Guidelines for Cryptographic Algorithm Agility
and Selecting Mandatory-to-Implement Algorithms

Abstract

   Many IETF protocols use cryptographic algorithms to provide
   confidentiality, integrity, authentication, or digital signature.
   Communicating peers must support a common set of cryptographic
   algorithms for these mechanisms to work properly. This memo provides
   guidelines to ensure that protocols have the ability to migrate from
   one mandatory-to-implement algorithm suite to another over time.

Status of This Memo

   This memo documents an Internet Best Current Practice.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   BCPs is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7696.

Table of Contents

1. Introduction

   Many IETF protocols use cryptographic algorithms to provide
   confidentiality, integrity, authentication, or digital signature.
   For interoperability, communicating peers must support a common set
   of cryptographic algorithms. In most cases, a combination of
   compatible cryptographic algorithms will be used to provide the
   desired security services. The set of cryptographic algorithms being
   used at a particular time is often referred to as a cryptographic
   algorithm suite or cipher suite. In a protocol, algorithm
   identifiers might name a single cryptographic algorithm or a full
   suite of algorithms.

   Cryptographic algorithms age; they become weaker with time. As new
   cryptanalysis techniques are developed and computing capabilities
   improve, the work required to break a particular cryptographic
   algorithm will reduce, making an attack on the algorithm more
   feasible for more attackers. While it is unknown how cryptoanalytic
   attacks will evolve, it is certain that they will get better. It is

unknown how much better they will become or when the advances will happen. Protocol designers need to assume that advances in computing power or advances in cryptoanalytic techniques will eventually make any algorithm obsolete. For this reason, protocols need mechanisms to migrate from one algorithm suite to another over time.

Algorithm agility is achieved when a protocol can easily migrate from one algorithm suite to another more desirable one, over time. For the protocol implementer, this means that implementations should be modular to easily accommodate the insertion of new algorithms or suites of algorithms. Ideally, implementations will also provide a way to measure when deployed implementations have shifted away from the old algorithms and to the better ones. For the protocol designer, algorithm agility means that one or more algorithm or suite identifiers must be supported, the set of mandatory-to-implement algorithms will change over time, and an IANA registry of algorithm identifiers will be needed.

Algorithm identifiers by themselves are not sufficient to ensure easy migration. Action by people that maintain implementations and operate services is needed to develop, deploy, and adjust configuration settings to enable the new more desirable algorithms and to deprecate or disable older, less desirable ones. For various reasons, most notably interoperability concerns, experience has shown that it has proven difficult for implementers and administrators to remove or disable weak algorithms. Further, the inability of legacy systems and resource-constrained devices to support new algorithms adds to those concerns. As a result, people live with weaker algorithms, sometimes seriously flawed ones, well after experts recommend migration.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Algorithm Agility Guidelines

These guidelines are for use by IETF working groups and protocol authors for IETF protocols that make use of cryptographic algorithms. Past attempts at algorithm agility have not been completely successful, and this section provides some insights from those experiences.

2.1. Algorithm Identifiers

   IETF protocols that make use of cryptographic algorithms MUST support
   one or more algorithms or suites. The protocol MUST include a
   mechanism to identify the algorithm or suite that is being used. An
   algorithm identifier might be explicitly carried in the protocol.
   Alternatively, a management mechanism can be used to identify the
   algorithm. For example, an entry in a key table that includes a key
   value and an algorithm identifier might be sufficient.

   If a protocol does not carry an algorithm identifier, then the
   protocol version number or some other major change is needed to
   transition from one algorithm to another. The inclusion of an
   algorithm identifier is a minimal step toward cryptographic algorithm
   agility.

   Sometimes a combination of protocol version number and explicit
   algorithm or suite identifiers is appropriate. For example, the
   Transport Layer Security (TLS) [RFC5246] version number names the
   default key derivation function, and the cipher suite identifier
   names the rest of the needed algorithms.

   Some approaches carry one identifier for each algorithm that is used.
   Other approaches carry one identifier for a full suite of algorithms.
   Both approaches are used in IETF protocols. Designers are encouraged
   to pick one of these approaches and use it consistently throughout
   the protocol or family of protocols. Suite identifiers make it
   easier for the protocol designer to ensure that the algorithm
   selections are complete and compatible for future assignments.
   However, suite identifiers inherently face a combinatoric explosion
   as new algorithms are defined. Algorithm identifiers, on the other
   hand, impose a burden on implementations by forcing a determination
   at run-time regarding which algorithm combinations are acceptable.

   Regardless of the approach used, protocols historically negotiate the
   symmetric cipher and cipher mode together to ensure that they are
   compatible.

   In the IPsec protocol suite, the Internet Key Exchange Protocol
   version 2 (IKEv2) [RFC7296] carries the algorithm identifiers for the
   Authentication Header (AH) [RFC4302] and the Encapsulating Security
   Payload (ESP) [RFC4303]. Such separation is a completely fine design
   choice. In contrast, TLS [RFC5246] carries cipher suite identifiers,
   which is also a completely fine design choice.

An IANA registry SHOULD be used for these algorithm or suite
identifiers. Once an algorithm identifier is added to the registry,
it should not be changed or removed. However, it is desirable to
mark a registry entry as deprecated when implementation is no longer
advisable.

2.2. Mandatory-to-Implement Algorithms

For secure interoperability, BCP 61 [RFC3365] recognizes that
communicating peers that use cryptographic mechanisms must support a
common set of strong cryptographic algorithms. For this reason, IETF
protocols that employ cryptography MUST specify one or more strong
mandatory-to-implement algorithms or suites. This does not require
all deployments to use this algorithm or suite, but it does require
that it be available to all deployments.

The IETF needs to be able to change the mandatory-to-implement
algorithms over time. It is highly desirable to make this change
without updating the base protocol specification. To achieve this
goal, it is RECOMMENDED that the base protocol specification includes
a reference to a companion algorithms document, allowing the update
of one document without necessarily requiring an update to the other.
This division also facilitates the advancement of the base protocol
specification on the standards maturity ladder even if the algorithm
document changes frequently.

The IETF SHOULD keep the set of mandatory-to-implement algorithms
small. To do so, the set of algorithms will necessarily change over
time, and the transition SHOULD happen before the algorithms in the
current set have weakened to the breaking point.

2.2.1. Platform Specifications

Note that mandatory-to-implement algorithms or suites are not
specified for protocols that are embedded in other protocols; in
these cases, the system-level protocol specification identifies the
mandatory-to-implement algorithm or suite. For example, S/MIME
[RFC5751] makes use of the cryptographic message Syntax (CMS)
[RFC5652], and S/MIME specifies the mandatory-to-implement
algorithms, not CMS. This approach allows other protocols to make
use of CMS and make different mandatory-to-implement algorithm
choices.

2.2.2. Cryptographic Key Size

Some cryptographic algorithms are inherently tied to a specific key
size, but others allow many different key sizes. Likewise, some
algorithms support parameters of different sizes, such as integrity

check values or nonces. The algorithm specification MUST identify
the specific key sizes and parameter sizes that are to be supported.
When more than one key size is available, expect the mandatory-to-
implement key size to increase over time.

Guidance on cryptographic key size for asymmetric keys can be found
in BCP 86 [RFC3766].

Guidance on cryptographic key size for symmetric keys can be found in
BCP 195 [RFC7525].

2.2.3. Providing Notice of Expected Changes

Fortunately, algorithm failures without warning are rare. More
often, algorithm transition is the result of age. For example, the
transition from DES to Triple-DES to AES took place over decades,
causing a shift in symmetric block cipher strength from 56 bits to
112 bits to 128 bits. Where possible, authors SHOULD provide notice
to implementers about expected algorithm transitions. One approach
that was first used in RFC 4307 [RFC4307] is to use SHOULD+, SHOULD-,
and MUST- in the specification of algorithms. The definitions below
are slightly modified from those in RFC 4307.

   SHOULD+ This term means the same as SHOULD. However, it is
            likely that an algorithm marked as SHOULD+ will be
            promoted to a MUST in the future.

   SHOULD- This term means the same as SHOULD. However, it is
            likely that an algorithm marked as SHOULD- will be
            deprecated to a MAY or worse in the future.

   MUST-   This term means the same as MUST. However, it is
            expected that an algorithm marked as MUST- will be
            downgraded in the future. Although the status of the
            algorithm will be determined at a later time, it is
            reasonable to expect that a the status of a MUST-
            algorithm will remain at least a SHOULD or a SHOULD-.

2.3. Transitioning from Weak Algorithms

Transition from an old algorithm that is found to be weak can be
tricky. It is of course straightforward to specify the use of a new,
better algorithm. And then, when the new algorithm is widely
deployed, the old algorithm ought no longer be used. However,
knowledge about the implementation and deployment of the new
algorithm will always be imperfect, so one cannot be completely
assured of interoperability with the new algorithm.

Algorithm transition is naturally facilitated as part of an algorithm selection or negotiation mechanism. Protocols traditionally select the best algorithm or suite that is supported by all communicating peers and acceptable by their policies. In addition, a mechanism is needed to determine whether the new algorithm has been deployed. For example, SMIMECapabilities [RFC5751] allows S/MIME mail user agents to share the list of algorithms that they are willing to use in preference order. For another example, the DNSSEC EDNS0 option [RFC6975] measures the acceptance and use of new digital signing algorithms.

In the Resource Public Key Infrastructure (RPKI), a globally recognized digital signature is needed. BCP 182 [RFC6916] provides an approach to transition, where a second signature algorithm is introduced and then the original one is phased out.

In the worst case, the old algorithm may be found to be tragically flawed, permitting a casual attacker to download a simple script to break it. Sadly, this has happened when a secure algorithm is used incorrectly or used with poor key management, resulting in a weak cryptographic algorithm suite. In such situations, the protection offered by the algorithm is severely compromised, perhaps to the point that one wants to stop using the weak suite altogether, rejecting offers to use the weak suite well before the new suite is widely deployed.

In any case, there comes a point in time where one refuses to use the old, weak algorithm or suite. This can happen on a flag day, or each installation can select a date on their own.

2.4. Algorithm Transition Mechanisms

Cryptographic algorithm selection or negotiation SHOULD be integrity protected. If selection is not integrity protected, then the protocol will be subject to a downgrade attack. Without integrity protection of algorithm or suite selection, the attempt to transition to a new algorithm or suite may introduce new opportunities for downgrade attacks.

Transition mechanisms need to consider the algorithm that is used to provide integrity protection for algorithm negotiation itself.

If a protocol specifies a single mandatory-to-implement integrity algorithm, eventually that algorithm will be found to be weak.

Extra care is needed when a mandatory-to-implement algorithm is used to provide integrity protection for the negotiation of other cryptographic algorithms. In this situation, a flaw in the mandatory-to-implement algorithm may allow an attacker to influence the choices of the other algorithms.

2.5. Cryptographic Key Establishment

Traditionally, protocol designers have avoided more than one approach to exchanges that establish cryptographic keys because it makes the security analysis of the overall protocol more difficult. When frameworks such as the Extensible Authentication Protocol (EAP) [RFC3748] and Simple Authentication and Security Layer (SASL) [RFC4422] are employed, key establishment is very flexible, often hiding many of the details from the application. This results in protocols that support multiple key establishment approaches. In fact, the key establishment approach itself is negotiable, which creates a design challenge to protect the negotiation of the key establishment approach before it is used to produce cryptographic keys.

Protocols can negotiate a key establishment approach, derive an initial cryptographic key, and then authenticate the negotiation. However, if the authentication fails, the only recourse is to start the negotiation over from the beginning.

Some environments will restrict the key establishment approaches by policy. Such policies tend to improve interoperability within a particular environment, but they cause problems for individuals that need to work in multiple incompatible environments.

2.6. Preserving Interoperability

Cryptographic algorithm deprecation is very difficult. People do not like to introduce interoperability problems, even to preserve security. As a result, flawed algorithms are supported for far too long. The impact of legacy software and long support tails on security can be reduced by making it easy to transition from old algorithms and suites to new ones. Social pressure is often needed to cause the transition to happen.

Implementers have been reluctant to remove deprecated algorithms or suites from server software, and server administrators have been reluctant to disable them over concerns that some party will no longer have the ability to connect to their server. Implementers and administrators want to improve security by using the best supported algorithms, but their actions are tempered by the desire to preserve connectivity. Recently, some browser vendors have started to provide

visual warnings when a deprecated algorithm or suite is used. These
visual warnings provide a new incentive to transition away from
deprecated algorithms and suites, prompting customers to ask for
improved security.

Transition in Internet infrastructure is particularly difficult. The
digital signature on the certificate for an intermediate
certification authority (CA) [RFC5280] is often expected to last
decades, which hinders the transition away from a weak signature
algorithm or short key length. Once a long-lived certificate is
issued with a particular signature algorithm, that algorithm will be
used by many relying parties, and none of them can stop supporting it
without invalidating all of the subordinate certificates. In a
hierarchical system, many subordinate certificates could be impacted
by the decision to drop support for a weak signature algorithm or an
associated hash function.

Organizations that have a significant influence can assist by
coordinating the demise of an algorithm suite, making the transition
easier for their own users as well as others.

2.7. Balancing Security Strength

When selecting or negotiating a suite of cryptographic algorithms,
the strength of each algorithm SHOULD be considered. The algorithms
in a suite SHOULD be roughly equal by providing comparable best-known
attack work factors. However, the security service provided by each
algorithm in a particular context needs to be considered when making
the selection. Algorithm strength needs to be considered at the time
a protocol is designed. It also needs to be considered at the time a
protocol implementation is deployed and configured. Advice from
experts is useful, but, in reality, such advice is often unavailable
to system administrators that are deploying a protocol
implementation. For this reason, protocol designers SHOULD provide
clear guidance to implementers, leading to balanced options being
available at the time of deployment.

Performance is always a factor is selecting cryptographic algorithms.
Performance and security need to be balanced. Some algorithms offer
flexibility in their strength by adjusting the key size, number of
rounds, authentication tag size, prime group size, and so on. For
example, TLS cipher suites include Diffie-Hellman or RSA without
specifying a particular public key length. If the algorithm
identifier or suite identifier named a particular public key length,
migration to longer ones would be more difficult. On the other hand,
inclusion of a public key length would make it easier to migrate away
from short ones when computational resources available to attacker
dictate the need to do so. The flexibility on asymmetric key length

has led to interoperability problems, and to avoid these problems in
the future any aspect of the algorithm not specified by the algorithm
identifiers need to be negotiated, including key size and parameters.

In CMS [RFC5652], a previously distributed symmetric key-encryption
key can be used to encrypt a content-encryption key, which in turn is
used to encrypt the content. The key-encryption and content-
encryption algorithms are often different. If, for example, a
message content is encrypted with a 128-bit AES key and the content-
encryption key is wrapped with a 256-bit AES key, then at most 128
bits of protection is provided. In this situation, the algorithm and
key size selections should ensure that the key encryption is at least
as strong as the content encryption. In general, wrapping one key
with another key of a different size yields the security strength of
the shorter key.

2.8. Balancing Protocol Complexity

   Protocol designs need to anticipate changes in the supported
   cryptographic algorithm set over time. There are a number of ways to
   enable the transition, and Section 3 discusses some of the related
   issues.

   Keep implementations as simple as possible. Complex protocol
   negotiation provides opportunities for attack, such as downgrade
   attacks. Support for many algorithm alternatives is also harmful.
   Both of these can lead to portions of the implementation that are
   rarely used, increasing the opportunity for undiscovered exploitable
   implementation bugs.

2.9. Opportunistic Security

   Despite the guidance in Section 2.4, opportunistic security [RFC7435]
   also deserves consideration, especially at the time a protocol
   implementation is deployed and configured. Opportunistic security,
   like other reasons for encrypting traffic, needs to make use of the
   strongest encryption algorithms that are implemented and allowed by
   policy. When communicating parties do not have strong algorithms in
   common, using algorithms that are weak against advanced attackers but
   sufficient against others is one way to make pervasive surveillance
   significantly more difficult. As a result, when communicating
   parties do not have strong algorithms in common, algorithms that
   would not be acceptable in many negotiated situations are acceptable
   for opportunistic security when legacy systems are in use for
   unauthenticated encrypted sessions (as discussed in Section 3 of
   [RFC7435]) as long as their use does not facilitate downgrade
   attacks. Similarly, weaker algorithms and shorter key sizes are also
   acceptable for opportunistic security with the same constraints.

   That said, the use of strong algorithms is always preferable.

3. Cryptographic Algorithm Specifications

   There are tradeoffs between the number of cryptographic algorithms
   that are supported and the time to deploy a new algorithm. This
   section provides some of the insights about the tradeoff faced by
   protocol designers.

   Ideally, two independent sets of mandatory-to-implement algorithms
   will be specified, allowing for a primary suite and a secondary
   suite. This approach ensures that the secondary suite is widely
   deployed if a flaw is found in the primary one.

3.1. Choosing Mandatory-to-Implement Algorithms

   It may seem as if the ability to use an algorithm of one's own
   choosing is very desirable; however, the selection is often better
   left to experts. When there are choices, end-users might select
   between configuration profiles that have been defined by experts.
   Further, experts need not specify each and every cryptographic
   algorithm alternative. Specifying all possible choices will not lead
   to them all being available in every implementation. Mandatory-to-
   implement algorithms MUST have a stable public specification and
   public documentation that has been well studied, giving rise to
   significant confidence. The IETF has always had a preference for
   unencumbered algorithms. There are significant benefits in selecting
   algorithms and suites that are widely deployed. The selected
   algorithms need to be resistant to side-channel attacks and also meet
   the performance, power, and code size requirements on a wide variety
   of platforms. In addition, inclusion of too many alternatives may
   add complexity to algorithm selection or negotiation. Specification
   of too many alternatives will likely hamper interoperability and may
   hamper security as well. When specifying new algorithms or suites,
   protocol designers would be prudent to consider whether existing ones
   can be deprecated.

   There is significant benefit in selecting the same algorithms and
   suites for different protocols. Using the same algorithms can
   simplify implementation when more than one of the protocols is used
   in the same device or system.

   Sometimes more than one mandatory-to-implement algorithm is needed to
   increase the likelihood of interoperability among a diverse
   population. For example, authenticated encryption is provided by
   AES-CCM [RFC3610] and AES-GCM [GCM]. Both of these algorithms are
   considered to be secure. AES-CCM is available in hardware used by
   many small devices, and AES-GCM is parallelizable and well suited to

high-speed devices. Therefore, an application needing authenticated
encryption might specify one of these algorithms or both of these
algorithms, depending on the population.

3.2. Too Many Choices Can Be Harmful

   It is fairly easy to specify the use of any arbitrary cryptographic
   algorithm, and once the specification is available, the algorithm
   gets implemented and deployed. Some people say that the freedom to
   specify algorithms independently from the rest of the protocol has
   led to the specification of too many cryptographic algorithms. Once
   deployed, even with moderate uptake, it is quite difficult to remove
   algorithms because interoperability with some party will be impacted.
   As a result, weaker ciphers stick around far too long. Sometimes
   implementers are forced to maintain cryptographic algorithm
   implementations well beyond their useful lifetime.

   In order to manage the proliferation of algorithm choices and provide
   an expectation of interoperability, many protocols specify mandatory-
   to-implement algorithms or suites. All implementers are expected to
   support the mandatory-to-implement cryptographic algorithm, and they
   can include any others algorithms that they desire. The mandatory-
   to-implement algorithms are chosen to be highly secure and follow the
   guidance in RFC 1984 [RFC1984]. Of course, many other factors,
   including intellectual property rights, have an impact on the
   cryptographic algorithms that are selected by the community.
   Generally, the mandatory-to-implement algorithms ought to be
   preferred, and the other algorithms ought to be selected only in
   special situations. However, it can be very difficult for a skilled
   system administrator to determine the proper configuration to achieve
   these preferences.

   In some cases, more than one mandatory-to-implement cryptographic
   algorithm has been specified. This is intended to ensure that at
   least one secure cryptographic algorithm will be available, even if
   other mandatory-to-implement algorithms are broken. To achieve this
   goal, the selected algorithms must be diverse, so that a
   cryptoanalytic advance against one of the algorithms does not also
   impact the other selected algorithms. The idea is to have an
   implemented and deployed algorithm as a fallback. However, all of
   the selected algorithms need to be routinely exercised to ensure
   quality implementation. This is not always easy to do, especially if
   the various selected algorithms require different credentials.
   Obtaining multiple credentials for the same installation is an
   unacceptable burden on system administrators. Also, the manner by
   which system administrators are advised to switch algorithms or
   suites is, at best, ad hoc and, at worst, entirely absent.

3.3. Picking One True Cipher Suite Can Be Harmful

   In the past, protocol designers have chosen one cryptographic
   algorithm or suite, and then tied many protocol details to that
   selection. Plan for algorithm transition, either because a mistake
   is made in the initial selection or because the protocol is
   successfully used for a long time and the algorithm becomes weak with
   age. Either way, the design should enable transition.

   Protocol designers are sometimes misled by the simplicity that
   results from selecting one true algorithm or suite. Since algorithms
   age, the selection cannot be stable forever. Even the most simple
   protocol needs a version number to signal which algorithm is being
   used. This approach has at least two desirable consequences. First,
   the protocol is simpler because there is no need for algorithm
   negotiation. Second, system administrators do not need to make any
   algorithm-related configuration decisions. However, the only way to
   respond to news that an algorithm that is part of the one true cipher
   suite has been broken is to update the protocol specification to the
   next version, implement the new specification, and then get it
   deployed.

   The first IEEE 802.11 [WiFi] specification included Wired Equivalent
   Privacy (WEP) as the only encryption technique. Many of the protocol
   details were driven by the selected algorithm. WEP was found to be
   quite weak [WEP], and a very large effort was needed to specify,
   implement, and deploy the alternative encryption techniques. This
   effort was made even harder by the protocol design choices that were
   tied to the initial algorithm selection and the desire for backward
   compatibility.

   Experience with the transition from SHA-1 to SHA-256 indicates that
   the time from protocol specification to widespread use takes more
   than five years. In this case, the protocol specifications and
   implementation were straightforward and fairly prompt. In many
   software products, the new algorithm was not considered an update to
   the existing release, so the roll-out of the next release, subsequent
   deployment, and finally adjustment of the configuration by system
   administrators took many years. In many consumer hardware products,
   firmware to implement the new algorithm was difficult to locate and
   install, or it was simply not available. Further, infrastructure
   providers were unwilling to make the transition until all of their
   potential clients were able to use the new algorithm.

3.4. National Cipher Suites

   Some nations specify cryptographic algorithms, and then require their
   use through legislation or regulations. These algorithms may not
   have wide public review, and they can have limited geographic scope
   in their deployment. Yet, the legislative or regulatory mandate
   creates a captive market. As a result, such algorithms will get
   specified, implemented, and deployed. The default server or
   responder configuration SHOULD disable such algorithms; in this way,
   explicit action by the system administrator is needed to enable them
   where they are actually required. For tiny devices with no user
   interface, an administrator action may only be possible at the time
   the device is purchased.

   National algorithms can force an implementer to produce several
   incompatible product releases for different countries or regions;
   this has significantly greater cost over development of a product
   using a globally acceptable algorithm. This situation could be even
   worse if the various national algorithms impose different
   requirements on the protocol, its key management, or its use of
   random values.

4. Security Considerations

   This document provides guidance to working groups and protocol
   designers. The security of the Internet is improved when broken or
   weak cryptographic algorithms can be easily replaced with strong
   ones.

   From a software development and maintenance perspective,
   cryptographic algorithms can often be added and removed without
   making changes to surrounding data structures, protocol parsing
   routines, or state machines. This approach separates the
   cryptographic algorithm implementation from the rest of the code,
   which makes it easier to tackle special security concerns such as key
   exposure and constant-time execution.

   Sometimes application-layer protocols can make use of transport-layer
   security protocols, such as TLS [RFC5246] or Datagram TLS (DTLS)
   [RFC6347]. This insulates the application-layer protocol from the
   details of cryptography, but it is likely to still be necessary to
   handle the transition from unprotected traffic to protected traffic
   in the application-layer protocol. In addition, the application-
   layer protocol may need to handle the downgrade from encrypted
   communication to plaintext communication.

Hardware offers challenges in the transition of algorithms, for both tiny devices and very high-end data center equipment. Many tiny devices do not include the ability to update the firmware at all. Even if the firmware can be updated, tiny devices are often deployed in places that make it very inconvenient to do so. High-end data center equipment may use special-purpose chips to achieve very high performance, which means that board-level replacement may be needed to change the algorithm. Cost and downtime are both factors in such an upgrade.

In most cases, the cryptographic algorithm remains strong, but an attack is found against the way that the strong algorithm is used in a particular protocol. In these cases, a protocol change will probably be needed. For example, the order of cryptographic operations in the TLS protocol has evolved as various attacks have been discovered. Originally, TLS performed encryption after computation of the message authentication code (MAC). This order of operations is called MAC-then-encrypt, which actually involves MAC computation, padding, and then encryption. This is no longer considered secure [BN] [K]. As a result, a mechanism was specified to use encrypt-then-MAC instead [RFC7366]. Future versions of TLS are expected to use exclusively authenticated encryption algorithms [RFC5116], which should resolve the ordering discussion altogether. After discovery of such attacks, updating the cryptographic algorithms is not likely to be sufficient to thwart the new attack. It may necessary to make significant changes to the protocol.

Some protocols are used to protect stored data. For example, S/MIME [RFC5751] can protect a message kept in a mailbox. To recover the protected stored data, protocol implementations need to support older algorithms, even when they no longer use the older algorithms for the protection of new stored data.

Support for too many algorithms can lead to implementation vulnerabilities. When many algorithms are supported, some of them will be rarely used. Any code that is rarely used can contain undetected bugs, and algorithm implementations are no different. Measurements SHOULD be used to determine whether implemented algorithms are actually being used, and if they are not, future releases should remove them. In addition, unused algorithms or suites SHOULD be marked as deprecated in the IANA registry. In short, eliminate the cruft.

Section 2.3 talks about algorithm transition without considering any other aspects of the protocol design. In practice, there are dependencies between the cryptographic algorithm and other aspects of

the protocol. For example, the BEAST attack [BEAST] against TLS
[RFC5246] caused many sites to turn off modern cryptographic
algorithms in favor of older and clearly weaker algorithms.

Mechanisms for timely update of devices are needed to deploy a
replacement algorithm or suite. It takes a long time to specify,
implement, and deploy a replacement; therefore, the transition
process needs to begin when practically exploitable flaws become
known. The update processes on some devices involve certification,
which further increases the time to deploy a replacement. For
example, devices that are part of health or safety systems often
require certification before deployment. Embedded systems and SCADA
(supervisory control and data acquisition) systems often have upgrade
cycles stretching over many years, leading to similar time-to-
deployment issues. Prompt action is needed if a replacement has any
hope of being deployed before exploitation techniques become widely
available.

5. IANA Considerations

This document does not establish any new IANA registries, nor does it
add any entries to existing registries.

This document does RECOMMEND a convention for new registries for
cryptographic algorithm or suite identifiers. Once an algorithm or
suite identifier is added to the registry, it SHOULD NOT be changed
or removed. However, it is desirable to include a means of marking a
registry entry as deprecated when implementation is no longer
advisable.

6. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
           Requirement Levels", BCP 14, RFC 2119,
           DOI 10.17487/RFC2119, March 1997,
           <http://www.rfc-editor.org/info/rfc2119>.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public
           Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766,
           DOI 10.17487/RFC3766, April 2004,
           <http://www.rfc-editor.org/info/rfc3766>.

7. Informative References

[BEAST]   Wikipedia, "BEAST attack" under "Transport Layer Security",
           November 2015, <https://en.wikipedia.org/w/index.php?title=
           Transport_Layer_Security&oldid=689441642#BEAST_attack>.

   [BN]   Bellare, M. and C. Namprempre, "Authenticated Encryption:
               Relations among notions and analysis of the generic
               composition paradigm", Proceedings of AsiaCrypt '00,
               Springer-Verlag LNCS No. 1976, p. 531,
               DOI 10.1007/3-540-44448-3_41, December 2000.

   [GCM]   Dworkin, M, "Recommendation for Block Cipher Modes of
               Operation: Galois/Counter Mode (GCM) and GMAC", NIST
               Special Publication 800-30D, November 2007.

   [K]   Krawczyk, H., "The Order of Encryption and Authentication
               for Protecting Communications (or: How Secure Is SSL?)",
               Proceedings of Crypto '01, Springer-Verlag LNCS No. 2139,
               p. 310, DOI 10.1007/3-540-44647-8_19, August 2001.

   [RFC1984]   IAB and IESG, "IAB and IESG Statement on Cryptographic
               Technology and the Internet", BCP 200, RFC 1984,
               DOI 10.17487/RFC1984, August 1996,
               <http://www.rfc-editor.org/info/rfc1984>.

   [RFC3365]   Schiller, J., "Strong Security Requirements for Internet
               Engineering Task Force Standard Protocols", BCP 61,
               RFC 3365, DOI 10.17487/RFC3365, August 2002,
               <http://www.rfc-editor.org/info/rfc3365>.

   [RFC3610]   Whiting, D., Housley, R., and N. Ferguson, "Counter with
               CBC-MAC (CCM)", RFC 3610, DOI 10.17487/RFC3610, September
               2003, <http://www.rfc-editor.org/info/rfc3610>.

   [RFC3748]   Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H.
               Levkowetz, Ed., "Extensible Authentication Protocol (EAP)",
               RFC 3748, DOI 10.17487/RFC3748, June 2004,
               <http://www.rfc-editor.org/info/rfc3748>.

   [RFC4302]   Kent, S., "IP Authentication Header", RFC 4302,
               DOI 10.17487/RFC4302, December 2005,
               <http://www.rfc-editor.org/info/rfc4302>.

   [RFC4303]   Kent, S., "IP Encapsulating Security Payload (ESP)",
               RFC 4303, DOI 10.17487/RFC4303, December 2005,
               <http://www.rfc-editor.org/info/rfc4303>.

   [RFC4307]   Schiller, J., "Cryptographic Algorithms for Use in the
               Internet Key Exchange Version 2 (IKEv2)", RFC 4307,
               DOI 10.17487/RFC4307, December 2005,
               <http://www.rfc-editor.org/info/rfc4307>.

   [RFC4422] Melnikov, A., Ed., and K. Zeilenga, Ed., "Simple
              Authentication and Security Layer (SASL)", RFC 4422,
              DOI 10.17487/RFC4422, June 2006,
              <http://www.rfc-editor.org/info/rfc4422>.

   [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated
              Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008,
              <http://www.rfc-editor.org/info/rfc5116>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <http://www.rfc-editor.org/info/rfc5246>.

   [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
              Housley, R., and W. Polk, "Internet X.509 Public Key
              Infrastructure Certificate and Certificate Revocation List
              (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
              <http://www.rfc-editor.org/info/rfc5280>.

   [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70,
              RFC 5652, DOI 10.17487/RFC5652, September 2009,
              <http://www.rfc-editor.org/info/rfc5652>.

   [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet
              Mail Extensions (S/MIME) Version 3.2 Message
              Specification", RFC 5751, DOI 10.17487/RFC5751, January
              2010, <http://www.rfc-editor.org/info/rfc5751>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <http://www.rfc-editor.org/info/rfc6347>.

   [RFC6916] Gagliano, R., Kent, S., and S. Turner, "Algorithm Agility
              Procedure for the Resource Public Key Infrastructure
              (RPKI)", BCP 182, RFC 6916, DOI 10.17487/RFC6916, April
              2013, <http://www.rfc-editor.org/info/rfc6916>.

   [RFC6975] Crocker, S. and S. Rose, "Signaling Cryptographic Algorithm
              Understanding in DNS Security Extensions (DNSSEC)",
              RFC 6975, DOI 10.17487/RFC6975, July 2013,
              <http://www.rfc-editor.org/info/rfc6975>.

   [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T.
              Kivinen, "Internet Key Exchange Protocol Version 2
              (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October
              2014, <http://www.rfc-editor.org/info/rfc7296>.

   [RFC7366] Gutmann, P., "Encrypt-then-MAC for Transport Layer Security
              (TLS) and Datagram Transport Layer Security (DTLS)",
              RFC 7366, DOI 10.17487/RFC7366, September 2014,
              <http://www.rfc-editor.org/info/rfc7366>.

   [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most
              of the Time", RFC 7435, DOI 10.17487/RFC7435, December
              2014, <http://www.rfc-editor.org/info/rfc7435>.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations
              for Secure Use of Transport Layer Security (TLS) and
              Datagram Transport Layer Security (DTLS)", BCP 195,
              RFC 7525, DOI 10.17487/RFC7525, May 2015,
              <http://www.rfc-editor.org/info/rfc7525>.

   [WEP]   Wikipedia, "Wired Equivalent Privacy", November 2015,
              <https://en.wikipedia.org/w/index.php?
              title=Wired_Equivalent_Privacy&oldid=688848497>.

   [WiFi]   IEEE, "Wireless LAN Medium Access Control (MAC) And
              Physical Layer (PHY) Specifications", IEEE Std 802.11-1997,
              1997.

Author's Address

   Russ Housley
   Vigil Security, LLC
   918 Spring Knoll Drive
   Herndon, VA 20170
   United States

   Email: housley@vigilsec.com