Polynomial evaluation and message authentication

Daniel J. Bernstein

Abstract. The cryptographic literature contains many provably secure high-speed authenticators. Some authenticators use n multiplications for length-n messages; some authenticators have the advantage of using only about $n/2$ multiplications. Some authenticators use n variables for length-n messages; some authenticators have the advantage of using only 1 variable. This paper, after reviewing relevant polynomial-evaluation algorithms, introduces the first authenticator that combines these advantages.

1. Introduction

Fix a finite field k. How quickly can we evaluate a polynomial f over k?

Let's focus on **polynomial algebraic algorithms**: i.e., chains of additions, subtractions, and multiplications. What is the smallest number of multiplications in a polynomial algebraic algorithm that evaluates f? What is the smallest total number of operations? What is the smallest number of operations with reasonable weights for the costs of additions, subtractions, and multiplications? What is the smallest number of cycles on an Intel Core 2 Duo?

Let's focus on the simplest cost measure, the number of multiplications. The most popular method of evaluating polynomials is "Horner's rule," a family of n-multiplication polynomial algebraic algorithms that evaluate every univariate polynomial of degree n.

It is easy to see that Horner's rule is not always optimal: for example, x^{1024} can be evaluated with just 10 squarings. A classic observation is that Horner's rule is essentially *never* optimal: for *every* univariate polynomial f of degree n there is a polynomial algebraic algorithm that evaluates f using only about $n/2$ multiplications. The original constructions (from 1955 Motzkin [**24**], 1958 Belaga [**2**], et al.) assumed complex coefficients and could not handle most polynomials over a finite field k; but Rabin and Winograd in [**26**] presented various families of polynomial algebraic algorithms that evaluate every univariate polynomial of degree n over k using only about $n/2$ multiplications.

2000 *Mathematics Subject Classification.* Primary 94A62. Secondary 68W30.

The author was supported by the National Science Foundation under grant ITR–0716498. He carried out parts of this work while visiting Technische Universiteit Eindhoven. Permanent ID of this document: `b1ef3f2d385a926123e1517392e20f8c`. Date: 2007.10.22.

1.1. Contents of this paper. Let's change the problem: instead of mapping a family of polynomial algebraic algorithms *surjectively* to polynomials, let's map a family of polynomial algebraic algorithms *injectively* to polynomials. Can we have n parameters in a family of m-multiplication polynomial algebraic algorithms that compute *distinct* low-degree polynomials?

One way to see the impact of the change from surjectivity to injectivity is to consider the same question for polynomials in more variables. Increasing the number of variables produces a larger space of low-degree polynomials, making surjectivity harder to achieve but making injectivity easier to achieve.

This question turns out to be directly relevant to high-speed cryptography, specifically high-speed computation of *authenticators* that protect messages against forgery. This paper explains the connection; explains the answers that are implicit in three state-of-the-art secret-key message-authentication algorithms; and explains a better answer, producing a faster message-authentication algorithm.

Specifically:

- Section 2 discusses n-variable n-multiplication authenticators: e.g.,
  - 1974 Gilbert/MacWilliams/Sloane [**13**] and
  - 1997 Halevi/Krawczyk "MMH" [**15**].

  These authenticators use the obvious n-multiplication algorithm for dot products.
- Section 3 discusses n-variable $(n/2)$-multiplication authenticators: e.g.,
  - unpublished Carter/Wegman "NMH*" discussed in [**15**, Section 5],
  - 1997 Halevi/Krawczyk "NMH" [**15**, Section 5],
  - 1999 Black/Halevi/Krawczyk/Krovetz/Rogaway "UMAC" [**8**],
  - 2005 Boesgaard/Scavenius/Pedersen/Christensen/Zenner "Rabbit" [**9**], and
  - 2007 Krovetz "VMAC" [**21**].

  These authenticators use a pseudo-dot-product algorithm introduced by Winograd in 1968 to speed up matrix multiplication.
- Section 4 discusses 1-variable n-multiplication authenticators: e.g.,
  - 1993 den Boer [**12**],
  - 1994 Johansson/Kabatianskii/Smeets [**18**],
  - 1994 Taylor [**33**],
  - 1996 Shoup [**29**],
  - 1999 Nevelsteen/Preneel [**25**],
  - 1999 Bernstein "hash127" [**3**],
  - 2000 Krovetz/Rogaway "PolyR" [**22**],
  - 2002 Bernstein [**4**],
  - 2004 Kohno/Viega/Whiting "CWC" [**20**],
  - 2004 McGrew/Viega "GCM" [**23**], and
  - 2005 Bernstein "Poly1305-AES" [**5**].

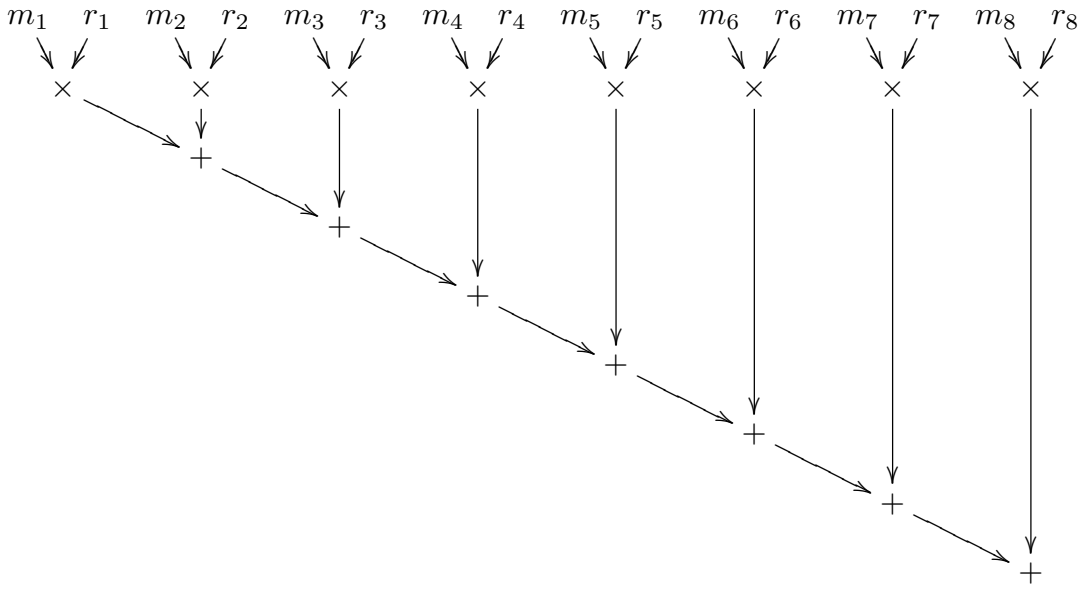
  These authenticators use Horner's n-multiplication rule for polynomial evaluation.
- Section 5 introduces new 1-variable $(n/2)$-multiplication authenticators. As far as I can tell, the ideas of Motzkin, Belaga, et al. are useless here; my authenticators instead use a tweaked version of an algorithm published by Rabin and Winograd in [**26**, Corollary 13].

The number of multiplications in these polynomial-evaluation algorithms has an obvious effect on cryptographic speed. The number of variables has a less obvious, but often much larger, effect on cryptographic speed, as discussed in Section 3. My new authenticators combine the advantages of a small number of multiplications and a minimal number of variables.

Probably the injectivity question has been asked before outside the context of cryptography. Perhaps the connection to cryptography has been pointed out before. But it is clear that the connection has not been sufficiently exploited: my new authenticator is faster than anything in the literature.

2. Linear polynomial evaluation and message authentication

2.1. Dot products. The most obvious way to compute a dot product $m_1r_1 + m_2r_2 + \cdots + m_8r_8$ is to multiply each m_i by r_i and then sequentially add the results, using 8 multiplications and 7 additions, as shown in the following diagram:



Each node in this diagram computes a polynomial according to the following rules:

- The node labelled "m_1" computes m_1; the node labelled "r_1" computes r_1; similarly for $m_2, r_2, \ldots$.
- Each node labelled "×" computes the product of its two predecessor nodes. For example, the leftmost node labelled "×" computes m_1r_1.
- Each node labelled "+" computes the sum of its two predecessor nodes. For example, the leftmost node labelled "+" computes $m_1r_1 + m_2r_2$.

Evidently the final node computes the desired dot product $m_1r_1 + \cdots + m_8r_8$.

More generally, the obvious chain of n multiplications and $n - 1$ additions computes an n-term dot product.

2.2. A secure message-authentication code. Messages are not safe while they are in transit through a public network such as the Internet. Attackers can inspect messages, block messages from reaching their destination, and insert forged messages of their choice. How do we distinguish forged messages from legitimate messages? Here is one good answer.

Fix a finite field k. The legitimate sender generates 9 independent uniform random secrets $r_1, r_2, \ldots, r_8, s \in k$. The sender meets the legitimate receiver in private and tells the receiver the same secrets $r_1, r_2, \ldots, r_8, s$.

Later the sender wants to send a message m to the receiver through a public network, where m is a sequence of 8 components $m_1, m_2, \ldots, m_8 \in k$. The sender transmits m together with the **authenticator** $a = m_1r_1 + \cdots + m_8r_8 + s \in k$.

The receiver receives (m', a') from the network. Perhaps an attacker changed (m, a) into a different (m', a'); or perhaps the attacker is inactive and $(m', a') = (m, a)$. The receiver checks whether $a' = m'_1r_1 + \cdots + m'_8r_8 + s$; the receiver discards (m', a') if $a' \neq m'_1r_1 + \cdots + m'_8r_8 + s$.

From the attacker's point of view, given (m, a), all $\#k^8$ choices of $(r_1, \ldots, r_8)$ are equally likely: each choice of $(r_1, \ldots, r_8)$ is consistent with exactly one choice of s. A forgery attempt (m', a') with $m' \neq m$ succeeds if and only if

$$m'_1r_1 + \cdots + m'_8r_8 - a' = m_1r_1 + \cdots + m_8r_8 - a,$$

i.e., $(m'_1 - m_1)r_1 + \cdots + (m'_8 - m_8)r_8 = a' - a$; this equation is satisfied for only $\#k^7$ choices of $(r_1, \ldots, r_8)$.

Each forgery attempt therefore has chance at most $1/\#k$ of success. If k is reasonably large, say $\#k \approx 2^{128}$, then this chance is negligible.

More generally, for any fixed n, a message $m_1, \ldots, m_n$ will be protected by the authenticator $m_1r_1 + \cdots + m_nr_n + s$, where $r_1, \ldots, r_n, s$ are independent uniform random elements of k.

2.3. History. This authenticator was, historically, the first provably secure authenticator. It was introduced by Gilbert, MacWilliams, and Sloane in [**13**]. Specifically, [**13**, Section V] presents the authentication equation "$si-j = c$," where "s" is a message in a field of size q, "i" and "j" are secret elements of the same field, and "c" is an authenticator; [**13**, Section VIII] generalizes to linear polynomials in more variables, using the language of finite geometries. (The language used in [**13**] seems to have deterred many potential readers; the Gilbert/MacWilliams/Sloane authenticator is often miscredited to Carter and Wegman.)

The speed of the Gilbert/MacWilliams/Sloane authenticator does not seem to have been appreciated until the 1997 Halevi/Krawczyk implementation report [**15**]. Halevi and Krawczyk chose a prime field $k = \mathbf{Z}/(2^{32} + 15)$ to take advantage of easy reductions modulo $2^{32} + 15$. Halevi and Krawczyk modified the authenticator slightly to gain some extra speed (losing a few bits of security): specifically, starting from 32-bit integers $m_1, r_1, \ldots$, they computed $m_1r_1 + \cdots$ modulo 2^{64}, and then reduced the result modulo $2^{32} + 15$. Halevi and Krawczyk went to some effort to multiply efficiently using "MMX" instructions on their target CPU. Many of the other papers mentioned in Section 1 have explored other choices of finite fields, other tweaks to the authenticator, and other techniques of multiplying efficiently.

2.4. Handling multiple messages. If the same secrets $r_1, \ldots, r_n, s$ are used to send two or more messages then security evaporates. However, if the sender and receiver share independent uniform random secrets $r_1, \ldots, r_n, s[1], s[2], s[3], \ldots \in k$ then they can safely use $(r_1, \ldots, r_n, s[1])$ for the first message, $(r_1, \ldots, r_n, s[2])$ for the second message, $(r_1, \ldots, r_n, s[3])$ for the third message, etc. This is an example of a structure introduced by Wegman and Carter in [**34**].

An attacker who succeeds at corrupting a message can trivially deduce a linear equation for $(r_1, \ldots, r_n)$ and can thus forge additional messages. This "re-forgery"

feature has attracted some attention: a few papers have breathlessly advertised it as a new discovery, a few papers have criticized it as a disadvantage of the Wegman/Carter structure, and a few papers have pointed out that it poses problems for the idea of occasional forgeries being acceptable. I recommend choosing k large enough that the attacker has no hope of ever succeeding.

Long secrets such as $r_1, \ldots, r_n, s[1], s[2], s[3], \ldots$ can be simulated by short keys; this is the whole point of "stream ciphers," "cryptographically strong PRNGs," etc. For example, one can expand a 128-bit AES key z into the 1280000-bit string $\mathrm{AES}_z(0), \mathrm{AES}_z(1), \mathrm{AES}_z(2), \ldots, \mathrm{AES}_z(9999)$; a uniform random choice of z does not produce a uniform random 1280000-bit string, but any attack on the resulting authenticator implies an attack on AES. I don't know whether this was considered obvious 25 years ago; Brassard in [**10**] in 1983 suggested feeding a short key through the Blum/Micali PRNG, the Blum/Blum/Shub PRNG, etc. to simulate the long secrets inside the Wegman/Carter protocol.

In many situations, the long string $r_1, \ldots, r_n, s[1], s[2], s[3], \ldots$ is not accessed sequentially; see, e.g., [**10**, pages 82–83]. Fortunately, some stream ciphers allow fast random access, allowing portions of the string to be dynamically generated from a short key upon request. For example, $i \mapsto \mathrm{AES}_z(i)$ can be evaluated efficiently.

2.5. Wegman/Carter reconsidered. Tanja Lange pointed out to me in 2006 the possibility of discarding the Wegman/Carter structure and simply using a new $(r_1, \ldots, r_n, s)$ for each message.

This possibility has not received serious attention in the literature. The obvious disadvantage is that the secrets $r_1[1], \ldots, r_n[1], s[1], r_1[2], \ldots, r_n[2], s[2], \ldots$ would consume $n+1$ times as much space (asymptotically) as the Wegman/Carter secrets $r_1, \ldots, r_n, s[1], s[2], \ldots$. Wegman and Carter emphasize in [**35**, page 274] that they use an "asymptotically optimal" number of secret bits. But counting bits seems fairly pointless in light of subsequent developments:

- More advanced authenticators replace $r_1, \ldots, r_n$ by a single $r \in k$, no matter how long the message is, as discussed in Sections 4 and 5 of this paper. This means that the secrets $r[1], s[1], r[2], s[2], \ldots$ consume only twice as much space as the Wegman/Carter secrets $r, s[1], s[2], \ldots$.
- The total space becomes irrelevant when secrets are generated dynamically from a short key. The Wegman/Carter structure needs to generate and inspect the same number of secrets as using a new r, s for each message.
- A stream cipher F that produces a large output block allows $r[i], s[i]$ to be computed from a single $F_z(i)$. The Wegman/Carter structure is *slower* in this situation: it obtains r from $F_z(0)$ and $s[i]$ from $F_z(i)$, requiring *two* evaluations of F. One could cache r along with the key z, but this effectively increases the space used for the key. Senders and receivers juggling many simultaneous keys miss the cache more often; hardware implementations become more expensive.
- Using a new $r[i], s[i]$ for the ith message makes each "re-forgery" just as difficult as an initial forgery, if the receiver takes the standard step of not allowing two authenticated messages with the same message number.

Perhaps the Wegman/Carter structure has outlived its usefulness!

2.6. Handling variable-length messages. If $\#k \geq 2^{8b}$ then an element of k can encode b bytes, so n elements of k can encode bn bytes. However, most

applications allow messages to vary in length; the set of messages is not $(\mathbf{Z}/256)^{bn}$ but the larger set $(\mathbf{Z}/256)^0 \cup (\mathbf{Z}/256)^1 \cup \cdots \cup (\mathbf{Z}/256)^{bn}$.

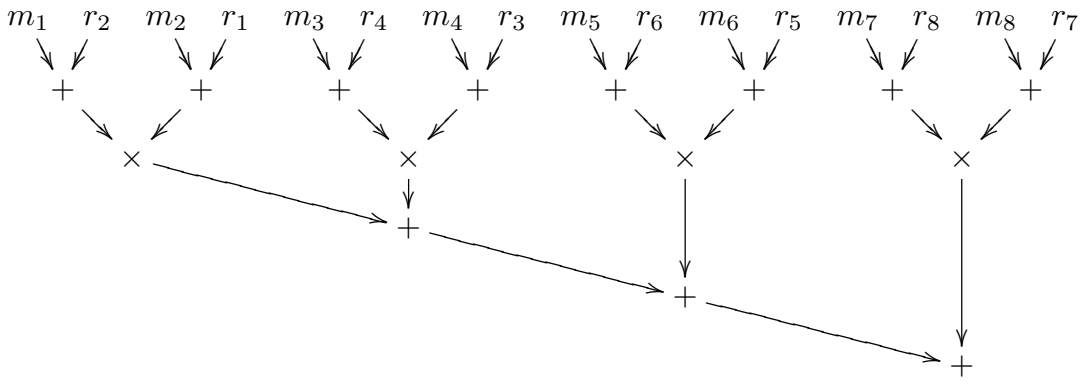
One reasonably efficient answer is to split each message into a sequence of b-byte blocks, possibly followed by one final block between 1 and $b-1$ bytes; to encode each block $(x_0, x_1, \ldots, x_{i-1})$ as the integer $x_0 + 256x_1 + \cdots + 256^{i-1}x_{i-1} + 256^i$; to encode each integer as an element of k, assuming $\#k \geq 2^{8b+1}$; and to 0-pad the result out to n elements of k. This encoding satisfies the security requirement that distinct byte strings produce distinct elements of k^n; it also has the virtue that strings of only bj bits, for $j < n$, involve only $r_1, \ldots, r_j$.

3. Fast pseudo-linear polynomial evaluation and message authentication

3.1. Pseudo dot products. Each entry in the product MR of two $n \times n$ matrices M, R is, by definition, the dot product of a row of M and a column of R. If each dot product is computed by the obvious algorithm discussed in Section 2, with n multiplications and $n-1$ additions, then the n^2 entries of MR are computed with a total of n^3 multiplications and $n^3 - n^2$ additions.

Winograd in [**37**] introduced a clever dot-product algorithm that reduces the number of multiplications to about $n^3/2$. The number of additions increases by a similar amount, but additions are much faster than multiplications in typical cost measures. (State-of-the-art matrix-multiplication algorithms combine Winograd's idea with subsequent improvements for large n, starting with Strassen's algorithm in [**32**].)

The following diagram shows Winograd's algorithm for $n = 8$. The algorithm uses 15 operations, as in Section 2, but those operations consist of 11 additions and only 4 multiplications:



The output of this algorithm is not exactly the dot product $m_1r_1 + \cdots + m_8r_8$. It is a **pseudo dot product**, specifically the dot product $m_1r_1 + \cdots + m_8r_8$ plus $m_1m_2 + m_3m_4 + m_5m_6 + m_7m_8$ plus $r_1r_2 + r_3r_4 + r_5r_6 + r_7r_8$.

In the context of matrix multiplication, the sum $m_1m_2 + \cdots$ can be cached along with the row m and reused for each column r, and the sum $r_1r_2 + \cdots$ can be cached along with the column r and reused for each row m, so Winograd's pseudo dot product is easily adjusted to produce the usual dot product. As we will see, in the context of message authentication, one can simply replace the dot product in the Gilbert/MacWilliams/Sloane authenticator with Winograd's pseudo dot product.

3.2. Another secure message-authentication code. Fix a finite field k. As in Section 2.2, the legitimate sender generates 9 independent uniform random

secrets $r_1, r_2, \ldots, r_8, s \in k$ and privately transmits those secrets to the legitimate receiver.

Later the sender wants to send a message m to the receiver through a public network, where m is a sequence of 8 components $m_1, m_2, \ldots, m_8 \in k$. The sender transmits m together with the **authenticator**

$$a = (m_1 + r_2)(m_2 + r_1) + \cdots (m_7 + r_8)(m_8 + r_7) + s \in k.$$

As in Section 2.2, the message (m', a') delivered to the receiver might not be (m, a). The receiver discards (m', a') if $a' \neq (m'_1+r_2)(m'_2+r_1)+\cdots+(m'_7+r_8)(m'_8+r_7)+s$.

From the attacker's point of view, given (m, a), all $\#k^8$ choices of $(r_1, \ldots, r_8)$ are equally likely; as in Section 2.2, the addition of a uniform random secret s hides all information about $(r_1, \ldots, r_8)$. A forgery attempt (m', a') with $m' \neq m$ succeeds if and only if

$$\begin{aligned}&(m'_1 + r_2)(m'_2 + r_1) + \cdots + (m'_7 + r_8)(m'_8 + r_7) - a' \\ &= (m_1 + r_2)(m_2 + r_1) + \cdots + (m_7 + r_8)(m_8 + r_7) - a,\end{aligned}$$

i.e., $(m'_1 - m_1)r_1 + (m'_2 - m_2)r_2 + \cdots + (m'_7 - m_7)r_7 + (m'_8 - m_8)r_8 = a' - a + m_1m_2 - m'_1m'_2 + \cdots + m_7m_8 - m'_7m'_8$. This equation is satisfied for only $\#k^7$ choices of $(r_1, \ldots, r_8)$. Each forgery attempt therefore has chance at most $1/\#k$ of success.

The same idea can easily be used for messages of any length n, for multiple messages, and for variable-length messages, as in Section 2. It uses only half as many multiplications as the Gilbert/MacWilliams/Sloane authenticator.

3.3. History. As far as I know, the first publication of this authenticator was in the 1997 Halevi/Krawczyk paper [**15**, Section 5], with credit to Wegman and Carter (and no credit to Winograd). The first implementation results were in a series of papers by Black, Halevi, Krawczyk, Krovetz, and Rogaway describing various functions called "UMAC," starting with [**8**] in 1999. Newer functions using the same idea include "Rabbit," published by Boesgaard, Scavenius, Pedersen, Christensen, and Zenner in [**9**], and "VMAC," published by Krovetz in [**21**].

3.4. The inefficiency of multiple variables. This authenticator uses $n+1$ secrets $r_1, r_2, \ldots, r_n, s \in k$ to handle a message $m_1, m_2, \ldots, m_n$ of length n. UMAC, for example, uses 1600 bytes of secret data to handle a typical-size network packet, and spends more than 20000 cycles on typical CPUs to generate that secret data from a short AES key.

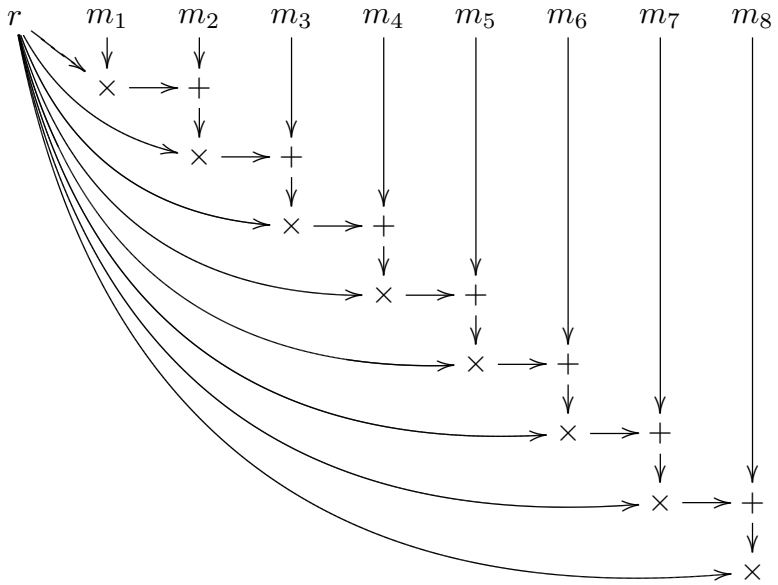
For comparison, consider the popular "HMAC-MD5" authenticator, essentially $\mathrm{MD5}(1, z, \mathrm{MD5}(0, z, m))$. HMAC-MD5 uses a short key z, for example 16 bytes, and finishes handling a typical-size network packet m in under 10000 cycles. There are many reasons to question the security of HMAC-MD5, but users are generally not happy to switch to an authentication function that is more than twice as slow!

One can reasonably ignore the cost of computing the variables $r_1, r_2, \ldots, r_n$ if those variables are reused for a large number of messages. The UMAC papers advertise much better speeds obtained *after* the 1600 bytes of secrets have been precomputed and cached. But this precomputation creates a different type of speed problem for busy Internet servers receiving messages from thousands of senders simultaneously: thousands of 1600-byte tables do not fit into cache simultaneously and take time to load into cache, often swamping the time used for the main UMAC computation. Large precomputed tables are even more troublesome for low-cost hardware devices.

The authenticators discussed in Sections 4 and 5 replace $r_1, r_2, \ldots, r_n$ with a single $r \in k$, eliminating these problems.

4. High-degree polynomial evaluation and message authentication

4.1. Horner's rule. The following diagram shows the standard method to compute the polynomial $m_1 r^8 + m_2 r^7 + \cdots + m_8 r$, using 8 multiplications and 7 additions:



4.2. Another secure message-authentication code. Fix a finite field k. The legitimate sender generates two independent uniform random secrets $r, s \in k$ and privately transmits those secrets to the legitimate receiver.

Later the sender wants to send a message m to the receiver through a public network, where m is a sequence of 8 components $m_1, m_2, \ldots, m_8 \in k$. The sender transmits m together with the **authenticator** $a = m_1 r^8 + \cdots + m_8 r + s \in k$. As in previous sections, the message (m', a') delivered to the receiver may be different from (m, a). The receiver discards (m', a') if $a' \neq m'_1 r^8 + \cdots + m'_8 r + s$.

From the attacker's point of view, given (m, a), all $\#k$ choices of r are equally likely. A forgery attempt (m', a') with $m' \neq m$ succeeds if and only if $m'_1 r^8 + \cdots + m'_8 r - a' = m_1 r^8 + \cdots + m_8 r - a$, i.e., $(m'_1 - m_1) r^8 + \cdots + (m'_8 - m_8) r = a' - a$. This is a nonzero polynomial equation in r of degree at most 8, so it has at most 8 roots in k. Each forgery attempt therefore has chance at most $8/\#k$ of success.

More generally, for any fixed n, a message $m_1, \ldots, m_n$ will be protected by the authenticator $m_1 r^n + \cdots + m_n r + s$, where r, s are independent uniform random elements of k. Each forgery attempt has chance at most $n/\#k$ of success. The same idea can also be used for any number of messages, as in Sections 2.4 and 2.5, and can be extended to variable-length messages.

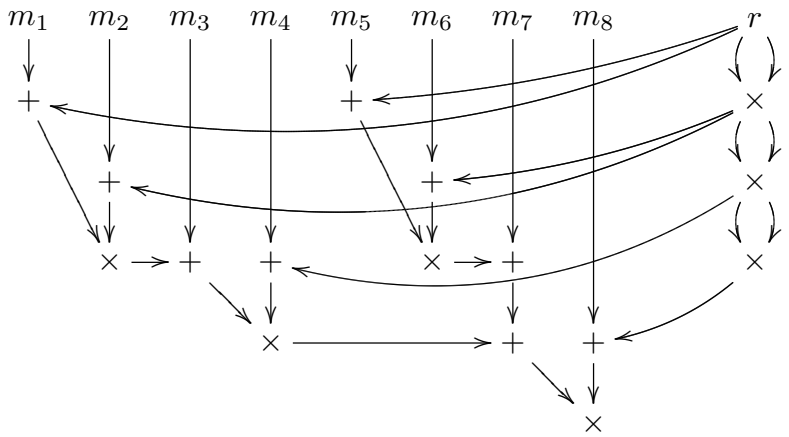
4.3. History. This authenticator was introduced in three independent papers: [**12**] by den Boer; [**18**] by Johansson, Kabatianskii, and Smeets; and [**33**] by Taylor. The first speed reports for this authenticator were [**29**, Section 5] by Shoup and [**25**, Section 3.5] by Nevelsteen and Preneel, in both cases with $\#k = 2^{64}$. Most subsequent papers have switched from characteristic 2 to large characteristic to

obtain better performance in software on common CPUs. I reported speeds for $\#k = 2^{127} - 1$ (foolishly assuming a large precomputed table) in [**3**]; I suggested $\#k = 2^{130} - 5$ in [**4**] and reported speeds (with no tables) in [**5**]. Meanwhile, Krovetz and Rogaway reported speeds for $\#k = 2^{32} - 5$ (with no tables) in [**22**]; Kohno, Viega, and Whiting reported speeds for $\#k = 2^{127} - 1$ in [**20**]; McGrew and Viega switched back to characteristic 2 for the sake of hardware performance, and reported speeds for $\#k = 2^{128}$ in [**23**].

This authenticator seems to have attracted more interest than the n-variable authenticators discussed in Sections 2 and 3; the obvious explanation is that it solves the problems discussed in Section 3.4. On the other hand, the authenticator in Section 3 has the advantage of using half as many multiplications. Fortunately, with more work, these advantages can be combined. See Section 5.

5. Fast high-degree polynomial evaluation and message authentication

5.1. Beyond Horner's rule. The following diagram presents an 8-parameter family of polynomial-evaluation algorithms using 3 squarings, 4 multiplications, and 9 additions:



The output of this algorithm is $H(m_1, m_2, \ldots, m_8)$, a degree-15 polynomial in r; see Section 5.2 for the general definition of H. The critical feature of H is that different vectors $(m_1, m_2, \ldots, m_8) \in k^8$ produce different polynomials $H(m_1, m_2, \ldots, m_8) \in k[r]$.

The central idea of this algorithm was published by Rabin and Winograd in [**26**, Corollary 13]. (I have found two sources attributing the idea to Winograd's earlier papers [**38**] and [**39**], but I have not been able to track down copies of [**38**] and [**39**]; my presumption is that those papers also give credit to Rabin.) I have changed a few details of the Rabin/Winograd algorithm, for reasons discussed below.

5.2. Definition of H. The polynomial $H(m_1, m_2, \ldots, m_n) \in k[r]$ is defined as follows for $n \geq 0$:

- $H() = 0$.
- $H(m_1) = m_1$.
- $H(m_1, m_2) = m_1 r + m_2$.
- $H(m_1, m_2, m_3) = (r + m_1)(r^2 + m_2) + m_3$.
- $H(m_1, m_2, \ldots, m_n) = H(m_1, \ldots, m_{t-1}) \cdot (r^t + m_t) + H(m_{t+1}, \ldots, m_n)$ if $t \in \{4, 8, 16, 32, \ldots\}$ and $t \leq n < 2t$.

Computing $H(m_1, \ldots, m_n)$ directly from the definition involves a few squarings to compute $r^2, r^4, r^8, \ldots$; $\lfloor n/2 \rfloor$ multiplications; and $n + \lfloor (n-3)/4 \rfloor$ additions. For example:

- $H() = 0$; degree $-\infty$; 0 squarings, 0 multiplications, 0 additions.
- $H(m_1) = m_1$; degree ≤ 0; 0 squarings, 0 multiplications, 0 additions.
- $H(m_1, m_2) = m_1 r + m_2$; degree ≤ 1; 0 squarings, 1 multiplication, 1 addition.
- $H(m_1, m_2, m_3) = (r + m_1)(r^2 + m_2) + m_3$; degree 3; 1 squaring, 1 multiplication, 3 additions.
- $H(m_1, m_2, m_3, m_4) = ((r + m_1)(r^2 + m_2) + m_3)(r^4 + m_4)$; degree 7; 2 squarings, 2 multiplications, 4 additions.
- $H(m_1, m_2, m_3, m_4, m_5) = ((r+m_1)(r^2+m_2)+m_3)(r^4+m_4)+m_5$; degree 7; 2 squarings, 2 multiplications, 5 additions.
- $H(m_1, m_2, m_3, m_4, m_5, m_6) = ((r+m_1)(r^2+m_2)+m_3)(r^4+m_4)+m_5 r+m_6$; degree 7; 2 squarings, 3 multiplications, 6 additions.
- $H(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = ((r+m_1)(r^2+m_2)+m_3)(r^4+m_4)+(r+m_5)(r^2+m_6)+m_7$; degree 7; 2 squarings, 3 multiplications, 8 additions.
- $H(m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8) = (((r+m_1)(r^2+m_2)+m_3)(r^4+m_4)+(r+m_5)(r^2+m_6)+m_7)(r^8+m_8)$; degree 15; 3 squarings, 4 multiplications, 9 additions.

It is easy to see that every monic degree-3 polynomial can be expressed as $H(m_1, m_2, m_3)$ for a unique (m_1, m_2, m_3); that every monic degree-7 polynomial can be expressed as $H(m_1, \ldots, m_7)$ for a unique $(m_1, \ldots, m_7)$; and so on for degrees 15, 31, etc. This is exactly the content of [**26**, Corollary 13].

For other values of n my function H is different from, and slightly faster than, the constructions in [**26**]. Rabin and Winograd sacrifice roughly $\lg n$ multiplications to achieve surjectivity. The novelty here is in asking the right question; the tweaks to the constructions are straightforward once the goal of top-speed injectivity has been identified.

5.3. Injectivity proofs. Theorem 5.5 states that H is injective for fixed-length inputs. Theorem 5.6 states that H is injective for variable-length inputs, when the inputs are limited to an appropriate subset of k.

THEOREM 5.4. *Let k be a finite field. Let n be a nonnegative integer. Let $m_1, \ldots, m_n$ be elements of k. If $n \leq 2$ then $\deg H(m_1, \ldots, m_n) \leq 1$. If $n \geq 3$ then $H(m_1, \ldots, m_n)$ is a monic polynomial of degree $2^{\lfloor \lg n \rfloor + 1} - 1$.*

PROOF. Induct on n. Write $h = H(m_1, \ldots, m_n)$.

Case 0: $n = 0$. Then $h = 0$ so $\deg h = -\infty \leq 1$.

Case 1: $n = 1$. Then $h = m_1$ so $\deg h \leq 0 \leq 1$.

Case 2: $n = 2$. Then $h = m_1 r + m_2$ so $\deg h \leq 1$.

Case 3: $n = 3$. Then $h = (r + m_1)(r^2 + m_2) + m_3$ so h is a monic polynomial of degree $3 = 2^{\lfloor \lg 3 \rfloor + 1} - 1$.

Case 4: $n \geq 4$. Write $t = 2^{\lfloor \lg n \rfloor}$. Then $t \in \{4, 8, 16, 32, \ldots\}$ and $t \leq n < 2t$ so $h = H(m_1, \ldots, m_{t-1}) \cdot (r^t + m_t) + H(m_{t+1}, \ldots, m_n)$ by definition of H. By the inductive hypothesis, $H(m_1, \ldots, m_{t-1})$ is a monic polynomial of degree $2^{\lfloor \lg (t-1) \rfloor + 1} - 1 = t - 1$, so $H(m_1, \ldots, m_{t-1}) \cdot (r^t + m_t)$ is a monic polynomial of degree $2t - 1$. By the inductive hypothesis again, $H(m_{t+1}, \ldots, m_n)$ has degree at most $t - 1$. Thus h is a monic polynomial of degree $2t - 1 = 2^{\lfloor \lg n \rfloor + 1} - 1$. □

THEOREM 5.5. *Let k be a finite field. Let n be a nonnegative integer. Then H maps k^n injectively to the polynomial ring $k[r]$.*

PROOF. Assume that $H(m_1, m_2, \ldots, m_n) = H(m'_1, m'_2, \ldots, m'_n)$. The goal is to prove that $(m_1, m_2, \ldots, m_n) = (m'_1, m'_2, \ldots, m'_n)$. The proof will induct on n.

If $n = 0$ then there is nothing to prove.

If $n = 1$ then $m_1 = H(m_1) = H(m'_1) = m'_1$.

If $n = 2$ then $m_1 r + m_2 = H(m_1, m_2) = H(m'_1, m'_2) = m'_1 r + m'_2$ so $m_1 = m'_1$ and $m_2 = m'_2$.

If $n = 3$ then $(r + m_1)(r^2 + m_2) + m_3 = H(m_1, m_2, m_3) = H(m'_1, m'_2, m'_3) = (r + m'_1)(r^2 + m'_2) + m'_3$. Compare coefficients of r^2 to see that $m_1 = m'_1$; compare coefficients of r^1 to see that $m_2 = m'_2$; compare coefficients of r^0 to see that $m_1 m_2 + m_3 = m'_1 m'_2 + m'_3$, implying $m_3 = m'_3$.

Assume from now on that $n \geq 4$. Define $t = 2^{\lfloor \lg n \rfloor}$. Then $t \leq n < 2t$, so $H(m_1, \ldots, m_n) = H(m_1, \ldots, m_{t-1}) \cdot (r^t + m_t) + H(m_{t+1}, \ldots, m_n)$ by definition of H.

By Theorem 5.4, $H(m_1, \ldots, m_{t-1})$ and $H(m_{t+1}, \ldots, m_n)$ have degree at most $t-1$, so $\lfloor H(m_1, \ldots, m_n)/r^t \rfloor = H(m_1, \ldots, m_{t-1})$. Similarly $\lfloor H(m'_1, \ldots, m'_n)/r^t \rfloor = H(m'_1, \ldots, m'_{t-1})$. Hence $H(m_1, \ldots, m_{t-1}) = H(m'_1, \ldots, m'_{t-1})$. By the inductive hypothesis, $(m_1, \ldots, m_{t-1}) = (m'_1, \ldots, m'_{t-1})$.

By Theorem 5.4 again, the coefficient of r^{t-1} in $H(m_1, \ldots, m_{t-1})$ is 1, and the coefficient of r^{t-1} in $H(m_{t+1}, \ldots, m_n)$ is $[n - t \geq t/2]$. The coefficient of r^{t-1} in $H(m_1, \ldots, m_n)$ is therefore $m_t + [n - t \geq t/2]$. Similarly, the coefficient of r^{t-1} in $H(m'_1, \ldots, m'_n)$ is $m'_t + [n - t \geq t/2]$. Hence $m_t = m'_t$.

Finally $H(m_{t+1}, \ldots, m_n) = H(m'_{t+1}, \ldots, m'_n)$, so by the inductive hypothesis $(m_{t+1}, \ldots, m_n) = (m'_{t+1}, \ldots, m'_n)$. □

THEOREM 5.6. *Let k be a finite field. Let S be a subset of k. Assume that $0 \notin S$ and that $S \cap (S + 1) = \{\}$. Then H maps $S^0 \cup S^1 \cup S^2 \cup \cdots$ injectively to the polynomial ring $k[r]$.*

The notation $S \cap (S + 1) = \{\}$ means that each $s \in S$ has $s + 1 \notin S$. For example, one can take $S = \{1, 3, 5, 7, \ldots, p - 2\}$ if k is a prime field $\mathbf{Z}/p$. A closer look at the proof reveals that the condition $0 \notin S$ matters only for $m_1, m_5, m_9, \ldots$, while the condition $s + 1 \notin S$ matters only for $m_4, m_8, m_{12}, \ldots$; one can therefore squeeze a few more message bits into k^n.

PROOF. Assume that $H(m_1, \ldots, m_n) = H(m'_1, \ldots, m'_{n'})$, with each $m_i \in S$ and each $m'_i \in S$. The goal is to prove that $(m_1, \ldots, m_n) = (m'_1, \ldots, m'_{n'})$. If $n = n'$ then we are done by Theorem 5.5; the difficulty is that n could be different from n'.

Observe that the degree of $H(m_1, \ldots, m_n)$ controls the range of n, and therefore also the range of n'. Specifically, $H() = 0$, with degree $-\infty$; $H(m_1) = m_1$, with degree 0, since $m_1 \neq 0$ by hypothesis; $H(m_1, m_2) = m_1 r + m_2$, with degree 1, for the same reason; $H(m_1, m_2, m_3)$ has degree 3; $H(m_1, \ldots, m_n)$ has degree 7 for $4 \leq n \leq 7$, by Theorem 5.4; $H(m_1, \ldots, m_n)$ has degree 15 for $8 \leq n \leq 15$, by Theorem 5.4; etc.

So fix $t \in \{4, 8, 16, \ldots\}$, and assume that $t \leq n < 2t$ and $t \leq n' < 2t$. Now

$$H(m_1, \ldots, m_n) = H(m_1, \ldots, m_{t-1}) \cdot (r^t + m_t) + H(m_{t+1}, \ldots, m_n)$$

$$H(m'_1, \ldots, m'_{n'}) = H(m'_1, \ldots, m'_{t-1}) \cdot (r^t + m'_t) + H(m'_{t+1}, \ldots, m'_{n'})$$

by definition of H, so $(m_1, \ldots, m_{t-1}) = (m'_1, \ldots, m'_{t-1})$ exactly as in the proof of Theorem 5.5.

The coefficient of r^{t-1} in $H(m_1, \ldots, m_{t-1})$ is 1, and the coefficient of r^{t-1} in $H(m_{t+1}, \ldots, m_n)$ is either 0 or 1, by Theorem 5.4. Therefore the coefficient of r^{t-1} in $H(m_1, \ldots, m_n)$ is either m_t or $m_t + 1$. Similarly, the coefficient of r^{t-1} in $H(m'_1, \ldots, m'_{n'})$ is either m'_t or $m'_t + 1$. Hence $m_t = m'_t$, or $m_t = m'_t + 1$, or $m_t + 1 = m'_t$, or $m_t + 1 = m'_t + 1$. Now $m_t, m'_t \in S$ so $m_t + 1, m'_t + 1 \notin S$ by hypothesis; hence $m_t \neq m'_t + 1$ and $m_t + 1 \neq m'_t$. Therefore $m_t = m'_t$.

Finally $H(m_{t+1}, \ldots, m_n) = H(m'_{t+1}, \ldots, m'_{n'})$, so by the inductive hypothesis $(m_{t+1}, \ldots, m_n) = (m'_{t+1}, \ldots, m'_{n'})$. □

5.7. A new secure message-authentication code. Fix a finite field k. As in Section 4.2, the legitimate sender generates two independent uniform random secrets $r, s \in k$ and privately transmits those secrets to the legitimate receiver.

Later the sender wants to send a message m to the receiver through a public network, where m is a sequence of 8 components $m_1, m_2, \ldots, m_8 \in k$. The sender transmits m together with the **authenticator** $a = rH(m_1, \ldots, m_8) + s \in k$. As in previous sections, the message (m', a') delivered to the receiver may be different from (m, a). The receiver discards (m', a') if $a' \neq rH(m'_1, \ldots, m'_8) + s$.

A forgery attempt (m', a') with $m' \neq m$ succeeds if and only if

$$r(H(m'_1, \ldots, m'_8) - H(m_1, \ldots, m_8)) = a' - a.$$

This is a nonzero polynomial equation in r of degree at most 15: $H(m'_1, \ldots, m'_8)$ and $H(m_1, \ldots, m_8)$ are monic polynomials of degree 15 by Theorem 5.4, and they are distinct by Theorem 5.5. This equation is therefore satisfied for at most 15 choices of $r \in k$. Each forgery attempt therefore has chance at most $15/\#k$ of success.

More generally, for any fixed n, a message $m_1, \ldots, m_n$ will be protected by the authenticator $rH(m_1, \ldots, m_n) + s$, where r, s are independent uniform random elements of k. Each forgery attempt has chance at most $(2n-1)/\#k$ of success if $n \geq 1$. The same idea can also be used for any number of messages, as in Sections 2.4 and 2.5.

Some care is required for variable-length messages. The authenticator can safely be used for messages in $S^0 \cup S^1 \cup \cdots$ if $S \subset k$ satisfies $0 \notin S$ and $S \cap (S + 1) = \{\}$. See Theorem 5.6.

5.8. Comparison to linear authenticators. The fast linear authenticators described in Section 3 use approximately $n/2$ multiplications, $3n/2$ additions, and n variables. The new authenticators described in this section use approximately $\lg n$ squarings, $n/2$ multiplications, $5n/4$ additions, and 1 variable.

The critical change here is from n variables to 1 variable; see Section 3.4. The improvement from $3n/2$ additions to $5n/4$ additions might also be noticeable. There is an extra cost of $\lg n$ squarings, but this cost becomes unnoticeable as n grows. One could cache $r^2, r^4, r^8, \ldots$, eliminating the squarings in favor of $\lg n$ variables.

References

[1] — (no editor), *20th annual symposium on foundations of computer science*, IEEE Computer Society, New York, 1979. MR 82a:68004. See [34].

[2] Edward G. Belaga, *Some problems in the computation of polynomials*, Doklady Akademii Nauk SSSR **123** (1958), 775–777. ISSN 0002–3264. Citations in this document: §1.

[3] Daniel J. Bernstein, *Floating-point arithmetic and message authentication*, to be incorporated into author's *High-speed cryptography* book (1999). URL: `http://cr.yp.to/papers.html#hash127`. ID `dabadd3095644704c5cbe9690ea3738e`. Citations in this document: §1.1, §4.3.

[4] Daniel J. Bernstein, *Speed records for cryptographic software: an update* (2002). URL: `http://cr.yp.to/talks.html#2002.06.15`. Citations in this document: §1.1, §4.3.

[5] Daniel J. Bernstein, *The Poly1305-AES message-authentication code*, in [**14**] (2005), 32–49. URL: `http://cr.yp.to/papers.html#poly1305`. ID `0018d9551b5546d97c340e0dd8cb5750`. Citations in this document: §1.1, §4.3.

[6] Eli Biham (editor), *Fast Software Encryption '97*, Lecture Notes in Computer Science, 1267, Springer-Verlag, Berlin, 1997. ISBN 3–540–63247–6. See [15].

[7] Eli Biham, Amr M. Youssef (editors), *Selected areas in cryptography, 13th international workshop, SAC 2006, Montreal, Canada, August 17-18, 2006 revised selected papers*, Lecture Notes in Computer Science, 4356, Springer, 2007. ISBN 978–3–540–74461–0. See [21].

[8] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, Phillip Rogaway, *UMAC: fast and secure message authentication*, in [**36**] (1999), 216–233. URL: `http://www.cs.ucdavis.edu/~rogaway/umac/`. Citations in this document: §1.1, §3.3.

[9] Martin Boesgaard, Ove Scavenius, Thomas Pedersen, Thomas Christensen, Erik Zenner, *Badger—a fast and provably secure MAC*, in [**17**] (2005), 176–191. Citations in this document: §1.1, §3.3.

[10] Gilles Brassard, *On computationally secure authentication tags requiring short secret shared keys*, in [**11**] (1983), 79–86. URL: `http://cr.yp.to/bib/entries.html#1983/brassard`. Citations in this document: §2.4, §2.4.

[11] David Chaum, Ronald L. Rivest, Alan T. Sherman (editors), *Advances in cryptology: proceedings of Crypto 82*, Plenum Press, New York, 1983. ISBN 0–306–41366–3. MR 84j:94004. See [10].

[12] Bert den Boer, *A simple and key-economical unconditional authentication scheme*, Journal of Computer Security **2** (1993), 65–71. ISSN 0926–227X. URL: `http://cr.yp.to/bib/entries.html#1993/denboer`. Citations in this document: §1.1, §4.3.

[13] Edgar N. Gilbert, F. Jessie MacWilliams, Neil J. A. Sloane, *Codes which detect deception*, Bell System Technical Journal **53** (1974), 405–424. ISSN 0005–8580. MR 55:5306. URL: `http://cr.yp.to/bib/entries.html#1974/gilbert`. Citations in this document: §1.1, §2.3, §2.3, §2.3, §2.3.

[14] Henri Gilbert, Helena Handschuh (editors), *Fast software encryption: 12th international workshop, FSE 2005, Paris, France, February 21–23, 2005, revised selected papers*, Lecture Notes in Computer Science, 3557, Springer, 2005. ISBN 3–540–26541–4. See [5].

[15] Shai Halevi, Hugo Krawczyk, *MMH: software message authentication in the Gbit/second rates*, in [**6**] (1997), 172–189. URL: `http://www.research.ibm.com/people/s/shaih/pubs/mmh.html`. Citations in this document: §1.1, §1.1, §1.1, §2.3, §3.3.

[16] Tor Helleseth (editor), *Advances in Cryptology—EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, Lecture Notes in Computer Science, 765, Springer, 1994. ISBN 3–540–57600–2. See [18].

[17] John Ioannidis, Angelos D. Keromytis, Moti Yung, *Applied cryptography and network security, third international conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, proceedings*, Lecture Notes in Computer Science, 3531, 2005. ISBN 3–540–26223–7. See [9].

[18] Thomas Johansson, Gregory Kabatianskii, Ben J. M. Smeets, *On the relation between A-codes and codes correcting independent errors*, in [**16**] (1994), 1–11. Citations in this document: §1.1, §4.3.

[19] Neal Koblitz (editor), *Advances in cryptology—CRYPTO '96*, Lecture Notes in Computer Science, 1109, Springer-Verlag, Berlin, 1996. See [28].

[20] Tadayoshi Kohno, John Viega, Doug Whiting, *CWC: a high-performance conventional authenticated encryption mode*, in [**27**] (2004), 408–426. Citations in this document: §1.1, §4.3.

[21] Ted Krovetz, *Message authentication on 64-bit architectures*, in [**7**] (2007), 327–341. Citations in this document: §1.1, §3.3.

[22] Ted Krovetz, Phillip Rogaway, *Fast universal hashing with small keys and no preprocessing: the PolyR construction* (2000). URL: `http://www.cs.ucdavis.edu/~rogaway/papers/poly.htm`. Citations in this document: §1.1, §4.3.

[23] David A. McGrew, John Viega, *The security and performance of the Galois/counter mode of operation* (2004). URL: `http://eprint.iacr.org/2004/193/`. Citations in this document: §1.1, §4.3.

[24] Theodore S. Motzkin, *Evaluation of polynomials and evaluation of rational functions*, Bulletin of the American Mathematical Society **61** (1955), 163–163. URL: `http://www.ams.org/bull/1955-61-02/S0002-9904-1955-09897-5/S0002-9904-1955-09897-5.pdf`. Citations in this document: §1.

[25] Wim Nevelsteen, Bart Preneel, *Software performance of universal hash functions*, in [**30**] (1999), 24–41. Citations in this document: §1.1, §4.3.

[26] Michael O. Rabin, Shmuel Winograd, *Fast evaluation of polynomials by rational preparation*, Communications on Pure and Applied Mathematics **25** (1972), 433–458. MR 48 #10200. Citations in this document: §1, §1.1, §5.1, §5.2, §5.2.

[27] Bimal K. Roy, Willi Meier (editors), *Fast software encryption, 11th international workshop, FSE 2004, Delhi, India, February 5-7, 2004, revised papers*, Lecture Notes in Computer Science, 3017, Springer, 2004. ISBN 3–540–22171–9. See [20].

[28] Victor Shoup, *On fast and provably secure message authentication based on universal hashing*, in [**19**] (1996), 313–328; see also newer version [**29**].

[29] Victor Shoup, *On fast and provably secure message authentication based on universal hashing* (1996); see also older version [**28**]. URL: `http://www.shoup.net/papers`. Citations in this document: §1.1, §4.3.

[30] Jacques Stern (editor), *Advances in cryptology: EUROCRYPT '99*, Lecture Notes in Computer Science, 1592, Springer-Verlag, Berlin, 1999. ISBN 3–540–65889–0. MR 2000i:94001. See [25].

[31] Douglas R. Stinson (editor), *Advances in cryptology—CRYPTO '93: 13th annual international cryptology conference, Santa Barbara, California, USA, August 22–26, 1993, proceedings*, Lecture Notes in Computer Science, 773, Springer-Verlag, Berlin, 1994. ISBN 3–540–57766–1, 0–387–57766–1. MR 95b:94002. See [33].

[32] Volker Strassen, *Gaussian elimination is not optimal*, Numerische Mathematik **13** (1969), 354–356. ISSN 0029–599X. MR 40:2223. URL: `http://cr.yp.to/bib/entries.html#1969/strassen`. Citations in this document: §3.1.

[33] Richard Taylor, *An integrity check value algorithm for stream ciphers*, in [**31**] (1994), 40–48. URL: `http://cr.yp.to/bib/entries.html#1994/taylor`. Citations in this document: §1.1, §4.3.

[34] Mark N. Wegman, J. Lawrence Carter, *New classes and applications of hash functions*, in [**1**] (1979), 175–182; see also newer version [**35**]. URL: `http://cr.yp.to/bib/entries.html#1979/wegman`. Citations in this document: §2.4.

[35] Mark N. Wegman, J. Lawrence Carter, *New hash functions and their use in authentication and set equality*, Journal of Computer and System Sciences **22** (1981), 265–279; see also older version [**34**]. ISSN 0022–0000. MR 82i:68017. URL: `http://cr.yp.to/bib/entries.html#1981/wegman`. Citations in this document: §2.5.

[36] Michael Wiener (editor), *Advances in cryptology—CRYPTO '99*, Lecture Notes in Computer Science, 1666, Springer-Verlag, Berlin, 1999. ISBN 3–5540–66347–9. MR 2000h:94003. See [8].

[37] Shmuel Winograd, *A new algorithm for inner product*, IEEE Transactions on Computers **17** (1968), 693–694. ISSN 0018–9340. Citations in this document: §3.1.

[38] Shmuel Winograd, *Evaluating polynomials using rational auxiliary functions*, IBM Technical Disclosure Bulletin **13** (1970), 1133–1135. Citations in this document: §5.1, §5.1.

[39] Shmuel Winograd, *Fast evaluation of polynomials by rational preparation*, IBM Technical Report CS–73–27 (1971). Citations in this document: §5.1, §5.1.

Department of Mathematics, Statistics, and Computer Science (M/C 249), The University of Illinois at Chicago, Chicago, IL 60607–7045

E-mail address: `djb@cr.yp.to`