Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 14, 2015

R. Van Rein
InternetWide.org

October 11, 2014

KRB5-KDH: Cryptographically binding Kerberos5 with Diffie-Hellman
draft-vanrein-krb5-kdh-00

Abstract

This specification extends Kerberos5 with primitives that create a cryptographic binding between Kerberos5 authentication and Diffie-Hellman encryption. This yields their combined advantages.

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 14, 2015.

Table of Contents

1. Introduction

Kerberos5 lends itself well to infrastructure-supported mutual authentication, which is mainly used on intranets. It provides additional facilities to authenticate across realm boundaries, in which case the control over the network infrastructure used is often distributed, and may in extreme usecases be as diverse as the Internet.

In such distributed-control scenario's, it is attractive to enhance the encryption facilities of Kerberos with support for Perfect Forward Secrecy. Diffie-Hellman key exchange [RFC2631] can be used to incorporate this desirable property, but its lack of authentication must then be overcome by cryptographically binding it to an authentication mechanism, for which Kerberos itself would be suited.

Kerberos-with-Diffie-Hellman, or KDH for short, defines Kerberos5 extensions and usage patterns that support this cryptographic binding between principals. Specifically, the session key will be replaced with one based on a Diffie-Hellman key exchange that is built into AP-REQ and AP-REP messages. A new ticket flag signals if a foreign principal can be expected to implement the KDH mechanism.

There is no need to change the GSS-API or code using it to deploy this mechanism. Implementations of GSS-API can choose to automatically implement the additional subkey mechanism against remote principals that support it; the KDC administrator can use the new ticket flag to signal that rejection of the KDH mechanism by given principals is not to be expected.

2. KDH Subkey Specification

When a client principal initiates mutual authentication against a server principal, it sends an AP-REQ message, containing a ticket obtained from the KDC. Appended to this message is a freshly created authenticator, which may include an optional subkey field. The normal response is an AP-REP message, with a freshly created enc-part appended, which may also include an optional subkey field.

To setup KDH, the subkey fields in these messages are used for the Diffie-Hellman key exchange process. Being part of the authenticator and enc-part fields, the subkey field is subject to encryption with the session key that the KDC created for the two parties. This requires the ability to encrypt on the sending side and to decrypt on the receiving side, and since each side sends, this mutually proves the authenticity of the two principals involved. In other words, the subkey fields are exchanged over an encrypted channel between principals whose authenticity is proven by the Kerberos framework.

Although it would be possible to send one of the subkeys in plain text, there is no good reason to do this, and the message formats of AP-REQ and AP-REP do not provide a suitable place for it.

The subkey fields defined for KDH have encryption types labelled dh-param-pubkey, dh-pubkey, ecdh-param-pubkey and ecdh-pubkey. The dh- and ecdh- are similar, but reflect Diffie-Hellman and its Elliptic Curve variation, respectively. The -param-pubkey version represents both a Diffie-Hellman paramaters type and a public key, where the other -pubkey variations represent just the Diffie-Hellman public key. More formally,

```
   -- The type of a dh-param-pubkey subkey:
   DHParamsPubkey ::= struct {
      INTEGER etype2;      -- Encryption Type
      ServerDHParams params;   -- From RFC 5246
      pktype pubkey;       -- Defined below
   };

   -- The type of a dh-pubkey subkey:
   DHPubkey ::= struct {
      pktype pubkey;       -- Defined below
   };

   -- Type pktype matches dh_Ys or dh_Yc as defined in RFC 5246
   -- Value is a DH public-key value (g^X mod p)
   pktype ::= opaque pubkey<1..2^16-1>;

   -- The type of a ecdh-param-pubkey subkey:
   ECDHParamsPubkey ::= struct {
      INTEGER etype2;      -- Encryption Type
      ECParameters params;         -- From RFC 4492
      ECPoint pubkey;      -- From RFC 4492
   };

   -- The type of a ecdh-pubkey subkey:
   ECDHPubkey ::= struct {
      ECPoint pubkey;      -- From RFC 4492
   }
```

The etype2 fields provide the encryption type to be applied once the (Elliptic-Curve) Diffie-Hellman shared secret has been derived. The possible values for this field are defined in the IANA Registry for Kerberos Encryption Type Numbers.

A Diffie-Hellman key generated for use in the subkey field of an AP-REQ authenticator or AP-REP enc-part SHOULD NOT be used in any other messages, except for literal resends of messages to overcome network connectivity problems. A principal that conforms to this principle enforces variation in the session keys, and so there is no gain to be had from a replay attack by the other side. This means that a conforming principal MAY bypass checks for replay attacks on its side, and also that the current session does not need to be stored in a replay buffer that may still exist in support of non-KDH traffic.

NOTE: The replay buffer is mostly useful in service principals; these may also be contacted by user principals that do not implement KDH. In such cases, the replay buffer must be used. This means that the replay buffer must still be implemented, but it is easier to scale,

and also, a lower performance level may be tolerable if switching to KDH on the client is a realistic alternative.

3. Signalling support for KDH

When KDH is used, the AP-REQ message MUST offer a -param-pubkey encryption type. When the subkey offer is accepted by a remote principal, its AP-REP response MUST contain a subkey with the -pubkey encryption type for the same algorithm (DH or ECDH).

A recipient uncapable of KDH will reject the subkey in the AP-REQ message and report an error instead of the expected AP-REP message. This error is not authenticated in Kerberos5, meaning that a potential of denial-of-service against KDH exists. An opportunistic procedure might try KDH and fall back to Kerberos5 without Diffie-Hellman, but such an approach would make the stack vulnerable to denial-of-service attacks that result in bypassing Perfect Forward Secrecy.

To mitigate this problem on the client side, we introduce a ticket flag have-kdh, that may be set by the KDC on tickets delivered as part of KDC-REP messages. When a ticket for a remote principal has this flag set, there is no reason for a KDH-capable sender to ever send it a message without a subkey following KDH formatting; so, there is never a need to bypass Perfect Forward Secrecy.

On the server side, things are much simpler. An AP-REP message from a conforming implementation MUST incorporate a Diffie-Hellman subkey field if and only if the responded-to AP-REQ message held a Diffie-Hellman subkey field. This is the behaviour of a Kerberos5 implementation that supports KDH; unsupporting implementations would return an empty subkey field or, more likely, respond with an error message. This behaviour is only acceptable from principals whose have-kdh flag is disabled in the KDC.

It is possible to send AP-REQ with KDH to a principal without the have-kdh flag, but the behaviour of unsupporting implementations means that fallback to an AP-REQ without KDH may be required when failures are reported. In cases where this opportunistic approach to KDH succeeds, the advantage is that passive observation is now subject to Perfect Forward Secrecy.

4. Negotiating Diffie-Hellman shared secrets

This specification defines how the introduced subkey encryption types may be used. Other uses SHOULD lead to an encryption type errors, unless future specifications extend upon the behaviour.

4.1. Relation to Kerberos messages

The Diffie-Hellman subkey encryption types are defined for use with the AP-REQ and AP-REP messages. These messages are assumed to occur as a pair, and the permitted combinations of these messages with encryption types is defined in the following table:

AP-REQ enctype	AP-REP enctype	Meaning
dh-params-pubkey	dh-pubkey	Construction of shared secret
dh-params-pubkey	dh-params-pubkey	Rejected parameters; new suggestion
dh-params-pubkey	ecdh-params-pubkey	Rejected plain DH; suggest ECDH
ecdh-params-pubkey	ecdh-pubkey	Construction of shared secret
ecdh-params-pubkey	dh-params-pubkey	Rejected ECDH; suggest plain DH

After both AP-REQ and AP-REP have been processed, and their joint meaning in the table above indicates that a shared secret can be constructed, then both parties MUST derive that shared secret and continue to communicate using that, employing the encryption type specified in the AP-REQ message's subkey field, notably the etype2 subtype.

The only exception to this enforced encryption will be any further AP-REQ and AP-REP messages which MUST NOT be subjected to the newly derived subkey, but to the session key provided by the KDC.

These semantics of enforced encryption are required with every use of the syntax introduced by this specification.

4.2. Negotiating Diffie-Hellman Parameters

Given that an authenticated source supplies Diffie-Hellman parameters as part of the subkey field, they can be assumed to be protected from rogue influence that would jeapourdise their security. Still, the option exists that principals' realms enforce different policies. This raises the issue of negotiation about Diffie-Hellman parameters.

In the previous subsection, situations were described that reject the algorithm (plain Diffie-Hellman or Elliptic-Curve Diffie-Hellman) and that reject the parameters suggested within the algorithm. These usage forms represent a rejection of the subkey suggestion in an AP-

REQ message. If left unhandled, the client's subkey field may be considered rejected by the server, but it is preferred that the client instead engages in another AP-REQ / AP-REP exchange based on the suggested response.

Most of this discussion comes down to desired minimum modulus, generator and generator-order sizes (for reasons of security) and maximum modulus sizes (for reasons of efficiency, and perhaps to counter denial-of-service on public services). In addition, there may be a whitelist or blacklist of certain Diffie-Hellman parameters. We suggest that implementations SHOULD aim to accept at least all RFC-published Diffie-Hellman parameters [RFC5114] that have an acceptable modulus, generator and generator-order, inasfar as they have not been publicly withdrawn. In addition, we suggest that offered Diffie-Hellman parameters are selected from RFC-published sources of acceptable sizes that have not been publicly withdrawn.

If a service receives Diffie-Hellman parameters that it dislikes, then it includes newly proposed Diffie-Hellman parameters in the response. If the proposal is rejected because aspects of the parameters fall under a set minimum, then the response should propose the smallest acceptable parameters. If the proposal is rejected because the modulus is too long, then the response should propose parameters with the longest acceptable modulus (and generator). If the proposal is rejected because the keys are blacklisted and/or not whitelisted, then the response should propose an alternative whose modulus and generator are "similar", according to the service's interpretation.

Each participant MAY impose a maximum on the number of iterations. And all participants SHOULD be liberal in what they accept, and conservative in what they send. This should mean that most the negotiations do find a suitable configuration.

5. Impact on GSS-API

This specification does not alter the GSS-API interface. The only thing that requires a change to integrate KDH is the Kerberos5 implementation of GSS-API.

When requesting a principal ticket, a supporting implementation SHOULD send an KDC-REQ message including the have-kdh ticket flag. According to the general behaviour of Kerberos5, the KDC will not return the flag if it either does not understand the flag, or if it rejects it on grounds of the KDC policy. Only when requested in the KDC-REQ and only if the requested principal is setup to enable the have-kdh ticket flag, may the flag be set in the ticket returned from the KDC. These remarks apply to all interactions with the KDC, even

when obtaining a Ticket Granting Ticket; so even the interactions with the Ticket Granting Service may be protected under Perfect Forward Secrecy.

Based on the ticket flag have-kdh, a GSS-API implementation can decide whether to supply a subkey field with a Diffie-Hellman public key. More precisely, after sending an AP-REQ message with a KDH subkey it is crystal clear whether to be assured that a corresponding subkey will be sent back in the AP-REQ message, after which it is possible to continue with the Diffie-Hellman shared secret.

Note that sending the subkey field with a Diffie-Hellman public key implies a responsibility to actually use it after the AP-REQ and AP-REP messages have been exchanged. The KDC-supplied session key MUST NOT be used after such an exchange, except for new AP-REQ and AP-REP messages. If the remote principal would use the KDC-supplied session key, it SHOULD receive an appropriate error message in return.

6. Efficiency Considerations

The Kerberos5 mechanism is among the most efficient frameworks for secure authentication in the World. This is a direct result of its reliance on symmetric cryptography. The one downside to this is that a user must stay in contact with the KDC to request new tickets.

As for Perfect Forward Secrecy, this currently requires Diffie-Hellman key exchanges. The minimum amount of work for this algorithm is done as part of KDH.

It is not possible to perform the work done by the KDH combination of mechanisms in either Kerberos or Diffie-Hellman, so the two need to work together. Furthermore, the way the two mechanisms are cryptographically bound together is almost optimal; there is a minimal need to encrypt at least one of the Diffie-Hellman key exchange messages, but encrypting both hardly adds work.

Finally, the use of fresh Diffie-Hellman keys for every authenticator to an AP-REQ and for every enc-part to an AP-REP with the encryption types introduced in this specification has the benefit that a replay cache can be bypassed. This saves a complex structure that is difficult to integrate into large-scale / redundant principal implementations. In cases where the replay cache is not always used, at least this complex structure is used less, making it weigh only on the applications that do not implement Diffie-Hellman key exchange, and only those will need to fill the replay cache.

7. Privacy Considerations

The information shared publicly by this protocol does not exceed what is published by other Kerberos5 implementations. The only added field is the subkey field, whose presence may be inferred from the message size, but its value is invisible to third parties because Kerberos encrypts it.

Overall, the privacy of the principals' sessions improves when KDH is applied, because future breakage of Kerberos5 credentials would not enable the decryption of a recorded session.

8. Security Considerations

The Diffie-Hellman key exchange can be performed in plain text, provided that it is authenticated. Kerberos5 implements authentication through encryption of the text or its secure hash; KDH encrypts the entire Diffie-Hellman key exchange for reasons of protocol simplicity.

In Kerberos5, session keys are supplied by the KDC. This realm service must be properly shielded from outside attacks, because it is a vital point in the Kerberos5 infrastructure.

The importance of KDC security does not necessarily reduce with KDH; parties that can read the KDC-generated session key between a pair of principals are in a position to mount an undetectable man-in-the-middle attack on the session even when KDH is added.

When client and server are in different realms, they have cross-signed directly or through a chain of KDC's, and all KDC's involved are potential places where the session key could be determined. The weakest KDC in the chain then defines the security of the entire chain.

Kerberos5 has introduced numerous refinements that are highly practical in daily use. One worth noting is S4U2Proxy, under which a service can upgrade a received ticket to one with which it can be a client using a ticket with the client's authenticated name. Such provisions are usually limited in the KDC through Constrained Delegation, but it nonetheless introduces an extra degree of freedom for attackers. Especially interesting is the combination with S4U2Self, which allows a service to obtain a client ticket without proving (in the Kerberos sense) that the client has actually authenticated to, or even contacted the server.

Kerberos requires accurate clocks in order to operate securely; without it, once-used and since-forgotten credentials could be

replayed by an attacker that has been able to recover an old service ticket's session key. This problem is worsened in cross-realm scenario's where clock synchronisation may be more difficult to realise. This is overcome when KDH is used with freshly generated Diffie-Hellman keys, but clients are never forced by this specification to send an AP-REQ with a Diffie-Hellman key, so replay attack handling is still needed to serve non-KDH clients securely.

9. IANA Considerations

This specification introduces four Diffie-Hellman exchange formats to be added to the registry of Encryption Type Numbers, representing a subkey field that matches the syntax as specified below:

Number	Kerberos Encryption Type Name	Syntax Corresponding to
TBD	dh-param-pubkey	[THIS SPEC] DHParamsPubkey
TBD	dh-pubkey	[THIS SPEC] DHPubkey
TBD	ecdh-param-pubkey	[THIS SPEC] ECDHParamsPubkey
TBD	ecdh-pubkey	[THIS SPEC] ECDHPubkey

This specification also introduces a new entry in the Kerberos registry for Ticket flag. The flag is named have-kdh:

```
TicketFlags ::= KerberosFlags
    -- have-kdh(TBD)
```

10. Normative References

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

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

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC5114] Lepinski, M. and S. Kent, "Additional Diffie-Hellman Groups for Use with IETF Standards", RFC 5114, January 2008.

Author's Address

Rick van Rein
InternetWide.org
Haarlebrink 5
Enschede, Overijssel 7544 WP
The Netherlands

Email: rick@openfortress.nl