Internet-Draft
Intended Status: Best Current Practice
Expires: 2 November 2015

R. Housley
Vigil Security

1 May 2015

Guidelines for Cryptographic Algorithm Agility
<draft-iab-crypto-alg-agility-03.txt>

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 algorithm suite to another over time.

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.

Cryptographic algorithms age; they become weaker with time. As new cryptanalysis techniques are developed and computing capabilities improve, the work factor to break a particular cryptographic algorithm will reduce or become more feasible for more attackers.

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. For the protocol designer, this means that one or more algorithm identifier must be carried, the set of mandatory-to-implement algorithms will change over time, and an IANA registry of algorithm identifiers will be needed. These algorithm identifiers might name a single cryptographic algorithm or a suite of algorithms.

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. Ideally, this takes place before the older algorithm or suite of algorithms is catastrophically weakened. Experience shows that many people are unwilling to disable older weaker algorithms; it seems that these people prefer to live with weaker algorithms, sometimes seriously flawed ones, to maintain interoperability with older software well after experts recommend migration.

1.1. Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear 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.

2.1. Algorithm Identifiers

IETF protocols that make use of cryptographic algorithms MUST carry one or more algorithm or suite identifier.

Some approaches carry one identifier for each algorithm that is used. Other approaches carry one identifier for a 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 completely compatible.

In the IPsec protocol suite, IKE [RFC2409][RFC4306] carries the algorithm identifiers for AH and ESP [RFC4302][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.

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, the protocol MUST specify one or more mandatory-to-implement algorithm or suite. Note that this is not done for protocols that are embedded in

other protocols, where 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 can make use of CMS and make different mandatory-to-implement algorithm choices.

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, 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.

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].

Symmetric keys used for protection of long-term values SHOULD be at least 128 bits.

2.3. Transition 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.

To facilitate transition, protocols MUST be able to advertise which algorithms are supported. This may naturally occur as part of an algorithm selection or negotiation mechanism, and other times a mechanism is needed to determine whether the new algorithm has been deployed. For example, the DNSSEC EDNS0 option [RFC6975] measures the acceptance and use of new digital signing algorithms.

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. Balance Security Strength

When selecting a suite of cryptographic algorithms, the strength of each algorithm SHOULD be considered. It 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 from experts is useful, but in reality, it is not often available to system administrators that are deploying and configuring a protocol implementation. For this reason, protocol designers SHOULD provide clear guidance to implementors, leading to balanced options being available at the time of deployment and configuration.

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. Therefore, flexibility on asymmetric key length is both desirable and undesirable at the same time.

In CMS [RFC5652], a previously distributed symmetric key-encryption key can be used to encrypt a content-encryption key, which is in turn used to encrypt the content. The key-encryption and content-encryption algorithms are often different. If, for example, a message content is encrypted with 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.

2.5. Opportunistic Security

Despite the guidance in Section 2.4, opportunistic security [RFC7435] SHOULD also be considered, especially at the time a protocol implementation is deployed and configured. While RSA with a 2048-bit public key is quite a bit stronger than SHA-1, it is quite reasonable to use them together if the alternative is no authentication whatsoever. That said, the use of strong algorithms is always preferable.

3. Algorithm Agility in Protocol Design

Some attempts at algorithm agility have not been completely successful. This section provides some of the insights based on protocol designs and deployments.

3.1. Algorithm Identifiers

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. In addition, an IANA registry is needed to pair the identifier with an algorithm specification.

Sometimes a combination of protocol version number and explicit algorithm or suite identifiers is appropriate. For example, the TLS version number names the default key derivation function and the cipher suite identifier names the rest of the needed algorithms.

Sometimes application layer protocols can make use of transport layer security protocols, such as TLS or DTLS. 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.

3.2. Migration 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 attack.

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.

Performance is always a factor is selecting cryptographic algorithms. In many algorithms, shorter keys offer higher performance, but less security. Performance and security need to be balanced. Yet, all algorithms age, and the advances in computing power available to the attacker will eventually make any algorithm obsolete. For this reason, protocols need mechanisms to migrate from one algorithm suite to another over time, including the algorithm used to provide integrity protection for algorithm negotiation.

3.3. Preserving Interoperability

Cryptographic algorithm deprecation is very hard. People do not like to introduce interoperability problems, even to preserve security. As a result, flawed algorithms are supported for far too long. The impacts of legacy software an long support tails on security can be reduced by making it easy to develop, deploy, and configure new algorithms.

3.4. Cryptographic Key Management

Traditionally, protocol designers have avoided more than one approach to key management because it makes the security analysis of the overall protocol more difficult. When frameworks such as EAP and GSSAPI are employed, the key management is very flexible, often hiding many of the details from the application. This results in protocols that support multiple key management approaches. In fact, the key management approach itself may be negotiable, which creates a design challenge to protect the negotiation of the key management approach before it is used to produce cryptographic keys.

Protocols can negotiate a key management 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 management 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.

4. Cryptographic Algorithm Specifications

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

4.1. Choosing Mandatory-to-Implement Algorithms

It seems like the ability to use an algorithm of one's own choosing is very desirable; however, the selection is often better left to experts. Further, any and all cryptographic algorithm choices ought not be available in every implementation. Mandatory-to-implement algorithms ought to be well studied, giving rise to significant confidence. The selected algorithms need to be resistant to side-channel attacks as well as meeting 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.

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

4.2. 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 lead 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 implementors 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 implementors 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.

4.3. Picking One True Cipher Suite Can Be Harmful

After careful study and evaluation, the protocol designer could select the one true cipher suite. Since algorithms age, such a decision cannot be stable forever, so a version number is needed to signal which algorithm that is being used. This has at least two desirable consequences. First, the protocol is simpler since 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 the 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 the Wired Equivalent Privacy (WEP) as the only encryption technique. WEP was found to be quite weak [WEP], and a very large effort was needed to specify, implement, and deploy the alternative encryption techniques.

Experience with the transition from SHA-1 to SHA-256 indicates that the time from protocol specificate 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 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 were difficult to locate and install, or the were 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.

4.4. 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 reach of deployments. Yet, the legislative or regulatory mandate creates a captive market. As a result, the use of such algorithms get specified, implemented, and deployed. The default server-side configuration SHOULD disable such algorithms; in this way, explicit action by the system administrator is needed to enable them where they are actually required.

4.5. Balance Protocol Complexity

Protocol designers MUST be prepared for the supported cryptographic algorithm set to change over time. As shown by the discussion in the previous two sections, there is a spectrum of ways to enable the transition.

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, as discussed in Section 4.1. Both of these can lead to portions of the implementation that are rarely used, increasing the opportunity for undiscovered exploitable implementation bugs.

5. 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.

The situation is different for hardware, 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 down-time 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, and it 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 [RFC5166], 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 protected 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.

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.

6. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.

7. Informative References

[BEAST] http://en.wikipedia.org/wiki/Transport_Layer_Security#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, 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, August 2001.

[RFC1984] IAB and IESG, "IAB and IESG Statement on Cryptographic Technology and the Internet", RFC 1984, August 1996.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols", BCP 61, RFC 3365, August 2002.

[RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, September 2003.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4306] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC5166] Floyd, S., Ed., "Metrics for the Evaluation of Congestion Control Mechanisms", RFC 5166, March 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[RFC6975] Crocker, S. and S. Rose, "Signaling Cryptographic Algorithm Understanding in DNS Security Extensions (DNSSEC)", RFC 6975, July 2013.

[RFC7366] Gutmann, P., "Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7366, September 2014.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, December 2014.

[WEP] http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy

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

Acknowledgements

Thanks to Bernard Aboba, Derek Atkins, David Black, Randy Bush, Jon Callas, Andrew Chi, Steve Crocker, Viktor Dukhovni, Stephen Farrell, Tony Finch, Ian Grigg, Peter Gutmann, Wes Hardaker, Joe Hildebrand, Christian Huitema, Watson Ladd, Paul Lambert, Ben Laurie, Eliot Lear, Nikos Mavrogiannopoulos, Yoav Nir, Rich Salz, Kristof Teichel, Jeffrey Walton, Nico Williams, and Peter Yee for their review and insightful comments. While some of these people do not agree with some aspects of this document, the discussion that resulted for their comments has certainly resulted in a better document.

Author's Address

Russ Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA
EMail: housley@vigilsec.com