encodes the length of the plaintext and additional data. It is therefore possible to choose a difference in the length field so that it corresponds to the length of the new message. In particular, forgeries can be created using high degree polynomial $q(x)$ regardless of the size of the message in the initial (message,tag) pair.

This is an important remark as it removes one significant limitation on the effectiveness of cycling attacks against GCM [36], which is the length of the message necessary to launch an attack. For a cycling attack to be attempted, an adversary requires as many blocks of correctly authenticated data as there are elements in the subgroup with which he wishes to forge, in order to swap the first and last blocks. By manipulating the length field any forgery probability can be realised starting with a valid authentication tag on a single message block.

A common criticism of GCM is that the maximum message length may be restrictive in the future as data rates increase [17]. However, it follows from our work (and the original security proofs [31]) that increasing the maximum permissible length would significantly decrease the security of the scheme.

5.4 Choosing Polynomials

To maximise the probability of a successful forgery it is important that the polynomial used to attempt a forgery has many distinct roots, as a root with multiplicities increases the degree of the polynomial (and hence the length of the attempted forgery) without increasing the probability of success. The naïve way to achieve this is to compute $q(x) = \prod_i (x - H_i)$ for as many H_i as is required to give the desired forgery probability.

Alternatively, if the polynomial defined by the hash function is evaluated in $\mathbb{F}_{p^r}$ and the irreducible factorisation of $x^{p^r} - x$ is computed in a subfield $\mathbb{F}_{p^d}$, a subset of these factors can be multiplied together (in $\mathbb{F}_{p^d}$). By choosing distinct irreducible factors, the roots of the product polynomial will be distinct. Cycling attacks [36] employ a variation on this method. The factorisation

$$2^{2^n} - 1 = \prod_{i=1}^{n} 2^{2^{i-1}} + 1$$

allows Saarinen to find factors of $x^{2^{128}} - x$ in $\mathbb{F}_2[x]$ which can be used in a cycling attack (although they are not necessarily irreducible):

$$\begin{aligned} x^{2^{128}} - x &= x(x-1)\frac{(x^3-1)}{x-1}\frac{(x^5-1)}{x-1}\frac{(x^{17}-1)}{x-1}\cdots \\ &= x(x-1)(1+x+x^2)(1+x+\cdots+x^4)(1+x+\cdots+x^{16})\ldots \end{aligned}$$

To carry out the attack using a subgroup of order t, the factors x, $(x-1)$ and $(1+x+\cdots+x^{t-1})$ are multiplied together to obtain the polynomial $x^{t+1} - x$. In general there is no requirement to select $(x-1)$ or to use only three factors, for example the polynomial $x(1+x+x^2)(1+x+\ldots x^{16})$ could be used to give a forgery probability of $\frac{19}{2^{128}}$. This is not a cycling attack, as the polynomial used contains more than two terms so the forgery does not involve simply swapping two message blocks, but it does rely on the same underlying algebraic structure.

A third option is to use a randomly selected polynomial in $\mathbb{F}_{p^r}[x]$. One potential issue with this method is the presence of repeated factors; if a factor appears more than once in the factorisation of the forgery polynomial then the degree of the forgery polynomial (and hence the length of the forged message) is increased, without improving the success probability. Square-free factorisation has been extensively studied as it is a common first step in many polynomial factorisation algorithms (for example, see [43, Ch. 14]). It may be feasible to sample polynomials from $\mathbb{F}_{p^r}[x]$ randomly and process this polynomial to make it more desirable by removing repeated factors. However, the presence of repeated factors is only a fairly small problem as the fraction of polynomials in $\mathbb{F}_{p^r}$ with a repeated factor is approximately $\frac{1}{p^r}$ [11,32].

A larger issue is that a random polynomial in $\mathbb{F}_{p^r}[x]$ does not necessarily split in $\mathbb{F}_{p^r}[x]$. Irreducible factors in $\mathbb{F}_{p^r}[x]$ do not have roots at possible hash keys and so will never evaluate to zero and hence do not increase the success probability. It is well known that the fraction of degree d polynomials that is irreducible is approximately $\frac{1}{d}$ [11,27,32], so this is not a significant problem as the forgery polynomial will probably be chosen to have a high degree in order to realise a larger success probability. However, a forgery polynomial consisting of only a few linear factors and a several high degree irreducible factors will give a low success probability and there are many polynomials in $\mathbb{F}_{p^r}[x]$ with this form. Irreducible polynomials in $\mathbb{F}_p$ that are known to have a root in $\mathbb{F}_{p^r}$ would be good candidates for attempting forgeries as the normality of $\mathbb{F}_{p^r}/\mathbb{F}_p$ guarantees that these polynomials will split into linear factors. Unfortunately this does not appear to be a well-studied area.

The main disadvantage of choosing random polynomials is that, although the roots of a polynomial in $\mathbb{K}[x]$ can be identified efficiently (see [2] for example), it would be unlikely that a non-intersecting subset of the keyspace would be used for a second forgery attempt if the first was unsuccessful. This rules out utilising the keyspace search described in Section 7.3.

6 The Algebraic Structure of Previous Attacks

In this section, we explain the relationship between the properties described in Section 5 and the known results against GCM and polynomial hash based MACs, which were described in Section 4.

6.1 Ferguson's Short Tag Attack

Ferguson's attack against GCM when short tags are used [17] begins by the adversary manipulating the message in blocks M_{2^i} (for some i). This is equivalent to attempting to forge using a linearised polynomial, that is, a polynomial $q(x) = \sum_i a_i x^{2^i}$ for which $q_j = 0$ unless $j = 2^i$ for some i. Linearised polynomials have the property that their roots form a linear subspace of the splitting field of the polynomial (see [27, Ch 3.4] for an overview). Ferguson uses polynomials in $\mathbb{F}_2[x]$ that split over $\mathbb{F}_{2^{128}}$, so the roots correspond to possible authentication keys and this guarantees that the each of the roots of the polynomial is distinct.

If the forgery is not successful then the adversary can choose a different (and distinct) subspace of the keyspace; if the forgery is successful, the adversary can choose a subspace that is contained within the original subspace. This corresponds to choosing a second forgery polynomial with either a distinct set of roots or a subset of the roots of the original forgery polynomial. Because of the structure of the roots of linearised polynomials, it is possible to describe the roots of a linearised polynomial using a matrix over $\mathbb{F}_2$. Multiple successful forgeries reduce the dimension of the subspace of the keyspace containing the authentication key, which is equivalent to reducing the number of roots of the forgery polynomial; eventually an adversary will recover the key by reducing the dimension of the subspace to zero, or equivalently by reducing the degree of the forgery polynomial to one.

6.2 Joux's Forbidden Attack

Joux's 'forbidden attack' against GCM [23] is also a specific case of the properties discussed in this paper. Joux observes that if an adversary obtains two messages that are authenticated using the same IV then they are able to derive a polynomial that is known to be satisfied by H, by finding the XOR of the authentication tags and recalling the definition of the polynomial hash. The suggestion in Joux's paper is that once one polynomial has been computed, the adversary may *'hesitate between a large number of possible H'*. The proposed solution to this issue is to compute more forgery polynomials using more pairs of messages that have been authenticated with the same IVs and then to compute the GCD of all of these. We note that the techniques described in Sections 5.2 and 7.3 can also be applied once one successful forgery polynomial has been identified.

6.3 Handschuh and Preneel

Handschuh and Preneel [20] describe a method to verify a guess for a key and a key-recovery attack.

The method for verifying a key guess $H^\star$ corresponds precisely with attempting to forge using the polynomial $x^2 - H^\star x$. A successful forgery confirms that either $H = H^\star$ or $H = 0$.

The key-recovery attack consists of attempting to create a forgery and then conducting a binary search through the roots of the polynomial defined by the difference between the original message and the forged message. As with Joux's forbidden attack, there is no reason why the techniques described in Sections 5.2 and 7.3 cannot be applied once one successful forgery polynomial has been identified. It is also possible to use the length-extension attack described in Section 5.3 to increase the forgery probability.

6.4 Saarinen's Cycling Attacks

Saarinen observed that, if a hash key H lies in a subgroup of order t, then $H^t = 1 \in \mathbb{K}$ and (for any i, j) message blocks M_i and M_{i+jt} can be swapped without changing the value of the hash.

We suggest that it is more natural and general to consider the authentication keys that fall in low order subgroups as roots of a low degree polynomial. Noting that $H^{t+1} = H$ is equivalent to $H^{t+1} - H = 0$, we can describe cycling attacks in terms of the more general attack introduced in this paper using the polynomial

$$q(x) = (M_i - M_{i+jt})(x^{t+1} - x),$$

noting that in fields of characteristic 2 subtraction is the same as $\oplus$.

This observation rephrases the example given in Section 4.4 as:

$$\begin{aligned} h_H(M_1||M_2||M_3||M_4) =& M_1 \cdot H \oplus M_2 \cdot H^2 \oplus M_3 \cdot H^3 \oplus M_4 \cdot H^4 \\ =& M_1 \cdot H \oplus M_2 \cdot H^2 \oplus M_3 \cdot H^3 \oplus M_4 \cdot H^4 \\ & \oplus (M_1 \oplus M_4) \cdot H \oplus (M_1 \oplus M_4) \cdot H^4 \\ =& M_4 \cdot H \oplus M_2 \cdot H^2 \oplus M_3 \cdot H^3 \oplus M_1 \cdot H^4 \\ =& h_H(M_4||M_2||M_3||M_1) \end{aligned}$$

Using the more general 'polynomial roots' description it is possible to forge using any subset of the keyspace, however if the authentication keys that we wish to attempt to forge with are the elements of a low order subgroup then the polynomial that is created corresponds precisely to Saarinen's cycling attack.

For example, the order three subgroup of $\mathbb{F}^\star_{2^{128}}$ (identified by Saarinen [36, Sect. 4.1]) plus the all zero key:

$H_0 =$ `00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`
$H_1 =$ `80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`
$H_2 =$ `10 D0 4D 25 F9 35 56 E6 9F 58 CE 2F 8D 03 5A 94`
$H_3 =$ `90 D0 4D 25 F9 35 56 E6 9F 58 CE 2F 8D 03 5A 94`

corresponds to the polynomial $(x - H_0)(x - H_1)(x - H_2)(x - H_3) = x^4 - x$.

7 Weak Keys for Polynomial-based Authentication Schemes

In this section we describe the existing results on weak keys for polynomial-based authentication schemes, expand the known weak key classes, and describe a general method for recovering the hash key of such a scheme.

7.1 Existing Results

Handschuh and Preneel [20] identify $H = 0$ as a weak authentication key for GCM and other similar constructions because $h_0(M) = 0$ for every message M. Following the definition given in Section 2.5 and because $|\mathcal{D}| = 1$, an adversary is not allowed to test any key by exhaustive search, nor are they allowed any verification queries. Therefore for a single element subset of the keyspace $\mathcal{D} = \{H^\star\}$ to be a weak key class, a nonce-respecting adversary must be able to identify whether or not $H = H^\star$ when they are given only (message, tag) pairs of their choosing, each created using a different IV; they may not test any key by exhaustive search or make any verification queries.

We note that it is possible for a nonce-respecting adversary to detect whether $\mathcal{D} = \{0\}$ if $|\mathsf{IV}| \neq 96$: in this case all IVs hash to give the same initial counter value and $h_0(M) = 0$ for every message M so all messages have the same authentication tag (as identified in [31, Sect. 5]). However, if $|\mathsf{IV}| = 96$ a different initial counter value is used to encrypt the output of the hash function and so, although the output of the hash function does not change, this cannot be detected given only the output of the MAC algorithm. Therefore, 0 is a weak key for GCM only if $|\mathsf{IV}| \neq 96$ and is not, in general, a weak key for polynomial evaluation based message authentication schemes.

However, the behaviour of the zero key is undesirable; the value of the authentication tag depends only on IV and not on the message. This means that, given a valid (message, tag) pair, an adversary would be able to substitute any message and still have a valid pair. This is not captured by Handshuh and Preneel's notion of weak keys, due to fact that the adversary is not permitted any verification queries. The zero key can be considered a weaker key than other keys: for any key guess it is possible to construct a forgery so that it is successful if the key guess is correct (see Section 5 and the discussion regarding cosets), but if the guess is the zero key then any forgery will be succesful if the key guess is correct.

Saarinen [36] demonstrated that the situation is much worse than described by Handschuh and Preneel, as he was able to find classes of weak keys where the authentication key falls in a low order subgroup of $\mathbb{K}^\star$. It is then possible to create a valid forgery by swapping two message blocks of a valid (message, tag) pair without changing the authentication tag if the authentication key lies in a subgroup with order dividing the distance between the swapped message blocks.

This forgery will be successful if and only if the key is an element of such a subgroup and therefore this provides a simple method for identifying weak keys which requires one valid (message, tag) pair and one verification query. These classes of weak keys therefore meet Handschuh and Preneel's definition of weak keys (given in Section 2.5).

For example, the subset of authentication keys corresponding to zero and the elements of the subgroup of order 3 in $\mathbb{F}_{2^{128}}$ is a weak key class. Membership of this subset can be confirmed by a successful forgery if M_i and M_j are swapped and $i \equiv j \mod 3$. This is equivalent to attempting a forgery using (a multiple of) the polynomial $x^4 - x$.

7.2 New Weak Key Classes

For each subset of the keyspace it is possible to construct several polynomials that will evaluate to zero on any element of that set and to a non-zero field element otherwise. By attempting to forge using one of these polynomials, a successful forgery confirms that H was in the subset of the keyspace used to define the polynomial and a failed forgery attempt confirms that the H was not in that subset. This polynomial may not have 'nice' binary coefficients like the polynomials for Saarinen's cycling attacks but instead will, in general, be an element of $\mathbb{F}_{2^{128}}[x]$; this is not problematic.

It follows that *almost every* subset of the keyspace for a polynomial evaluation based MAC is weak, regardless of the method of encryption used to form the authentication tag from the output of the hash function. Using the properties discussed in Section 5 and the observation above, it is possible to test for membership of any subset of the keyspace using at most two verification queries. Membership of a subset $\mathcal{D}$ that includes the zero key can be tested by setting $q(x) = \prod_{H \in \mathcal{D}} (x - H)$. This therefore requires one verification query, independent of the size of $\mathcal{D}$. To test for membership of a subset $\mathcal{D}$ that does not include zero, first test whether $H \in \mathcal{D} \cup \{0\}$ and then rule out $H = 0$ using the method described below. This therefore requires two verification queries, but again is independent of the size of $\mathcal{D}$.

The distinction between subsets including zero or not including zero is a consequence of the constant term of $g_M(x)$ being zero to avoid predictable changes in the output of the hash from flipping low order bits. Therefore, using Handschuh and Preneel's definition, a set $\mathcal{D}$ of hash keys for a universal hash based authentication scheme is a weak key class if either: $|\mathcal{D}| \geq 3$ or $|\mathcal{D}| \geq 2$ and $0 \in \mathcal{D}$, regardless of how $h_H(M)$ is encrypted to form T.

If the encryption is performed additively (as it is in GCM), then it is possible to extend this result to include every subset $\mathcal{D}$ with $|\mathcal{D}| \geq 2$, using the observation in Section 5 regarding the quotient ring $\mathbb{K}[x]/\langle x - H \rangle$. This has also been noted independently by Zhu, Tan, and Gong [46].

Given one valid (message, tag) pair for a single block message and one verification query it is easy to determine whether or not $H = 0$. If the adversary attempts to forge using any other single block message and the same tag, then the forgery is successful if and only if $H = 0$ as seen below.

If no length encoding is used:

$$\begin{aligned} T &= E(\mathsf{CTR}_0) + (M \cdot H) \\ &= E(\mathsf{CTR}_0) + (M' \cdot H) \\ &\Leftrightarrow (M - M') \cdot H = 0 \\ &\Leftrightarrow M = M' \text{ or } H = 0 \end{aligned}$$

If a GCM style length encoding is used:

$$\begin{aligned} T &= E(\mathsf{CTR}_0) + (\text{length} \cdot H) + (M \cdot H^2) \\ &= E(\mathsf{CTR}_0) + (\text{length} \cdot H) + (M' \cdot H^2) \\ &\Leftrightarrow (M - M') \cdot H^2 = 0 \\ &\Leftrightarrow M = M' \text{ or } H = 0 \end{aligned}$$

We discuss this issue further in Section 10.

7.3 Key Recovery

Saarinen suggests that once a weak key has been identified (by a successful cycling attack), the adversary would create many forgeries by further cycling attacks [36, Sect. 9]. Translating this to the more general polynomial root description: once a successful forgery occurs, the authentication key is known to be one of the roots of the 'forgery polynomial' $q(x)$. Therefore rather than making repeated 'cycling forgeries' with guaranteed success but limited control of the plaintext, the adversary can aim to recover the authentication key and forge authentication tags for arbitrary messages. By attempting to forge using a subset of the roots of the forgery polynomial (and reducing the number of roots in the subset after each successful attempt), an adversary can gradually recover the authentication key using a method that is independent of encryption method or key used. This would give a forgery probability less than 1 at each stage, however the adversary can choose a trade-off between the forgery probability and the speed of recovering the authentication key. This is analogous to the key recovery attack described by Handschuh and Preneel [20] (where the subsets are chosen to realise a binary search of the keyspace).

By testing for membership of subsets of the keyspace, it is plausible that an adversary could recover one bit of the authentication key with each forgery attempt. If $q(x) = \prod_{H \in \mathcal{Y}} (x - H)$, where $\mathcal{Y}$ is the set of authentication keys for which the first bit is zero, then a successful forgery confirms that the first bit of the authentication key is zero and a failure confirms that the first bit is one. Repeating this for each bit of the authentication key, the whole key could be recovered using 128 verification queries.

This would require unfeasibly large messages to be used in the forgery attempts in the case of authentication keys corresponding to elements of a field with $|\mathbb{K}| \approx 2^{128}$, but it is a strong argument against using a hash function based on polynomial evaluation in a field with $|\mathbb{K}| \ll 2^{128}$. This may be a direction taken by variants of GCM designed to improve performance (see [1] for one such example and Section 8 for an analysis of this scheme), however we recommend extreme caution when considering these modifications. In the case of GCM the size of the subsets that can be tested is limited to around 2^{56} as the maximum message length is limited.

One advantage of being able to test for membership of arbitrary subsets is that it allows the adversary to use any partial knowledge of the authentication key that they may have. Note that in the case of GCM, recovery of the hash key H does not lead to the recovery of the encryption key k as $H = E_k(0)$.

8 GCM with Short Multiplications

In 2012, Yasuda and Aoki [1] proposed GCM/2^+, a variant of GCM that evaluates the hash function using 'short' multiplications in $\mathbb{F}_{2^{64}}$ rather than multiplications in $\mathbb{F}_{2^{128}}$. The motivation for this change is to increase the efficiency in software, where $\mathbb{F}_{2^{128}}$ multiplications are significantly more expensive than $\mathbb{F}_{2^{64}}$ multiplications. However this change to the specification makes the attacks in Sections 5 and 7 much more efficient, as described below.

The most significant difference from the GCM specification relates to the polynomial $\overline{g}$ that is used to define the hash function $\overline{h}$ in GCM/2^+. The hash key H is split into two 'half keys' $H = L||R$, where $|L| = |R| = \frac{n}{2}$ and each message block is considered as two 'half blocks' $M_i = M_i^{(L)}||M_i^{(R)}$. We will use $M^{(L)}$ to denote $M_1^{(L)}||\ldots||M_m^{(L)}$ and similarly for $M^{(R)}$. Of particular relevance is the message block that encodes the length of the message. The length of the additional authenticated data is encoded in $M_1^{(L)}$ and the length of the ciphertext is encoded in $M_1^{(R)}$. This is identical to the GCM specification but is noteworthy because GCM/2^+ carries out all operations on half blocks.

The hash function is evaluated in two halves, $\overline{h_H} = h_L||h_R$, and

$$\begin{aligned}\overline{h_H}(M) &= \overline{g_M}(H)\\ &= h_L(M^{(L)})||h_R(M^{(R)})\\ &= g_{M^{(L)}}(L)||g_{M^{(R)}}(R)\end{aligned}$$

where $g_{M^{(\cdot)}}(\cdot)$ is evaluated in $\mathbb{F}_{2^{n/2}}$. The key remark at this point is that $\overline{h_H}$ is simply the concatenation of the evaluation of two polynomials, $g_{M^{(L)}}$ and $g_{M^{(R)}}$.

GCM/2^+ also makes the following changes to the GCM specification:

Block size: GCM/2^+ supports the use of a block cipher with any block size (denoted by n).

Tag Encryption: An extra block cipher call is added to the end of the GCM authentication tag generation algorithm. The authentication tag is computed as $T = E_k(\overline{h_H}(M) \oplus E_k(\mathsf{CTR}_0))$.

Final Multiplication: There is no final multiplication by the authentication key in the evaluation of the hash function. The hash function polynomial is $\overline{g_M}(H) = M_m H^{m-1} + \ldots + M_2 H + M_1$, rather than $\overline{g_M}(H) = M_m H^m + \ldots + M_2 H^2 + M_1 H$. The requirement for the constant term of $g_M(x)$ to be zero (in order to prevent the predictable bit flipping, as described in Section 5) can be relaxed due to the introduction of the tag encryption.

We will consider the half-sized multiplications and the removal of the final multiplication below. The support for other block sizes need not be considered as all of the results in this paper are independent of the block size of the block cipher. The introduction of an extra block cipher call slightly affects our results. We now require a full collision and hence a forgery polynomial with $q_0 = 0$ and $q(H) = 0$; we are unable to use the general technique described in Section 5 that allows us to utilise any polynomial. We also note that the specification of GCM/2^+ makes no recommendations regarding maximum message length.

The hash function consists of the concatenation of two polynomial hash functions (each evaluated in $\mathbb{F}_{2^{n/2}}$). There is no interaction between the two sides of the computation until the output is encrypted with the block cipher. We may, therefore, choose forgery polynomials $q^{(L)}(x)$ and $q^{(R)}(x)$ for the two polynomial hash functions, as described in Section 5. The forgery will be successful if $q^{(L)}(L) = q^{(R)}(R) = 0$. However, as there is no interaction between the two sides of the $\overline{g_M}(H)$, we can choose (without loss of generality) $q^{(R)} \equiv 0$. In this case, we have reduced finding a collision for $\overline{h_H}$ to finding hash collisions for h_L.

If we set $q^{(R)} \equiv 0$ then we are unable to alter the right half of any message block, in particular $M_1^{(R)}$. In this case we cannot change the length of the ciphertext, but still have total control over the length of the additional authenticated data (provided that $q^{(L)} \not\equiv 0$). Similarly, we could choose $q^{(L)} \equiv 0$ and lose the ability to increase the length of the additional authenticated data, while retaining control of the ciphertext length. In the GCM specification the maximum length of the additional authenticated data is significantly larger than the maximum length of the ciphertext and so, if this is mimicked in the specification of GCM/2^+, setting $q^{(R)} \equiv 0$ does not significantly reduce the potential for a length extension attack.

We note that it is possible to attack both L and R simultaneously, by choosing both $q^{(R)} \not\equiv 0$ and $q^{(L)} \not\equiv 0$. However, this forgery will only be successful if $q^{(L)}(L) = q^{(R)}(R) = 0$. As the left and right components of the hash function behave independently, the overall success probability is simply the product of the success probability for the components. Choosing $q^{(R)} \equiv 0$ results in a success probability of 1 for the right component, which allows the adversary to identify a subset containing L, without any chance of the forgery failing due to the right half key. This leads to a faster key recovery than attacking both halves simultaneously; if an adversary can try every half key by using t queries, then they can cover the entire keyspace with $2t$ queries by attacking the two halves separately whereas t^2 queries are required to cover the keyspace if both halves are attacked together.

If the maximum length of the message is close to 2^{64} blocks then we can achieve a significant success probability. Every half key is a root of $x^{2^{64}} - x$ and therefore if messages of at least 2^{64} blocks are permitted a forgery can be made with probability $1 - \frac{1}{2^{64}}$ given a single valid (message, tag) pair. This probability is not quite one, as we may need to manipulate the length field (the constant term in each polynomial). It is not possible to do this using $q(x) = x^{2^{64}} - x$, so we will use $x^{2^{64}-1} - 1$. This forgery will fail only if the half key is zero. Alternatively, an adversary could recover a half key with one valid (message, tag) pair and 65 verification queries by utilising a binary search (as described in Section 7.3 and [20]), with an additional query at the end to decide whether or not $H = 0$.

Not permitting 2^{64}-block messages prevents this highly efficient attack, but similar attacks are still possible. If m-block messages are permitted, it is possible to recover the key with at most $2^{64-\log m} + \log m + 1$ verification queries. In this case, m keys can be tested with each verification query, so we will partition the keyspace into $2^{64-\log m}$ subsets of size m. The attack begins by using up to $2^{64-\log m}$ verification queries to establish which subset contains the authentication key. We then conduct a binary search of the appropriate partition, requiring $\log m$ queries and finally use one more query to establish whether or not the half key is zero. This demonstrates that the attack remains feasible if $\log m$ is not much smaller than 64.

For example, if 2^{56}-block messages are permitted (as is the case for GCM) then 2^{56} keys can be tested with a single verification query. By partitioning the keyspace into 2^8 sets it is possible to identify which of these sets contains the authentication key using no more than 2^8 verification queries and a binary search of the relevant set. In this case, at most $2^8+56+1 \approx 315$ verification queries are required to recover a half key.

We also remark that the original GCM proposal [30] includes an appendix describing GCM with a 64-bit block cipher. In this case the polynomial evaluation is computed in $\mathbb{F}_{2^{64}}$. This leads to exactly the same problem as described above and the (full) authentication key can be recovered using approximately 315 verification queries.

The attacks in this section highlight the relationship between the field size, maximum message length, and the forgery probability or speed of key recovery. We recommend against the use of GCM/2^+ as, even if the maximum message length is a single block, it offers a worse security guarantee than GCM.

9 Weak Keys for Square Hash

Square Hash was proposed by Etzel et al. in 1999 [16]. It is a universal hash function family, $\mathcal{H} = \{h_k : \mathbb{Z}_p^m \to \mathbb{Z}_p \mid k \in \mathcal{K}_{\mathcal{H}}\}$, and is based on MMH [19]. For each $k, M \in \mathbb{Z}_p^m$, h_k is defined by

$$h_k(M) = \sum_{i=1}^{m} (M_i + k_i)^2 \mod p$$

This is a fundamentally different construction than those considered so far in this paper, however we will assume that Square Hash is being used in a MAC algorithm, as described in Section 2.2, and identify classes of weak keys for the hash component of the MAC algorithm. That is, we will demonstrate sets $\mathcal{D}$ for which it is possible to assert whether the hash key is an element of that set, using fewer than $|\mathcal{D}|$ verification queries and fewer than $|\mathcal{D}|$ generation queries.

Handschuh and Preneel [20] have identified a number of weak keys for Square Hash, in particular those keys with $k_i = k_j$ for some i and j, which can be identified by a successful forgery when M_i and M_j are swapped. We demonstrate that every Square Hash key is an element of several weak key classes of the form: $\mathcal{D}_{j,\mu}^{i,\lambda} = \{k \in \mathbb{Z}_p^m | \lambda k_i = \mu k_j\}$, where $\lambda, \mu \in \mathbb{Z}_p$. We assume that an adversary can ask for a message of their choice to be authenticated and then aims to forge using a different message but the same authentication tag. All of the queries that our adversary will ask consist of just two message blocks. We will use $M_1||M_2$ to represent the message sent to the MAC generation oracle, and $M_1'||M_2'$ to represent the message sent to the verification oracle (with the authentication tag that is valid for $M_1||M_2$). The results can be trivially extended to messages consisting of several blocks provided that $M_r = M_r'$ for all $r \neq i, j$ and analogous methods can be applied to identify $\mathcal{D}_{j,\mu}^{i,\lambda}$ for any $i \neq j$ with $i, j \leq m$.

The key observation is that:

$$\begin{aligned} h_k(M) &= \sum_{i=1}^{m} (M_i + k_i)^2 \mod p \\ &= \sum_{i=1}^{m} M_i^2 + 2\sum_{i=1}^{m} (M_i \cdot k_i) + \sum_{i=1}^{m} k_i^2 \mod p. \end{aligned}$$

So for a fixed key, it is possible to find a hash collision (and hence a MAC forgery) if it is possible to find two messages M and M' that meet the following two conditions:

Condition 1

$$\sum_{i=1}^{m} M_i'^2 = \sum_{i=1}^{m} M_i'^2 \mod p$$

Condition 2

$$\sum_{i=1}^{m} (M_i \cdot k_i) = \sum_{i=1}^{m} (M_i' \cdot k_i) \mod p$$

It is possible to identify whether or not a particular relationship holds between two key blocks (e.g. $\lambda k_i = \mu k_j$ for some $\lambda, \mu \in \mathbb{Z}_p$) using one MAC generation and one MAC verification query. As $|\mathcal{D}_{j,\mu}^{i,\lambda}| > 1$ for every $i, j \in \{1, \ldots, m\}$, $\lambda, \mu \in \mathbb{Z}_p$, these are weak key classes for Square Hash. We now describe a method for identifying the two messages required to determine whether $k \in \mathcal{D}_{2,\mu}^{1,\lambda}$ in the case where the message consists of only two blocks. This is a specific case of a method that uses techniques from [28].

Condition 1 described above can be satisfied by choosing pairs of messages (M_1, M_2), $(0, M_2')$ such that $M_1^2 + M_2^2 = M_2'^2$. We will test for the class of weak keys $\mathcal{D}_{2,\mu}^{1,\lambda}$ where $\lambda, \mu \neq 0$ using a well-known formula of Euclid.

We set:

$$\begin{aligned} M_1 &= 2\lambda\mu & M_1' &= 0 \\ M_2 &= \lambda^2 - \mu^2 & M_2' &= \lambda^2 + \mu^2 \end{aligned}$$

Then:

$$\begin{aligned} M_1^2 + M_2^2 &= (4\lambda^2\mu^2) + (\lambda^4 - 2\lambda^2\mu^2 + \mu^4) \\ &= \lambda^4 + 2\lambda^2\mu^4 + \mu^4 \\ &= (\lambda^2 + \mu^2)^2 \\ &= M_2'^2 \end{aligned}$$

Also:

$$\begin{aligned} k_1M_1 + k_2M_2 &= k_1(2\lambda\mu) + k_2(\lambda^2 - \mu^2) \\ &= 2k_1\lambda\mu + \frac{\lambda k_1}{\mu}(\lambda^2 - \mu^2) \\ &= k_1(\frac{\lambda^3}{\mu} + \lambda\mu) \\ &= \frac{k_1\lambda}{\mu}(\lambda^2 + \mu^2) \\ &= k_2M_2' \end{aligned}$$

The hash of (M_1, M_2) is equal to the hash of (M'_1, M'_2) and therefore the MAC forgery is successful if $k \in \mathcal{D}^{1,\lambda}_{2,\mu}$. It can easily be seen that a successful MAC forgery is in fact both necessary and sufficient for $k \in \mathcal{D}^{1,\lambda}_{2,\mu}$. Using this technique it is possible to recover the relationship between any k_i and k_j using no more than 1 valid (message, tag) pair and p verification queries.

10 Discussions and Conclusions

10.1 Choice of Fields

It is true that the security against cycling attacks, as presented in [36], can be increased by evaluating a hash function in a field with a multiplicative group, the order of which does not have many factors. However the attack introduced in this paper (of which cycling attacks are a special case) applies equally well in any finite field, so the claim that *'The security of polynomial-evaluation MACs against attacks of this type of attack can be determined from the factorization of the group size in a straightforward manner'* [36, Sect. 8] is somewhat misleading.

Saarinen's claim is valid in the sense that the factorisation of $|\mathbb{K}| - 1$ determines the extent to which the process of computing irreducible factors will succeed; however an attack using $\prod_{H \in \mathcal{D}} (x - H)$ will work equally well in every field. In particular, it follows from our work that the SGCM variant of GCM has the same inherent weaknesses regarding polynomial based forgery attacks.

The size of the field has another important implication to the security of a scheme, as demonstrated in Section 8. It is therefore important to choose an appropriately large field in which to evaluate the polynomial, or to employ some other mechanism to ensure that multiple copies of a small field do not behave independently (as in GCM/2^+ [1]).

10.2 Length Extension

It is unfortunate that including the length of the additional authenticated data and plaintext in the input to the hash function is not sufficient to prevent the length extension attack presented in this paper. In schemes that use a GCM–like length encoding, if the value of the length field were encrypted using a block cipher before being input to the hash function, it would not be possible to alter the message length as described in Section 5. However, one of the design goals of GCM was to take advantage of AES pipelining, which precludes the use of the block cipher to compute the authentication tag.

10.3 Malleability

Part of the reason that the algebraic structure of polynomial hashing is problematic for GCM is that it allows an adversary to choose the changes that are made to the plaintext in a forged message. This is because addition in a field of characteristic 2 is used for both the counter mode encryption and the hash function evaluation.

One way to avoid this issue is to use different operations during encryption and MAC generation. This is one advantage that the combination of CTR & Poly1305–AES [5] has over GCM, as in this scheme the MAC is computed using addition in a prime order field while the message is encrypted using addition in a field of characteristic 2.

An alternative method to increase the difficulty for an adversary attempting to make meaningful manipulations of plaintext is to use a mode of operation other than CTR as this will prevent the 'targeted bit forgeries' [36, Sect. 6] and the analogous forgeries in this paper.

GCM roughly follows the Encrypt–then–MAC paradigm, as is generally perceived to be best practice (although MAC–then–Encrypt has also been proved secure in the nonce-based AEAD setting [34]). Despite going against the perceived best practice, using a MAC–then–Encrypt approach (in addition to the proposed changes described above) would make it harder for an adversary to create ciphertexts that correctly decrypt to a plaintext known to be related to a (plaintext,ciphertext) pair obtained from a query. We note that we have not analysed this construction and that the introduction of other weaknesses caused by making these changes has not been ruled out.

10.4 Weak Keys

The weak key classes that are identified in Section 7 cause the forgery probability to be higher than expected because an adversary can detect whether the authentication key that is being used is a member of that class and can then forge with probability one.

The broader issue with polynomial evaluation based hash functions is that it is possible to test for membership of large subsets of the keyspace with only one or two verification queries and once an adversary has successfully confirmed membership of a subset they can either continue to forge messages or conduct a search of a much reduced keyspace. This is an unusual and undesirable property of a cryptosystem.

It is interesting that the two-element subsets of the keyspace containing zero are always weak key classes, whereas two-element subsets not containing zero may be weak depending on how the output of the hash function is encrypted.

This perhaps suggests a problem with the definition of a weak key class. In our opinion the definition is correct; the observations made in this paper are unavoidable properties of hash functions based on polynomial evaluation that result from the algebraic structure of the construction. The distinction between the two methods to encrypt the output of the hash function arises from the use of the same algebraic structure to encrypt additively and the fact that the application of a block cipher removes this structure so requires a full collision. These results are therefore not best described in terms of the number of weak keys.

The most important discussion around this issue is whether an algorithm in which almost every subset of the keyspace is a weak key class is a weak algorithm or whether this is a property of the construction that, although highly undesirable, is not considered to reduce the security of the scheme to an unacceptable level. We suggest that, in the case of GCM, it is the latter; in other polynomial-based MAC schemes with different parameters it may be the former and this property must be considered when designing and evaluating schemes.

Acknowledgments

We would like to thank Jean Paul Degabriele for the extensive and insightful conversations during the early stages of this work and also Bertfried Fauser for helpful comments regarding the extension to include all two-element subsets as weak key classes. We also thank the anonymous reviewers from FSE 2013 and the Journal of Cryptology.

The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

References

1. Kazumaro Aoki and Kan Yasuda. The Security and Performance of "GCM" when Short Multiplications Are Used Instead. In Mirosław Kutyłowski and Moti Yung, editors, *Information Security and Cryptology*, volume 7763 of *Lecture Notes in Computer Science*, pages 225–245. Springer Berlin Heidelberg, 2013.
2. E. R. Berlekamp. Factoring Polynomials Over Large Finite Fields. *Math. Comp.*, 24:713–735, 1970.
3. D. J. Bernstein. The Poly1305-AES message-authentication code. Slides from FSE, 2005. `http://cr.yp.to/talks/2005.02.21-1/slides.pdf`.

4. Daniel J. Bernstein. Stronger Security Bounds for Wegman-Carter-Shoup Authenticators. In Ronald Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 164–180. Springer Berlin Heidelberg, 2005.
5. Daniel J. Bernstein. The Poly1305-AES Message-Authentication Code. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer Berlin Heidelberg, 2005.
6. Jürgen Bierbrauer, Thomas Johansson, Gregory Kabatianskii, and Ben Smeets. On Families of Hash Functions via Geometric Codes and Concatenation. In Douglas R. Stinson, editor, *Advances in Cryptology CRYPTO 93*, volume 773 of *Lecture Notes in Computer Science*, pages 331–342. Springer Berlin Heidelberg, 1994.
7. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and Secure Message Authentication. In Michael Wiener, editor, *Advances in Cryptology CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer Berlin Heidelberg, 1999. Full version, available at `http://www.cs.ucdavis.edu/~rogaway/papers/umac`.
8. John Black and Martin Cochran. MAC Reforgeability. Cryptology ePrint Archive, Report 2006/095, 2006.
9. John Black and Martin Cochran. MAC Reforgeability. In Orr Dunkelman, editor, *Fast Software Encryption*, volume 5665 of *Lecture Notes in Computer Science*, pages 345–362. Springer Berlin Heidelberg, 2009.
10. Gilles Brassard. On Computationally Secure Authentication Tags Requiring Short Secret Shared Keys. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology*, pages 79–86. Springer US, 1983.
11. Leonard Carlitz. The Arithmetic of Polynomials in a Galois Field. *Proceedings of the National Academy of Sciences*, 17(2):120–122, 1931.
12. J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions (extended abstract). In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, pages 106–112, New York, NY, USA, 1977. ACM.
13. J. Lawrence Carter and Mark N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.
14. B. den Boer. A simple and key-economical unconditional authentication scheme. *Journal of Computer Security*, 2:65–72, 1993.
15. Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, `http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf`, Nov 2007.
16. Mark Etzel, Sarvar Patel, and Zulfikar Ramzan. Square Hash: Fast Message Authentication via Optimized Universal Hash Functions. In Michael Wiener, editor, *Advances in Cryptology CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 234–251. Springer Berlin Heidelberg, 1999.
17. Niels Ferguson. Authentication weaknesses in GCM. Comments submitted to NIST Modes of Operation Process, `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf`, 2005.
18. E. N. Gilbert, F. J. MacWilliams, and N. J. A. Sloane. Codes Which Detect Deception. Technical Report 3, Bell Sys. Tech. J., Mar 1974.
19. Shai Halevi and Hugo Krawczyk. MMH: Software message authentication in the Gbit/second rates. In Eli Biham, editor, *Fast Software Encryption*, volume 1267 of *Lecture Notes in Computer Science*, pages 172–189. Springer Berlin Heidelberg, 1997.
20. Helena Handschuh and Bart Preneel. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In David Wagner, editor, *Advances in Cryptology CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 144–161. Springer Berlin Heidelberg, 2008.
21. K. Igoe and J. Solinas. AES Galois Counter Mode for the Secure Shell Transport Layer Protocol. IETF Request for Comments 5647, `http://tools.ietf.org/html/rfc5647`, 2009.
22. Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and Repairing GCM Security Proofs. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 31–49. Springer Berlin Heidelberg, 2012.
23. Antoine Joux. Authentication Failures in NIST version of GCM. Comments submitted to NIST Modes of Operation Process, `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/GCM/Joux_comments.pdf`, 2006.
24. Tadayoshi Kohno, John Viega, and Doug Whiting. CWC: A High-Performance Conventional Authenticated Encryption Mode. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption*, volume 3017 of *Lecture Notes in Computer Science*, pages 408–426. Springer Berlin Heidelberg, 2004.
25. Hugo Krawczyk. LFSR-based Hashing and Authentication. In Yvo G. Desmedt, editor, *Advances in Cryptology CRYPTO 94*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer Berlin Heidelberg, 1994.
26. L. Law and J. Solinas. Suite B Cryptographic Suites for IPsec. IETF Request for Comments 6379, `http://tools.ietf.org/html/rfc6379`, 2011.

27. Rudolf Lidl and Harald Niederreiter. *Finite Fields*, volume 20 of *Encylopedia of Mathematics and its Applications.* Cambridge University Press, 2nd edition, 1997.
28. David J.C. MacKay and Sanjoy Mahajan. Numbers that are Sums of Squares in Several Ways. `http://www.cs.toronto.edu/~mackay/sumsquares.pdf`, 2001.
29. David A. McGrew and Scott R. Fluhrer. Multiple forgery attacks against Message Authentication Codes. Comments submitted to NIST on the Choice Between CWC or GCM, `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/multi-forge-01.pdf`, 2005.
30. David A. McGrew and John Viega. The Galois/Counter Mode of Operation (GCM). Submission to NIST Modes of Operation Process, `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-revised-spec.pdf`, May 2005.
31. David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In Anne Canteaut and Kapaleeswaran Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer Berlin Heidelberg, 2005.
32. Daniel Panario. What Do Random Polynomials over Finite Fields Look Like? In Gary L. Mullen, Alain Poli, and Henning Stichtenoth, editors, *Finite Fields and Applications*, volume 2948 of *Lecture Notes in Computer Science*, pages 89–108. Springer Berlin Heidelberg, 2004.
33. M. O. Rabin. Fingerprinting by random polynomials. Center for Research in Computing Technology, Harvard University, Technical Report TR-15-81, 1981.
34. Phillip Rogaway. Authenticated-Encryption with Associated-Data. In V. Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 98–107. ACM, 2002.
35. Markku-Juhani O. Saarinen. SGCM: The Sophie Germain Counter Mode. Cryptology ePrint Archive, Report 2011/326, 2011.
36. Markku-Juhani Olavi Saarinen. Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes. In Anne Canteaut, editor, *Fast Software Encryption*, volume 7549 of *Lecture Notes in Computer Science*, pages 216–225. Springer Berlin Heidelberg, 2012.
37. M. Salter and R. Housley. Suite B Profile for Transport Layer Security (TLS). IETF Request for Comments 6460, `http://tools.ietf.org/html/rfc6460`, 2011.
38. Victor Shoup. On Fast and Provably Secure Message Authentication Based on Universal Hashing. In Neal Koblitz, editor, *Advances in Cryptology CRYPTO 96*, volume 1109 of *Lecture Notes in Computer Science*, pages 313–328. Springer Berlin Heidelberg, 1996.
39. Gustavus J. Simmons, editor. *Contemporary Cryptology: The Science of Information Integrity.* IEEE Press, 1992.
40. D. R. Stinson. On the Connections Between Universal Hashing, Combinatorial Designs and Error-Correcting Codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(52), 1995.
41. Douglas R. Stinson. Universal hashing and authentication codes. *Designs, Codes and Cryptography*, 4(3):369–380, 1994.
42. Richard Taylor. Near optimal unconditionally secure authentication. In Alfredo Santis, editor, *Advances in Cryptology EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 244–253. Springer Berlin Heidelberg, 1995.
43. Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, Cambridge, 2nd edition, 2003.
44. Mark N. Wegman and J. Lawrence Carter. New classes and applications of hash functions. In *Foundations of Computer Science, 20th Annual Symposium on*, pages 175–182, 1979.
45. Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.
46. Bo Zhu, Yin Tan, and Guang Gong. Revisiting MAC Forgeries, Weak Keys and Provable Security of Galois/Counter Mode of Operation. In Michel Abdalla, Cristina Nita-Rotaru, and Ricardo Dahab, editors, *Cryptology and Network Security*, volume 8257 of *Lecture Notes in Computer Science*, pages 20–38. Springer International Publishing, 2013.