

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 24, 2014

Z. Cao
China Mobile
X. He
Hitachi (China) Research and Development Corporat
M. Kovatsch
ETH Zurich
H. Tian
China Academy of Telecommunication Research
C. Gomez
Universitat Politecnica de Catalunya/i2CAT
October 21, 2013

Energy Efficient Implementation of IETF Constrained Protocol Suite
draft-hex-lwig-energy-efficient-02

Abstract

This document summarizes the problems and current practices of energy efficient protocol implementation on constrained devices, mostly about how to make the protocols within IETF scope behave energy friendly. This document also summarizes the impact of link layer protocol power saving behaviors to the upper layer protocols, so that they can coordinately make the system energy efficient.

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 April 24, 2014.

Copyright Notice

Copyright (c) 2013 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	2
1.1. Conventions used in this document	3
1.2. Terminology	3
2. Overview	3
3. MAC and Radio Duty Cycling	4
3.1. Power Save Services Provided by IEEE 802.11v	5
3.2. Power Save Services Provided by Bluetooth Low Energy . .	6
3.3. Power Save Services in IEEE 802.15.4	7
4. IP Adaptation and Transport Layer	7
5. Routing Protocols	8
6. Application Layer	8
7. Cross Layer Optimization	9
8. Summary	9
9. Acknowledgments	10
10. IANA Considerations	10
11. Security Considerations	10
12. References	10
12.1. Normative References	10
12.2. Informative References	11
Authors' Addresses	12

1. Introduction

In many scenarios, the network systems comprises many battery-powered or energy-harvesting devices. For example, in an environmental monitoring system or a temperature and humidity monitoring system in the data center, there are no always-on and handy sustained power supplies for the large number of small devices. In such deployment environments, it is necessary to optimize the energy consumption of the entire system, including computing, application layer behavior, and lower layer communication.

Various research efforts have been spent on this "energy efficiency" problem. Most of this research has focused on how to optimize the system's power consumption regarding a certain deployment scenario or how could an existing network function such as routing or security be

more energy-efficient. Only few efforts were spent on energy-efficient designs for IETF protocols and standardized network stacks for such constrained devices [I-D.kovatsch-lwig-class1-coap].

The IETF has developed a suite of Internet protocols suitable for such small devices, including 6LoWPAN ([RFC6282],[RFC6775],[RFC4944]), RPL[RFC6550], and CoAP[I-D.ietf-core-coap]. This document tries to summarize the design considerations of making the IETF protocol suite as energy-efficient as possible. While this document does not provide detailed and systematic solutions to the energy efficiency problem, it summarizes the design efforts and analyzes the design space of this problem.

After reviewing the energy-efficient design of each layer, an overall conclusion is summarized. Though the lower layer communication optimization is the key part of energy efficient design, the protocol design at the network and application layers is also important to make the device battery-friendly.

1.1. Conventions used in this document

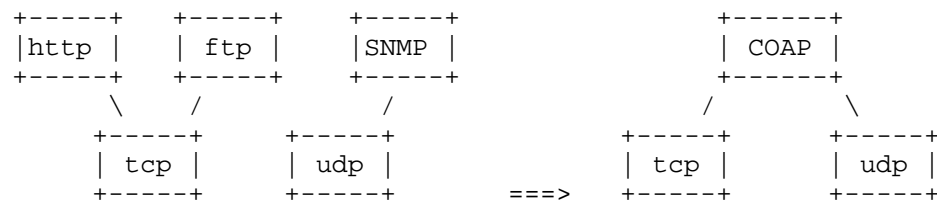
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]

1.2. Terminology

The terminologies used in this document can be referred to [I-D.ietf-lwig-terminology].

2. Overview

The IETF has developed multiple protocols to enable end-to-end IP communication between constrained nodes and fully capable nodes. This work has witnessed the evolution of the traditional Internet protocol stack to a light-weight Internet protocol stack. As show in Figure 1 below, the IETF has developed CoAP as the application layer and 6LoWPAN as the adaption layer to run IPv6 over IEEE 802.15.4 and Bluetooth Low-Energy, with the support of routing by RPL and efficient neighbor discovery by 6LoWPAN-ND.



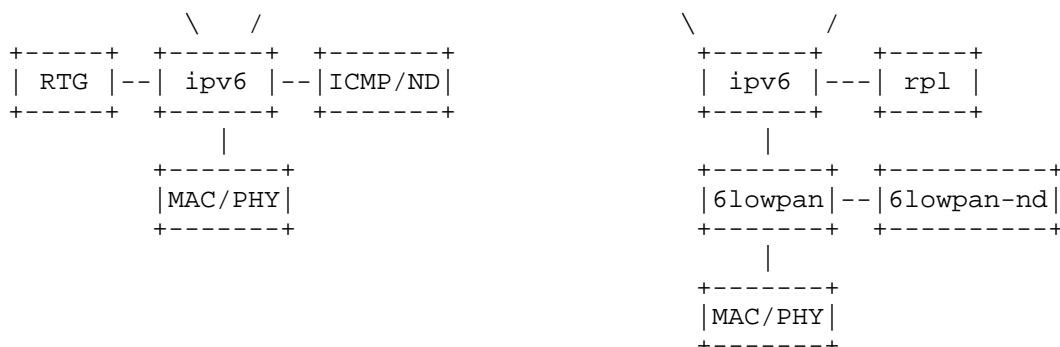


Figure 1: Traditional and Lightweight Internet Protocol Stack

There are comprehensive measurements of wireless communication [Powertrace]. Below we list the energy consumption profile of the most common atom operations on a prevalent sensor node platform. The measurement was based on the Tmote Sky with ContikiMAC as the radio duty cycling algorithm. From the measurement, we can see that optimized transmissions and reception consume almost the same amount of energy. For IEEE 802.15.4 and UWB radios, transmitting is actually even cheaper than receiving. Only for broadcast and non-synchronized communication transmissions become costly in terms of energy because they need to flood the medium for a long time.

Activity	Energy (uJ)
Broadcast reception	178
Unicast reception	222
Broadcast transmission	1790
Non-synchronized unicast transmission	1090
Synchronized unicast transmission	120
Unicast TX to awake receiver	96

Figure 2: Power consumption of atom operations on the Tmote Sky with ContikiMAC

3. MAC and Radio Duty Cycling

In low-power wireless networks, communication and power consumption are intertwined. The communication device is typically the most power-consuming component, but merely refraining from transmissions is not enough to attain a low power consumption: the radio consumes as much power in listen mode as when actively transmitting, as show in Figure 2 . To reduce power consumption, the radio must be switched completely off -- duty-cycled -- as much as possible. ContikiMAC is a very typical Radio Duty Cycling (RDC) protocol [ContikiMAC].

From the perspective of MAC&RDC, all upper layer protocols, such as routing, RESTful communication, adaptation, and management flows, are all applications. Since the duty cycling algorithm is the key to energy-efficiency of the wireless medium, it synchronizes the TX/RX request from the higher layer.

The MAC&RDC are not in the scope of the IETF, yet lower layer designers and chipset manufactures take great care of the problem. For the IETF protocol designers, however, it is good to know the behaviors of lower layers so that the designed protocols can work perfectly with them.

Once again, the IETF protocols we are going to talk about in the following sections are the customers of the lower layer. If they want to get better service in a cooperative way, they should be considerate and understand each other.

3.1. Power Save Services Provided by IEEE 802.11v

IEEE 802.11v [IEEE80211v] defines mechanisms and services for power save of stations/nodes that include flexible multicast service (FMS), proxy ARP advertisement, extended sleep modes, traffic filtering. It would be useful if upper layer protocols knows such capabilities provided by the lower layer, so that they can coordinate with each other.

These services include:

Proxy ARP: The Proxy ARP capability enables an Access Point (AP) to indicate that the non-AP station (STA) will not receive ARP frames. The Proxy ARP capability enables the non-AP STA to remain in power-save for longer periods of time.

Basic Service Set (BSS) Max Idle Period management enables an AP to indicate a time period during which the AP does not disassociate a STA due to non-receipt of frames from the STA. This supports improved STA power saving and AP resource management.

FMS: A service in which a non-access point (non-AP) station (STA) can request a multicast delivery interval longer than the delivery traffic indication message (DTIM) interval for the purposes of lengthening the period of time a STA may be in a power save state.

Traffic Filtering Service (TFS): A service provided by an access point (AP) to a non-AP station (STA) that can reduce the number of frames sent to the non-AP STA by not forwarding individually addressed frames addressed to the non-AP STA that do not match traffic filters specified by the non-AP STA.

Using the above services provided by the lower layer, the constrained nodes can achieve either client initiated power save (via TFS) or network assisted power save (Proxy-ARP, BSS Max Idle Period and FMS).

Upper layer protocols would better synchronize with the parameters such as FMS interval and BSS MAX Idle Period, so that the wireless transmissions are not triggered periodically.

3.2. Power Save Services Provided by Bluetooth Low Energy

Bluetooth Low Energy (BT-LE) is a wireless low-power communications technology that is the hallmark component of the Bluetooth 4.0 specification. BT-LE has been designed for the goal of ultra-low-power consumption. Currently, it is possible to run IPv6 over BT-LE networks by using a 6LoWPAN variant adapted to BT-LE [I-D.ietf-6lowpan-btle].

BT-LE networks comprise a master and one or more slaves which are connected to the master. The BT-LE master is assumed to be a relatively powerful device, whereas a slave is typically a constrained device (e.g. a class 1 device).

Medium access in BT-LE is based on a TDMA scheme which is coordinated by the master. This device determines the start of connection events, in which communication between the master and a slave takes place. At the beginning of a connection event, the master sends a poll message, which may encapsulate data, to the slave. The latter must send a response, which may also contain data. The master and the slave may continue exchanging data until the end of the connection event. The next opportunity for communication between the master and the slave will be in the next connection event scheduled for the slave.

The time between consecutive connection events is defined by the connInterval parameter, which may range between 7.5 ms and 4 s. The slave may remain in sleep mode since the end of its last connection event until the beginning of its next connection event. Therefore,

BT-LE is duty-cycled by nature. Furthermore, after having replied to the master, a slave is not required to listen to the master (and thus may keep the radio in sleep mode) for `connSlaveLatency` consecutive connection events. `connSlaveLatency` is an integer parameter between 0 and 499 which should not cause link inactivity for more than `connSupervisionTimeout` time. The `connSupervisionTimeout` parameter is in the range between 100 ms and 32 s.

Upper layer protocols should take into account the medium access and duty-cycling behavior of BT-LE. In particular, `connInterval`, `connSlaveLatency` and `connSupervisionTimeout` determine the time between two consecutive connection events for a given slave. The upper layer packet generation pattern and rate should be consistent with the settings of the aforementioned parameters (and vice versa).

3.3. Power Save Services in IEEE 802.15.4

To be added.

4. IP Adaptation and Transport Layer

6LoWPAN is the adaption layer to run IPv6 over IEEE 802.15.4 MAC&PHY. It was born to fill the gap that the IPv6 layer does not support fragmentation and assembly of <1280-byte packets while IEEE 802.15.4 only supports a MTU of 127 bytes.

IPv6 is the basis for the higher layer protocols, including both TCP/UDP transport and applications. So they are quite ignorant of the lower layers, and are almost neutral to the energy-efficiency problem.

What the network stack can optimize is to save the computing power. For example the Contiki implementation has multiple cross layer optimizations for buffers and energy management, e.g., the computing and validation of UDP/TCP checksums without the need of reading IP headers from a different layer. These optimizations are software implementation techniques, and out of the scope of IETF and the LWIG working group.

The 6LoWPAN contributes to the energy-efficiency problem in two ways. First of all, it swaps computing with communication. 6LoWPAN applies compression of the IPv6 header. This means less amount of data will be handled by the lower layer, but both the sender and receiver should spend more computing power on the compression and decompression of the packets over the air. Secondly, the 6LoWPAN working group developed the energy-efficient Neighbor Discovery called 6LoWPAN-ND, which is an energy efficient replacement of the IPv6 ND in constrained environments. IPv6 Neighbor Discovery was not

designed for non-transitive wireless links, as its heavy use of multicast makes it inefficient and sometimes impractical in a low-power and lossy network. 6LoWPAN-ND describes simple optimizations to IPv6 Neighbor Discovery, its addressing mechanisms, and duplicate address detection for Low-power Wireless Personal Area Networks and similar networks. However, 6LoWPAN ND does not modify Neighbor Unreachability Detection (NUD) timeouts, which are very short (by default three transmissions spaced one second apart). NUD timeout settings should be tuned taking into account the latency that may be introduced by duty-cycled mechanisms at the link layer, or alternative, less impatient NUD algorithms should be considered [I-D.ietf-6man-impatient-nud].

5. Routing Protocols

The routing protocol designed by the IETF for constrained environments is called RPL [RFC6550]. As a routing protocol, RPL has to exchange messages periodically and keep routing states for each destination. RPL is optimized for the many-to-one communication pattern, where network nodes primarily send data towards the border router, but has provisions for any-to-any routing as well.

The authors of the Powertrace tool [Powertrace] studied the power profile of RPL. It divides the routing protocol into control and data traffic. The control channel uses ICMP messages to establish and maintain the routing states. The data channel is any application that uses RPL for routing packets. The study has shown that the power consumption of the control traffic goes down over time and data traffic stays relatively constant. The study also reflects that the routing protocol should keep the control traffic as low as possible to make it energy-friendly. The amount of RPL control traffic can be tuned by setting the Trickle algorithm parameters (i.e. Imin, Imax and k) to adequate values. However, there exists a trade-off between energy consumption and other performance parameters such as network convergence time and robustness.

Todo: more discussion of energy efficient routing.

6. Application Layer

CoAP [I-D.ietf-core-coap] was designed as a RESTful application protocol, connecting the services of smart devices to the World Wide Web. CoAP is not a chatty protocol, it provides basic communication services such as service discovery and GET/POST/PUT/DELETE methods with a binary header.

The energy-efficient design is implicitly included in the CoAP protocol design. To reduce regular and frequent queries of the

resources, CoAP provides an observe mode, in which the requester registers its interest of a certain resource and the responder will report the value whenever it was updated. This reduces the request response roundtrip while keeping information exchange a ubiquitous service.

CoAP offers mechanisms for reliable communication between two CoAP endpoints. A CoAP message may be signaled as a confirmable (CON) message, and an acknowledgment (ACK) is issued by the receiver if the CON message is correctly received. The sender starts a Retransmission TimeOut (RTO) for every CON message sent. The initial RTO value is chosen randomly between 2 and 3 s. If an RTO expires, the new RTO value is doubled (unless a limit on the number of retransmissions has been reached). Since duty-cycling at the link layer may lead to large latencies (i.e. even greater than the initial RTO value), CoAP RTO parameters should be tuned accordingly in order to avoid spurious RTOs which would unnecessarily waste node energy and other resources.

7. Cross Layer Optimization

The cross layer optimization is a technique used in many scenarios. There are some technologies for power efficient optimization via PHY to Routing cross layer design [Cross-layer-Optimization]. In this research, cross-layer optimization frameworks have been developed to minimize the total power consumption or to maximize the utility-power trade-off using cooperative diversity.

Also a cross-layer design in multihop wireless networks is proposed for congestion control, routing and scheduling - in transport, network and link layers into a coherent framework [Cross-layer-design]. This method and thinking could be applied to the implementation of energy effective cross layer design.

Todo: more discussion of Cross layer issues.

8. Summary

We find a summary section necessary although most IETF documents do not contain it. The points we would like to summarize are as follows.

- a. All Internet protocols, which are in the scope of the IETF, are customers of the lower layers (PHY, MAC, and Duty-cycling). In order to get a better service, the designers of higher layers should know them better.

- b. The IETF has developed multiple protocols for constrained networked devices. A lot of implicit energy efficient design principles have been used in these protocols.
- c. The power trace analysis of different protocol operations showed that for radio-duty-cycled networks broadcasts should be avoided. Saving unnecessary states maintenance is also an effective method to be energy-friendly.

9. Acknowledgments

Carles Gomez has been supported by Ministerio de Economia y Competitividad and FEDER through project TEC2012-32531.

10. IANA Considerations

This document has no IANA requests.

11. Security Considerations

This document discusses the energy efficient protocol design, and does not incur any changes or challenges on security issues besides what the protocol specifications have analyzed.

12. References

12.1. Normative References

[Announcementlayer]

Dunkels, A., "The Announcement Layer: Beacon Coordination for the Sensornet Stack. In Proceedings of EWSN 2011", .

[ContikiMAC]

Dunkels, A., "The ContikiMAC Radio Duty Cycling Protocol, SICS Technical Report T2011:13", December 2011.

[Cross-layer-Optimization]

Le, . and . Hossain, "Cross-Layer Optimization Frameworks for Multihop Wireless Networks Using Cooperative Diversity", July 2008.

[Cross-layer-design]

Chen, ., Low, ., and . Doyle, "Cross-layer design in multihop wireless networks", 2011.

[I-D.ietf-6lowpan-btle]

Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "Transmission of IPv6 Packets

over BLUETOOTH Low Energy", draft-ietf-6lowpan-btle-12 (work in progress), February 2013.

[I-D.ietf-6man-impatient-nud]

Gashinsky, I. and E. Nordmark, "Neighbor Unreachability Detection is too impatient", draft-ietf-6man-impatient-nud-06 (work in progress), April 2013.

[I-D.ietf-core-coap]

Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.

[I-D.ietf-lwig-terminology]

Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-05 (work in progress), July 2013.

[I-D.kovatsch-lwig-class1-coap]

Kovatsch, M., "Implementing CoAP for Class 1 Devices", draft-kovatsch-lwig-class1-coap-00 (work in progress), October 2012.

[IEEE80211v]

IEEE, ., "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 8: IEEE 802.11 Wireless Network Management.", February 2012.

[Powertrace]

Dunkels, ., Eriksson, ., Finne, ., and . Tsiftes, "Powertrace: Network-level Power Profiling for Low-power Wireless Networks", March 2011.

12.2. Informative References

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

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.

- [RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

Authors' Addresses

Zhen Cao (Ed.)
China Mobile
Xuanwumenxi Ave. No.32
Beijing 100871
P.R.China

Email: zehn.cao@gmail.com, caozhen@chinamobile.com

Xuan He
Hitachi (China) Research and Development Corporation
301, Tower C North, Raycom, 2 Kexuyuan Nanlu, Haidian District
Beijing 100190
P.R.China

Email: xhe@hitachi.cn

Matthias Kovatsch
ETH Zurich
Universitaetstrasse 6
Zurich, CH-8092
Switzerland

Email: kovatsch@inf.ethz.ch

Hui Tian
China Academy of Telecommunication Research
Huayuanbeilu No.52
Beijing, Haidian District 100191
China

Email: tianhui@mail.ritt.com.cn

Carles Gomez
Universitat Politecnica de Catalunya/i2CAT
C/Esteve Terradas, 7
Castelldefels 08860
Spain

Email: carlesgo@entel.upc.edu

Light-Weight Implementation Guidance (lwig)
Internet-Draft
Intended status: Informational
Expires: May 26, 2016

T. Kivinen
INSIDE Secure
November 23, 2015

Minimal IKEv2 Initiator Implementation
draft-ietf-lwig-ikev2-minimal-05.txt

Abstract

This document describes a minimal initiator version of the Internet Key Exchange version 2 (IKEv2) protocol for constrained nodes. IKEv2 is a component of IPsec used for performing mutual authentication and establishing and maintaining Security Associations (SAs). IKEv2 includes several optional features, which are not needed in minimal implementations. This document describes what is required from the minimal implementation, and also describes various optimizations which can be done. The protocol described here is interoperable with a full IKEv2 implementation using shared secret authentication (IKEv2 does not require the use of certificate authentication). This minimal initiator implementation can only talk to a full IKEv2 implementation acting as responder, thus two minimal initiator implementations cannot talk to each other.

This document does not update or modify RFC 7296, but provides more compact description of the minimal version of the protocol. If this document and RFC 7296 conflicts then RFC 7296 is the authoritative description.

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 26, 2016.

Copyright Notice

Copyright (c) 2015 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Use Cases	4
2. Exchanges	5
2.1. Initial Exchange	5
2.2. Other Exchanges	11
2.3. Generating Keying Material	11
3. Conformance Requirements	12
4. Implementation Status	13
5. Security Considerations	13
6. IANA Considerations	13
7. Acknowledgements	13
8. References	13
8.1. Normative References	13
8.2. Informative References	14
Appendix A. Header and Payload Formats	14
A.1. The IKE Header	15
A.2. Generic Payload Header	17
A.3. Security Association Payload	18
A.3.1. Proposal Substructure	20

A.3.2.	Transform Substructure	21
A.3.3.	Valid Transform Types by Protocol	23
A.3.4.	Transform Attributes	23
A.4.	Key Exchange Payload	24
A.5.	Identification Payloads	25
A.6.	Certificate Payload	26
A.7.	Certificate Request Payload	27
A.8.	Authentication Payload	28
A.9.	Nonce Payload	28
A.10.	Notify Payload	29
A.10.1.	Notify Message Types	30
A.11.	Traffic Selector Payload	31
A.11.1.	Traffic Selector	33
A.12.	Encrypted Payload	34
Appendix B.	Useful Optional Features	36
B.1.	IKE SA Delete Notification	36
B.2.	Raw Public Keys	37
Author's Address	38

1. Introduction

The Internet Protocol Suite is increasingly used on small devices with severe constraints on power, memory, and processing resources. This document describes a minimal IKEv2 implementation designed for use on such constrained nodes that is interoperable with Internet Key Exchange Protocol Version 2 (IKEv2) [RFC7296].

A minimal IKEv2 implementation only supports the initiator end of the protocol. It only supports the initial IKE_SA_INIT and IKE_AUTH exchanges and does not initiate any other exchanges. It also replies with empty (or error) message to all incoming requests.

This means that most of the optional features of IKEv2 are left out: NAT Traversal, IKE SA rekey, Child SA rekey, Multiple Child SAs, Deleting Child / IKE SAs, Configuration payloads, EAP authentication, COOKIES etc.

Some optimizations can be done because of the limited set of supported features, and this text should not be considered for generic IKEv2 implementations (for example Message IDs can be done as specified because minimal implementation is only sending out IKE_SA_INIT and IKE_AUTH request, and do not send any other request).

This document is intended to be stand-alone, meaning everything needed to implement IKEv2 is copied here except the description of the cryptographic algorithms. The IKEv2 specification has lots of background information and rationale which has been omitted from this document.

Numerous additional numeric values from IANA registries have been omitted from this document, only those which are of interest for a minimal implementation are listed in this document.

The main body of this document describes how to use the shared secret authentication in IKEv2, as it is easiest to implement. In some cases that is not enough and Appendix B.2 describes how to use Raw Public keys instead of shared secret authentication.

For more information check the full IKEv2 specification in RFC 7296 [RFC7296] and [IKEV2IANA].

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]. The term "Constrained Node" is defined in the Terminology for Constrained-Node Networks document [RFC7228].

1.1. Use Cases

One use case for this kind of minimal implementation is in small devices doing machine-to-machine communication. In such environments the node initiating connections can be very small and the other end of the communication channel is some kind of larger device.

An example of the small initiating node could be a remote garage door opener device, i.e., a device having buttons which open and close a garage door, and which connects to the home area network server over wireless link.

Another example of such a device is some kind of sensor device, for example a room temperature sensor, which sends periodic temperature data to some centralized node.

Those devices are usually sleeping for a long time, and only wake up because of user interaction or periodically. The data transfer is always initiated from that sleeping node when they wake up and after they send packets there might be ACKs or other packets coming back before they go back to sleep. If some data needs to be transferred from a server node to the small device, it can be implemented by polling, i.e. the small node periodically polls for the server to see if it for example has some configuration changes or similar. While the device is sleeping it will not maintain the IKEv2 SA. That is, it will always create the IKEv2 SA again when it wakes up. This means there is no need to do liveness checks for the server, as after the device wakes up again the minimal implementation will start from the beginning again.

2. Exchanges

2.1. Initial Exchange

All IKEv2 communications consist of pairs of messages: a request and a response. The pair is called an "exchange", and is sometimes called a "request/response pair". Every request requires a response.

For every pair of IKEv2 messages, the initiator is responsible for retransmission in the event of a timeout. The responder **MUST** never retransmit a response unless it receives a retransmission of the request.

IKEv2 is a reliable protocol: the initiator **MUST** retransmit a request until it either receives a corresponding response or deems the IKE SA to have failed. A retransmission from the initiator **MUST** be bitwise identical to the original request. Retransmission times **MUST** increase exponentially.

IKEv2 is run over UDP port 500. All IKEv2 implementations **MUST** be able to send, receive, and process IKEv2 messages that are up to 1280 octets long. An implementation **MUST** accept incoming requests even if the source port is not 500, and **MUST** respond to the address and port from which the request was received.

The minimal implementation of IKEv2 only uses the first two exchanges, called `IKE_SA_INIT` and `IKE_AUTH`. These are used to create the IKE SA and the first Child SA. In addition to those messages, a minimal IKEv2 implementation needs to understand the `CREATE_CHILD_SA` request enough to generate an `CREATE_CHILD_SA` response containing the `NO_ADDITIONAL_SAS` error notify. It needs to understand the `INFORMATIONAL` request enough to generate an empty `INFORMATIONAL` response to it. There is no requirement to be able to respond to any other requests.

All messages following the `IKE_SA_INIT` exchange are cryptographically protected using the cryptographic algorithms and keys negotiated in the `IKE_SA_INIT` exchange.

Every IKEv2 message contains a Message ID as part of its fixed header. This Message ID is used to match up requests and responses, and to identify retransmissions of messages.

Minimal implementations only need to support the role of initiator, so so it typically only sends an `IKE_SA_INIT` request which, when answered, is followed by an `IKE_AUTH`. As those messages have fixed Message IDs (0 and 1) it does not need to keep track of its own Message IDs for outgoing requests after that.

Minimal implementations can also optimize Message ID handling of the incoming requests, as they do not need to protect incoming requests against replays. This is possible because minimal implementations will only return error or empty notification replies to incoming requests. This means that any of those incoming requests do not have any effect on the minimal implementation, thus processing them again does not cause any harm. Because of this a minimal implementation can always answer to request coming in, with the same Message ID than what the request had and then forget the request/response pair immediately. This means there is no need to keep track of Message IDs of the incoming requests.

In the following descriptions, the payloads contained in the message are indicated by the names listed below.

Notation	Payload

AUTH	Authentication
CERTREQ	Certificate Request
D	Delete
HDR	IKE header (not a payload)
IDi	Identification - Initiator
IDr	Identification - Responder
KE	Key Exchange
Ni, Nr	Nonce
N	Notify
SA	Security Association
SK	Encrypted and Authenticated
TSi	Traffic Selector - Initiator
TSr	Traffic Selector - Responder

The initial exchanges are as follows:

Initiator	Responder

HDR(SPIi=xxx, SPIr=0, IKE_SA_INIT, Flags: Initiator, Message ID=0), SAi1, KEi, Ni -->	<-- HDR(SPIi=xxx, SPIr=yyy, IKE_SA_INIT, Flags: Response, Message ID=0), SAr1, KEr, Nr, [CERTREQ]

HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. Each endpoint chooses one of the two SPIs and MUST choose them so as to be unique identifiers of an IKE SA. An SPI value of zero is special: it indicates that the remote SPI value is not yet known by the sender.

Incoming IKEv2 packets are mapped to an IKE SA using only the packet's SPI, not using (for example) the source IP address of the packet.

The SA₁ payload states the cryptographic algorithms the initiator supports for the IKE SA. The KE_i and KE_r payload contain Diffie-Hellman values and N_i and N_r are the nonces. The SA₁ contains the chosen cryptographic suite from initiator's offered choices. A minimal implementation using shared secrets will ignore the CERTREQ payload.

Minimal implementation will most likely support exactly one set of cryptographic algorithms, meaning the SA₁ payload will be static. It needs to check that the SA₁ received matches the proposal it sent.

At this point in the negotiation, each party can generate SKEYSEED, from which all keys are derived for that IKE SA.

$$\text{SKEYSEED} = \text{prf}(\text{Ni} \parallel \text{Nr}, g^{\text{ir}})$$

$$\{\text{SK}_d \parallel \text{SK}_{ai} \parallel \text{SK}_{ar} \parallel \text{SK}_{ei} \parallel \text{SK}_{er} \parallel \text{SK}_{pi} \parallel \text{SK}_{pr}\} \\ = \text{prf}^+ (\text{SKEYSEED}, \text{Ni} \parallel \text{Nr} \parallel \text{SPI}_i \parallel \text{SPI}_r)$$

$$\text{prf}^+ (K, S) = T_1 \parallel T_2 \parallel T_3 \parallel T_4 \parallel \dots$$

where:

$$\begin{aligned} T_1 &= \text{prf} (K, S \parallel 0x01) \\ T_2 &= \text{prf} (K, T_1 \parallel S \parallel 0x02) \\ T_3 &= \text{prf} (K, T_2 \parallel S \parallel 0x03) \\ T_4 &= \text{prf} (K, T_3 \parallel S \parallel 0x04) \\ &\dots \end{aligned}$$

(indicating that the quantities SK_d, SK_{ai}, SK_{ar}, SK_{ei}, SK_{er}, SK_{pi}, and SK_{pr} are taken in order from the generated bits of the prf⁺). g^{ir} is the shared secret from the ephemeral Diffie-Hellman exchange. g^{ir} is represented as a string of octets in big endian order padded with zeros if necessary to make it the length of the modulus. N_i and N_r are the nonces, stripped of any headers.

The SK_d is used for deriving new keys for the Child SAs. The SK_{ai} and SK_{ar} are used as a key to the integrity protection algorithm for authenticating the component messages of subsequent exchanges. The SK_{ei} and SK_{er} are used for encrypting (and of course decrypting) all subsequent exchanges. The SK_{pi} and SK_{pr} are used when generating an AUTH payload. The lengths of SK_d, SK_{pi}, and SK_{pr} MUST be the preferred key length of the PRF agreed upon.

A separate SK_e and SK_a is computed for each direction. The keys used to protect messages from the original initiator are SK_ai and SK_ei. The keys used to protect messages in the other direction are SK_ar and SK_er. The notation SK { ... } indicates that these payloads are encrypted and integrity protected using that direction's SK_e and SK_a.

Initiator	Responder

HDR(SPIi=xxx, SPIr=yyy, IKE_AUTH,	
Flags: Initiator, Message ID=1),	
SK {IDi, AUTH, SAI2, TSi, TSr,	
N(INITIAL_CONTACT)} -->	
	<-- HDR(SPIi=xxx, SPIr=yyy, IKE_AUTH, Flags:
	Response, Message ID=1),
	SK {IDr, AUTH, SAR2, TSi, TSr}

The initiator asserts its identity with the IDi payload, proves knowledge of the secret corresponding to IDi and integrity protects the contents of the first message using the AUTH payload. The responder asserts its identity with the IDr payload, authenticates its identity and protects the integrity of the second message with the AUTH payload.

As minimal implementation usually has only one host where it connects, and that means it has only one shared secret. This means it does not need to care about IDr payload that much. If the other end sends AUTH payload which initiator can verify using the shared secret it has, then it knows the other end is the peer it was configured to talk to.

In the IKE_AUTH request, the initiator sends the SA offer(s) in the SAI2 payload, and the proposed Traffic Selectors for the Child SA in the TSi and TSr payloads. The responder replies with the accepted offer in an SAR2 payload, and with the selected Traffic Selectors. The selected Traffic Selectors may be a subset of what the initiator proposed.

In the minimal implementation both SA payloads and TS payloads are going to be mostly static. The SA payload will have the SPI value used in the Encapsulating Security Payload (ESP), but the algorithms are most likely going to be the one and only supported set. The TS payloads on the initiator end will most likely say from any to any, i.e. full wildcard ranges, or from the local IP to the remote IP. In the wildcard case the responder quite often narrows the range down to the one IP address pair. Using a single IP address pair as the Traffic Selectors when sending the IKE_AUTH request will simplify

processing as the responder will either accept the IP address pair or return an error. If wildcard ranges are used, there is a possibility that the responder will narrow the Traffic Selector range to range that is not acceptable by the initiator.

The IKE_AUTH (and IKE_SA_INIT) responses may contain multiple status notification payloads which can be ignored by minimal implementations. There can also be Vendor ID, Certificate, Certificate Request or Configuration payloads, but any payload unknown to minimal implementations can simply be skipped over (response messages cannot have critical unsupported payloads).

The exchange above includes N(INITIAL_CONTACT) notification in the request as that is quite commonly sent by a minimal implementation. It indicates to the other end that the initiator does not have any other IKE SAs between it and the responder, and if there is any left from previous runs those can be deleted by the responder. As minimal implementations delete IKE SAs without sending IKE SA delete requests, this will help the responder to clean up leftover state.

When using shared secret authentication, the peers are authenticated by having each calculating a MAC over a block of data:

For the initiator:

```
AUTH = prf( prf(Shared Secret, "Key Pad for IKEv2"),
             <InitiatorSignedOctets>)
```

For the responder:

```
AUTH = prf( prf(Shared Secret, "Key Pad for IKEv2"),
             <ResponderSignedOctets>)
```

The string "Key Pad for IKEv2" is 17 ASCII characters without null termination. The implementation can precalculate the inner prf and only store the output of it. This is possible because a minimal IKEv2 implementation usually only supports one PRF.

In following calculations, IDi' and IDr' are the entire ID payloads excluding the fixed header and the Ni and Nr are only the value, not the payload containing it. Note that neither the nonce Ni/Nr nor the value prf(SK_pr, IDr')/prf(SK_pi, IDi') are transmitted.

The initiator signs the first message (IKE_SA_INIT request), starting with the first octet of the first SPI in the header and ending with the last octet of the last payload in that first message. Appended to this (for purposes of computing the signature) are the responder's nonce Nr, and the value prf(SK_pi, IDi').

For the responder, the octets to be signed start with the first octet of the first SPI in the header of the second message (IKE_SA_INIT

response) and end with the last octet of the last payload in that second message. Appended to this are the initiator's nonce N_i , and the value $\text{prf}(\text{SK}_{\text{pr}}, \text{IDr}')$.

The initiator's signed octets can be described as:

```
InitiatorSignedOctets = RealMessage1 | NonceRData | MACedIDForI
RealIKEHDR = SPIi | SPIr | . . . | Length
RealMessage1 = RealIKEHDR | RestOfMessage1
NonceRData = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)
```

The responder's signed octets can be described as:

```
ResponderSignedOctets = RealMessage2 | NonceIDData | MACedIDForR
RealIKEHDR = SPIi | SPIr | . . . | Length
RealMessage2 = RealIKEHDR | RestOfMessage2
NonceIDData = PayloadHeader | NonceIDData
ResponderIDPayload = PayloadHeader | RestOfRespIDPayload
RestOfRespIDPayload = IDType | RESERVED | RespIDData
MACedIDForR = prf(SK_pr, RestOfRespIDPayload)
```

Note that all of the payloads inside the `RestOfMessageX` are included under the signature, including any payload types not listed in this document.

The initiator might also get an unauthenticated response back having a notification payload with an error code inside. As that error code will be unauthenticated and may be faked, there is no need to do anything for those. A minimal implementation can simply ignore those errors, and retransmit its request until it times out and if that happens then the IKE SA (and Child SA) creation failed.

The responder might also reply with an `IKE_AUTH` response packet which does not contain the payloads needed to set up a Child SA (`SAr2`, `TSi` and `TSr`), but instead contain AUTH payload and an error. Minimal implementation that do not support the `CREATE_CHILD_SA` exchange cannot recover from this scenario. It can delete the IKE SA and start over from the beginning (which might fail again if this is a configuration error, or it might succeed if this was temporal failure).

2.2. Other Exchanges

Minimal implementations MUST be able to reply to INFORMATIONAL requests by sending back an empty INFORMATIONAL response:

Minimal implementation	Other end

	<-- HDR(SPIi=xxx, SPIr=yyy, INFORMATIONAL, Flags: none, Message ID=m), SK {...}
HDR(SPIi=xxx, SPIr=yyy, INFORMATIONAL, Flags: Initiator Response, Message ID=m), SK {} -->	

Minimal implementations MUST be able to reply to incoming CREATE_CHILD_SA requests. A typical implementation will reject the CREATE_CHILD_SA exchanges by sending a NO_ADDITIONAL_SAS error notify back:

Minimal implementation	Other end

	<-- HDR(SPIi=xxx, SPIr=yyy, CREATE_CHILD_SA, Flags: none, Message ID=m), SK {...}
HDR(SPIi=xxx, SPIr=yyy, CREATE_CHILD_SA, Flags: Initiator Response, Message ID=m), SK {N(NO_ADDITIONAL_SAS)} -->	

Note that INFORMATIONAL and CREATE_CHILD_SA requests might contain unsupported critical payloads, in which case a compliant implementation MUST ignore the request, and send a response message back having the UNSUPPORTED_CRITICAL_PAYLOAD notification. That notification payload data contains a one-octet payload type of the unsupported critical payload.

2.3. Generating Keying Material

The keying material for the Child SA created by the IKE_AUTH exchange is generated as follows:

KEYMAT = prf+(SK_d, Ni | Nr)

Where Ni and Nr are the nonces from the IKE_SA_INIT exchange.

A single CHILD_SA negotiation may result in multiple Security Associations. ESP and AH SAs exist in pairs (one in each direction), so two SAs are created in a single Child SA negotiation for them. The keying material for each Child SA MUST be taken from the expanded KEYMAT using the following rules:

- o All keys for SAs carrying data from the initiator to the responder are taken before SAs going from the responder to the initiator.
- o If an IPsec protocol requires multiple keys, the order in which they are taken from the SA's keying material needs to be described in the protocol's specification. For ESP and AH, [IPSECARCH] defines the order, namely: the encryption key (if any) MUST be taken from the first bits and the integrity key (if any) MUST be taken from the remaining bits.

Each cryptographic algorithm takes a fixed number of bits of keying material specified as part of the algorithm, or negotiated in SA payloads.

3. Conformance Requirements

For an implementation to be called conforming to RFC 7296 specification, it MUST be possible to configure it to accept the following:

- o Public Key Infrastructure using X.509 (PKIX) Certificates containing and signed by RSA keys of size 1024 or 2048 bits, where the ID passed is any of ID_KEY_ID, ID_FQDN, ID_RFC822_ADDR, or ID_DER_ASN1_DN.
- o Shared key authentication where the ID passed is any of ID_KEY_ID, ID_FQDN, or ID_RFC822_ADDR.
- o Authentication where the responder is authenticated using PKIX Certificates and the initiator is authenticated using shared key authentication.

This document only supports the second bullet, it does not support PKIX certificates at all. As full RFC 7296 responders must also support that shared key authentication, this allows a minimal implementation to be able to interoperate with all RFC 7296 compliant implementations.

PKIX certificates are left out from the minimal implementation as those would add quite a lot of complexity to the implementation. The actual code changes needed in the IKEv2 protocol are small, but the certificate validation code would be more complex than the whole

minimal IKEv2 implementation itself. If public key based authentication is needed for scalability reasons, then raw public keys would probably be the best compromise (see Appendix B.2).

4. Implementation Status

This document describes a minimal implementation written by the author of this document. The minimal implementation supported the base IKE_SA_INIT and IKE_AUTH exchanges, and successfully interoperated with a full IKEv2 server. This minimal implementation was presented in the Interconnecting Smart Objects with Internet Workshop in Prague March 2011 ([Kiv11]). This implementation was written as proof of concept in perl.

There was another proof of concept implementation written in python, which also interoperated with a full IKEv2 server.

Both implementations were written just for demonstration purposes, and included fixed configuration built in to the code, and both also implemented ESP, ICMP and IP layers to the level that was needed to send and receive one ICMP echo packet. Both implementations were about 1000 lines of code excluding cryptographic libraries but including ESP, ICMP and IP layers.

5. Security Considerations

As this implements same protocol as RFC 7296 this means all security considerations from it also apply to this document.

6. IANA Considerations

There is no new IANA considerations in this document.

7. Acknowledgements

Most of the content of this document is copied from the RFC 7296.

8. References

8.1. Normative References

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

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

8.2. Informative References

- [I-D.kivinen-ipsecme-oob-pubkey] Kivinen, T., Wouters, P., and H. Tschofenig, "Generic Raw Public Key Support for IKEv2", draft-kivinen-ipsecme-oob-pubkey-14 (work in progress), October 2015.
- [IKEV2IANA] "Internet Key Exchange Version 2 (IKEv2) Parameters", <<http://www.iana.org>>.
- [Kiv11] Kivinen, T., "IKEv2 and Smart Objects", March 2011, <<https://www.iab.org/wp-content/IAB-uploads/2011/04/Kivinen.pdf>>.
- [MODES] National Institute of Standards and Technology, U.S. Department of Commerce, "Recommendation for Block Cipher Modes of Operation", SP 800-38A, 2001.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<http://www.rfc-editor.org/info/rfc5280>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<http://www.rfc-editor.org/info/rfc7228>>.
- [RFC7619] Smyslov, V. and P. Wouters, "The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7619, DOI 10.17487/RFC7619, August 2015, <<http://www.rfc-editor.org/info/rfc7619>>.

Appendix A. Header and Payload Formats

This appendix describes actual packet payload formats. This is required to make the document self contained. The descriptions are mostly copied from the RFC7296 and more information can be found from there.

Various payload contains RESERVED fields and those MUST be sent as zero and MUST be ignored on receipt.

All multi-octet fields representing integers are laid out in big endian order (also known as "most significant byte first", or "network byte order").

A.1. The IKE Header

Each IKEv2 message begins with the IKE header, denoted HDR in this document. Following the header are one or more IKE payloads each identified by a "Next Payload" field in the preceding payload. Payloads are identified in the order in which they appear in an IKE message by looking in the "Next Payload" field in the IKE header, and subsequently according to the "Next Payload" field in the IKE payload itself until a "Next Payload" field of zero indicates that no payloads follow. If a payload of type "Encrypted" is found, that payload is decrypted and its contents parsed as additional payloads. An Encrypted payload MUST be the last payload in a packet and an Encrypted payload MUST NOT contain another Encrypted payload.

The format of the IKE header is shown in Figure 1.

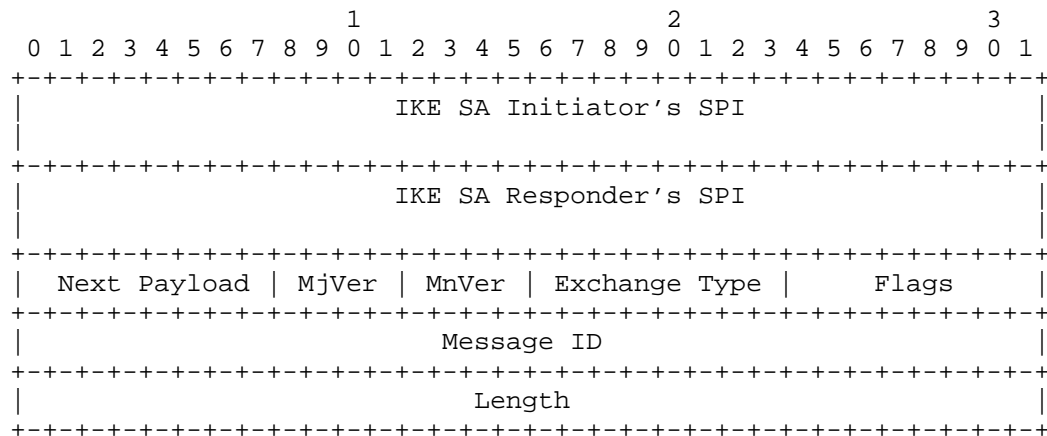


Figure 1: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE Security Association. This value MUST NOT be zero.
- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE Security Association. This value MUST be zero in the first message of an IKE initial exchange.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.
- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the major version to 2 and MUST drop the messages with a higher major version number.
- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the minor version to 0. They MUST ignore the minor version number of received messages.
- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message in an exchange.

Exchange Type	Value
-----	-----
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37

- o Flags (1 octet) - Indicates specific options that are set for the message. Presence of options is indicated by the appropriate bit in the flags field being set. The bits are as follows:

```

+-----+
|X|X|R|V|I|X|X|X|
+-----+

```

In the description below, a bit being 'set' means its value is '1', while 'cleared' means its value is '0'. 'X' bits MUST be cleared when sending and MUST be ignored on receipt.

- * R (Response) - This bit indicates that this message is a response to a message containing the same Message ID. This bit MUST be cleared in all request messages and MUST be set in all responses. An IKEv2 endpoint MUST NOT generate a response to a message that is marked as being a response.
- * V (Version) - This bit indicates that the transmitter is capable of speaking a higher major version number of the protocol than the one indicated in the major version number field. Implementations of IKEv2 MUST clear this bit when sending and MUST ignore it in incoming messages.

- * I (Initiator) - This bit MUST be set in messages sent by the original initiator of the IKE SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient. This bit changes to reflect who initiated the last rekey of the IKE SA.
- o Message ID (4 octets, unsigned integer) - Message identifier used to control retransmission of lost packets and matching of requests and responses. It is essential to the security of the protocol because it is used to prevent message replay attacks.
- o Length (4 octets, unsigned integer) - Length of the total message (header + payloads) in octets.

A.2. Generic Payload Header

Each IKE payload begins with a generic payload header, shown in Figure 2. Figures for each payload below will include the generic payload header, but for brevity, the description of each field will be omitted.

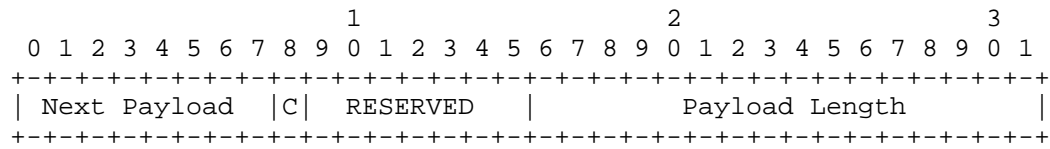


Figure 2: Generic Payload Header

The Generic Payload Header fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0. This field provides a "chaining" capability whereby additional payloads can be added to a message by appending each one to the end of the message and setting the "Next Payload" field of the preceding payload to indicate the new payload's type. An Encrypted payload, which must always be the last payload of a message, is an exception. It contains data structures in the format of additional payloads. In the header of an Encrypted payload, the Next Payload field is set to the payload type of the first contained payload (instead of 0); conversely, the Next Payload field of the last contained payload is set to zero). The payload type values needed for minimal implementations are listed here.

Next Payload Type	Notation	Value

No Next Payload		0
Security Association	SA	33
Key Exchange	KE	34
Identification - Initiator	IDi	35
Identification - Responder	IDr	36
Certificate	CERT	37
Certificate Request	CERTREQ	38
Authentication	AUTH	39
Nonce	Ni, Nr	40
Notify	N	41
Delete	D	42
Traffic Selector - Initiator	TSi	44
Traffic Selector - Responder	TSr	45
Encrypted and Authenticated	SK	46

- o Critical (1 bit) - MUST be set to zero if the sender wants the recipient to skip this payload if it does not understand the payload type code in the Next Payload field of the previous payload. MUST be set to one if the sender wants the recipient to reject this entire message if it does not understand the payload type. MUST be ignored by the recipient if the recipient understands the payload type code. MUST be set to zero for payload types defined in this document. Note that the critical bit applies to the current payload rather than the "next" payload whose type code appears in the first octet.
- o Payload Length (2 octets, unsigned integer) - Length in octets of the current payload, including the generic payload header.

A.3. Security Association Payload

The Security Association payload, denoted SA in this document, is used to negotiate attributes of a Security Association.

An SA payload consists of one or more proposals. Each proposal includes one protocol. Each protocol contains one or more transforms -- each specifying a cryptographic algorithm. Each transform contains zero or more attributes (attributes are needed only if the Transform ID does not completely specify the cryptographic algorithm, currently only attribute is key length attribute for variable length ciphers, meaning there is exactly zero or one attribute).

The responder MUST choose a single suite, which may be any subset of the SA proposal following the rules below.

Each proposal contains one protocol. If a proposal is accepted, the SA response MUST contain the same protocol. Each IPsec protocol proposal contains one or more transforms. Each transform contains a Transform Type. The accepted cryptographic suite MUST contain exactly one transform of each type included in the proposal. For example: if an ESP proposal includes transforms ENCR_3DES, ENCR_AES w/keysize 128, ENCR_AES w/keysize 256, AUTH_HMAC_MD5, and AUTH_HMAC_SHA, the accepted suite MUST contain one of the ENCR_ transforms and one of the AUTH_ transforms. Thus, six combinations are acceptable.

Minimal implementation can create very simple SA proposal, i.e. include one proposal, which contains exactly one transform for each transform type. It is important to only include one Diffie-Hellman group in proposal, so there is no need to do INVALID_KEY_PAYLOAD processing in responses.

When parsing an SA, an implementation MUST check that the total Payload Length is consistent with the payload's internal lengths and counts. Proposals, Transforms, and Attributes each have their own variable-length encodings. They are nested such that the Payload Length of an SA includes the combined contents of the SA, Proposal, Transform, and Attribute information. The length of a Proposal includes the lengths of all Transforms and Attributes it contains. The length of a Transform includes the lengths of all Attributes it contains.

Each Proposal/Protocol structure is followed by one or more transform structures. The number of different transforms is generally determined by the Protocol. AH generally has two transforms: Extended Sequence Numbers (ESNs) and an integrity check algorithm. ESP generally has three: ESN, an encryption algorithm, and an integrity check algorithm. IKEv2 generally has four transforms: a Diffie-Hellman group, an integrity check algorithm, a PRF algorithm, and an encryption algorithm. For each Protocol, the set of permissible transforms is assigned Transform ID numbers, which appear in the header of each transform.

If there are multiple transforms with the same Transform Type, the proposal is an OR of those transforms. If there are multiple transforms with different Transform Types, the proposal is an AND of the different groups.

A given transform MAY have one or more Attributes. Attributes are necessary when the transform can be used in more than one way, as when an encryption algorithm has a variable key size. The transform would specify the algorithm and the attribute would specify the key size. To propose alternate values for an attribute (for example,

multiple key sizes for the AES encryption algorithm), an implementation MUST include multiple transforms with the same Transform Type each with a single Attribute.

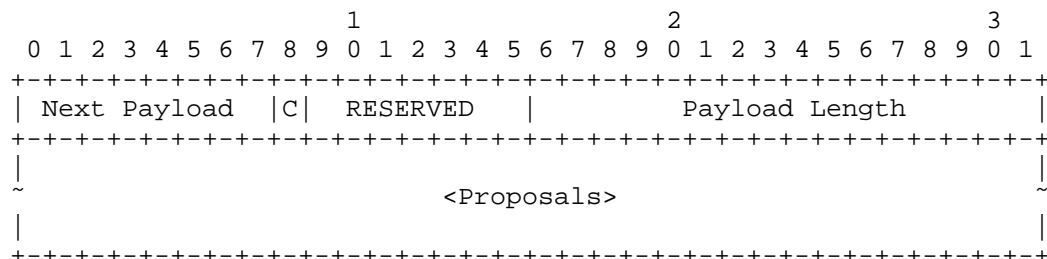


Figure 3: Security Association Payload

- o Proposals (variable) - One or more proposal substructures.

A.3.1. Proposal Substructure

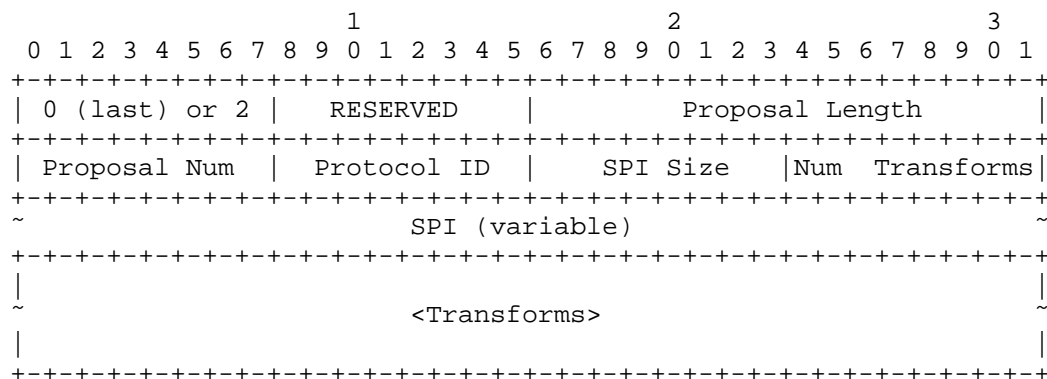


Figure 4: Proposal Substructure

- o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA.
- o Proposal Length (2 octets, unsigned integer) - Length of this proposal, including all transforms and attributes that follow.
- o Proposal Num (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be 1, and subsequent proposals MUST be one more than the previous proposal. When a proposal is accepted, the proposal number in the SA payload MUST match the number on the proposal sent that was accepted.

- o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation.

Protocol	Protocol ID
-----	-----
IKE	1
AH	2
ESP	3

- o SPI Size (1 octet) - For an initial IKE SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).
- o Num Transforms (1 octet) - Specifies the number of transforms in this proposal.
- o SPI (variable) - The sending entity's SPI. When the SPI Size field is zero, this field is not present in the Security Association payload.
- o Transforms (variable) - One or more transform substructures.

A.3.2. Transform Substructure

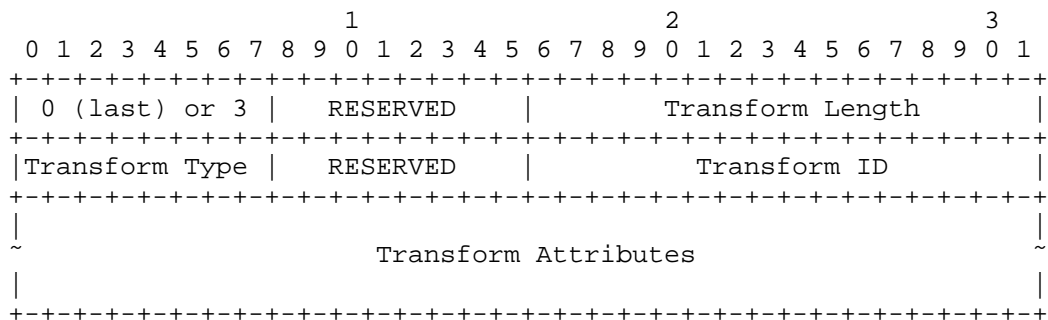


Figure 5: Transform Substructure

- o 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal.
- o Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.
- o Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different

Transform Types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with Transform ID = 0 as one of the options.

- o Transform ID (2 octets) - The specific instance of the Transform Type being proposed.

The relevant Transform Type values are listed below. For more information see [RFC7296].

Description	Trans. Type	Used In
Encryption Algorithm (ENCR)	1	IKE and ESP
Pseudorandom Function (PRF)	2	IKE
Integrity Algorithm (INTEG)	3	IKE, AH, optional in ESP
Diffie-Hellman group (D-H)	4	IKE, optional in AH & ESP
Extended Sequence Numbers (ESN)	5	AH and ESP

For Transform Type 1 (Encryption Algorithm), the relevant Transform IDs are listed below.

Name	Number
ENCR_AES_CBC	12
ENCR_AES-CCM_8	14

For Transform Type 2 (Pseudorandom Function), the relevant Transform IDs are listed below.

Name	Number
PRF_HMAC_SHA1	2

For Transform Type 3 (Integrity Algorithm), relevant Transform IDs are listed below.

Name	Number
AUTH_HMAC_SHA1_96	2
AUTH_AES_XCBC_96	5

For Transform Type 4 (Diffie-Hellman group), relevant Transform IDs are listed below.

Name	Number
-----	-----
1536-bit MODP	5
2048-bit MODP	14

For Transform Type 5 (Extended Sequence Numbers), relevant Transform IDs are listed below.

Name	Number
-----	-----
No Extended Sequence Numbers	0
Extended Sequence Numbers	1

Note that an initiator who supports ESNs will usually include two ESN transforms, with values "0" and "1", in its proposals. A proposal containing a single ESN transform with value "1" means that using normal (non-extended) sequence numbers is not acceptable.

A.3.3. Valid Transform Types by Protocol

The number and type of transforms that accompany an SA payload are dependent on the protocol in the SA itself. An SA payload proposing the establishment of an SA has the following mandatory and optional Transform Types. A compliant implementation MUST understand all mandatory and optional types for each protocol it supports (though it need not accept proposals with unacceptable suites). A proposal MAY omit the optional types if the only value for them it will accept is NONE.

Protocol	Mandatory Types	Optional Types
-----	-----	-----
IKE	ENCR, PRF, INTEG, D-H	
ESP	ENCR, ESN	INTEG, D-H
AH	INTEG, ESN	D-H

A.3.4. Transform Attributes

Transform type 1 (Encryption Algorithm) transforms might include one transform attribute: Key Length.

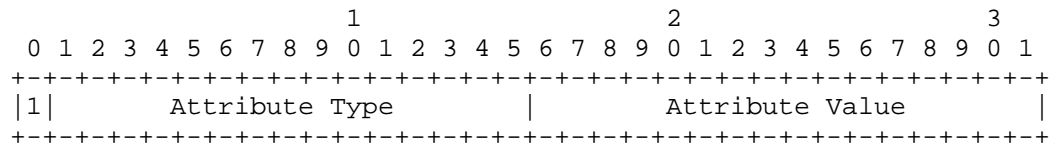


Figure 6: Data Attributes

- o Attribute Type (15 bits) - Unique identifier for each type of attribute (see below).
- o Attribute Value - Value of the attribute associated with the attribute type.

Attribute Type	Value
-----	-----
Key Length (in bits)	14

The Key Length attribute specifies the key length in bits (MUST use network byte order) for certain transforms as follows:

- o The Key Length attribute MUST NOT be used with transforms that use a fixed-length key.
- o Some transforms specify that the Key Length attribute MUST be always included. For example ENCR_AES_CBC.

A.4. Key Exchange Payload

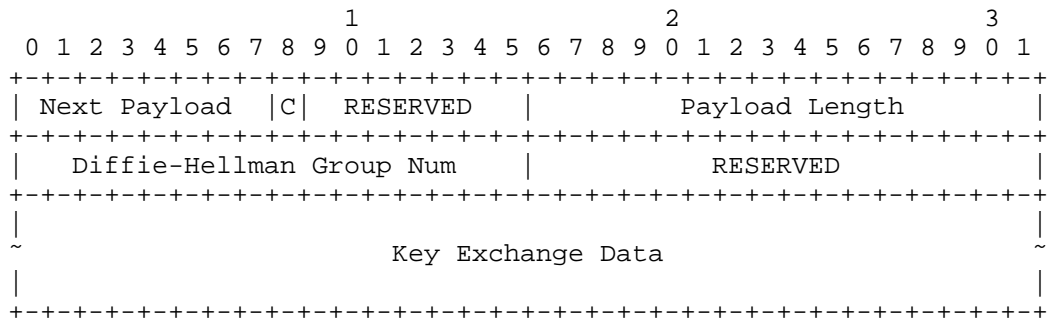


Figure 7: Key Exchange Payload Format

A Key Exchange payload is constructed by copying one's Diffie-Hellman public value into the "Key Exchange Data" portion of the payload. The length of the Diffie-Hellman public value for modular exponentiation group (MODP) groups MUST be equal to the length of the prime modulus over which the exponentiation was performed, prepending zero bits to the value if necessary.

The Diffie-Hellman Group Num identifies the Diffie-Hellman group in which the Key Exchange Data was computed. This Diffie-Hellman Group Num MUST match a Diffie-Hellman group specified in a proposal in the SA payload that is sent in the same message

A.5. Identification Payloads

The Identification payloads, denoted IDi and IDr in this document, allow peers to assert an identity to one another. When using the ID_IPV4_ADDR/ID_IPV6_ADDR identity types in IDi/IDr payloads, IKEv2 does not require this address to match the address in the IP header of IKEv2 packets, or anything in the TSi/TSr payloads. The contents of IDi/IDr are used purely to fetch the policy and authentication data related to the other party. In minimal implementation it might be easiest to always use KEY_ID type. This allows the ID payload to be static. Using IP address has problems in environments where IP addresses are dynamically allocated.

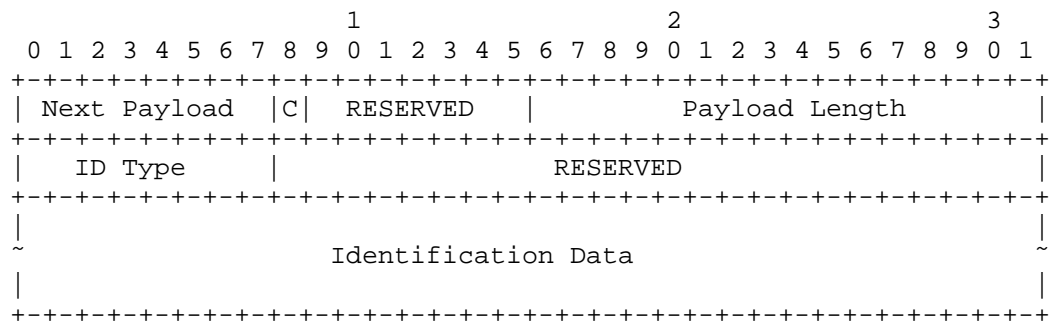


Figure 8: Identification Payload Format

- o ID Type (1 octet) - Specifies the type of Identification being used.
- o Identification Data (variable length) - Value, as indicated by the Identification Type. The length of the Identification Data is computed from the size in the ID payload header.

The following table lists the assigned semantics for the Identification Type field.

ID Type	Value

ID_IPV4_ADDR	1
A single four (4) octet IPv4 address.	
ID_FQDN	2
A fully-qualified domain name string. An example of an ID_FQDN is "example.com". The string MUST NOT contain any terminators (e.g., NULL, CR, etc.). All characters in the ID_FQDN are ASCII; for an "internationalized domain name", the syntax is as defined in [IDNA], for example "xn--tmonesimerkki-bfbb.example.net".	
ID_RFC822_ADDR	3
A fully-qualified RFC 822 email address string. An example of a ID_RFC822_ADDR is "jsmith@example.com". The string MUST NOT contain any terminators. Because of [EAI], implementations would be wise to treat this field as UTF-8 encoded text, not as pure ASCII.	
ID_IPV6_ADDR	5
A single sixteen (16) octet IPv6 address.	
ID_KEY_ID	11
An opaque octet stream that may be used to pass vendor-specific information necessary to do certain proprietary types of identification. Minimal implementation might use this type to send out serial number or similar device specific unique static identification data for the device.	

A.6. Certificate Payload

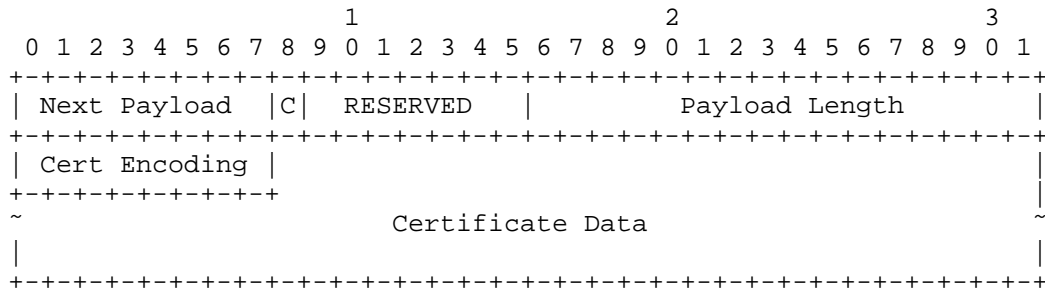


Figure 9: Certificate Payload Format

- o Certificate Encoding (1 octet) - This field indicates the type of certificate or certificate-related information contained in the Certificate Data field.

Certificate Encoding	Value
-----	-----
X.509 Certificate - Signature	4
Raw Public Key	TBD

- o Certificate Data (variable length) - Actual encoding of certificate data. The type of certificate is indicated by the Certificate Encoding field.

The syntax of the types above are:

- o "X.509 Certificate - Signature" contains a DER-encoded X.509 certificate whose public key is used to validate the sender's AUTH payload. Note that with this encoding, if a chain of certificates needs to be sent, multiple CERT payloads are used, only the first of which holds the public key used to validate the sender's AUTH payload.
- o "Raw Public Key" contains a raw public key. In essence the Certificate Payload contains the SubjectPublicKeyInfo part of the PKIX certificate (See Section 4.1.2.7 of [RFC5280]). This is quite simple ASN.1 object which contains mostly static parts before the actual public key values. See [I-D.kivinen-ipsecme-oob-pubkey] for more information.

A.7. Certificate Request Payload

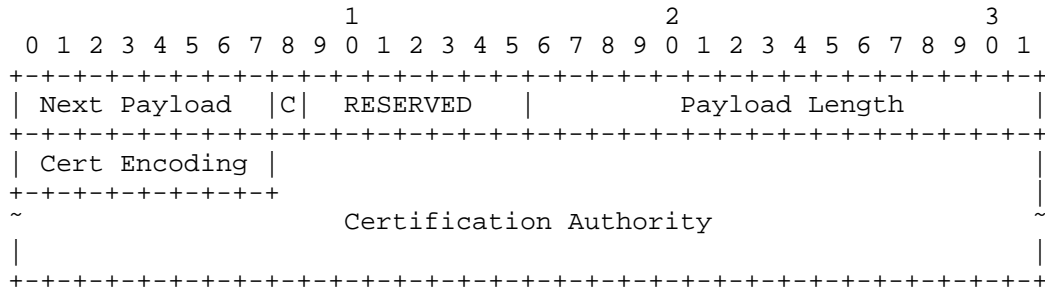


Figure 10: Certificate Request Payload Format

- o Certificate Encoding(1 octet) - Contains an encoding of the type or format of certificate requested.
- o Certification Authority (variable length) - Contains an encoding of an acceptable certification authority for the type of certificate requested.

The Certificate Encoding field has the same values as those defined certificate payload. The Certification Authority field contains an indicator of trusted authorities for this certificate type. The Certification Authority value is a concatenated list of SHA-1 hashes of the public keys of trusted Certification Authorities (CAs). Each is encoded as the SHA-1 hash of the Subject Public Key Info element (see Section 4.1.2.7 of [RFC5280]) from each Trust Anchor certificate. The 20-octet hashes are concatenated and included with no other formatting.

A.8. Authentication Payload

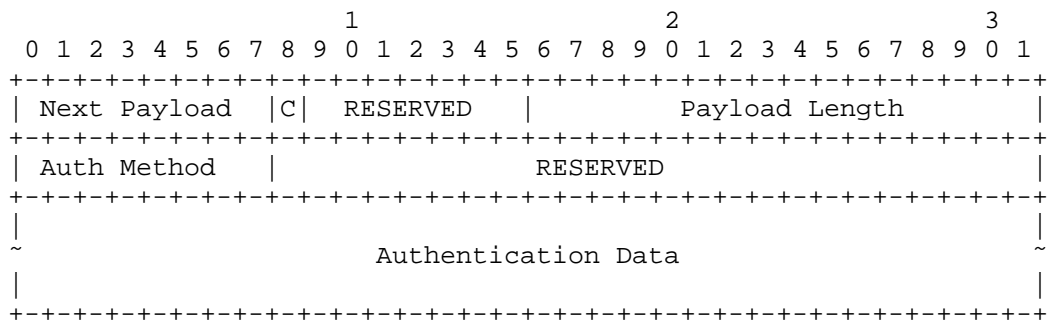


Figure 11: Authentication Payload Format

- o Auth Method (1 octet) - Specifies the method of authentication used.

Mechanism	Value

RSA Digital Signature	1
Using an RSA private key with RSASSA-PKCS1-v1_5 signature scheme specified in [PKCS1], see [RFC7296] Section 2.15 for details.	
Shared Key Message Integrity Code	2
Computed as specified earlier using the shared key associated with the identity in the ID payload and the negotiated PRF.	

- o Authentication Data (variable length) - see Section 2.1.

A.9. Nonce Payload

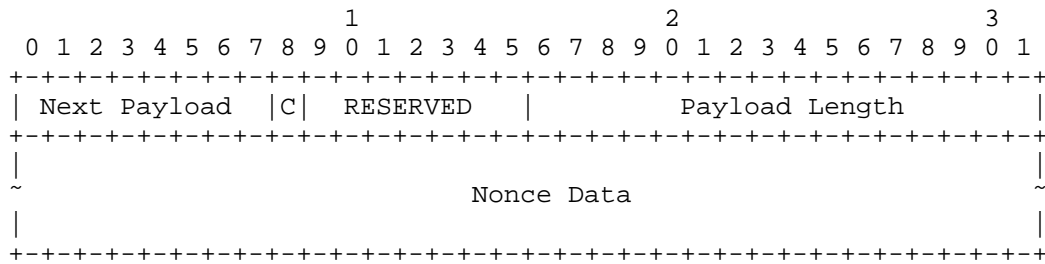


Figure 12: Nonce Payload Format

- o Nonce Data (variable length) - Contains the random data generated by the transmitting entity.

The size of the Nonce Data MUST be between 16 and 256 octets, inclusive. Nonce values MUST NOT be reused.

A.10. Notify Payload

The Notify payload, denoted N in this document, is used to transmit informational data, such as error conditions and state transitions, to an IKE peer. A Notify payload may appear in a response message (usually specifying why a request was rejected), in an INFORMATIONAL Exchange (to report an error not in an IKE request), or in any other message to indicate sender capabilities or to modify the meaning of the request.

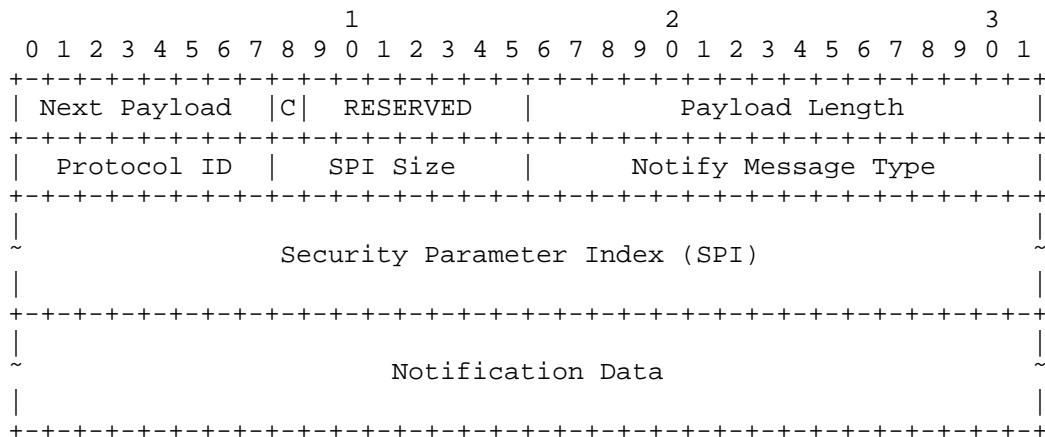


Figure 13: Notify Payload Format

- o Protocol ID (1 octet) - If this notification concerns an existing SA whose SPI is given in the SPI field, this field indicates the

type of that SA. If the SPI field is empty, this field MUST be sent as zero and MUST be ignored on receipt.

- o SPI Size (1 octet) - Length in octets of the SPI as defined by the IPsec protocol ID or zero if no SPI is applicable. For a notification concerning the IKE SA, the SPI Size MUST be zero and the field must be empty.
- o Notify Message Type (2 octets) - Specifies the type of notification message.
- o SPI (variable length) - Security Parameter Index.
- o Notification Data (variable length) - Status or error data transmitted in addition to the Notify Message Type. Values for this field are type specific.

A.10.1. Notify Message Types

Notification information can be error messages specifying why an SA could not be established. It can also be status data that a process managing an SA database wishes to communicate with a peer process.

Types in the range 0 - 16383 are intended for reporting errors. An implementation receiving a Notify payload with one of these types that it does not recognize in a response MUST assume that the corresponding request has failed entirely. Unrecognized error types in a request and status types in a request or response MUST be ignored, and they should be logged.

Notify payloads with status types MAY be added to any message and MUST be ignored if not recognized. They are intended to indicate capabilities, and as part of SA negotiation, are used to negotiate non-cryptographic parameters.

NOTIFY messages: error types	Value

UNSUPPORTED_CRITICAL_PAYLOAD	1
Indicates that the one-octet payload type included in the Notification Data field is unknown.	
INVALID_SYNTAX	7
Indicates the IKE message that was received was invalid because some type, length, or value was out of range or because the request was rejected for policy reasons. To avoid a DoS attack using forged messages, this status may only be returned for and in an encrypted packet if the Message ID and cryptographic checksum were valid. To avoid leaking information to someone probing a node, this status MUST be sent in response to any error not covered by one of the other status types. To aid debugging, more detailed error information should be written to a console or log.	
NO_PROPOSAL_CHOSEN	14
None of the proposed crypto suites was acceptable. This can be sent in any case where the offered proposals are not acceptable for the responder.	
NO_ADDITIONAL_SAS	35
Specifies that the node is unwilling to accept any more Child SAs.	
NOTIFY messages: status types	Value

INITIAL_CONTACT	16384
Asserts that this IKE SA is the only IKE SA currently active between the authenticated identities.	

A.11. Traffic Selector Payload

Traffic Selector (TS) payloads allow endpoints to communicate some of the information from their SPD to their peers. TS payloads specify the selection criteria for packets that will be forwarded over the newly set up SA.

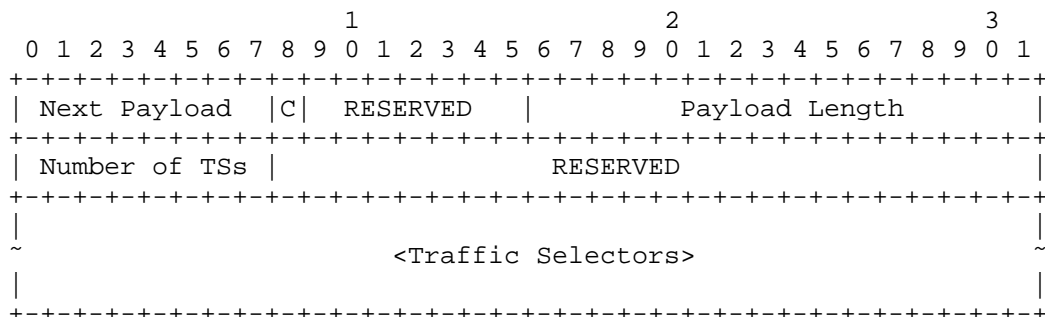


Figure 14: Traffic Selectors Payload Format

- o Number of TSSs (1 octet) - Number of Traffic Selectors being provided.
- o Traffic Selectors (variable length) - One or more individual Traffic Selectors.

The length of the Traffic Selector payload includes the TS header and all the Traffic Selectors.

There is no requirement that TS_i and TS_r contain the same number of individual Traffic Selectors. Thus, they are interpreted as follows: a packet matches a given TS_i/TS_r if it matches at least one of the individual selectors in TS_i, and at least one of the individual selectors in TS_r.

Two TS payloads appear in each of the messages in the exchange that creates a Child SA pair. Each TS payload contains one or more Traffic Selectors. Each Traffic Selector consists of an address range (IPv4 or IPv6), a port range, and an IP protocol ID.

The first of the two TS payloads is known as TSi (Traffic Selector-initiator). The second is known as TSr (Traffic Selector-responder). TSi specifies the source address of traffic forwarded from (or the destination address of traffic forwarded to) the initiator of the Child SA pair. TSr specifies the destination address of the traffic forwarded to (or the source address of the traffic forwarded from) the responder of the Child SA pair.

IKEv2 allows the responder to choose a subset of the traffic proposed by the initiator.

When the responder chooses a subset of the traffic proposed by the initiator, it narrows the Traffic Selectors to some subset of the initiator's proposal (provided the set does not become the null set).

If the type of Traffic Selector proposed is unknown, the responder ignores that Traffic Selector, so that the unknown type is not returned in the narrowed set.

To enable the responder to choose the appropriate range, if the initiator has requested the SA due to a data packet, the initiator SHOULD include as the first Traffic Selector in each of TSi and TSr a very specific Traffic Selector including the addresses in the packet triggering the request. If the initiator creates the Child SA pair not in response to an arriving packet, but rather, say, upon startup, then there may be no specific addresses the initiator prefers for the initial tunnel over any other. In that case, the first values in TSi and TSr can be ranges rather than specific values.

As minimal implementations might only support one SA, the traffic selectors will usually be from initiator's IP address to responders IP address (i.e. no port or protocol selectors and only one range).

A.11.1. Traffic Selector

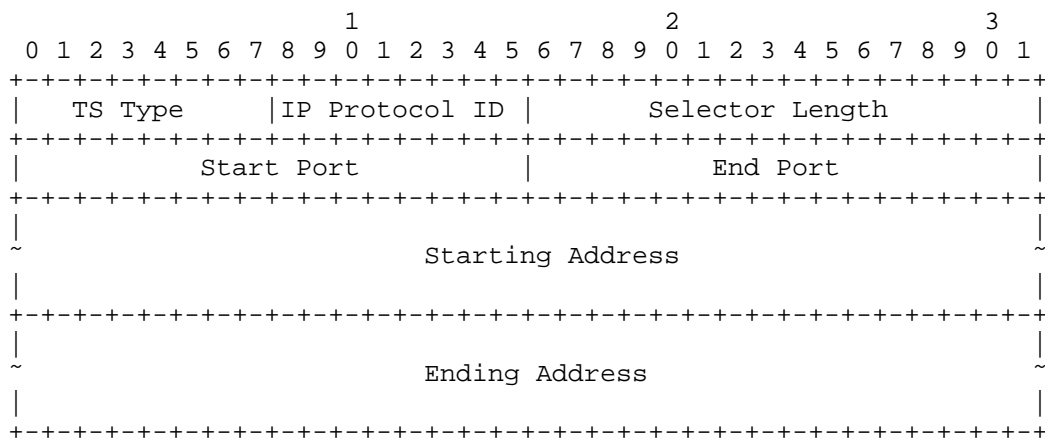


Figure 15: Traffic Selector

- o TS Type (one octet) - Specifies the type of Traffic Selector.
- o IP protocol ID (1 octet) - Value specifying an associated IP protocol ID (such as UDP, TCP, and ICMP). A value of zero means that the protocol ID is not relevant to this Traffic Selector -- the SA can carry all protocols.
- o Selector Length - Specifies the length of this Traffic Selector substructure including the header.

- o Start Port (2 octets, unsigned integer) - Value specifying the smallest port number allowed by this Traffic Selector. For protocols for which port is undefined (including protocol 0), or if all ports are allowed, this field MUST be zero.
- o End Port (2 octets, unsigned integer) - Value specifying the largest port number allowed by this Traffic Selector. For protocols for which port is undefined (including protocol 0), or if all ports are allowed, this field MUST be 65535.
- o Starting Address - The smallest address included in this Traffic Selector (length determined by TS Type).
- o Ending Address - The largest address included in this Traffic Selector (length determined by TS Type).

The following table lists values for the Traffic Selector Type field and the corresponding Address Selector Data.

TS Type	Value

TS_IPV4_ADDR_RANGE	7

A range of IPv4 addresses, represented by two four-octet values. The first value is the beginning IPv4 address (inclusive) and the second value is the ending IPv4 address (inclusive). All addresses falling between the two specified addresses are considered to be within the list.

TS_IPV6_ADDR_RANGE	8
--------------------	---

A range of IPv6 addresses, represented by two sixteen-octet values. The first value is the beginning IPv6 address (inclusive) and the second value is the ending IPv6 address (inclusive). All addresses falling between the two specified addresses are considered to be within the list.

A.12. Encrypted Payload

The Encrypted payload, denoted SK{...} in this document, contains other payloads in encrypted form.

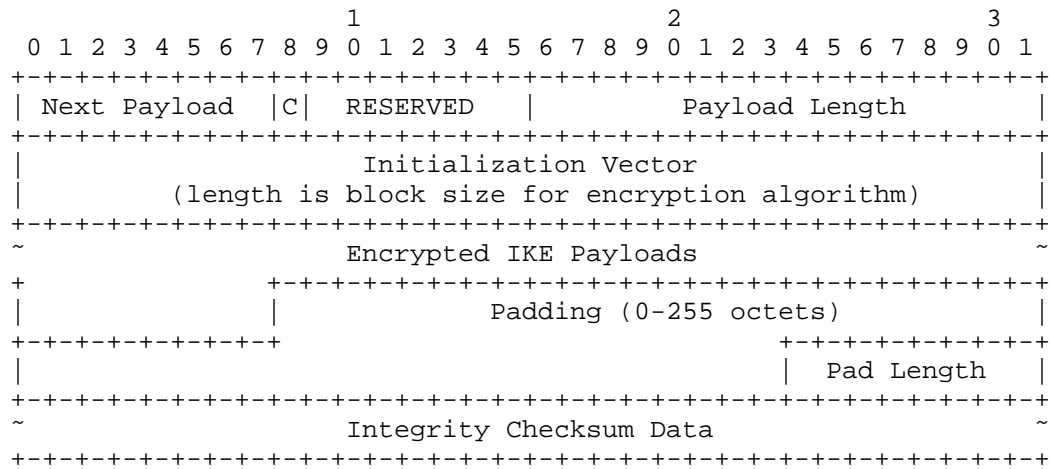


Figure 16: Encrypted Payload Format

- o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.
- o Payload Length - Includes the lengths of the header, initialization vector (IV), Encrypted IKE payloads, Padding, Pad Length, and Integrity Checksum Data.
- o Initialization Vector - For CBC mode ciphers, the length of the initialization vector (IV) is equal to the block length of the underlying encryption algorithm. Senders MUST select a new unpredictable IV for every message; recipients MUST accept any value. The reader is encouraged to consult [MODES] for advice on IV generation. In particular, using the final ciphertext block of the previous message is not considered unpredictable. For modes other than CBC, the IV format and processing is specified in the document specifying the encryption algorithm and mode.
- o IKE payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.
- o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the payloads, the Padding, and the Pad Length to be a multiple of the encryption block size. This field is encrypted with the negotiated cipher.

- o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.
- o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Appendix B. Useful Optional Features

There are some optional features of IKEv2, which might be useful for minimal implementations in some scenarios. Such features include Raw public keys authentication, and sending IKE SA delete notification.

B.1. IKE SA Delete Notification

In some scenarios, a minimal implementation device creates an IKE SA, sends one or few packets, perhaps gets some packets back, and then the device goes back to sleep forgetting the IKE SA. In such scenarios it would be nice for the minimal implementation to send the IKE SA delete notification to tell the other end that the IKE SA is going away, so it can free the resources.

Deleting the IKE SA can be done by sending one packet with a fixed Message ID, and with only one payload inside the encrypted payload. The other end will send back an empty response:

Initiator	Responder
<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <pre>HDR(SPIi=xxx, SPIr=yyy, INFORMATIONAL, Flags: Initiator, Message ID=2), SK {D} --></pre> </div> <div style="width: 45%; text-align: right;"> <pre><-- HDR(SPIi=xxx, SPIr=yyy, INFORMATIONAL, Flags: Response, Message ID=2), SK {}</pre> </div> </div>	

The delete payload format is:

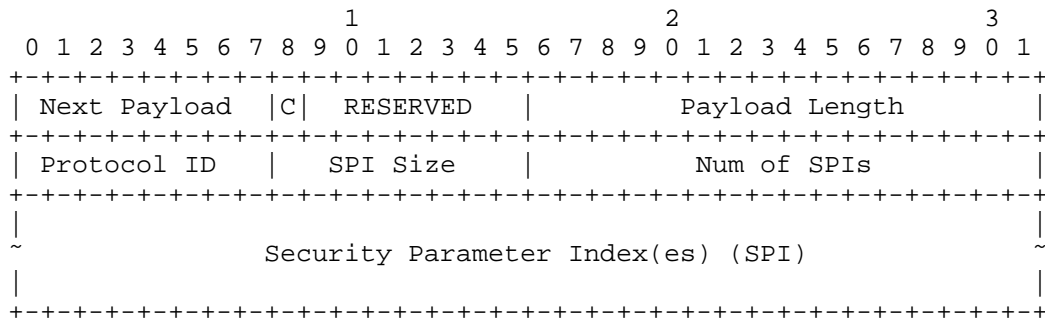


Figure 17: Delete Payload Format

- o Protocol ID (1 octet) - Must be 1 for an IKE SA.
- o SPI Size (1 octet) - Length in octets of the SPI as defined by the protocol ID. It MUST be zero for IKE (SPI is in message header).
- o Num of SPIs (2 octets, unsigned integer) - The number of SPIs contained in the Delete payload. This MUST be zero for IKE.
- o Security Parameter Index(es) (variable length) - Identifies the specific Security Association(s) to delete. The length of this field is determined by the SPI Size and Num of SPIs fields. This field is empty for the IKE SA delete.

B.2. Raw Public Keys

In some scenarios the shared secret authentication is not safe enough, as anybody who knows the secret can impersonate the server. If the shared secret is printed on the side of the device, then anybody who gets physical access to the device can read it. In such environments, public key authentication allows stronger authentication with minimal operational overhead. Certificate support is quite complex, and minimal implementations do not usually have need for them. Using Raw Public Keys is much simpler, and it scales similar to certificates. The fingerprint of the Raw Public Key can still be distributed by, for example, printing it on the side of the device allowing setup similar to using a shared secret.

Raw Public Keys can also be used in a "leap of faith" or baby duck style initial setup, where the device imprints itself to the first device it sees when it boots up the first time. After that initial connection it stores the fingerprint of the Raw Public Key of the server in its own configuration and verifies that it never changes (unless a "reset to factory settings" or similar command is issued).

This changes the initial IKE_AUTH payloads as follows:

Initiator	Responder

HDR(SPIi=xxx, SPIr=yyy, IKE_AUTH,	
Flags: Initiator, Message ID=1),	
SK {IDi, CERT, AUTH, SAI2, TSi, TSr,	
N(INITIAL_CONTACT)} -->	
	<-- HDR(SPIi=xxx, SPIr=yyy, IKE_AUTH, Flags:
	Response, Message ID=1),
	SK {IDr, CERT, AUTH, SAR2, TSi, TSr}

The CERT payloads contains the Raw Public Keys used to sign the hash of the InitiatorSignedOctects/ResponderSignedOctects when generating an AUTH payload. Minimal implementations should use SHA-1 as the hash function as that is the "SHOULD" support algorithm specified in RFC 7296, so it is the most likely one that is supported by all devices.

Note, that RFC 7296 already obsoleted the old Raw RSA Key method, and More Raw Public Keys for IKEv2 ([I-D.kivinen-ipsecme-oob-pubkey]) adds a new format to allow using any types of Raw Public Keys with IKEv2. This document only specifies how to use the new format.

In these setups it might be possible that authenticating the server is not needed at all. If a minimal device is sending, for example, sensor information to the server, the server wants to verify that the sensor is who it claims to be using raw public keys, but the sensor does not really care who the server is. In such cases the NULL authentication method ([RFC7619]) would be useful, as it allows devices to do one-way authentication.

Author's Address

Tero Kivinen
 INSIDE Secure
 Eerikinkatu 28
 HELSINKI FI-00180
 FI

Email: kivinen@iki.fi

LWIG Working Group
Internet-Draft
Intended status: Informational
Expires: January 10, 2014

C. Bormann
Universitaet Bremen TZI
M. Ersue
Nokia Siemens Networks
A. Keranen
Ericsson
July 09, 2013

Terminology for Constrained Node Networks
draft-ietf-lwig-terminology-05

Abstract

The Internet Protocol Suite is increasingly used on small devices with severe constraints, creating constrained node networks. This document provides a number of basic terms that have turned out to be useful in the standardization work for constrained environments.

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 January 10, 2014.

Copyright Notice

Copyright (c) 2013 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	2
2. Terminology	3
2.1. Constrained Nodes	3
2.2. Constrained Networks	4
2.2.1. Challenged Networks	5
2.3. Constrained Node Networks	5
2.3.1. LLN ("low-power lossy network")	6
2.3.2. LoWPAN, 6LoWPAN	6
3. Classes of Constrained Devices	8
4. Power Terminology	10
4.1. Scaling Properties	10
4.2. Classes of Energy Limitation	10
4.3. Strategies of Using Power for Communication	11
5. Security Considerations	13
6. IANA Considerations	13
7. Acknowledgements	13
8. Informative References	13
Authors' Addresses	15

1. Introduction

Small devices with limited CPU, memory, and power resources, so called constrained devices (also known as sensor, smart object, or smart device) can constitute a network, becoming "constrained nodes" in that network. Such a network may itself exhibit constraints, e.g. with unreliable or lossy channels, limited and unpredictable bandwidth, and a highly dynamic topology.

Constrained devices might be in charge of gathering information in diverse settings including natural ecosystems, buildings, and factories and sending the information to one or more server stations. Constrained devices may work under severe resource constraints such as limited battery and computing power, little memory, as well as insufficient wireless bandwidth and ability to communicate. Other entities on the network, e.g., a base station or controlling server, might have more computational and communication resources and could support the interaction between the constrained devices and applications in more traditional networks.

Today diverse sizes of constrained devices with different resources and capabilities are becoming connected. Mobile personal gadgets, building-automation devices, cellular phones, Machine-to-machine (M2M) devices, etc. benefit from interacting with other "things"

nearby or somewhere in the Internet. With this, the Internet of Things (IoT) becomes a reality, built up out of uniquely identifiable and addressable objects (things). And over the next decade, this could grow to large numbers [fifty-billion] of Internet-connected constrained devices, greatly increasing the Internet's size and scope.

The present document provides a number of basic terms that have turned out to be useful in the standardization work for constrained environments. The intention is not to exhaustively cover the field, but to make sure a few core terms are used consistently between different groups cooperating in this space.

In this document, the term "byte" is used in its now customary sense as a synonym for "octet". Where sizes of semiconductor memory are given, the prefix "kibi" (1024) is combined with "byte" to "kibibyte", abbreviated "KiB", for 1024 bytes [ISQ-13].

2. Terminology

The main focus of this field of work appears to be `_scaling_`:

- o Scaling up Internet technologies to a large number [fifty-billion] of inexpensive nodes, while
- o scaling down the characteristics of each of these nodes and of the networks being built out of them, to make this scaling up economically and physically viable.

The need for scaling down the characteristics of nodes leads to `_constrained nodes_`.

2.1. Constrained Nodes

The term "constrained node" is best defined by contrasting the characteristics of a constrained node with certain widely held expectations on more familiar Internet nodes:

Constrained Node: A node where some of the characteristics that are otherwise pretty much taken for granted for Internet nodes in 2013 are not attainable, often due to cost constraints and/or physical constraints on characteristics such as size, weight, and available power and energy.

While this is less than satisfying as a rigorous definition, it is grounded in the state of the art and clearly sets apart constrained nodes from server systems, desktop or laptop computers, powerful mobile devices such as smartphones etc. There may be many design

considerations that lead to these constraints, including cost, size, weight, and other scaling factors.

(An alternative name, when the properties as a network node are not in focus, is "constrained device".)

There are multiple facets to the constraints on nodes, often applying in combination, e.g.:

- o constraints on the maximum code complexity (ROM/Flash);
- o constraints on the size of state and buffers (RAM);
- o constraints on the available power.

Section 3 defines a small number of interesting classes ("class-N" for N=0,1,2) of constrained nodes focusing on relevant combinations of the first two constraints. With respect to available power, [RFC6606] distinguishes "power-affluent" nodes (mains-powered or regularly recharged) from "power-constrained nodes" that draw their power from primary batteries or by using energy harvesting; more detailed power terminology is given in Section 4.

The use of constrained nodes in networks often also leads to constraints on the networks themselves. However, there may also be constraints on networks that are largely independent from those of the nodes. We therefore distinguish constrained networks and constrained node networks.

2.2. Constrained Networks

We define "constrained network" in a similar way:

Constrained Network: A network where some of the characteristics pretty much taken for granted with link layers in common use in the Internet by 2013, are not attainable.

Again, there may be several reasons for this:

- o cost constraints on the network,
- o constraints of the nodes (for constrained node networks),
- o physical constraints (e.g., power constraints, environmental constraints, media constraints such as underwater operation, limited spectrum for very high density, electromagnetic compatibility),

- o regulatory constraints, such as very limited spectrum availability (including limits on effective radiated power and duty cycle), or explosion safety,
- o technology constraints, such as older and lower speed technologies that are still operational and may need to stay in use for some more time.

Constraints may include:

- o low achievable bit rate (including limits on duty cycle),
- o high packet loss, packet loss (delivery rate) variability,
- o severe penalties for using larger packets (e.g., high packet loss due to link layer fragmentation),
- o lack of (or severe constraints on) advanced services such as IP multicast.

2.2.1. Challenged Networks

A constrained network is not necessarily a `_challenged_` network [FALL]:

Challenged Network: A network that has serious trouble maintaining what an application would today expect of the end-to-end IP model, e.g., by:

- o not being able to offer end-to-end IP connectivity at all;
- o exhibiting serious interruptions in end-to-end IP connectivity;
- o exhibiting delay well beyond the Maximum Segment Lifetime (MSL) defined by TCP [RFC0793].

All challenged networks are constrained networks in some sense, but not all constrained networks are challenged networks. There is no well-defined boundary between the two, though. Delay-Tolerant Networking (DTN) has been designed to cope with challenged networks [RFC4838].

2.3. Constrained Node Networks

Constrained Node Network: A network whose characteristics are influenced by being composed of a significant portion of constrained nodes.

A constrained node network always is a constrained network because of the network constraints stemming from the node constraints, but may also have other constraints that already make it a constrained network.

2.3.1. LLN ("low-power lossy network")

A related term that has been used recently is "low-power lossy network" (LLN). In its terminology document, the ROLL working group is saying [I-D.ietf-roll-terminology]:

LLN: Low power and Lossy networks (LLNs) are typically composed of many embedded devices with limited power, memory, and processing resources interconnected by a variety of links, such as IEEE 802.15.4 or Low Power WiFi. There is a wide scope of application areas for LLNs, including industrial monitoring, building automation (HVAC, lighting, access control, fire), connected home, healthcare, environmental monitoring, urban sensor networks, energy management, assets tracking and refrigeration.. [sic]

In common usage, LLN often stands for "the network characteristics that RPL has been designed for". Beyond what is said in the ROLL terminology document, LLNs do appear to have significant loss at the physical layer, with significant variability of the delivery rate, and some short-term unreliability, coupled with some medium term stability that makes it worthwhile to construct medium-term stable directed acyclic graphs for routing and do measurements on the edges such as ETX [RFC6551]. Actual "low power" does not seem to be required for an LLN [I-D.hui-vasseur-roll-rpl-deployment], and the positions on scaling of LLNs appear to vary widely [I-D.clausen-lln-rpl-experiences].

The ROLL terminology document states that LLNs typically are composed of constrained nodes; this is also supported by the design of operation modes such as RPL's "non-storing mode". So, in the terminology of the present document, an LLN seems to be a constrained node network with certain network characteristics, which include constraints on the network as well.

2.3.2. LoWPAN, 6LoWPAN

One interesting class of a constrained network often used as a constrained node network is the "LoWPAN" [RFC4919], a term inspired from the name of the IEEE 802.15.4 working group (low-rate wireless personal area networks (LR-WPANs)). The expansion of that acronym, "Low-Power Wireless Personal Area Network" contains a hard to justify "Personal" that is due to the history of task group naming in IEEE 802 more than due to an orientation of LoWPANs around a single

person. Actually, LoWPANs have been suggested for urban monitoring, control of large buildings, and industrial control applications, so the "Personal" can only be considered a vestige. Maybe the term is best read as "Low-Power Wireless Area Networks" (LoWPANs) [WEI]. Originally focused on IEEE 802.15.4, "LoWPAN" (or when used for IPv6, "6LoWPAN") is now also being used for networks built from similarly constrained link layer technologies [I-D.ietf-6lowpan-btle] [I-D.mariager-6lowpan-v6over-dect-ule] [I-D.brandt-6man-lowpanz].

3. Classes of Constrained Devices

Despite the overwhelming variety of Internet-connected devices that can be envisioned, it may be worthwhile to have some succinct terminology for different classes of constrained devices. In this document, the class designations in Table 1 may be used as rough indications of device capabilities:

Name	data size (e.g., RAM)	code size (e.g., Flash)
Class 0, C0	<< 10 KiB	<< 100 KiB
Class 1, C1	~ 10 KiB	~ 100 KiB
Class 2, C2	~ 50 KiB	~ 250 KiB

Table 1: Classes of Constrained Devices (KiB = 1024 bytes)

As of the writing of this document, these characteristics correspond to distinguishable clusters of commercially available chips and design cores for constrained devices. While it is expected that the boundaries of these classes will move over time, Moore's law tends to be less effective in the embedded space than in personal computing devices: Gains made available by increases in transistor count and density are more likely to be invested in reductions of cost and power requirements than into continual increases in computing power.

Class 0 devices are very constrained sensor-like nodes. Most likely they will not be able to communicate directly with the Internet in a secure manner. Class 0 devices will participate in Internet communications with the help of larger devices acting as proxies, gateways or servers. Class 0 devices generally cannot be secured or managed comprehensively in the traditional sense. They will most likely be preconfigured (and will be reconfigured rarely, if at all), with a very small data set. For management purposes, they could answer keepalive signals and send on/off or basic health indications.

Class 1 devices cannot easily talk to other Internet nodes employing a full protocol stack such as using HTTP, TLS and related security protocols and XML-based data representations. However, they have enough power to use a protocol stack specifically designed for constrained nodes (e.g., CoAP over UDP) and participate in meaningful conversations without the help of a gateway node. In particular, they can provide support for the security functions required on a large network. Therefore, they can be integrated as fully developed peers into an IP network, but they need to be parsimonious with state

memory, code space, and often power expenditure for protocol and application usage.

Class 2 can already support mostly the same protocol stacks as used on notebooks or servers. However, even these devices can benefit from lightweight and energy-efficient protocols and from consuming less bandwidth. Furthermore, using fewer resources for networking leaves more resources available to applications. Thus, using the protocol stacks defined for very constrained devices also on Class 2 devices might reduce development costs and increase the interoperability.

Constrained devices with capabilities significantly beyond Class 2 devices exist. They are less demanding from a standards development point of view as they can largely use existing protocols unchanged. The present document therefore does not make any attempt to define classes beyond Class 2. These devices can still be constrained by a limited energy supply.

With respect to examining the capabilities of constrained nodes, particularly for Class 1 devices, it is important to understand what type of applications they are able to run and which protocol mechanisms would be most suitable. Because of memory and other limitations, each specific Class 1 device might be able to support only a few selected functions needed for its intended operation. In other words, the set of functions that can actually be supported is not static per device type: devices with similar constraints might choose to support different functions. Even though Class 2 devices have some more functionality available and may be able to provide a more complete set of functions, they still need to be assessed for the type of applications they will be running and the protocol functions they would need. To be able to derive any requirements, the use cases and the involvement of the devices in the application and the operational scenario need to be analyzed. Use cases may combine constrained devices of multiple classes as well as more traditional Internet nodes.

4. Power Terminology

Devices not only differ in their computing capabilities, but also in available electrical power and/or energy. While it is harder to find recognizable clusters in this space, it is still useful to introduce some common terminology.

4.1. Scaling Properties

The power and/or energy available to a device may vastly differ, from kilowatts to microwatts, from essentially unlimited to hundreds of microjoules.

Instead of defining classes or clusters, we propose simply stating, in SI units, an approximate value for one or both of the quantities listed in Table 2:

Name	Definition	SI Unit
Ps	Sustainable average power available for the device over the time it is functioning	W (Watt)
Et	Total electrical energy available before the energy source is exhausted	J (Joule)

Table 2: Quantities Relevant to Power and Energy

The value of Et may need to be interpreted in conjunction with an indication over which period of time the value is given; see the next subsection.

4.2. Classes of Energy Limitation

As discussed above, some devices are limited in available energy as opposed to (or in addition to) being limited in available power. Where no relevant limitations exist with respect to energy, the device is classified as E3. The energy limitation may be in total energy available in the usable lifetime of the device (e.g. a device with a non-replaceable primary battery, which is discarded when this battery is exhausted), classified as E2. Where the relevant limitation is for a specific period, this is classified as E1, e.g. a limited amount of energy available for the night with a solar-powered device, or for the period between recharges with a device that is manually connected to a charger, or by a periodic (primary) battery replacement interval. Finally, there may be a limited amount of energy available for a specific event, e.g. for a button press in

an energy harvesting light switch; this is classified as E0. Note that many E1 devices in a sense also are E2, as the rechargeable battery has a limited number of useful recharging cycles.

In summary, we distinguish (Table 3):

Name	Type of energy limitation	Example Power Source
E0	Event energy-limited	Event-based harvesting
E1	Period energy-limited	Battery that is periodically recharged or replaced
E2	Lifetime energy-limited	Non-replaceable primary battery
E3	No direct quantitative limitations to available energy	Mains powered

Table 3: Classes of Energy Limitation

4.3. Strategies of Using Power for Communication

Especially when wireless transmission is used, the radio often consumes a big portion of the total energy consumed by the device. Design parameters such as the available spectrum, the desired range, and the bitrate aimed for, influence the power consumed during transmission and reception; the duration of transmission and reception (including potential reception) influence the total energy consumption.

Based on the type of the energy source (e.g., battery or mains power) and how often device needs to communicate, it may use different kinds of strategies for power usage and network attachment.

The general strategies for power usage can be described as follows:

Always-on: This strategy is most applicable if there is no reason for extreme measures for power saving. The device can stay on in the usual manner all the time. It may be useful to employ power-friendly hardware or limit the number of wireless transmissions, CPU speeds, and other aspects for general power saving and cooling needs, but the device can be connected to the network all the time.

Always-off: Under this strategy, the device sleeps such long periods at a time that once it wakes up, it makes sense for it to not pretend that it has been connected to the network during sleep: The device re-attaches to the network as it is woken up. The main optimization goal is to minimize the effort during such re-attachment process and any resulting application communications.

If the device sleeps for long periods of time, and needs to communicate infrequently, the relative increase in energy expenditure during reattachment may be acceptable.

Low-power: This strategy is most applicable to devices that need to operate on a very small amount of power, but still need to be able to communicate on a relatively frequent basis. This implies that extremely low power solutions needs to be used for the hardware, chosen link layer mechanisms, and so on. Typically, given the small amount of time between transmissions, despite their sleep state these devices retain some form of network attachment to the network. Techniques used for minimizing power usage for the network communications include minimizing any work from re-establishing communications after waking up, tuning the frequency of communications, and other parameters appropriately.

In summary, we distinguish (Table 4):

Name	Strategy	Ability to communicate
S0	Always-off	Re-attach when required
S1	Low-power	Appears connected, perhaps with high latency
S2	Always-on	Always connected

Table 4: Strategies of Using Power for Communication

Note that the discussion above is at the device level; similar considerations can apply at the communications interface level. This document does not define terminology for the latter.

5. Security Considerations

This document introduces common terminology that does not raise any new security issue. Security considerations arising from the constraints discussed in this document need to be discussed in the context of specific protocols. For instance, [I-D.ietf-core-coap] section 11.6, "Constrained node considerations", discusses implications of specific constraints on the security mechanisms employed.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

Dominique Barthel and Peter van der Stok provided useful comments; Charles Palmer provided a full editorial review.

Peter van der Stok insisted that we should have power terminology, hence Section 4. The text for Section 4.3 is mostly lifted from [I-D.arkko-lwig-cellular] and has been adapted for this document.

8. Informative References

- [FALL] Fall, K., "A Delay-Tolerant Network Architecture for Challenged Internets", SIGCOMM 2003, 2003.
- [I-D.arkko-lwig-cellular] Arkko, J., Eriksson, A., and A. Keraenen, "Building Power-Efficient CoAP Devices for Cellular Networks", draft-arkko-lwig-cellular-00 (work in progress), February 2013.
- [I-D.brandt-6man-lowpanz] Brandt, A. and J. Buron, "Transmission of IPv6 packets over ITU-T G.9959 Networks", draft-brandt-6man-lowpanz-02 (work in progress), June 2013.
- [I-D.clausen-lln-rpl-experiences] Clausen, T., Verdiere, A., Yi, J., Herberg, U., and Y. Igarashi, "Observations of RPL: IPv6 Routing Protocol for Low power and Lossy Networks", draft-clausen-lln-rpl-experiences-06 (work in progress), February 2013.
- [I-D.hui-vasseur-roll-rpl-deployment]

Vasseur, J., Hui, J., Dasgupta, S., and G. Yoon, "RPL deployment experience in large scale networks", draft-hui-vasseur-roll-rpl-deployment-01 (work in progress), July 2012.

[I-D.ietf-6lowpan-btle]

Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "Transmission of IPv6 Packets over BLUETOOTH Low Energy", draft-ietf-6lowpan-btle-12 (work in progress), February 2013.

[I-D.ietf-core-coap]

Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.

[I-D.ietf-roll-terminology]

Vasseur, J., "Terminology in Low power And Lossy Networks", draft-ietf-roll-terminology-12 (work in progress), March 2013.

[I-D.mariager-6lowpan-v6over-dect-ule]

Mariager, P. and J. Petersen, "Transmission of IPv6 Packets over DECT Ultra Low Energy", draft-mariager-6lowpan-v6over-dect-ule-02 (work in progress), May 2012.

[ISQ-13] International Electrotechnical Commission, "International Standard -- Quantities and units -- Part 13: Information science and technology", IEC 80000-13, March 2008.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, April 2007.

[RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

[RFC6551] Vasseur, JP., Kim, M., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, March 2012.

[RFC6606] Kim, E., Kaspar, D., Gomez, C., and C. Bormann, "Problem Statement and Requirements for IPv6 over Low-Power

Wireless Personal Area Network (6LoWPAN) Routing", RFC 6606, May 2012.

[WEI] Shelby, Z. and C. Bormann, "6LoWPAN: the Wireless Embedded Internet", ISBN 9780470747995, 2009.

[fifty-billion]

Ericsson, "More Than 50 Billion Connected Devices", Ericsson White Paper 284 23-3149 Uen, February 2011, <<http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf>>.

Authors' Addresses

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Mehmet Ersue
Nokia Siemens Networks
St.-Martinstrasse 76
81541 Munich
Germany

Phone: +49 172 8432301
Email: mehmet.ersue@nsn.com

Ari Keranen
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: ari.keranen@ericsson.com

Core Working Group
Internet Draft
Intended status: Informational
Expires: March 29, 2014

J. Zhu
M. Qi
China Mobile
Sep 29, 2013

Group Authentication
draft-zhu-core-groupauth-01

Abstract

The group communication is designed for the communication of Internet of Things. A threat is identified in [I-D.ietf-core-groupcomm] that current DTLS based approach is unicast oriented and there is no supporting on group authentication feature. Unicast oriented authentication will causing serious burden when a large number of terminal nodes will be involved inevitably. In another aspect, some terminals will own the same characteristics, such as owning same features, in the same place, working in the same time, etc. With this mechanism, all terminals can be authenticated together with little signaling and calculation at the same time. It will reduce the network burden and save time. This draft describes the security of group authentication and an group authentication implementation method for the Internet of things.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on March 29, 2014.

Copyright Notice

Copyright (c) 2013 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.

Table of Contents

1. Introduction	2
2. Conventions used in this document	3
2.1. Definitions.....	3
3. Problem Statement	4
3.1. Use cases	4
3.2. Problem statement	5
4. Requirement	6
5. Group Authentication Solution	7
5.1. Introduction	7
5.2. Detailed group scenario description	7
5.3. Group scenario procedure	9
6. Security Considerations	10
7. IANA Considerations	11
8. Conclusions	11
9. Acknowledgement.....	11
10. References	11
10.1. Normative References	11
10.2. Informative References	11

1. Introduction

With the development of Internet of Things, a large number of

terminal nodes will be involved inevitably. The unicast authentication communication from big amount terminals will merge together in the network, and causing serious burden to the server. Although IP multicast technical is introduced for group communication in [I-D.ietf-core-groupcomm], IP multicast relies on the unicast authentication at initial stage. In another aspect, some terminals will own the same characteristics, such as owning same features, in the same place, working in the same time, etc. With this mechanism, all terminals can be authenticated with little signaling and calculation at the same time. It will reduce the network burden and save time.

This draft describes the security of group authentication and an group authentication implementation method for the Internet of things.

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 RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

2.1. Definitions

Terminals: It is such a constrained device also recognized as group node in this document which communicates with server, through direct or indirect connections, which can be seen by the server. If some constrained devices like Zigbee node only communicates with sink node and it can't be seen by the server, such constrained device is not recognized as terminal.

Group agent: It is a device on behalf of group nodes (terminals) to make mutual authentication with network server.

Network server: It is the core network server which is responsible for authenticating the legality of terminal and provides specific services for them.

3. Problem Statement

3.1. Use cases

Nowadays the normal authentication mechanism in network is a traditional unicast authentication method between a single terminal and a single network entity. The authentication mechanism will be finished based one 1-2 round of challenge-response conversation separately. If there are many terminals, the cost for authentication will be increased.

Now for some M2M service, it may be a large amount of terminal used for an M2M service. These terminals are placed in the same location, will be used for the same purpose, and own same behavior. These terminals can be worked together as a group. In these scenarios, the existing authentication mechanism is no longer appropriate. When a large number of terminals want to access network server, huge number of authentication signaling will be generated by the unicast authentication method. What is more, it will cause network congestion and lead to DoS attack. For some terminal devices which is restricted with limited computing capability and power, the traditional unicast authentication will increase the computational burden of these terminals and drain their poor battery.

The following use cases are identified at this point:

Smart Metering: A large amount of smart power meter terminals are deployed in a block. The smart meter uploads meter report frequently through the network to smart meter server. What is more, smart meter server queries all terminals periodically to check whether the terminal is workable or not. So smart meters report at the same time, or the smart meter server need to re-configure all smart meters at the same time. In fact, there are other type of smart metering which has no agent node. This will lead to overload since each apartment needs to equip with a meter and they access network parallel at the same time and it will not be considered in this draft.

Remote Vehicle Management: IOT terminals contains GPS location reporter, remote air condition control, etc. would be installed in some special Vehicles like Taxi. It will send information such as position information, navigation, remote diagnosis, on-board communication, news and

entertainment information etc. to the network server in order to make better vehicle scheduling, vehicle monitoring and vehicle controlling. So it needs to connect to the network and make authentication at first. However, such vehicles would gather in a small place like airport, train station, etc. The frequency of connection from these terminals to server will cause overloading.

Intelligent home: various sensors equipped with communication modules are deployed in a house to monitor house conditions and make a control when necessary. These sensors collect and report house related information like the status of door open/close, indoor temperature to its owner through a network, and take actions by following the regulating instructions send by the owner.

3.2. Problem statement

In the current smart metering service use cases, a large amount of smart power meter terminals are deployed in a block. The smart meter uploads meter report frequently through the network to smart meter server. What is more, smart meter server queries all terminals periodically to check whether the terminal is workable or not. Therefore, the meter requires frequent and network communication.

In such use cases, when all the meters access network parallel at the same time, or when the server sends message to all meters, the terminals will connect to the network in a short time period (1sec ~ 1min). Assume there are 19 buildings in the block, and each building has 25 floors on average with 10 apartments in each floor. If each apartment is equipped with 1 smart power meter, then 4760 meters will be deployed in total in the block. This will cause pressure to the network.

So an agent node has been introduced to aggregate the message from these meters and then send out these meters data to the server together. After the agent is introduced, the connection between meters and servers is split into two parts: one is the connection between meters, the other is and the one between agent and server. Usually the agent is responsible for the authentication of the meters.

The server is responsible for the authentication of the agent only and gets all information about meters such as ID, data, from agent.

The current security mechanism is:

1. Each meter is authenticated with the agent. Agent will authenticate the meter one by one. After that, agent should make mutual authentication with server. Then server can confirm agent identity.
2. Meter will set up security connection with agent, and agent will also set up security connection with server. When a meter wants to send data to server. It should send the data confidential protected to agent first. Agent will decrypt the data and transfer it to server by using the security protection mechanism between agent and server.

However, this procedure has the following security problems:

1. Since all meters are authenticated by the agent and no direct authentication from server to meter. The server can get meter's ID and data only through agent. So the agent Due to the key position in the authentication, the security protection about agent is very important. Server could not authenticate meters directly. It can only rely on the agent. However, the agent would be placed in un-secure place or owned by different user rather than the server owner. If the agent is compromised or lay to server, agent can act as a middle attacker that makes fake authentication to meters and report fake ID to servers.
2. Another security problem is related with agent and server. Under this scenario, all information from meters will be transferred through agent. So agent will know all information generated by meters. However, under some scenario, agent would be owned and used by different user other than the meters' and servers' owner. So under this assumption, the agent should not get the message from meter to server. So meters should set up an secure end-to-end tunnel with server. It should request another authentication and key generation procedure in addition to authenticate with agent. This will bring complexity and overhead to the system.

4. Requirement

In order to reduce the cost and simplify a lot of overhead with the same characteristics of these groups of meter or sensor node group-based operations, it is needed to provide group authentication. For example, when smart meters perform bulk configuration information updates, it is needed to ensure that the correct identity of the user node within the group, to prevent the configuration information is wrong node receives. In addition, when smart meters report meter readings to the electricity system platform, it is also needed to be able to prove the correctness of the identity of smart meters, to prevent malicious node reporting false readings.

5. Group Authentication Solution

5.1. Introduction

Group authentication is a kind of authentication technologies that a group of users or terminals can be authenticated together at the same time. Instead of authenticating a number of terminals of a group one by one, group authentication mechanism treats these terminals in the group as a whole, and authenticates them together. Each group has a unique identifier, and an agent, which can be called as group agent, group gateway, etc.

Group authentication comprises following two phases as following:

1. The first phase is that user/terminal should be authenticated whether it belongs to a given group. This can be implemented through the proprietary authentication technology in a group, such as Zigbee or any others.
2. The second phase is that mutual authentication should be made between a given network entity, and a group agent who is responsible to delegate all terminals in the group.

After the authentication, terminals and network entity can generate separated session keys individually if there is some demand to make individual communication between network entity and each terminal.

5.2. Detailed group scenario description

For group authentication, there is detailed network description as following. There are 5 nodes inside a given group. They are A1, A2, A3, A4, and A5 which is group agent. And the given group can be named as group A. All nodes in group A can communicate with each other. What is more, A5 is able to communicate with network entity directly. Network entity will store the group information, such as identifiers, root keys used for all nodes inside the group. Network entity is also responsible for generating group authentication vector. The scenario is shown as below.

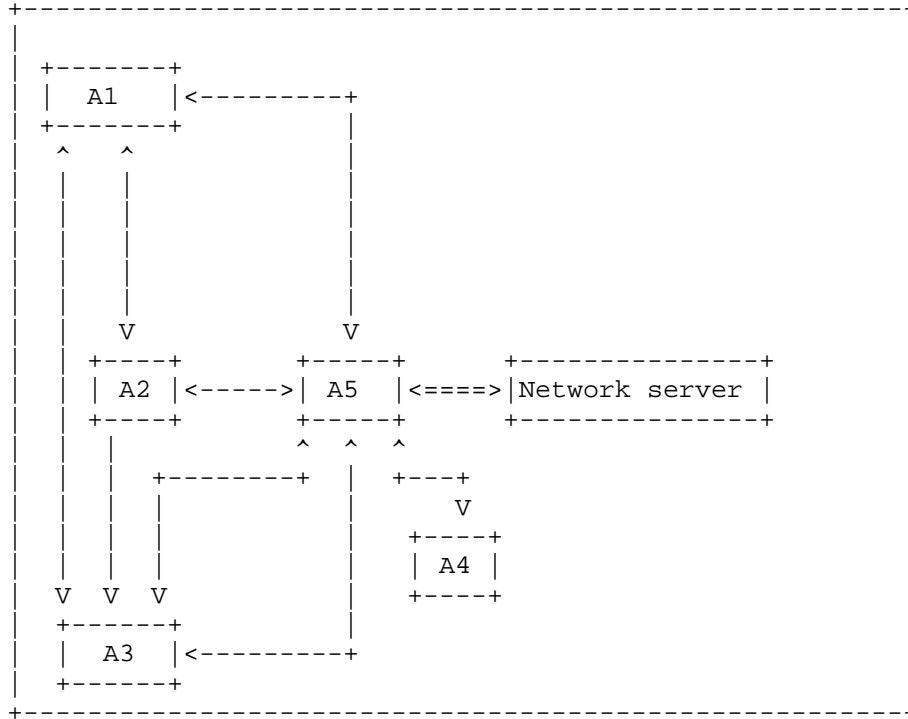


Figure 1 Group Authentication Architecture

- o A5 (group agent) communicates with other nodes, i.e. A1, A2, A3, A4 by inner group protocol. All nodes should contain such models as inner group communication model, group authentication mode. Inner group communication model can be used to sending/receiving

the group authentication message. Group authentication model can be used to generate authentication vectors/response and to authenticate peers.

- o Group agent will make mutual authentication with network entities. There are two kinds of network entities. Network server is responsible for mutual authentication action with group agent. And Network Server is responsible for group authentication vector generation and forwarding AV to network server. After the authentication, terminals and network entity can generate separated session keys individually if there is some demand to make individual communication between network entity and each terminals.

- o Group agent who represents the whole group, communicates with network entity, and generate group session key through authentication with the network server.

- o Pre-configure of the group

All the group nodes should be configured with sub key k_1 , k_2 , k_3 , k_4 , k_g , which will be used for mutual authentication in the group and separated communication.

5.3. Group scenario procedure

As mentioned above, group authentication can be divided into two phases.

In the first phase, group member, say A_i , sends authentication request to group agent at first as following.

1. Group member A_i sends message to trigger authentication at first.
2. Group agent sends authentication request to each group member.
3. Group member A_i verifies group agent at first. If success, A_i will generate session key for the communication with group agent, and sends response containing such session key back to group agent. If not success, the authentication is failed and group authentication procedure will be abort.
4. Group agent authenticates each group member A_i through the response message and record the authentication result in a mapping

table.

After the inner group authentication, all of group members are authenticated by group agent, and second phase can be performed.

5. Group agent sends message to network server to trigger the authentication outside the group.
6. Meanwhile, group agent sends authentication vector request to network server with group agent identity.
7. Network Server will generate authentication vector according to group agent identity.
8. What is more, network server should be able to recognize that is a group authentication is performed based on group agent identity. Network Server will generate session key for each group members by using pre-configured group member information and the same keying material in above step.
9. Network Server will send such authentication vector and session keys together back to network server.
10. Network server will perform mutual authentication with group agent.
11. Group agent authenticates group agent and send authentication response back to network server.
12. Network server authenticates group agent. If success, it can be considered that group agent and all group terminals is authenticated successfully.
13. Group agent will communicate with network server to choose the confidential and integrity protection algorithms.
14. After that, group agent will send keying material, selected algorithms to each group member.
15. Group member will generate session keys.

After these two phases, each terminal is authenticated with network server and generate independently session key with network server.

6. Security Considerations

TBD

7. IANA Considerations

There are no IANA considerations associated to this memo.

8. Conclusions

This memo describes the problem raised by using one-to-one authentication for huge number of Internet of Things terminals.

After that, group authentication requirement is raised and a group authentication mechanism is proposed. By using the proposed group authentication mechanism, the exploited group agent can behalf of all involved group nodes to make mutual authentication with network server, but the agent can not realize the content transmitted between both of them since it is just a intermediate node to forward messages which are encrypted by specific session key between the group node and network server. The trust relationship is between group nodes and network server. After authentication, the trust relationship between group nodes and group agent are not needed.

9. Acknowledgement

Thanks very much to Bert Greevenbosch, Stefanie Gerdes, Kepeng Li for their helpful comments and significant suggestions to revise this document.

10. References

10.1. Normative References

10.2. Informative References

Authors' Addresses

Judy Zhu
China Mobile
Unit 2, 32 Xuanwumenxi Ave,
Xicheng District,
Beijing 100053, China
Email: zhuhongru@chinamobile.com

Minpeng Qi
China Mobile
Unit 2, 32 Xuanwumenxi Ave,
Xicheng District,
Beijing 100053, China
Email: qiminpeng@chinamobile.com