Internet Engineering Task Force
Internet-Draft
Updates: 3961 (if approved)
Intended status: Standards Track
Expires: April 23, 2018

N. McCallum
S. Sorce
R. Harwood
Red Hat, Inc.
G. Hudson
MIT

October 20, 2017

SPAKE Pre-Authentication
draft-ietf-kitten-krb-spake-preauth-02

Abstract

This document defines a new pre-authentication mechanism for the Kerberos protocol that uses a password authenticated key exchange. This document has three goals. First, increase the security of Kerberos pre-authentication exchanges by making offline brute-force attacks infeasible. Second, enable the use of second factor authentication without relying on FAST. This is achieved using the existing trust relationship established by the shared first factor. Third, make Kerberos pre-authentication more resilient against time synchronization errors by removing the need to transfer an encrypted timestamp from the client.

Status of This Memo

This Internet-Draft will expire on April 23, 2018.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Properties of PAKE 3
  1.2. PAKE Algorithm Selection 3
  1.3. PAKE and Two-Factor Authentication 4
  1.4. SPAKE Overview 5
2. Document Conventions 5
3. Prerequisites 6
  3.1. PA-ETYPE-INFO2 6
  3.2. Cookie Support 6
  3.3. More Pre-Authentication Data Required 6
4. Update to Checksum Specifications 6
5. SPAKE Pre-Authentication Message Protocol 7
  5.1. First Pass 7
  5.2. Second Pass 8
  5.3. Third Pass 9
  5.4. Subsequent Passes 10
  5.5. Reply Key Strengthening 10
  5.6. Optimizations 11
6. SPAKE Parameters and Conversions 11
7. Transcript Checksum 12
8. Key Derivation 13
9. Second Factor Types 14
10. Security Considerations 14
  10.1. Unauthenticated Plaintext 14
  10.2. Side Channels 14
  10.3. KDC State 15
  10.4. Dictionary Attacks 16
  10.5. Brute Force Attacks 16
  10.6. Denial of Service Attacks 17
  10.7. Reply-Key Encryption Type 17
  10.8. KDC Authentication 17
11. Assigned Constants 17
12. IANA Considerations 17
  12.1. Kerberos Second Factor Types 18
    12.1.1. Registration Template 18
    12.1.2. Initial Registry Contents 18

  12.2. Kerberos SPAKE Groups 18
    12.2.1. Registration Template 19
    12.2.2. Initial Registry Contents 19
13. References 20
  13.1. Normative References 20
  13.2. Non-normative References 22
Appendix A. ASN.1 Module 23
Appendix B. SPAKE M and N Value Selection 24
Appendix C. Test Vectors 24
Appendix D. Acknowledgements 31
Authors' Addresses 31

1. Introduction

When a client uses PA-ENC-TIMESTAMP (or similar schemes, or the KDC does not require preauthentication), a passive attacker that observes either the AS-REQ or AS-REP can perform an offline brute-force attack against the transferred ciphertext. When the client principal's long-term key is based on a password, offline dictionary attacks can successfuly recover the key, with only modest effort needed if the password is weak.

1.1. Properties of PAKE

Password authenticated key exchange (PAKE) algorithms provide several properties which are useful to overcome this problem and make them ideal for use as a Kerberos pre-authentication mechanism.

1. Each side of the exchange contributes entropy.

2. Passive attackers cannot determine the shared key.

3. Active attackers cannot perform a man-in-the-middle attack.

These properties of PAKE allow us to establish high-entropy encryption keys resistant to offline brute force attack, even when the passwords used are weak (low-entropy).

1.2. PAKE Algorithm Selection

The SPAKE algorithm works by encrypting the public keys of a Diffie-Hellman key exchange with a shared secret. SPAKE was selected for this pre-authentication mechanism for the following properties:

1. Because SPAKE's encryption method ensures that the result is a member of the underlying group, it can be used with elliptic curve cryptography, which is believed to provide equivalent

security levels to finite-field DH key exchange at much smaller key sizes.

2. It can compute the shared key after just one message from each party.

3. It requires a small number of group operations, and can therefore be implemented simply and efficiently.

1.3. PAKE and Two-Factor Authentication

Using PAKE in a pre-authentication mechanism also has another benefit when used as a component of two-factor authentication (2FA). 2FA methods often require the secure transfer of plaintext material to the KDC for verification. This includes one-time passwords, challenge/response signatures and biometric data. Attempting to encrypt this data using the long-term secret results in packets that are vulnerable to offline brute-force attack if either authenticated encryption is used or if the plaintext is distinguishable from random data. This is a problem that PAKE solves for first factor authentication. So a similar technique can be used with PAKE to encrypt second-factor data.

In the OTP pre-authentication [RFC6560] specification, this problem is mitigated by using FAST, which uses a secondary trust relationship to create a secure encryption channel within which pre-authentication data can be sent. However, the requirement for a secondary trust relationship has proven to be cumbersome to deploy and often introduces third parties into the trust chain (such as certification authorities). These requirements lead to a scenario where FAST cannot be enabled by default without sufficient configuration. SPAKE pre-authentication, in contrast, can create a secure encryption channel implicitly, using the key exchange to negotiate a high-entropy encryption key. This key can then be used to securely encrypt 2FA plaintext data without the need for a secondary trust relationship. Further, if the second factor verifiers are sent at the same time as the first factor verifier, and the KDC is careful to prevent timing attacks, then an online brute-force attack cannot be used to attack the factors separately.

For these reasons, this draft departs from the advice given in Section 1 of RFC 6113 [RFC6113] which states that "Mechanism designers should design FAST factors, instead of new pre-authentication mechanisms outside of FAST." However, this pre-authentication mechanism does not intend to replace FAST, and may be used with it to further conceal the metadata of the Kerberos messages.

1.4. SPAKE Overview

The SPAKE algorithm can be broadly described in a series of four steps:

1. Calculation and exchange of the public key

2. Calculation of the shared secret (K)

3. Derivation of an encryption key (K')

4. Verification of the derived encryption key (K')

Higher level protocols must define their own verification step. In the case of this mechanism, verification happens implicitly by a successful decryption of the 2FA data.

This mechanism provides its own method of deriving encryption keys from the calculated shared secret K, for several reasons: to fit within the framework of [RFC3961], to ensure negotiation integrity using a transcript checksum, to derive different keys for each use, and to bind the KDC-REQ-BODY to the pre-authentication exchange.

2. Document Conventions

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].

This document refers to numerous terms and protocol messages defined in [RFC4120].

The terms "encryption type", "required checksum mechanism", and "get_mic" are defined in [RFC3961].

The terms "FAST", "PA-FX-COOKIE", "KDC_ERR_PREAUTH_EXPIRED", "KDC_ERR_MORE_PREAUTH_DATA_REQUIRED", "KDC_ERR_PREAUTH_FAILED", "pre-authentication facility", and "authentication set" are defined in [RFC6113].

The [SPAKE] paper defines SPAKE as a family of two key exchange algorithms differing only in derivation of the final key. This mechanism uses a derivation similar to the second algorithm (SPAKE2) with differences in detail. For simplicity, this document refers to the algorithm as "SPAKE". The normative reference for this algorithm is [I-D.irtf-cfrg-spake2].

The terms "ASN.1" and "DER" are defined in [CCITT.X680.2002] and [CCITT.X690.2002] respectively.

3. Prerequisites

3.1. PA-ETYPE-INFO2

This mechanism requires the initial KDC pre-authentication state to contain a singular reply key. Therefore, a KDC which offers SPAKE pre-authentication as a stand-alone mechanism MUST supply a PA-ETYPE-INFO2 value containing a single ETYPE-INFO2-ENTRY, as described in [RFC6113] section 2.1. PA-ETYPE-INFO2 is specified in [RFC4120] section 5.2.7.5.

3.2. Cookie Support

KDCs which implement SPAKE pre-authentication MUST have some secure mechanism for retaining state between AS-REQs. For stateless KDC implementations, this method will most commonly be an encrypted PA-FX-COOKIE. Clients which implement SPAKE pre-authentication MUST support PA-FX-COOKIE, as described in [RFC6113] section 5.2.

3.3. More Pre-Authentication Data Required

Both KDCs and clients which implement SPAKE pre-authentication MUST support the use of KDC_ERR_MORE_PREAUTH_DATA_REQUIRED, as described in [RFC6113] section 5.2.

4. Update to Checksum Specifications

[RFC3961] section 4 specifies the Kerberos checksum algorithm profile. It does not require checksums to be deterministic. In practice, DES-based checksum types (deprecated by [RFC6649]) use a random confounder; all other current checksum types are deterministic.

Future checksum types required by an encryption type MUST be deterministic. All future checksum types SHOULD be deterministic.

This mechanism requires a deterministic checksum type for the transcript checksum. Therefore, a KDC MUST NOT offer this mechanism if the initial reply key is of type des-cbc-crc, des-cbc-md4, or des-cbc-md5.

5. SPAKE Pre-Authentication Message Protocol

This mechanism uses the reply key and provides the Client Authentication and Strengthening Reply Key pre-authentication facilities ([RFC6113] section 3). When the mechanism completes successfully, the client will have proved knowledge of the original reply key and possibly a second factor, and the reply key will be strengthened to a more uniform distribution based on the PAKE exchange. This mechanism also ensures the integrity of the KDC-REQ-BODY contents. This mechanism can be used in an authentication set; no pa-hint value is required or defined.

This mechanism negotiates a choice of group for the SPAKE algorithm. Groups are defined in the IANA "Kerberos SPAKE Groups" registry created by this document. Clients and KDCs MUST implement the edwards25519 group, but MAY choose not to offer or accept it by default.

This section will describe the flow of messages when performing SPAKE pre-authentication. We will begin by explaining the most verbose version of the protocol which all implementations MUST support. Then we will describe several optional optimizations to reduce round-trips.

Mechanism messages are communicated using PA-DATA elements within the padata field of KDC-REQ messages or within the METHOD-DATA in the e-data field of KRB-ERROR messages. All PA-DATA elements for this mechanism MUST use the following padata-type:

```
PA-SPAKE  TBD
```

The padata-value for all PA-SPAKE PA-DATA values MUST be empty or contain a DER encoding for the ASN.1 type PA-SPAKE.

```
PA-SPAKE ::= CHOICE {
    support     [0] SPAKESupport,
    challenge   [1] SPAKEChallenge,
    response    [2] SPAKEResponse,
    encdata     [3] EncryptedData,
    ...
}
```

5.1. First Pass

The SPAKE pre-authentication exchange begins when the client sends an initial authentication service request (AS-REQ) without pre-authentication data. Upon receipt of this AS-REQ, a KDC which requires pre-authentication and supports SPAKE SHOULD reply with a

KDC_ERR_PREAUTH_REQUIRED error, with METHOD-DATA containing an empty PA-SPAKE PA-DATA element (possibly in addition to other PA-DATA elements). This message indicates to the client that the KDC supports SPAKE pre-authentication.

5.2. Second Pass

Once the client knows that the KDC supports SPAKE pre-authentication and the client desires to use it, the client will generate a new AS-REQ message containing a PA-SPAKE PA-DATA element using the support choice. This message indicates to the KDC which groups the client prefers for the SPAKE operation. The group numbers are defined in the IANA "Kerberos SPAKE Groups" registry created by this document. The groups sequence is ordered from the most preferred group to the least preferred group.

```
SPAKESupport ::= SEQUENCE {
    groups      [0] SEQUENCE (SIZE(1..MAX)) OF Int32,
    ...
}
```

The client and KDC initialize a transcript checksum (Section 7) and update it with the DER-encoded PA-SPAKE message.

Upon receipt of the support message, the KDC will select a group. The KDC SHOULD choose a group from the groups provided by the support message. However, if the support message does not contain any group that is supported by the KDC, the KDC MAY select another group in hopes that the client might support it. Otherwise, the KDC MUST respond with a KDC_ERR_PREAUTH_FAILED error.

Once the KDC has selected a group, the KDC will reply to the client with a KDC_ERR_MORE_PREAUTH_DATA_REQUIRED error containing a PA-SPAKE PA-DATA element using the challenge choice. The client and KDC update the transcript checksum with the DER-encoded PA-SPAKE message.

```
SPAKEChallenge ::= SEQUENCE {
    group       [0] Int32,
    pubkey      [1] OCTET STRING,
    factors     [2] SEQUENCE (SIZE(1..MAX)) OF SPAKESecondFactor,
    ...
}
```

The group field indicates the KDC-selected group used for all SPAKE calculations as defined in the IANA "Kerberos SPAKE Groups" registry created by this document.

The pubkey field indicates the KDC's public key generated using the M constant in the SPAKE algorithm, with inputs and conversions as specified in Section 6.

The factors field contains an unordered list of second factors which can be used to complete the authentication. Each second factor is represented by a SPAKESecondFactor.

```
SPAKESecondFactor ::= SEQUENCE {
    type        [0] Int32,
    data        [1] OCTET STRING OPTIONAL
}
```

The type field is a unique integer which identifies the second factor type. The factors field of SPAKEChallenge MUST NOT contain more than one SPAKESecondFactor with the same type value.

The data field contains optional challenge data. The contents in this field will depend upon the second factor type chosen.

5.3. Third Pass

Upon receipt of the challenge message, the client will complete its part of of the SPAKE algorithm, generating a public key and computing the shared secret K. The client will then choose one of the second factor types listed in the factors field of the challenge message and gather whatever data is required for the chosen second factor type, possibly using the associated challenge data. Finally, the client will send an AS-REQ containing a PA-SPAKE PA-DATA element using the response choice.

```
SPAKEResponse ::= SEQUENCE {
    pubkey      [0] OCTET STRING,
    factor      [1] EncryptedData, -- SPAKESecondFactor
    ...
}
```

The client and KDC will update the transcript checksum with the pubkey value, and use the resulting checksum for all encryption key derivations.

The pubkey field indicates the client's public key generated using the N constant in the SPAKE algorithm, with inputs and conversions as specified in Section 6.

The factor field indicates the client's chosen second factor data. The key for this field is K'[1] as specified in Section 8. The key usage number for the encryption is KEY_USAGE_SPAKE_FACTOR. The plain

text inside the EncryptedData is an encoding of SPAKESecondFactor. Once decoded, the SPAKESecondFactor contains the type of the second factor and any optional data used. The contents of the data field will depend on the second factor type chosen. The client MUST NOT send a response containing a second factor type which was not listed in the factors field of the challenge message.

When the KDC receives the response message from the client, it will use the pubkey to compute the SPAKE result, derive K'[1], and decrypt the factors field. If decryption is successful, the first factor is successfully validated. The KDC then validates the second factor. If either factor fails to validate, the KDC SHOULD respond with a KDC_ERR_PREAUTH_FAILED error.

If validation of the second factor requires further round-trips, the KDC MUST reply to the client with KDC_ERR_MORE_PREAUTH_DATA_REQUIRED containing a PA-SPAKE PA-DATA element using the encdata choice. The key for the EncryptedData value is K'[2] as specified in Section 8, and the key usage number is KEY_USAGE_SPAKE_FACTOR. The plain text of this message contains a DER-encoded SPAKESecondFactor message. As before, the type field of this message will contain the second factor type, and the data field will optionally contain second factor type specific data.

```
KEY_USAGE_SPAKE_FACTOR                  65
```

5.4. Subsequent Passes

Any number of additional round trips may occur using the encdata choice. The contents of the plaintexts are specific to the second factor type. If a client receives a PA-SPAKE PA-DATA element using the encdata choice from the KDC, it MUST reply with a subsequent AS-REQ with a PA-SPAKE PA-DATA using the encdata choice, or abort the AS exchange.

The key for client-originated encdata messages in subsequent passes is K'[3] as specified in Section 8 for the first subsequent pass, K'[5] for the second, and so on. The key for KDC-originated encdata messages is K'[4] for the first subsequent pass, K'[6] for the second, and so on.

5.5. Reply Key Strengthening

When the KDC has successfully validated both factors, the reply key is strengthened and the mechanism is complete. To strengthen the reply key, the client and KDC replace it with K'[0] as specified in Section 8. The KDC then replies with a KDC-REP message, or continues

on to the next mechanism in the authentication set. There is no final PA-SPAKE PA-DATA message from the KDC to the client.

Reply key strengthening occurs only once at the end of the exchange. The client and KDC MUST use the initial reply key as the base key for all K'[n] derivations.

5.6. Optimizations

The full protocol has two possible optimizations.

First, the KDC MAY reply to the initial AS-REQ (containing no pre-authentication data) with a PA-SPAKE PA-DATA element using the challenge choice, instead of an empty padata-value. In this case, the KDC optimistically selects a group which the client may not support. If the group chosen by the challenge message is supported by the client, the client MUST skip to the third pass by issuing an AS-REQ with a PA-SPAKE message using the response choice. If the KDC's chosen group is not supported by the client, the client MUST initialize and update the transcript checksum with the KDC's challenge message, and then continue to the second pass. Clients MUST support this optimization.

Second, clients MAY skip the first pass and send an AS-REQ with a PA-SPAKE PA-DATA element using the support choice. If the KDC accepts the support message and generates a challenge, it MUST include a PA-ETYPE-INFO2 value within the METHOD-DATA of the KDC_ERR_MORE_PREAUTH_DATA_REQUIRED error response, as the client may not otherwise be able to compute the initial reply key. If the KDC cannot continue with SPAKE (either because initial reply key type is incompatible with SPAKE or because it does not support any of the client's groups) but can offer other pre-authentication mechanisms, it MUST respond with a KDC_ERR_PREAUTH_FAILED error containing METHOD-DATA. A client supporting this optimization MUST continue after a KDC_ERR_PREAUTH_FAILED error as described in [RFC6113] section 2. KDCs MUST support this optimization.

6. SPAKE Parameters and Conversions

Group elements are converted to octet strings using the serialization method defined in the IANA "Kerberos SPAKE Groups" registry created by this document.

The SPAKE algorithm requires constants M and N for each group. These constants are defined in the IANA "Kerberos SPAKE Groups" registry created by this document.

The SPAKE algorithm requires a shared secret input w to be used as a scalar multiplier (see [I-D.irtf-cfrg-spake2] section 2). This value MUST be produced from the initial reply key as follows:

1. Determine the length of the multiplier octet string as defined in the IANA "Kerberos SPAKE Groups" registry created by this document.

2. Compose a pepper string by concatenating the string "SPAKEsecret" and the group number as a big-endian four-byte unsigned binary number.

3. Produce an octet string of the required length using PRF+(K, pepper), where K is the initial reply key and PRF+ is defined in [RFC6113] section 5.1.

4. Convert the octet string to a multiplier scalar using the multiplier conversion method defined in the IANA "Kerberos SPAKE Groups" registry created by this document.

The KDC chooses a secret scalar value x and the client chooses a secret scalar value y. As required by the SPAKE algorithm, these values are chosen randomly and uniformly. The KDC and client MUST NOT reuse x or y values for authentications involving different initial reply keys (see Section 10.3).

7. Transcript Checksum

The transcript checksum is an octet string of length equal to the output length of the required checksum type of the encryption type of the initial reply key. The initial value consists of all bits set to zero.

When the transcript checksum is updated with an octet string input, the new value is the get_mic result computed over the concatenation of the old value and the input, for the required checksum type of the initial reply key's encryption type, using the initial reply key and the key usage number KEY_USAGE_SPAKE_TRANSCRIPT.

In the normal message flow or with the second optimization described in Section 5.6, the transcript checksum is first updated with the client's support message, then the KDC's challenge message, and finally with the client's pubkey value. It therefore incorporates the client's supported groups, the KDC's chosen group, the KDC's initial second-factor messages, and the client and KDC public values. Once the transcript checksum is finalized, it is used without change for all key derivations (Section 8).

If the first optimization described in Section 5.6 is used successfully, the transcript checksum is updated only with the KDC's challenge message and the client's pubkey value.

If first optimization is used unsuccessfully (i.e. the client does not accept the KDC's selected group), the transcript checksum is updated with the KDC's optimistic challenge message, then with the client's support message, then the KDC's second challenge message, and finally with the client's pubkey value.

KEY_USAGE_SPAKE_TRANSCRIPT	66

8. Key Derivation

Implementations MUST NOT use the SPAKE result (denoted by K in Section 2 of SPAKE [I-D.irtf-cfrg-spake2]) directly for any cryptographic operation. Instead, the SPAKE result is used to derive keys K'[n] as defined in this section. This method differs slightly from the method used to generate K' in Section 3 of SPAKE [I-D.irtf-cfrg-spake2].

An input string is assembled by concatenating the following values:

- The fixed string "SPAKEkey".

- The group number as a big-endian four-byte unsigned binary number.

- The encryption type of the initial reply key as a big-endian four-byte unsigned binary number.

- The SPAKE result K, converted to an octet string as specified in Section 6.

- The transcript checksum.

- The KDC-REQ-BODY encoding for the request being sent or responded to. Within a FAST channel, the inner KDC-REQ-BODY encoding MUST be used.

- The value n as a big-endian four-byte unsigned binary number.

The derived key K'[n] has the same encryption type as the initial reply key, and has the value random-to-key(PRF+(initial-reply-key, input-string)). PRF+ is defined in [RFC6113] section 5.1.

9. Second Factor Types

This document defines one second factor type:

SF-NONE 1

This second factor type indicates that no second factor is used. Whenever a SPAKESecondFactor is used with SF-NONE, the data field MUST be omitted. The SF-NONE second factor always successfully validates.

10. Security Considerations

All of the security considerations from SPAKE [I-D.irtf-cfrg-spake2] apply here as well.

10.1. Unauthenticated Plaintext

This mechanism includes unauthenticated plaintext in the support and challenge messages. Beginning with the third pass, the integrity of this plaintext is ensured by incorporating the transcript checksum into the derivation of the final reply key and second factor encryption keys. Downgrade attacks on support and challenge messages will result in the client and KDC deriving different reply keys and EncryptedData keys. The KDC-REQ-BODY contents are also incorporated into key derivation, ensuring their integrity. The unauthenticated plaintext in the KDC-REP message is not protected by this mechanism.

Unless FAST is used, the factors field of a challenge message is not integrity-protected until the response is verified. Second factor types MUST account for this when specifying the semantics of the data field. Second factor data in the challenge should not be included in user prompts, as it could be modified by an attacker to contain misleading or offensive information.

Subsequent factor data, including the data in the response, are encrypted in a derivative of the shared secret K. Therefore, it is not possible to exploit the untrustworthiness of the challenge to turn the client into an encryption or signing oracle, unless the attacker knows the client's long-term key.

10.2. Side Channels

An implementation of this pre-authentication mechanism can have the property of indistinguishability, meaning that an attacker who guesses a long-term key and a second factor value cannot determine whether one of the factors was correct unless both are correct. Indistinguishability is only maintained if the second factor can be

validated solely based on the data in the response; the use of additional round trips will reveal to the attacker whether the long-term key is correct. Indistinguishability also requires that there are no side channels. When processing a response message, whether or not the KDC successfully decrypts the factor field, it must reply with the same error fields, take the same amount of time, and make the same observable communications to other servers.

Both the size of the EncryptedData and the number of EncryptedData messages used for second-factor data (including the factor field of the SPAKEResponse message and messages using the encdata PA-SPAKE choice) may reveal information about the second factor used in an authentication. Care should be taken to keep second factor messages as small and as few as possible.

Any side channels in the creation of the shared secret input w, or in the multiplications wM and wN, could allow an attacker to recover the client long-term key. Implementations MUST take care to avoid side channels, particularly timing channels. Generation of the secret scalar values x and y need not take constant time, but the amount of time taken MUST NOT provide information about the resulting value.

The conversion of the scalar multiplier for the SPAKE w parameter may produce a multiplier that is larger than the order of the group. Some group implementations may be unable to handle such a multiplier. Others may silently accept such a multiplier, but proceed to perform multiplication that is not constant time. This is a minor risk in all known groups, but is a major risk for P-521 due to the extra seven high bits in the input octet string. A common solution to this problem is achieved by reducing the multiplier modulo the group order, taking care to ensure constant time operation.

10.3. KDC State

A stateless KDC implementation generally must use a PA-FX-COOKIE value to remember its private scalar value x and the transcript checksum. The KDC MUST maintain confidentiality and integrity of the cookie value, perhaps by encrypting it in a key known only to the realm's KDCs. Cookie values may be replayed by attackers. The KDC SHOULD limit the time window of replays using a timestamp, and SHOULD prevent cookie values from being applied to other pre-authentication mechanisms or other client principals. Within the validity period of a cookie, an attacker can replay the final message of a pre-authentication exchange to any of the realm's KDCs and make it appear that the client has authenticated.

If an x or y value is reused for pre-authentications involving two different client long-term keys, an attacker who observes both

authentications and knows one of the long-term keys can conduct an offline dictionary attack to recover the other one.

This pre-authentication mechanism is not designed to provide forward secrecy. Nevertheless, some measure of forward secrecy may result depending on implementation choices. A passive attacker who determines the client long-term key after the exchange generally will not be able to recover the ticket session key; however, an attacker who also determines the PA-FX-COOKIE encryption key (if the KDC uses an encrypted cookie) will be able to recover the ticket session key. The KDC can mitigate this risk by periodically rotating the cookie encryption key. If the KDC or client retains the x or y value for reuse with the same client long-term key, an attacker who recovers the x or y value and the long-term key will be able to recover the ticket session key.

10.4. Dictionary Attacks

Although this pre-authentication mechanism is designed to prevent an offline dictionary attack by an active attacker posing as the KDC, such an attacker can attempt to downgrade the client to encrypted timestamp. Client implementations SHOULD provide a configuration option to disable encrypted timestamp on a per-realm basis to mitigate this attack.

If the user enters the wrong password, the client might fall back to encrypted timestamp after receiving a KDC_ERR_PREAUTH_FAILED error from the KDC, if encrypted timestamp is offered by the KDC and not disabled by client configuration. This fallback will enable a passive attacker to mount an offline dictionary attack against the incorrect password, which may be similar to the correct password. Client implementations SHOULD assume that encrypted timestamp and encrypted challenge are unlikely to succeed if SPAKE pre-authentication fails in the second pass and no second factor was used.

Like any other pre-authentication mechanism using the client long-term key, this pre-authentication mechanism does not prevent online password guessing attacks. The KDC is made aware of unsuccessful guesses, and can apply facilities such as password lockout to mitigate the risk of online attacks.

10.5. Brute Force Attacks

The selected group's resistance to offline brute-force attacks may not correspond to the size of the reply key. For performance reasons, a KDC MAY select a group whose brute-force work factor is less than the reply key length. A passive attacker who solves the

group discrete logarithm problem after the exchange will be able to conduct an offline attack against the client long-term key. Although the use of password policies and costly, salted string-to-key functions may increase the cost of such an attack, the resulting cost will likely not be higher than the cost of solving the group discrete logarithm.

10.6. Denial of Service Attacks

Elliptic curve group operations are more computationally expensive than secret-key operations. As a result, the use of this mechanism may affect the KDC's performance under normal load and its resistance to denial of service attacks.

10.7. Reply-Key Encryption Type

This mechanism does not upgrade the encryption type of the initial reply key, and relies on that encryption type for confidentiality, integrity, and pseudo-random functions. If the client long-term key uses a weak encryption type, an attacker might be able to subvert the exchange, and the replaced reply key will also be of the same weak encryption type.

10.8. KDC Authentication

This mechanism does not directly provide the KDC Authentication pre-authentication facility, because it does not send a key confirmation from the KDC to the client. When used as a stand-alone mechanism, the traditional KDC authentication provided by the KDC-REP enc-part still applies.

11. Assigned Constants

The following key usage values are assigned for this mechanism:

KEY_USAGE_SPAKE_TRANSCRIPT	65
KEY_USAGE_SPAKE_FACTOR	66

12. IANA Considerations

The notes for the "Kerberos Checksum Type Numbers" registry should be updated with the following addition: "If the checksum algorithm is non-deterministic, see [this document] Section 4."

This document establishes two registries with the following procedure, in accordance with [RFC5226]:

Registry entries are to be evaluated using the Specification Required method. All specifications must be be published prior to entry inclusion in the registry. There will be a three-week review period by Designated Experts on the krb5-spake-review@ietf.org mailing list. Prior to the end of the review period, the Designated Experts must approve or deny the request. This decision is to be conveyed to both the IANA and the list, and should include reasonably detailed explanation in the case of a denial as well as whether the request can be resubmitted.

12.1. Kerberos Second Factor Types

This section species the IANA "Kerberos Second Factor Types" registry. This registry records the number, name, and reference for each second factor protocol.

12.1.1. Registration Template

ID Number: This is a value that uniquely identifies this entry. It is a signed integer in range -2147483648 to 2147483647, inclusive. Positive values must be assigned only for algorithms specified in accordance with these rules for use with Kerberos and related protocols. Negative values should be used for private and experimental algorithms only. Zero is reserved and must not be assigned.

Name: Brief, unique, human-readable name for this algorithm.

Reference: URI or otherwise unique identifier for where the details of this algorithm can be found. It should be as specific as reasonably possible.

12.1.2. Initial Registry Contents

- ID Number: 1
- Name: NONE
- Reference: this draft.

12.2. Kerberos SPAKE Groups

This section specifies the IANA "Kerberos SPAKE Groups" registry. This registry records the number, name, specification, serialization, multiplier length, multiplier conversion, SPAKE M constant and SPAKE N constant.

12.2.1. Registration Template

ID Number: This is a value that uniquely identifies this entry. It is a signed integer in range -2147483648 to 2147483647, inclusive. Positive values must be assigned only for algorithms specified in accordance with these rules for use with Kerberos and related protocols. Negative values should be used for private and experimental use only. Zero is reserved and must not be assigned. Values should be assigned in increasing order.

Name: Brief, unique, human readable name for this entry.

Specification: Reference to the definition of the group parameters and operations.

Serialization: Reference to the definition of the method used to serialize group elements.

Multiplier Length: The length of the input octet string to multiplication operations.

Multiplier Conversion: Reference to the definition of the method used to convert an octet string to a multiplier scalar.

SPAKE M Constant: The serialized value of the SPAKE M constant in hexadecimal notation.

SPAKE N Constant: The serialized value of the SPAKE N constant in hexadecimal notation.

12.2.2. Initial Registry Contents

- ID Number: 1
- Name: edwards25519
- Specification: [RFC7748] section 4.1 (edwards25519)
- Serialization: [RFC8032] section 3.1
- Multiplier Length: 32
- Multiplier Conversion: [RFC8032] section 3.1
- SPAKE M Constant: 5ada7e4bf6ddd9adb6626d32131c6b5c51a1e347a3478f53cfcf441b88eed12e
- SPAKE N Constant: 10e3df0ae37d8e7a99b5fe74b44672103dbddcbd06af680d71329a11693bc778

- ID Number: 2
- Name: P-256
- Specification: [SEC2] section 2.4.2
- Serialization: [SEC1] section 2.3.3 (compressed).
- Multiplier Length: 32
- Multiplier Conversion: [SEC1] section 2.3.8.
- SPAKE M Constant:
  02886e2f97ace46e55ba9dd7242579f2993b64e16ef3dcab95afd497333d8fa12f
- SPAKE N Constant:
  03d8bbd6c639c62937b04d997f38c3770719c629d7014d49a24b4f98baa1292b49

- ID Number: 3
- Name: P-384
- Specification: [SEC2] section 2.5.1
- Serialization: [SEC1] section 2.3.3 (compressed).
- Multiplier Length: 48
- Multiplier Conversion: [SEC1] section 2.3.8.
- SPAKE M Constant:
  030ff0895ae5ebf6187080a82d82b42e2765e3b2f8749c7e05eba3664
  34b363d3dc36f15314739074d2eb8613fceec2853
- SPAKE N Constant:
  02c72cf2e390853a1c1c4ad816a62fd15824f56078918f43f922ca215
  18f9c543bb252c5490214cf9aa3f0baab4b665c10

- ID Number: 4
- Name: P-521
- Specification: [SEC2] section 2.6.1
- Serialization: [SEC1] section 2.3.3 (compressed).
- Multiplier Length: 66
- Multiplier Conversion: [SEC1] section 2.3.8.
- SPAKE M Constant:
  02003f06f38131b2ba2600791e82488e8d20ab889af753a41806c5db1
  8d37d85608cfae06b82e4a72cd744c719193562a653ea1f119eef9356907edc9b5
  6979962d7aa
- SPAKE N Constant:
  0200c7924b9ec017f3094562894336a53c50167ba8c5963876880542b
  c669e494b2532d76c5b53dfb349fdf69154b9e0048c58a42e8ed04cef052a3bc34
  9d95575cd25

13. References

13.1. Normative References

[CCITT.X680.2002]
International Telephone and Telegraph Consultative Committee, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", CCITT Recommendation X.680, July 2002.

[CCITT.X690.2002]
International Telephone and Telegraph Consultative Committee, "ASN.1 encoding rules: Specification of basic encoding Rules (BER), Canonical encoding rules (CER) and Distinguished encoding rules (DER)", CCITT Recommendation X.690, July 2002.

[I-D.irtf-cfrg-spake2]
Ladd, W., "SPAKE2, a PAKE", draft-irtf-cfrg-spake2-01 (work in progress), February 2015.

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

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, DOI 10.17487/RFC3961, February 2005, <https://www.rfc-editor.org/info/rfc3961>.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, DOI 10.17487/RFC4120, July 2005, <https://www.rfc-editor.org/info/rfc4120>.

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

[RFC6113] Hartman, S. and L. Zhu, "A Generalized Framework for Kerberos Pre-Authentication", RFC 6113, DOI 10.17487/RFC6113, April 2011, <https://www.rfc-editor.org/info/rfc6113>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[SEC1] Standards for Efficient Cryptography Group, "SEC 1: Elliptic Curve Cryptography", May 2009.

[SEC2] Standards for Efficient Cryptography Group, "SEC 2: Recommended Elliptic Curve Domain Parameters", January 2010.

13.2. Non-normative References

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC6560] Richards, G., "One-Time Password (OTP) Pre-Authentication", RFC 6560, DOI 10.17487/RFC6560, April 2012, <https://www.rfc-editor.org/info/rfc6560>.

[RFC6649] Hornquist Astrand, L. and T. Yu, "Deprecate DES, RC4-HMAC-EXP, and Other Weak Cryptographic Algorithms in Kerberos", BCP 179, RFC 6649, DOI 10.17487/RFC6649, July 2012, <https://www.rfc-editor.org/info/rfc6649>.

[SPAKE] Abdalla, M. and D. Pointcheval, "Simple Password-Based Encrypted Key Exchange Protocols", February 2005.

Appendix A. ASN.1 Module

```
   KerberosV5SPAKE {
           iso(1) identified-organization(3) dod(6) internet(1)
           security(5) kerberosV5(2) modules(4) spake(8)
   } DEFINITIONS EXPLICIT TAGS ::= BEGIN

   IMPORTS
       EncryptedData, Int32
         FROM KerberosV5Spec2 { iso(1) identified-organization(3)
           dod(6) internet(1) security(5) kerberosV5(2) modules(4)
           krb5spec2(2) };
           -- as defined in RFC 4120.

   SPAKESupport ::= SEQUENCE {
       groups      [0] SEQUENCE (SIZE(1..MAX)) OF Int32,
       ...
   }

   SPAKEChallenge ::= SEQUENCE {
       group       [0] Int32,
       pubkey      [1] OCTET STRING,
       factors     [2] SEQUENCE (SIZE(1..MAX)) OF SPAKESecondFactor,
       ...
   }

   SPAKESecondFactor ::= SEQUENCE {
       type        [0] Int32,
       data        [1] OCTET STRING OPTIONAL
   }

   SPAKEResponse ::= SEQUENCE {
       pubkey      [0] OCTET STRING,
       factor      [1] EncryptedData, -- SPAKESecondFactor
       ...
   }

   PA-SPAKE ::= CHOICE {
       support     [0] SPAKESupport,
       challenge   [1] SPAKEChallenge,
       response    [2] SPAKEResponse,
       encdata     [3] EncryptedData,
       ...
   }

   END
```

Appendix B. SPAKE M and N Value Selection

The M and N constants for the edwards25519 group are the SHA-256 hashes [RFC6234] hashes of "edwards25519 point generation seed (M)" and "edwards25519 point generation seed (N)" respectively. Both hashes decode to valid curve points.

Appendix C. Test Vectors

For the following text vectors:

- The key is the string-to-key of "password" with the salt "ATHENA.MIT.EDUraeburn" for the designated initial reply key encryption type.

- x and y were chosen randomly within the order of the designated group.

- The SPAKESupport message contains only the designated group's number.

- The SPAKEChallenge message offers only the SF-NONE second factor type.

- The KDC-REQ-BODY message contains no KDC options, the client principal name "raeburn@ATHENA.MIT.EDU", the server principal name "krbtgt/ATHENA.MIT.EDU", the realm "ATHENA.MIT.EDU", the till field "19700101000000Z", the nonce zero, and an etype list containing only the designated encryption type.

```
DES3 edwards25519
key: 850bb51358548cd05e86768c313e3bfef7511937dcf72c3e
w: a1f1a25cbd8e3092667e2fddba8ecd24f2c9cef124f7a3371ae81e11cad42a07
x: 0442027aaf1fa95b7f86589578df43e413167ae8d9dceb377628338123ee7404
y: 6a21e5e81217568835d497fb212b86b49e656acf0641169a0b59f4e665e8b304
X: 13d13759fb4b46273b6b74a2d399c6cce686058b8d44ac51837caf8d860c5c01
Y: 3e0b035ef980fdf99f39c754603fea028813e41315f937d8cdfbbd2d1840f35c
T: ee72f8b68efe6d77bd5f1dcd3d65a08f7e257684fa05a470b6be22dad58e2fe3
S: e6eac1972008a77f23d130d681957159781527ef33d708fab0457bd254073526
K: db0ac82ea8fc1c57ee4291a06a2d0a63c02584753edaf87c473027c44bdc509a
SPAKESupport: a0093007a0053003020101
Checksum after SPAKESupport: 9037756a58a060f80c13
                             354b1a743a66837f1d4d
SPAKEChallenge: a1363034a003020101a1220420ee72f8b68efe6d77bd5f1d
                cd3d65a08f7e257684fa05a470b6be22dad58e2fe3a20930
                073005a003020101
Checksum after SPAKEChallenge: 07e110094bee01edfd17
                               26564bf03f4055aafa41
```

```
   Checksum after pubkey: 7852ee1878c7bd54c0c5b328c0518ceda8ad0340
   KDC-REQ-BODY: 3075a00703050000000000a1143012a003020101a10b3009
                 1b077261656275726ea2101b0e415448454e412e4d49542e
                 454455a3233021a003020102a11a30181b066b7262746774
                 1b0e415448454e412e4d49542e454455a511180f31393730
                 303130313030303030305aa703020100a8053003020110
   K'[0]: 5410d075499b3b64f17a837ff8861975d97cdf26ae8c13a8
   K'[1]: 64cd34ad02fefed0f467d55e2ff4946b0431cb40460bda29
   K'[2]: 6125fd160b67f7979808c7ab135246702c0d67d55be9923e
   K'[3]: 19d9e91adfd940b392ab40d67fe5a85e97f8d5a1a8348f01

   RC4 edwards25519
   key: 8846f7eaee8fb117ad06bdd830b7586c
   w: 2713c1583c53861520b849bfef0525cd4fe82215b3ea6fcd896561d48048f40c
   x: d9d4654fec8d4819fb40d6bab2c8e0121c441ae614a7b8d92a9219afc3f9900a
   y: c39faf42028ef10fba4d16cee68320eba68e778c499d7eeaa83c98034715fc02
   X: 74cce10c08b2f6e2571a7e80a0a035b47f0e0b1b80a44155a4174dff4f8f6774
   Y: adab0adcf18e7ea95a9014374bd1dee64fd7782c5b67c995cd44b20ad1f712e8
   T: 46dc9383672d47c63796912d1092d1dc14cef1c0aaf306caa59afaafac18fa48
   S: e69493aba8f3661fca0781aff5ffecd8f35f477e903d35a6fcd4a277299a3816
   K: c489b1a851ed7c3cd6c2a4289591d6ec973263013a8d28261ce4f845110a0acd
   SPAKESupport: a0093007a0053003020101
   Checksum after SPAKESupport: c8bb7fb72f6b142557fd5de9b1b8bb4c
   SPAKEChallenge: a1363034a003020101a122042046dc9383672d47c6379691
                   2d1092d1dc14cef1c0aaf306caa59afaafac18fa48a20930
                   073005a003020101
   Checksum after SPAKEChallenge: 69d146f2ac5e644d6eb721cd28519dc0
   Checksum after pubkey: 6f0b5d591e98ee3824e7e2b90385de3d
   KDC-REQ-BODY: 3075a00703050000000000a1143012a003020101a10b3009
                 1b077261656275726ea2101b0e415448454e412e4d49542e
                 454455a3233021a003020102a11a30181b066b7262746774
                 1b0e415448454e412e4d49542e454455a511180f31393730
                 303130313030303030305aa703020100a8053003020117
   K'[0]: 741ace018b652387f2c45cc72e3c6eaf
   K'[1]: 552ab0343b2ca5dba61ca0b37793a015
   K'[2]: 5717b7326e1a8dcb3f11556d9246d976
   K'[3]: 139dd7e1bfe7cc30a91e86f4e00f256f

   AES128 edwards25519
   key: fca822951813fb252154c883f5ee1cf4
   w: 17c2a9030afb7c37839bd4ae7fdfeb179e99e710e464e62f1fb7c9b67936f30b
   x: ca974053ebea21b4e833d7deccbc1f10ccc5e9304fc1c1ea4f624891c5321807
   y: f7ee153d6c05a8cdd2b7b0f733837a247d2b9dcdc163018b4422ae72b83deb02
   X: 30786a5754047d939f98e13f6fb3f352cf89c3cda326e4269c502bcdb29137e0
   Y: f5e82d4295cee37147d9b049d29738b846ab3474fc63c02ee57c7140eed5979a
   T: 1aeb6ef3ea5028d0f9fb3780599234aa3fddea33ca6f554a8b1cb9f338f6eca8
   S: a5e731c0d9a73a49deb8ac2125d89927ea378012b6e8aa114e68d314860e68fe
   K: ee96a6407b4379b0ca7242219d98fb27d3430eb6e997f333dd27f05da5a7a804
```

```
SPAKESupport: a0093007a0053003020101
Checksum after SPAKESupport: ce5052873534f00424e38897
SPAKEChallenge: a1363034a003020101a12204201aeb6ef3ea5028d0f9fb37
                80599234aa3fddea33ca6f554a8b1cb9f338f6eca8a20930
                073005a003020101
Checksum after SPAKEChallenge: a0cfeee54b5dd9f4707c2167
Checksum after pubkey: cdc77b01d9b0c4b0dfc56227
KDC-REQ-BODY: 3075a0070305000000000000a1143012a003020101a10b3009
              1b077261656275726ea2101b0e415448454e412e4d49542e
              454455a3233021a003020102a11a30181b066b7262746774
              1b0e415448454e412e4d49542e454455a511180f31393730
              303130313030303030305aa703020100a8053003020111
K'[0]: 8cac5d0d7cbd1482e5adc141059957ea
K'[1]: 24f8217603d6ea9519f3d7013d29a547
K'[2]: f9b1ef3f0ad56f04a3fcd4d10a9fa54d
K'[3]: c69d0d5204496487c7beb7600146b384

AES256 edwards25519
key: 01b897121d933ab44b47eb5494db15e50eb74530dbdae9b634d65020ff5d88c1
w: 35b35ca126156b5bf4ec8b90e9545060f2108f1b6aa97b381012b9400c9e3f0e
x: d11898149ec443fe2219ef5160b805e0d66508828e117bff8b32ceeec03ca80f
y: 1137eb1beb9d2b4db7d91f8d03eb844a7d35e1b44998f31f6a16258c67e4a50a
X: d95d62d3bfd3d7b7b18e7dad3a86abe92c59927327bf4da0787bb04f4d1076b2
Y: 7482c6fdcd2a1717d7a4cdec56cf6e6d1f4663d69477d293c014fd148e2eb5cb
T: 5c09d3507551adaa8bd6099bda92090a7ba0f92a24100101948332331b225ceb
S: 3f7a09276fa650e5c9d855eb412b87784b92f1c8c235503ea107b5c1d6434354
K: 202e57295aef245c3aa535e10de31edf8728887081f22a2752193bd082adb10d
SPAKESupport: a0093007a0053003020101
Checksum after SPAKESupport: 14b16e16da078fab9830a66c
SPAKEChallenge: a1363034a003020101a12204205c09d3507551adaa8bd609
                9bda92090a7ba0f92a24100101948332331b225ceba20930
                073005a003020101
Checksum after SPAKEChallenge: 6318b92b6540bae5d18a5454
Checksum after pubkey: 926d71c006fe53fa8bb50105
KDC-REQ-BODY: 3075a0070305000000000000a1143012a003020101a10b3009
              1b077261656275726ea2101b0e415448454e412e4d49542e
              454455a3233021a003020102a11a30181b066b7262746774
              1b0e415448454e412e4d49542e454455a511180f31393730
              303130313030303030305aa703020100a8053003020112
K'[0]: 0f69df747af8f060d3013f545ceefafe
       9aca4784fc0983f1e343698457e499f9
K'[1]: f8f39aca22a32f1500c21a8ad47af00f
       02db1ec84c4cf72b4d7c9c65999b5175
K'[2]: 623edb38085c0c35c9832ec3ef0cc976
       502e47dd5d4438013c9bfaf67d2febb5
K'[3]: d697c2d57c868a4494eee568f29aa003
       d7223994caf1b18c7cc936cf45e7d453
```

```
AES256 P-256
key: 01b897121d933ab44b47eb5494db15e50eb74530dbdae9b634d65020ff5d88c1
w: eb2984af18703f94dd5288b8596cd36988d0d4e83bfb2b44de14d0e95e2090bd
x: 935ddd725129fb7c6288e1a5cc45782198a6416d1775336d71eacd0549a3e80e
y: e07405eb215663abc1f254b8adc0da7a16febaa011af923d79fdef7c42930b33
X: 03bc802165aea7dbd98cc155056249fe0a37a9c203a7c0f7e872d5bf687bd105e2
Y: 0340b8d91ce3852d0a12ae1f3e82c791fc86df6b346006431e968a1b869af7c735
T: 024f62078ceb53840d02612195494d0d0d88de21feeb81187c71cbf3d01e71788d
S: 021d07dc31266fc7cfd904ce2632111a169b7ec730e5f74a7e79700f86638e13c8
K: 0268489d7a9983f2fde69c6e6a1307e9d252259264f5f2dfc32f58cca19671e79b
SPAKESupport: a0093007a0053003020102
Checksum after SPAKESupport: 61f93e7f998dec5f54cac55c
SPAKEChallenge: a1373035a003020102a1230421024f62078ceb53840d0261
                2195494d0d0d88de21feeb81187c71cbf3d01e71788da209
                30073005a003020101
Checksum after SPAKEChallenge: 949916036d3c524608533206
Checksum after pubkey: 1024bfe60a1e22b5bf2838c3
KDC-REQ-BODY: 3075a0070305000000000000a1143012a003020101a10b3009
              1b077261656275726ea2101b0e415448454e412e4d49542e
              454455a3233021a003020102a11a30181b066b7262746774
              1b0e415448454e412e4d49542e454455a511180f31393730
              303130313030303030305aa703020100a8053003020112
K'[0]: b3a882eccd2f31df46880f6235522a4d
       87523a34442547778c46780f5b35800a
K'[1]: 6e18ebfd20a9a05af11b320eaab15870
       93f3e21a5efcb261307786661330344d
K'[2]: 11e1a36e87c729a89bbda12cfa15652f
       a1848c0ba9b72cb3e69562648744fb09
K'[3]: 9875d491c6d0bb7cbe6d374c368e1242
       97e506becbf8ec6aa539a0d70b9e430a

AES256 P-384
key: 01b897121d933ab44b47eb5494db15e50eb74530dbdae9b634d65020ff5d88c1
w: 0304cfc55151c6bbe889653db96dbfe0ba4acafc024c1e8840cb3a486f6d80c1
   6e1b8974016aa4b7fa43042a9b3825b1
x: f323ca74d344749096fd35d0adf20806e521460637176e84d977e9933c49d76f
   cfc6e62585940927468ff53d864a7a50
y: 5b7c709acb175a5afb82860deabca8d0b341facdff0ac0f1a425799aa905d750
   7e1ea9c573581a81467437419466e472
X: 0211e3334f117b76635dd802d4022f601680a1fd066a56606b7f246493a10351
   7797b81789b225bd5bb1d9ae1da2962250
Y: 0383dfa413496e5e7599fc8c6430f8d6910d37cf326d81421bc92c0939b555c4
   ca2ef6a993f6d3db8cb7407655ef60866e
T: 02a1524603ef14f184696f854229d3397507a66c63f841ba748451056be07879
   ac298912387b1c5cdff6381c264701be57
S: 020d5adfdb92bc377041cf5837412574c5d13e0f4739208a4f0c859a0a302bc6
   a533440a245b9d97a0d34af5016a20053d
K: 0264aa8c61da9600dfb0beb5e46550d63740e4ef29e73f1a30d543eb43c25499
```

```
      037ad16538586552761b093cf0e37c703a
   SPAKESupport: a0093007a0053003020103
   Checksum after SPAKESupport: a0024c7b5ff667ae074a9988
   SPAKEChallenge: a1473045a003020103a133043102a1524603ef14f184696f
                   854229d3397507a66c63f841ba748451056be07879ac2989
                   12387b1c5cdff6381c264701be57a20930073005a003020101
   Checksum after SPAKEChallenge: ecd0f64ed7c0d4e18fa4c5b4
   Checksum after pubkey: a238108c88afd856f04d3aa5
   KDC-REQ-BODY: 3075a0070305000000000a1143012a003020101a10b3009
                 1b077261656275726ea2101b0e415448454e412e4d49542e
                 454455a3233021a003020102a11a30181b066b7262746774
                 1b0e415448454e412e4d49542e454455a511180f31393730
                 3031303130303030305aa703020100a8053003020112
   K'[0]: ff59fb5fb83c7bafe197b62c853eb7c3
          a2902301dfe8326851626a0e9c714c47
   K'[1]: e3c741ac7041feed0f0b5c36cb74c179
          cb565e509b6d65594d0badafe318c4dc
   K'[2]: 9c7a73087f22b52db38a14eb8292df61
          54516eaadb7149b14d35864bdb85aa22
   K'[3]: 75ea14f0f53ee8dbabd78f446462cfda
          590d4ace0fa93708a00f26f26c565e56

   AES256 P-521
   key: 01b897121d933ab44b47eb5494db15e50eb74530dbdae9b634d65020ff5d88c1
   w: 003a095a2b2386eff3eb15b735398da1caf95bc8425665d82370aff58b0471f3
      4cce63791cfed967f0c94c16054b3e1703133681bece1e05219f5426bc944b0f
      bfb3
   x: 017c38701a14b490b6081dfc83524562be7fbb42e0b20426465e3e37952d30bc
      ab0ed857010255d44936a1515607964a870c7c879b741d878f9f9cdf5a865306
      f3f5
   y: 003e2e2950656fa231e959acdd984d125e7fa59cec98126cbc8f3888447911eb
      cd49428a1c22d5fdb76a19fbeb1d9edfa3da6cf55b158b53031d05d51433ade9
      b2b4
   X: 03003e95272223b210b48cfd908b956a36add04a7ff443511432f94ddd87e064
      1d680ba3b3d532c21fa6046192f6bfae7af81c4b803aa154e12459d1428f8f2f
      56e9f2
   Y: 030064916687960df496557ecab08298bf075429eca268c6dabbae24e258d568
      c62841664dc8ecf545369f573ea84548faa22f118128c0a87e1d47315afabb77
      3bb082
   T: 02017d3de19a3ec53d0174905665ef37947d142535102cd9809c0dfbd0dfe007
      353d54cf406ce2a59950f2bb540df6fbe75f8bbbef811c9ba06cc275adbd9675
      6696ec
   S: 02004d142d87477841f6ba053c8f651f3395ad264b7405ca5911fb9a55abd454
      fef658a5f9ed97d1efac68764e9092fa15b9e0050880d78e95fd03abf5931791
      6822b5
   K: 03007c303f62f09282cc849490805bd4457a6793a832cbeb55df427db6a31e99
      b055d5dc99756d24d47b70ad8b6015b0fb8742a718462ed423b90fa3fe631ac1
      3fa916
```

```
   SPAKESupport: a0093007a0053003020104
   Checksum after SPAKESupport: 1b69d116036e141e45d4f7d7
   SPAKEChallenge: a1593057a003020104a145044302017d3de19a3ec53d0174
                   905665ef37947d142535102cd9809c0dfbd0dfe007353d54
                   cf406ce2a59950f2bb540df6fbe75f8bbbef811c9ba06cc2
                   75adbd96756696eca20930073005a003020101
   Checksum after SPAKEChallenge: cac3da1e9ab1261723ece823
   Checksum after pubkey: 654493ca7e47f3c5200f4b84
   KDC-REQ-BODY: 3075a00703050000000000a1143012a003020101a10b3009
                 1b0772616562757726ea2101b0e415448454e412e4d49542e
                 454455a3233021a003020102a11a30181b066b7262746774
                 1b0e415448454e412e4d49542e454455a511180f31393730
                 303130313030303030305aa703020100a8053003020112
   K'[0]: c91635dfd1de3884b635b58b30d3cfd5
          26fe78f8dade6f19e4eb2fb23ef594ca
   K'[1]: 03d38e139bb3f66cc76c5da720f3bf11
          4280f64ed708e69e96094bb62aa28f32
   K'[2]: 515eaa3c45b08bc9d77468059e64a8e1
          96cfcd15db92ad431cae5edbe721d07e
   K'[3]: 898ae786e58391d8a00eb7a7cbddd005
          3aff9147b42a3076d934608e70a6f0ff

   AES256 edwards25519 with accepted optimistic challenge
   key: 01b897121d933ab44b47eb5494db15e50eb74530dbdae9b634d65020ff5d88c1
   w: 35b35ca126156b5bf4ec8b90e9545060f2108f1b6aa97b381012b9400c9e3f0e
   x: 6e52ee8086e995770ab9140a6e3cb39870f9d51900d706b381fafcfca955850c
   y: af0b2d4034a9011e0552c7986813e2eb057e3b71ab2465aa59f8c02c492a6c00
   X: aa5229fe8f321388ed3b491f2fa72dc71a1ae0fec2cd3a965076230612d690bb
   Y: e02e82965f8159f12b581bb6ccbb7e13a08d2950f1ad88734581d1634ba00029
   T: 34a986c6301cd22dfc7e693ca1701ce0272e735c549056dbd6796282f55df4bb
   S: deae3975cc2319c9aa1747527fd8a82a7f09d3de363695beb537050fb71867f9
   K: 96f99e03cf272219ef406c8b4dd1f067f04c08a3c5bc59f17047337cb9e3aba7
   SPAKEChallenge: a1363034a003020101a122042034a986c6301cd22dfc7e69
                   3ca1701ce0272e735c549056dbd6796282f55df4bba20930
                   073005a003020101
   Checksum after SPAKEChallenge: c757f735cbf0a4b2b9919eb4
   Checksum after pubkey: 12bb1d11bd225e3b93212802
   KDC-REQ-BODY: 3075a00703050000000000a1143012a003020101a10b3009
                 1b0772616562757726ea2101b0e415448454e412e4d49542e
                 454455a3233021a003020102a11a30181b066b7262746774
                 1b0e415448454e412e4d49542e454455a511180f31393730
                 303130313030303030305aa703020100a8053003020112
   K'[0]: 28add62799076c5cd996db4660b0f8b5
          685ec8f669fb3d19cbb2dc9efa2761e3
   K'[1]: 7df1db9587d473bf7f82259afc5758e2
          ed3968c1f4cefe70c4d446086c45f588
   K'[2]: bb8817c50846ed064701cc809fceedfa
          8eb9e95cf4868eab8e42e6155928f35e
```

```
   K'[3]: 5e521d6dc898d3737e5c1d8039a8cb56
          306270ab6fd58c59ec3db3a392605a82

   AES256 P-521 with rejected optimistic edwards25519 challenge
   key: 01b897121d933ab44b47eb5494db15e50eb74530dbdae9b634d65020ff5d88c1
   w: 003a095a2b2386eff3eb15b735398da1caf95bc8425665d82370aff58b0471f3
      4cce63791cfed967f0c94c16054b3e1703133681bece1e05219f5426bc944b0f
      bfb3
   x: 01687b59051bf40048d7c31d5a973d792fa12284b7a447e7f5938b5885ca0bb2
      c3f0bd30291a55fea08e143e2e04bdd7d19b753c7c99032f06cab0d9c2aa8f83
      7ef7
   y: 01ded675ebf74fe30c9a53710f577e9cf84f09f6048fe245a4600004884cc167
      733f9a9e43108fb83babe8754cd37cbd7025e28bc9ff870f084c7244f536285e
      25b4
   X: 03001bed88af987101ef52db5b8876f6287eb49a72163876c2cf99deb94f4c74
      9bfd118f0f400833cc8daad81971fe40498e6075d8ba0a2acfac35eb9ec8530e
      e0edd5
   Y: 02007bd3bf214200795ea449852976f241c9f50f445f78ff2714fffe42983f25
      cd9c9094ba3f9d7adadd6c251e9dc0991fc8210547e7769336a0ac406878fb94
      be2f1f
   T: 02014cb2e5b592ece5990f0ef30d308c061de1598bc4272b4a6599bed466fd15
      21693642abcf4dbe36ce1a2d13967de45f6c4f8d0fa8e14428bf03fb96ef5f1e
      d3e645
   S: 02016c64995e804416f748fd5fa3aa678cbc7cbb596a4f523132dc8af7ce84e5
      41f484a2c74808c6b21dcf7775baefa6753398425becc7b838b210ac5daa0cb0
      b710e2
   K: 0200997f4848ae2e7a98c23d14ac662030743ab37fccc2a45f1c721114f40bcc
      80fe6ec6aba49868f8aea1aa994d50e81b86d3e4d3c1130c8695b68907c673d9
      e5886a
   Optimistic SPAKEChallenge: a1363034a003020102a1220420ffc334
                              3df010544de2e3aa1d8573ebb5a5a960
                              d5e6a44c151610e12874155be3a20930
                              073005a003020101
   Checksum after optimist SPAKEChallenge: 8d319e6c9799f1d52545a571
   SPAKESupport: a0093007a0053003020104
   Checksum after SPAKESupport: 4ae061f319946f2a8bed3f77
   SPAKEChallenge: a1593057a003020104a145044302014cb2e5b592ece5990f
                   0ef30d308c061de1598bc4272b4a6599bed466fd15216936
                   42abcf4dbe36ce1a2d13967de45f6c4f8d0fa8e14428bf03
                   fb96ef5f1ed3e645a20930073005a003020101
   Checksum after SPAKEChallenge: 169d594dfe425e7b6bd494aa
   Checksum after pubkey: 12191866bc5d6b92676aa83b
   KDC-REQ-BODY: 3075a00703050000000000a1143012a003020101a10b3009
                 1b077261656275726ea2101b0e415448454e412e4d49542e
                 454455a3233021a003020102a11a30181b066b7262746774
                 1b0e415448454e412e4d49542e454455a511180f31393730
                 3031303130303030305aa703020100a8053003020112
   K'[0]: 0ed9e6e9fcc30d9fcd5163c520ce0b58
```

```
         47772b94ba684d786b46a0fdd2b79364
  K'[1]: 981037863dd4049823a59213b68a0b60
         78e8140a3a921273aa6c777c858504ed
  K'[2]: d1bb237ea8df724265d21c8dc2080db0
         3d1949a07fab64f4a149aa936b1e4d2e
  K'[3]: 7ed29c8842536cc61be9814747e99ba2
         b20686d26e2bf9f6dbedae011f8d069b
```

Appendix D. Acknowledgements

Nico Williams (Cryptonector)
Taylor Yu (MIT)

Authors' Addresses

Nathaniel McCallum
Red Hat, Inc.

EMail: npmccallum@redhat.com

Simo Sorce
Red Hat, Inc.

EMail: ssorce@redhat.com

Robbie Harwood
Red Hat, Inc.

EMail: rharwood@redhat.com

Greg Hudson
MIT

EMail: ghudson@mit.edu