Stronger security bounds for Wegman-Carter-Shoup authenticators

Daniel J. Bernstein *

Department of Mathematics, Statistics, and Computer Science (M/C 249)
The University of Illinois at Chicago
Chicago, IL 60607-7045
djb@cs.uchicago.edu

Abstract. Shoup proved that various message-authentication codes of the form \((n, m) \rightarrow h(m) + f(n)\) are secure against all attacks that see at most \(\sqrt{1/\varepsilon}\) authenticated messages. Here \(m\) is a message; \(n\) is a nonce chosen from a public group \(G\); \(f\) is a secret uniform random permutation of \(G\); \(h\) is a secret random function; and \(\varepsilon\) is a differential probability associated with \(h\).

Shoup’s result implies that if AES is secure then various state-of-the-art message-authentication codes of the form \((n, m) \rightarrow h(m) + \text{AES}_h(n)\) are secure up to \(\sqrt{1/\varepsilon}\) authenticated messages. Unfortunately, \(\sqrt{1/\varepsilon}\) is only about \(2^{50}\) for some state-of-the-art systems, so Shoup’s result provides no guarantees for long-term keys.

This paper proves that security of the same systems is retained up to \(\sqrt{|G|}\) authenticated messages. In a typical state-of-the-art system, \(\sqrt{|G|}\) is \(2^{64}\). The heart of the paper is a very general “one-sided” security theorem: \((n, m) \rightarrow h(m) + f(n)\) is secure if there are small upper bounds on differential probabilities for \(h\) and on interpolation probabilities for \(f\).

1 Introduction

This paper proves that various state-of-the-art 128-bit authenticators are secure against all attacks that see at most \(2^{64}\) authenticated messages. Previous proofs broke down at a smaller number of messages, often below \(2^{50}\).

A typical example

Here is a well-known polynomial-evaluation message-authentication code over a field of size \(2^{128}\):

\* The author was supported by the National Science Foundation under grant CCR-9983650, and by the Alfred P. Sloan Foundation. Date of this document: 2005.02.27. Permanent ID of this document: 216037f7195642f3027da283229c1ad. This version is final and may be freely cited. Priority dates: The first version of this document was posted 2004.10.19.
Each message is a polynomial in one variable over the field. The polynomial’s constant coefficient is required to be 0. One authenticates the polynomial by evaluating it at a point and adding a function of the message number: the sender’s nth message, say \( m_n \), is transmitted as \( (n, m_n, m_n(r) + f(n)) \). Here \( r \) and \( f \) are secrets shared by the sender and the receiver.

It is easy to prove information-theoretic security of this system if \( r \) and \( f \) are independent, \( r \) is a uniform random element of the field, and \( f \) is a uniform random function from \( \{ n \} \) to the field—in other words, if \( r, f(1), f(2), \ldots \) are independent uniform random elements of the field. The attacker’s chance of successfully forging a message is at most \( LD/2^{128} \), where \( L \) is the maximum degree of a message and \( D \) is the number of forgeries attempted. The idea of the proof is that \( m_n(r) + f(n) \) leaks no information about \( m_n(r) \).

What if \( f \) is a uniform random injective function—in other words, what if \( f(1), f(2), \ldots \) are chosen to be distinct? What is the attacker’s chance of successfully forging a message? Here are three answers:

- The easy bound: Say the sender transmits only \( C \) messages, where \( C \) is small. Then \( f(1), f(2), \ldots, f(C) \) are nearly independent, and one can easily prove that the attacker’s chance of success is at most \( LD/2^{128} + C(C - 1)/2^{129} \). This bound becomes useless as \( C \) approaches \( 2^{64} \).
- Another bound: Shoup proved in [19, Theorem 2] that the attacker’s chance of success is at most \( 2LD/2^{128} \) if \( C \leq 2^{60} \), and below \( 1.7LD/2^{128} \) if \( C \leq 2^{64} \), and below \( 3000LD/2^{128} \) if \( C \leq 2^{66} \).

For example, say the sender authenticates \( C = 2^{60} \) messages, the attacker tries \( D = 2^{60} \) forgeries, and the maximum message degree is \( L = 2^{16} \). The easy bound is about \( 1/2^{61} \), which is not at all comforting. Shoup’s bound is inapplicable. The bound in this paper is \( 1.002/2^{52} \).

Consequences for AES-based authenticators

Despite the high speed and information-theoretic security of \( m_n(r) + f(n) \), users often prefer \( m_n(r) + \text{AES}_k(n) \). Why? Because \( r, k \) occupy only 32 bytes, whereas \( r, f(1), f(2), \ldots \) occupy an additional 16 bytes for each message.

Define \( \delta \) as the attacker’s chance of distinguishing \( \text{AES}_k \) from \( f \), where \( f \) is a uniform random injective function on 16-byte blocks. The important feature of AES for this paper is that \( \delta \) is believed to be extremely small, even after \( 2^{64} \) or more queries, even for an attacker with incredible computational resources. This was an explicit design goal of AES: [2, Section 4] identified “the extent to which the algorithm output is indistinguishable from the output of a uniform random permutation” as one of the “most important” factors in evaluating AES proposals, and [17, Table 1] shows that AES evaluators considered many attacks aimed at this feature.

The attacker’s success chance against \( m_n(r) + \text{AES}_k(n) \) is at most \( \delta \) plus the attacker’s success chance against \( m_n(r) + f(n) \). The point of this paper is
that the second chance is extremely small, even for $C = 2^{64}$. Consequently, $m_n(r) + \text{AES}_k(n)$ is secure if $\delta$ is small, i.e., if AES$_k$ meets its design goals.

In short, this paper guarantees that $m_n(r) + \text{AES}_k(n)$ is as secure as AES up to $2^{64}$ messages. The best previous results did not handle nearly as many messages.

The importance of injectivity

Suppose that, in the above discussion of AES, I modify the definition of $f$ by omitting the word “injective.” Does the rest of the argument lead to the same security guarantee? No!

It is still true that the attacker’s success chance against $m_n(r) + \text{AES}_k(n)$ is bounded by the sum of two chances: first, the attacker’s chance of distinguishing AES$_k$ from $f$; second, the attacker’s success chance against $m_n(r) + f(n)$. It is still true—and easy to prove, without the new techniques in this paper—that the second chance is small. But it is not true that AES was designed to make the first chance small. In fact, for $C = 2^{64}$, the first chance is not small. The attacker has a good chance of distinguishing AES$_k$ from $f$ by trying $2^{64}$ inputs and checking for collisions.

The importance of injectivity in this context was highlighted by Shoup in [19, Section 1] nearly ten years ago. As $C$ and $D$ grow, the usual theorems say “nothing at all about the security of the message authentication scheme,” Shoup wrote, pointing out examples of this problem in the literature.

Unfortunately, the literature has continued to sprout problems of this type. Example: [9, Section 6.1] claims, for one message-authentication code, that any attack with success probability larger than “about $2^{-60}$” has been “rigorously proven” to imply an attack that distinguishes AES from “a [uniform] family of random permutations.” In fact, the security analysis considered the uniform distribution on all functions, not the uniform distribution on permutations; see [16, page 15]. The error is below $2^{-60}$ if $C + D < 2^{34}$, but neither [16] nor [9] puts any such limit on $C + D$. Apparently the security “guarantee” in [9] has not been proven.

Generalization

This paper considers much more general message-authentication codes of the form $(n,m) \mapsto h(m) + f(n)$. The main theorem of this paper, Theorem 5.1, is that $h(m) + f(n)$ is secure if (1) differential probabilities for $h$ are small and (2) interpolation probabilities for $f$ are small.

The “differential probabilities for $h$” are probabilities of the form $\text{Pr}[h(m') = h(m) + g]$. The “interpolation probabilities for $f$” are probabilities of the form $\text{Pr}([f(n_1), \ldots, f(n_k)] = (x_1, \ldots, x_k])$. See Appendix A for further discussion of terminology.

In particular, assume that $f$ is a uniform random injective function from $\{n\}$ to a finite commutative group $G$, and that the differential probabilities for $h$
are small. Then \( h(m) + f(n) \) is secure against all attacks that see at most \( \sqrt{2^G} \)
authenticated messages. As a special case, if \( G \) is the set of 16-byte strings with a
group structure, then \( h(m) + f(n) \) is secure against all attacks that see at most
\( 2^{64} \) authenticated messages. Consequently \( h(m) + AES_k(n) \) is secure against
any attacker who cannot break AES and who sees at most \( 2^{64} \) authenticated
messages.

The form \( h(m) \oplus f(n) \) for an authenticator, where \( f \) is a uniform random
function, was introduced by Wegman and Carter in [22, Section 4]. Here \( \oplus \) is
vector addition modulo 2. Brassard in [10] considered \( h(m) \oplus f(n) \) where \( f \)
is a random function determined by a short key. Shoup in [19], as discussed above,
considered \( h(m) \oplus f(n) \) where \( f \) is a uniform random injective function.

There are many choices of \( h \) in the literature. The choices for any particular
output size (say 128 bits) vary in speed, entropy (e.g., 128 bits), maximum
differential probability \( \epsilon \) (e.g., \( L/2^{128} \)), et al. For example, Gilbert, MacWilliams,
and Sloane in [12] proposed \( m_1, m_2, \ldots, m_L \mapsto m_1 r_1 + m_2 r_2 + \cdots + m_L r_L \), which
has \( \epsilon = 1/2^{128} \) but a very long \( r = (r_1, r_2, \ldots, r_L) \). This is, at first glance, just
as fast as univariate polynomial evaluation, but in practice the large \( r \) creates a
huge speed penalty: cache misses become much more common and much more
expensive. Evaluating a polynomial in a few variables over a larger field achieves
the same \( \epsilon \) with a much smaller speed penalty. The larger field means a larger
output size, but Bierbrauer, Johansson, Kabatianski, and Smeets in [7, page
336] pointed out that the result could be safely truncated after an appropriate
twist; the bandwidth savings of truncation may justify the computation cost of
the twist. There is much more to say about the choice of \( h \); see [4, Section 10]
for a survey.

The more general shape \( h(m) + f(n) \) for an authenticator, where \( + \) can be any
commutative group operation, accommodates choices of \( h \) that rely on addition
in large characteristic rather than characteristic 2—in particular, functions that
can take advantage of the high-speed multiplication circuits included in common
CPUs. Several examples of such functions are “MMH,” “hash127,” “UMAC,”
“CWC-HASH,” and my new “Poly1305”; see [13], [4], [16], [15], and [5].

Shoup stated his theorems only for \( \oplus \), but his proof technique can be used in
the same level of generality as mine. His proof technique breaks down as \( C \)
passes \( \sqrt{1/\epsilon} \), where \( \epsilon \) is the maximum differential probability of \( h \), whereas
mine continues working until \( \sqrt{2^G} \), where \( G \) is the authenticator group. The
magnitude of the improvement depends on how far \( \epsilon \) is from \( 1/\#G \). In the
Gilbert-MacWilliams-Sloane system, for example, \( \epsilon = 1/\#G \) and there is no
improvement; in Poly1305, \( \epsilon \approx 2^8 L/\#G \) and there is a large improvement.

All of the security proofs in the literature rely on two-sided bounds for the
interpolation probabilities for \( f \). One computes lower bounds on the probability
of any particular sequence of authenticators; one computes upper bounds
on the probability that sequence of authenticators gives \( h \); one deduces that
the authenticators reveal very little information about \( h \), and hence very little
information about the authenticator for a new message. See, e.g., [22, Section
4, Theorem] and [19, Appendix A, Lemma 1]. The heart of the improvement in
this paper is a new “one-sided” proof strategy that moves directly from upper bounds for \( f \) and \( h \) to upper bounds on the attacker’s chance of success.

2 Protocol

This section describes a very general message-authentication protocol. Section 3 formalizes the notion of an attack on the protocol. Section 5 analyzes the success chance of all attacks.

The protocol has several parameters:

- \( G \), a finite commutative group of authenticators. I will always write the group operation as +. (More general groups, or even loops, would suffice, but I see no application of the extra generality.) Typical example: \( G \) is the set of 16-byte strings, with the group operation being exclusive-or. Another example: \( G \) is the set \( \{0, 1, 2, \ldots, \cdot 1_{2^{28}} - 1\} \), with the group operation being addition modulo \( 2^{28} \).
- \( M \), a nonempty set of messages. Typical example: \( M \) is the set of all strings of bytes. Another example: \( M \) is the set of all strings of at most 1024 bytes.
- \( N \), a finite set of nonces, with \( \#N \leq \#G \). Typical example: \( N \) is the set \( \{1, 2, 3, \ldots, 2^{32} - 1\} \). Another example: \( N \) is the set of 16-byte strings.

The protocol has several participants:

- A message generator creates messages.
- A nonce generator accepts messages \( m \) from the message generator and attaches a nonce \( n \) to each message \( m \). The nonce generator must never use the same nonce for two different messages: if it generates \((n_1, m_1)\) and \((n_2, m_2)\), and if \( m_1 \neq m_2 \), then \( n_1 \) must not equal \( n_2 \). This uniqueness rule is automatically satisfied if the nonce generator uses nonce 1 for the first message, nonce 2 for the second message, etc.
- A sender accepts pairs \((n, m)\) from the nonce generator and attaches an authenticator \( a \) to each pair, as discussed below.
- A network accepts a sequence of vectors \((n, m, a)\) from the sender and transmits a sequence of vectors \((n', m', a')\). Perhaps the sequence of vectors transmitted is the same as the sequence of vectors sent; perhaps not.
- A receiver receives vectors \((n', m', a')\) from the network. It accepts \((n', m')\) if \( a' \) is the authenticator that the sender would have attached to \((n', m')\); otherwise it discards \((n', m')\).

If the network transmits exactly what the sender sent, then the pairs \((n, m)\) accepted by the receiver are exactly the pairs \((n, m)\) given to the sender; but what if the network makes changes? The objective of the protocol is forgery elimination: ensuring that each pair \((n', m')\) accepted by the receiver is one of the pairs \((n, m)\) that was authenticated by the sender.

Users also want additional protocol features:
• The receiver should notice if the network repeats messages or transmits messages out of order. One standard way to do this is for the nonce generator to use increasing nonces (in some specified ordering of the set $N$), and for the receiver to discard $(n', m', a')$ unless $n'$ is larger than the last accepted nonce.

• The receiver should notice if the network loses a message. There's no way to recover if the network is losing all messages, but there are retransmission protocols that eventually succeed in transmitting all data if the network delivers (e.g.) 1% of all messages.

But this paper focuses on the cryptographic problem of forger y elimination.

The sender’s authenticator for a pair $(n, m)$ is $h(m) + f(n)$: i.e., the sender gives $(n, m, h(m) + f(n))$ to the network. Here $h$ is a random function from $M$ to $G$, and $f$ is a random function from $N$ to $G$. The pair $(f, h)$ is a secret shared by the sender and receiver; this means that the actions of the message generator, nonce generator, and network are independent of $(f, h)$. In particular, if the message generator encrypts messages, it does so using a key independent of $(f, h)$. The proof strategy in this paper can be extended to cover protocols that reuse $f$ for encryption, as long as separate $f$ inputs are used for encryption and for authentication; but that extension is not included in the statement of Theorem 5.1.

### 3 Attacks

The combined behavior of the message generator, nonce generator, and network is called an "attack." The attack creates messages; it creates nonces, subject to the rule that nonces never repeat; it inspects the authenticators provided by the sender; and it provides some number of forgery attempts to the receiver.

The network is presumed to be able to provide data to the message generator and nonce generator, so each message can depend on previous authenticators. The attacker is presumed to be able to tell whether the receiver has accepted a forgery attempt.

More formally: An **attack** is an algorithm given oracle access to two functions $S$ and $R$. The algorithm feeds **chosen messages** to the first oracle:

• The algorithm chooses a nonce $n_1$ and message $m_1$. The algorithm issues the query $(n_1, m_1)$ and receives an authenticator $a_1 = S(n_1, m_1)$.

• The algorithm then chooses a nonce $n_2$ and message $m_2$, obeying the rule that $n_2 \neq n_1$ if $m_2 \neq m_1$. The algorithm issues the query $(n_2, m_2)$ and receives an authenticator $a_2 = S(n_2, m_2)$.

• The algorithm then chooses a nonce $n_3$ and message $m_3$, obeying the rule that $n_3 \neq n_1$ if $m_3 \neq m_1$, and the rule that $n_3 \neq n_2$ if $m_3 \neq m_2$. The algorithm issues the query $(n_3, m_3)$ and receives an authenticator $a_3 = S(n_3, m_3)$.

• The algorithm continues in this way for any number of messages.
Meanwhile, the algorithm feeds any number of forgery attempts \((n', m', a')\) to the second oracle, receiving responses \(R(n', m', a')\).

The attack succeeds against \(S\) and \(R\) if at least one forgery attempt \((n', m', a')\) has \(R(n', m', a') = 1\) with \((n', m')\) different from the previous queries \((n_1, m_1), (n_2, m_2), \ldots\) to the first oracle.

**Attacks against this system**

Take, in particular, 
\[
S(n, m) = h(m) + f(n) \quad \text{and} \quad R(n, m, a) = |a - h(m) + f(n)|;
\]
i.e., 
\[
R(n, m, a) = 1 \text{ if } a = h(m) + f(n) \quad \text{and} \quad R(n, m, a) = 0 \text{ if } a \neq h(m) + f(n).
\]
Is there an attack that succeeds against \(S\) and \(R\) with noticeable probability?

Theorem 5.1 states, under certain assumptions on \(f\) and \(h\), that the answer is no. The receiver is overwhelmingly likely to discard every forgery—no matter how the message generator chooses messages; no matter how the nonce generator chooses unique nonces; no matter how the network chooses forgeries.

The rest of this section discusses the strength of this theorem, under the same assumptions on \(f\) and \(h\).

**Forgeries versus selective forgeries**

A selective forgery is a forged message chosen in advance by the attacker. Some protocols prevent selective forgeries but allow attackers to find authenticators for random-looking messages. These protocols assume—often incorrectly—that random-looking messages will not cause any damage. In contrast, \(h(m) + f(n)\) rejects all forgeries.

**Attacks versus blind attacks**

Some protocols prevent blind attacks but allow forgeries when attackers can inspect authenticated messages. (Trivial example: use a secret password as an authenticator for every message.) In contrast, \(h(m) + f(n)\) rejects all forgeries even after the attacker sees a large number of authenticated messages.

**Chosen messages versus known messages**

Some protocols are secure for some message generators but insecure for others. An attacker who can influence the message generator can often obtain enough information to forge messages. In contrast, \(h(m) + f(n)\) rejects all forgeries no matter what the message generator does.

Of course, if an attacker can convince the message generator to produce a message, then he does not need to forge an authenticator for that message. An easily corrupted message generator is often a problem. It is, however, not the cryptographic problem considered in this paper.
Receiver interaction

As pointed out by Bellare, Goldreich, and Mitaygin in [3], some (admittedly unrealistic) protocols are secure against an attacker carrying out a single forgery attempt but insecure against an attacker that tries several forgery attempts. In contrast, the security bound for $h(m) + f(n)$ is linear in the number of forgery attempts.

The crucial point, as emphasized by Bellare et al., is that the attacker can recognize all $(m', m', d')$ that will be accepted by the receiver without being forgeries: namely, the results $(n_1, n_2, a_1), (n_2, n_2, a_2), \ldots$ already obtained from the sender. In other words, the only way an attacker can learn anything new from the receiver is by succeeding at a forgery. Thus receiver interaction does not improve the attacker’s success probability. Receiver interaction can change the number of successful forgeries if the attacker succeeds, but this paper guarantees that the attacker will not succeed in the first place.

4 Interpolation probabilities

Let $f$ be a random function from $N$ to $G$. The hypothesis on $f$ in Section 5 is that $f$ has maximum $k$-interpolation probability on the scale of $1/\#G^k$, for various $k \in \{0, 1, \ldots, \#N\}$. Here the maximum $k$-interpolation probability of $f$ is the maximum, for all $x_1, x_2, \ldots, x_k \in G$ and all distinct $n_1, n_2, \ldots, n_k \in N$, of the probability that $(f(n_1), f(n_2), \ldots, f(n_k)) = (x_1, x_2, \ldots, x_k)$.

This section proves that this condition is satisfied by a uniform random function and by a uniform random injective function.

**Theorem 4.1.** Let $f$ be a uniform random function from a finite set $N$ to a finite set $G$. Assume that $\#N \leq \#G$. Then $f$ has maximum $k$-interpolation probability $1/\#G^k$ for each $k \in \{0, 1, \ldots, \#N\}$.

**Proof.** $(f(n_1), f(n_2), \ldots, f(n_k)) = (x_1, x_2, \ldots, x_k)$ with probability $1/\#G^k$. □

**Theorem 4.2.** Let $f$ be a uniform random injective function from a finite set $N$ to a finite set $G$. Assume that $\#N \leq \#G$. Then $f$ has maximum $k$-interpolation probability at most $(1 - (k - 1)/\#G)^{k/2}/\#G^k$ for each $k \in \{0, 1, \ldots, \#N\}$.

**Proof.** Fix distinct $n_1, n_2, \ldots, n_k \in N$. Fix $x_1, x_2, \ldots, x_k \in G$.

Case 1: There are collisions in $x_1, x_2, \ldots, x_k$. Then $(f(n_1), \ldots, f(n_k)) = (x_1, \ldots, x_k)$ with probability 0.

Case 2: There are no collisions. Then $f(n_1) = x_1$ with probability $1/\#G$; if that happens then $f(n_2) = x_2$ with conditional probability $1/(\#G - 1)$; if that happens then $f(n_3) = x_3$ with conditional probability $1/(\#G - 2)$; and so on. The probability that $(f(n_1), f(n_2), \ldots, f(n_k)) = (x_1, x_2, \ldots, x_k)$ is exactly $\prod_{0 \leq i \leq k-1} 1/(\#G - i) = \sqrt{\prod_{0 \leq i \leq k-1} 1/((\#G - i)\#G - (k - 1 - i))} \leq \sqrt{\prod_{0 \leq i \leq k-1} 1/((\#G)^2(1 - (k - 1)\#G) = \sqrt{(1 - (k - 1)\#G)^{-k}}/(\#G)^{2k}}$. □
5 The main theorem

Theorem 5.1 is the main theorem of this paper: \((n, m) \mapsto h(m) + f(n)\) is secure if \(h\) has small differential probabilities and \(f\) has small interpolation probabilities.

Theorems 5.2 and 5.3 consider two special cases: a uniform random function \(f\), and a uniform random injective function \(f\).

Theorem 5.4 proves that \((n, m) \mapsto h(m) + \text{AES}_k(n)\) is secure if \(h\) has small differential probabilities and \(\text{AES}_k\) is secure, i.e., \(\text{AES}_k\) is difficult to distinguish from a uniform random injective function.

**Theorem 5.1.** Let \(h\) be a random function from a nonempty set \(M\) to a finite commutative group \(G\). Let \(f\) be a random function from a finite set \(N\) to \(G\). Let \(C\) and \(D\) be positive integers. Assume that \(C + 1 \leq \#N \leq \#G\). Assume, for all \(g \in G\) and all distinct \(m, m' \in M\), that \(h(m) = h(m') + g\) with probability at most \(\epsilon\). Assume that \(f\) has maximum \(C\)-interpolation probability at most \(\delta / \#G^C\) and maximum \((C + 1)\)-interpolation probability at most \(\delta / \#G^C\). Assume that \(h\) and \(f\) are independent. Then any attack using at most \(C\) distinct chosen messages and at most \(D\) forgery attempts succeeds against \((n, m) \mapsto h(m) + f(n)\) and \((n, m, a) \mapsto [a = h(m) + f(n)]\) with probability at most \(D \epsilon\).

**Proof.** Standard reduction \#1: It suffices to consider \(D = 1\), for the following reason. For \(D > 1\), split the attack into two pieces:

- The first piece is the original attack with one change: it stops immediately after the first forgery attempt (if there are any forgery attempts). This piece uses at most \(C\) distinct chosen messages and at most 1 forgery attempt.
- The second piece is the original attack with one change: it simulates the first forgery attempt internally (if there are any forgery attempts) rather than sending the forgery attempt as an oracle query. The simulator returns 1 if the forgery attempt \(n', m', a'\) matches an authenticator \(a'\) already provided in response to a chosen message \(m'\) with nonce \(n'\); otherwise the simulator returns 0. This piece uses at most \(C\) distinct chosen messages and at most \(D - 1\) forgery attempts.

Success of the original attack on its first forgery attempt is equivalent to success of the first piece—which occurs with probability at most \(\delta \epsilon\). Failure of the original attack on its first forgery attempt, but success on a subsequent attempt, implies success of the second piece—which, by induction on \(D\), occurs with probability at most \((D - 1) \delta \epsilon\). Therefore the original attack succeeds with probability at most \(D \delta \epsilon\).

Standard reduction \#2: If there are no forgery attempts then the attack succeeds with probability \(0 \leq \delta \epsilon\). Assume from now on that there is exactly one forgery attempt.

Standard reduction \#3: If the attack chooses any messages after the forgery attempt, modify it to discard those oracle queries; this has no effect on the attack’s success chance. Assume from now on that all chosen messages are issued before the forgery attempt.
Standard reduction #4: If the attack might use fewer than \( C \) distinct chosen messages, modify it to use additional chosen messages with new nonces and to discard the results; new nonces are available since \( \#N \geq C \), and at least one message is available since \( \#M \geq 1 \). Assume from now on that the attack uses exactly \( C \) distinct chosen messages.

Standard reduction #5: If the attack might repeat chosen messages, modify it to cache queries and responses to the sender oracle. Assume from now on that the attack does not repeat chosen messages.

Write \((n_i, m_i)\) for the \(i\)th query to the sender oracle. Then \(n_1, n_2, \ldots, n_C\) are distinct. Write \(a_i\) for the \(i\)th response from the oracle, when the attack is applied to \((n, m) \mapsto h(m) + f(n)\); then \(a_i = h(m_i) + f(n_i)\). Write \((n', m', a')\) for the attempted forgery.

Everything that the attack does is determined by (1) an infinite sequence \( b \) of coin flips, by definition independent of \(h\) and \(f\), and (2) the sequence of sender responses \(a_1, a_2, \ldots, a_C\). In particular, \(n_1, n_2, \ldots, n_C, m_1, m_2, \ldots, m_C, n', m', a'\) are equal to various functions evaluated at \(b, a_1, a_2, \ldots, a_C\). Furthermore, \(f(n_i)\) is determined by \(a_i\) and \(h(m_i)\), so \(f(n_i)\) equal to a function evaluated at \(h, b, a_1, a_2, \ldots, a_C\).

Fix \((g_1, g_2, \ldots, g_C) \in G^C\). Define \(X\) as the following event: \((n', m', a')\) is a successful forgery and \((a_1, a_2, \ldots, a_C) = (g_1, g_2, \ldots, g_C)\). It suffices to show that event \(X\) has probability at most \(\delta e/\#G^C\).

(A referee suggests some added emphasis: I am considering the probability that the forgery attempt succeeds and the authenticators match \((g_1, g_2, \ldots, g_C)\). Previous proofs considered the probability that the forgery attempt succeeds given that the authenticators match \((g_1, g_2, \ldots, g_C)\).)

Define \(p\) as the probability that \(b\) satisfies the following measurable constraint: if \((a_1, a_2, \ldots, a_C) = (g_1, g_2, \ldots, g_C)\) then \(n' \notin \{n_1, n_2, \ldots, n_C\}\). I claim, for each \(b\) satisfying the constraint and for each \(h\), that \(f\) has conditional probability at most \(\delta e/\#G^C\) of producing event \(X\).

Indeed, assume that \(b\) satisfies the constraint, that \((n', m', a')\) is a successful forgery, and that \((a_1, a_2, \ldots, a_C) = (g_1, g_2, \ldots, g_C)\). Then \(\#\{n_1, \ldots, n_C, n'\} = C + 1\), and the pairs \((n_1, f(n_1)), \ldots, (n_C, f(n_C)), (n', f(n'))\) are equal to various functions evaluated at \(h, b, g_1, g_2, \ldots, g_C\). By hypothesis, \(f\) is independent of \(h\); \(f\) is also independent of \(b\); and \(g_1, g_2, \ldots, g_C\) are fixed. The conditional probability of \(f\) interpolating those pairs is at most the maximum \((C + 1)\)-interpolation probability of \(f\), which by hypothesis is at most \(\delta e/\#G^C\).

I also claim, for each \(b\) not satisfying the constraint, that \(h\) has conditional probability at most \(\delta e\) of satisfying a necessary differential condition; and, for each \(b\) and each qualifying \(h\), that \(f\) has conditional probability at most \(\delta e/\#G^C\) of producing event \(X\).

Indeed, assume that \(b\) does not satisfy the constraint, that \((a_1, a_2, \ldots, a_C) = (g_1, g_2, \ldots, g_C)\), and that \((n', m', a')\) is a successful forgery. Then \(n' = n_i\) for a unique \(i\); note that \(m' \neq m_i\). Next \(a' = h(m') + f(n_i)\) and \(a_i = h(m_i) + f(n_i)\) so \(h(m_i) - h(m') = a_i - a'\). The inputs \(m_i, m'\) and the output \(a_i - a'\) are equal to various functions evaluated at \(b, g_1, g_2, \ldots, g_C\), and thus are independent of \(h\); by
hypothesis, \( h \) satisfies the condition \( h(m_i) - h(m'_i) = a_i - a'_i \) with probability at most \( \epsilon \). Furthermore, the pairs \( (a_1, f(n_1)), (a_2, f(n_2)), \ldots, (a_C, f(n_C)) \) are equal to various functions evaluated at \( a, b, g_1, g_2, \ldots, g_C \); \( f \) is once again independent of \( h, b, g_1, g_2, \ldots, g_C \); so the conditional probability of \( f \) interpolating those pairs is at most the maximum \( C \)-interpolation probability of \( f \), which by hypothesis is at most \( \delta / \#G^C \).

The total probability of event \( X \) is at most \( p(\delta \epsilon / \#G^C) + (1-p)(\epsilon)(\delta / \#G^C) = \delta \epsilon / \#G^C \). \( \Box \)

**Theorem 5.2.** Let \( h \) be a random function from a nonempty set \( M \) to a finite commutative group \( G \). Let \( f \) be a uniform random function from a finite set \( N \) to \( G \). Let \( C \) and \( D \) be positive integers. Assume that \( C + 1 \leq \#N \leq \#G \). Assume, for all \( g \in G \) and all distinct \( m, m' \in M \), that \( h(m) = h(m') + g \) with probability at most \( \epsilon \). Assume that \( h \) and \( f \) are independent. Assume that \( \epsilon \geq 1 / \#G \). Then any attack using at most \( C \) distinct chosen messages and at most \( D \) forgery attempts succeeds against \( (n, m) \mapsto h(m) + f(n) \) and \( (n, m, a) \mapsto [a = h(m) + f(n)] \) with probability at most \( D \delta \epsilon = D \epsilon \).

Proof. Write \( \delta = 1 \). Then \( f \) has maximum \( C \)-interpolation probability \( 1 / \#G^C = \delta / \#G^C \), and maximum \((C + 1)\)-interpolation probability \( 1 / \#G^{C+1} \leq \delta \epsilon / \#G^C \), by Theorem 4.1. By Theorem 5.1, the attack succeeds with probability at most \( D \delta \epsilon = D \epsilon \). \( \Box \)

**Theorem 5.3.** Let \( h \) be a random function from a nonempty set \( M \) to a finite commutative group \( G \). Let \( f \) be a uniform random injective function from a finite set \( N \) to \( G \). Let \( C \) and \( D \) be positive integers. Assume that \( C + 1 \leq \#N \leq \#G \). Assume, for all \( g \in G \) and all distinct \( m, m' \in M \), that \( h(m) = h(m') + g \) with probability at most \( \epsilon \). Assume that \( h \) and \( f \) are independent. Assume that \( \epsilon \geq 1 / \#G \). Then any attack using at most \( C \) distinct chosen messages and at most \( D \) forgery attempts succeeds against \( (n, m) \mapsto h(m) + f(n) \) and \( (n, m, a) \mapsto [a = h(m) + f(n)] \) with probability at most \( D(1 - (C / \#G)^{(C+1)/2}) \epsilon \).

In the special case \( C = \lceil \sqrt{\#G} \rceil \), the extra factor \( (1-C / \#G)^{(C+1)/2} \) is below 1.7 for all reasonably large \( G \); it converges to \( \exp(1/2) \approx 1.6482 \) as \( \#G \to \infty \).

Proof. Write \( \delta = (1-C / \#G)^{(C+1)/2} \). By Theorem 4.2, \( f \) has maximum \( C \)-interpolation probability at most \( (1 - (C - 1) / \#G)^{(C+2)/2} / \#G^C \leq \delta / \#G^C \). By Theorem 4.2 again, \( f \) has maximum \((C + 1)\)-interpolation probability at most \( (1 - C / \#G)^{(C+1)/2} / \#G^{C+1} \leq \delta \epsilon / \#G^C \). By Theorem 5.1, the attack succeeds with probability at most \( D \delta \epsilon \). \( \Box \)

**Theorem 5.4.** Let \( G \) be the set of 16-byte strings with a group structure. Let \( k \) be a random AES key. Let \( h \) be a random function from a nonempty set \( M \) to \( G \). Assume that the distribution of \( h \) is computable. Let \( C \) and \( D \) be positive integers.
Assume that $C + 1 \leq 2^{128}$. Assume, for all $g \in G$ and all distinct $m, m' \in M$, that $h(m) = h(m') + g$ with probability at most $\varepsilon$. Assume that $h$ and $k$ are independent. Assume that $\varepsilon \geq 1/2^{128}$. Let $A$ be an attack using at most $C$ distinct chosen messages and at most $D$ forgery attempts. Assume that $A$ succeeds against $(n, m) \mapsto h(m) + \text{AES}_k(n)$ and $(n, m, a) \mapsto [a = h(m) + \text{AES}_k(n)]$ with probability $\gamma$. Define $A'$ as the algorithm that, given an oracle for a function $f$, chooses $h$ randomly, applies $A$ to $(n, m) \mapsto h(m) + f(n)$ and $(n, m, a) \mapsto [a = h(m) + f(n)]$, and prints 1 if $A$ succeeded. Then $A'$ distinguishes $\text{AES}_k$ from a uniform random permutation of $G$ with probability at least $\gamma - D - 1 - C/2^{2128})^{-1/2}$. using at most $C + D$ oracle queries.

Consequently, $A$ succeeds against $(n, m) \mapsto h(m) + \text{AES}_k(n)$ and $(n, m, a) \mapsto [a = h(m) + \text{AES}_k(n)]$ with probability at most $\delta + D - 1 - C/2^{2128})^{-1/2}$, where $\delta$ is the probability that an algorithm as fast as $A'$ can distinguish $\text{AES}_k$ from a uniform random permutation of $G$.

Proof. $A'$ makes one oracle query for each chosen message from $A$, and one oracle query for each attempted forgery from $A$, for a total of at most $C + D$ oracle queries.

When $A'$ is given an oracle for $\text{AES}_k$, it applies $A$ to $(n, m) \mapsto h(m) + \text{AES}_k(n)$ and $(n, m, a) \mapsto [a = h(m) + \text{AES}_k(n)]$, so it prints 1 with probability $\gamma$ by hypothesis.

When $A'$ is given an oracle for a uniform random permutation $f$ of $G$, it applies $A$ to $(n, m) \mapsto h(m) + f(n)$ and $(n, m, a) \mapsto [a = h(m) + f(n)]$, so it prints 1 with probability at most $D - 1 - C/2^{2128})^{-1/2}$ by Theorem 5.3.

Therefore $A'$ distinguishes $\text{AES}_k$ from a uniform random permutation of $G$ with probability at least $\gamma - D - 1 - C/2^{2128})^{-1/2}$.

\section*{References}

4. Daniel J. Bernstein, Floating-point arithmetic and message authentication, to be incorporated into author’s High-speed cryptography book. URL: http://cr.yp.to/papers.html#hash127. ID dabadd3095644704c5be9690e37388e.


A Appendix: General terminology

This section discusses various conflicts between (1) the terminology used in some cryptographic papers and (2) the standard terminology of probability theory used by a much larger community of mathematicians. This section also discusses my choice of terminology for a few concepts that are more specialized.

“Random” versus “uniform random”

A random element $v$ of a finite set $S$ is **uniform** if $\Pr[v = s] = 1/\#S$ for each $s \in S$. This terminology is standard in probability theory and is used in this paper.

Examples: A coin flip $c$—a random bit—is uniform if $\Pr[c = 0] = 1/2$ and $\Pr[c = 1] = 1/2$. If $k$ is a uniform random 16-byte string then $(k, 0)$ is a non-uniform random 17-byte string; $k[0]$, the first byte of $k$, is a uniform random byte; AES$_k$ is a non-uniform random permutation of the set of 16-byte strings.

Some cryptographic papers use the word “random” to mean uniform random. If these papers state theorems regarding a “random element of $S$,” for example, then those theorems don’t apply to a random element of $\{0, 1, 2, \ldots, 2^{127} - 2\}$ that’s 0 twice as often as anything else—even though this slightly non-uniform distribution is much more widely used than the uniform distribution. If these papers state theorems regarding a “random RSA key” then those theorems are incompatible with every prime-generation algorithm that’s actually used.

Some people try to work around this terminological deficiency by viewing all random variables as images of uniform random variables. My random element of $\{0, 1, 2, \ldots, 2^{127} - 2\}$ might be viewed as the reduction modulo $2^{127} - 1$ of a uniform random element of $\{0, 1, 2, \ldots, 2^{256} - 1\}$; maybe this is how it was generated in the first place.

The big problem with this workaround is that it buries every random variable in a pointless thicket of notation—one has to introduce an irrelevant input set and an irrelevant function instead of simply focusing on the resulting variable. For example, rather than simply stating a theorem for a random function $h$ from messages to 16-byte strings, I’d have to state a theorem for a (uniform) random element $r$ of some set $R$ together with some function $H$ that, for each $r$, gives me a function from messages to 16-byte strings.

“Random” versus “discrete random”

The set of values of a random variable is not required to be finite, or even countable. A **discrete** random variable is a random element of a countable set. This terminology is standard in probability theory and is used in this paper.

Consider, for example, the coin flips provided as an auxiliary input to a probabilistic algorithm: an infinite random sequence of bits $b = (b_1, b_2, b_3, \ldots)$. This random sequence is a non-discrete random variable; it has uncountably many values.
As a concrete example, consider the usual probabilistic algorithm to generate a uniform random element of \( \{1, 2, 3\} \): flip two coins; if the results are \((0, 1)\) or \((1, 0)\) or \((1, 1)\), print \(1\) or \(2\) or \(3\) correspondingly and stop; otherwise try again. It is easy to prove that this algorithm succeeds with probability \(1\) (i.e., that \( b = (0,0,0,0,\ldots) \) with probability \(0\)), that the algorithm flips \(8/3\) coins on average, etc. These are statements about non-discrete probabilities.

Some cryptographic papers use the word “random” to mean discrete random, thus excluding such fundamental objects as coin-flip sequences. This restriction is intolerable for any serious discussion of probabilistic algorithms.

Warning to undergraduates: \( \Pr[b \in S] \), the probability that \( b \) is in \( S \), is defined only for some sets \( S \), namely the measurable sets. Here are the rules:

1. The empty set is measurable;
2. If \( S \) is measurable then its complement \( \overline{S} \) is measurable;
3. If \( S_1, S_2, \ldots \) are measurable then \( S_1 \cup S_2 \cup \cdots \) is measurable;
4. If \( S \) is measurable then \( \Pr[b \in S] \geq 0 \);
5. If \( S \) is measurable then \( \Pr[b \in S] + \Pr[b \in \overline{S}] = 1 \);
6. \( \Pr[b \in S_1 \cup S_2 \cup \cdots] = \Pr[b \in S_1] + \Pr[b \in S_2] + \cdots \) if \( S_1, S_2, \ldots \) are disjoint measurable sets;
7. If \( u_1, \ldots, u_k \) are bits then \( \{ b: (b_1, \ldots, b_k) = (u_1, \ldots, u_k) \} \) is measurable and \( \Pr[(b_1, \ldots, b_k) = (u_1, \ldots, u_k)] = 1/2^k \);
8. Nothing is measurable except as guaranteed above.

“Random” versus “independent random”

Random variables \( u, v, w \) are independent if the distribution of \((u, v, w)\) is the product of the distribution of \( u \), the distribution of \( v \), and the distribution of \( w \): i.e., \( \Pr[(u, v, w) \in A \times B \times C] = \Pr[u \in A] \Pr[v \in B] \Pr[w \in C] \) for all measurable sets \( A, B, C \). This terminology is standard in probability theory and is used in this paper.

For example, if \( k \) is a uniform random 16-byte string, then \( k[0], k[1], \) and \( k[2] \) are independent uniform random bytes; \( k[0], k[1], \) and \( k[0] \oplus k[1] \) are non-independent uniform random bytes; \( k[0] \) and \( k[0] \oplus k[1] \) are independent uniform random bytes; \( (k[0], 0) \) and \( (k[1], 0) \) are independent non-uniform random 2-byte strings.

Suppose Theorem X says “Let \( u \) and \( v \) be independent random bytes. Then \( u \) and \( v \) satisfy ….” I can apply Theorem X to the independent random bytes \( k[1], k[2] \). I can apply it to the independent random bytes \( k[0], k[0] \oplus k[1] \). I simply have to say “\( k[0] \) and \( k[0] \oplus k[1] \) are independent; therefore, by Theorem X, \( k[0] \) and \( k[0] \oplus k[1] \) satisfy ….”

Some cryptographic papers omit the word “independent” in many situations. For example, Theorem X would say “Let \( u \) and \( v \) be random bytes. Then \( u \) and \( v \) satisfy ….” implicitly also requiring that \( u \) and \( v \) be independent. The scope of this implicit independence is unclear to me: for example, if the proof begins “Note that if \( r \) is a random byte then ….” then is \( r \) implicitly required to be independent of \( u \) and \( v \)? The obvious way to avoid confusion is to make all independence assumptions explicit.
Distributions versus random variables

What is a random variable?

A random element \( v \) of \( X \) is, intuitively, a function to \( X \) from the set of possible universes. The value that \( v \) takes in a possible universe \( u \) is exactly the function value \( v(u) \). For example, a coin flip is a function that assigns 0 to some possible universes and 1 to other possible universes.

To formalize this, we fix a probability space \( \text{Pr} \)—intuitively, the set of possible universes, although this intuition does not constrain the definition. A random element of \( X \), where \( X \) is a measurable space, is a measurable function from \( \text{Pr} \) to \( X \). This terminology is standard in probability theory and is used in this paper.

(Notes for undergraduates: A measurable space is a set together with a designated collection of measurable subsets satisfying rules 1, 2, 3 above. A probability space is a set together with a designated collection of measurable subsets \( S \) and probabilities \( \text{Pr}[S] \) satisfying rules 1, 2, 3, 4, 5, 6 above. A function \( v \) is measurable if \( \{ u \in \text{Pr} : v(u) \in S \} \) is measurable for every measurable set \( S \).)

Random variables can be combined to produce new random variables. For example, if \( v \) is a random element of \( X \) and \( w \) is a random element of \( Y \) then \( (v, w) \), the function that takes \( u \) to \( (v(u), w(u)) \), is a random element of \( X \times Y \). Similarly, if \( \varphi \) is a measurable function from \( X \) to \( Y \), then \( \varphi(v) \), the function that takes \( u \) to \( \varphi(v(u)) \), is a random element of \( Y \).

Consider, for example, the measurable function \( s \mapsto s[0] \) that extracts the first byte of a 16-byte string. This function induces a function \( k \mapsto k[0] \) that extracts a random byte from a random 16-byte string: the composition of \( k \) (a function from \( \text{Pr} \) to 16-byte strings) with \( [0] \) (a function from 16-byte strings to bytes) is \( k[0] \) (a function from \( \text{Pr} \) to bytes).

Some cryptographers forbid all use of random variables. For example, one is forbidden from considering a random function \( f \) and defining the maximum 2-interpolation probability of \( f \) as the maximum, over all \( x_1, x_2 \) and all distinct \( n_1, n_2 \), of \( \text{Pr}[(f(n_1), f(n_2)) = (x_1, x_2)] \). One is forced to use distributions instead: consider a distribution \( F \) on the set of functions, and define the maximum 2-interpolation probability of \( F \) as the maximum, over all \( x_1, x_2 \) and all distinct \( n_1, n_2 \), of the fraction of functions \( f \) in \( F \) such that \( (f(n_1), f(n_2)) = (x_1, x_2) \).

The most glaring deficiency in this approach is its inability to discuss the dependence of separate random variables. The independence of \( p \) and \( q \) is not determined by the distribution of \( p \) and the distribution of \( q \); in any situation where \( p \) and \( q \) might be dependent, these papers are forced to start from the distribution of the pair \( (p, q) \). How, then, does one feed \( p \) by itself to a lower-level theorem? One has to average the joint distribution to obtain the distribution of \( p \), then apply the lower-level theorem, then undo the averaging; or generalize the lower-level theorem to allow a joint distribution as input—effectively reinventing the concept of random variables but with a much less pleasant notation.
Interpolation probabilities, collision probabilities, differential probabilities

The following concepts are much more specialized than the fundamental concepts of probability theory discussed earlier in this appendix. They are nevertheless sufficiently common that the community would benefit from settling on good terminology for them.

Let \( f \) be a random function from a set \( X \) to a finite set \( Y \). Consider the probability that \( f \) interpolates the points \((x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\), where \( x_1, x_2, \ldots, x_k \) are distinct: i.e., that \((f(x_1), f(x_2), \ldots, f(x_k)) = (y_1, y_2, \ldots, y_k)\). This is what I call an interpolation probability, and more specifically a \( k \)-interpolation probability.

Now consider the sum of 2-interpolation probabilities along the diagonal: fix distinct \( x_1, x_2 \in X \), and consider the probability that \( f(x_1) = f(x_2) \). This is what I call a collision probability. More generally, assume that \( Y \) is a commutative group, fix \( g \in Y \), and consider the probability that \( f(x_1) = f(x_2) = g \). This is what I call a differential probability.

In the context of message authentication, it is useful to have upper bounds on interpolation probabilities and upper bounds on differential probabilities, as illustrated by this paper. Other authenticators can use upper bounds on collision probabilities. It is also useful to have lower bounds on interpolation probabilities, as illustrated by [6].

Many papers write “\( h \) is \( \epsilon \)-almost-universal” (sometimes replacing \( h \) by its distribution) to mean that all collision probabilities of \( h \) are below \( \epsilon \). There are several flaws in this “\( \epsilon \)-almost-universal” terminology:

- The phrase “almost universal” is highly non-descriptive. Readers who have not seen the definition cannot even begin to guess what it refers to. Readers who have seen the definition need to expend unnecessary mental energy to remember it.
- The phrase “almost universal” provides no way to refer to individual collision probabilities, lower bounds for collision probabilities, etc. The same papers often end up talking about “collision probabilities” anyway.
- The phrase “almost universal” begs the question of what “almost” refers to. The answer is that “\( h \) is universal” is reserved for the special case \( \epsilon = 1/\#Y \), which is close to (although not exactly) the minimum achievable.

This terminology misleads readers into believing that the special case is important; in fact, the general case is far more important.

Some papers on Wegman-Carter authenticators use the phrase “\( h \) is \( \epsilon \)-almost-xor-universal” to mean that all differential probabilities of \( h \) are below \( \epsilon \), in the special case of the group operation being \( \oplus \). Similar criticisms apply to this phrase. A few papers refer to the general case using the phrase “\( h \) is \( \epsilon \)-almost-\( \Delta \)-universal,” which is a poor choice of terminology for yet another reason: the letter \( \Delta \) is part of the terminology, not a modifiable variable name.