

DHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 26, 2015

S. Jiang, Ed.
Huawei Technologies Co., Ltd
D. Zhang
June 24, 2015

Secure DHCPv4
draft-jiang-dhc-sedhcpv4-01

Abstract

The Dynamic Host Configuration Protocol for IPv4 (DHCPv4) enables DHCPv4 servers to pass configuration parameters. It offers configuration flexibility. If not being secured, DHCPv4 is vulnerable to various attacks, particularly spoofing attacks. This document analyzes the security issues of DHCPv4 and specifies a Secure DHCPv4 mechanism for communications between DHCPv4 clients and servers. This document provides a DHCPv4 client/server authentication mechanism based on sender's public/private key pairs or certificates with associated private keys. The DHCPv4 message exchanges are protected by the signature option and the timestamp option newly defined in this document.

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 December 26, 2015.

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

Internet-Draft

Secure DHCPv4

June 2015

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.	Requirements Language and Terminology	3
3.	Security Overview of DHCPv4	3
4.	Overview of Secure DHCPv4 Mechanism with Public Key	4
4.1.	New Components	5
4.2.	Support for Algorithm Agility	6
4.3.	Applicability	6
5.	Extensions for Secure DHCPv4	7
5.1.	Public Key Option	7
5.2.	Certificate Option	8
5.3.	Signature Option	9
5.4.	Timestamp Option	11
5.5.	Status Codes	11
6.	Processing Rules and Behaviors	12
6.1.	Processing Rules of Sender	12
6.2.	Processing Rules of Recipient	13
6.3.	Processing Rules of Relay Agent	16
6.4.	Timestamp Check	16
7.	Security Considerations	17
8.	IANA Considerations	19
9.	Acknowledgements	20
10.	Change log [RFC Editor: Please remove]	20
11.	References	21
11.1.	Normative References	21
11.2.	Informative References	22
	Authors' Addresses	22

[1.](#) Introduction

The Dynamic Host Configuration Protocol version 4 (DHCPv4, [[RFC2131](#)]) enables DHCPv4 servers to pass configuration parameters and offers configuration flexibility. If not being secured, DHCPv4 is vulnerable to various attacks, particularly spoofing attacks.

This document analyzes the security issues of DHCPv4 in details.
This document provides mechanisms for improving the security of
DHCPv4 between client and server:

- o the identity of a DHCPv4 message sender, which can be a DHCPv4 server or a client, can be verified by a recipient.
- o the integrity of DHCPv4 messages can be checked by the recipient of the message.
- o anti-replay protection based on timestamps.

Note: this secure mechanism in this document does not protect the relay-relevant options, either added by a relay agent toward a server or added by a server toward a relay agent, are considered less vulnerable, because they are only transported within operator networks.

The security mechanisms specified in this document is based on sender's public/private key pairs or certificates with associated private keys. It also integrates message signatures for the integrity and timestamps for anti-replay. The sender authentication procedure using certificates defined in this document depends on deployed Public Key Infrastructure (PKI, [RFC5280](#)). However, the deployment of PKI is out of the scope of this document.

Secure DHCPv4 is applicable in environments where physical security on the link is not assured (such as over wireless) and attacks on DHCPv4 are a concern.

[2.](#) Requirements Language and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119](#) when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119](#) key words.

[3.](#) Security Overview of DHCPv4

DHCPv4 is a client/server protocol that provides managed configuration of devices. It enables a DHCPv4 server to automatically configure relevant network parameters on clients.

Although [\[RFC3118\]](#) provide an optional DHCPv4 authentication mechanism. It depends on the client's key that is "initially distributed to the client through some out-of-band mechanism", and "the DHCPv4 server MUST know the keys for all authorized clients", [Section 5.4 of \[RFC3118\]](#). However, [\[RFC3118\]](#) does not provides no mechanism for distributing the client's key.

For the keyed hash function, there are two key management mechanisms. The first one is a key management done out of band, usually through some manual process. The second approach is to use Public Key Infrastructure (PKI).

As an example of the first approach, operators can set up a key database for both servers and clients from which the client obtains a key before running DHCPv4. Manual key distribution runs counter to the goal of minimizing the configuration data needed at each host.

In comparison, the security mechanism defined in this document allows the public key database on the client or sever to be populated opportunistically or manually, depending on the degree of confidence desired in a specific application. PKI security mechanism is simpler in the local key management respect.

[4.](#) Overview of Secure DHCPv4 Mechanism with Public Key

This document introduces a mechanism that uses public key signatures as a mechanism for securing the DHCPv4 protocol. In order to enable DHCPv4 clients and servers to perform mutual authentication without previous key deployment, this solution provides a DHCPv4 client/server authentication mechanism based on public/private key pairs and, optionally, PKI certificates. The purpose of this design is to make it easier to deploy DHCPv4 authentication and provides protection of DHCPv4 message within the scope of whatever trust relationship exists for the particular key used to authenticate the message.

In this document, we introduce a public key option, a certificate option, a signature option and a timestamp option with corresponding verification mechanisms. A DHCPv4 message can include a public key option, and carrying a digital signature and a timestamp option. The signature can be verified using the supplied public key. The recipient processes the payload of the DHCPv4 message only if the validation is successful: the signature validates, and a trust relationship exists for the key. Alternatively, a DHCPv4 message can include a certificate option, and also carrying a digital signature and a timestamp option. The signature can be verified by the recipient. The recipient processes the payload of the DHCPv4 message only if the validation is successful: the certificate validates, and some trust relationship exists on the recipient for the provided certificate. The end-to-end security protection can be bidirectional, covering messages from servers to clients and from clients to servers. Additionally, the optional timestamp mechanism provides anti-replay protection.

A trust relationship for a public key can be the result either of a Trust-on-first-use (TOFU) policy, or a list of trusted keys configured on the recipient.

A trust relationship for a certificate could also be treated either as TOFU or configured in a list of trusted certificate authorities, depending on the application.

TOFU can be used by a client to authenticate a server and its messages. It can be deployed without a pre-established trust relationship between the client and the server. It can be used for all DHCPv4 messages, and the same single key can be used for all clients since the server does not send a secret in plain text on the wire. Overall this will provide a reasonable balance of easy deployment and moderate level of security, as long as the risk of the attack window on the first use is acceptable.

TOFU can also be used by a server to protect an existing DHCPv4 session with a particular client by preventing a malicious client from hijacking the session. In this case the server does not even have to store the client's public key or certificate after the session; it only has to remember the public key during that

particular session and check if it can verify received messages with that key. This type of authentication can be deployed without a pre-established trust relationship.

If authentication has to be provided from the initial use, the Secure DHCPv4 mechanism needs some infrastructure such as PKI so the recipient of a public key or certificate can verify it securely. It is currently a subject of further study how such an infrastructure can be integrated to DHCPv4 in a way it makes the deployment easier.

The signature on a Secure DHCPv4 message can be expected to significantly increase the size of the message. One example is normal DHCPv4 message length plus a 1 KB for a X.509 certificate and signature and 256 Byte for a signature. Packet fragments are highly possible. In practise, the total length would be various in a large range. Hence, deployment of Secure DHCPv4 should also consider the issues of IP fragment, PMTU, etc.

[4.1.](#) New Components

The components of the solution specified in this document are as follows:

- o Servers and clients using public keys in their secure DHCPv4 messages generate a public/private key pair. A new DHCPv4 option that carries the public key is defined.

- o Servers and clients that use certificates first generate a public/private key pair and then obtain a public key certificate from a Certificate Authority that signs the public key. Another new DHCPv4 option is defined to carry the certificate.
- o A signature generated using the private key which is used by the receiver to verify the integrity of the DHCPv4 messages and then the identity of the sender.
- o A timestamp, to detect replayed packet. The secure DHCPv4 nodes need to meet some accuracy requirements and be synced to global time, while the timestamp checking mechanism allows a configurable time value for clock drift. The real time provision is out of scope of this document.

[4.2.](#) Support for Algorithm Agility

Hash functions are used to provide message integrity checks. In order to provide a means of addressing problems that may emerge in the future with existing hash algorithms, as recommended in [\[RFC4270\]](#), this document provides a mechanism for negotiating the use of more secure hashes in the future.

In addition to hash algorithm agility, this document also provides a mechanism for signature algorithm agility.

The support for algorithm agility in this document is mainly a unilateral notification mechanism from sender to recipient. A recipient MAY support various algorithms simultaneously among different senders, and the different senders in a same administrative domain may be allowed to use various algorithms simultaneously. It is NOT RECOMMENDED that the same sender and recipient use various algorithms in a single communication session.

If the recipient does not support the algorithm used by the sender, it cannot authenticate the message. In the client-to-server case, the server SHOULD reply with an AlgorithmNotSupported status code (defined in [Section 5.5](#)). Upon receiving this status code, the client MAY resend the message protected with the mandatory algorithm (defined in [Section 5.3](#)).

[4.3.](#) Applicability

By default, a secure DHCPv4 enabled client or server SHOULD start with secure mode by sending secure DHCPv4 messages. If the recipient is secure DHCPv4 enabled and the key or certificate authority is trusted by the recipient, then their communication would be in secure mode. In the scenario where the secure DHCPv4 enabled client and

server fail to build up secure communication between them, the secure DHCPv4 enabled client MAY choose to send unsecured DHCPv4 message towards the server according to its local policies.

In the scenario where the recipient is a legacy DHCPv4 server that does not support secure mechanism, the DHCPv4 server (for all of known DHCPv4 implementations) would just omit or disregard unknown options (secure options defined in this document) and still process

the known options. The reply message would be unsecured, of course. It is up to the local policy of the client whether to accept such messages. If the client accepts the unsecured messages from the DHCPv4 server, the subsequent exchanges will be in the unsecured mode.

In the scenario where a legacy client sends an unsecured message to a secure DHCPv4 enabled server, there are two possibilities depending on the server policy. If the server's policy requires the authentication, an UnspecFail (value 1, [\[RFC6926\]](#)) error status code, SHOULD be returned. In such case, the client cannot build up the connection with the server. If the server has been configured to support unsecured clients, the server MAY fall back to the unsecured DHCPv4 mode, and reply unsecured messages toward the client; depending on the local policy, the server MAY continue to send the secured reply messages with the consumption of computing resource. The resources allocated for unsecured clients SHOULD be separated and restricted.

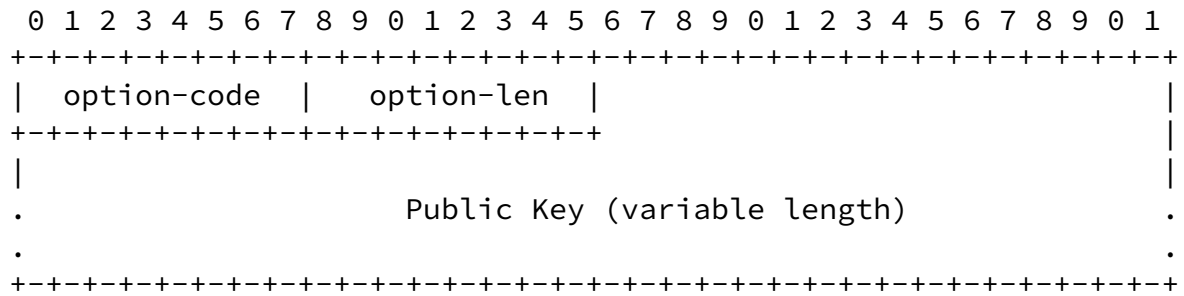
These are all examples of how interactions can go, but there is nothing to prevent clients from behaving adaptively in response to secure messages from servers.

[5.](#) Extensions for Secure DHCPv4

This section describes the extensions to DHCPv4. Four new options have been defined. The new options MUST be supported in the Secure DHCPv4 message exchange.

[5.1.](#) Public Key Option

The Public Key option carries the public key of the sender. The format of the Public Key option is described as follows:



option-code OPTION_PUBLIC_KEY (TBA1).

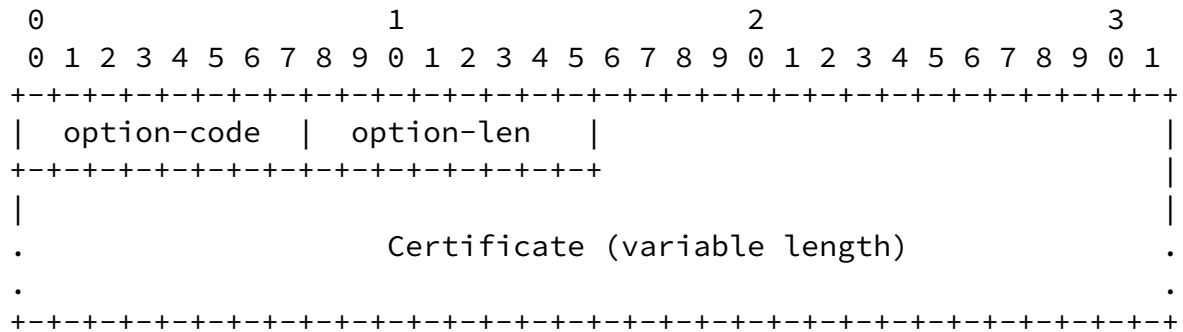
option-len Length of public key in octets.

Public Key A variable-length field containing a SubjectPublicKeyInfo object specified in [\[RFC5280\]](#). The SubjectPublicKeyInfo structure is comprised with a public key and an AlgorithmIdentifier object which is specified in [section 4.1.1.2](#), [\[RFC5280\]](#). The object identifiers for the supported algorithms and the methods for encoding the public key materials (public key and parameters) are specified in [\[RFC3279\]](#), [\[RFC4055\]](#), and [\[RFC4491\]](#).

Note: when the option exceeds 255 octets in size (the maximum size of a single option), multiple instances of the same option are generated according to [\[RFC3396\]](#). And they should be put in sequential order in the same DHCPv4 message.

5.2. Certificate Option

The Certificate option carries the public key certificate of the client. The format of the Certificate option is described as follows:



option-code OPTION_CERTIFICATE (TBA2).

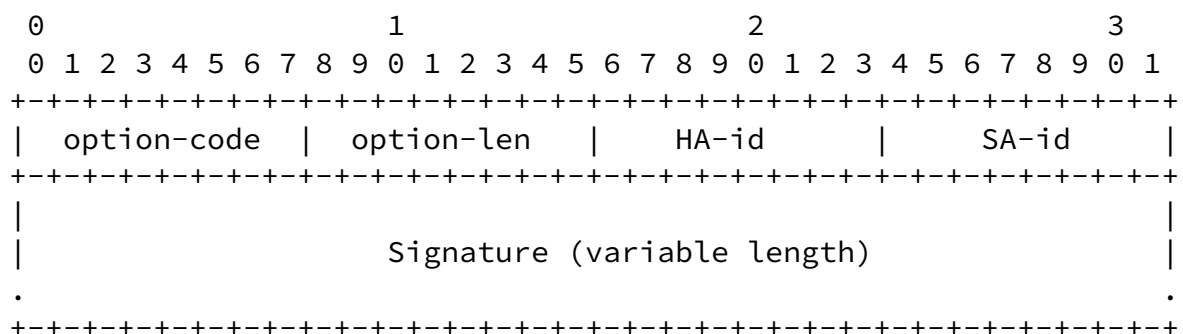
option-len Length of certificate in octets.

Certificate A variable-length field containing certificate. The encoding of certificate and certificate data MUST be in format as defined in [Section 3.6](#), [RFC7296]. The support of X.509 certificate - Signature (4) is mandatory.

Note: when the option exceeds 255 octets in size (the maximum size of a single option), multiple instances of the same option are generated according to [RFC3396]. And they should be put in sequential order in the same DHCPv4 message.

5.3. Signature Option

The Signature option allows a signature that is signed by the private key to be attached to a DHCPv4 message. The Signature option could be any place within the DHCPv4 message while it is logically created after the entire DHCPv4 header and options, except for the Authentication Option. It protects the entire DHCPv4 header and options, including itself, except for the 'giaddr' field, the 'hops' fields, the Authentication Option [RFC3118] and the Relay Agent Information Option [RFC3046]. The format of the Signature option is described as follows:



option-code OPTION_SIGNATURE (TBA3).

Internet-Draft

Secure DHCPv4

June 2015

option-len 2 + Length of Signature field in octets.

HA-id Hash Algorithm id. The hash algorithm is used for computing the signature result. This design is adopted in order to provide hash algorithm agility. The value is from the Hash Algorithm for Secure DHCPv4 registry in IANA. The support of SHA-256 is mandatory. A registry of the initial assigned values is defined in [Section 8](#).

SA-id Signature Algorithm id. The signature algorithm is used for computing the signature result. This design is adopted in order to provide signature algorithm agility. The value is from the Signature Algorithm for Secure DHCPv4 registry in IANA. The support of RSASSA-PKCS1-v1_5 is mandatory. A registry of the initial assigned values is defined in [Section 8](#).

Signature A variable-length field containing a digital signature. The signature value is computed with the hash algorithm and the signature algorithm, as described in HA-id and SA-id. The signature constructed by using the sender's private key protects the following sequence of octets:

1. The DHCPv4 message header with the 'giaddr' and 'hops' fields are set to all 0. It is because DHCPv4 relay agents may change their values.
2. All DHCPv4 options including the Signature option (fill the signature field with zeroes) except for the Authentication Option and the Relay Agent Information Option, if there are any.

The signature field MUST be padded, with all 0, to the next octet boundary if its size is not an even multiple of 8 bits. The padding length depends on the signature algorithm, which is indicated in the

SA-id field.

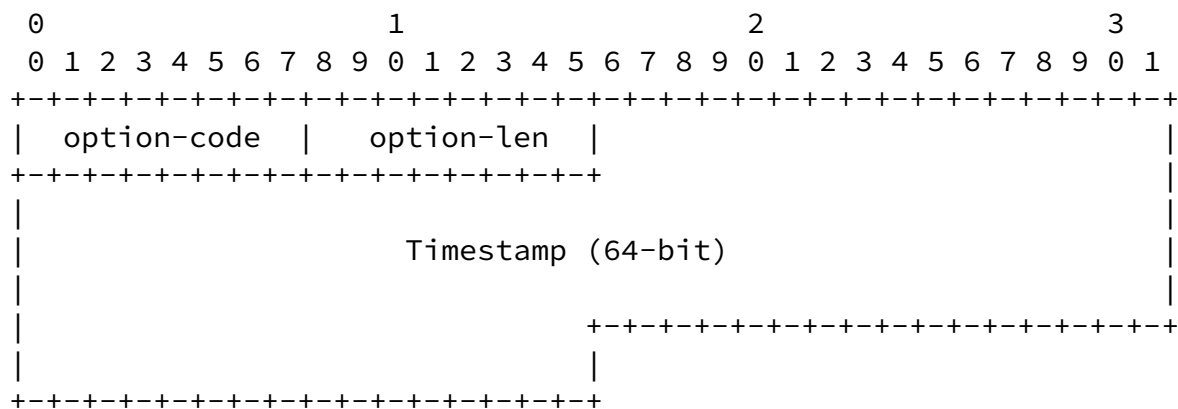
Note: when the option exceeds 255 octets in size (the maximum size of a single option), multiple instances of the same option are generated according to [\[RFC3396\]](#). And they should be put in sequential order in the same DHCPv4 message.

Note: if both signature and authentication option are present, signature option does not protect the Authentication Option. It is

because both options need to apply hash algorithm to whole message, so there must be a clear order and there could be only one last-created option. In order to avoid update auth option, the authors chose not include authentication option in the signature. This design allows the Authentication Option to be created after signature has been calculated and filled with the valid message authentication code (MAC).

[5.4.](#) Timestamp Option

The Timestamp option carries the current time on the sender. It adds the anti-replay protection to the DHCPv4 messages. It is optional.



option-code `OPTION_TIMESTAMP` (TBA4).

option-len 8, in octets.

Timestamp The current time of day (NTP-format timestamp [\[RFC5905\]](#) in UTC (Coordinated Universal Time), a 64-bit unsigned fixed-point number, in seconds

relative to 0h on 1 January 1900.). It can reduce the danger of replay attacks.

[5.5.](#) Status Codes

The following new status codes, see [[RFC6926](#)], are defined. They are carried by the Status Code Option (value 151, [[RFC6926](#)]).

- o AlgorithmNotSupported (TBD5): indicates that the DHCPv4 server does not support algorithms that sender used.
- o AuthenticationFail (TBD6): indicates that the DHCPv4 client fails authentication check.
- o TimestampFail (TBD7): indicates the message from DHCPv4 client fails the timestamp check.

Jiang & Zhang

Expires December 26, 2015

[Page 11]

Internet-Draft

Secure DHCPv4

June 2015

- o SignatureFail (TBD8): indicates the message from DHCPv4 client fails the signature check.

[6.](#) Processing Rules and Behaviors

This section only covers the scenario where both DHCPv4 client and server are secure enabled.

[6.1.](#) Processing Rules of Sender

The sender of a Secure DHCPv4 message could be a DHCPv4 server or a DHCPv4 client.

The sender must have a public/private key pair in order to create Secure DHCPv4 messages. The sender may also have a public key certificate, which is signed by a CA assumed to be trusted by the recipient, and its corresponding private key.

To support Secure DHCPv4, the Secure DHCPv4 enabled sender MUST construct the DHCPv4 message following the rules defined in [[RFC2131](#)].

A Secure DHCPv4 message sent by a DHCPv4 server or client MUST either contain a Public Key option, which MUST be constructed as explained in [Section 5.1](#), or a Certificate option, which MUST be constructed as

explained in [Section 5.2](#).

A Secure DHCPv4 message MUST contain one and only one Signature option, which MUST be constructed as explained in [Section 5.3](#). It protects the message header and all DHCPv4 options except for the 'giaddr' field, the 'hops' fields, the Authentication Option and the Relay Agent Information Option.

A Secure DHCPv4 message SHOULD contain one and only one Timestamp option, which MUST be constructed as explained in [Section 5.4](#). The Timestamp field SHOULD be set to the current time, according to sender's real time clock.

If the sender is a DHCPv4 server and also sends back Relay Agent Information Options to relay agents, it MUST NOT include the Relay Agent Information Options in the computation of signature, as defined in [Section 5.3](#).

If the sender is a DHCPv4 client, in the failure cases, it receives a Reply message with an error status code. The error status code indicates the failure reason on the server side. According to the received status code, the client MAY take follow-up action:

- o Upon receiving an AlgorithmNotSupported error status code, the client SHOULD resend the message protected with one of the mandatory algorithms.
- o Upon receiving an AuthenticationFail error status code, the client is not able to build up the secure communication with the recipient. However, there may be more than one DHCPv4 servers, one of which may send AuthenticationFail and the other of which may succeed. The client MAY use the AuthenticationFail as a hint and switch to other public key certificate if it has another one; but otherwise treat the message containing the status code as if it had not been received. But it SHOULD NOT retry with the same certificate. However, if the client decides to retransmit using the same certificate after receiving AuthenticationFail, it MUST NOT retransmit immediately. [Section 4.1 of \[RFC2131\]](#) has enforced that "the client MUST adopt a retransmission strategy that incorporates a randomized exponential backoff algorithm to determine the delay between retransmissions."

- o Upon receiving a TimestampFail error status code, the client MAY resend the message with an adjusted timestamp according to the returned clock from the DHCPv4 server. The client SHOULD NOT change its own clock, but only compute an offset for the communication session.
- o Upon receiving a SignatureFail error status code, the client MAY resend the message following normal retransmission routines.

6.2. Processing Rules of Recipient

The recipient of a Secure DHCPv4 message could be a DHCPv4 server or a DHCPv4 client. In the failure cases, either DHCPv4 server or client SHOULD NOT process the received message, and the server SHOULD reply with a correspondent error status code, while the client behaves as if no response had been received from that server. The specific behavior depends on the configured local policy.

When receiving a DHCPv4 message, a Secure DHCPv4 enabled recipient SHOULD discard any DHCPv4 messages that meet any of the following conditions:

- o the Signature option is absent,
- o multiple Signature options are present,
- o both the Public Key option and the Certificate option are absent,
- o both the Public Key option and the Certificate option are present.

In such failure, if the recipient is a DHCPv4 server, the server SHOULD reply an UnspecFail error (value 1, [[RFC6926](#)]) status code. If none of the Signature, Public Key or Certificate options is present, the sender MAY be a legacy node or in unsecured mode, then, the recipient MAY fall back to the unsecured DHCPv4 mode if its local policy allows.

The recipient SHOULD first check the support of the hash and signature algorithms that the sender used. If the check fails for a client, the message SHOULD be dropped. If the check fails for a server, the server SHOULD reply with an AlgorithmNotSupported error

status code, defined in [Section 5.5](#), back to the client. If both hash and signature algorithms are supported, the recipient then checks the authority of this sender. The recipient SHOULD also use the same algorithms in the return messages.

If a Public Key option is provided, the recipient SHOULD validate it by finding a matching public key from the local trust public key list, which is pre-configured or recorded from previous communications (TOFU). A local trust public key list is a data table maintained by the recipient. It stores public keys from all senders that are considered trustworthy.

If a Certificate option is provided, the recipient SHOULD validate the certificate according to the rules defined in [[RFC5280](#)]. An implementation may create a local trust certificate record for verified certificates in order to avoid repeated verification procedure in the future. A certificate that finds a match in the local trust certificate list is treated as verified.

When the local policy of the recipient allows the use of TOFU, if a Public Key option is provided but it is not found in the local trust public key list, the recipient MAY accept the public key. The recipient will normally store the key in the local list for subsequent DHCPv4 sessions, but it may not necessarily have to do so depending on the purpose of the authentication (see the case of authenticating a client with TOFU described in [Section 4](#)).

The message that fails authentication check, either because the certificate validation fails or because the public key is not recognized, MUST be dropped. In such failure, the DHCPv4 server SHOULD reply an AuthenticationFail error status code, defined in [Section 5.5](#), back to the client.

The recipient MAY choose to further process messages from a sender when there is no matched public key. When a message is authenticated using a key that has not previously been seen, the recipient may, if

permitted by policy, treat the sender as trustworthy and record the key for future use (i.e, TOFU).

At this point, the recipient has either recognized the authentication

of the sender, or decided to drop the message. The recipient MUST now authenticate the sender by verifying the signature and checking timestamp (see details in [Section 6.4](#)), if there is a Timestamp option. The order of two procedures is left as an implementation decision. It is RECOMMENDED to check timestamp first, because signature verification is much more computationally expensive. Depending on server's local policy, the message without a Timestamp option MAY be acceptable or rejected. If the server rejects such a message, a TimestampFail error status code, defined in [Section 5.5](#), should be sent back to the client. The reply message that carries the TimestampFail error status code SHOULD carry a timestamp option, which indicates the server's clock for the client to use.

The signature field verification MUST show that the signature has been calculated as specified in [Section 5.3](#). Only the messages that get through both the signature verifications and timestamp check (if there is a Timestamp option) are accepted as secured DHCPv4 messages and continue to be handled for their contained DHCPv4 options as defined in [\[RFC2131\]](#). Messages that do not pass the above tests MUST be discarded or treated as unsecured messages. In the case the recipient is DHCPv4 server, the DHCPv4 server SHOULD reply a SignatureFail error status code, defined in [Section 5.5](#), for the signature verification failure; or a TimestampFail error status code, defined in [Section 5.5](#), for the timestamp check failure, back to the client.

Furthermore, the node that supports the verification of the Secure DHCPv4 messages MAY impose additional constraints for the verification. For example, it may impose limits on minimum and maximum key lengths.

Minbits The minimum acceptable key length for public keys. An upper limit MAY also be set for the amount of computation needed when verifying packets that use these security associations. The appropriate lengths SHOULD be set according to the signature algorithm and also following prudent cryptographic practice. For example, minimum length 1024 and upper limit 2048 may be used for RSA [\[RSA\]](#).

A Relay-forward or Relay-reply message with any Public Key, Certificate or the Signature option is invalid. The message MUST be discarded silently.

[6.3.](#) Processing Rules of Relay Agent

To support Secure DHCPv4, relay agents just need to follow the same processing rules defined in [\[RFC2131\]](#). There is nothing more the relay agents have to do, either verify the messages from client or server, or add any secure DHCPv4 options. Actually, by definition in this document, relay agents SHOULD NOT add any secure DHCPv4 options.

[6.4.](#) Timestamp Check

In order to check the Timestamp option, defined in [Section 5.4](#), recipients SHOULD be configured with an allowed timestamp Delta value, a "fuzz factor" for comparisons, and an allowed clock drift parameter. The recommended default value for the allowed Delta is 300 seconds (5 minutes); for fuzz factor 1 second; and for clock drift, 0.01 second.

Note: the Timestamp mechanism is based on the assumption that communication peers have roughly synchronized clocks, with certain allowed clock drift. So, accurate clock is not necessary. If one has a clock too far from the current time, the timestamp mechanism would not work.

To facilitate timestamp checking, each recipient SHOULD store the following information for each sender, from which at least one accepted secure DHCPv4 message is successfully verified (for both timestamp check and signature verification):

- o The receive time of the last received and accepted DHCPv4 message. This is called RDlast.
- o The timestamp in the last received and accepted DHCPv4 message. This is called TSlast.

A verified (for both timestamp check and signature verification) secure DHCPv4 message initiates the update of the above variables in the recipient's record.

Recipients MUST check the Timestamp field as follows:

- o When a message is received from a new peer (i.e., one that is not stored in the cache), the received timestamp, TSnew, is checked, and the message is accepted if the timestamp is recent enough to the reception time of the packet, RDnew:

$$-\text{Delta} < (\text{RDnew} - \text{TSnew}) < +\text{Delta}$$

After the signature verification also succeeds, the RDnew and TSnew values SHOULD be stored in the cache as RDlast and TSlast.

- o When a message is received from a known peer (i.e., one that already has an entry in the cache), the timestamp is checked against the previously received Secure DHCPv4 message:

$$TS_{new} + fuzz > TS_{last} + (RD_{new} - RD_{last}) \times (1 - drift) - fuzz$$

If this inequality does not hold or $RD_{new} < RD_{last}$, the recipient SHOULD silently discard the message. If, on the other hand, the inequality holds, the recipient SHOULD process the message.

Moreover, if the above inequality holds and $TS_{new} > TS_{last}$, the recipient SHOULD update RDlast and TSlast after the signature verification also successes. Otherwise, the recipient MUST NOT update RDlast or TSlast.

An implementation MAY use some mechanism such as a timestamp cache to strengthen resistance to replay attacks. When there is a very large number of nodes on the same link, or when a cache filling attack is in progress, it is possible that the cache holding the most recent timestamp per sender will become full. In this case, the node MUST remove some entries from the cache or refuse some new requested entries. The specific policy as to which entries are preferred over others is left as an implementation decision.

An implementation MAY statefully record the latest timestamps from senders. In such implementation, the timestamps MUST be strictly monotonously increasing. This is reasonable given that DHCPv4 messages are rarely misordered.

[7.](#) Security Considerations

This document provides new security features to the DHCPv4 protocol.

Using public key based security mechanism and its verification mechanism in DHCPv4 message exchanging provides the authentication and data integrity protection. Timestamp mechanism provides anti-replay function.

The Secure DHCPv4 mechanism is based on the pre-condition that the recipient knows the public key of the sender or the sender's public key certificate can be verified through a trust CA. Clients may discard the DHCPv4 messages from unknown/unverified servers, which may be fake servers; or may prefer DHCPv4 messages from known/verified servers over unsigned messages or messages from unknown/unverified servers. The pre-configuration operation also needs to be

protected, which is out of scope. The deployment of PKI is also out of scope.

When a recipient first encounters a new public key, it may also store the key using a Trust On First Use policy. If the sender that used that public key is in fact legitimate, then all future communication with that sender can be protected by storing the public key. This does not provide complete security, but it limits the opportunity to mount an attack on a specific recipient to the first time it communicates with a new sender.

When using TOFU, if the recipient automatically and unlimitedly stores the public key, an attacker could force the recipient to exhaust the storage by sending DHCPv4 messages with many different keys. There are several possible ways to address this concern:

First, the new public key should only be stored after the signature and timestamp validations succeed. It does not prevent the attack itself, but will at least increase the cost of mounting the attack. Another approach is that as long as a client recipient has an uninterrupted connection to a particular network medium, it could limit the number of keys that it will remember as a result of messages received on that medium. Network events like a link state transition would clear the counter, but there might also need to be a counter based on absolute time. In addition, there should probably be a mechanism for purging keys that have only been seen once after a certain period.

Downgrade attacks cannot be avoided if nodes are configured to accept both secured and unsecured messages. A future specification may provide a mechanism on how to treat unsecured DHCPv4 messages.

[RFC6273] has analyzed possible threats to the hash algorithms used

in SEND. Since the Secure DHCPv4 defined in this document uses the same hash algorithms in similar way to SEND, analysis results could be applied as well: current attacks on hash functions do not constitute any practical threat to the digital signatures used in the signature algorithm in the Secure DHCPv4.

A server, whose local policy accepts messages without a Timestamp option, may have to face the risk of replay attacks.

A window of vulnerability for replay attacks exists until the timestamp expires. Secure DHCPv4 nodes are protected against replay attacks as long as they cache the state created by the message containing the timestamp. The cached state allows the node to protect itself against replayed messages. However, once the node flushes the state for whatever reason, an attacker can re-create the

state by replaying an old message while the timestamp is still valid. In addition, the effectiveness of timestamps is largely dependent upon the accuracy of synchronization between communicating nodes. However, how the two communicating nodes can be synchronized is out of scope of this work.

Attacks against time synchronization protocols such as NTP [[RFC5905](#)] may cause Secure DHCPv4 nodes to have an incorrect timestamp value. This can be used to launch replay attacks, even outside the normal window of vulnerability. To protect against these attacks, it is recommended that Secure DHCPv4 nodes keep independently maintained clocks or apply suitable security measures for the time synchronization protocols.

One more consideration is that this protocol does reveal additional client information in their certificate. It means less privacy. In current practice, the client privacy and the client authentication are mutually exclusive.

[8.](#) IANA Considerations

This document defines four new DHCPv4 [[RFC2131](#)] options. The IANA is requested to assign values for these four options from the DHCPv4 Option Codes table of the DHCPv4 Parameters registry maintained in <http://www.iana.org/assignments/bootp-dhcp-parameters>. The four options are:

The Public Key Option (TBA1), described in [Section 5.1](#).

The Certificate Option (TBA2), described in [Section 5.2](#).

The Signature Option (TBA3), described in [Section 5.3](#).

The Timestamp Option (TBA4), described in [Section 5.4](#).

The IANA is also requested to add two new registry tables to the DHCPv4 Parameters registry maintained in <http://www.iana.org/assignments/bootp-dhcp-parameters>. The two tables are the Hash Algorithm for Secure DHCPv4 table and the Signature Algorithm for Secure DHCPv4 table.

Initial values for these registries are given below. Future assignments are to be made through Standards Action [[RFC5226](#)]. Assignments for each registry consist of a name, a value and a RFC number where the registry is defined.

Hash Algorithm for Secure DHCPv4. The values in this table are 8-bit unsigned integers. The following initial values are assigned for Hash Algorithm for Secure DHCPv4 in this document:

Name	Value	RFCs
SHA-256	0x01	this document
SHA-512	0x02	this document

Signature Algorithm for Secure DHCPv4. The values in this table are 8-bit unsigned integers. The following initial values are assigned for Signature Algorithm for Secure DHCPv4 in this document:

Name	Value	RFCs
RSASSA-PKCS1-v1_5	0x01	this document

IANA is requested to assign the following new DHCPv4 Status Codes, defined in [Section 5.5](#), in the DHCPv4 Parameters registry maintained

in <http://www.iana.org/assignments/bootp-dhcp-parameters:>

Code	Name	Reference
TBD5	AlgorithmNotSupported	this document
TBD6	AuthenticationFail	this document
TBD7	TimestampFail	this document
TBD8	SignatureFail	this document

9. Acknowledgements

This document is developed based on the efforts from its brother document Secure DHCPv6 [[I-D.ietf-dhc-sedhcpv6](#)]. Therefore, the authors would like to thank these who helped to develop both documents: Bernie Volz, Ted Lemon, Ralph Droms, Jari Arkko, Sean Turner, Stephen Kent, Thomas Huth, David Schumacher, Francis Dupont, Tomek Mrugalski, Gang Chen, Qi Sun, Suresh Krishnan, Fred Templin, Robert Elz and other members of the IETF DHC working group for their valuable comments.

This document was produced using the xml2rfc tool [[RFC2629](#)].

10. Change log [RFC Editor: Please remove]

[draft-jiang-dhc-sedhcpv4-01](#): change according to the latest change in Secure DHCPv6 during AD review and the comments received during IETF 92, 2015-06-18.

Jiang & Zhang Expires December 26, 2015 [Page 20]

Internet-Draft Secure DHCPv4 June 2015

[draft-jiang-dhc-sedhcpv4-00](#): original version, this draft is largely based on a mature counterpart, Secure DHCPv6, [draft-ietf-dhc-secure-dhcpv6](#). 2015-01-29.

11. References

11.1. Normative References

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

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", [RFC](#)

[2131](#), March 1997.

- [RFC3046] Patrick, M., "DHCP Relay Agent Information Option", [RFC 3046](#), January 2001.
- [RFC3118] Droms, R. and W. Arbaugh, "Authentication for DHCP Messages", [RFC 3118](#), June 2001.
- [RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 3279](#), April 2002.
- [RFC3396] Lemon, T. and S. Cheshire, "Encoding Long Options in the Dynamic Host Configuration Protocol (DHCPv4)", [RFC 3396](#), November 2002.
- [RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 4055](#), June 2005.
- [RFC4491] Leontiev, S. and D. Shefanovski, "Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Internet X.509 Public Key Infrastructure Certificate and CRL Profile", [RFC 4491](#), May 2006.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 5280](#), May 2008.

- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", [RFC 5905](#), June 2010.
- [RFC6926] Kinnear, K., Stapp, M., Desetti, R., Joshi, B., Russell,

N., Kurapati, P., and B. Volz, "DHCPv4 Bulk Leasequery", [RFC 6926](#), April 2013.

- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, [RFC 7296](#), October 2014.

[11.2.](#) Informative References

- [I-D.ietf-dhc-sedhcpv6]
Jiang, S., Shen, S., Zhang, D., and T. Jinmei, "Secure DHCPv6", [draft-ietf-dhc-sedhcpv6-08](#) (work in progress), June 2015.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", [RFC 2629](#), June 1999.
- [RFC4270] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", [RFC 4270](#), November 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 2008.
- [RFC6273] Kukec, A., Krishnan, S., and S. Jiang, "The Secure Neighbor Discovery (SEND) Hash Threat Analysis", [RFC 6273](#), June 2011.
- [RSA] RSA Laboratories, "RSA Encryption Standard, Version 2.1, PKCS 1", November 2002.

Authors' Addresses

Sheng Jiang (editor)
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
CN

Email: jiangsheng@huawei.com

Dacheng Zhang
Beijing, 100095 100025
P.R. China

Email: dacheng.zhang@gmail.com

