

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

A. Malhotra
Boston University
M. Hoffmann
Open Netlabs
W. Toorop
NLnet Labs
October 30, 2017

On Implementing Time
draft-aanchal-time-implementation-guidance-00

Abstract

This document describes the properties of different types of time values available on digital systems and provides guidance on choices of these time values to the implementors of applications that use time in some form to provide the basic functionality and security guarantees.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction

The basic functionality and security guarantees claimed by many applications running on digital systems locally or in the Internet hinge on some notion of time. These applications have to choose one of the many types of time values available on the system, each of which has its own specific properties. However, currently these applications seem to be oblivious to the implications of choosing one or the other time value for implementation. This behaviour can be attributed to: a) the lack of clear understanding of the distinct properties of these time values, b) trade-offs of using one or the other for an application, and c) availability and compatibility of these time values on different operating systems.

In this document we describe the properties of various available time values on modern operating systems, discuss the trade-offs of using one over the other, and provide guidance to help implementors make an informed choice with some real-life examples.

2. Keeping Time: Different Clocks

Because time is relative to an observer, there cannot be a universally agreed upon time. At best we can achieve an approximation by updating our own observed time with a common reference time shared with other observers.

As this reference time is what we naively assume clocks on a wall are showing, we shall call it the "wall time." For most applications, it is based on the Universal Coordinated Time (UTC), an international standard time determined by averaging the output of several high-precision time-keeping devices. However, as UTC is following Earth's solar time, it occasionally needs to be adjusted through leap seconds.

An individual computer system's perception of time differs from this idealized wall time. Staying close to it requires some effort that comes with its own set of drawbacks. Systems therefore provide access to different types of clocks with different properties. Unfortunately, there is no standard terminology and definitions for these types. For the purpose of this document, we therefore define three different kinds of clocks that a system may or may not provide.

2.1. Raw Time

At its most fundamental, a system has its own perception of time; its unmodified, "raw time." This time is typically measured by counting cycles of an oscillator. Its quality therefore relies on the stability of this oscillator.

As it is a purely subjective time, no general meaning can be attached to any specific value. Only the amount of time passed can be determined by comparing two values.

Because raw time is unaltered, it is continuous and strictly monotonically increasing. Its value will always grow at a steady pace, never decrease, never make unexpected jumps, or stip. Such a time is sometimes called a "monotonic time."

2.2. Adjusted Raw Time

Even if highly accurate oscillators are used, raw time passes at a slightly different rate than wall time. This difference is called clock drift. It depends not only on the quality of the time source but also on environmental factors such as temperature.

When this drift is compensated by comparing the passage of raw time to some external time source that is considered to be closer to wall time, the result is "adjusted raw time." This adjustment doesn't happen sporadically but rather, the rate of advance of time is slowed down or sped up slightly until it approaches the reference time again. As a result, adjusted raw time is still monotonic. Like raw time, adjusted raw time is subjective with no specific meaning attached to its values.

The most frequently used method of acquiring an external time source is through network timing protocols such as NTP [RFC5905]. As a result, adjusted raw time is susceptible to vulnerabilities of these protocols which may be exploited to maliciously manipulate this time.

2.3. Real Time

With adjusted raw time, a system already has access to a time that passes at a rate very similar to wall time. By adjusting the time value so that it represents the time passed since an epoch, a well-defined point of wall time such as seconds since midnight January 1st, 1970 on Unix systems, time values themselves gather meaning. The result is "real time."

While it is often assumed that real time is set to match wall time, this doesn't need to be the case. A system's operator is free to

change the value of real time at any time, likewise, system services such as a local NTP client may decide to do so.

As a consequence, real time is not monotonic. Not only may it jump forward, its value may even decrease.

2.4. Differences from Wall Time

These three clock types differ from wall time in three aspects:

- o Both raw time and adjusted raw time can only represent differences in time by comparing two clock values. Only real time provides absolute time values that can be compared to wall time values.
- o On the other hand, raw time and adjusted raw time are always monotonic whereas real time may experience sudden changes in value in either direction.
- o Only adjusted raw time and real time are subject to external adjustments so that time passes at approximately the same rate as wall time. Raw time will over time drift away due to inevitable imperfections of the clock.

3. Expressing Time

Protocols or applications can express time in one of the two forms, depending on whether global agreement over the point in time is necessary.

3.1. Time Stamps

A "time stamp" expresses an absolute point in time. In order to reference the same point across multiple systems, it needs to be stated in wall time.

Time stamps are often used to express the validity of objects with a limited lifetime that are shared over the network. For instance, PKIX certificates [RFC5280] carry two time stamps expressing their earliest and latest validity.

In order to validate a time stamp, a system needs access to a clock that is reasonably close to wall time.

3.2. Time Spans

In contrast, a "time span" expresses a desired length of time. Examples of time spans are timeout values used in protocols to

determine packet loss or Time to Live (TTL) values that govern the lifetime of a local copy of an object.

While no access to wall time is necessary for correctly dealing with time spans, using a clock whose time passes at a different rate than wall time will result in different interpretations of time spans by different systems. However, in a network environment, the uncertainty introduced by differing transmission times is likely larger than that introduced by clock drift.

4. Current Implementations and Their Flaws

Currently, some software takes a common approach towards time stamps and time spans. Time stamps are registered with their wall time value, and time spans are registered with two time stamp values marking the start and the end of the span. Conversion of a time span into those time stamp markers is regularly based on real time.

Note that the start of a time span will be the current (real) time in case of a TTL. So, in case something needs to be cached for a certain time, the start time stamp is irrelevant and it is registered together with only the (real) expiration time.

Programmers might have had different reasons to base those markings on real time, for example:

1. A point in time is intuitively thought of as a wall clock time stamp. Time stamps from outside the software, which the software has to manage are already in wall clock time. The POSIX function to get the current (real) time which is regularly used for this, is `gettimeofday()`, which comes accross as something providing near wall clock time and which can be used for this purpose.
2. Managing time stamps and time span similarly, prevents code complexity.

For example, many software is organized around I/O event notification mechanisms like the POSIX `select()` and `poll()` system C API functions. These functions wait for a given time span for file descriptors to become ready to perform I/O. The given time span is determined by subtracting the current real time value from smallest registered time stamp. When file descriptors are ready, the non-blocking I/O is performed, otherwise the given time span has passed and the action associated with the smallest registered time stamp needs to be performed.

For this programming pattern, a sorted list of time stamps has to be maintained by the software. To avoid coding complexity,

programmers might prefer a single list for both actual wall clock time stamps and those generated from real time to mark the end of a time span.

Using real time as a basis for the time stamps marking the start and end of a time span is bad because of the following reasons.

1. It can be set or overwritten manually,
2. It is subject to adjustments by timing protocols which on one hand is important to make sure that this time is in sync with the rest of the world but on the other hand makes it dependent on the correctness and security of timing protocols.

Recent attacks [SECNTP], [MCBG] show how timing protocols like NTP can be leveraged to shift real time on systems.

Time stamps are always based on wall time, so the best one can do is to use real time while dealing with them. However, this limitation does not hold for the time spans. Managing time spans may be implemented in alternative ways which may prove to be more secure and robust.

An obvious question to ask is: Why do we need inception and expiration time stamps in the first place to define the validity period of cryptographic objects? Why can't we just use time spans like TTL values instead? The reason is straightforward.

The authority determining and setting the validity period on the object can be different from the operator delivering the object. For example the TTL value on DNS resource records indicates to caching DNS resolvers how long to cache those records. These are an operational matter and are thus left to the operators of the DNS zone.

The content of the resource records are however determined by the signer of the records. When she is not also the zone operator, she has no way to determine when the records will be queried for, and thus has to depend on cryptographically signed wall clock based time stamps to limit the validity.

Note however that DNSSEC signatures do contain the original TTL of a resource record set, restricting the maximum TTL value with which the operator may deliver the resource records.

5. Alternative Approaches

For time spans, where we only need the rate of passage of time to be close enough to the rest of the world, one should not use the real time to establish the start and end time for the reasons mentioned above. The other two types of time are raw time and adjusted raw time. The important aspect of these monotonic time sources is not their current value but the guarantee that the time source is strictly linearly increasing and thus useful for calculating the difference in time between two samplings. But each comes with its own caveats.

Raw time is not subject to any adjustments by timing protocols, i.e., it is not adjusted for the error introduced by clock drift. This could have two repercussions. First, this makes correctness of raw time independent from the errors or security vulnerabilities of the timing protocols. Second, its correctness depends on the clock drift which further depends on various factors such as quality of the oscillator, work load, or ambient temperature on the system and may vary.

Adjusted raw time, on the other hand, is subject to adjustments by timing protocols. While it therefore compensates for the errors introduced by the drift of the local clock, this time can be incorrect as it is vulnerable to accuracy and security vulnerabilities of the underlying timing protocol.

The choice of time value to be used is application-specific. For instance in applications that can tolerate a certain amount of clock drift [CLOCKDRIFT], implementers can use raw time. However, if that is an issue then one has no choice but to fall back to adjusted raw time.

POSIX defines a system C API function which may provide raw time: `clock_gettime()`, when used with a `clock_id` of `CLOCK_MONOTONIC` (when supported by the system). POSIX does not make a distinction between raw time and adjusted raw time in the definition of this function. Beware that with some systems, `CLOCK_MONOTONIC` delivers adjusted raw time and that `CLOCK_MONOTONIC_RAW` needs to be used as `clock_id` to get unadjusted raw time. Non-POSIX systems may provide different APIs

Software employing the pattern organized around I/O event notification mechanisms, as described in Section 4, should maintain two sorted lists of two different types of time stamps:

1. One to register events based on time stamps expressed in wall clock time

2. One to register the start and end of time spans in (adjusted) raw time

To determine the timeout value for a call to `select()` or `poll()`, the program needs to get the current time in both real time and in (adjusted) raw time. The current real time is subtracted from the lowest value of the time stamps expressed in wall time list. The current (adjusted) raw time from the lowest value of the time stamps expressed in (adjusted) raw time list. The lowest of the values should be used as the timeout value for `select()` or `poll()` and determines which action should be performed when the function times out.

Alternatively a single list of (adjusted) raw time could be used for both time stamps and time spans. In that case time stamps expressed in wall clock time should be converted into (adjusted) raw time, by first converting it into a time span by subtracting real time from it, and then adding the current time in (adjusted) raw time.

6. Acknowledgements

We are thankful to Sharon Goldberg and Benno Overreinder for useful discussions.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

Time is a fundamental component for the security guarantees claimed by various applications. Therefore, any implementor concerned with security should be concerned with how these time values are implemented. This document discusses the security considerations with respect to implementing time values in applications in various sections.

9. Informative References

[CLOCKDRIFT]

Marouani, H. and M. Dagenais, "Internal clock drift estimation in computer clusters", 2008, <<http://downloads.hindawi.com/journals/jcnc/2008/583162.pdf>>.

[MCBG]

Malhotra, A., Cohen, I., Brakke, E., and S. Goldberg, "Attacking the Network Time Protocol", 2015, <<https://eprint.iacr.org/2015/1020>>.

- [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, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [SECNTP] Malhotra, A., Gundy, M., Varia, M., Kennedy, H., Gardner, J., and S. Goldberg, "The Security of NTP's Datagram Protocol", 2016, <<http://eprint.iacr.org/2016/1006>>.

Authors' Addresses

Aanchal Malhotra
Boston University
111 Cummington Mall
Boston 02215
USA

Email: aanchal4@bu.edu

Martin Hoffmann
Open Netlabs
Science Park 400
Amsterdam 1098 XH
Netherlands

Email: martin@opennetlabs.com

Willem Toorop
NLnet Labs
Science Park 400
Amsterdam 1098 XH
Netherlands

Email: willem@nlnetlabs.nl

Network Working Group
Internet-Draft
Updates: 5448 (if approved)
Intended status: Informational
Expires: May 3, 2018

J. Arkko
K. Norrman
V. Torvinen
Ericsson
October 30, 2017

Perfect-Forward Secrecy for the Extensible Authentication Protocol
Method for Authentication and Key Agreement (EAP-AKA' PFS)
draft-arkko-eap-aka-pfs-00

Abstract

Many different attacks have been reported as part of revelations associated with pervasive surveillance. Some of the reported attacks involved compromising smart cards, such as attacking SIM card manufacturers and operators in an effort to compromise shared secrets stored on these cards. Since the publication of those reports, manufacturing and provisioning processes have gained much scrutiny and have improved. However, the danger of resourceful attackers for these systems is still a concern.

This specification is an optional extension to the EAP-AKA' authentication method which was defined in RFC 5448. The extension provides Perfect Forward Secrecy for the session key generated as a part of the authentication run in EAP-AKA'. This prevents an attacker who has gained access to the long-term pre-shared secret in a SIM card from merely passively eavesdropping the EAP-AKA' exchanges and deriving associated session keys, forcing attackers to use active attacks instead.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 2. Background | 4 |
| 2.1. AKA | 4 |
| 2.2. EAP-AKA' Protocol | 5 |
| 2.3. Attacks Against Long-Term Shared Secrets in Smart Cards . | 6 |
| 3. Requirements Language | 7 |
| 4. Protocol Overview | 7 |
| 5. Extensions to EAP-AKA' | 9 |
| 5.1. AT_PUB_DH | 9 |
| 5.2. AT_KDF_DH | 10 |
| 5.3. New Key Derivation Function | 12 |
| 5.4. Diffie-Hellman Groups | 13 |
| 5.5. Message Processing | 13 |
| 5.5.1. EAP-Request/AKA'-Identity | 13 |
| 5.5.2. EAP-Response/AKA'-Identity | 13 |
| 5.5.3. EAP-Request/AKA'-Challenge | 13 |
| 5.5.4. EAP-Response/AKA'-Challenge | 14 |
| 5.5.5. EAP-Request/AKA'-Reauthentication | 14 |
| 5.5.6. EAP-Response/AKA'-Reauthentication | 14 |
| 5.5.7. EAP-Response/AKA'-Synchronization-Failure | 15 |
| 5.5.8. EAP-Response/AKA'-Authentication-Reject | 15 |
| 5.5.9. EAP-Response/AKA'-Client-Error | 15 |
| 5.5.10. EAP-Request/AKA'-Notification | 15 |
| 5.5.11. EAP-Response/AKA'-Notification | 15 |
| 6. Security Considerations | 15 |
| 7. IANA Considerations | 17 |
| 8. References | 18 |
| 8.1. Normative References | 18 |
| 8.2. Informative References | 19 |
| Appendix A. Acknowledgments | 20 |
| Authors' Addresses | 20 |

1. Introduction

Many different attacks have been reported as part of revelations associated with pervasive surveillance. Some of the reported attacks involved compromising smart cards, such as attacking SIM card manufacturers and operators in an effort to compromise shared secrets stored on these cards. Such attacks are conceivable, for instance, during the manufacturing process of cards, or the transfer of cards and associated information to the operator. Since the publication of reports about such attacks, manufacturing and provisioning processes have gained much scrutiny and have improved.

However, the danger of resourceful attackers attempting to gain information about SIM cards is still a concern. They are a high-value target and concern a large number of people. Note that the attacks are largely independent of the used authentication technology; the issue is not vulnerabilities in algorithms or protocols, but rather the possibility of someone gaining unlawful access to key material. While the better protection of manufacturing and other processes is essential in protecting against this, there is one question that we as protocol designs can ask. Is there something that we can do to limit the consequences of attacks, should they occur?

This specification is an optional extension to the EAP-AKA' authentication method [RFC5448]. The extension provides Perfect Forward Secrecy for the session key generated as a part of the authentication run in EAP-AKA'. This prevents an attacker who has gained access to the long-term pre-shared secret in a SIM card from merely passively eavesdropping the EAP-AKA' exchanges and deriving associated session keys, forcing attackers to use active attacks instead.

This extension specified here re-uses large portions of the current structure of 3GPP interfaces and functions, with the rationale that this will make the construction more easily adopted. In particular, the construction maintains the interface between the Universal Subscriber Identification Module (USIM) and the mobile terminal intact. As a consequence, there is no need to roll out new credentials to existing subscribers. The work is based on an earlier paper [TrustCom2015], and uses much of the same material, but applied to EAP rather than the underlying AKA method. This 00 version of the specification is an initial proposal for ensuring SIM-based authentication in EAP makes attacks difficult. Comments and suggestions are much appreciated, including design improvements.

It has been a goal to implement this change as an extension of the widely supported EAP-AKA' method, rather than a completely new

authentication method. The extension is implemented as a set of new, optional attributes, that are provided alongside the base attributes in EAP-AKA'. Old implementations can ignore these attributes, but their presence will nevertheless be verified as part of base EAP-AKA' integrity verification process, helping protect against bidding down attacks. This extension does not increase the number of rounds necessary to complete the protocol.

The use of this extension is at the discretion of the authenticating parties. There are currently no requirements mandating the use of this extension from 3GPP or otherwise, but the authors want to provide a public specification of an extension that helps defend against one aspect of pervasive surveillance. It should be noted that PFS and defenses against passive attacks are by no means a panacea, but they can provide a partial defense that increases the cost and risk associated with pervasive surveillance.

It should also be noted that the planned 5G network architecture includes the use of the EAP framework for authentication. The default authentication method within that context will be EAP-AKA', but other methods can certainly also be run.

2. Background

2.1. AKA

AKA is based on challenge-response mechanisms and symmetric cryptography. AKA typically runs in a UMTS Subscriber Identity Module (USIM) or a CDMA2000 (Removable) User Identity Module ((R)UIM). In contrast with its earlier GSM counterparts, 3rd generation AKA provides long key lengths and mutual authentication.

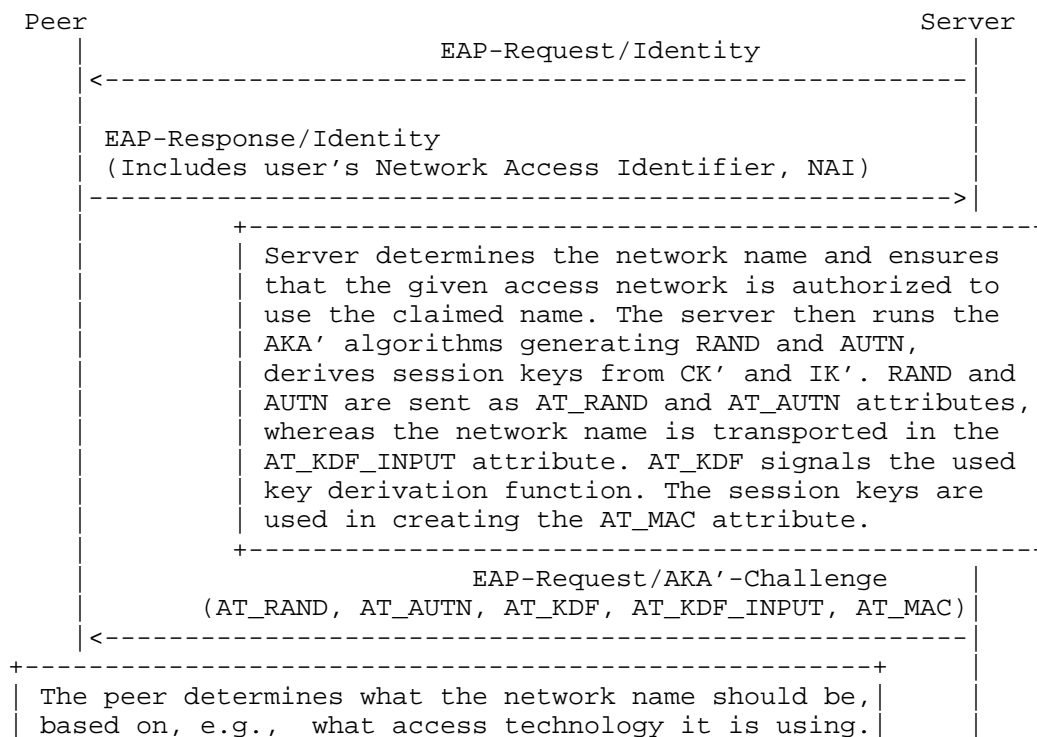
AKA works in the following manner:

- o The identity module and the home environment have agreed on a secret key beforehand.
- o The actual authentication process starts by having the home environment produce an authentication vector, based on the secret key and a sequence number. The authentication vector contains a random part RAND, an authenticator part AUTN used for authenticating the network to the identity module, an expected result part XRES, a 128-bit session key for integrity check IK, and a 128-bit session key for encryption CK.
- o The authentication vector is passed to the serving network, which uses it to authenticate the device.

- o The RAND and the AUTN are delivered to the identity module.
- o The identity module verifies the AUTN, again based on the secret key and the sequence number. If this process is successful (the AUTN is valid and the sequence number used to generate AUTN is within the correct range), the identity module produces an authentication result RES and sends it to the serving network.
- o The serving network verifies the correct result from the identity module. If the result is correct, IK and CK can be used to protect further communications between the identity module and the home environment.

2.2. EAP-AKA' Protocol

When AKA (and AKA') are embedded into EAP, the authentication on the network side is moved to the home environment; the serving network performs the role of a pass-through authenticator. Figure 1 describes the basic flow in the EAP-AKA' authentication process. The definition of the full protocol behaviour, along with the definition of attributes AT_RAND, AT_AUTN, AT_MAC, and AT_RES can be found in [RFC5448] and [RFC4187].



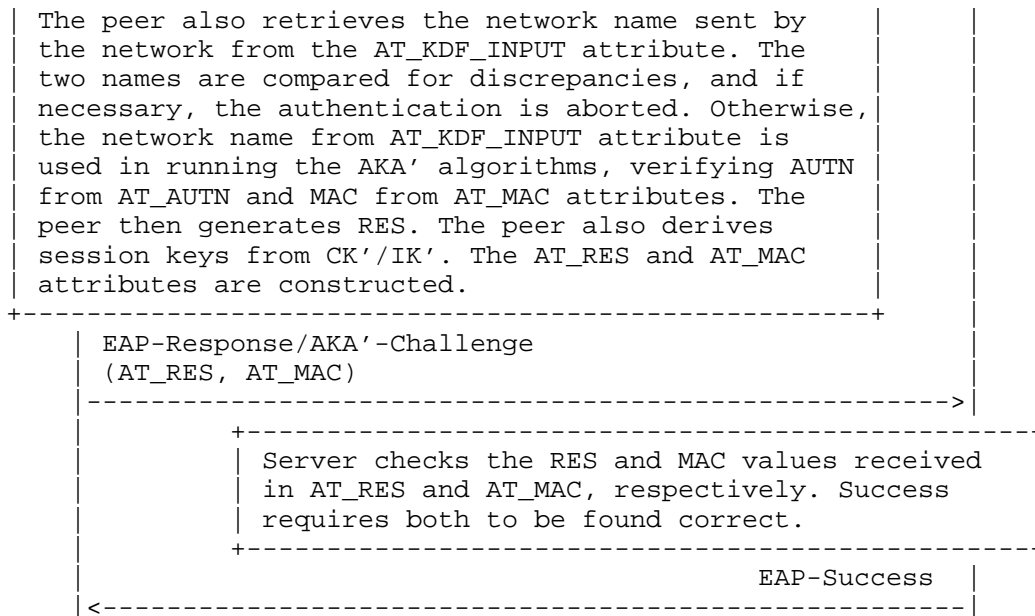


Figure 1: EAP-AKA' Authentication Process

2.3. Attacks Against Long-Term Shared Secrets in Smart Cards

Current 3GPP systems use (U)SIM pre-shared key based protocols to authenticate subscribers. Since the addition of replay protection and mutual authentication in the third generation 3GPP systems, there have been no published attacks that violate the security properties defined for the Authentication and Key Agreement (AKA) in, at least not within the assumed trust model. (However, there have been attacks using a different trust model [CB2014] [MT2012]; the protocol was not designed to counter those situations. There have also been attacks against systems where AKA is used in a different setting than initially intended, e.g. [BT2013].)

Recent reports of compromised long term pre-shared keys used in AKA [Heist2015] indicate a need to look into solutions that allow a weaker trust model, in particular for future 5G systems. It is also noted in [Heist2015] that, even if the current trust model is kept, some security can be retained in this situation by providing Perfect Forward Security (PFS) [DOW1992] for the session key. If AKA would have provided PFS, compromising the pre-shared key would not be sufficient to perform passive attacks; the attacker is, in addition, forced to be a Man-In-The-Middle (MITM) during the AKA run. Introducing PFS for authentication in 3GPP systems can be achieved by adding a Diffie-Hellman (DH) exchange.

3. Requirements Language

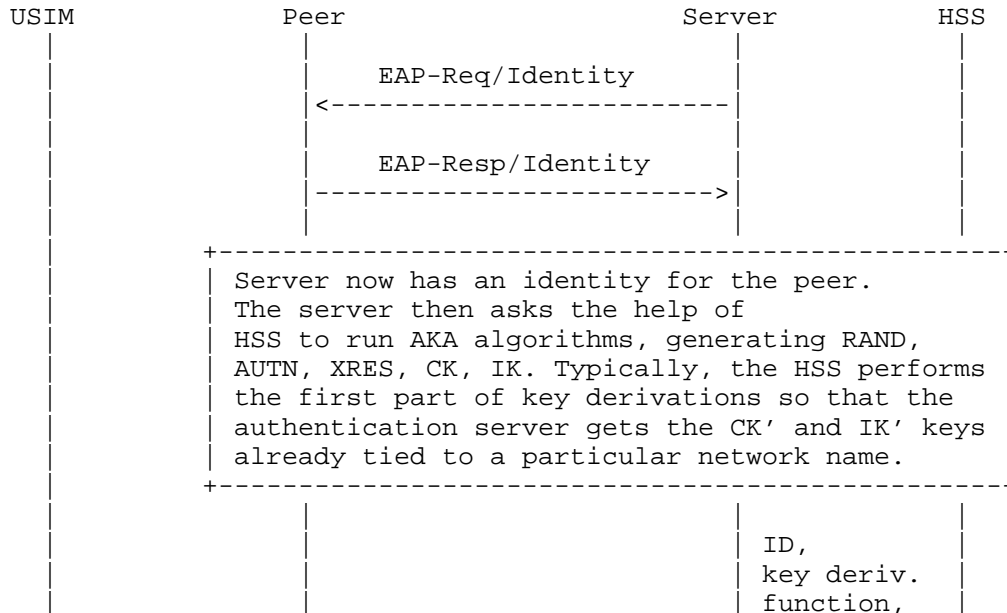
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

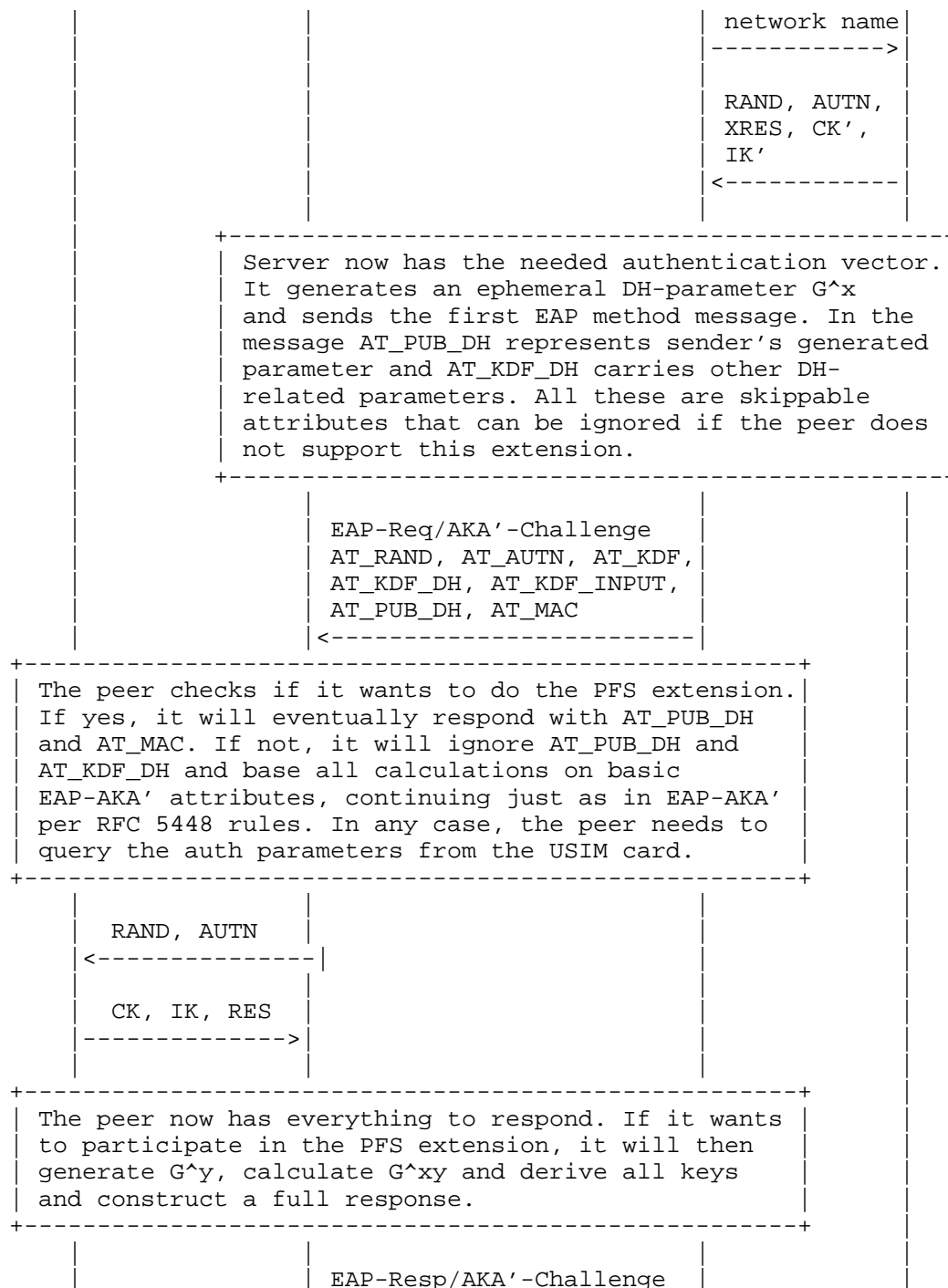
4. Protocol Overview

The enhancements in the protocol specified here are compatible with the signaling flow and other basic structures of both AKA and EAP-AKA'. The intent is to implement the enhancement as optional attributes that legacy implementations can ignore.

The purpose of the protocol is to achieve mutual authentication between the EAP server and peer, and to establish keying material for secure communication between the two. The enhancements brought in this document change the calculation of key material, providing new properties that are not present in key material provided by EAP-AKA' in its original form.

Figure 2 below describes the overall process. Since our goal has been to not require new infrastructure or credentials, the flow diagrams also show the conceptual interaction with the USIM card and the 3GPP authentication server (HSS). The details of those interactions are outside the scope of this document, however, and the reader is referred to the the 3GPP specifications .





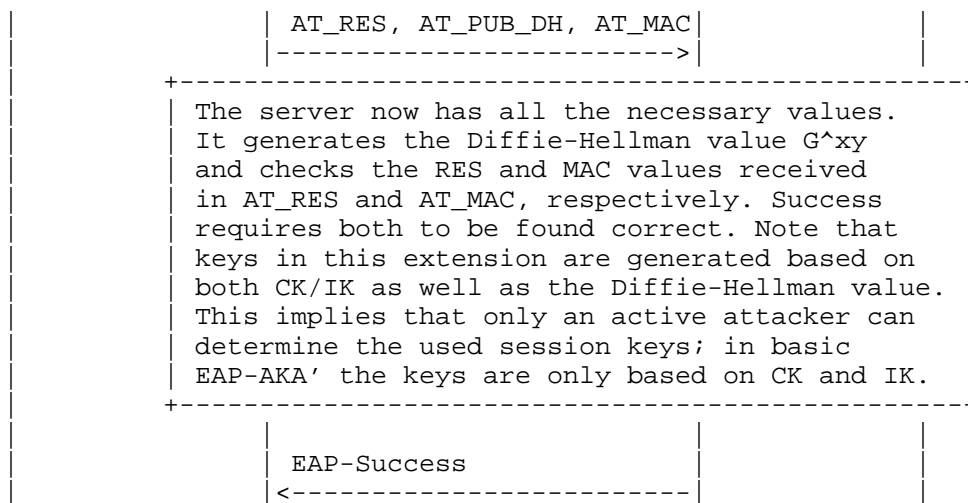


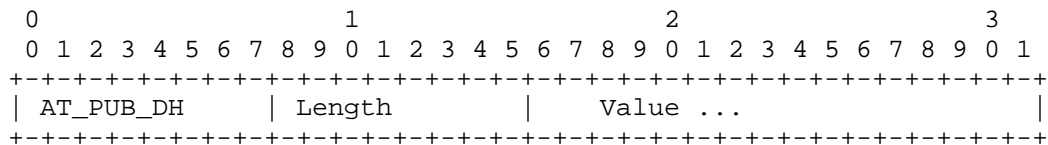
Figure 2: EAP-AKA' PFS Authentication Process

5. Extensions to EAP-AKA'

5.1. AT_PUB_DH

The AT_PUB_DH carries a Diffie-Hellman value.

The format of the AT_PUB_DH attribute is shown below.



The fields are as follows:

AT_PUB_DH

This is set to TBA1 BY IANA.

Length

The length of the attribute, set as other attributes in EAP-AKA [RFC4187].

Value

This value is the sender's Diffie-Hellman public value. For Curve25519, the length of this value is 32 bytes, represented as specified in [RFC8031] and [RFC7748].

To retain the security of the keys, the sender SHALL generate a fresh value for each run of the protocol.

5.2. AT_KDF_DH

The AT_KDF_DH indicates the used or desired key generation function, if the Perfect Forward Secrecy extension is taken into use. It will also at the same time indicate the used or desired Diffie-Hellman group. A new attribute is needed to carry this information, as AT_KDF carries the legacy KDF value for those EAP peers that cannot or do not want to use this extension.

The format of the AT_KDF_DH attribute is shown below.

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-----+-----+-----+-----+-----+-----+-----+-----+
      | AT_KDF_DH      | Length      | Key Derivation Function |
      +-----+-----+-----+-----+-----+-----+-----+-----+

```

The fields are as follows:

AT_KDF_DH

This is set to TBA2 BY IANA.

Length

The length of the attribute, MUST be set to 1.

Key Derivation Function

An enumerated value representing the key derivation function that the server (or peer) wishes to use. See Section 5.3 for the functions specified in this document. Note: This field has a different name space than the similar field in the AT_KDF attribute Key Derivation Function defined in [RFC5448].

Servers MUST send one or more AT_KDF_DH attributes in the EAP-Request /AKA'-Challenge message. These attributes represent the desired functions ordered by preference, the most preferred function being the first attribute.

Upon receiving a set of these attributes, if the peer supports and is willing to use the key derivation function indicated by the first attribute, and is willing and able to use the extension defined in this specification, the function is taken into use without any further negotiation. However, if the peer does not support this function or is unwilling to use it, it responds to the server with an indication that a different function is needed. Similarly with the negotiation process defined in [RFC5448] for AT_KDF, the peer sends EAP-Response/AKA'-Challenge message that contains only one attribute, AT_KDF_DH with the value set to the desired alternative function from among the ones suggested by the server earlier. If there is no suitable alternative, the peer has a choice of either falling back to EAP-AKA' or behaving as if AUTN had been incorrect and failing authentication (see Figure 3 of [RFC4187]). The peer MUST fail the authentication if there are any duplicate values within the list of AT_KDF_DH attributes (except where the duplication is due to a request to change the key derivation function; see below for further information).

If the peer does not recognize the extension defined in this specification or is unwilling to use it, it ignores the AT_KDF_DH attribute.

Upon receiving an EAP-Response/AKA'-Challenge with AT_KDF_DH from the peer, the server checks that the suggested AT_KDF_DH value was one of the alternatives in its offer. The first AT_KDF_DH value in the message from the server is not a valid alternative. If the peer has replied with the first AT_KDF_DH value, the server behaves as if AT_MAC of the response had been incorrect and fails the authentication. For an overview of the failed authentication process in the server side, see Section 3 and Figure 2 in [RFC4187]. Otherwise, the server re-sends the EAP-Response/AKA'-Challenge message, but adds the selected alternative to the beginning of the list of AT_KDF_DH attributes, and retains the entire list following it. Note that this means that the selected alternative appears twice in the set of AT_KDF values. Responding to the peer's request to change the key derivation function is the only legal situation where such duplication may occur.

When the peer receives the new EAP-Request/AKA'-Challenge message, it MUST check that the requested change, and only the requested change occurred in the list of AT_KDF_DH attributes. If yes, it continues. If not, it behaves as if AT_MAC had been incorrect and fails the authentication. If the peer receives multiple EAP-Request/AKA'-Challenge messages with differing AT_KDF_DH attributes without having requested negotiation, the peer MUST behave as if AT_MAC had been incorrect and fail the authentication.

5.3. New Key Derivation Function

A new Key Derivation Function type is defined for "EAP-AKA' with DH and Curve25519", represented by value 1. It represents a particular choice of key derivation function and at the same time selects a Diffie-Hellman group to be used.

The Key Derivation Function type value is only used in the AT_KDF_DH attribute, and should not be confused with the different range of key derivation functions that can be represented in the AT_KDF attribute as defined in [RFC5448].

Key derivation in this extension produces exactly the same keys for internal use within one authentication run as RFC 5448 EAP-AKA' did. For instance, K_aut that is used in AT_MAC is still exactly as it was in EAP-AKA'. The only change to key derivation is in re-authentication keys and keys exported out of the EAP method, MSK and EMSK. As a result, EAP-AKA' attributes such as AT_MAC continue to be usable even when this extension is in use.

When the Key Derivation Function field in the AT_KDF_DH attribute is set to 1 and the Key Derivation Function field in the AT_KDF attribute is also set to 1, the Master Key (MK) is derived and as follows below.

```
MK = PRF'(IK'|CK', "EAP-AKA'" | Identity)
MK_DH = PRF'(IK'|CK'|G^xy, "EAP-AKA' PFS" | Identity)
K_encr = MK[0..127]
K_aut = MK[128..383]
K_re = MK_DH[0..255]
MSK = MK_DH[256..767]
EMSK = MK_DH[768..1279]
```

The rest of computation proceeds as defined in Section 3.3 of [RFC5448].

For readability, an explanation of the notation used above is copied here: [n..m] denotes the substring from bit n to m. PRF' is a new pseudo-random function specified in [RFC5448]. K_encr is the encryption key, 128 bits, K_aut is the authentication key, 256 bits, K_re is the re-authentication key, 256 bits, MSK is the Master Session Key, 512 bits, and EMSK is the Extended Master Session Key, 512 bits. MSK and EMSK are outputs from a successful EAP method run [RFC3748].

CK and IK are produced by the AKA algorithm. IK' and CK' are derived as specified in [RFC5448] from IK and CK.

The value "EAP-AKA'" is an eight-characters-long ASCII string. It is used as is, without any trailing NUL characters. Similarly, "EAP-AKA' PFS" is a twelve-characters-long ASCII string, also used as is.

Identity is the peer identity as specified in Section 7 of [RFC4187].

5.4. Diffie-Hellman Groups

The selection of suitable groups for the Diffie-Hellman computation is necessary. The choice of a group is made at the same time as deciding to use of particular key derivation function in AT_KDF_DH. For "EAP-AKA' with DH and Curve25519" the Diffie-Hellman group is the Curve25519 group specified in [RFC8031].

5.5. Message Processing

This section specifies the changes related to message processing when this extension is used in EAP-AKA'. It specifies when a message may be transmitted or accepted, which attributes are allowed in a message, which attributes are required in a message, and other message-specific details, where those details are different for this extension than the base EAP-AKA' or EAP-AKA protocol. Unless otherwise specified here, the rules from [RFC5448] or [RFC4187] apply.

5.5.1. EAP-Request/AKA'-Identity

No changes, except that the AT_KDF_DH or AT_PUB_DH attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

5.5.2. EAP-Response/AKA'-Identity

No changes, except that the AT_KDF_DH or AT_PUB_DH attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

5.5.3. EAP-Request/AKA'-Challenge

The server sends the EAP-Request/AKA'-Challenge on full authentication as specified by [RFC4187] and [RFC5448]. The attributes AT_RAND, AT_AUTN, and AT_MAC MUST be included and checked on reception as specified in in [RFC4187]. They are also necessary for backwards compatibility.

In EAP-Request/AKA'-Challenge, there is no message-specific data covered by the MAC for the AT_MAC attribute. The AT_KDF_DH and AT_PUB_DH attributes MUST be included. The AT_PUB_DH attribute

carries the server's public Diffie-Hellman key. If either AT_KDF_DH or AT_PUB_DH is missing on reception, the peer MUST treat them as if neither one was sent, and the assume that the extension defined in this specification is not in use.

The AT_RESULT_IND, AT_CHECKCODE, AT_IV, AT_ENCR_DATA, AT_PADDING, AT_NEXT_PSEUDONYM, AT_NEXT_REAUTH_ID and other attributes may be included as specified in Section 9.3 of [RFC4187].

When processing this message, the peer MUST process AT_RAND, AT_AUTN, AT_KDF_DH, AT_PUB_DH before processing other attributes. Only if these attributes are verified to be valid, the peer derives keys and verifies AT_MAC. If the peer is unable or unwilling to perform the extension specified in this document, it proceeds as defined in [RFC5448]. Finally, the operation in case an error occurs is specified in Section 6.3.1. of [RFC4187].

5.5.4. EAP-Response/AKA'-Challenge

The peer sends EAP-Response/AKA'-Challenge in response to a valid EAP-Request/AKA'-Challenge message, as specified by [RFC4187] and [RFC5448]. If the peer supports and is willing to perform the extension specified in this protocol, and the server had made a valid request involving the attributes specified in Section 5.5.3, the peer responds per the rules specified below. Otherwise, the peer responds as specified in [RFC4187] and [RFC5448] and ignores the attributes related to this extension.

The AT_MAC attribute MUST be included and checked as specified in [RFC5448]. In EAP-Response/AKA'-Challenge, there is no message-specific data covered by the MAC. The AT_PUB_DH attribute MUST be included, and carries the peer's public Diffie-Hellman key.

The AT_RES attribute MUST be included and checked as specified in [RFC4187].

The AT_CHECKCODE, AT_RESULT_IND, AT_IV, AT_ENCR_DATA and other attributes may be included as specified in Section 9.4 of [RFC4187].

5.5.5. EAP-Request/AKA'-Reauthentication

No changes, but note that the re-authentication process uses the keys generated in the original EAP-AKA' authentication, which, if the extension specified in this documents is in use, employs key material from the Diffie-Hellman procedure.

5.5.6. EAP-Response/AKA'-Reauthentication

No changes, but as discussed in Section 5.5.5, re-authentication is based on the key material generated by EAP-AKA' and the extension defined in this document.

5.5.7. EAP-Response/AKA'-Synchronization-Failure

No changes, except that the AT_KDF_DH or AT_PUB_DH attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

5.5.8. EAP-Response/AKA'-Authentication-Reject

No changes, except that the AT_KDF_DH or AT_PUB_DH attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

5.5.9. EAP-Response/AKA'-Client-Error

No changes, except that the AT_KDF_DH or AT_PUB_DH attributes MUST NOT be added to this message. The appearance of these messages in a received message MUST be ignored.

5.5.10. EAP-Request/AKA'-Notification

No changes.

5.5.11. EAP-Response/AKA'-Notification

No changes.

6. Security Considerations

This section deals only with the changes to security considerations as they differ from EAP-AKA', or as new information has been gathered since the publication of [RFC5448].

The possibility of attacks against key storage offered in SIM or other smart cards has been a known threat. But as the discussion in Section 2.3 shows, the likelihood of practically feasible attacks has increased. Many of these attacks can be best dealt with improved processes, e.g., limiting the access to the key material within the factory or personnel, etc. But not all attacks can be entirely ruled out for well-resourced adversaries, irrespective of what the technical algorithms and protection measures are.

This extension can provide assistance in situations where there is a danger of attacks against the key material on SIM cards by adversaries that can not or who are unwilling to mount active attacks

against large number of sessions. This extension is most useful when used in a context where EAP keys are used without further mixing that can provide Perfect Forward Secrecy. For instance, when used with IKEv2, the session keys produced by IKEv2 have this property, so better characteristics of EAP keys is not that useful. However, typical link layer usage of EAP does not involve running Diffie-Hellman, so using EAP to authenticate access to a network is one situation where the extension defined in this document can be helpful.

The following security properties of EAP-AKA' are impacted through this extension:

Protected ciphersuite negotiation

EAP-AKA' has a negotiation mechanism for selecting the key derivation functions, and this mechanism has been extended by the extension specified in this document. The resulting mechanism continues to be secure against bidding down attacks.

There are two specific needs in the negotiation mechanism:

Negotiating key derivation function within the extension

The negotiation mechanism allows changing the offered key derivation function, but the change is visible in the final EAP-Request/KA'-Challenge message that the server sends to the peer. This message is authenticated via the AT_MAC attribute, and carries both the chosen alternative and the initially offered list. The peer refuses to accept a change it did not initiate. As a result, both parties are aware that a change is being made and what the original offer was.

Negotiating the use of this extension

This extension is offered by the server through presenting the AT_KDF_DH and AT_PUB_DH attributes in the EAP-Request/KA'-Challenge message. These attributes are protected by AT_MAC, so attempts to change or omit them by an adversary will be detected. (Except of course, if the adversary holds the long-term shared secret and is willing to engage in an active attack, but that is a case that cannot be solved by this protocol, or any protocol for that matter.) However, as discussed in the introduction, even an attacker with access to the long-term keys is required to be MITM on each KA run, which makes mass surveillance slightly more laborious.

Key derivation

This extension provides key material that is based on the Diffie-Hellman keys, yet bound to the authentication through the (U)SIM card. This means that subsequent payload communications between the parties are protected with keys that are not solely based on information in the clear (such as the RAND) and information derivable from the long-term shared secrets on the (U)SIM card. As a result, if anyone successfully recovers shared secret information, they are unable to decrypt communications protected by the keys generated through this extension. Note that the recovery of shared secret information could occur either before or after the time that the protected communications are used. When this extension is used, communications at time t_0 can be protected if at some later time t_1 an adversary learns of long-term shared secret and has access to a recording of the encrypted communications.

Obviously, this extension is still vulnerable to attackers that are willing to perform an active attack and who at the time of the attack have access to the long-term shared secret.

This extension does not change the properties of related to re-authentication. No new Diffie-Hellman run is performed during the re-authentication allowed by EAP-AKA'. However, if this extension was in use when the original EAP-AKA' authentication was performed, the keys used for re-authentication (K_{re}) are based on the Diffie-Hellman keys, and hence continue to be equally safe against expose of the long-term secrets as the original authentication.

7. IANA Considerations

This extension of EAP-AKA' shares its attribute space and subtypes with EAP-SIM [RFC4186], EAP-AKA [RFC4186], and EAP-AKA' [RFC5448].

Two new Attribute Type value (TBA1, TBA2) in the skippable range need to be assigned for `AT_PUB_DH` (Section 5.1) and `AT_KDF_DH` (Section 5.2) in the EAP-AKA and EAP-SIM Parameters registry under Attribute Types.

Also, a new registry should be created to represent Diffie-Hellman Key Derivation Function types. The "EAP-AKA' with DH and Curve25519" type (1, see Section 5.3) needs to be assigned, along with one reserved value. The initial contents of this namespace are therefore as below; new values can be created through the Specification Required policy [RFC8126].

| Value | Description | Reference |
|---------|---------------------------------|-------------------------|
| ----- | ----- | ----- |
| 0 | Reserved | [TBD BY IANA: THIS RFC] |
| 1 | EAP-AKA' with DH and Curve25519 | [TBD BY IANA: THIS RFC] |
| 2-65535 | Unassigned | |

8. References

8.1. Normative References

- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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, <<https://www.rfc-editor.org/info/rfc3748>>.
- [RFC4187] Arkko, J. and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, DOI 10.17487/RFC4187, January 2006, <<https://www.rfc-editor.org/info/rfc4187>>.
- [RFC5448] Arkko, J., Lehtovirta, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", RFC 5448, DOI 10.17487/RFC5448, May 2009, <<https://www.rfc-editor.org/info/rfc5448>>.
- [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, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <<https://www.rfc-editor.org/info/rfc7748>>.

- [RFC8031] Nir, Y. and S. Josefsson, "Curve25519 and Curve448 for the Internet Key Exchange Protocol Version 2 (IKEv2) Key Agreement", RFC 8031, DOI 10.17487/RFC8031, December 2016, <<https://www.rfc-editor.org/info/rfc8031>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

8.2. Informative References

- [RFC4186] Haverinen, H., Ed. and J. Salowey, Ed., "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", RFC 4186, DOI 10.17487/RFC4186, January 2006, <<https://www.rfc-editor.org/info/rfc4186>>.
- [TrustCom2015] Arkko, J., Norrman, K., Naslund, M., and B. Sahlin, "A USIM compatible 5G AKA protocol with perfect forward secrecy", August 2015 in Proceedings of the TrustCom 2015, IEEE.
- [CB2014] Choudhary, A. and R. Bhandari, "3GPP AKA Protocol: Simplified Authentication Process", December 2014, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 12.
- [MT2012] Mjolsnes, S. and J-K. Tsay, "A vulnerability in the UMTS and LTE authentication and key agreement protocols", October 2012, in Proceedings of the 6th international conference on Mathematical Methods, Models and Architectures for Computer Network Security: computer network security.
- [BT2013] Beekman, J. and C. Thompson, "Breaking Cell Phone Authentication: Vulnerabilities in AKA, IMS and Android", August 2013, in 7th USENIX Workshop on Offensive Technologies, WOOT '13.
- [Heist2015] Scahill, J. and J. Begley, "The great SIM heist", February 2015, in <https://firstlook.org/theintercept/2015/02/19/great-sim-heist/>.
- [DOW1992] Diffie, W., vanOorschot, P., and M. Wiener, "Authentication and Authenticated Key Exchanges", June

1992, in Designs, Codes and Cryptography 2 (2): pp. 107-125.

Appendix A. Acknowledgments

The authors would like to note that the technical solution in this document came out of the TrustCom paper [TrustCom2015], whose authors were J. Arkko, K. Norrman, M. Naslund, and B. Sahlin. This document uses also a lot of material from [RFC4187] by J. Arkko and H. Haverinen as well as [RFC5448] by J. Arkko, V. Lehtovirta, and P. Eronen.

The authors would also like to thank John Mattson, Mohit Sethi, Vesa Lehtovirta, Bengt Sahlin, Prajwol Kumar Nakarmi and many people at the GSMA and 3GPP groups for interesting discussions in this problem space.

Authors' Addresses

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@piuha.net

Karl Norrman
Ericsson
Stockholm 16483
Sweden

Email: karl.norrman@ericsson.com

Vesa Torvinen
Ericsson
Jorvas 02420
Finland

Email: vesa.torvinen@ericsson.com

Network Working Group
Internet-Draft
Obsoletes: 5448 (if approved)
Intended status: Informational
Expires: May 3, 2018

J. Arkko
V. Lehtovirta
Ericsson
P. Eronen
Nokia
October 30, 2017

Improved Extensible Authentication Protocol Method for 3rd Generation
Authentication and Key Agreement (EAP-AKA')
draft-arkko-eap-rfc5448bis-00

Abstract

This specification defines a new EAP method, EAP-AKA', a small revision of the EAP-AKA method. The change is a new key derivation function that binds the keys derived within the method to the name of the access network. The new key derivation mechanism has been defined in the 3rd Generation Partnership Project (3GPP). This specification allows its use in EAP in an interoperable manner. In addition, EAP-AKA' employs SHA-256 instead of SHA-1.

This specification also updates RFC 4187 EAP-AKA to prevent bidding down attacks from EAP-AKA'.

This version of the EAP-AKA' specification updates a reference to constructing one field in the protocol, so that EAP-AKA' becomes compatible with 5G deployments as well.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 3 |
| 2. Requirements Language | 4 |
| 3. EAP-AKA' | 4 |
| 3.1. AT_KDF_INPUT | 6 |
| 3.2. AT_KDF | 9 |
| 3.3. Key Generation | 11 |
| 3.4. Hash Functions | 13 |
| 3.4.1. PRF' | 13 |
| 3.4.2. AT_MAC | 13 |
| 3.4.3. AT_CHECKCODE | 13 |
| 4. Bidding Down Prevention for EAP-AKA | 14 |
| 5. Security Considerations | 15 |
| 5.1. Security Properties of Binding Network Names | 18 |
| 6. IANA Considerations | 19 |
| 6.1. Type Value | 19 |
| 6.2. Attribute Type Values | 19 |
| 6.3. Key Derivation Function Namespace | 20 |
| 7. Contributors | 20 |
| 8. Acknowledgments | 20 |
| 9. References | 20 |
| 9.1. Normative References | 20 |
| 9.2. Informative References | 22 |
| Appendix A. Changes from RFC 5448 | 23 |
| Appendix B. Changes from RFC 4187 to RFC 5448 | 23 |
| Appendix C. Importance of Explicit Negotiation | 23 |
| Appendix D. Test Vectors | 24 |
| Authors' Addresses | 27 |

1. Introduction

This specification defines a new Extensible Authentication Protocol (EAP)[RFC3748] method, EAP-AKA', a small revision of the EAP-AKA method originally defined in [RFC4187]. What is new in EAP-AKA' is that it has a new key derivation function, specified in [TS-3GPP.33.402]. This function binds the keys derived within the method to the name of the access network. This limits the effects of compromised access network nodes and keys. This specification defines the EAP encapsulation for AKA when the new key derivation mechanism is in use.

3GPP has defined a number of applications for the revised AKA mechanism, some based on native encapsulation of AKA over 3GPP radio access networks and others based on the use of EAP.

For making the new key derivation mechanisms usable in EAP-AKA, additional protocol mechanisms are necessary. Given that RFC 4187 calls for the use of CK (the encryption key) and IK (the integrity key) from AKA, existing implementations continue to use these. Any change of the key derivation must be unambiguous to both sides in the protocol. That is, it must not be possible to accidentally connect old equipment to new equipment and get the key derivation wrong or attempt to use wrong keys without getting a proper error message. The change must also be secure against bidding down attacks that attempt to force the participants to use the least secure mechanism.

This specification therefore introduces a variant of the EAP-AKA method, called EAP-AKA'. This method can employ the derived keys CK' and IK' from the 3GPP specification and updates the used hash function to SHA-256 [FIPS.180-2.2002]. But it is otherwise equivalent to RFC 4187. Given that a different EAP method type value is used for EAP-AKA and EAP-AKA', a mutually supported method may be negotiated using the standard mechanisms in EAP [RFC3748].

Note: Appendix C explains why it is important to be explicit about the change of semantics for the keys, and why other approaches would lead to severe interoperability problems.

This version of the EAP-AKA' specification is an update to RFC 5448. The update is to the reference on how the Network Name field is constructed in the protocol. The update helps ensure that EAP-AKA' becomes compatible with 5G deployments as well. RFC 5448 referred to the 2008 version of that reference ([TS-3GPP.24.302]) and this update points to the 5G version of that reference.

Arguably, the update is small, as the 3GPP specification number for the updated calculation has not changed, only the version. But this

reference is crucial in correct calculation of the keys resulting from running the EAP-AKA' method, so an update of the RFC with the newest version pointer may be warranted. As always, feedback is welcome on that point as well as on any other topic within this document.

Note: It is an open issue whether this update should refer to only the 5G version of the definition, or be explicit that any further update of that specification is something that EAP-AKA' implementations should take into account. Note that one should keep in mind that specification being automatically updated is different from implementations taking notice of new things.

It is an explicit non-goal of this draft to include any other technical modifications, addition of new features or other changes. The EAP-AKA' base protocol is stable and needs to stay that way. If there are any extensions or variants, those need to be proposed as standalone extensions or even as different authentication methods.

The rest of this specification is structured as follows. Section 3 defines the EAP-AKA' method. Section 4 adds support to EAP-AKA to prevent bidding down attacks from EAP-AKA'. Section 5 explains the security differences between EAP-AKA and EAP-AKA'. Section 6 describes the IANA considerations and Appendix A and Appendix B explains what updates to RFC 5448 AKA' and RFC 4187 EAP-AKA have been made in this specification. Appendix C explains some of the design rationale for creating EAP-AKA' Finally, Appendix D provides test vectors.

Editor's Note: The publication of this RFC depends on its normative references [TS-3GPP.24.302] and [TS-3GPP.33.501] from 3GPP reaching their final Release 15 status at 3GPP. This is expected to happen shortly. The RFC Editor should check with the 3GPP liaisons that this has happened. RFC Editor: Please delete this note upon publication of this specification as an RFC.

2. Requirements Language

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].

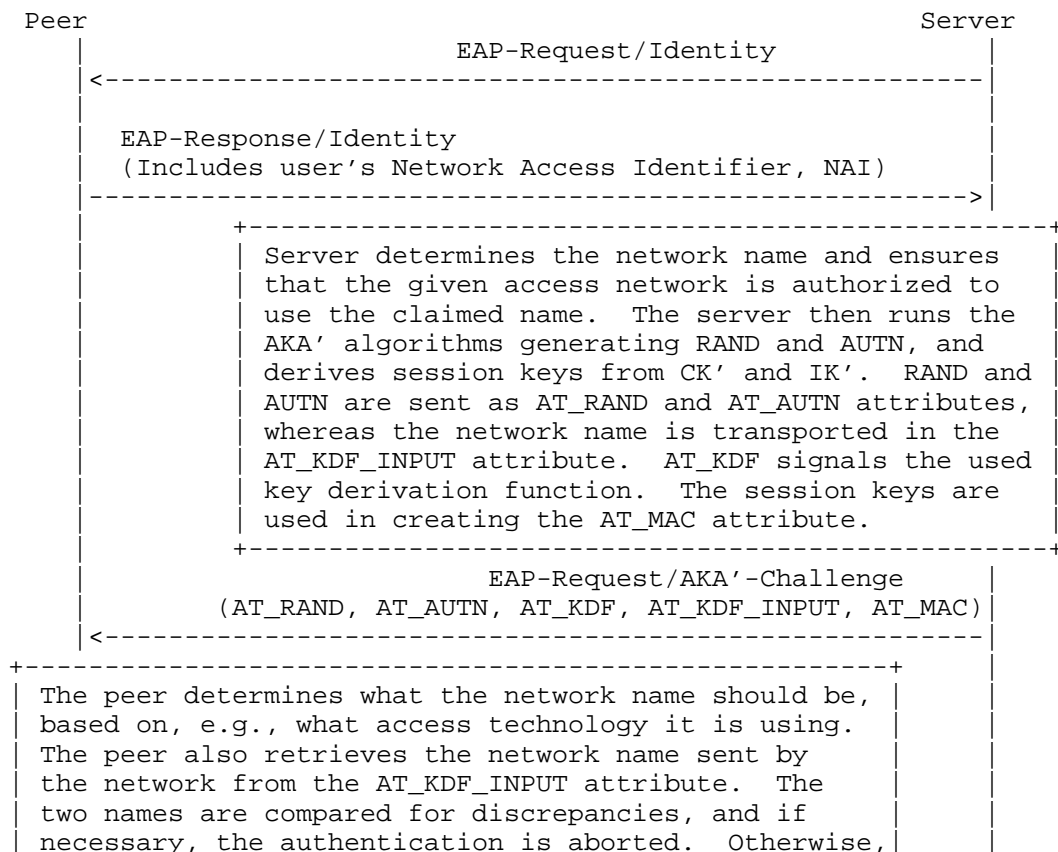
3. EAP-AKA'

EAP-AKA' is a new EAP method that follows the EAP-AKA specification [RFC4187] in all respects except the following:

- o It uses the Type code 50, not 23 (which is used by EAP-AKA).

- o It carries the AT_KDF_INPUT attribute, as defined in Section 3.1, to ensure that both the peer and server know the name of the access network.
- o It supports key derivation function negotiation via the AT_KDF attribute (Section 3.2) to allow for future extensions.
- o It calculates keys as defined in Section 3.3, not as defined in EAP-AKA.
- o It employs SHA-256 [FIPS.180-2.2002], not SHA-1 [FIPS.180-1.1995] (Section 3.4).

Figure 1 shows an example of the authentication process. Each message AKA'-Challenge and so on represents the corresponding message from EAP-AKA, but with EAP-AKA' Type code. The definition of these messages, along with the definition of attributes AT_RANDOM, AT_AUTN, AT_MAC, and AT_RES can be found in [RFC4187].



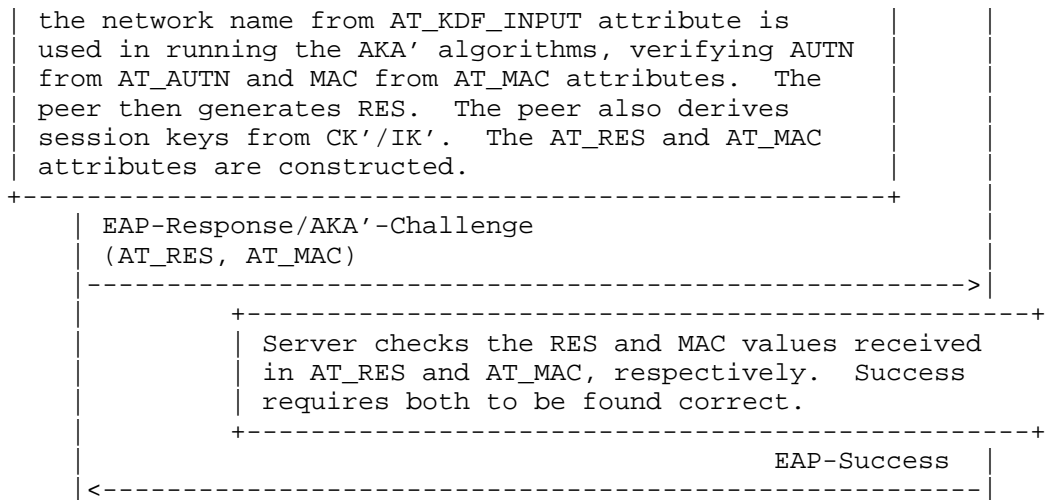
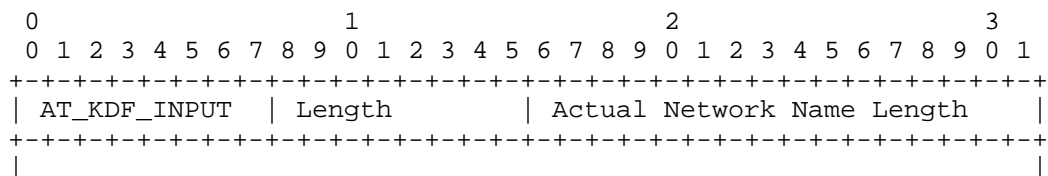


Figure 1: EAP-AKA' Authentication Process

EAP-AKA' can operate on the same credentials as EAP-AKA and employ the same identities. However, EAP-AKA' employs different leading characters than EAP-AKA for the conventions given in Section 4.1.1 of [RFC4187] for International Mobile Subscriber Identifier (IMSI) based usernames. EAP-AKA' MUST use the leading character "6" (ASCII 36 hexadecimal) instead of "0" for IMSI-based permanent usernames. All other usage and processing of the leading characters, usernames, and identities is as defined by EAP-AKA [RFC4187]. For instance, the pseudonym and fast re-authentication usernames need to be constructed so that the server can recognize them. As an example, a pseudonym could begin with a leading "7" character (ASCII 37 hexadecimal) and a fast re-authentication username could begin with "8" (ASCII 38 hexadecimal). Note that a server that implements only EAP-AKA may not recognize these leading characters. According to Section 4.1.4 of [RFC4187], such a server will re-request the identity via the EAP-Request/AKA-Identity message, making obvious to the peer that EAP-AKA and associated identity are expected.

3.1. AT_KDF_INPUT

The format of the AT_KDF_INPUT attribute is shown below.



```

      .                               Network Name                               .
      |                                                                           |
      +-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

The fields are as follows:

AT_KDF_INPUT

This is set to 23.

Length

The length of the attribute, calculated as defined in [RFC4187], Section 8.1.

Actual Network Name Length

This is a 2 byte actual length field, needed due to the requirement that the previous field is expressed in multiples of 4 bytes per the usual EAP-AKA rules. The Actual Network Name Length field provides the length of the network name in bytes.

Network Name

This field contains the network name of the access network for which the authentication is being performed. The name does not include any terminating null characters. Because the length of the entire attribute must be a multiple of 4 bytes, the sender pads the name with 1, 2, or 3 bytes of all zero bits when necessary.

Only the server sends the AT_KDF_INPUT attribute. Per [TS-3GPP.33.402], the server always verifies the authorization of a given access network to use a particular name before sending it to the peer over EAP-AKA'. The value of the AT_KDF_INPUT attribute from the server MUST be non-empty. If it is empty, the peer behaves as if AUTN had been incorrect and authentication fails. See Section 3 and Figure 3 of [RFC4187] for an overview of how authentication failures are handled.

In addition, the peer MAY check the received value against its own understanding of the network name. Upon detecting a discrepancy, the peer either warns the user and continues, or fails the authentication process. More specifically, the peer SHOULD have a configurable policy that it can follow under these circumstances. If the policy indicates that it can continue, the peer SHOULD log a warning message

or display it to the user. If the peer chooses to proceed, it MUST use the network name as received in the AT_KDF_INPUT attribute. If the policy indicates that the authentication should fail, the peer behaves as if AUTN had been incorrect and authentication fails.

The Network Name field contains a UTF-8 string. This string MUST be constructed as specified in [TS-3GPP.24.302] for "Access Network Identity". The string is structured as fields separated by colons (:). The algorithms and mechanisms to construct the identity string depend on the used access technology.

On the network side, the network name construction is a configuration issue in an access network and an authorization check in the authentication server. On the peer, the network name is constructed based on the local observations. For instance, the peer knows which access technology it is using on the link, it can see information in a link-layer beacon, and so on. The construction rules specify how this information maps to an access network name. Typically, the network name consists of the name of the access technology, or the name of the access technology followed by some operator identifier that was advertised in a link-layer beacon. In all cases, [TS-3GPP.24.302] is the normative specification for the construction in both the network and peer side. If the peer policy allows running EAP-AKA' over an access technology for which that specification does not provide network name construction rules, the peer SHOULD rely only on the information from the AT_KDF_INPUT attribute and not perform a comparison.

If a comparison of the locally determined network name and the one received over EAP-AKA' is performed on the peer, it MUST be done as follows. First, each name is broken down to the fields separated by colons. If one of the names has more colons and fields than the other one, the additional fields are ignored. The remaining sequences of fields are compared, and they match only if they are equal character by character. This algorithm allows a prefix match where the peer would be able to match "", "FOO", and "FOO:BAR" against the value "FOO:BAR" received from the server. This capability is important in order to allow possible updates to the specifications that dictate how the network names are constructed. For instance, if a peer knows that it is running on access technology "FOO", it can use the string "FOO" even if the server uses an additional, more accurate description, e.g., "FOO:BAR", that contains more information.

The allocation procedures in [TS-3GPP.24.302] ensure that conflicts potentially arising from using the same name in different types of networks are avoided. The specification also has detailed rules about how a client can determine these based on information available

to the client, such as the type of protocol used to attach to the network, beacons sent out by the network, and so on. Information that the client cannot directly observe (such as the type or version of the home network) is not used by this algorithm.

The AT_KDF_INPUT attribute MUST be sent and processed as explained above when AT_KDF attribute has the value 1. Future definitions of new AT_KDF values MUST define how this attribute is sent and processed.

3.2. AT_KDF

AT_KDF is an attribute that the server uses to reference a specific key derivation function. It offers a negotiation capability that can be useful for future evolution of the key derivation functions.

The format of the AT_KDF attribute is shown below.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| AT_KDF | Length | Key Derivation Function |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The fields are as follows:

AT_KDF

This is set to 24.

Length

The length of the attribute, MUST be set to 1.

Key Derivation Function

An enumerated value representing the key derivation function that the server (or peer) wishes to use. Value 1 represents the default key derivation function for EAP-AKA', i.e., employing CK' and IK' as defined in Section 3.3.

Servers MUST send one or more AT_KDF attributes in the EAP-Request/ AKA'-Challenge message. These attributes represent the desired functions ordered by preference, the most preferred function being the first attribute.

Upon receiving a set of these attributes, if the peer supports and is willing to use the key derivation function indicated by the first attribute, the function is taken into use without any further negotiation. However, if the peer does not support this function or is unwilling to use it, it does not process the received EAP-Request/AKA'-Challenge in any way except by responding with the EAP-Response/AKA'-Challenge message that contains only one attribute, AT_KDF with the value set to the selected alternative. If there is no suitable alternative, the peer behaves as if AUTN had been incorrect and authentication fails (see Figure 3 of [RFC4187]). The peer fails the authentication also if there are any duplicate values within the list of AT_KDF attributes (except where the duplication is due to a request to change the key derivation function; see below for further information).

Upon receiving an EAP-Response/AKA'-Challenge with AT_KDF from the peer, the server checks that the suggested AT_KDF value was one of the alternatives in its offer. The first AT_KDF value in the message from the server is not a valid alternative. If the peer has replied with the first AT_KDF value, the server behaves as if AT_MAC of the response had been incorrect and fails the authentication. For an overview of the failed authentication process in the server side, see Section 3 and Figure 2 of [RFC4187]. Otherwise, the server re-sends the EAP-Response/AKA'-Challenge message, but adds the selected alternative to the beginning of the list of AT_KDF attributes and retains the entire list following it. Note that this means that the selected alternative appears twice in the set of AT_KDF values. Responding to the peer's request to change the key derivation function is the only legal situation where such duplication may occur.

When the peer receives the new EAP-Request/AKA'-Challenge message, it MUST check that the requested change, and only the requested change, occurred in the list of AT_KDF attributes. If so, it continues with processing the received EAP-Request/AKA'-Challenge as specified in [RFC4187] and Section 3.1 of this document. If not, it behaves as if AT_MAC had been incorrect and fails the authentication. If the peer receives multiple EAP-Request/AKA'-Challenge messages with differing AT_KDF attributes without having requested negotiation, the peer MUST behave as if AT_MAC had been incorrect and fail the authentication.

Note that the peer may also request sequence number resynchronization [RFC4187]. This happens after AT_KDF negotiation has already completed. An AKA'-Synchronization-Failure message is sent as a response to the newly received EAP-Request/AKA'-Challenge (the last message of the AT_KDF negotiation). The AKA'-Synchronization-Failure message MUST contain the AUTS parameter as specified in [RFC4187] and a copy the AT_KDF attributes as they appeared in the last message of

the AT_KDF negotiation. If the AT_KDF attributes are found to differ from their earlier values, the peer and server MUST behave as if AT_MAC had been incorrect and fail the authentication.

3.3. Key Generation

Both the peer and server MUST derive the keys as follows.

AT_KDF set to 1

In this case, MK is derived and used as follows:

```
MK = PRF'(IK'|CK',"EAP-AKA'"|Identity)
K_encr = MK[0..127]
K_aut  = MK[128..383]
K_re   = MK[384..639]
MSK    = MK[640..1151]
EMSK   = MK[1152..1663]
```

Here [n..m] denotes the substring from bit n to m. PRF' is a new pseudo-random function specified in Section 3.4. The first 1664 bits from its output are used for K_encr (encryption key, 128 bits), K_aut (authentication key, 256 bits), K_re (re-authentication key, 256 bits), MSK (Master Session Key, 512 bits), and EMSK (Extended Master Session Key, 512 bits). These keys are used by the subsequent EAP-AKA' process. K_encr is used by the AT_ENCR_DATA attribute, and K_aut by the AT_MAC attribute. K_re is used later in this section. MSK and EMSK are outputs from a successful EAP method run [RFC3748].

IK' and CK' are derived as specified in [TS-3GPP.33.402]. The functions that derive IK' and CK' take the following parameters: CK and IK produced by the AKA algorithm, and value of the Network Name field comes from the AT_KDF_INPUT attribute (without length or padding) .

The value "EAP-AKA'" is an eight-characters-long ASCII string. It is used as is, without any trailing NUL characters.

Identity is the peer identity as specified in Section 7 of [RFC4187].

When the server creates an AKA challenge and corresponding AUTN, CK, CK', IK, and IK' values, it MUST set the Authentication Management Field (AMF) separation bit to 1 in the AKA algorithm [TS-3GPP.33.102]. Similarly, the peer MUST check that the AMF separation bit is set to 1. If the bit is not set to 1, the peer

behaves as if the AUTN had been incorrect and fails the authentication.

On fast re-authentication, the following keys are calculated:

```
MK = PRF'(K_re, "EAP-AKA' re-auth" | Identity | counter | NONCE_S)
MSK = MK[0..511]
EMSK = MK[512..1023]
```

MSK and EMSK are the resulting 512-bit keys, taking the first 1024 bits from the result of PRF'. Note that K_encr and K_aut are not re-derived on fast re-authentication. K_re is the re-authentication key from the preceding full authentication and stays unchanged over any fast re-authentication(s) that may happen based on it. The value "EAP-AKA' re-auth" is a sixteen-characters-long ASCII string, again represented without any trailing NUL characters. Identity is the fast re-authentication identity, counter is the value from the AT_COUNTER attribute, NONCE_S is the nonce value from the AT_NONCE_S attribute, all as specified in Section 7 of [RFC4187]. To prevent the use of compromised keys in other places, it is forbidden to change the network name when going from the full to the fast re-authentication process. The peer SHOULD NOT attempt fast re-authentication when it knows that the network name in the current access network is different from the one in the initial, full authentication. Upon seeing a re-authentication request with a changed network name, the server SHOULD behave as if the re-authentication identifier had been unrecognized, and fall back to full authentication. The server observes the change in the name by comparing where the fast re-authentication and full authentication EAP transactions were received at the Authentication, Authorization, and Accounting (AAA) protocol level.

AT_KDF has any other value

Future variations of key derivation functions may be defined, and they will be represented by new values of AT_KDF. If the peer does not recognize the value, it cannot calculate the keys and behaves as explained in Section 3.2.

AT_KDF is missing

The peer behaves as if the AUTN had been incorrect and MUST fail the authentication.

If the peer supports a given key derivation function but is unwilling to perform it for policy reasons, it refuses to calculate the keys and behaves as explained in Section 3.2.

3.4. Hash Functions

EAP-AKA' uses SHA-256 [FIPS.180-2.2002], not SHA-1 [FIPS.180-1.1995] as in EAP-AKA. This requires a change to the pseudo-random function (PRF) as well as the AT_MAC and AT_CHECKCODE attributes.

3.4.1. PRF'

The PRF' construction is the same one IKEv2 uses (see Section 2.13 of [RFC4306]). The function takes two arguments. K is a 256-bit value and S is an octet string of arbitrary length. PRF' is defined as follows:

$$\text{PRF}'(K, S) = T1 \mid T2 \mid T3 \mid T4 \mid \dots$$

where:

T1 = HMAC-SHA-256 (K, S | 0x01)
 T2 = HMAC-SHA-256 (K, T1 | S | 0x02)
 T3 = HMAC-SHA-256 (K, T2 | S | 0x03)
 T4 = HMAC-SHA-256 (K, T3 | S | 0x04)
 ...

PRF' produces as many bits of output as is needed. HMAC-SHA-256 is the application of HMAC [RFC2104] to SHA-256.

3.4.2. AT_MAC

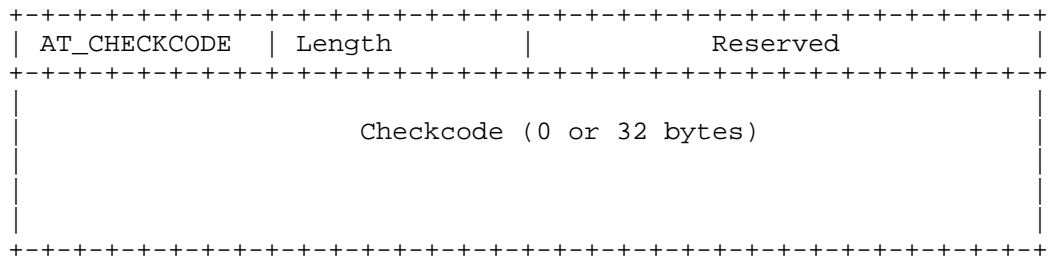
When used within EAP-AKA', the AT_MAC attribute is changed as follows. The MAC algorithm is HMAC-SHA-256-128, a keyed hash value. The HMAC-SHA-256-128 value is obtained from the 32-byte HMAC-SHA-256 value by truncating the output to the first 16 bytes. Hence, the length of the MAC is 16 bytes.

Otherwise, the use of AT_MAC in EAP-AKA' follows Section 10.15 of [RFC4187].

3.4.3. AT_CHECKCODE

When used within EAP-AKA', the AT_CHECKCODE attribute is changed as follows. First, a 32-byte value is needed to accommodate a 256-bit hash output:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 1 | | 2 | | 3 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |



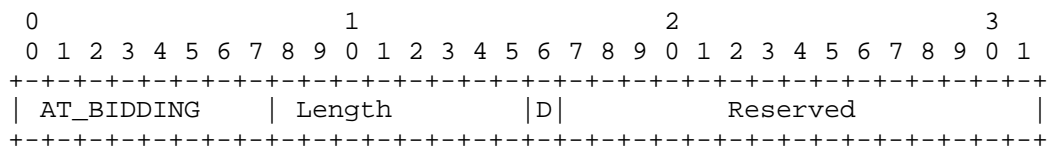
Second, the checkcode is a hash value, calculated with SHA-256 [FIPS.180-2.2002], over the data specified in Section 10.13 of [RFC4187].

4. Bidding Down Prevention for EAP-AKA

As discussed in [RFC3748], negotiation of methods within EAP is insecure. That is, a man-in-the-middle attacker may force the endpoints to use a method that is not the strongest that they both support. This is a problem, as we expect EAP-AKA and EAP-AKA' to be negotiated via EAP.

In order to prevent such attacks, this RFC specifies a new mechanism for EAP-AKA that allows the endpoints to securely discover the capabilities of each other. This mechanism comes in the form of the AT_BIDDING attribute. This allows both endpoints to communicate their desire and support for EAP-AKA' when exchanging EAP-AKA messages. This attribute is not included in EAP-AKA' messages as defined in this RFC. It is only included in EAP-AKA messages. This is based on the assumption that EAP-AKA' is always preferable (see Section 5). If during the EAP-AKA authentication process it is discovered that both endpoints would have been able to use EAP-AKA', the authentication process SHOULD be aborted, as a bidding down attack may have happened.

The format of the AT_BIDDING attribute is shown below.



The fields are as follows:

AT_BIDDING

This is set to 136.

Length

The length of the attribute, MUST be set to 1.

D

This bit is set to 1 if the sender supports EAP-AKA', is willing to use it, and prefers it over EAP-AKA. Otherwise, it should be set to zero.

Reserved

This field MUST be set to zero when sent and ignored on receipt.

The server sends this attribute in the EAP-Request/AKA-Challenge message. If the peer supports EAP-AKA', it compares the received value to its own capabilities. If it turns out that both the server and peer would have been able to use EAP-AKA' and preferred it over EAP-AKA, the peer behaves as if AUTN had been incorrect and fails the authentication (see Figure 3 of [RFC4187]). A peer not supporting EAP-AKA' will simply ignore this attribute. In all cases, the attribute is protected by the integrity mechanisms of EAP-AKA, so it cannot be removed by a man-in-the-middle attacker.

Note that we assume (Section 5) that EAP-AKA' is always stronger than EAP-AKA. As a result, there is no need to prevent bidding "down" attacks in the other direction, i.e., attackers forcing the endpoints to use EAP-AKA'.

5. Security Considerations

A summary of the security properties of EAP-AKA' follows. These properties are very similar to those in EAP-AKA. We assume that SHA-256 is at least as secure as SHA-1. This is called the SHA-256 assumption in the remainder of this section. Under this assumption, EAP-AKA' is at least as secure as EAP-AKA.

If the AT_KDF attribute has value 1, then the security properties of EAP-AKA' are as follows:

Protected ciphersuite negotiation

EAP-AKA' has no ciphersuite negotiation mechanisms. It does have a negotiation mechanism for selecting the key derivation functions. This mechanism is secure against bidding down attacks. The negotiation mechanism allows changing the offered key

derivation function, but the change is visible in the final EAP-Request/AKA'-Challenge message that the server sends to the peer. This message is authenticated via the AT_MAC attribute, and carries both the chosen alternative and the initially offered list. The peer refuses to accept a change it did not initiate. As a result, both parties are aware that a change is being made and what the original offer was.

Mutual authentication

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Integrity protection

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good (most likely better) as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details. The only difference is that a stronger hash algorithm, SHA-256, is used instead of SHA-1.

Replay protection

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Confidentiality

The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Key derivation

EAP-AKA' supports key derivation with an effective key strength against brute force attacks equal to the minimum of the length of the derived keys and the length of the AKA base key, i.e., 128 bits or more. The key hierarchy is specified in Section 3.3.

The Transient EAP Keys used to protect EAP-AKA packets (K_{encr} , K_{aut} , K_{re}), the MSK, and the EMSK are cryptographically separate. If we make the assumption that SHA-256 behaves as a pseudo-random function, an attacker is incapable of deriving any non-trivial information about any of these keys based on the other keys. An attacker also cannot calculate the pre-shared secret from IK , CK , IK' , CK' , K_{encr} , K_{aut} , K_{re} , MSK, or EMSK by any practically feasible means.

EAP-AKA' adds an additional layer of key derivation functions within itself to protect against the use of compromised keys. This is discussed further in Section 5.1.

EAP-AKA' uses a pseudo-random function modeled after the one used in IKEv2 [RFC4306] together with SHA-256.

Key strength

See above.

Dictionary attack resistance

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Fast reconnect

Under the SHA-256 assumption, the properties of EAP-AKA' are at least as good as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details. Note that implementations MUST prevent performing a fast reconnect across method types.

Cryptographic binding

Note that this term refers to a very specific form of binding, something that is performed between two layers of authentication. It is not the same as the binding to a particular network name. The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect, i.e., as it is not a tunnel method, this property is not applicable to it. Refer to [RFC4187], Section 12 for further details.

Session independence

The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Fragmentation

The properties of EAP-AKA' are exactly the same as those of EAP-AKA in this respect. Refer to [RFC4187], Section 12 for further details.

Channel binding

EAP-AKA', like EAP-AKA, does not provide channel bindings as they're defined in [RFC3748] and [RFC5247]. New skippable attributes can be used to add channel binding support in the future, if required.

However, including the Network Name field in the AKA' algorithms (which are also used for other purposes than EAP-AKA') provides a form of cryptographic separation between different network names, which resembles channel bindings. However, the network name does not typically identify the EAP (pass-through) authenticator. See the following section for more discussion.

5.1. Security Properties of Binding Network Names

The ability of EAP-AKA' to bind the network name into the used keys provides some additional protection against key leakage to inappropriate parties. The keys used in the protocol are specific to a particular network name. If key leakage occurs due to an accident, access node compromise, or another attack, the leaked keys are only useful when providing access with that name. For instance, a malicious access point cannot claim to be network Y if it has stolen keys from network X. Obviously, if an access point is compromised, the malicious node can still represent the compromised node. As a result, neither EAP-AKA' nor any other extension can prevent such attacks; however, the binding to a particular name limits the attacker's choices, allows better tracking of attacks, makes it possible to identify compromised networks, and applies good cryptographic hygiene.

The server receives the EAP transaction from a given access network, and verifies that the claim from the access network corresponds to the name that this access network should be using. It becomes impossible for an access network to claim over AAA that it is another access network. In addition, if the peer checks that the information it has received locally over the network-access link layer matches with the information the server has given it via EAP-AKA', it becomes

impossible for the access network to tell one story to the AAA network and another one to the peer. These checks prevent some "lying NAS" (Network Access Server) attacks. For instance, a roaming partner, R, might claim that it is the home network H in an effort to lure peers to connect to itself. Such an attack would be beneficial for the roaming partner if it can attract more users, and damaging for the users if their access costs in R are higher than those in other alternative networks, such as H.

Any attacker who gets hold of the keys CK and IK, produced by the AKA algorithm, can compute the keys CK' and IK' and, hence, the Master Key (MK) according to the rules in Section 3.3. The attacker could then act as a lying NAS. In 3GPP systems in general, the keys CK and IK have been distributed to, for instance, nodes in a visited access network where they may be vulnerable. In order to reduce this risk, the AKA algorithm MUST be computed with the AMF separation bit set to 1, and the peer MUST check that this is indeed the case whenever it runs EAP-AKA'. Furthermore, [TS-3GPP.33.402] requires that no CK or IK keys computed in this way ever leave the home subscriber system.

The additional security benefits obtained from the binding depend obviously on the way names are assigned to different access networks. This is specified in [TS-3GPP.24.302]. See also [TS-3GPP.23.003]. Ideally, the names allow separating each different access technology, each different access network, and each different NAS within a domain. If this is not possible, the full benefits may not be achieved. For instance, if the names identify just an access technology, use of compromised keys in a different technology can be prevented, but it is not possible to prevent their use by other domains or devices using the same technology.

6. IANA Considerations

6.1. Type Value

EAP-AKA' has the EAP Type value 50 in the Extensible Authentication Protocol (EAP) Registry under Method Types. Per Section 6.2 of [RFC3748], this allocation can be made with Designated Expert and Specification Required.

6.2. Attribute Type Values

EAP-AKA' shares its attribute space and subtypes with EAP-SIM [RFC4186] and EAP-AKA [RFC4187]. No new registries are needed.

However, a new Attribute Type value (23) in the non-skippable range has been assigned for AT_KDF_INPUT (Section 3.1) in the EAP-AKA and EAP-SIM Parameters registry under Attribute Types.

Also, a new Attribute Type value (24) in the non-skippable range has been assigned for AT_KDF (Section 3.2).

Finally, a new Attribute Type value (136) in the skippable range has been assigned for AT_BIDDING (Section 4).

6.3. Key Derivation Function Namespace

IANA has also created a new namespace for EAP-AKA' AT_KDF Key Derivation Function Values. This namespace exists under the EAP-AKA and EAP-SIM Parameters registry. The initial contents of this namespace are given below; new values can be created through the Specification Required policy [RFC5226].

| Value | Description | Reference |
|---------|-----------------------|------------|
| 0 | Reserved | [RFC 5448] |
| 1 | EAP-AKA' with CK'/IK' | [RFC 5448] |
| 2-65535 | Unassigned | |

7. Contributors

The test vectors in Appendix C were provided by Yogendra Pal and Jouni Malinen, based on two independent implementations of this specification.

8. Acknowledgments

The authors would like to thank Guenther Horn, Joe Salowey, Mats Naslund, Adrian Escott, Brian Rosenberg, Laksminath Dondeti, Ahmad Muhanna, Stefan Rommer, Miguel Garcia, Jan Kall, Ankur Agarwal, Jouni Malinen, Brian Weis, Russ Housley, Alfred Hoenes, Vesa Torvinen, and Mohit Sethi for their in-depth reviews and interesting discussions in this problem space.

9. References

9.1. Normative References

[TS-3GPP.24.302]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks; Stage 3; (Release 15)", 3GPP Draft Technical Specification 24.302, September 2017.

[TS-3GPP.33.102]

3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture (Release 8)", 3GPP Technical Specification 33.102, December 2008.

[TS-3GPP.33.402]

3GPP, "3GPP System Architecture Evolution (SAE); Security aspects of non-3GPP accesses; Release 8", 3GPP Technical Specification 33.402, December 2008.

[TS-3GPP.33.501]

3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture and procedures for 5G System; Release 15", 3GPP Technical Specification 33.501, August 2017.

[FIPS.180-2.2002]

National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-2, August 2002, <<http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf>>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.

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

[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, <<https://www.rfc-editor.org/info/rfc3748>>.

[RFC4187] Arkko, J. and H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", RFC 4187, DOI 10.17487/RFC4187, January 2006, <<https://www.rfc-editor.org/info/rfc4187>>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.

9.2. Informative References

- [TS-3GPP.23.003]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Numbering, addressing and identification (Release 8)", 3GPP Technical Specification 23.003, December 2008.
- [TS-3GPP.35.208]
3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 4: Design Conformance Test Data (Release 8)", 3GPP Technical Specification 35.208, December 2008.
- [FIPS.180-1.1995]
National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-1, April 1995, <<http://www.itl.nist.gov/fipspubs/fip180-1.htm>>.
- [RFC4186] Haverinen, H., Ed. and J. Salowey, Ed., "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", RFC 4186, DOI 10.17487/RFC4186, January 2006, <<https://www.rfc-editor.org/info/rfc4186>>.
- [RFC4284] Adrangi, F., Lortz, V., Bari, F., and P. Eronen, "Identity Selection Hints for the Extensible Authentication Protocol (EAP)", RFC 4284, DOI 10.17487/RFC4284, January 2006, <<https://www.rfc-editor.org/info/rfc4284>>.
- [RFC4306] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, DOI 10.17487/RFC4306, December 2005, <<https://www.rfc-editor.org/info/rfc4306>>.
- [RFC5113] Arkko, J., Aboba, B., Korhonen, J., Ed., and F. Bari, "Network Discovery and Selection Problem", RFC 5113, DOI 10.17487/RFC5113, January 2008, <<https://www.rfc-editor.org/info/rfc5113>>.
- [RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <<https://www.rfc-editor.org/info/rfc5247>>.

Appendix A. Changes from RFC 5448

The changes consist solely of referring to a newer version of [TS-3GPP.24.302]. The new version includes an updated definition of the Network Name field, to include 5G.

Appendix B. Changes from RFC 4187 to RFC 5448

The changes to RFC 4187 relate only to the bidding down prevention support defined in Section 4. In particular, this document does not change how the Master Key (MK) is calculated in RFC 4187 (it uses CK and IK, not CK' and IK'); neither is any processing of the AMF bit added to RFC 4187.

Appendix C. Importance of Explicit Negotiation

Choosing between the traditional and revised AKA key derivation functions is easy when their use is unambiguously tied to a particular radio access network, e.g., Long Term Evolution (LTE) as defined by 3GPP or evolved High Rate Packet Data (eHRPD) as defined by 3GPP2. There is no possibility for interoperability problems if this radio access network is always used in conjunction with new protocols that cannot be mixed with the old ones; clients will always know whether they are connecting to the old or new system.

However, using the new key derivation functions over EAP introduces several degrees of separation, making the choice of the correct key derivation functions much harder. Many different types of networks employ EAP. Most of these networks have no means to carry any information about what is expected from the authentication process. EAP itself is severely limited in carrying any additional information, as noted in [RFC4284] and [RFC5113]. Even if these networks or EAP were extended to carry additional information, it would not affect millions of deployed access networks and clients attaching to them.

Simply changing the key derivation functions that EAP-AKA [RFC4187] uses would cause interoperability problems with all of the existing implementations. Perhaps it would be possible to employ strict separation into domain names that should be used by the new clients and networks. Only these new devices would then employ the new key derivation mechanism. While this can be made to work for specific cases, it would be an extremely brittle mechanism, ripe to result in problems whenever client configuration, routing of authentication requests, or server configuration does not match expectations. It also does not help to assume that the EAP client and server are running a particular release of 3GPP network specifications. Network vendors often provide features from future releases early or do not

provide all features of the current release. And obviously, there are many EAP and even some EAP-AKA implementations that are not bundled with the 3GPP network offerings. In general, these approaches are expected to lead to hard-to-diagnose problems and increased support calls.

Appendix D. Test Vectors

Test vectors are provided below for four different cases. The test vectors may be useful for testing implementations. In the first two cases, we employ the Milenage algorithm and the algorithm configuration parameters (the subscriber key K and operator algorithm variant configuration value OP) from test set 19 in [TS-3GPP.35.208].

The last two cases use artificial values as the output of AKA, and is useful only for testing the computation of values within EAP-AKA', not AKA itself.

Case 1

The parameters for the AKA run are as follows:

Identity: "0555444333222111"

Network name: "WLAN"

RAND: 81e9 2b6c 0ee0 e12e bceb a8d9 2a99 dfa5

AUTN: bb52 e91c 747a c3ab 2a5c 23d1 5ee3 51d5

IK: 9744 871a d32b f9bb d1dd 5ce5 4e3e 2e5a

CK: 5349 fbe0 9864 9f94 8f5d 2e97 3a81 c00f

RES: 28d7 b0f2 a2ec 3de5

Then the derived keys are generated as follows:

CK': 0093 962d 0dd8 4aa5 684b 045c 9edf fa04

IK': ccfc 230c a74f cc96 c0a5 d611 64f5 a76c

K_encr: 766f a0a6 c317 174b 812d 52fb cd11 a179

K_aut: 0842 ea72 2ff6 835b fa20 3249 9fc3 ec23
c2f0 e388 b4f0 7543 ffc6 77f1 696d 71ea

K_re: cf83 aa8b c7e0 aced 892a cc98 e76a 9b20

95b5 58c7 795c 7094 715c b339 3aa7 d17a

MSK: 67c4 2d9a a56c 1b79 e295 e345 9fc3 d187
d42b e0bf 818d 3070 e362 c5e9 67a4 d544
e8ec fe19 358a b303 9aff 03b7 c930 588c
055b abee 58a0 2650 b067 ec4e 9347 c75a

EMSK: f861 703c d775 590e 16c7 679e a387 4ada
8663 11de 2907 64d7 60cf 76df 647e a01c
313f 6992 4bdd 7650 ca9b ac14 1ea0 75c4
ef9e 8029 c0e2 90cd bad5 638b 63bc 23fb

Case 2

The parameters for the AKA run are as follows:

Identity: "0555444333222111"

Network name: "HRPD"

RAND: 81e9 2b6c 0ee0 e12e bceb a8d9 2a99 dfa5

AUTN: bb52 e91c 747a c3ab 2a5c 23d1 5ee3 51d5

IK: 9744 871a d32b f9bb d1dd 5ce5 4e3e 2e5a

CK: 5349 fbe0 9864 9f94 8f5d 2e97 3a81 c00f

RES: 28d7 b0f2 a2ec 3de5

Then the derived keys are generated as follows:

CK': 3820 f027 7fa5 f777 32b1 fb1d 90c1 a0da

IK': db94 a0ab 557e f6c9 ab48 619c a05b 9a9f

K_encr: 05ad 73ac 915f ce89 ac77 e152 0d82 187b

K_aut: 5b4a caef 62c6 ebb8 882b 2f3d 534c 4b35
2773 37a0 0184 f20f f25d 224c 04be 2afd

K_re: 3f90 bf5c 6e5e f325 ff04 eb5e f653 9fa8
cca8 3981 94fb d00b e425 b3f4 0dba 10ac

MSK: 87b3 2157 0117 cd6c 95ab 6c43 6fb5 073f
f15c f855 05d2 bc5b b735 5fc2 1ea8 a757
57e8 f86a 2b13 8002 e057 5291 3bb4 3b82
f868 a961 17e9 1a2d 95f5 2667 7d57 2900

```

EMSK: c891 d5f2 0f14 8a10 0755 3e2d ea55 5c9c
b672 e967 5f4a 66b4 bafa 0273 79f9 3aee
539a 5979 d0a0 042b 9d2a e28b ed3b 17a3
1dc8 ab75 072b 80bd 0c1d a612 466e 402c

```

Case 3

The parameters for the AKA run are as follows:

```

Identity:      "0555444333222111"

Network name:  "WLAN"

RAND:          e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0

AUTN:          a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0

IK:            b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0

CK:            c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0

RES:           d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0

```

Then the derived keys are generated as follows:

```

CK':           cd4c 8e5c 68f5 7dd1 d7d7 dfd0 c538 e577

IK':           3ece 6b70 5dbb f7df c459 a112 80c6 5524

K_encr:        897d 302f a284 7416 488c 28e2 0dcb 7be4

K_aut:         c407 00e7 7224 83ae 3dc7 139e b0b8 8bb5
                58cb 3081 eccd 057f 9207 d128 6ee7 dd53

K_re:          0a59 1a22 dd8b 5b1c f29e 3d50 8c91 dbbd
                b4ae e230 5189 2c42 b6a2 de66 ea50 4473

MSK:           9f7d ca9e 37bb 2202 9ed9 86e7 cd09 d4a7
                0d1a c76d 9553 5c5c ac40 a750 4699 bb89
                61a2 9ef6 f3e9 0f18 3de5 861a d1be dc81
                ce99 1639 1b40 1aa0 06c9 8785 a575 6df7

EMSK:          724d e00b db9e 5681 87be 3fe7 4611 4557
                d501 8779 537e e37f 4d3c 6c73 8cb9 7b9d
                c651 bc19 bfad c344 ffe2 b52c a78b d831
                6b51 dacc 5f2b 1440 cb95 1552 1cc7 ba23

```

Case 4

The parameters for the AKA run are as follows:

Identity: "0555444333222111"
Network name: "HRPD"
RAND: e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0 e0e0
AUTN: a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0 a0a0
IK: b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0 b0b0
CK: c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0 c0c0
RES: d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0 d0d0

Then the derived keys are generated as follows:

CK': 8310 a71c e6f7 5488 9613 da8f 64d5 fb46
IK': 5adf 1436 0ae8 3819 2db2 3f6f cb7f 8c76
K_encr: 745e 7439 ba23 8f50 fcac 4d15 d47c d1d9
K_aut: 3e1d 2aa4 e677 025c fd86 2a4b e183 61a1
3a64 5765 5714 63df 833a 9759 e809 9879
K_re: 99da 835e 2ae8 2462 576f e651 6fad 1f80
2f0f a119 1655 dd0a 273d a96d 04e0 fcd3
MSK: c6d3 a6e0 cee a 951e b20d 74f3 2c30 61d0
680a 04b0 b086 ee87 00ac e3e0 b95f a026
83c2 87be ee44 4322 94ff 98af 26d2 cc78
3bac e75c 4b0a f7fd feb5 511b a8e4 cbd0
EMSK: 7fb5 6813 838a dafa 99d1 40c2 f198 f6da
cebf b6af ee44 4961 1054 02b5 08c7 f363
352c b291 9644 b504 63e6 a693 5415 0147
ae09 cbc5 4b8a 651d 8787 a689 3ed8 536d

Authors' Addresses

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@piuha.net

Vesa Lehtovirta
Ericsson
Jorvas 02420
Finland

Email: vesa.lehtovirta@ericsson.com

Pasi Eronen
Nokia Research Center
P.O. Box 407
FIN-00045 Nokia Group
Finland

Email: pasi.eronen@nokia.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 20, 2018

E. Foudil
September 10, 2017

A Method for Web Security Policies
draft-foudil-securitytxt-00

Abstract

When security risks in web services are discovered by independent security researchers who understand the severity of the risk, they often lack the channels to properly disclose them. As a result, security issues may be left unreported. Security.txt defines a standard to help organizations define the process for security researchers to securely disclose security vulnerabilities.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 20, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction

1.1. Motivation

Many security researchers encounter situations where they are unable to responsibly disclose security issues to companies because there is no course of action laid out. Security.txt is designed to help assist in this process by making it easier for companies to designate the preferred steps for researchers to take when trying to reach out.

1.2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

2. The Specification

Security.txt is a text file located in the website's top-level directory. This text file contains 4 directives with different values. The "directive" is the first part of a field all the way up to the colon ("In-scope:"). Directives are case-insensitive. The "value" comes after the directive ("example.com"). A "field" always consists of a directive and a value ("In-scope: example.com"). A security.txt file can have an unlimited number of fields. It is important to note that you need a separate line for every field. One MUST NOT chain multiple values for a single directive. Everything MUST be in a separate field.

A security.txt file only applies to the application it is located in.

2.1. Comments

Comments can be added using the # symbol:

```
<CODE BEGINS>
# This is a comment.
<CODE ENDS>
```

You MAY use one or more comments as descriptive text immediately before the field. Parsers can then associate the comments with the respective field.

2.2. Separate Fields

A separate line is required for every new value and field. You MUST NOT chain everything in to a single field. Every line must end with a line feed character (%x0A).

2.3. Contact:

Add an address that researchers MAY use for reporting security issues. The value can be an email address, a phone number and/or a security page with more information. The "Contact:" directive MUST always be present in a security.txt file.

```
<CODE BEGINS>
Contact: security@example.com
Contact: +1-201-555-0123
Contact: https://example.com/security
<CODE ENDS>
```

2.4. Encryption:

This directive allows you to add your key for encrypted communication. You MUST NOT directly add your PGP key. The value MUST be a link to a page which contains your key. Keys SHOULD be loaded over HTTPS.

```
<CODE BEGINS>
Encryption: https://example.com/pgp-key.txt
<CODE ENDS>
```

2.5. Disclosure:

Specify your disclosure policy. This directive MUST be a disclosure type. The "Full" value stands for full disclosure, "Partial" for partial disclosure and "None" means you do not want to disclose reports after the issue has been resolved. The presence of a disclosure field is NOT permission to disclose vulnerabilities and explicit permission MUST be sought where possible.

```
<CODE BEGINS>
Disclosure: Full
<CODE ENDS>
```

2.6. Acknowledgement:

This directive allows you to link to a page where security researchers are recognized for their reports.

```
Acknowledgement: https://example.com/hall-of-fame.html
```

2.7. Example

```
<CODE BEGINS>
# Our security address

Contact: security@example.com
Encryption: https://example.com/pgp-key.txt
Disclosure: Full
<CODE ENDS>
```

3. File Format Description

The expected file format of the security.txt file is plain text encoded in UTF-8.

The following is an ABNF definition of the security.txt format, using the conventions defined in [RFC5234].

```
body                = *line (contact-field eol) *line
line                = *1(field / comment) eol
eol                 = *WSP [CR] LF

field               = contact-field /
                     encryption-field /
                     disclosure-field /
                     acknowledgement-field

fs                  = ":"

comment             = "#" *(WSP / VCHAR / %xA0-E007F)

contact-field       = "Contact" fs SP (email / uri / phone)
email               = <Email address as per RFC 5322>
phone               = "+" *1(DIGIT / "-" / "(" / ")" / SP)
uri                 = <URI as per RFC 3986>

encryption-field    = "Encryption" fs SP uri

disclosure-field    = "Disclosure" fs SP disclosure
disclosure          = "Full" / "Partial" / "None"

acknowledgement-field = "Acknowledgement" fs SP uri
```

4. Security Considerations

Companies creating security.txt files will need to take several security-related issues into consideration. These include exposure of sensitive information and attacks where limited access to a server could grant the ability to modify the contents of the security.txt file or affect how it is served.

As stated in Section 2.4, keys specified using the "Encryption:" directive SHOULD be loaded over HTTPS.

5. IANA Considerations

example.com is used in this document following the uses indicated in [RFC2606]

6. Contributors

The editor would like to acknowledge the help provided during the development of this document by the following individuals:

Tom Hudson helped writing the "File Format Description" and wrote several security.txt parsers.

Joel Margolis was a big help when it came to wording this document appropriately.

Jobert Abma for raising issues and concerns that might arise when using certain directives.

Gerben Janssen van Doorn for reviewing this document multiple times.

Justin Calmus was always there to answer questions related to writing this document.

Casey Ellis had several ideas related to security.txt that helped shape security.txt itself.

7. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC2606] Eastlake 3rd, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, DOI 10.17487/RFC2606, June 1999, <<https://www.rfc-editor.org/info/rfc2606>>.

Author's Address

Edwin Foudil
Email: contact@edoverflow.com

Network Working Group
Internet-Draft
Updates: 4880, 6637 (if approved)
Intended status: Standards Track
Expires: March 18, 2018

R. Tse
Ribose
W. Wong
Hang Seng Management College
J. Lloyd
D. Wyatt
E. Borsboom
Ribose
September 14, 2017

OSCCA Extensions For OpenPGP
draft-openpgp-oscca-02

Abstract

This document enables OpenPGP (RFC4880) usage in an compliant manner with OSCCA regulations for use within China.

Specifically, it extends OpenPGP to support the usage of SM2, SM3 and SM4 algorithms, and provides the OSCCA-compliant OpenPGP profile "OSCCA-SM234".

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 18, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 2. Conventions Used in This Document | 4 |
| 2.1. Definitions | 4 |
| 2.2. Basic Operations | 4 |
| 3. SM2 ECC Algorithms | 5 |
| 3.1. SM2 Digital Signature Algorithm | 5 |
| 3.2. SM2 Key Exchange Protocol | 6 |
| 3.3. SM2 Public Key Encryption | 6 |
| 3.4. Recommended SM2 Curve | 7 |
| 3.4.1. Definitions | 7 |
| 3.4.2. Elliptic Curve Formula | 7 |
| 3.4.3. Curve Parameters | 7 |
| 4. SM3 Hash Algorithm | 8 |
| 5. SM4 Symmetric Encryption Algorithm | 9 |
| 6. Supported Algorithms | 9 |
| 6.1. Public Key Algorithms | 9 |
| 6.2. Symmetric Key Algorithms | 9 |
| 6.3. Hash Algorithms | 10 |
| 7. Conversion Primitives | 10 |
| 8. SM2 Key Derivation Function | 10 |
| 8.1. Prerequisites | 11 |
| 8.2. Inputs | 11 |
| 8.3. Outputs | 11 |
| 9. Encoding of Public and Private Keys | 12 |
| 9.1. Public-Key Packet Formats | 12 |
| 9.2. Secret-Key Packet Formats | 13 |
| 10. Message Encoding with Public Keys | 13 |
| 10.1. Public-Key Encrypted Session Key Packets (Tag 1) | 13 |
| 10.2. Signature Packet (Tag 2) | 14 |
| 10.2.1. Version 3 Signature Packet Format | 14 |
| 10.2.2. Version 4 Signature Packet Format | 14 |
| 11. SM2 ECC Curve OID | 14 |
| 12. Compatibility Profiles | 14 |
| 12.1. OSCCA SM234 Profile | 15 |
| 13. Security Considerations | 15 |
| 14. IANA Considerations | 16 |
| 15. Examples | 16 |
| 15.1. Public Key Example | 16 |
| 15.2. Signature Example | 17 |

| | |
|------------------------------|----|
| 16. References | 17 |
| 16.1. Normative References | 17 |
| 16.2. Informative References | 18 |
| Appendix A. Acknowledgements | 23 |
| Authors' Addresses | 23 |

1. Introduction

SM2 [GBT.32918.1-2016] [ISO.IEC.14888-3] [GMT-0003-2012] [SM2] [I-D.shen-sm2-ecdsa], SM3 [GBT.32905-2016] [ISO.IEC.10118-3] [GMT-0004-2012] [SM3] [I-D.shen-sm3-hash] and SM4 [GBT.32907-2016] [ISO.IEC.18033-3.AMD2] [GMT-0002-2012] [SM4] [I-D.ribose-cfrg-sm4] are cryptographic standards issued by the Organization of State Commercial Administration of China [OSCCA] as authorized cryptographic algorithms for use within China. These algorithms are published in public.

Adoption of this document enables exchange of OpenPGP-secured email [RFC4880] in a OSCCA-compliant manner through usage of the authorized combination of SM2, SM3 and SM4.

SM2 is a set of public key cryptographic algorithms based on elliptic curves that include:

- o Digital Signature Algorithm [GBT.32918.2-2016] [ISO.IEC.14888-3] [SM2-2]
- o Key Exchange Protocol [GBT.32918.3-2016] [SM2-3]
- o Public Key Encryption Algorithm [GBT.32918.4-2016] [SM2-4]

SM3 [GBT.32905-2016] [ISO.IEC.10118-3] is a hash algorithm designed for electronic authentication purposes.

SM4 [GBT.32907-2016] [ISO.IEC.18033-3.AMD2] is a symmetric encryption algorithm designed for data encryption.

This document extends OpenPGP [RFC4880] and its ECC extension [RFC6637] to support SM2, SM3 and SM4:

- o support the SM3 hash algorithm for data validation purposes
- o support signatures utilizing the combination of SM3 with other digital signing algorithms, such as RSA, ECDSA and SM2
- o support the SM2 asymmetric encryption algorithm for public key operations

- o support usage of SM2 in combination with supported hash algorithms, such as SHA-256 and SM3
- o support the SM4 symmetric encryption algorithm for data protection purposes
- o defines the OpenPGP profile "OSCCA-SM234" to enable usage of OpenPGP in an OSCCA-compliant manner.

2. Conventions Used in This Document

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].

Compliant applications are a subset of the broader set of OpenPGP applications described in [RFC4880]. Any [RFC2119] keyword within this document applies to compliant applications only.

2.1. Definitions

OSCCA-compliant

All cryptographic algorithms used are compliant with OSCCA [OSCCA] regulations.

SM2DSA

The elliptic curve digital signature algorithm defined in [GBT.32918.2-2016]

SM2KEP

The elliptic curve key exchange protocol defined in [GBT.32918.3-2016]

SM2PKE

The public key encryption algorithm defined in [GBT.32918.4-2016]

2.2. Basic Operations

This document utilizes definitions of operations from [RFC7253] and are included here for reference.

c^i

The integer c raised to the i -th power.

$S || T$

String S concatenated with string T (e.g., $000 || 111 == 000111$).

3. SM2 ECC Algorithms

SM2 is an elliptic curve based cryptosystem (ECC) [GBT.32918.1-2016] [GMT-0003-2012] [SM2] [I-D.shen-sm2-ecdsa] designed by Xiaoyun Wang et al. and published by [OSCCA].

It was first published by the OSCCA in public in 2010 [SM2], then standardized as [GMT-0003-2012] in 2012, included in [ISO.IEC.11889] in 2015, published as a Chinese National Standard as [GBT.32918.1-2016], and published in [ISO.IEC.14888-3] in 2017.

The SM2 cryptosystem is composed of three distinct algorithms:

- o an elliptical curve digital signature algorithm ("SM2DSA") [GBT.32918.2-2016], [ISO.IEC.14888-3], [SM2-2], also described in [I-D.shen-sm2-ecdsa];
- o a key exchange protocol ("SM2KEP") [GBT.32918.3-2016] [SM2-3]; and
- o a public key encryption algorithm ("SM2PKE") [GBT.32918.4-2016] [SM2-4].

This document will refer to all three algorithms for the usage of OpenPGP [RFC4880].

3.1. SM2 Digital Signature Algorithm

The SM2 Digital Signature Algorithm is intended for digital signature and verifications in commercial cryptographic applications, including, but not limited to:

- o identity authentication
- o protection of data integrity
- o verification of data authenticity

The process of digital signature signing and verification along with their examples are found in [GBT.32918.2-2016], [ISO.IEC.14888-3], [SM2-2], and also described in [I-D.shen-sm2-ecdsa].

The SM2DSA process requires usage of a hash function within. For OSCCA-compliant usage, a OSCCA-compliant hash function such as SM3 [GBT.32905-2016] MUST also be used.

Formal security proofs for SM2 are provided in [SM2-SigSecurity] indicating that it satisfies both EUF-CMA security and security against generalized strong key substitution attacks.

The SM2DSA algorithm has been cryptanalyzed by multiple parties with the current strongest attack being nonce [SM2-DSA-Nonces] [SM2-DSA-Nonces2] and lattice attacks [SM2-DSA-Lattice].

In terms of OpenPGP usage, SM2DSA is an alternative to the ECDSA algorithm specified in [RFC6637].

For OpenPGP compatibility, these additional requirements MUST be adhered to:

- o SM2DSA allows use of an optional "user identity" string which is hashed into "ZA" (Section 3.5 of [SM2-2] and Section 5.1.4.4 of [I-D.shen-sm2-ecdsa]). In OpenPGP, the user identifier "IDA" MUST be the empty string.
- o While SM2DSA usually signs "H(ZA || msg)" (Section 4.1 [SM2-2]), but in OpenPGP, following the convention of [RFC6637], we do not directly sign the raw message "msg", but its hash "H(msg)". Therefore when a message is signed by SM2DSA in OpenPGP, the algorithm MUST sign the content of "H(ZA || H(msg))" instead of "H(ZA || msg)". Both hash algorithms used here MUST be identical.

3.2. SM2 Key Exchange Protocol

The SM2 Key Exchange Protocol is used for cryptographic key exchange, allowing the negotiation and exchange of a session key within two to three message transfers.

The process of key exchange and verification along with their examples are found in [GBT.32918.3-2016] [SM2-3], and also described in [I-D.shen-sm2-ecdsa].

SM2KEP is not used with OpenPGP as it is a two- to three- pass key exchange mechanism, while in OpenPGP, public keys of recipients are available initially.

The SM2KEP is now considered insecure due to [SM2-KEP-Comments], similar in status to the Unified Model and MQV schemes described in [NIST.SP.800-56Ar2].

3.3. SM2 Public Key Encryption

The SM2 Public Key Encryption algorithm is an elliptic curve (ECC) based asymmetric encryption algorithm. It is used for cryptographic encryption and decryption, allowing the message sender to utilize the public key of the message receiver to encrypt the message, with the recipient decrypting the messaging using his private key.

The full description of SM2PKE is provided in [GBT.32918.4-2016].

It utilizes a public key size of 512 bits and private key size of 256 bits [GBT.32918.4-2016] [GMT-0003-2012].

The process of encryption and decryption, along with their examples are found in [GBT.32918.4-2016] and [SM2-4].

The SM2PKE process requires usage of a hash function within. For OSCCA-compliant usage, a OSCCA-compliant hash function such as SM3 [GBT.32905-2016] MUST also be used.

In OpenPGP, SM2PKE is an alternative to RSA specified in [RFC4880].

3.4. Recommended SM2 Curve

The recommended curve is specified in [GBT.32918.5-2017] [SM2-5] and provided here for reference. SM2 uses a 256-bit elliptic curve.

3.4.1. Definitions

P
an integer larger than 3

a, b
elements of F_q , defines an elliptic curve E on F_q

n
Order of base point G (n is a prime factor of $E(F_q)$)

x_G
x-coordinate of generator G

y_G
y-coordinate of generator G

3.4.2. Elliptic Curve Formula

$$y^2 = x^3 + ax + b$$

3.4.3. Curve Parameters

```

p   = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
      FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
a   = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
      FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
b   = 28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7
      F39789F5 15AB8F92 DDBCBD41 4D940E93
n   = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
      7203DF6B 21C6052B 53BBF409 39D54123
x_G = 32C4AE2C 1F198119 5F990446 6A39C994
      8FE30BBF F2660BE1 715A4589 334C74C7
y_G = BC3736A2 F4F6779C 59BDCEE3 6B692153
      D0A9877C C62A4740 02DF32E5 2139F0A0

```

4. SM3 Hash Algorithm

The SM3 Cryptographic Hash Algorithm [GBT.32905-2016] is an iterative hash function designed by Xiaoyun Wang et al., published by [OSCCA] as an alternative to SHA-2 [NIST.FIPS.180-4].

It was first published by the OSCCA in public in 2010 [SM3], then published in the OSCCA standard [GMT-0004-2012] in 2012, published as a Chinese National Standard as [GBT.32905-2016] in 2016, and included in the [ISO.IEC.10118-3] standard in 2017.

The algorithm is designed to be used for commercial cryptographic applications including, but not limited to:

- o digital signatures and their verification
- o message authentication code generation and their verification
- o generation of random numbers

SM3 has a Merkle-Damgard construction and is similar to SHA-2 [NIST.FIPS.180-4] of the MD4 [RFC6150] family, with the addition of several strengthening features including a more complex step function and stronger message dependency than SHA-256 [SM3-Boomerang].

SM3 produces an output hash value of 256 bits long, based on 512-bit input message blocks [SM3-Boomerang], on input lengths up to 2^m .

The specification of SM3 is described in [GBT.32905-2016], [SM3] and [I-D.shen-sm3-hash].

5. SM4 Symmetric Encryption Algorithm

SM4 [GBT.32907-2016] [I-D.ribose-cfrg-sm4] [ISO.IEC.18033-3.AMD2] [GMT-0002-2012] [SM4] is a symmetric encryption algorithm designed by Shuwang Lu et al. originally intended for the usage of wireless local area network (Wireless LAN) products.

SM4 is a 128-bit blockcipher, uses a key size of 128 bits and internally uses an 8-bit S-box. It performs 32 rounds per block. Decryption is achieved by reversing the order of encryption.

SMS4 was first published in public as part of WAPI (Wired Authentication and Privacy Infrastructure), the Chinese National Standard for Wireless LAN [GB.15629.11-2003]. It was then published independently by the OSCCA in 2006 [SM4], formally renamed to SM4 in 2012 [GMT-0002-2012], published as a Chinese National Standard in 2016 [GBT.32907-2016], and included in [ISO.IEC.18033-3.AMD2] in 2017.

It is a required encryption algorithm specified in WAPI [GB.15629.11-2003].

6. Supported Algorithms

6.1. Public Key Algorithms

The SM2 algorithm is supported with the following extension.

The following public key algorithm IDs are added to expand Section 9.1 of [RFC4880], "Public-Key Algorithms":

| ID | Description of Algorithm |
|-----|--------------------------|
| TBD | SM2 |

Compliant applications MUST support both usages of SM2 Section 3:

- o SM2 Digital Signature Algorithm (SM2DSA) [GBT.32918.2-2016]
- o SM2 Public Key Encryption (SM2PKE) [GBT.32918.4-2016]

6.2. Symmetric Key Algorithms

The SM4 algorithm is supported with the following extension.

The following symmetric encryption algorithm ID is added to expand Section 9.2 of [RFC4880], "Symmetric-Key Algorithms":

| ID | Description of Algorithm |
|-----|--------------------------|
| TBD | SM4 |

Compliant applications MUST support SM4 Section 5.

6.3. Hash Algorithms

The SM3 algorithm is supported with the following extension.

The following symmetric encryption algorithm IDs are added to expand Section 9.3 of [RFC4880], "Hash Algorithms":

| ID | Description of Algorithm |
|-----|--------------------------|
| TBD | SM3 |

Compliant applications MUST support SM3 Section 4.

7. Conversion Primitives

The encoding method of [RFC6637] Section 6 MUST be used, and is compatible with the definition given in [SEC1].

For clarity, according to the EC curve MPI encoding method of [RFC6637], the exact size of the MPI payload for the "SM2 Recommended" 256-bit curve [GBT.32918.5-2017], is 515 bits.

8. SM2 Key Derivation Function

A key derivation function (KDF) is necessary to implement EC encryption.

The SM2PKE KDF is defined in Section 3.4.3 of [GBT.32918.4-2016] (and Section 5.4.3 of [I-D.shen-sm2-ecdsa], Section 3.4.3 of [SM2-4]).

For OSCCA-compliance, it SHOULD be used in conjunction with an OSCCA-approved hash algorithm, such as SM3 [GBT.32905-2016].

The SM2PKE KDF is equivalent to the KDF2 function defined in Section 13.2 of [IEEE.1363a.2004] given the following assignments:

- o Parameter
 - * v as $hBits$, the output length of the selected hash function Hash
- o Input
 - * $KEYLEN$ as $oBits$
 - * Z as the plaintext string; and
 - * PB is set to the empty bit string.

Pseudocode of the SM2KDF function is provided here for convenience. This function contains edited variable names for clarity.

8.1. Prerequisites

- o $Hash(S)$ is a hash function that outputs a v -bit long hash value based on input S .
- o $MSB(b, S)$ is a function that outputs the b most significant bits of the bitstream S .
- o $Floor(r)$ and $Ceil(r)$ are the floor and ceiling functions respectively for the input of real number r . Both functions outputs an integer.

8.2. Inputs

$KEYLEN$
Desired key length. A positive integer less than $(2^{32} - 1) \times v$.

Z
Plaintext. String of any length.

8.3. Outputs

K
Generated key. String of length $KEYLEN$.

K is defined as follows.

```

Counter = 1                                // a 32-bit counter
n = KEYLEN / v

for each 1 <= i <= Ceil(n)
  Ha_i = Hash( Z || Counter )
  Counter = Counter + 1
end for

if n is a whole number then
  Ha! = Ha_{Ceil(n)}
else
  Ha! = MSB(KEYLEN - (v x Floor(n)), Ha_{Ceil(n)})
end if

K = Ha_1 || Ha_2 || ... || Ha_{Ceil(n)-1} || Ha!

```

9. Encoding of Public and Private Keys

9.1. Public-Key Packet Formats

The following algorithm-specific packets are added to Section 5.5.2 of [RFC4880], "Public-Key Packet Formats", to support SM2DSA and SM2PKE.

This document extends the algorithm-specific portion with the following fields.

Algorithm-Specific Fields for SM2DSA keys:

- o a variable-length field containing a curve OID, formatted as follows:
 - * a one-octet size of the following field; values 0 and 0xFF are reserved for future extensions
 - * octets representing a curve OID, described in Section 11
- o MPI of an EC point representing a public key

Algorithm-Specific Fields for SM2PKE keys:

- o a variable-length field containing a curve OID, formatted as follows:
 - * a one-octet size of the following field; values 0 and 0xFF are reserved for future extensions
 - * octets representing a curve OID, described in Section 11

- o MPI of an EC point representing a public key

Note that both SM2DSA and SM2PKE public keys are composed of the same sequence of fields, and use the same codepoint to identify them. They are distinguished by the key usage flags.

9.2. Secret-Key Packet Formats

The following algorithm-specific packets are added to Section 5.5.3. of [RFC4880], "Secret-Key Packet Formats", to support SM2DSA and SM2PKE.

This document extends the algorithm-specific portion with the following fields.

Algorithm-Specific Fields for SM2DSA or SM2PKE secret keys:

- o an MPI of an integer representing the secret key, which is a scalar of the public EC point

10. Message Encoding with Public Keys

10.1. Public-Key Encrypted Session Key Packets (Tag 1)

Section 5.1 of [RFC4880], "Public-Key Encrypted Session Key Packets (Tag 1)" is extended to support SM2PKE using the following algorithm specific fields for SM2PKE, through applying the KDF described in Section 8.

Algorithm Specific Fields for SM2 encryption:

- o The SM2 ciphertext is formatted in the OpenPGP bitstream as a single MPI. This consists of:
 - * "C = (C1 || C3 || C2)" (step A8 of Section 4.1 [SM2-4]), followed by
 - * a single octet giving the code for the hash algorithm used within the calculation of the KDF mask "t" (step A5 of Section 4.1 [SM2-4]) and the calculation of "C3" (step A7 of Section 4.1 [SM2-4]). For OSCCA compliance, this MUST be an OSCCA-approved hash function, and in any case, it SHOULD be a hash which is listed in the receiving keys "Preferred Hash Algorithms" list (Section 5.2.3.8 of [RFC4880]).

10.2. Signature Packet (Tag 2)

10.2.1. Version 3 Signature Packet Format

Section 5.2.2 of [RFC4880] defines the signature format for "Version 3 Signature Packet Format". Similar to ECDSA [RFC6637], no change in the format is necessary for SM2DSA.

10.2.2. Version 4 Signature Packet Format

Section 5.2.3 of [RFC4880] defines the signature format for "Version 4 Signature Packet Format". Similar to ECDSA [RFC6637], no change in the format is necessary for SM2DSA.

11. SM2 ECC Curve OID

This section provides the curve OID of the "SM2 Recommended Curve" [GBT.32918.5-2017] described in Section 3, according to the method of [RFC6637].

We specify the curve OID of the "SM2 Recommended Curve" to be the registered OID entry of "SM2 Elliptic Curve Cryptography" according to [GMT-0006-2012], which is "1.2.156.10197.1.301".

The table below specifies the exact sequence of bytes of the mentioned curve:

| ASN.1 Object Identifier | OID len | Curve OID bytes in hexadecimal representation | Curve name |
|-------------------------|---------|---|-----------------|
| 1.2.156.10197.1.301 | 8 | 2A 81 1C CF 55 01 82 2D | SM2 Recommended |

The complete ASN.1 DER encoding for the SM2 Recommended curve OID is "06 08 2A 81 1C CF 55 01 82 2D", from which the first entry in the table above is constructed by omitting the first two octets. Only the truncated sequence of octets is the valid representation of a curve OID.

12. Compatibility Profiles

12.1. OSCCA SM234 Profile

The "OSCCA SM234" profile is designed to be compliant to OSCCA regulations. A compliant OpenPGP implementation MUST implement the following items as described by this document:

- o SM2 Recommended Curve (Section 11)
- o SM2 (SM2DSA and SM2PKE) (Section 3)
 - * The hash function selected in SM2DSA and SM2PKE MUST also be OSCCA-compliant, such as SM3 [SM3]
- o SM3 (Section 4)
- o SM4 (Section 5)

13. Security Considerations

- o Products and services that utilize cryptography are regulated by the OSCCA [OSCCA]; they must be explicitly approved or certified by the OSCCA before being allowed to be sold or used in China.
- o SM2 [GBT.32918.1-2016] is an elliptic curve cryptosystem (ECC) published by the OSCCA [OSCCA]. Its security relies on the assumption that the elliptic curve discrete logarithm problem (ECLP) is computationally infeasible. With advances in cryptanalysis, new attack algorithms may reduce the complexity of ECLP, making it easier to attack the SM2 cryptosystem that is considered secure at the time this document is published. You SHOULD check current literature to determine if the algorithms in SM2 have been found vulnerable.
- o SM3 [GBT.32905-2016] is a cryptographic hash algorithm published by the OSCCA [OSCCA]. No formal proof of security is provided. As claimed in [I-D.shen-sm3-hash], the security properties of SM3 are under public study. There are no known feasible attacks against the SM3 algorithm at the time this document is published.
- o SM4 [GBT.32907-2016] is a blockcipher certified by the OSCCA [OSCCA]. No formal proof of security is provided. There are no known feasible attacks against the SM4 algorithm by the time of publishing this document. On the other hand, there are security concerns with regards to side-channel attacks, when the SM4 algorithm is implemented in a device [SM4-Power]. For instance, [SM4-Power] illustrated an attack by measuring the power consumption of the device. A chosen ciphertext attack, assuming a fixed correlation between the sub-keys and data mask, is able to

recover the round key successfully. When the SM4 algorithm is implemented in hardware, the parameters/keys SHOULD be randomly generated without fixed correlation.

- o SM2 has a key length of 512 bits for the public key and 256 bits for the private key. It is considered an alternative to ECDSA P-256 [RFC6637]. Its security strength is comparable to a 128-bit symmetric key strength [I-D.ietf-msec-mikey-ecc], e.g., AES-128 [NIST.FIPS.197].
- o SM3 is a hash function that generates a 256-bit hash value. It is considered as an alternative to SHA-256 [RFC6234].
- o SM4 is a blockcipher symmetric algorithm with a key length of 128 bits. It is considered as an alternative to AES-128 [NIST.FIPS.197].
- o Security considerations offered in [RFC6637] and [RFC4880] also apply.

14. IANA Considerations

The IANA "Pretty Good Privacy (PGP)" registry [RFC8126] has made the following assignments for algorithms described in this document, namely:

- o ID XXX of the "Public Key Algorithms" namespace for SM2 Section 3
- o ID XXX of the "Hash Algorithms" namespace for SM3 Section 4
- o ID XXX of the "Symmetric Key Algorithms" namespace for SM4 Section 5

15. Examples

15.1. Public Key Example

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
xliEWbGKWmMIKoEcZlUBgi0CAwQx5lUJNwGp01AB7YfAye0oMmyIPYe/cQPVwh8/7RCu
ywZLMDDAM7qn6TNqTtdKW+7tLFhtOC4yzDVK8UjN/ccazSBTTTIgMjU2LWJpdCBzZXkg
PGphyY2tAbG9jYWxob3N0PsJ0BBNjaQAmBQJZsYpfAhsDBQsJCACCBhUICQoLAgUWAgMB
AAkQC/UcNw0bAZcAAJt5AP4oXvi3xl2RUwAvVjlzXtLL87g6x9cIBS7EB/cvAsw78AEA
/Wt6qWlBVZ6TYiqNPt9An/4cjKyNpAv7S9u3neGXWUU=
=RJ3C
-----END PGP PUBLIC KEY BLOCK-----
```

15.2. Signature Example

Detached signature of the string "SM2 example" using the above key:

```
-----BEGIN PGP SIGNATURE-----  
wmQEAGMIABYFAlmxj+cFAwAAAAAJEAvlHDcNGwGXAAB+SQEAY5AHKgiRxgOogB/2sfge  
JaVoLgpxvDp9yIcaLfP++xkBAPGuZ1f9FjxVd5jlCGd1jFzAPpt8N2Lc3FQDqVjgJvV9  
=Xbbj  
-----END PGP SIGNATURE-----
```

16. References

16.1. Normative References

[GBT.32905-2016]

Standardization Administration of the People's Republic of China, "GB/T 32905-2016 Information Security Techniques -- SM3 Cryptographic Hash Algorithm", August 2016, <<http://www.gb688.cn/bzgk/gb/newGbInfo?hcno=45B1A67F20F3BF339211C391E9278F5E>>.

[GBT.32907-2016]

Standardization Administration of the People's Republic of China, "GB/T 32907-2016 Information Security Technology -- SM4 Block Cipher Algorithm", August 2016, <<http://www.gb688.cn/bzgk/gb/newGbInfo?hcno=7803DE42D3BC5E80B0C3E5D8E873D56A>>.

[GBT.32918.1-2016]

Standardization Administration of the People's Republic of China, "GB/T 32918.1-2016 Information Security Technology -- Public Key Cryptographic Algorithm SM2 Based On Elliptic Curves -- Part 1: General", August 2016, <http://www.sac.gov.cn/was5/web/search?channelid=97779&templet=gjcxjg_detail.jsp&searchword=STANDARD_CODE=%27GB/T%2032918.1-2016%27>.

[GBT.32918.2-2016]

Standardization Administration of the People's Republic of China, "GB/T 32918.2-2016 Information Security Technology -- Public Key Cryptographic Algorithm SM2 Based On Elliptic Curves -- Part 2: Digital Signature Algorithm", August 2016, <http://www.sac.gov.cn/was5/web/search?channelid=97779&templet=gjcxjg_detail.jsp&searchword=STANDARD_CODE=%27GB/T%2032918.2-2016%27>.

[GBT.32918.3-2016]

Standardization Administration of the People's Republic of China, "GB/T 32918.3-2016 Information Security Technology -- Public Key Cryptographic Algorithm SM2 Based On Elliptic Curves -- Part 3: Key Exchange", August 2016, <http://www.sac.gov.cn/was5/web/search?channelid=97779&templet=gjcxjg_detail.jsp&searchword=STANDARD_CODE=%27GB/T%2032918.3-2016%27>.

[GBT.32918.4-2016]

Standardization Administration of the People's Republic of China, "GB/T 32918.4-2016 Information Security Technology -- Public Key Cryptographic Algorithm SM2 Based On Elliptic Curves -- Part 4: Public Key Encryption Algorithm", August 2016, <http://www.sac.gov.cn/was5/web/search?channelid=97779&templet=gjcxjg_detail.jsp&searchword=STANDARD_CODE=%27GB/T%2032918.4-2016%27>.

[GBT.32918.5-2017]

Standardization Administration of the People's Republic of China, "GB/T 32918.5-2017 Information Security Technology -- Public Key Cryptographic Algorithm SM2 Based On Elliptic Curves -- Part 5: Parameter Definition", May 2017, <<http://www.gb688.cn/bzgk/gb/newGbInfo?hcno=728DEA8B8BB32ACFB6EF4BF449BC3077>>.

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

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <<https://www.rfc-editor.org/info/rfc4880>>.

[RFC6637] Jivsov, A., "Elliptic Curve Cryptography (ECC) in OpenPGP", RFC 6637, DOI 10.17487/RFC6637, June 2012, <<https://www.rfc-editor.org/info/rfc6637>>.

16.2. Informative References

[GB.15629.11-2003]

Standardization Administration of the People's Republic of China, "Information technology -- Telecommunications and information exchange between systems -- Local and metropolitan area networks -- Specific requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", May 2003, <<http://www.gb688.cn/bzgk/gb/newGbInfo?hcno=74B9DD11287E72408C19C4D3A360D1BD>>.

[GMT-0002-2012]

Organization of State Commercial Administration of China, "GM/T 0002-2012: SM4 Block Cipher Algorithm", March 2012, <http://www.oscca.gov.cn/Column/Column_32.htm>.

[GMT-0003-2012]

Organization of State Commercial Administration of China, "GM/T 0003-2012: Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves", March 2012, <http://www.oscca.gov.cn/Column/Column_32.htm>.

[GMT-0004-2012]

Organization of State Commercial Administration of China, "GM/T 0004-2012: SM3 Hash Algorithm", March 2012, <http://www.oscca.gov.cn/Column/Column_32.htm>.

[GMT-0006-2012]

Organization of State Commercial Administration of China, "GM/T 0006-2012: Cryptographic Application Identifier Criterion Specification", March 2012, <http://www.oscca.gov.cn/Column/Column_32.htm>.

[I-D.ietf-msec-mikey-ecc]

Milne, A., "ECC Algorithms for MIKEY", draft-ietf-msec-mikey-ecc-03 (work in progress), June 2007.

[I-D.ribose-cfrg-sm4]

Tse, R. and W. Wong, "The SM4 Block Cipher Algorithm And Its Modes Of Operations", draft-ribose-cfrg-sm4-00 (work in progress), September 2017.

[I-D.shen-sm2-ecdsa]

Shen, S., Shen, S., and X. Lee, "SM2 Digital Signature Algorithm", draft-shen-sm2-ecdsa-02 (work in progress), February 2014.

[I-D.shen-sm3-hash]

Shen, S. and S. Shen, "SM3 Hash function", draft-shen-sm3-hash-01 (work in progress), February 2014.

[IEEE.1363a.2004]

Institute of Electrical and Electronics Engineers, "IEEE Std 1363a-2004: IEEE Standard Specifications for Public-Key Cryptography -- Amendment 1: Additional Techniques", September 2004, <<http://grouper.ieee.org/groups/1363/>>.

[ISO.IEC.10118-3]

International Organization for Standardization, "ISO/IEC FDIS 10118-3 -- Information technology -- Security techniques -- Hash-functions -- Part 3: Dedicated hash-functions", June 2017, <<https://www.iso.org/standard/67116.html>>.

[ISO.IEC.11889]

International Organization for Standardization, "ISO/IEC 11889-1:2015 -- Information technology -- Trusted platform module library", August 2015, <<https://www.iso.org/standard/66510.html>>.

[ISO.IEC.14888-3]

International Organization for Standardization, "ISO/IEC 14888-3:2016-03 -- Information technology -- Security techniques -- Digital signatures with appendix -- Part 3: Discrete logarithm based mechanisms", September 2017, <<https://www.iso.org/standard/70631.html>>.

[ISO.IEC.18033-3.AMD2]

International Organization for Standardization, "ISO/IEC WD1 18033-3/AMD2 -- Information technology -- Security techniques -- Encryption algorithms -- Part 3: Block ciphers -- Amendment 2", June 2017, <<https://www.iso.org/standard/54531.html>>.

[NIST.FIPS.180-4]

National Institute of Standards and Technology, "FIPS 180-4 Secure Hash Standard (SHS)", August 2015, <<http://dx.doi.org/10.6028/NIST.FIPS.180-4>>.

[NIST.FIPS.197]

National Institute of Standards and Technology, "FIPS 197 Advanced Encryption Standard (AES)", November 2001, <<https://doi.org/10.6028/NIST.FIPS.197>>.

- [NIST.SP.800-56Ar2] Barker, B., Chen, L., Roginsky, A., and M. Smid, "SP 800-56Ar2 Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", May 2013, <<http://dx.doi.org/10.6028/NIST.SP.800-56Ar2>>.
- [OSCCA] Organization of State Commercial Administration of China, "Organization of State Commercial Administration of China", May 2017, <<http://www.oscca.gov.cn>>.
- [RFC6150] Turner, S. and L. Chen, "MD4 to Historic Status", RFC 6150, DOI 10.17487/RFC6150, March 2011, <<https://www.rfc-editor.org/info/rfc6150>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC7253] Krovetz, T. and P. Rogaway, "The OCB Authenticated-Encryption Algorithm", RFC 7253, DOI 10.17487/RFC7253, May 2014, <<https://www.rfc-editor.org/info/rfc7253>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [SEC1] Standards for Efficient Cryptography Group, "SEC 1: Elliptic Curve Cryptography", September 2010, <<http://www.secg.org/SEC1-Ver-1.0.pdf>>.
- [SM2] Organization of State Commercial Administration of China, "Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves", December 2010, <<http://www.oscca.gov.cn/UpFile/2010122214822692.pdf>>.
- [SM2-2] Organization of State Commercial Administration of China, "Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves -- Part 2: Digital Signature Algorithm", December 2010, <<http://www.oscca.gov.cn/UpFile/2010122214822692.pdf>>.
- [SM2-3] Organization of State Commercial Administration of China, "Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves -- Part 3: Key Exchange Protocol", December 2010, <<http://www.oscca.gov.cn/UpFile/2010122214822692.pdf>>.

- [SM2-4] Organization of State Commercial Administration of China, "Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves -- Part 4: Public Key Encryption Algorithm", December 2010, <<http://www.oscca.gov.cn/UpFile/2010122214822692.pdf>>.
- [SM2-5] Organization of State Commercial Administration of China, "Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves -- Part 5: Parameter definitions", December 2010, <<http://www.oscca.gov.cn/UpFile/2010122214836668.pdf>>.
- [SM2-DSA-Lattice] Cao, W., Feng, J., Zhu, S., Chen, H., Wu, W., Han, X., and X. Zheng, "Practical Lattice-Based Fault Attack and Countermeasure on SM2 Signature Algorithm", November 2016, <https://doi.org/10.1007/978-3-319-29814-6_6>.
- [SM2-DSA-Nonces] Liu, M., Chen, J., and H. Li, "Partially Known Nonces and Fault Injection Attacks on SM2 Signature Algorithm", November 2013, <https://dx.doi.org/10.1007/978-3-319-12087-4_22>.
- [SM2-DSA-Nonces2] Chen, J., Liu, M., Shi, H., and H. Li, "Mind Your Nonces Moving: Template-Based Partially-Sharing Nonces Attack on SM2 Digital Signature Algorithm", November 2015, <<https://doi.acm.org/10.1145/2714576.2714587>>.
- [SM2-KEP-Comments] Xu, X. and D. Feng, "Comments on the SM2 Key Exchange Protocol", December 2011, <https://dx.doi.org/10.1007/978-3-642-25513-7_12>.
- [SM2-SigSecurity] Zhang, Z., Yang, K., Zhang, J., and C. Chen, "Security of the SM2 Signature Scheme Against Generalized Key Substitution Attacks", December 2015, <https://link.springer.com/chapter/10.1007/978-3-319-27152-1_7>.
- [SM3] Organization of State Commercial Administration of China, "SM3 Cryptographic Hash Algorithm", December 2010, <<http://www.oscca.gov.cn/UpFile/20101222141857786.pdf>>.

[SM3-Boomerang]

Bai, D., Yu, H., Wang, G., and X. Wang, "Improved Boomerang Attacks on Round-Reduced SM3 and Keyed Permutation of BLAKE-256", April 2015, <<https://doi.org/10.1049/iet-ifs.2013.0380>>.

[SM4]

Organization of State Commercial Administration of China, "SM4 block cipher algorithm", December 2010, <<http://www.oscca.gov.cn/UpFile/200621016423197990.pdf>>.

[SM4-Power]

Du, Z., Wu, Z., Wang, M., and J. Rao, "Improved chosen-plaintext power analysis attack against SM4 at the round-output", October 2015, <<http://dx.doi.org/10.6028/NIST.FIPS.180-4>>.

Appendix A. Acknowledgements

The authors would like to thank the following persons for their valuable advice and input.

- o The Ribose RNP team for their input and implementation

Authors' Addresses

Ronald Henry Tse
Ribose
Suite 1111, 1 Pedder Street
Central, Hong Kong
Hong Kong

Email: ronald.tse@ribose.com
URI: <https://www.ribose.com>

Dr. Wai Kit Wong
Hang Seng Management College
Hang Shin Link, Siu Lek Yuen
Shatin, New Territories
Hong Kong

Email: wongwk@hsmc.edu.hk
URI: <https://www.hsmc.edu.hk>

Jack E. Lloyd
Ribose
United States of America

Email: jack@randombit.net
URI: <https://www.ribose.com>

Daniel Elliot Wyatt
Ribose
608 W Cork St, Apt 2
Winchester, VA
United States of America

Email: daniel.wyatt@ribose.com
URI: <https://www.ribose.com>

Erick Borsboom
Ribose
Suite 1111, 1 Pedder Street
Central, Hong Kong
Hong Kong

Email: erick.borsboom@ribose.com
URI: <https://www.ribose.com>

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 17, 2018

M. Pala
CableLabs
November 13, 2017

OCSP over DNS (ODIN)
draft-pala-odin-03

Abstract

With the increase number of protocols and applications that rely on digital certificates to authenticate either the communication channel (TLS) or the data itself (PKIX), the need for providing an efficient revocation system is paramount. Although the Online Certificate Status Protocol (OCSP) allows for efficient lookup of the revocation status of a certificate, the distribution of this information via HTTP (or very rarely) HTTPS is not particularly efficient for high volume websites without incurring in high distribution costs (e.g., CDN).

In particular, this specification defines how to distribute OCSP responses over DNS and how to define OCSP-over-DNS URLs in certificates. The use of the DNS system to distribute such information is meant to lower the costs of providing revocation services (by leveraging the distributed nature of DNS cache) and increase the availability of revocation information (by providing an additional access method for revocation information retrieval).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 17, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|---|
| 1. Requirements notation | 2 |
| 2. Introduction | 2 |
| 3. Overview of existing solutions | 3 |
| 4. Scope Statement | 3 |
| 5. Protocol Overview | 3 |
| 6. The OCSP Resource Record (OCSPRR) | 4 |
| 6.1. The OCSP RDATA Wire Format | 4 |
| 6.2. The OCSP RRType | 4 |
| 6.3. Time Validity | 5 |
| 7. Specifying DNS URLs for OCSP RR | 5 |
| 7.1. URL definition | 5 |
| 7.2. DNS URL Processing | 6 |
| 7.3. OCSPRR URI Examples | 6 |
| 8. IANA Considerations | 6 |
| 9. Security Considerations | 7 |
| 10. Acknowledgments | 7 |
| 11. Normative References | 8 |
| Author's Address | 8 |

1. Requirements notation

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. Introduction

With the increasing number of highly available and highly utilized websites that require secure communications to protect the flow of information from the server to the client and the raising number of devices (IoT) that require strong authentication capabilities, the

need for a low-cost efficient approach to revocation information availability is crucial. The OCSP-over-DNS approach allows clients to determine the revocation status of digital certificates by optimizing the delivery mechanism for revocation information distribution to the client. This transport protocol can be used in lieu of or in addition to other PKIX endorsed transport mechanisms such as HTTP. This specification addresses the problem of providing a highly-available distributed system for OCSP responses [RFC6960].

This document defines the DNS records to be used for OCSP data publication and the definition of additional URLs for the AuthorityInfoAccess (AIA) extension in certificates.

3. Overview of existing solutions

Currently there are three main options to retrieve the revocation information associated with a digital certificates:

- o by retrieving the freshest CRL
- o by querying an OCSP responder for a freshly computed response
- o by retrieving a pre-signed OCSP response from a web site (typically a content distribution network or CDN)
- o by verifying pre-computed OCSP responses embedded (stapled) during the TLS negotiation (only in the TLS case, though)

All of these methods are based on the ability from the application to extract URLs out of the CRL (CrlDistributionPoint) or of the OCSP responder (AuthorityInfoAccess) from the certificate and query (almost uniquely via HTTP/HTTPS, although supported protocols might include LDAP and FTP) the corresponding server to retrieve the required data.

4. Scope Statement

This document focuses only on the definition of the required options for providing OCSP responses over DNS as an alternative transport protocol. The reliability and accessibility of DNS records (e.g., issues related to TCP vs. UDP DNS responses) are out of the scope of this document.

5. Protocol Overview

In order to validate a certificate using OCSP-over-DNS, the client should check the certificate for a DNS-based OCSP URI ("dns://") and then retrieve the OCSP response from the DNS. After this point, all

procedures are to be performed according to the OCSP protocol as defined in [RFC5019]. In particular, clients using OCSP-over-DNS, SHOULD:

1. Lookup the OCSP URI provided in the AIA of the certificate to be checked. The format of the URI comprises the id-ad-ocsp identifier and a base URL where the scheme ('`dns://`') is used. The format of the full URI is discussed in Section 7.
 2. Retrieve the DNS record carrying the required OCSP response.
6. The OCSP Resource Record (OCSPRR)

The OCSP DNS resource record (RR) is used to distribute a certificate's revocation status to clients. The contents of the OCSP RR record are described in Section 6.1.

The type value for the OCSP RR type is defined in Section 6.2.

The OCSPP RR is class independent.

The OCSP RR Time to Live (TTL) should not exceed the validity period of the OCSP response that is contained in the record.

6.1. The OCSP RDATA Wire Format

The RDATA for an OCSP RR consists of a single field which carries the DER encoded OCSP response for the identified certificate.

[illegible]

The OCSP response should contain only one response that refers to the certificate which contains that URL. Following this schema, the OCSP DNS URIs within the AIA extension SHOULD be unique for each certificate issued by a single CA.

6.2. The OCSP RRType

This document uses a new DNS RR type, OCSP, whose value (TBD) was allocated by IANA from the Resource Record (RR) TYPEs subregistry of the Domain Name System (DNS) Parameters registry.

6.3. Time Validity

The time validity should reflect the frequency of updates in revocation information (i.e., the TTL should not be set to expire after the OCSP response expiration). In practice, as an operational matter, operators SHOULD ensure that the records are published in a way that the TTL is low enough that they expire from caches before the OCSP response expiration.

7. Specifying DNS URLs for OCSP RR

The Authority Information Access extension, as defined in [RFC5280], provides information about the certificate in which the extension appears. In order to specify the availability of OCSP responses over DNS, Certification Authorities should use the OCSP accessMethod OID (id-ad-ocsp) and use "dns" as the transport.

Please note that, when using this accessMethod, the use of the dnsauthority in the specified URI is discouraged as this might reduce the benefits coming from the caching infrastructure of DNS and, possibly, overload the referred DNS server.

7.1. URL definition

A DNS URL [RFC3986] begins with the protocol prefix "dns" and is defined by the following grammar, following the ABNF notation defined in [RFC5234].

```
dnsurl = scheme COLON SLASH SLASH [target]
      [QUESTION [ TYPE=rr_type ]
      ; target: is the dns entry for
      ; the lookup operation.
      ; rr_type: is the type of record
      ; to be retrieved. If not specified,
      ; the default type is OCSPRR

scheme  = "dns"

SLASH   = %x2F           ; forward slash ("/")
COLON   = %x3A           ; colon (":")
QUESTION = %x3F          ; question mark ("?")
TYPE     = "type"        ; the keyword ("type")
```

Although this specification does not mandate for any specific format for the <target> component of the DNS URL, some examples are provided in Section 7.3 with the intent to illustrate, not define, the format.

7.2. DNS URL Processing

In order to process the OCSP DNS URLs in a certificate, clients have to extract the <target> and, if provided, the <type> of record from the URL. After that, client MUST query for the specified record. When the ``OCSPRR`` record type is used, the returned value MUST contain the DER encoded OCSP response related to the certificate that the client is going to validate.

7.3. OCSPRR URI Examples

When using the issuing CA's DNS sub-domain in the DNS URL, the hex (or decimal) representation of the certificate's serialNumber MAY be used as the hostname of the DNS URL. When combined with the specific sub-domain of the issuing CA this provides a unique entry that can be easily queried. For example, given that the sub-domain of the issuing CA is "cal.example.com", the resulting URL in the issued certificate can be constructed as follows:

```
dns://04A3E45534A1B5.cal.example.com?type=OCSPRR
```

Because the serialNumber of a certificate is guaranteed to be unique within a (single) CA, different Certification Authorities MUST use different sub-domains when using this publication algorithm to avoid collisions across different CAs.

However, in some environments, the serial number that will be used in the certificate to be issued can not be pre-fetched and embedded in the AIA's DNS URL entry. In this case, the use of a monotonically increasing or random integer number can be used instead.

In any case, it is important to notice that since the DNS entry is to be used "AS IS" by the relying party that wants to fetch the OCSP response by using the DNS URL, other techniques (e.g., the use of prefixes for different issuing CAs combined with high-resolution clock entries and small random or monotonic integer suffixes) can be implemented independently by different Certificate Service Providers.

8. IANA Considerations

This document uses a new DNS RR type, OCSPRR, whose value (TBD) MUST be allocated by IANA from the Resource Record (RR) TYPEs subregistry of the Domain Name System (DNS) Parameters registry.

9. Security Considerations

Several security considerations need to be explicitly considered for the system administrators and application developers to understand the weaknesses of the overall architecture. It is important to highlight, however, that the following considerations are inherently derived from the nature of the DNS infrastructure and that deployment of the DNSSEC protocol might provide an efficient protection against them.

By lacking the ability to authenticate the originating server directly, the DNS (not DNSSEC) protocol (both in TCP and UDP mode) is vulnerable to attacks where false responses are provided. Although all the information stored in the OSCP RR is signed, the data returned to the client could potentially be altered (e.g., by providing an empty or old response). This type of attack can lead to the application's inability to retrieve the revocation information, thus this approach is vulnerable to Denial of Service (DoS), Man-in-the-middle (MITM), and Reply Attacks.

As mentioned earlier, the deployment of DNSSEC can help in mitigating the described family of attacks by providing a mean for the client (or its resolver) to verify signatures of the DNS records themselves via the DNS keys. This said, the use of DNS (instead of DNSSEC) is equivalent, from a security considerations point of view, to today's deployment best practices for OSCP where pre-computed responses are delivered by CDNs via HTTP (not HTTPS). Therefore, the provisioning of OSCP responses via DNS does not lower or alter the security considerations that apply to the use of OSCP. Last but not least, because of the availability (in most cases) of independent DNS servers that an application can query, the use of multiple requests to different DNS servers (for the same DNS record) might be implemented as a mitigating measure in case an attack is suspected or detected.

10. Acknowledgments

The authors would like to thank everybody who provided insightful comments and helped in the definition of the deployment considerations. In particular, the authors would like to thank Scott A. Rea for his support. We also would like to thank DigiCert and the initial discussion and support for the initial idea. Last but not least, the authors would like to thank all the people that expressed interest in implementing support for this proposal.

11. 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4501] Josefsson, S., "Domain Name System Uniform Resource Identifiers", RFC 4501, DOI 10.17487/RFC4501, May 2006, <<https://www.rfc-editor.org/info/rfc4501>>.
- [RFC5019] Deacon, A. and R. Hurst, "The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments", RFC 5019, DOI 10.17487/RFC5019, September 2007, <<https://www.rfc-editor.org/info/rfc5019>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [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, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.

Author's Address

Massimiliano Pala
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: m.pala@cablelabs.com
URI: <http://www.linkedin.com/in/mpala>

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: March 19, 2018

Y. Sheffer
Intuit
D. Migault
Ericsson
September 15, 2017

TLS Server Identity Pinning with Tickets
draft-sheffer-tls-pinning-ticket-05

Abstract

Misissued public-key certificates can prevent TLS clients from appropriately authenticating the TLS server. Several alternatives have been proposed to detect this situation and prevent a client from establishing a TLS session with a TLS end point authenticated with an illegitimate public-key certificate, but none is currently in wide use.

This document proposes to extend TLS with opaque pinning tickets as a way to pin the server's identity. During an initial TLS session, the server provides an original encrypted pinning ticket. In subsequent TLS session establishment, upon receipt of the pinning ticket, the server proves its ability to decrypt the pinning ticket and thus the ownership of the pinning protection key. The client can now safely conclude that the TLS session is established with the same TLS server as the original TLS session. One of the important properties of this proposal is that no manual management actions are required.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 19, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--|----|
| 1. Introduction | 3 |
| 1.1. Conventions used in this document | 6 |
| 2. Protocol Overview | 6 |
| 2.1. Initial Connection | 6 |
| 2.2. Subsequent Connections | 8 |
| 2.3. Indexing the Pins | 9 |
| 3. Message Definitions | 9 |
| 4. Cryptographic Operations | 10 |
| 4.1. Pinning Secret | 10 |
| 4.2. Pinning Ticket | 10 |
| 4.3. Pinning Protection Key | 11 |
| 4.4. Pinning Proof | 11 |
| 5. Operational Considerations | 12 |
| 5.1. Protection Key Synchronization | 12 |
| 5.2. Ticket Lifetime | 12 |
| 5.3. Certificate Renewal | 13 |
| 5.4. Certificate Revocation | 13 |
| 5.5. Disabling Pinning | 13 |
| 5.6. Server Compromise | 13 |
| 5.7. Disaster Recovery | 14 |
| 6. Previous Work | 14 |
| 6.1. Comparison: HPKP | 14 |
| 6.2. Comparison: TACK | 17 |
| 7. Implementation Status | 17 |
| 7.1. Mint Fork | 18 |
| 7.1.1. Overview | 18 |
| 7.1.2. Description | 18 |
| 7.1.3. Level of Maturity | 18 |
| 7.1.4. Coverage | 18 |
| 7.1.5. Version Compatibility | 18 |
| 7.1.6. Licensing | 19 |

| | |
|---|----|
| 7.1.7. Contact Information | 19 |
| 8. Security Considerations | 19 |
| 8.1. Trust on First Use (TOFU) and MITM Attacks | 19 |
| 8.2. Pervasive Monitoring | 19 |
| 8.3. Server-Side Error Detection | 19 |
| 8.4. Client Policy and SSL Proxies | 20 |
| 8.5. Client-Side Error Behavior | 20 |
| 8.6. Stolen and Forged Tickets | 20 |
| 8.7. Client Privacy | 20 |
| 8.8. Ticket Protection Key Management | 21 |
| 9. IANA Considerations | 21 |
| 10. Acknowledgements | 21 |
| 11. References | 22 |
| 11.1. Normative References | 22 |
| 11.2. Informative References | 22 |
| Appendix A. Document History | 24 |
| A.1. draft-sheffer-tls-pinning-ticket-05 | 24 |
| A.2. draft-sheffer-tls-pinning-ticket-04 | 24 |
| A.3. draft-sheffer-tls-pinning-ticket-03 | 24 |
| A.4. draft-sheffer-tls-pinning-ticket-02 | 24 |
| A.5. draft-sheffer-tls-pinning-ticket-01 | 24 |
| A.6. draft-sheffer-tls-pinning-ticket-00 | 25 |
| Authors' Addresses | 25 |

1. Introduction

The weaknesses of the global PKI system are by now widely known. Essentially, any valid CA may issue a certificate for any organization without the organization's approval (a misissued or "fake" certificate), and use the certificate to impersonate the organization. There are many attempts to resolve these weaknesses, including Certificate Transparency (CT) [RFC6962], HTTP Public Key Pinning (HPKP) [RFC7469], and TACK [I-D.perrin-tls-tack]. CT requires cooperation of a large portion of the hundreds of extant certificate authorities (CAs) before it can be used "for real", in enforcing mode. It is noted that the relevant industry forum (CA/Browser Forum) is indeed pushing for such extensive adoption. TACK has some similarities to the current proposal, but work on it seems to have stalled. Section 6.2 compares our proposal to TACK.

HPKP is an IETF standard, but so far has proven hard to deploy. HPKP pins (fixes) a public key, one of the public keys listed in the certificate chain. As a result, HPKP needs to be coordinated with the certificate management process. Certificate management impacts HPKP and thus increases the probability of HPKP failures. This risk is made even higher given the fact that, even though work has been done at the ACME WG to automate certificate management, in many or even most cases, certificates are still managed manually. As a

result, HPKP cannot be completely automated resulting in error-prone manual configuration. Such errors could prevent the web server from being accessed by some clients. In addition, HPKP uses a HTTP header which makes this solution HTTPS specific and not generic to TLS. On the other hand, the current document provides a solution that is independent of the server's certificate management and that can be entirely and easily automated. Section 6.1 compares HPKP to the current draft in more detail.

The ticket pinning proposal augments these mechanisms with a much easier to implement and deploy solution for server identity pinning, by reusing some of the ideas behind TLS session resumption.

Ticket pinning is a second factor server authentication method and is not proposed as a substitute of the authentication method provided in the TLS key exchange. More specifically, the client only uses the pinning identity method after the TLS key exchange is successfully completed. In other words, the pinning identity method is only performed over an authenticated TLS session. Note that Ticket Pinning does not pin certificate information and as such should be considered a "real" independent second factor authentication.

Ticket pinning is a Trust On First Use (TOFU) mechanism, in that the first server authentication is only based on PKI certificate validation, but for any follow-on sessions, the client is further ensuring the server's identity based on the server's ability to decrypt the ticket, in addition to normal PKI certificate authentication.

During initial TLS session establishment, the client requests a pinning ticket from the server. Upon receiving the request the server generates a pinning secret which is expected to be unpredictable for peers other than the client or the server. In our case, the pinning secret is generated from parameters exchanged during the TLS key exchange, so client and server can generate it locally and independently. The server constructs the pinning ticket with the necessary information to retrieve the pinning secret. The server then encrypts the ticket and returns the pinning ticket to the client with an associated pinning lifetime.

The pinning lifetime value indicates for how long the server promises to retain the server-side ticket-encryption key, which allows it to complete the protocol exchange correctly and prove its identity. The committed lifetime is typically on the order of weeks or months.

Once the key exchange is completed and the server is deemed authenticated, the client generates locally the pinning secret and

caches the server's identifiers to index the pinning secret as well as the pinning ticket and its associated lifetime.

When the client re-establishes a new TLS session with the server, it sends the pinning ticket to the server. Upon receiving it, the server returns a proof of knowledge of the pinning secret. Once the key exchange is completed and the server has been authenticated, the client checks the pinning proof returned by the server using the client's stored pinning secret. If the proof matches, the client can conclude that the server it is currently connecting to is in fact the correct server.

This version of the draft only applies to TLS 1.3. We believe that the idea can also be back-fitted into earlier versions of the protocol.

The main advantages of this protocol over earlier pinning solutions are:

- The protocol is at the TLS level, and as a result is not restricted to HTTP at the application level.
- The protocol is robust to server IP, CA, and public key changes. The server is characterized by the ownership of the pinning protection key, which is never provided to the client. Server configuration parameters such as the CA and the public key may change without affecting the pinning ticket protocol.
- Once a single parameter is configured (the ticket's lifetime), operation is fully automated. The server administrator need not bother with the management of backup certificates or explicit pins.
- For server clusters, we reuse the existing [RFC5077] infrastructure where it exists.
- Pinning errors, presumably resulting from MITM attacks, can be detected both by the client and the server. This allows for server-side detection of MITM attacks using large-scale analytics, and with no need to rely on clients to explicitly report the error.

A note on terminology: unlike other solutions in this space, we do not do "certificate pinning" (or "public key pinning"), since the protocol is oblivious to the server's certificate. We prefer the term "server identity pinning" for this new solution. In our solution, the server proves its identity by generating a proof that it can read and decrypt an encrypted ticket. As a result, the

identity proof relies on proof of ownership of the pinning protection key. However, this key is never exchanged with the client or known by it, and so cannot itself be pinned.

1.1. Conventions used in this document

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. Protocol Overview

The protocol consists of two phases: the first time a particular client connects to a server, and subsequent connections.

This protocol supports full TLS handshakes, as well as 0-RTT handshakes. Below we present it in the context of a full handshake, but behavior in 0-RTT handshakes should be identical.

The document presents some similarities with the ticket resumption mechanism described in [RFC5077]. However the scope of this document differs from session resumption mechanisms implemented with [RFC5077] or with other mechanisms. Specifically, the pinning ticket does not carry any state associated with a TLS session and thus cannot be used for session resumption, or to authenticate the client. Instead, the pinning ticket only contains the Pinning Secret used to generate the proof.

With TLS 1.3, session resumption is based on a preshared key (PSK). This is orthogonal to this protocol. With TLS 1.3, a TLS session can be established using PKI and a pinning ticket, and later resumed with PSK.

However, the protocol described in this document addresses the problem of misissued certificates. Thus, it is not expected to be used outside a certificate-based TLS key exchange, such as in PSK. As a result, PSK handshakes MUST NOT include the extension defined here.

2.1. Initial Connection

When a client first connects to a server, it requests a pinning ticket by sending an empty PinningTicket extension, and receives it as part of the server's first response, in the returned PinningTicket extension.

```

Client                                     Server

ClientHello
+ key_share
+ signature_algorithms*
+ PinningTicket ----->

                                     ServerHello
                                     + key_share
                                     {EncryptedExtensions
                                     + PinningTicket}
                                     {CertificateRequest*}
                                     {Certificate*}
                                     {CertificateVerify*}
                                     {Finished}
                                     <-----
{Certificate*}
{CertificateVerify*}
{Finished} ----->
[Application Data] <-----> [Application Data]

```

* Indicates optional or situation-dependent messages that are not always sent.

{ } Indicates messages protected using keys derived from the ephemeral secret.

[] Indicates messages protected using keys derived from the master secret.

If a client supports the pinning ticket extension and does not have any pinning ticket associated with the server, the exchange is considered as an initial connection. Other reasons the client may not have a pinning ticket include the client having flushed its pinning ticket store, or the committed lifetime of the pinning ticket having expired.

Upon receipt of the PinningTicket extension, the server computes a pinning secret (Section 4.1), and sends the pinning ticket (Section 4.2) encrypted with the pinning protection key (Section 4.3). The pinning ticket is associated with a lifetime value by which the server assumes the responsibility of retaining the pinning protection key and being able to decrypt incoming pinning tickets during the period indicated by the committed lifetime.

Once the pinning ticket has been generated, the server returns the pinning ticket and the committed lifetime in a PinningTicket extension embedded in the EncryptedExtensions message. We note that a PinningTicket extension MUST NOT be sent as part of a HelloRetryRequest.

Upon receiving the pinning ticket, the client MUST NOT accept it until the key exchange is completed and the server authenticated. If the key exchange is not completed successfully, the client MUST ignore the received pinning ticket. Otherwise, the client computes the pinning secret and SHOULD cache the pinning secret and the pinning ticket for the duration indicated by the pinning ticket lifetime. The client SHOULD clean up the cached values at the end of the indicated lifetime.

2.2. Subsequent Connections

When the client initiates a connection to a server it has previously seen (see Section 2.3 on identifying servers), it SHOULD send the pinning ticket for that server. The pinning ticket, pinning secret and pinning ticket lifetime computed during the establishment of the previous TLS session are designated in this document as the "original" ones, to distinguish them from a new ticket that may be generated during the current session.

The server MUST extract the original `pinning_secret` value from the ticket and MUST respond with a `PinningTicket` extension, which includes:

- A proof that the server can understand the ticket that was sent by the client; this proof also binds the pinning ticket to the server's (current) public key, as well as the ongoing TLS session. The proof is MANDATORY if a pinning ticket was sent by the client.
- A fresh pinning ticket. The main reason for refreshing the ticket on each connection is privacy: to avoid the ticket serving as a fixed client identifier. It is RECOMMENDED to include a fresh ticket with each response.

If the server cannot validate the received ticket, that might indicate an earlier MITM attack on this client. The server MUST then abort the connection with a `handshake_failure` alert, and SHOULD log this failure.

The client MUST verify the proof, and if it fails to do so, MUST issue a `handshake_failure` alert and abort the connection (see also Section 8.5). It is important that the client does not attempt to "fall back" by omitting the `PinningTicket` extension.

When the connection is successfully set up, i.e. after the `Finished` message is verified, the client SHOULD store the new ticket along with the corresponding `pinning_secret`, replacing the original ticket.

Although this is an extension, if the client already has a ticket for a server, the client MUST interpret a missing PinningTicket extension in the server's response as an attack, because of the server's prior commitment to respect the ticket. The client MUST abort the connection in this case. See also Section 5.5 on ramping down support for this extension.

2.3. Indexing the Pins

Each pin is associated with a host name, protocol (TLS or DTLS) and port number. In other words, the pin for port TCP/443 may be different from that for DTLS or from the pin for port TCP/8443. The host name MUST be the value sent inside the Server Name Indication (SNI) extension. This definition is similar to a Web Origin [RFC6454], but does not assume the existence of a URL.

The purpose of ticket pinning is to pin the server identity. As a result, any information orthogonal to the server's identity MUST NOT be considered in indexing. More particularly, IP addresses are ephemeral and forbidden in SNI and therefore pins MUST NOT be associated with IP addresses. Similarly, CA names or public keys associated with server MUST NOT be used for indexing as they may change over time.

3. Message Definitions

This section defines the format of the PinningTicket extension. We follow the message notation of [I-D.ietf-tls-tls13].

```
opaque pinning_ticket<0..2^16-1>;

opaque pinning_proof<0..2^8-1>;

struct {
    select (Role) {
        case client:
            pinning_ticket ticket<0..2^16-1>; //omitted on 1st connection

        case server:
            pinning_proof proof<0..2^8-1>; //no proof on 1st connection
            pinning_ticket ticket<0..2^16-1>; //omitted on ramp down
            uint32 lifetime;
    }
} PinningTicketExtension;

ticket:    a pinning ticket sent by the client or returned by the
           server. The ticket is opaque to the client. The extension MUST
           contain exactly 0 or 1 tickets.
```

proof: a demonstration by the server that it understands the received ticket and therefore that it is in possession of the secret that was used to generate it originally. The extension MUST contain exactly 0 or 1 proofs.

lifetime: the duration (in seconds) that the server commits to accept offered tickets in the future.

4. Cryptographic Operations

This section provides details on the cryptographic operations performed by the protocol peers.

4.1. Pinning Secret

The pinning secret is generated locally by the client and the server which means they must use the same inputs to generate it. This value must be generated before the ServerHello message is sent, as the server includes the corresponding pinning ticket in the ServerHello message. In addition, the pinning secret must be unpredictable to any party other than the client and the server.

The pinning secret is derived using the Derive-Secret function provided by TLS 1.3, described in Section "Key Schedule" of [I-D.ietf-tls-tls13].

```
pinning secret = Derive-Secret(Handshake Secret, "pinning secret",
                               ClientHello...ServerHello)
```

4.2. Pinning Ticket

The pinning ticket contains the pinning secret. The pinning ticket is provided by the client to the server which decrypts it in order to extract the pinning secret and responds with a pinning proof. As a result, the characteristics of the pinning ticket are:

- Pinning tickets MUST be encrypted and integrity-protected using strong cryptographic algorithms.
- Pinning tickets MUST be protected with a long-term pinning protection key.
- Pinning tickets MUST include a pinning protection key ID or serial number as to enable the pinning protection key to be refreshed.
- The pinning ticket MAY include other information, in addition to the pinning secret.

The pinning ticket's format is not specified by this document, but we RECOMMEND a format similar to the one proposed by [RFC5077].

4.3. Pinning Protection Key

The pinning protection key is only used by the server and so remains server implementation specific. [RFC5077] recommends the use of two keys, but when using AEAD algorithms only a single key is required.

When a single server terminates TLS for multiple virtual servers using the Server Name Indication (SNI) mechanism, we strongly RECOMMEND to use a separate protection key for each one of them, in order to allow migrating virtual servers between different servers while keeping pinning active.

As noted in Section 5.1, if the server is actually a cluster of machines, the protection key MUST be synchronized between all the nodes that accept TLS connections to the same server name. When [RFC5077] is deployed, an easy way to do it is to derive the protection key from the session-ticket protection key, which is already synchronized. For example:

```
pinning_protection_key = HKDF-Expand(resumption_protection_key,  
                                     "pinning protection", L)
```

4.4. Pinning Proof

The pinning proof is sent by the server to demonstrate that it has been able to decrypt the pinning ticket and retrieve the pinning secret. The proof must be unpredictable and must not be replayed. Similarly to the pinning secret, the pinning proof is sent by the server in the ServerHello message. In addition, it must not be possible for a MITM server with a fake certificate to obtain a pinning proof from the original server.

In order to address these requirements, the pinning proof is bound to the TLS session as well as the public key of the server:

```
proof = HMAC(original_pinning_secret, "pinning proof" +  
          Handshake-Secret + Hash(server_public_key))
```

where HMAC [RFC2104] uses the Hash algorithm that was negotiated in the handshake, and the same hash is also used over the server's public key. The `original_pinning_secret` value refers to the secret value extracted from the ticket sent by the client, to distinguish it from a new pinning secret value that is possibly computed in the current exchange. The `server_public_key` value is the DER

representation of the public key, specifically the SubjectPublicKeyInfo structure as-is.

5. Operational Considerations

The main motivation behind the current protocol is to enable identity pinning without the need for manual operations. To achieve this goal operations described in identity pinning are only performed within the current TLS session, and there is no dependence on any TLS configuration parameters such as CA identity or public keys. As a result, configuration changes are unlikely to lead to desynchronized state between the client and the server. Manual operations are susceptible to human error and in the case of public key pinning, can easily result in "server bricking": the server becoming inaccessible to some or all of its users.

5.1. Protection Key Synchronization

The only operational requirement when deploying this protocol is that if the server is part of a cluster, protection keys (the keys used to encrypt tickets) **MUST** be synchronized between all cluster members. The protocol is designed so that if resumption ticket protection keys [RFC5077] are already synchronized between cluster members, nothing more needs to be done.

Moreover, synchronization does not need to be instantaneous, e.g. protection keys can be distributed a few minutes or hours in advance of their rollover. In such scenarios, each cluster member **MUST** be able to accept tickets protected with a new version of the protection key, even while it is still using an old version to generate keys. This ensures that a client that receives a "new" ticket does not next hit a cluster member that still rejects this ticket.

Misconfiguration can lead to the server's clock being off by a large amount of time. Therefore we **RECOMMEND** never to automatically delete protection keys, even when they are long expired.

5.2. Ticket Lifetime

The lifetime of the ticket is a commitment by the server to retain the ticket's corresponding protection key for this duration, so that the server can prove to the client that it knows the secret embedded in the ticket. For production systems, the lifetime **SHOULD** be between 7 and 31 days.

5.3. Certificate Renewal

The protocol ensures that the client will continue speaking to the correct server even when the server's certificate is renewed. In this sense, we are not "pinning certificates" and the protocol should more precisely be called "server identity pinning".

Note that this property is not impacted by the use of the server's public key in the pinning proof, because the scope of the public key used is only the current TLS session.

5.4. Certificate Revocation

The protocol is orthogonal to certificate validation in the sense that, if the server's certificate has been revoked or is invalid for some other reason, the client MUST refuse to connect to it regardless of any ticket-related behavior.

5.5. Disabling Pinning

A server implementing this protocol MUST have a "ramp down" mode of operation where:

- The server continues to accept valid pinning tickets and responds correctly with a proof.
- The server does not send back a new pinning ticket.

After a while no clients will hold valid tickets any more and the feature may be disabled. Note that clients that do not receive a new pinning ticket do not remove the original ticket. Instead, the client keeps on using the ticket until its lifetime expires.

Issuing a new pinning ticket with a shorter lifetime would only delay the ramp down process, as the shorter lifetime can only affect clients that actually initiated a new connection. Other clients would still see the original lifetime for their pinning tickets.

5.6. Server Compromise

If a server compromise is detected, the pinning protection key MUST be rotated immediately, but the server MUST still accept valid tickets that use the old, compromised key. Clients that still hold old pinning tickets will remain vulnerable to MITM attacks, but those that connect to the correct server will immediately receive new tickets protected with the newly generated pinning protection key.

The same procedure applies if the pinning protection key is compromised directly, e.g. if a backup copy is inadvertently made public.

5.7. Disaster Recovery

All web servers in production need to be backed up, so that they can be recovered if a disaster (including a malicious activity) ever wipes them out. Backup typically includes the certificate and its private key, which must be backed up securely. The pinning secret, including earlier versions that are still being accepted, must be backed up regularly. However since it is only used as an authentication second factor, it does not require the same level of confidentiality as the server's private key.

Readers should note that [RFC5077] session resumption keys are more security sensitive, and should normally not be backed up but rather treated as ephemeral keys. Even when servers derive pinning secrets from resumption keys (Section 4.1), they MUST NOT back up resumption keys.

6. Previous Work

This section compares ticket pinning to two earlier proposals, HPKP and TACK.

6.1. Comparison: HPKP

The current IETF standard for pinning the identity of web servers is the Public Key Pinning Extension for HTTP, or HPKP [RFC7469].

The main differences between HPKP and the current document are the following:

- HPKP limits its scope to HTTPS, while the current document considers all application above TLS.
- HPKP pins the public key of the server (or another public key along the certificate chain) and as such is highly dependent on the management of certificates. Such dependency increases the potential error surface, especially as certificate management is not yet largely automated. The current proposal, on the other hand is independent of certificate management.
- HPKP pins public keys which are public and used for the standard TLS authentication. Identity pinning relies on the ownership of the pinning key which is not disclosed to the public and not involved in the standard TLS authentication. As a result,

identity pinning is a completely independent second factor authentication mechanism.

- HPKP relies on a backup key to recover the mis-issuance of a key. We believe such backup mechanisms add excessive complexity and cost. Reliability of the current mechanism is primarily based on its being highly automated.
- HPKP relies on the client to report errors to the report-uri. The current document does not need any out-of band mechanism, and the server is informed automatically. This provides an easier and more reliable health monitoring.

On the other hand, HPKP shares the following aspects with identity pinning:

- Both mechanisms provide hard failure. With HPKP only the client is aware of the failure, while with the current proposal both client and server are informed of the failure. This provides room for further mechanisms to automatically recover such failures.
- Both mechanisms are subject to a server compromise in which users are provided with an invalid ticket (e.g. a random one) or HTTP Header, with a very long lifetime. For identity pinning, this lifetime cannot be longer than 31 days. In both cases, clients will not be able to reconnect the server during this lifetime. With the current proposal, an attacker needs to compromise the TLS layer, while with HPKP, the attacker needs to compromise the HTTP server. Arguably, the TLS-level compromise is typically more difficult for the attacker.

Unfortunately HPKP has not seen wide deployment yet. As of March 2016, the number of servers using HPKP was less than 3000 [Netcraft]. This may simply be due to inertia, but we believe the main reason is the interactions between HPKP and manual certificate management which is needed to implement HPKP for enterprise servers. The penalty for making mistakes (e.g. being too early or too late to deploy new pins) is having the server become unusable for some of the clients.

To demonstrate this point, we present a list of the steps involved in deploying HPKP on a security-sensitive Web server.

1. Generate two public/private key-pairs on a computer that is not the Live server. The second one is the "backup1" key-pair.

```
"openssl genrsa -out "example.com.key" 2048;"
```

```
"openssl genrsa -out "example.com.backup1.key" 2048;"
```

2. Generate hashes for both of the public keys. These will be used in the HPKP header:

```
"openssl rsa -in "example.com.key" -outform der -pubout |  
openssl dgst -sha256 -binary | openssl enc -base64"
```

```
"openssl rsa -in "example.com.backup1.key" -outform der  
-pubout | openssl dgst -sha256 -binary | openssl enc -base64"
```

3. Generate a single CSR (Certificate Signing Request) for the first key-pair, where you include the domain name in the CN (Common Name) field:

```
"openssl req -new -subj "/C=GB/ST=Area/L=Town/O=Company/  
CN=example.com" -key "example.com.key" -out "example.com.csr";"
```

4. Send this CSR to the CA (Certificate Authority), and go through the dance to prove you own the domain. The CA will give you back a single certificate that will typically expire within a year or two.
5. On the Live server, upload and setup the first key-pair (and its certificate). At this point you can add the "Public-Key-Pins" header, using the two hashes you created in step 2.

Note that only the first key-pair has been uploaded to the server so far.
6. Store the second (backup1) key-pair somewhere safe, probably somewhere encrypted like a password manager. It won't expire, as it's just a key-pair, it just needs to be ready for when you need to get your next certificate.
7. Time passes... probably just under a year (if waiting for a certificate to expire), or maybe sooner if you find that your server has been compromised and you need to replace the key-pair and certificate.
8. Create a new CSR (Certificate Signing Request) using the "backup1" key-pair, and get a new certificate from your CA.
9. Generate a new backup key-pair (backup2), get its hash, and store it in a safe place (again, not on the Live server).
10. Replace your old certificate and old key-pair, and update the "Public-Key-Pins" header to remove the old hash, and add the new "backup2" key-pair.

Note that in the above steps, both the certificate issuance as well as the storage of the backup key pair involve manual steps. Even with an automated CA that runs the ACME protocol, key backup would be a challenge to automate.

6.2. Comparison: TACK

Compared with HPKP, TACK [I-D.perrin-tls-tack] is a lot more similar to the current draft. It can even be argued that this document is a symmetric-cryptography variant of TACK. That said, there are still a few significant differences:

- Probably the most important difference is that with TACK, validation of the server certificate is no longer required, and in fact TACK specifies it as a "MAY" requirement (Sec. 5.3). With ticket pinning, certificate validation by the client remains a MUST requirement, and the ticket acts only as a second factor. If the pinning secret is compromised, the server's security is not immediately at risk.
- Both TACK and the current draft are mostly orthogonal to the server certificate as far as their life cycle, and so both can be deployed with no manual steps.
- TACK uses ECDSA to sign the server's public key. This allows cooperating clients to share server assertions between themselves. This is an optional TACK feature, and one that cannot be done with pinning tickets.
- TACK allows multiple servers to share its public keys. Such sharing is disallowed by the current document.
- TACK does not allow the server to track a particular client, and so has better privacy properties than the current draft.
- TACK has an interesting way to determine the pin's lifetime, setting it to the time period since the pin was first observed, with a hard upper bound of 30 days. The current draft makes the lifetime explicit, which may be more flexible to deploy. For example, Web sites which are only visited rarely by users may opt for a longer period than other sites that expect users to visit on a daily basis.

7. Implementation Status

Note to RFC Editor: please remove this section before publication, including the reference to [RFC7942].

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. Mint Fork

7.1.1. Overview

A fork of the Mint TLS 1.3 implementation, developed by Yaron Sheffer and available at <https://github.com/yaronf/mint>.

7.1.2. Description

This is a fork of the TLS 1.3 implementation, and includes client and server code. In addition to the actual protocol, several utilities are provided allowing to manage pinning protection keys on the server side, and pinning tickets on the client side.

7.1.3. Level of Maturity

This is a prototype.

7.1.4. Coverage

The entire protocol is implemented.

7.1.5. Version Compatibility

The implementation is compatible with draft-sheffer-tls-pinning-ticket-02.

7.1.6. Licensing

Mint itself and this fork are available under an MIT license.

7.1.7. Contact Information

See author details below.

8. Security Considerations

This section reviews several security aspects related to the proposed extension.

8.1. Trust on First Use (TOFU) and MITM Attacks

This protocol is a "trust on first use" protocol. If a client initially connects to the "right" server, it will be protected against MITM attackers for the lifetime of each received ticket. If it connects regularly (depending of course on the server-selected lifetime), it will stay constantly protected against fake certificates.

However if it initially connects to an attacker, subsequent connections to the "right" server will fail. Server operators might want to advise clients on how to remove corrupted pins, once such large scale attacks are detected and remediated.

The protocol is designed so that it is not vulnerable to an active MITM attacker who has real-time access to the original server. The pinning proof includes a hash of the server's public key, to ensure the client that the proof was in fact generated by the server with which it is initiating the connection.

8.2. Pervasive Monitoring

Some organizations, and even some countries perform pervasive monitoring on their constituents [RFC7258]. This often takes the form of always-active SSL proxies. Because of the TOFU property, this protocol does not provide any security in such cases.

8.3. Server-Side Error Detection

Uniquely, this protocol allows the server to detect clients that present incorrect tickets and therefore can be assumed to be victims of a MITM attack. Server operators can use such cases as indications of ongoing attacks, similarly to fake certificate attacks that took place in a few countries in the past.

8.4. Client Policy and SSL Proxies

Like it or not, some clients are normally deployed behind an SSL proxy. Similarly to [RFC7469], it is acceptable to allow pinning to be disabled for some hosts according to local policy. For example, a UA MAY disable pinning for hosts whose validated certificate chain terminates at a user-defined trust anchor, rather than a trust anchor built-in to the UA (or underlying platform). Moreover, a client MAY accept an empty PinningTicket extension from such hosts as a valid response.

8.5. Client-Side Error Behavior

When a client receives a malformed or empty PinningTicket extension from a pinned server, it MUST abort the handshake and MUST NOT retry with no PinningTicket in the request. Doing otherwise would expose the client to trivial fallback attacks, similar to those described in [RFC7507].

This rule can however have negative affects on clients that move from behind SSL proxies into the open Internet and vice versa, if the advice in Section 8.4 is not followed. Therefore, we RECOMMEND that browser and library vendors provide a documented way to remove stored pins.

8.6. Stolen and Forged Tickets

Stealing pinning tickets even in conjunction with other pinning parameters, such as the associated pinning secret, provides no benefit to the attacker since pinning tickets are used to secure the client rather than the server. Similarly, it is useless to forge a ticket for a particular sever.

8.7. Client Privacy

This protocol is designed so that an external attacker cannot correlate between different requests of a single client, provided the client requests and receives a fresh ticket upon each connection.

On the other hand, the server to which the client is connecting can easily track the client. This may be an issue when the client expects to connect to the server (e.g., a mail server) with multiple identities. Implementations SHOULD allow the user to opt out of pinning, either in general or for particular servers.

8.8. Ticket Protection Key Management

While the ticket format is not mandated by this document, we RECOMMEND using authenticated encryption to protect it. Some of the algorithms commonly used for authenticated encryption, e.g. GCM, are highly vulnerable to nonce reuse, and this problem is magnified in a cluster setting. Therefore implementations that choose AES-128-GCM MUST adopt one of these two alternatives:

- Partition the nonce namespace between cluster members and use monotonic counters on each member, e.g. by setting the nonce to the concatenation of the cluster member ID and an incremental counter.
- Generate random nonces but avoid the so-called birthday bound, i.e. never generate more than 2^{64} encrypted tickets for the same ticket pinning protection Key.

An alternative design which has been attributed to Karthik Bhargavan is as follows. Start with a 128-bit master key "K_master" and then for each encryption, generate a 256-bit random nonce and compute:

```
K = HKDF(K_master, Nonce || "key")
N = HKDF(K_master, Nonce || "nonce")
```

And use these values to encrypt the ticket, AES-GCM(K, N, <data>).

9. IANA Considerations

IANA is requested to allocate a TicketPinning extension value in the TLS ExtensionType Registry.

No registries are defined by this document.

10. Acknowledgements

The original idea behind this proposal was published in [Oreo] by Moty Yung, Benny Pinkas and Omer Berkman. The current protocol is but a distant relative of the original Oreo protocol, and any errors are the draft authors' alone.

We would like to thank Dave Garrett, Daniel Kahn Gillmor, Eric Rescorla and Yoav Nir for their comments on this draft. Special thanks to Craig Francis for contributing the HPKP deployment script, and to Ralph Holz for several fruitful discussions.

11. References

11.1. Normative References

- [I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-21 (work in progress), July 2016.
- [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>>.
- [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <<http://www.rfc-editor.org/info/rfc5077>>.

11.2. Informative References

- [I-D.perrin-tls-tack]
Marlinspike, M., "Trust Assertions for Certificate Keys", draft-perrin-tls-tack-02 (work in progress), January 2013.
- [Netcraft]
Mutton, P., "HTTP Public Key Pinning: You're doing it wrong!", March 2016, <<http://news.netcraft.com/archives/2016/03/30/http-public-key-pinning-youre-doing-it-wrong.html>>.
- [Oreo] Berkman, O., Pinkas, B., and M. Yung, "Firm Grip Handshakes: A Tool for Bidirectional Vouching", Cryptology and Network Security, pp. 142-157, 2012.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<http://www.rfc-editor.org/info/rfc2104>>.
- [RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <<http://www.rfc-editor.org/info/rfc6454>>.
- [RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <<http://www.rfc-editor.org/info/rfc6962>>.

- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<http://www.rfc-editor.org/info/rfc7258>>.
- [RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <<http://www.rfc-editor.org/info/rfc7469>>.
- [RFC7507] Moeller, B. and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks", RFC 7507, DOI 10.17487/RFC7507, April 2015, <<http://www.rfc-editor.org/info/rfc7507>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<http://www.rfc-editor.org/info/rfc7942>>.

Appendix A. Document History

A.1. draft-sheffer-tls-pinning-ticket-05

- Multiple comments from Eric Rescorla.

A.2. draft-sheffer-tls-pinning-ticket-04

- Editorial changes.
- Two-phase rotation of protection key.

A.3. draft-sheffer-tls-pinning-ticket-03

- Deleted redundant length fields in the extension's formal definition.
- Modified cryptographic operations to align with the current state of TLS 1.3.
- Numerous textual improvements.

A.4. draft-sheffer-tls-pinning-ticket-02

- Added an Implementation Status section.
- Added lengths into the extension structure.
- Changed the computation of the pinning proof to be more robust.
- Clarified requirements on the length of the pinning_secret.
- Revamped the HPKP section to be more in line with current practices, and added recent statistics on HPKP deployment.

A.5. draft-sheffer-tls-pinning-ticket-01

- Corrected the notation for variable-sized vectors.
- Added a section on disaster recovery and backup.
- Added a section on privacy.
- Clarified the assumptions behind the HPKP procedure in the comparison section.
- Added a definition of pin indexing (origin).

- Adjusted to the latest TLS 1.3 notation.

A.6. draft-sheffer-tls-pinning-ticket-00

Initial version.

Authors' Addresses

Yaron Sheffer
Intuit

EMail: yaronf.ietf@gmail.com

Daniel Migault
Ericsson

EMail: daniel.migault@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2018

C. Cramers
L. Garratt
University of Oxford
N. Sullivan
Cloudflare
October 30, 2017

Randomness Improvements for Security Protocols
draft-sullivan-randomness-improvements-00

Abstract

Randomness is a crucial ingredient for TLS and related transport security protocols. Weak or predictable cryptographically-strong pseudorandom number generators (CSPRNGs) can be abused or exploited for malicious purposes. See the Dual EC random number backdoor for a relevant example of this problem. This document describes a way for security protocol participants to mix their long-term private key into the entropy pool from which random values are derived. This may help mitigate problems that stem from broken CSPRNGs.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|--------------------------------------|---|
| 1. Introduction | 2 |
| 2. Randomness Wrapper | 2 |
| 3. Application to TLS | 3 |
| 4. IANA Considerations | 4 |
| 5. Security Considerations | 4 |
| 6. Normative References | 4 |
| Authors' Addresses | 4 |

1. Introduction

Randomness is a crucial ingredient for TLS and related transport security protocols. TLS in particular uses Random Number Generators (RNGs) to generate several values: session IDs, ephemeral key shares, and ClientHello and ServerHello random values. RNG failures such as the Debian bug described in [DebianBug] can lead to insecure TLS connections. RNGs may also be intentionally weakened to cause harm [DualEC]. In such cases where RNGs are poorly implemented or insecure, an adversary may be able to predict its output and recover secret Diffie-Hellman key shares that protect the connection.

This document proposes an improvement to randomness generation in security protocols inspired by the "NAXOS trick" [NAXOS]. Specifically, instead of using raw entropy where needed, e.g., in generating ephemeral key shares, a party's long-term private key is mixed into the entropy pool. In the NAXOS key exchange protocol, raw entropy output x is replaced by $H(x, sk)$, where sk is the sender's private key. Unfortunately, as private keys are often isolated in HSMs, direct access to compute $H(x, sk)$ is impossible. An alternate but functionally equivalent construction is needed.

The approach described herein replaces the NAXOS hash with the keyed hash, or PRF, wherein the key is derived from raw entropy output and a private key signature.

2. Randomness Wrapper

Let x be the raw entropy output of a CSPRNG. When properly instantiated, x should be indistinguishable from a random string of length $|x|$. However, as previously discussed, this is always true. To mitigate this problem, we propose an approach for wrapping the

CSPRNG output with a construction that artificially injects randomness into a value that may be lacking entropy.

Let $\text{PRF}(k, m)$ be a cryptographic pseudorandom function, e.g., HMAC [RFC2104], that takes as input a key k of length L and message m and produces an output of length M . For example, when using HMAC with SHA256, L and M are 256 bits. Let $\text{Sig}(sk, m)$ be a function that computes a signature of message m given private key sk . Let G be an algorithm that generates random numbers from raw entropy, i.e., the output of a CSPRNG. Let tag be a fixed, context-dependent string. Lastly, let KDF be a key derivation function, e.g., HKDF-Extract [RFC5869], that extracts a key of length L suitable for cryptographic use.

The construction is simple: instead of using x when randomness is needed, use:

```
PRF(KDF(G(x) || Sig(sk, tag)), tag)
```

Functionally, this computes the PRF of a fixed string with a key derived from the CSPRNG output and signature over the fixed string. The PRF behaves like a truly random function from 2^L to 2^M assuming the key is selected at random. Thus, the security of this construction depends on secrecy of $\text{Sig}(sk, \text{tag})$ and $G(x)$. If both are leaked, then the security reduces to the scenario wherein this wrapping construction is not applied.

In systems where signature computations are not cheap, these values may be precomputed in anticipation of future randomness requests. This is possible since the construction depends solely upon the CSPRNG output and private key.

3. Application to TLS

The PRF randomness wrapper can be applied to any protocol wherein a party has a long-term private key and also generates randomness. This is true of most TLS servers. Thus, to apply this construction to TLS, one simply replaces the "private" PRNG, i.e., the PRNG that generates private values, such as key shares, with:

```
HMAC(HKDF-Extract(nil, G(x) || Sig(sk, tag)), tag)
```

Moreover, we fix tag as "TLS 1.3 Additional Entropy" for TLS 1.3. Older variants use similarly constructed strings.

4. IANA Considerations

This document makes no request to IANA.

5. Security Considerations

A security analysis was performed by two authors of this document. Generally speaking, security depends on keeping the private key secret. If this secret is compromised, the scheme reduces to the scenario wherein the PRF random wrapper was not applied in the first place.

6. Normative References

[DebianBug]

Yilek, Scott, et al, ., "When private keys are public - Results from the 2008 Debian OpenSSL vulnerability", n.d., <<https://pdfs.semanticscholar.org/fcf9/fe0946c20e936b507c023bbf89160cc995b9.pdf>>.

[DualEC]

Bernstein, Daniel et al, ., "Dual EC - A standardized back door", n.d., <<https://projectbullrun.org/dual-ec/documents/dual-ec-20150731.pdf>>.

[NAXOS]

LaMacchia, Brian et al, ., "Stronger Security of Authenticated Key Exchange", n.d., <<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/strongake-submitted.pdf>>.

[RFC2104]

Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.

[RFC5869]

Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.

Authors' Addresses

Cas Cremers
University of Oxford
Wolfson Building, Parks Road
Oxford
England

Email: cas.cremers@cs.ox.ac.uk

Luke Garratt
University of Oxford
Wolfson Building, Parks Road
Oxford
England

Email: luke.garratt@wolfson.ox.ac.uk

Nick Sullivan
Cloudflare
101 Townsend St
San Francisco
United States of America

Email: nick@cloudflare.com