syslog Working Group	J. Kelsey
Internet-Draft	NIST
Intended status: Standards Track	J. Callas
Expires: June 4, 2010	PGP Corporation
	A. Clemm
	Cisco Systems
	December 01, 2009

Signed syslog Messages
draft-ietf-syslog-sign-29.txt

Abstract

This document describes a mechanism to add origin authentication, message integrity, replay resistance, message sequencing, and detection of missing messages to the transmitted syslog messages. This specification is intended to be used in conjunction with the work defined in RFC 5424, "The syslog Protocol".

Status of this Memo

Copyright Notice

Table of Contents

---

1. Introduction   [TOC](#)

This document describes a mechanism, called syslog-sign in this
document, that adds origin authentication, message integrity, replay
resistance, message sequencing, and detection of missing messages to
syslog. Essentially, this is accomplished by sending a special syslog
message. The contents of this syslog message is called a Signature
Block. Each Signature Block contains, in effect, a detached signature
on some number of previously sent messages. It is cryptographically
signed and contains the hashes of previously sent syslog messages. The
originator of syslog-sign messages is also simply referred to as
"signer". The signer can be the same originator as the originator whose
messages it signs, or it can be a separate originator.
While most implementations of syslog involve only a single originator
and a single collector of each message, provisions need to be made to
cover situations in which messages are sent to multiple collectors.
This concerns, in particular, situations in which different messages
from the same originator are sent to different collectors, which means
that some messages are sent to some collectors but not to others. The

required differentiation of messages is generally performed based on the Priority value of the individual messages. For example, messages from any Facility with a Severity value of 3, 2, 1, or 0 may be sent to one collector while all messages of Facilities 4, 10, 13, and 14 may be sent to another collector. Appropriate syslog-sign messages must be kept with their proper syslog messages. To address this, syslog-sign uses a Signature Group. A Signature Group identifies a group of messages that are all kept together for signing purposes by the signer. A Signature Block always belongs to exactly one Signature Group and always signs messages belonging only to that Signature Group. Additionally, a signer sends Certificate Blocks to provide key management information between the signer and the collector. A Certificate Block has a field to denote the type of key material which may be such things as a PKIX certificate, an OpenPGP certificate, or even an indication that a key had been pre-distributed. In the cases of certificates being sent, the certificates may have to be split across multiple Certificate Blocks carried in separate messages.

It is possible that the same host contains multiple signers that each use their own keys to sign syslog messages. In this case, each signer sends its own Certificate Block and Signature Blocks. Furthermore, each signer defines its own Signature Groups. Each signer on a given host needs to use a distinct combination of APP-NAME and PROCID for its Signature Block and Certificate Block message. (This implies that the combination of HOSTNAME, APP-NAME and PROCID uniquely distinguishes originators of syslog-sign messages across hosts, provided that the signers use a unique HOSTNAME.)

The collector may verify that the hash of each received message matches the signed hash contained in the corresponding Signature Block. A collector may process these Signature Blocks as they arrive, building an authenticated log file. Alternatively, it may store all the log messages in the order they were received. This allows a network operator to authenticate the log file at the time the logs are reviewed.

Because the mechanism that is described in this specification uses the concept of STRUCTURED-DATA elements defined in [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.), compliant implementations of this specification MUST also implement [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.). It is conceivable that the concepts underlying this specification could also be used in conjunction with other message delivery mechanisms. Designers of other efforts to define event notification mechanisms are therefore encouraged to consider this specification in their designs.

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] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.).

---

3. syslog Message Format

This specification is intended to be used in conjunction with the syslog protocol as defined in [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.). The syslog protocol therefore MUST be supported by implementations of this specification.
Because the originator generating the Signature Block message, also simply referred to as "signer", signs each message in its entirety, the messages MUST NOT be changed in transit. By the same token, the syslog-sign messages MUST NOT be changed in transit. One of the effects of such behavior, including message alteration by relays, would be to render any signing invalid and hence make the mechanism useless. Likewise, any truncation of messages that occurs between sending and receiving renders the mechanism useless. For this reason, syslog signer and collector implementations implementing this specification MUST support messages of up to and including 2048 octets in length, in order to minimize the chance of truncation. While syslog signer and collector implementations MAY support messages with a length larger than 2048 octets, implementers need to be aware that any message truncations that occur render the mechanism useless. In such cases, it is up to the operator to ensure that the syslog messages can be received properly and can be validated.
[RFC5426] (Okmianski, A., "Transmission of syslog Messages over UDP," March 2009.) recommends using the TLS transport and deliberately constrains the use of UDP. UDP is NOT RECOMMENDED for use with signed syslog, because its recommended payload size of 480 octets is too restrictive for the purposes of syslog-sign. A 480 octet Signature Block could sign only 9 normal messages, meaning that at a significant proportion of messages would be Signature Block messages. The 480 octet limitation is primarily geared towards small embeded systems with significant resource constraints which, because of those constraints, would not implement syslog-sign in the first place. In addition, the use of UDP is geared towards syslog messages that are primarily intended for troubleshooting, a very different purpose from the application targeted by syslog-sign. Where syslog UDP transport is used, it is the responsibility of operators to ensure that network paths are configured in a way that messages of sufficient length (up to and including 2048 octets) can be properly delivered.

This specification uses the syslog message format described in [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.). Along with other fields, that document describes the concept of Structured Data (SD). Structured Data is defined in terms of SD ELEMENTS (SDEs). An SDE consists of a name and a set of parameter name - value pairs. The SDE name is referred to as SD-ID. The name-value pairs are referred to as SD-PARAM, or SD Parameters, with the name constituting the SD-PARAM-NAME, and the value constituting the SD-PARAM-VALUE.

The syslog messages defined in this document carry the data that is associated with Signature Blocks and Certificate Blocks as Structured Data. The special syslog messages defined in this document include for this purpose definitions of SDEs to convey parameters that relate to the signing of syslog messages. The MSG part of the syslog messages defined in this document SHOULD simply be empty -- the content of the messages is not intended for interpretation by humans but by applications that use those messages to build an authenticated log.

Because the syslog messages defined in this document adhere to the format described in [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.), they identify the machine that originates the syslog message in the HOSTNAME field. Therefore, the Signature Block and Certificate Block data do not need to include any additional parameter to identify the machine that orginates the message.

In addition, several signers MAY sign messages on a single host independently of each other, each using their own Signature Groups. In that case, each unique signer is distinguished by the combination of APP-NAME and PROCID. (By the same token, the same message might be signed by multiple signers.) Each unique signer MUST have a unique APP-NAME and PROCID on each host. (This implies that the combination of HOSTNAME, APP-NAM and PROCID uniquely distinguishes the originator of syslog-sign messages, provided that the signers use a unique HOSTNAME.) A Signature Block message MUST use the same combination of HOSTNAME, APP-NAME, and PROC-ID that was used to send the corresponding Certificate Block messages containing the Payload Block.

---

4. Signature Blocks

This section describes the format of the Signature Block and the fields used within the Signature Block, as well as the syslog messages used to carry the Signature Block.

4.1. syslog Messages Containing a Signature Block

There is a need to distinguish the Signature Block itself from the syslog message that is used to carry a Signature Block. Signature

Blocks MUST be encompassed within completely formed syslog messages. Syslog messages that contain a Signature Block are also referred to as Signature Block messages.

A Signature Block message is identified by the presence of an SD ELEMENT with an SD-ID with the value "ssign". In addition, a Signature Block message MUST contain valid APP-NAME, PROCID, and MSGID fields to be compliant with [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.). This specification does not mandate particular values for these fields; however, for consistency, a signer MUST use the same values for APP-NAME, PROCID, and MSGID fields for every Signature Block message that is sent, whichever values are chosen. It MUST also use the same value for its HOSTNAME field. To allow for the possibility of multiple signers per host, the combination of APP-NAME and PROCID MUST be unique for each such signer on any given host. If a signer daemon is restarted, it MAY use a new PROCID for what is otherwise the same signer but MUST continue to use the same APP-NAME. If it uses a new PROCID, it MUST send a new Payload Block using Certificate Block messages that use the same new PROCID (and the same APP-NAME). It is RECOMMENDED (but not required) to use 110 as value for the PRI field, corresponding to facility 13 (log audit) and severity 6 (informational). The Signature Block is carried as Structured Data within the Signature Block message, per the definitions that follow in the next section. A Signature Block message MAY carry other Structured Data besides the Structured Data of the Signature Block itself. The MSG part of a Signature Block message SHOULD be empty.

The syslog messages defined as part of syslog-sign themselves (Signature Block messages and Certificate Block messages) MUST NOT be signed by a Signature Block. Collectors that implement syslog-sign know to distinguish syslog messages that are associated with syslog-sign from those that are subjected to signing and process them differently. The intent of syslog-sign is to sign a stream of syslog messages, not to alter it.

4.2. Signature Block Format and Fields   TOC

The content of a Signature Block message is the Signature Block. The Signature Block MUST be encoded as an SD ELEMENT, as defined in [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.).
The SD-ID MUST have the value of "ssign".
The SDE contains the fields of the Signature Block encoded as SD Parameters, as specified in the following. The Signature Block is composed of the following fields. The value of each field MUST be printable ASCII, and any binary values MUST be base 64 encoded, as defined in [RFC4648] (Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.).

```
Field                    SD-PARAM-NAME        Size in octets
-----                    -------------        ---- -- ------

Version                       VER                   4

Reboot Session ID             RSID                1-10

Signature Group               SG                    1

Signature Priority            SPRI                 1-3

Global Block Counter          GBC                 1-10

First Message Number          FMN                 1-10

Count                         CNT                  1-2

Hash Block                    HB          variable, size of hash
                                          times the number of hashes
                                            (base 64 encoded binary)

Signature                     SIGN               variable
                                          (base 64 encoded binary)
```

The fields MUST be provided in the order listed. Each SD parameter MUST
occur once and only once in the Signature Block. New SD parameters MUST
NOT be added unless a new Version of the protocol is defined.
(Implementations that wish to add proprietary extensions will need to
define a separate SD ELEMENT.) A Signature Block is accordingly encoded
as follows, where xxx denotes a placeholder for the particular values:
[ssign VER="xxx" RSID="xxx" SG="xxx" SPRI="xxx" GBC="xxx" FMN="xxx"
CNT="xxx" HB="xxx" SIGN="xxx"]
Values of the fields constitute SD parameter values and are hence
enclosed in quotes, per [RFC5424] (Gerhards, R., "The syslog Protocol,"
March 2009.). The fields are separated by single spaces and are
described in the subsequent subsections.

4.2.1. Version

The Version field is an alphanumeric value that has a length of 4
octets, which may include leading zeroes. The first two octets and the
last octet contain a decimal character in the range of "0" to "9",
whereas the third octet contains an alphanumeric character in the range
of "0" to "9", "a" to "z", or "A" to "Z". The value in this field
specifies the version of the syslog-sign protocol. This is extensible
to allow for different hash algorithms and signature schemes to be used

in the future. The value of this field is the grouping of the protocol version (2 octets), the hash algorithm (1 octet) and the signature scheme (1 octet).

> Protocol Version - 2 octets, with "01" as the value for the protocol version that is described in this document.

> Hash Algorithm - 1 octet, where, in conjunction with Protocol Version 01, a value of "1" denotes SHA1 and a value of "2" denotes SHA256, as defined in [FIPS.180-2.2002] (National Institute of Standards and Technology, "Secure Hash Standard," August 2002.). (This is the octet that can have a value of not just "0" to "9" but also "a" to "z" and "A" to "Z".)

> Signature Scheme - 1 octet, where, in conjunction with Protocol Version 01, a value of "1" denotes OpenPGP DSA, defined in [RFC4880] (Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format," November 2007.) and [FIPS.186-2.2000] (National Institute of Standards and Technology, "Digital Signature Standard," January 2000.).

The version, hash algorithm and signature scheme defined in this document would accordingly be represented as "0111" (if SHA1 is used as Hash Algorithm) and "0121" (if SHA256 is used as Hash Algorithm), respectively (without the quotation marks).
The values of the Hash Algorithm and Signature Scheme are defined relative to the Protocol Version. If the single-octet representation of the values for Hash Algorithm and Signature Scheme were to ever represent a limitation, this limitation could be overcome by defining a new Protocol Version with additional Hash Algorithms and/or Signature Schemes, and having implementations support both Protocol Versions concurrently.
As long as sender and receiver are both adhering to [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.), the prerequisites are in place so that signed messages can be received by the receiver and validated with a Signature Block. To ensure immediate validation of received messages all implementations MUST support SHA1, and SHA256 SHOULD be supported.

4.2.2. Reboot Session ID

The Reboot Session ID is a decimal value that has a length between 1 and 10 octets. The acceptable values for this are between 0 and 9999999999. Leading zeroes MUST be omitted.
A Reboot Session ID is expected to strictly monotonically increase (i.e., to never repeat or decrease) whenever a signer reboots in order to allow collectors to distinguish messages and message signatures

across reboots. There are several ways in which this may be accomplished. In one way, the Reboot Session ID may increase by 1, starting with a value of 1. Note that in this case, a signer is required to retain the previous Reboot Session ID across reboots. In another way, a value of the unix time (number of seconds since 1 January 1970) may be used. Implementers of this method need to beware of the possibility of multiple reboots occurring within a single second. Implementers need to also beware of the year 2038 problem, which will cause the 32-bit representation of unix time to wrap in the year 2038. In yet another way, implementations where the SNMP engine and the signer always reboot at the same time might consider using the snmpEngineBoots value as a source for this counter as defined in [RFC3414] (Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)," December 2002.).

In cases where a signer is not able to guarantee that the Reboot Session ID is always increased after a reboot, the Reboot Session ID MUST always be set to a value of 0. If the value can no longer be increased (e.g., because it reaches 9999999999), it SHOULD be reset to a value of 1. Implementations SHOULD ensure that such a reset does not go undetected, for example by requesting operator acknowledgment when a reset is performed upon reboot. (Operator acknowledgment may not be possible in all situations, e.g. in the case of embedded devices.)

If a reboot of a signer takes place, Signature Block messages MAY use a new PROCID. However, Signature Block messages of the same signer MUST continue to use the same HOSTNAME, APP-NAME, and MSGID.

4.2.3. Signature Group and Signature Priority

The SG parameter may take any value from 0-3 inclusive. The SPRI parameter may take any value from 0-191 inclusive. These fields taken together allow network administrators to associate groupings of syslog messages with appropriate Signature Blocks and Certificate Blocks. Groupings of syslog messages that are signed together are also called Signature Groups. A Signature Block contains only hashes of those syslog messages that are part of the same Signature Group.

For example, in some cases, network administrators might have originators send syslog messages of Facilities 0 through 15 to one collector and those with Facilities 16 through 23 to another. In such cases, associated Signature Blocks should likely be sent to the corresponding collectors as well, signing the syslog messages that are intended for each collector separately. This way, each collector receives Signature Blocks for all syslog messages that it receives, and only for those. The ability to associate different categories of syslog messages with different Signature Groups, signed in separate Signature Blocks, provides administrators with flexibility in this regard.

Syslog-sign provides four options for handling Signature Groups, linking them with PRI values so they may be routed to the destination commensurate with the corresponding syslog messages. In all cases, no more than 192 distinct Signature Groups (0-191) are permitted. The Signature Group to which a Signature Block pertains is indicated by the Signature Priority (SPRI) field. The Signature Group (SG) field indicates how to interpret the Signature Priority field. (Note that the SG field does not indicate the Signature Group itself, as its name might suggest.) The SG field can have one of the following values:

a. "0" -- There is only one Signature Group. In this case, the administrators want all Signature Blocks to be sent to a single destination; in all likelihood, all of the syslog messages will also be going to that same destination. Signature Blocks contain signatures for all messages regardless of their PRI value. This means that, in effect, the Signature Block's SPRI value can be ignored. However, it is RECOMMENDED that a single SPRI value be used for all Signature Blocks. Furthermore, it is RECOMMENDED to set that value to the same value as the PRI field of the Signature Block message. This way, the PRI of the Signature Block message matches the SPRI of the Signature Block that it contains.

b. "1" -- Each PRI value is associated with its own Signature Group. Signature Blocks for a given Signature Group have SPRI = PRI for that Signature Group. In other words, the SPRI of the Signature Block matches the PRI value of the syslog messages that are part of the Signature Group and hence signed by the Signature Block. An SG value of 1 can, for example, be used when the administrator of a signer does not know where any of the syslog messages will ultimately go but anticipates that messages with different PRI values will be collected and processed separately. Having a Signature Group per PRI value provides administrators with a large degree of flexibility with regard to how to divide up the processing of syslog messages and their signatures after they are received, at the same time allowing Signature Blocks to follow the corresponding syslog messages to their eventual destination.

c. "2" -- Each Signature Group contains a range of PRI values. Signature Groups are assigned sequentially. A Signature Block for a given Signature Group has its own SPRI value denoting the highest PRI value of syslog messages in that Signature Group. The lowest PRI value of syslog messages in that Signature Group will be one larger than the SPRI value of the previous Signature Group or "0" in case there is no other Signature Group with a lower SPRI value. The specific Signature Groups and ranges they are associated with are subject to configuration by a system administrator.

d.

> "3" -- Signature Groups are not assigned with any of the above relationships to PRI values of the syslog messages they sign. Instead, another scheme is used, which is outside the scope of this specification. There has to be some predefined arrangement between the originator and the intended collectors as to which syslog messages are to be included in which Signature Group, requiring configuration by a system administrator. This provides administrators also with the flexibility to group syslog messages into Signature Groups according to criteria that are not tied to the PRI value. Note that this option is is not intended for deployments which lack such an arrangement, as in those cases a collector could misinterpret the intended meaning of the Signature Group. A collector that receives Signature Block messages of a Signature Group of whose scheme it is not aware SHOULD bring this fact to the attention of the system administrator. The particular mechanism used for that is implementation-specific and outside the scope of this specification.

One reasonable way to configure some installations is to have only one Signature Group, indicated with SG=0, and have the signer send a copy of each Signature Block to each collector. In that case, collectors that are not configured to receive every syslog message will still receive signatures for every message, even ones they are not supposed to receive. While the collector will not be able to detect gaps in the messages (because the presence of a signature of a message that is missing does not tell the collector whether or not the corresponding message would be of the collector's concern), it does allow all messages that do arrive at each collector to be put into the right order and to be verified. It also allows each collector to detect duplicates. Likewise, configuring only one Signature Group can be a reasonable way to configure installations that involve relay chains, where one or more interim relays may or may not relay all messages to the same destination.

4.2.4. Global Block Counter

The Global Block Counter is a decimal value representing the number of Signature Blocks sent by syslog-sign before the current one, in this reboot session. This takes at least 1 octet and at most 10 octets displayed as a decimal counter. The acceptable values for this are between 0 and 9999999999, starting with 0. Leading zeroes MUST be omitted. If the value of the Global Block Counter has reached 9999999999 and the Reboot Session ID has a value other than 0 (indicating the fact that persistence of the Reboot Session ID is

supported), then the Reboot Session ID MUST be incremented by 1 and the Global Block Counter resumes at 0. When the Reboot Session ID is 0 (i.e., persistent Reboot Session IDs are not supported) and the Global Block Counter reaches its maximum value, then the Global Block Counter is reset to 0 and the Reboot Session ID MUST remain at 0.

Note that the Global Block Counter crosses Signature Groups; it allows one to roughly synchronize when two messages were sent, even though they went to different collectors and are part of different Signature Groups.

Because a reboot results in the start of a new reboot session, the signer MUST reset the Global Block Counter to 0 after a reboot occurs. Applications need to take into account the possibility that a reboot occurred when authenticating a log, and situations in which reboots occur frequently may result in losing the ability to verify the proper sequence in which messages were sent, hence jeopardizing the integrity of the log.

4.2.5. First Message Number

This is a decimal value between 1 and 10 octets, with leading zeroes omitted. It contains the unique message number within this Signature Group of the first message whose hash appears in this block. The very first message of the reboot session is numbered "1". This implies that when the Reboot Session ID increases, the message number is reset to 1. For example, if this Signature Group has processed 1000 messages so far and message number 1001 is the first message whose hash appears in this Signature Block, then this field contains 1001. The message number is relative to the Signature Group to which it belongs; hence, a message number does not identify a message beyond its Signature Group.

Should the message number reach 9999999999 within the same reboot session and Signature Group, the message number subsequently restarts at 1. In such event, the Global Block Counter will be vastly different between two occurrences of the same message number.

4.2.6. Count

The count is a 1 or 2 octet field that indicates the number of message hashes to follow. The valid values for this field are 1 through 99. The number of hashes included in the Signature Block MUST be chosen such that the length of the resulting syslog message does not exceed the maximum permissible syslog message length.

4.2.7. Hash Block

The hash block is a block of hashes, each separately encoded in base 64. Each hash in the hash block is the hash of the entire syslog message represented by the hash, independent of the underlying transport. Hashes are ordered from left to right in the order of occurrence of the syslog messages that they represent. The space character is used to separate the hashes. Note, the hash block constitutes a single SD-Param; a Signature Block message MUST include all its hashes in a single hash block and MUST NOT spread its hashes across several hash blocks.

The "entire syslog message" refers to what is described as the syslog message excluding transport parts that are described in [RFC5425] (Miao, F., Yuzhi, M., and J. Salowey, "TLS Transport Mapping for syslog," March 2009.) and [RFC5426] (Okmianski, A., "Transmission of syslog Messages over UDP," March 2009.), and excluding other parts that may be defined in future transports. The hash value will be the result of the hashing algorithm run across the syslog message, starting with the "<" of the PRI portion of the header part of the message. The hash algorithm used and indicated by the Version field determines the size of each hash, but the size MUST NOT be shorter than 160 bits without the use of padding. It is base 64 encoded as per [RFC4648] (Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.). The number of hashes in a hash block SHOULD be chosen such that the resulting Signature Block message does not exceed a length of 2048 octets in order to avoid the possibility that truncation occurs. When more hashes need to be sent than fit inside a Signature Block message, it is advisable to start a new Signature Block.

4.2.8. Signature

This is a digital signature, encoded in base 64 per [RFC4648] (Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.). The signature is calculated over the completely formatted Signature Block message (starting from the first octet of PRI and continuing to the last octet of MSG, or STRUCTURED-DATA if MSG is not present), before the SIGN parameter (SD Parameter Name and the space before it [" SIGN"], "=", and the corresponding value) is added. /**** EDITOR'S NOTE: PLEASE ENSURE THAT '[SIGN"],' APPEARS ON THE SAME LINE AND IN PARTICULAR THAT THE LINE DOES NOT WRAP AFTER THE '['; THE SPACE ' ' HAS SIGNIFICANCE AND IS MISSED IF THE LINE WRAPS. ALSO, PLEASE DELETE THIS NOTE. ****/ (In other words, the digital signature is calculated over the whole message, with the "SIGN=value" portion removed.) For the OpenPGP DSA signature scheme, the value of the signature field contains the DSA values r and s, encoded as two

multiprecision integers (see [RFC4880] (Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format," November 2007.), Sections 5.2.2 and 3.2), concatenated, and then encoded in base 64 [RFC4648] (Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.).

4.2.9. Example

An example of a Signature Block message is depicted below, broken into lines to fit internet-draft publication rules. There is a space at the end of each line, with the exception of the last line which ends with "]".

```
<110>1 2009-05-03T14:00:39.529966+02:00 host.example.org syslogd
2138 - [ssign VER="0111" RSID="1" SG="0" SPRI="0" GBC="2" FMN="1"
CNT="7" HB="K6wzcombEvKJ+UTMcn9bPryAeaU= zrkDcIeaDluypaPCY8WWzwHpPok=
zgrWOdpx16ADc7UmckyIFY53icE= XfopJ+S8/hODapiBBCgVQaLqBKg=
J67gKMFl/OauTC20ibbydwIlJC8= M5GziVgB6KPY3ERU1HXdSi2vtdw=
Wxd/lU7uG/ipEYT9xeqnsfohyH0="
SIGN="AKBbX4J7QkrwuwdbV7Taujk2lvOf8gCgC62We1QYfnrNHz7FzAvdySuMyfM="]
```

The message is of syslog-sign protocol version "01". It uses SHA1 as hash algorithm and an OpenPGP DSA signature scheme. Its reboot session ID is 1. Its Signature Group is 0 which means that all syslog messages go to the same destination; its Signature Priority (which can effectively be ignored because all syslog messages will be signed regardless of their PRI value) is 0. Its Global Block Counter is 2. The first message number is 1; the message contains 7 message hashes.

---

5. Payload and Certificate Blocks

Certificate Blocks and Payload Blocks provide key management for syslog-sign. Their purpose is to support key management that uses public key cryptosystems.

5.1. Preliminaries: Key Management and Distribution Issues

A Payload Block contains public key certificate information that is to be conveyed to the collector. A Payload Block is sent at the beginning of a new reboot session, carrying public key information in effect for the reboot session. However, a Payload Block is not sent directly, but

in (one or more) fragments. Those fragments are termed Certificate
Blocks. Therefore, signers send at least one Certificate Block at the
beginning of a new reboot session.
There are three key points to understand about Certificate Blocks:

a. They handle a variable-sized payload, fragmenting it if
    necessary and transmitting the fragments as legal syslog
    messages. This payload is built (as described below) at the
    beginning of a reboot session and is transmitted in pieces with
    each Certificate Block carrying a piece. There is exactly one
    Payload Block per reboot session.

b. The Certificate Blocks are digitally signed. The signer does not
    sign the Payload Block, but the signatures on the Certificate
    Blocks ensure its authenticity. Note that it may not even be
    possible to verify the signature on the Certificate Blocks
    without the information in the Payload Block; in this case the
    Payload Block is reconstructed, the key is extracted, and then
    the Certificate Blocks are verified. (This is necessary even when
    the Payload Block carries a certificate, because some other
    fields of the Payload Block are not otherwise verified.) In
    practice, most installations keep the same public key over long
    periods of time, so that most of the time, it is easy to verify
    the signatures on the Certificate Blocks, and use the Payload
    Block to provide other useful per-session information.

c. The kind of Payload Block that is expected is determined by what
    kind of key material is on the collector that receives it. The
    signer and collector (or offline log viewer) both have some key
    material (such as a root public key or pre-distributed public
    key) and an acceptable value for the Key Blob Type in the Payload
    Block, below. The collector or offline log viewer MUST NOT accept
    a Payload Block of the wrong type.

5.2. Payload Block

The Payload Block is built when a new reboot session is started. There
is a one-to-one correspondence between reboot sessions and Payload
Blocks. A signer creates a new Payload Block after each reboot. The
Payload Block is used until the next reboot.

5.2.1. Block Format and Fields

A Payload Block MUST have the following fields:

a. Full local time stamp for the signer at the time the reboot session started. This must be in the time stamp format specified in [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.) (essentially, time stamp format per [RFC3339] (Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps," July 2002.) with some further restrictions).

b. Key Blob Type, a one-octet field containing one of five values:

    1. 'C' -- a PKIX certificate (per [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," May 2008.)).

    2. 'P' -- an OpenPGP KeyID and and OpenPGP certificate (a Transferable Public Key as defined in [RFC4880] (Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format," November 2007.), Section 11.1). The first 8 octets of the key blob field contain the OpenPGP KeyID (identifying which key or subkey inside the OpenPGP certificate is used), followed by the OpenPGP certificate itself.

    3. 'K' -- the public key whose corresponding private key is being used to sign these messages. For the OpenPGP DSA signature scheme, the key blob field contains the DSA prime p, DSA group order q, DSA group generator g, and DSA public-key value y, encoded as four multiprecision integers (see [RFC4880] (Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format," November 2007.), Sections 5.5.2 and 3.2).

    4. 'N' -- no key information sent; key is pre-distributed.

    5. 'U' -- installation-specific key exchange information.

c. The key blob, if any, base 64 encoded per [RFC4648] (Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.) and consisting of the raw key data.

The fields are separated by single space characters. Because a Payload Block is not carried in a syslog message directly, only the corresponding Certificate Blocks, it does not need to be encoded as an SD ELEMENT. The Payload Block does not contain a field that identifies the reboot session; instead, the reboot session can be inferred from

the Reboot Session ID parameter of the Certificate Blocks that are used to carry the Payload Block.

To ensure that the sender and receiver have at least one common key blob type, for immediate validation of received messages, all implementations MUST support key blob type "C" (PKIX certificate). When a PKIX certificate is used ("C" key blob type), it is the certificate specified in [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," May 2008.). Per [RFC5425] (Miao, F., Yuzhi, M., and J. Salowey, "TLS Transport Mapping for syslog," March 2009.), syslog messages may be transported over the TLS protocol, even where there is no PKI. If that transport is used, then the device will already have a PKIX certificate and it MAY use the private key associated with that certificate to sign messages. In the case where there is no PKI, the chain of trust of a PKIX certificate must still be established to meet conventional security requirements. The methods for doing this are described in [RFC5425] (Miao, F., Yuzhi, M., and J. Salowey, "TLS Transport Mapping for syslog," March 2009.).

5.2.2. Signer Authentication and Authorization

When the collector receives a Payload Block, it needs to determine whether the signatures are to be trusted. The following methods are in scope of this specification:

a. X.509 certification path validation: The collector is configured with one or more trust anchors (typically root CA certificates), which allow it to verify a binding between the subject name and the public key. Certification path validation is performed as specified in [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," May 2008.).
    If the HOSTNAME contains an FQDN or an IP address, it is then compared against the certificate as described in [RFC5425] (Miao, F., Yuzhi, M., and J. Salowey, "TLS Transport Mapping for syslog," March 2009.), Section 5.2. Comparing other forms of HOSTNAMEs is beyond the scope of this specification.

    Collectors SHOULD support this method.

    Note that due to message size restrictions, syslog-sign sends only the end-entity certificate in the Payload Block. Depending on the PKI deployment, the collector may need to obtain intermediate certificates by other means (for example, from a directory).

b.

   X.509 end-entity certificate matching: The collector is configured with information necessary to identify the valid end-entity certificates of its valid peers, and for each peer, the HOSTNAME(s) it is authorized to use.
   To ensure interoperability, collectors MUST support fingerprints of X.509 certificates as described below. Other methods MAY be supported.

   Collectors MUST support key blob type 'C', and specifying the list of valid peers using certificate fingerprints. The fingerprint is calculated and formatted as specified in [RFC5425] (Miao, F., Yuzhi, M., and J. Salowey, "TLS Transport Mapping for syslog," March 2009.), Section 4.2.2.

   For each peer, the collector MUST support specifying a list of HOSTNAME(s) this peer is allowed to use either as FQDNs or IP addresses. Other forms of HOSTNAMEs are beyond the scope of this specification.

   If the locally configured FQDN is an internationalized domain name, conforming implementations MUST convert it to the ASCII Compatible Encoding (ACE) format for performing comparisons as specified in Section 7 of [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," May 2008.). An exact case-insensitive string match MUST be supported, but the implementation MAY also support wildcards of any type ("*", regular expressions, etc.) in locally configured names.

   Signer implementations MUST provide a means to generate a key pair and self-signed certificate in the case that a key pair and certificate are not available through another mechanism, and MUST make the certificate fingerprint available through a management interface.

c. OpenPGP V4 fingerprints: Like X.509 fingerprints, except key blob type 'P' is used, and the fingerprint is calculated as specified in [RFC4880] (Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format," November 2007.), Section 12.2. When the fingerprint value is display or configured, each byte is represented in hexadecimal (using two uppercase ASCII characters), and space is added after every second byte. For example: "0830 2A52 2CD1 D712 6E76 6EEC 32A5 CAE1 03C8 4F6E".
   Signers and collectors MAY support this method.

Other methods, such as "web of trust", are beyond the scope of this document.

5.3. Certificate Block

This section describes the format of the Certificate Block and the fields used within the Certificate Block, as well as the syslog messages used to carry Certificate Blocks.

5.3.1. syslog Messages Containing a Certificate Block

Certificate Blocks are used to get the Payload Block to the collector. As with a Signature Block, each Certificate Block is carried in its own syslog message, called Certificate Block message. In case separate collectors are associated with different Signature Groups, Certificate Block messages need to be sent to each collector.
Because certificates can legitimately be much longer than 2048 octets, the Payload Block can be split up into several pieces, with each Certificate Block carrying a piece of the Payload Block. Note that the signer MAY make the Certificate Blocks of any legal length (that is, any length that keeps the entire Certificate Block message within 2048 octets) that holds all the required fields. Software that processes Certificate Blocks MUST deal correctly with blocks of any legal length. The length of the fragment of the Payload Block that a Certificate Block carries MUST be at least 1 octet. The length SHOULD be chosen such that the length of the Certificate Block message does not exceed 2048 octets.
A Certificate Block message is identified by the presence of an SD ELEMENT with an SD-ID with the value "ssign-cert". In addition, a Certificate Block message MUST contain valid APP-NAME, PROCID, and MSGID fields to be compliant with syslog protocol. Syslog-sign does not mandate particular values for these fields; however, for consistency, a signer MUST use the same value for APP-NAME, PROCID, and MSGID fields for every Certificate Block message, whichever values are chosen. It MUST also use the same value for its HOSTNAME field. To allow for the possibility of multiple signers per host, the combination of APP-NAME and PROCID MUST be unique for each such originator. If a signer daemon is restarted, it MAY use a new PROCID for what is otherwise the same signer. The combination of APP-NAME and PROCID MUST be the same that is used for Signature Block messages of the same signer; however, a different MSGID MAY be used for Signature Block and Certificate Block messages. It is RECOMMENDED to use 110 as value for the PRI field, corresponding to facility 13 (log audit) and severity 6 (informational). The Certificate Block is carried as Structured Data

within the Certificate Block message. A Certificate Block message MAY carry other Structured Data besides the Structured Data of the Certificate Block itself. The MSG part of a Certificate Block message SHOULD be empty.

5.3.2. Certificate Block Format and Fields

The contents of a Certificate Block message is the Certificate Block itself. Like a Signature Block, the Certificate Block is encoded as an SD ELEMENT. The SD-ID of the Certificate Block is "ssign-cert". The Certificate Block is composed of the following fields, each of which is encoded as an SD Parameter with parameter name as indicated. Each field must be printable ASCII, and any binary values are base 64 encoded per [RFC4648] (Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.).

Field	SD-PARAM-NAME	Size in octets
-----	-------------	---- -- ------
Version	VER	4
Reboot Session ID	RSID	1-10
Signature Group	SG	1
Signature Priority	SPRI	1-3
Total Payload Block Length	TPBL	1-8
Index into Payload Block	INDEX	1-8
Fragment Length	FLEN	1-4
Payload Block Fragment	FRAG	variable (base 64 encoded binary)
Signature	SIGN	variable (base 64 encoded binary)

The fields MUST be provided in the order listed. New SD parameters MUST NOT be added unless a new Version of the protocol is defined. (Implementations that wish to add proprietary extensions will need to define a separate SD ELEMENT.) A Certificate Block is accordingly encoded as follows, where xxx denotes a placeholder for the particular values:
[ssign-cert VER="xxx" RSID="xxx" SG="xxx" SPRI="xxx" TPBL="xxx" INDEX="xxx" FLEN="xxx" FRAG="xxx" SIGN="xxx"]

Values of the fields constitute SD parameter values and are hence enclosed in quotes, per [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.). The fields are separated by single spaces and are described below. Each SD parameter MUST occur once and only once.

5.3.2.1. Version

The Version field is 4 octets in length. This field is identical in format and meaning to the Version field described in Section 4.2.1 (Version).

5.3.2.2. Reboot Session ID

The Reboot Session ID is identical in format and meaning to the RSID field described in Section 4.2.2 (Reboot Session ID).

5.3.2.3. Signature Group and Signature Priority

The SIG field is identical in format and meaning to the SIG field described in Section 4.2.3 (Signature Group and Signature Priority). The SPRI field is identical in format and meaning to the SPRI field described there.
A signer SHOULD send separate Certificate Block messages for each Signature Group. This ensures that each collector that is associated with a Signature Group will receive the necessary key material in the case that messages of different Signature Groups are sent to different collectors. Note that the signer needs to get the same Payload Block to each collector, as for any given signer there is a one-to-one relationship between Payload Block and Reboot Session across all Signature Groups. Deployments that wish to associate different key material (and hence different Payload Blocks) with different Signature Groups can use separate signers for that purpose, each distinguished by its own combination of HOSTNAME, APP-NAME, PROCID.

5.3.2.4. Total Payload Block Length

The Total Payload Block Length is a value representing the total length of the Payload Block in octets, expressed as a decimal with one to eight octets with leading zeroes omitted.

5.3.2.5. Index into Payload Block

This is a decimal value between 1 and 8 octets, with leading zeroes omitted. It contains the number of octets into the Payload Block at which this fragment starts. The first octet of the first fragment is numbered "1". (Note, it is not numbered "0".)

5.3.2.6. Fragment Length

The total length of this fragment expressed as a decimal integer with one to four octets with leading zeroes omitted. The fragment length must be at least 1.

5.3.2.7. Payload Block Fragment

The Payload Block Fragment contains a fragment of the payload block. Its length must match the indicated fragment length.

5.3.2.8. Signature

This is a digital signature, encoded in base 64, as per [RFC4648] (Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.). The Version field effectively specifies the original encoding of the signature. The signature is calculated over the completely formatted Certificate Block message, before the SIGN parameter is added (see Section 4.2.8 (Signature)). For the OpenPGP DSA signature scheme, the value of the signature field contains the DSA values r and s, encoded as two multiprecision integers (see [RFC4880] (Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format," November 2007.), Sections 5.2.2 and 3.2), concatenated, and then encoded in base 64 [RFC4648] (Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.).

5.3.2.9. Example

An example of a Certificate Block message is is depicted below, broken
into lines to fit internet-draft publication rules. There are no spaces
at the end of the lines that contain the key blob and the signature.

```
<110>1 2009-05-03T14:00:39.519307+02:00 host.example.org syslogd
2138 - [ssign-cert VER="0111" RSID="1" SG="0" SPRI="0" TPBL="587"
INDEX="1" FLEN="587" FRAG="2009-05-03T14:00:39.519005+02:00 K BACsLMZ
NCV2NUAwe4RAeAnSQuvv2KS51SnHFAaWJNU2XVDYvW1LjmJgg4vKvQPo3HEOD+2hEkt1z
cXADe03u5pmHoWy5FGiyCbglYxJkUJJrQqlTSS6vID9yhsmEnh07w3pOsxmb4qYo0uWQr
AAenBweVMlBgV3ZA5IMA8xq8l+i8wCgkWJjCjfLar7s+0X3HVrRroyARv8EAIYoxofh9m
N8n821BTTuQnz5hp40d6Z3UudKePu2di5Mx3GFelwnV0Qh5mSs0YkuHJg0mcXyUAoeYry
5X6482fUxbm+gOHVmYSDtBmZEB8PTEt8Os8aedWgKEt/E4dT+Hmod4omECLteLXxtScTM
gDXyC+bSBMjRRCaeWhHrYYdYBACCWMdTc12hRLJTn8LX99kv1I7qwgieyna8GCJv/rEgC
ssS9E1qARM+h19KovIUOhl4VzBw3rK7v8Dlw/CJyYDd5kwSvCwjhO21LiReeS90VPYuZF
RC1B82Sub152zOqIcAWsgd4myCCiZbWBsuJ8P0gtarFIpleNacCc6OV3i2Rg=="
SIGN="AKAQEUiQptgpd0lKcXbuggGXH/dCdQCgdysrTBLUlbeGAQ4vwrnLOqSL7+c="]
```

The message is of syslog-sign protocol version "01". It uses SHA1 as
hash algorithm and an OpenPGP DSA signature scheme. Its reboot session
ID is 1. Its Signature Group is 0; its Signature Priority is 0. The
Total Payload Block Length is 587. The index into the payload block is
1 (meaning this is the first fragment). The length of the fragment is
587 (meaning that the Certificate Block message contains the entire
Payload Block). The Payload Block has the time stamp
2009-05-03T14:00:39.519005+02:00. The Key Blob Type is 'K', meaning
that it contains a public key whose corresponding private key is being
used to sign these messages.
Note that the Certificate Block message in this example has a time
stamp that is very close to the time stamp in the Payload Block. The
fact that the time stamps are so close implies that this is the first
Certificate Block message sent in this reboot session; additional
Certificate Block messages can be sent later with a later time stamp,
which will carry the same Payload Block that will still contain the
same time stamp.

---

6. Redundancy and Flexibility

As described in Section 8.5 of [RFC5424] (Gerhards, R., "The syslog
Protocol," March 2009.), a transport sender may discard syslog
messages. Likewise, when syslog messages are sent over unreliable
transport, they can be lost in transit. However, if a collector does
not receive Signature and Certificate Blocks, many messages may not be
able to be verified. The signer is allowed to send Signature and
Certificate Blocks multiple times. Sending Signature and Certificate
Blocks multiple times provides redundancy with the intent to ensure

that the collector or relay does get the Signature Blocks and in particular the Payload Block at some point in time. In the meantime, any online review of logs as described in Section 7.2 (Online Review of Logs) is delayed until the needed blocks are received. The collector MUST ignore duplicats of Signature Blocks and Certificate Blocks that it has already received and authenticated. The signer can in principle change its redundancy level for any reason, without communicating this fact to the collector.

A signer that is also the originator of messages that it signs does not need to queue up other messages while sending redundant Certificate Block and Signature Block messages. It MAY send redundant Certificate Block messages even after Signature Block messages and regular syslog messages have been sent. By the same token, it MAY send redundant Signature Block messages even after newer syslog messages that are signed by a subsequent Signature Block have been sent, or even after a subsequent Signature Block message.

In addition, the signer has flexibility in how many hashes to include within a Signature Block. It is legitimate for an originator to send short Signature Blocks to allow the collector to verify messages with minimal delay.

6.1. Configuration parameters

Although the transport sender is not constrained in how it decides to send redundant Signature and Certificate Blocks, or even in whether it decides to send along multiple copies of normal syslog messages, we define some redundancy parameters below which may be useful in controlling redundant transmission from the transport sender to the transport receiver, and which may be useful for administrators to configure.

6.1.1. Configuration Parameters for Certificate Blocks

Certificate Blocks are always sent at the beginning of a new reboot session. One technique to ensure reliable delivery (see Section 8.5) is to send multiple copies. This can be controlled by a "certInitialRepeat" parameter:

> certInitialRepeat = number of times each Certificate Block should be sent before the first message is sent.

It is also useful to resend Certificate Blocks every now and then for long-lived reboot sessions. This can be controlled by the certResendDelay and certResendCount parameters:

> certResendDelay = maximum time delay in seconds until resending the
> Certificate Block.
>
> certResendCount = maximum number of other syslog messages to send
> until resending the Certificate Block.

In some cases, it may be desirable to allow for configuration of the
transport sender such that Certificate Blocks are not sent at all after
the first normal syslog message has been sent. This could be expressed
by setting both certResendDelay and certResendCount to "0". However, it
is RECOMMENDED to configure the transport sender to send redundant
Certificate Blocks even after the first message, in particular when the
UDP transport [RFC5426] (Okmianski, A., "Transmission of syslog
Messages over UDP," March 2009.) is used.

In one set of circumstances, the receiver may receive a Certificate
Block, some group of syslog messages and some corresponding Signature
Blocks. If the receiver reboots after that then the conditions of
recovery will vary depending upon the transport. For UDP [RFC5426]
(Okmianski, A., "Transmission of syslog Messages over UDP,"
March 2009.), the receiver SHOULD continue to use the cached
Certificate Block, but MUST validate the RSID value to make sure that
it has the most current one. If the receiver cannot validate that it
has the most current Certificate Block, then it MUST wait for a
retransmission of the Certificate Block, which may be controlled by the
certResendDelay and certResendCount parameters. It is up to the
operators to ensure that Certificate Blocks are sent frequently enough
to meet this set of circumstances.

For TLS transport [RFC5425] (Miao, F., Yuzhi, M., and J. Salowey, "TLS
Transport Mapping for syslog," March 2009.), the sender MUST send a
fresh Certificate Block when a session is established. This will keep
the sender and receiver synchronized with the most current Certificate
Block.

Implementations which support sending syslog messages of different
Signature Groups to different collectors and which wish to offer very
granular controls MAY allow the above parameters to be configured on a
per Signature Group basis.

The choice of reasonable values in a given deployment depends on
several factors, including the acceptable delay that that may be
incurred from the receipt of a syslog message until the corresponding
Signature Block is received, whether UDP or TLS transport is used, and
the available management bandwidth. The following might be a reasonable
choice for a deployment in which reliability of underlying transport
and of collector implementation are of little concern:
certInitialRepeat=1, certResendDelay=1800 seconds,
certResendCount=10000

The following might be a reasonable choice for a deployment in which
reliability of transmission over UDP transport could be an issue:
certInitialRepeat=2, certResendDelay=300 seconds, certResendCount=1000

6.1.2. Configuration Parameters for Signature Blocks

Verification of log messages involves a certain delay of time that is caused by the lag in time between the sending of the message itself and the corresponding Signature Block. The following configuration parameter can be useful to limit the time lag that will be incurred (note that the maximum message length may also force generating a Signature Block; see Sections Section 4.2.6 (Count) and Section 4.2.7 (Hash Block)):

> sigMaxDelay = generate a new Signature Block if this many seconds have elapsed since the message with the First Message Number of the Signature Block was sent.

Retransmissions of Signature Blocks are not sent immediately after the original transmission, but slightly later. The following parameters control when those retransmissions are done:

> sigNumberResends = number of times a Signature Block is resent. (It is recommended so select a value of greater than "0" in particular when the UDP transport [RFC5426] (Okmianski, A., "Transmission of syslog Messages over UDP," March 2009.) is used.)
>
> sigResendDelay = send the next retransmission when this many seconds have elapsed since the previous sending of this Signature Block.
>
> sigResendCount = send the next retransmission when this many other syslog messages have been sent since the previous sending of this Signature Block.

The choice of reasonable values in a given deployment depends on several factors, including the acceptable delay that that may be incurred from the receipt of a syslog message until the corresponding Signature Block is received so that the syslog message can be verified, the reliability of the underlying transport, and the available management bandwidth. The following might be a reasonable choice for a deployment where reliability of transport and collector are of little concern and where there is a need to have syslog messages generally signed within 5 minutes:
sigMaxDelay=300 seconds, sigNumberResends=2, sigResendDelay=300 seconds, sigResendCount=500
The following would be a reasonable choice for a deployment that needs to validate syslog messages typically within 60 seconds, but no more than 3 minutes after receipt:
sigMaxDelay=30 seconds, sigNumberResends=5, sigResendDelay=30 seconds, sigResendCount=100

6.2. Overlapping Signature Blocks

Notwithstanding the fact that the signer is not constrained in whether it decides to send redundant Signature Block messages, Signature Blocks SHOULD NOT overlap. This facilitates their processing by the receiving collector. This means that an originator of Signature Block messages, after having sent a first message with some First Message Number and a Count, SHOULD NOT send a second message with the same First Message Number but a different Count. It also means that an originator of Signature Block messages SHOULD NOT send a second message whose First Message Number is greater than the First Message Number, but smaller than the First Message Number plus the Count indicated in the first message.
That said, the possibility of Signature Blocks that overlap does provide additional flexibility with regards to redundancy; it provides an additional option that may be desirable in some deployments. Therefore collectors MUST be designed in a way that they can cope with overlapping Signature Blocks when confronted with them. The collector MUST ignore hashes of messages that it has already received and validated.

---

7. Efficient Verification of Logs

The logs secured with syslog-sign may be reviewed either online or offline. Online review is somewhat more complicated and computationally expensive, but not prohibitively so. This section outlines a method for online and a method for offline verification of logs which implementations MAY choose to implement to verify logs efficiently. Implementations MAY also choose to implement a different method; it is ultimately up to each implementation how to process the messages that it receives.

7.1. Offline Review of Logs

When the collector stores logs to be reviewed later, they can be authenticated offline just before they are reviewed. Reviewing these logs offline is simple and relatively inexpensive in terms of resources used, so long as there is enough space available on the reviewing machine.
To do so, we first go through the stored log file. Each message in the log file is classified as a normal message, a Signature Block message, or a Certificate Block message. Signature Blocks and Certificate Blocks are then separated by signer (as identified by HOSTNAME, APP-NAME,

PROCID), Reboot Session ID, and Signature Group, and stored in their own files. Normal messages are stored in a keyed file, indexed on their hash values. They are not separated by signer, as their (HOSTNAME, APP-NAME, PROCID) identifies the application that generated the message. The application that generated the message does not have to coincide with the signer.

For each signer, Reboot Session ID, and Signature Group, we then do the following:

a. We sort the Certificate Block file by INDEX value, and check to see whether we have a set of Certificate Blocks that can reconstruct the Payload Block. If so, we reconstruct the Payload Block, verify any key-identifying information, and then use this to verify the signatures on the Certificate Blocks we have received. When this is done, we have verified the reboot session and key used for the rest of the process.

b. We sort the Signature Block file by First Message Number. We now create an authenticated log file, which consists of some header information and then a sequence of message number, message text pairs. We next go through the Signature Block file. We initialize a cursor for the last message number processed with the number 0. For each Signature Block in the file, we do the following:

    1. Verify the signature on the Signature Block.

    2. If the value of the First Message Number of the Signature Block is less than or equal to the last message number processed, skip the first (last message number processed minus First Message Number plus 1) hashes.

    3. For each remaining hashed message in the Signature Block:

        a. Look up the hash value in the keyed message file.

        b. If the message is found, write (message number, message text) to the authenticated log file.

    4. Set the last message number processed to the value of the First Message Number plus the Count of the Signature Block minus 1.

    5. Skip all other Signature Blocks with the same First Message Number unless one with a larger Count is encountered.

The resulting authenticated log file contains all messages that have been authenticated. In addition, it implicitly indicates all gaps in the authenticated messages (specifically in the case when

> all messages of the same Signature Group are sent to the same
> collector), because their message numbers are missing.

One can see that, assuming sufficient space for building the keyed
file, this whole process is linear in the number of messages (generally
two seeks, one to write and the other to read, per normal message
received), and O(N lg N) in the number of Signature Blocks. This
estimate comes with two caveats: first, the Signature Blocks arrive
very nearly in sorted order, and so can probably be sorted more cheaply
on average than O(N lg N) steps. Second, the signature verification on
each Signature Block almost certainly is more expensive than the
sorting step in practice. We have not discussed error-recovery, which
may be necessary for the Certificate Blocks. In practice, a simple
error-recovery strategy is probably enough: if the Payload Block is not
valid, then we can just try alternate instances of each Certificate
Block, if such are available, until we get the Payload Block right.
It is easy for an attacker to flood us with plausible-looking messages,
Signature Blocks, and Certificate Blocks.

7.2. Online Review of Logs

Some collector implementations may need to monitor log messages in
close to real-time. This can be done with syslog-sign, though it is
somewhat more complex than offline verification. This is done as
follows:

a. We have an authenticated message file, into which we write
    (message number, message text) pairs which have been
    authenticated. We will assume that we are handling only one
    signer, Signature Group, and Reboot Session ID at any given time.
    (For the concurrent support of multiple signers, Signature
    Groups, and Reboot Session IDs, the same procedure is applied
    analogously to each. Signature Block mesages and Certificate
    Block messages clearly indicate their respective signer,
    Signature Group, and Reboot Session ID.)

b. We have two data structures: A "Waiting for Signature" queue in
    which (arrival sequence, hash of message) pairs are kept in
    sorted order, and a "Waiting for Message" queue in which (message
    number, hash of message) pairs are kept in sorted order. In
    addition, we have a hash table that stores (message text, count)
    pairs indexed by hash value. In the hash table, count may be any
    number greater than zero; when count is zero, the entry in the
    hash table is cleared.
    Note: The "Waiting For Signature" queue gets used in the normal
    case, when the signature arrives after the message itself. It
    holds messages that have been received but whose signature has

yet to arrive. The "Waiting for Message" queue gets used in the case that messages are lost or misordered (either in the network or in relays). It holds signatures that have been received but whose corresponding messages have yet to arrive. Since a single Signature Block can cover only a limited number of messages (due to size restrictions), and massive reordering/delaying is rare, it is expected that both queues would be relatively small.

c. We must receive all the Certificate Blocks before any other processing can really be done. (This is why they are sent first.) Once that is done, any additional Certificate Block message that arrives is discarded. Any syslog messages or Signature Block messages that arrive before all Certificate Blocks have been received need to be buffered. Once all Certificate Blocks have been received, the messages in the buffer can be retrieved and processed as if they were just arriving.

d. Whenever a normal message arrives, we first check if its hash value is found in the "Waiting For Message" queue. If it is, we write the message number (from the "Waiting for Message" queue) and the message into the authenticated message file and remove the entry from the queue.
    Otherwise, we add (arrival sequence, hash of message) to the "Waiting For Signature" queue. If our hash table already has an entry for the message's hash value, we increment its count by one; otherwise, we create a new entry with count = 1.

    If the "Waiting For Signature" message queue is full, we remove the oldest message from the queue. That message could not be validated close enough to real time. In order to update the hash table accordingly, we use that entry's hash to index the hash table. If that entry has count 1, we delete the entry from the hash table; otherwise, we decrement its count. By removing the message from the "Waiting for Signature" message queue without having actually received the message's signature, we make it impossible to authenticate the message should its signature arrive later. Implementors therefore need to ensure that queues are dimensioned sufficiently large to not expose the collector against DoS attacks that attempt to flood the collector with unsigned messages.

e. Whenever a Signature Block message arrives, we check its originator, (i.e. the signer) by way of HOSTNAME, APP-NAME, PROCID, as well as its Signature Group and Reboot Session ID to ensure it matches our Certificate Blocks. We then check to see whether the First Message Number value is too old to still be of interest, or if another Signature Block with that First Message Number and the same Count or a greater Count has already been received. If so, we discard the Signature Block. We then check

the signature. Again, we discard the Signature Block if the
signature is not valid.
Otherwise, we proceed with processing the hashes in the Signature
Block. A Signature Block contains a sequence of hashes, each of
which is associated with a message number, starting with the
First Message Number for the first hash and incrementing by one
for each subsequent hash. For each hash, we first check to see
whether the message hash is in the hash table. If this is the
case, it means that we have received the signature for a message
that was received earlier, and we do the following:

1. We check if a message with the same message number is
   already in the authenticated message file. If that is the
   case, the signed hash is a duplicate and we discard it.

2. Otherwise (the signed hash is not a duplicate), we write
   the (message number, message text) into the authenticated
   message file. We also update the hash table accordingly,
   using that entry's hash to index the hash table. If that
   entry has count 1, we delete the entry from the hash
   table; otherwise, we decrement its count.

Otherwise (the message hash is not in the hash table), we write
the (message number, message hash) to the "Waiting for Message"
queue.

If the "Waiting for Message" queue is full, we remove the oldest
entry. In that case, a message that was signed by the signer
could not be validated by the receiver, either because the
message was lost, or because the signature arrived way ahead of
the actual message. By removing the entry from the "Waiting for
Message" queue without having actually received the message, we
make it impossible to authenticate the a legitimate message
should that message still arrive later. Implementors need to
ensure queues are dimensioned sufficiently large so that the
chances of such a scenario actually occcurring is minimized.

f. The result of this is a sequence of messages in the
    authenticated message file. Each message in the message file has
    been authenticated. The sequence is labeled with numbers showing
    the order in which the messages were originally transmitted.

One can see that this whole process is roughly linear in the number of
messages, and also in the number of Signature Blocks received. The
process is susceptible to flooding attacks; an attacker can send enough
normal messages that the messages roll off their queue before their
Signature Blocks can be processed.

8. Security Considerations

Normal syslog event messages are unsigned and have most of the security attributes described in Section 8 of [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.). This document also describes Certificate Blocks and Signature Blocks, which are signed syslog messages. The Signature Blocks contain signature information for previously sent syslog event messages. All of this information can be used to authenticate syslog messages and to minimize or obviate many of the security concerns described in [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.).
The model for syslog-sign is a direct trust system where the certificate transferred is its own trust anchor. If a transport sender sends a stream of syslog messages that is signed using a certificate, the operator or application will transfer to the transport receiver the certificate that was used when signing. There is no need for a certificate chain.

8.1. Cryptographic Constraints

As with any technology involving cryptography, it is advisable to check the current literature to determine whether any algorithms used here have been found to be vulnerable to attack.
This specification uses Public Key Cryptography technologies. The proper party or parties have to control the private key portion of a public-private key pair. Any party that controls a private key can sign anything it pleases.
Certain operations in this specification involve the use of random numbers. An appropriate entropy source SHOULD be used to generate these numbers. See [RFC4086] (Eastlake, D., Schiller, J., and S. Crocker, "Randomness Recommendations for Security," June 2005.) and [NIST800.90] (National Institute of Standards and Technology, "NIST Special Publication 800-90: Recommendation for Random Number Generation using Deterministic Random Bit Generators," June 2006.).

8.2. Packet Parameters

As a signer, it is advisable to avoid message lengths exceeding 2048 octets. Various problems might result if a signer were to send messages with a length greater than 2048 octets, because relays MAY truncate messages with lengths greater than 2048 octets which would make it impossible for collectors to validate a hash of the packet. To increase

the chance of interoperability, it tends to be best to be conservative with what you send but liberal in what you are able to receive. Signers need to rigidly adhere to the RFC5424 format when sending messages. If a collector receives a message that is not formatted properly then it might drop it, or it may modify it while receiving it. (See A.2 of [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.).) If that were to happen, the hash of the sent message would not match the hash of the received message.

Collectors are not to malfunction in case they receive malformed syslog messages or messages containing characters other than those specified in this document. In other words, they are to ignore such messages and continue working.

8.3. Message Authenticity

Syslog does not strongly associate the message with the message originator. That association is established by the collector upon verification of the Signature Block. Before a Signature Block is used to ascertain the authenticity of an event message, it might be received, stored, and reviewed by a person or automated parser. It is advisable not to assume a message is authentic until after a message has been validated by checking the contents of the Signature Block. With the Signature Block checking, an attacker may only forge messages if he or she can compromise the private key of the true originator.

8.4. Replaying

Event messages might be recorded and replayed by an attacker. Using the information contained in the Signature Blocks, a reviewer can determine whether the received messages are the ones originally sent by an originator. The reviewer can also identify messages that have been replayed. Using a method for verification of logs such as the one outlined in Section 7 (Efficient Verification of Logs), a replayed message can be detected by checking prior to writing a message to the authenticated log file whether the message is already contained in it.

8.5. Reliable Delivery

Event messages sent over UDP might be lost in transit. [RFC5425] (Miao, F., Yuzhi, M., and J. Salowey, "TLS Transport Mapping for syslog," March 2009.) can be used for the reliable delivery of syslog messages;

however, it does not protect against loss of syslog messages at the
application layer, for example if the TCP connection or TLS session has
been closed by the transport receiver for some reason. A reviewer can
identify any messages sent by the originator but not received by the
collector by reviewing the Signature Block information. In addition,
the information in subsequent Signature Blocks allows a reviewer to
determine whether any Signature Block messages were lost in transit.

8.6. Sequenced Delivery

Syslog messages delivered over UDP might not only be lost, but also
arrive out of sequence. A reviewer can determine the original order of
syslog messages and identify which messages were delivered out of order
by examining the information in the Signature Block along with any
timestamp information in the message.

8.7. Message Integrity

Syslog messages might be damaged in transit. A review of the
information in the Signature Block determines whether the received
message was the intended message sent by the originator. A damaged
Signature Block or Certificate Block is evident because the collector
will not be able to validate that it was signed by the signer.

8.8. Message Observation

Unless TLS is used as a secure transport [RFC5425] (Miao, F., Yuzhi,
M., and J. Salowey, "TLS Transport Mapping for syslog," March 2009.),
event messages, Certificate Blocks, and Signature Blocks are all sent
in plaintext. This allows network administrators to read the message
when sniffing the wire. However, this also allows an attacker to see
the contents of event messages and perhaps to use that information for
malicious purposes.

8.9. Man In The Middle Attacks

It is conceivable that an attacker might intercept Certificate Block
messages and insert its own Certificate information. In that case, the
attacker would be able to receive event messages from the actual

originator and then relay modified messages, insert new messages, or delete messages. It would then be able to construct a Signature Block and sign it with its own private key. Network administrators need to verify that the key contained in the Payload Block is indeed the key being used on the actual signer. If that is the case, then this MITM attack will not succeed. Methods for establishing a chain of trust are also described in [RFC5425] (Miao, F., Yuzhi, M., and J. Salowey, "TLS Transport Mapping for syslog," March 2009.).

8.10. Denial of Service

An attacker might send invalid Signature Block messages to overwhelm the collector's processing capability and consume all available resources. For this reason, it can be appropriate to simply receive the Signature Block messages and process them only as time permits.
An attacker might also just overwhelm a collector by sending more messages to it than it can handle. Implementers are advised to consider features that minimize this threat, such as only accepting syslog messages from known IP addresses.

8.11. Covert Channels

Nothing in this protocol attempts to eliminate covert channels. In fact, just about every aspect of syslog messages lends itself to the conveyance of covert signals. For example, a collusionist could send odd and even PRI values to indicate Morse Code dashes and dots.

---

9. IANA Considerations

9.1. Structured Data and syslog messages

With regard to [RFC5424] (Gerhards, R., "The syslog Protocol," March 2009.), IANA is requested to add the following values to the registry entitled "syslog Structured Data id values":

```
            SD-ID        PARAM_NAME
            -----        ----------
            ssign

                         VER
                         RSID
                         SG
                         SPRI
                         GBC
                         FMN
                         CNT
                         HB
                         SIGN


        ssign-cert

                         VER
                         RSID
                         SG
                         SPRI
                         TPBL
                         INDEX
                         FLEN
                         FRAG
                         SIGN
```

In addition, several fields need to be controlled by the IANA in both
the Signature Block and the Certificate Block, as outlined in the
following sections.

9.2. Version Field

IANA is requested to create three registries, each associated with a
different subfield of the Version field of Signature Blocks and
Certificate Blocks, described in Section 4.2.1 (Version) and
Section 5.3.2.1 (Version), respectively.
The first registry that IANA is requested to create is entitled
"syslog-sign protocol version values". It is for the values of the
Protocol Version subfield. The Protocol Version subfield constitutes
the first 2 octets in the Version field. New values shall be assigned
by the IANA using the "IETF Review" policy defined in [RFC5226]
(Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA
Considerations Section in RFCs," May 2008.). Assigned numbers are to be
increased by 1, up to a maximum value of "50". Protocol Version numbers
of "51" through "99" are vendor-specific; values in this range are not
to be assigned by the IANA.
IANA is requested to register the Protocol Version values shown below.

```
VALUE                    PROTOCOL VERSION
-----                    ----------------
00                       Reserved
01                       Defined in RFC yyyy
```

The second registry that IANA is requested to create is entitled
"syslog-sign hash algorithm values". It is for the values of the Hash
Algorithm subfield. The Hash Algorithm subfield constitutes the third
octet in the Version field Signature Blocks and Certificate Blocks. New
values shall be assigned by the IANA using the "IETF Consensus" policy
defined in [RFC5226] (Narten, T. and H. Alvestrand, "Guidelines for
Writing an IANA Considerations Section in RFCs," May 2008.). Assigned
values are to be increased sequentially, first up to a maximum value of
"9", then from "a" to "z", then from "A" to "Z". The values are
registered relative to the Protocol Version. This means that the same
Hash Algorithm value can be reserved for different Protocol Versions,
possibly referring to a different hash algorithm each time. This makes
it possible to deal with future scenarios in which the single octet
representation becomes a limitation, as more Hash Algorithms can be
supported by defining additional Protocol Versions that implementations
might support concurrently.
IANA is requested to register the Hash Algorithm values shown below.

```
VALUE      PROTOCOL VERSION      HASH ALGORITHM
-----      ----------------      --------------
0          01                    Reserved
1          01                    SHA1
2          01                    SHA256
```

The third registry that IANA is requested to create is entitled
"syslog-sign signature scheme values". It is for the values of the
Signature Scheme subfield. The Signature Scheme subfield constitutes
the fourth octet in the Version field of Signature Blocks and
Certificate Blocks. New values shall be assigned by the IANA using the
"IETF Consensus" policy defined in [RFC5226] (Narten, T. and H.
Alvestrand, "Guidelines for Writing an IANA Considerations Section in
RFCs," May 2008.). Assigned values are to be increased by 1, up to a
maximum value of "9". This means that the same Signature Scheme value
can be reserved for different Protocol Versions, possibly in each case
referring to a different Signature Scheme each time. This makes it
possible to deal with future scenarios in which the single octet
representation becomes a limitation, as more Signature Schemes can be
supported by defining additional Protocol Versions that implementations
might support concurrently.
IANA is requested to register the Signature Scheme values shown below.

```
        VALUE     PROTOCOL VERSION     SIGNATURE SCHEME
        -----     ----------------     ----------------
        0         01                   Reserved
        1         01                   OpenPGP DSA
```

9.3. SG Field

IANA is requested to create a registry entitled "syslog-sign sg field
values". It is for values of the SG Field as defined in Section 4.2.3
(Signature Group and Signature Priority). New values shall be assigned
by the IANA using the "IETF Consensus" policy defined in [RFC5226]
(Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA
Considerations Section in RFCs," May 2008.). Assigned values are to be
incremented by 1, up to a maximum value of "7". Values "8" and "9"
shall be left as vendor specific and shall not be assigned by the IANA.
IANA is requested to register the SG Field values shown below.

```
        VALUE     MEANING
        -----     -------
        0         per RFC yyyy
        1         per RFC yyyy
        2         per RFC yyyy
        3         per RFC yyyy
```

9.4. Key Blob Type

IANA is requested to create a registry entitled "syslog-sign key blob
type values". It is to register one-character identifiers for the key
blob type, per Section 5.2 (Payload Block). New values shall be
assigned by the IANA using the "IETF Consensus" policy defined in
[RFC5226] (Narten, T. and H. Alvestrand, "Guidelines for Writing an
IANA Considerations Section in RFCs," May 2008.). Uppercase letters may
be assigned as values. Lowercase letters are left as vendor specific
and shall not be assigned by the IANA.
IANA is requested to register the key blob type values shown below.

```
          VALUE    KEY BLOB TYPE
          -----    ------------
          'C'      a PKIX certificate
          'P'      an OpenPGP certificate
          'K'      the public key whose corresponding private key is
                   used to sign the messages
          'N'      no key information sent, key is pre-distributed
          'U'      installation-specific key exchange information
```

10. Working Group

The working group can be contacted via the mailing list:

        syslog@ietf.org

The current Chairs of the Working Group can be contacted at:

        Chris Lonvick
        Cisco Systems
        Email: clonvick@cisco.com

        David Harrington
        Huawei Technologies (USA)
        Email: ietfdbh@comcast.net
                dharrington@huawei.com
        Tel: +1-603-436-8634

11. Acknowledgements

The authors wish to thank Alex Brown, Chris Calabrese, Steve Chang,
Pasi Eronen, Carson Gaspar, Rainer Gerhards, Drew Gross, David
Harrington, Chris Lonvick, Albert Mietus, Darrin New, Marshall Rose,
Andrew Ross, Martin Schuette, Holt Sorenson, Rodney Thayer, and the
many Counterpane Internet Security engineering and operations people
who commented on various versions of this proposal.

12. References

12.1. Normative References

[FIPS. 186-2.2000]	National Institute of Standards and Technology, "Digital Signature Standard," FIPS PUB 186-2, January 2000.
[FIPS. 180-2.2002]	National Institute of Standards and Technology, "Secure Hash Standard," FIPS PUB 180-2, August 2002.
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC4648]	Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," RFC 4648, October 2006 (TXT).
[RFC4880]	Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format," RFC 4880, November 2007.
[RFC5226]	Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs," BCP 26, RFC 5226, May 2008.
[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, May 2008.
[RFC5424]	Gerhards, R., "The syslog Protocol," RFC 5424, March 2009.
[RFC5425]	Miao, F., Yuzhi, M., and J. Salowey, "TLS Transport Mapping for syslog," RFC 5425, March 2009.
[RFC5426]	Okmianski, A., "Transmission of syslog Messages over UDP," RFC 5426, March 2009.

12.2. Informative References

[NIST800.90]	National Institute of Standards and Technology, "NIST Special Publication 800-90: Recommendation for Random Number Generation using Deterministic Random Bit Generators," June 2006.
[RFC3339]	Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps," RFC 3339, July 2002 (TXT).
[RFC3414]	Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)," RFC 3414, December 2002 (TXT).
[RFC4086]	Eastlake, D., Schiller, J., and S. Crocker, "Randomness Recommendations for Security," RFC 4086, June 2005 (TXT).

Authors' Addresses

	John Kelsey
	NIST
Email:	john.kelsey@nist.gov
	Jon Callas
	PGP Corporation
Email:	jon@callas.org
	Alexander Clemm
	Cisco Systems
Email:	alex@cisco.com