Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 12, 2020

V. Smyslov
ELVIS-PLUS

September 9, 2019

Usage of PAKE Protocols with IKEv2
draft-smyslov-ikev2-pake-00

Abstract

This memo discusses how PAKE (Password Authenticated Key Exchange) protocols can be integrated into the IKEv2 (Internet Key Exchange) protocol.

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 https://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 March 12, 2020.

Copyright Notice

Table of Contents

1. Introduction

Recent interest in PAKE protocols in IETF resulted in launching the PAKE selection process in CFRG. The goal of the process is to select one (or more, or zero) PAKE protocol(s) that will be recommended to be used in IETF security protocols, such as TLS 1.3, IKEv2 etc. There are eight candidates nominated: four balanced PAKEs (SPAKE2 [I-D.irtf-cfrg-spake2], J-PAKE [RFC8236], SPERE [SPEKE] and CPace [CPace-AuCPace]) and four augmented ones (OPAQUE [I-D.krawczyk-cfrg-opaque], AuCPace [CPace-AuCPace], VTBPEKE [VTBPEKE] and BSPAKE [BSPAKE]). The part of the selection process is an evaluation of how well the candidates can be fit into existing IETF security protocols. This memo aims to discuss how each of the candidates can be integrated into IKEv2.

The IKEv2 protocol defined in [RFC7296] is a key part of IPsec (IP Security) architecture, as it provides an authenticated key exchange between peers who wish to establish an IPsec SA (Security Association). Core IKEv2 specification allows peers to authenticate each other either by using PSK (Pre-Shared Key) or by means of digital signatures. In addition, the core IKEv2 specification allows optional use of EAP (Extensible Authentication Protocol) in the protocol. Note, that the way EAP was originally integrated into IKEv2 still required using digital signatures by responder. However, later an extension for the IKEv2 was developed, which allowed using

EAP-only authentication [RFC5998].

2. Terminology and Notation

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Integrating PAKE protocols into IKEv2

Authentication by means of PAKE protocols in IKEv2 is of high interest in IPSECME WG. Both balanced and augmented PAKEs can be employed - the former are suitable for peer to peer communication, when each side can initiate, while the latter are good for remote access scenarios.

Note, that for augmented PAKE protocols a password setup step is needed, during which password verifier is generated and transferred to server. This step is out of scope of this memo, we assume that all the setup steps are already somehow done.

Early attempts to integrate PAKE protocols into IKEv2 were made in IPSECME WG around 2009-2011. The easiest way to do it would be to define new EAP methods based on PAKE protocols and to rely on build-in EAP authentication (along with "EAP-only" extension). However, the disadvantage of this approach is its inefficiency and complexity. As a result, the WG developed a new experimental IKEv2 extension, that aimed to provide a framework for integration of any PAKE protocol into IKEv2 [RFC6467]. This extension generalizes any PAKE protocol as a series of message exchanges resulted in the computing of a shared secret. The extension defines how use of a particular PAKE is negotiated and how PAKE messages are transferred in multi-round IKE_AUTH exchange. Several PAKE protocols were then instantiated using this extension: "Dragonfly" [RFC6617], AugPAKE [RFC6628] and PACE [RFC6631].

3.1. Using EAP methods

Integration of PAKE protocols into IKEv2 using EAP methods is the simplest possible way from protocol point of view, because it requires no modification to the IKEv2 (provided EAP is already supported and "EAP-only" extension to IKEv2 is implemented). However, this approach has the following drawbacks.

- Lack of flexibility. Using EAP is not negotiated in IKEv2, it's done on initiator's discretion. It's better to have ability to negotiate using PAKE or convenient authentication.

- Duplicated functionality. If some PAKE EAP methods are available (or a single method with different parameters) it is EAP that would negotiate using a particular one. IKEv2 has its own negotiation mechanism that's better to be used instead. In addition, EAP methods would have use their own variables, like session id, peers identities etc. These variables are available in IKEv2 and it's better to use them instead.

- Inefficiency. EAP methods for PAKE protocols would have included a key confirmation step that would have required an additional round trip and some computations. This step is not needed in case of IKEv2, since IKEv2 will do this step itself anyway.

- Complexity. EAP adds an additional layer of encapsulation, that is not always needed.

For these reasons integration of PAKE protocols into IKEv2 using EAP methods is NOT RECOMMENDED. Note, that currently from the eight PAKE candidates listed in Section 1, only for SPEKE a ready to use EAP method is defined (see [EAP-IANA]. EAP methods for other PAKE candidates are not yet defined.

3.2. Using RFC6467 framework

This framework was specifically designed as a lightweight alternative to EAP for the purpose of PAKE protocols integration into IKEv2. The framework models any PAKE protocol as a series of exchanges of opaque data and defines a new IKEv2 payload called GSPM to carry this data in the IKE_AUTH exchange. The number of these data exchanges depends

on the particular PAKE protocol and potentially is not limited, so this framework makes the IKE_AUTH exchanges multi-round (as well as in case of using EAP). The framework also defines how the use of PAKE protocol is negotiated between peers in the IKE_SA_INIT exchange. The framework however doesn't define how the AUTH payload content is calculated in case of PAKE. SO, each PAKE protocol must define the content of the GSPM payloads in each message and the way the AUTH payload is calculated. Each PAKE protocol must also have a unique identifier, registered by IANA in IKEv2 registry "IKEv2 Secure Password Methods" [IKEv2-IANA], so that IKE is able to negotiate its use.

RFC6467 has an "Experimental" status and, to the best of author's knowledge, is not widely supported. It also complicates IKEv2 state machine by making the IKE_AUTH exchange multi-round (note, that if EAP is implemented in IKEv2, this complication must be already there). Nevertheless it seems to be a primary way for integration various PAKE protocols into IKEv2 and is RECOMMENDED to be used for this purpose. Alternative approaches are discussed in Section 3.3.

3.2.1. Algorithm Agility

IKEv2 is able to negotiate usage of some cryptographic primitives, such as (EC)DH group, PRF, encryption and integrity algorithms. Some PAKE protocols are defined so that they can be used with different cryptographic parameters. For the sake of algorithm agility it is desirable when possible to use IKE-negotiated primitives in PAKE protocols. Note, that IKEv2 doesn't negotiate some cryptographic primitives that are widely used in PAKE protocols, such as hash function, memory-hard function or KDF. In addition, negotiation of MAC (integrity algorithm) can be omitted if AEAD algorithm is used in IKEv2. Nevertheless, some of missing cryptographic primitives can be constructed using those, that are negotiated in IKEv2 (e.g. use prf+ construction defined in [RFC7296] as KDF, use AEAD with no encryption as MAC in case no integrity algorithm is negotiated etc.).

We think the possibility to use as many of IKEv2 negotiated primitives in PAKE protocols as possible should be considered, however this may lead to the need to re-evaluate some of PAKE protocols' security proof, especially if it highly depends on the particular combination of used cryptographic primitives. In case of augmented PAKEs an ability to use IKEv2 negotiation capabilities is

further restricted, because usually the group (and possibly some other primitives) are fixed at the time the password verifier is generated and cannot be changed at the time the protocol runs (unless several password verifiers are generated for different primitives, that is impractical).

Alternatively, it is possible to allocate several identifiers in the "IKEv2 Secure Password Methods" registry for the same PAKE protocol with different parameters. This approach is NOT RECOMMENDED, because it increases the size of the IKE_SA_INIT request message, that may lead to problems with IP fragmentation.

3.2.2. AUTH Payload Calculation

Note, that the way AUTH payload is computed in core IKEv2 specification ensures that all data exchanged between peers is authenticated. So, when PAKE protocol redefines AUTH payload calculation, it MAY take advantage of this fact and mot explicitly include variables that are exchanged between peers, since they are already implicitly included. Note also, that IKE SA always has a unique session identifier (IKE SPIs) and IKE always include identities of the peers in the IKE_AUTH exchange, so there is no need to transfer them separately in PAKE protocol.

Note, that RFC6467 doesn't assume any change in IKEv2 key derivation scheme, so that shared secrets generated by PAKE protocols are

assumed to be used only for the purpose of authentication and are not included (either directly or indirectly) in calculation of SK_* keys in IKEv2.

3.2.3. Possible PAKE Protocols Instantiation

Below are examples of how PAKE selection process candidates can be instantiated with RFC6467 framework. Note, that these examples are for illustrative purposes only, they need to be carefully examined and may be reconsidered when the selection process is over.

3.2.3.1. SPAKE2

SPAKE2 is defined in [I-D.irtf-cfrg-spake2]. Possible instantiation of SPAKE2 in IKEv2 is shown on Figure 1.

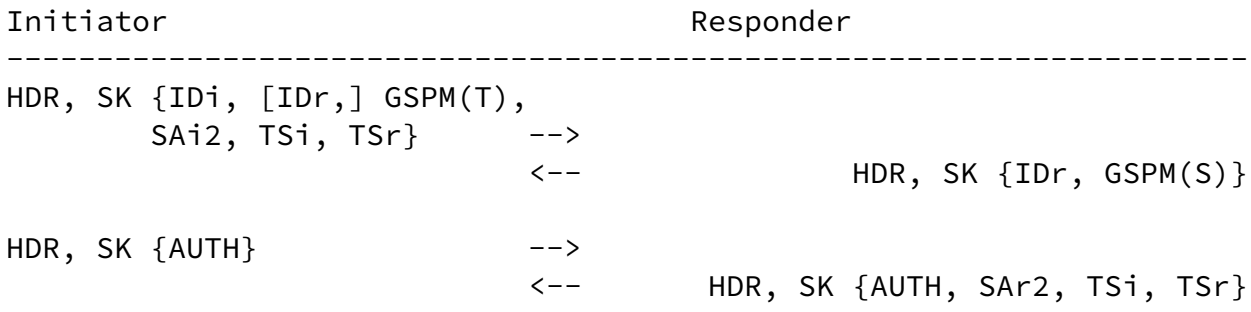

Figure 1: IKE_AUTH Exchange with SPAKE2

[I-D.irtf-cfrg-spake2] defines nine ciphersuites for SPAKE2 by parameterizing the following cryptographic primitives: group, hash, KDF, MAC and Memory-Hard hash Function (MHF). It is believed, that some of these parameters may be directly borrowed from (or constructed from) the IKEv2 negotiated cryptographic primitives. Note also, that some parameters (e.g. KDF, MAC) are only used for key derivation and confirmation, that in case of RFC6467 is done by IKEv2 itself.

According to [I-D.irtf-cfrg-spake2] the result of a SPAKE2 protocol run is a pair of shared secrets KcA and KcB that must be used in AUTH payload computation for key confirmation. One of the possible ways of doing this is shown in Figure 2.

```
AUTHi = prf( prf(KcA, "SPAKE2 for IKEv2"), <InitiatorSignedOctets>)

AUTHr = prf( prf(KcB, "SPAKE2 for IKEv2"), <ResponderSignedOctets>)
```

Figure 2: AUTH Payload calculation in case of SPAKE2

We believe key derivation procedure for KcA and KcB defined in [I-D.irtf-cfrg-spake2] may be optimized for the specific use in IKEv2, however it's out of scope of this memo.

3.2.3.2. J-PAKE

J-PAKE is defined in [RFC8236]. Possible instantiation of J-PAKE in IKEv2 is shown on Figure 3.

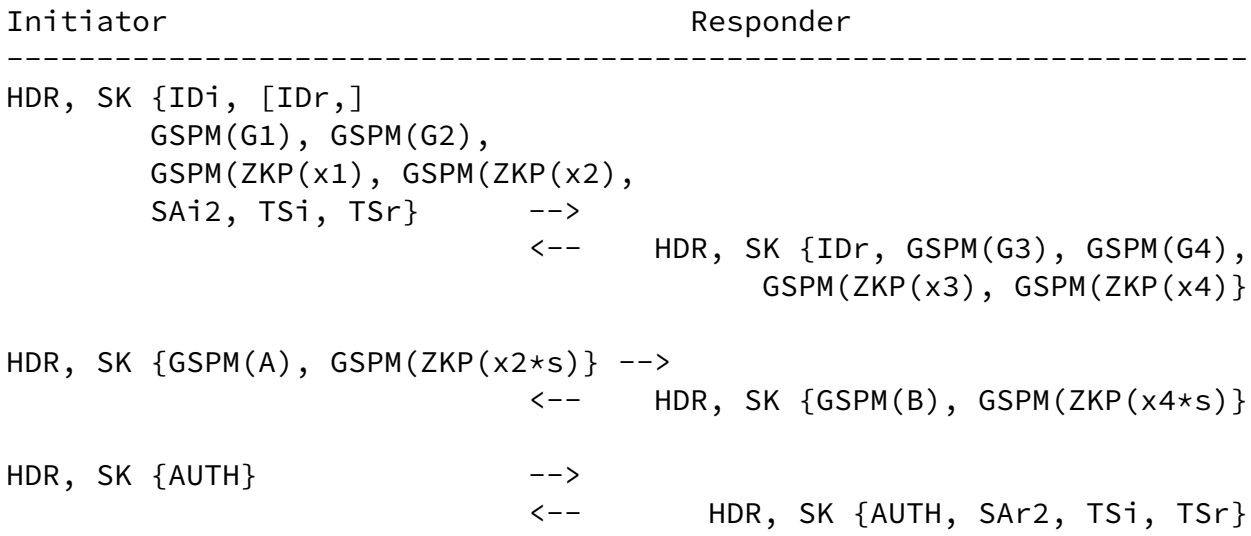

Figure 3: IKE_AUTH Exchange with J-PAKE

Note, that unlike other candidates J-PAKE requires two round trips before shared secret is computed.

[RFC8236] allows using J-PAKE with different groups. It is believed that the IKEv2 negotiated group can be used in case of J-PAKE. On the other hand, J-PAKE uses a specific cryptographic primitive called Zero Knowledge Proof (ZKP), for which Schnorr NIZK proof is used, and it seems that it cannot be constructed from IKEv2 negotiated primitives. Some other cryptographic primitives (H, MAC, KDF) are only used for key derivation and confirmation, that in case of RFC6467 is done by IKEv2 itself.

According to [RFC8236] the result of a J-PAKE protocol run is a shared secret K that must be used in AUTH payload computation for key confirmation. One of the possible ways of doing this is shown in Figure 4.

AUTHi = prf(prf(K, "J-PAKE for IKEv2 Initiator"),

```
                                                  <InitiatorSignedOctets>)

   AUTHr = prf( prf(K, "J-PAKE for IKEv2 Responder"),
                                                  <ResponderSignedOctets>)
```

Figure 4: AUTH Payload calculation in case of J-PAKE

3.2.3.3. SPEKE

SPEKE is defined in [SPEKE]. Possible instantiation of SPEKE in IKEv2 is shown on Figure 5.

```
   Initiator                                Responder
   ------------------------------------------------------------------------
   HDR, SK {IDi, [IDr,] GSPM(g^x),
           SAi2, TSi, TSr}          -->
                                    <--          HDR, SK {IDr, GSPM(g^y)}

   HDR, SK {AUTH}                   -->
                                    <--    HDR, SK {AUTH, SAr2, TSi, TSr}
```

Figure 5: IKE_AUTH Exchange with SPEKE

Although it seems that SPEKE can be used with different groups, [SPEKE] describes using MODP groups only. It is not clear whether IKEv2 negotiated group can generally be used in case of SPEKE. SPEKE also uses hash function H for generating of key confirmation messages, but we believe it is not necessary in case of RFC6467, since key confirmation is done by IKEv2 itself.

According to [SPEKE] the result of a SPEKE protocol run is a shared secret k that must be used in AUTH payload computation for key confirmation. One of the possible ways of doing this is shown in Figure 6.

```
   AUTHi = prf( prf(k, "SPEKE for IKEv2 Initiator"),
                                                  <InitiatorSignedOctets>)

   AUTHr = prf( prf(k, "SPEKE for IKEv2 Responder"),
                                                  <ResponderSignedOctets>)
```

Figure 6: AUTH Payload calculation in case of SPEKE

3.2.3.4. CPace

CPace is defined in [CPace-AuCPace]. Possible instantiation of CPace in IKEv2 is shown on Figure 7.

```
Initiator                                Responder
-------------------------------------------------------------------
HDR, SK {IDi, [IDr,] GSPM(Ya),
        SAi2, TSi, TSr}      -->
                             <--              HDR, SK {IDr, GSPM(Yb)}

HDR, SK {AUTH}               -->
                             <--       HDR, SK {AUTH, SAr2, TSi, TSr}
```

Figure 7: IKE_AUTH Exchange with CPace

CPace uses ssid (session identifier) and CI (connection identifier), for which IKE SPIs can be used.

[CPace-AuCPace] specifies that CPace must only be used with (hyper-)elliptic curves. It is not clear whether IKEv2 negotiated group can generally be used in case of CPace. CPace also uses hash function and memory-hard function primitives which are not negotiated by IKEv2.

According to [CPace-AuCPace] the result of a CPace protocol run is a shared secret sk1 that must be used in AUTH payload computation for key confirmation. One of the possible ways of doing this is shown in Figure 8.

```
AUTHi = prf( prf(sk1, "CPace for IKEv2 Initiator"),
                                              <InitiatorSignedOctets>)

AUTHr = prf( prf(sk1, "CPace for IKEv2 Responder"),
                                              <ResponderSignedOctets>)
```

Figure 8: AUTH Payload calculation in case of CPace

3.2.3.5. OPAQUE

OPAQUE is defined in [I-D.krawczyk-cfrg-opaque]. Possible instantiation of OPAQUE in IKEv2 is shown on Figure 9.

```
   Initiator                                  Responder
   -----------------------------------------------------------------------
   HDR, SK {IDi, [IDr,] GSPM(alpha),
           SAi2, TSi, TSr}          -->
                                    <--            HDR, SK {IDr, GSPM(beta),
                                                       GSPM(EnvU), GSPM(vU)}

   HDR, SK {AUTH}                   -->
                                    <--       HDR, SK {AUTH, SAr2, TSi, TSr}
```

Figure 9: IKE_AUTH Exchange with OPAQUE

Unlike other PAKE candidates, the result of an OPAQUE protocol run is a pair of private keys PrivU and PrivS (each known only to the corresponding party), that must be used in AUTH payload computation for key confirmation. One of the possible ways of doing this is shown in Figure 10.

```
               AUTHi = sig(PrivU, <InitiatorSignedOctets>)

               AUTHr = sig(PrivS, <ResponderSignedOctets>)
```

Figure 10: AUTH Payload calculation in case of OPAQUE

3.2.3.6. AuCPace

AuCPace is defined in [CPace-AuCPace]. Possible instantiation of AuCPace in IKEv2 is shown on Figure 11.

```
   Initiator                                  Responder
   -----------------------------------------------------------------------
   HDR, SK {IDi, [IDr,] GSPM(),
           SAi2, TSi, TSr}          -->
                                    <--               HDR, SK {IDr, GSPM(X),
                                                        GSPM(salt), GSPM(Ya)}

   HDR, SK {GSPM(Yb), AUTH}         -->
                                    <--       HDR, SK {AUTH, SAr2, TSi, TSr}
```

Figure 11: IKE_AUTH Exchange with AuCPace

According to [CPace-AuCPace] the result of an AuCPace protocol run is a shared secret sk1 that must be used in AUTH payload computation for key confirmation. One of the possible ways of doing this is shown in Figure 8.


```
AUTHi = prf( prf(sk1, "AuCPace for IKEv2 Initiator"),
                                              <InitiatorSignedOctets>)

AUTHr = prf( prf(sk1, "AuCPace for IKEv2 Responder"),
                                              <ResponderSignedOctets>)
```

Figure 12: AUTH Payload calculation in case of AuCPace

3.2.3.7. VTBPEKE

VTBPEKE is defined in [VTBPEKE]. Possible instantiation of VTBPEKE in IKEv2 is shown on Figure 13.

```
Initiator                                Responder
-------------------------------------------------------------------
HDR, SK {IDi, [IDr,] GSPM(R'),
        SAi2, TSi, TSr}      -->
                             <--          HDR, SK {IDr, GSPM(salt),
                                                   GSPM(e), GSPM(Y)}

HDR, SK {GSPM(X), GSPM(r),
         AUTH}               -->
                             <--      HDR, SK {AUTH, SAr2, TSi, TSr}
```

Figure 13: IKE_AUTH Exchange with VTBPEKE

According to [VTBPEKE] the result of a VTBPERE protocol run is a shared secret sk that must be used in AUTH payload computation for key confirmation. One of the possible ways of doing this is shown in Figure 14.

```
AUTHi = prf( prf(sk, "VTBPEKE for IKEv2 Initiator"),
```

```
                                                  <InitiatorSignedOctets>)

   AUTHr = prf( prf(sk, "VTBPEKE for IKEv2 Responder"),
                                                  <ResponderSignedOctets>)
```

Figure 14: AUTH Payload calculation in case of VTBPEKE

3.2.3.8. BSPAKE

BSPAKE is defined in [BSPAKE]. Possible instantiation of BSPAKE in IKEv2 is shown on Figure 15.


```
   Initiator                                Responder
   ------------------------------------------------------------------
   HDR, SK {IDi, [IDr,] GSPM(R),
           SAi2, TSi, TSr}      -->
                                <--     HDR, SK {IDr, GSPM(B), GSPM(R')}

   HDR, SK {GSPM(A), AUTH}       -->
                                <--       HDR, SK {AUTH, SAr2, TSi, TSr}
```

Figure 15: IKE_AUTH Exchange with BSPAKE

According to [BSPAKE] the result of a BSPAKE protocol run is a shared secret secretKey that must be used in AUTH payload computation for key confirmation. One of the possible ways of doing this is shown in Figure 16.

```
   AUTHi = prf( prf(secretKey, "BSPAKE for IKEv2 Initiator"),
                                                  <InitiatorSignedOctets>)

   AUTHr = prf( prf(secretKey, "BSPAKE for IKEv2 Responder"),
                                                  <ResponderSignedOctets>)
```

Figure 16: AUTH Payload calculation in case of BSPAKE

3.3. Alternative Approaches

3.3.1. Using Secret Generator in IKE_SA_INIT

An alternative approach for using (at least some) PAKE protocols in IKEv2 was suggested by Dan Harkins in [PAKE-ALT]. The idea is that if peers share a secret generator (calculated from the password), they can use it directly in (EC)DH computation performed in the IKE_SA_INIT exchange. This approach has some advantages:

- requires little changes to IKEv2 (but see below)
- increases IKEv2 performance because only single (EC)DH operation is needed (with RFC6467 framework all PAKE protocols discussed in this memo require some group field operations in addition to standard IKEv2 (EC)DH operation)

However, this approach has some limitations and drawbacks:

- initiator's identity need to be sent in the IKE_SA_INIT, that require some change to the protocol
- identity protection is lost (at least for initiator); a method was suggested to get identity protection in this case, but it requires

an additional round trip and in terms of complexity is no better than RFC6467 approach

- a new authentication method is needed (probably NULL Auth [RFC7619] will suffice)
- algorithm agility is limited (since password is tied to a particular group, to change the group peers need to re-share secret generator out of band)
- the approach is applicable only to specific PAKEs which have static secret generator computed from the password (among the candidates only SPEKE satisfies this requirement)

All these considerations make using this approach problematic in general, however in some specific cases (no need for identity protection, no need for algorithm agility etc.) it can be attractive.

3.3.2. Using IKE_INTERMEDIATE Exchange

It is possible to make use of the IKE_INTERMEDIATE exchange [I-D.ietf-ipsecme-ikev2-intermediate] for the purpose of transferring PAKE's messages. This approach would be similar to using RFC6467 framework, except that RFC6467 has "Experimental" status and is not widely supported, while IKE_INTERMEDIATE framework will probably have wider adoption (mostly for the purposes of Post-Quantum Key Exchange).

On the other hand, using IKE_INTERMEDIATE exchange for PAKE protocols yet has been to be defined.

4. Conclusions

IKEv2 proved to be a flexible protocol that can accommodate various extensions. This memo discusses several ways of integration PAKE protocols into IKEv2, making focus on using RFC6467 framework. It is shown that there should be no problems integrating any of the candidate PAKE protocols. Some alternative approaches are also discussed.

5. Security Considerations

Discussion the security of CFRG PAKE candidates is out of scope of this document. This memo only discusses how each of the candidates can be integrated into IKEv2. Moreover, the details of this integration (in particular how the content of AUTH payload is calculated so that key confirmation is properly achieved) is only

sketched and need to be carefully re-examined when concrete PAKEs are selected.

6. References

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC

2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

[RFC5998] Eronen, P., Tschofenig, H., and Y. Sheffer, "An Extension for EAP-Only Authentication in IKEv2", RFC 5998, DOI 10.17487/RFC5998, September 2010, <https://www.rfc-editor.org/info/rfc5998>.

[RFC6467] Kivinen, T., "Secure Password Framework for Internet Key Exchange Version 2 (IKEv2)", RFC 6467, DOI 10.17487/RFC6467, December 2011, <https://www.rfc-editor.org/info/rfc6467>.

[RFC6617] Harkins, D., "Secure Pre-Shared Key (PSK) Authentication for the Internet Key Exchange Protocol (IKE)", RFC 6617, DOI 10.17487/RFC6617, June 2012, <https://www.rfc-editor.org/info/rfc6617>.

[RFC6628] Shin, S. and K. Kobara, "Efficient Augmented Password-Only Authentication and Key Exchange for IKEv2", RFC 6628, DOI 10.17487/RFC6628, June 2012, <https://www.rfc-editor.org/info/rfc6628>.

[RFC6631] Kuegler, D. and Y. Sheffer, "Password Authenticated Connection Establishment with the Internet Key Exchange Protocol version 2 (IKEv2)", RFC 6631, DOI 10.17487/RFC6631, June 2012, <https://www.rfc-editor.org/info/rfc6631>.

[RFC8236] Hao, F., Ed., "J-PAKE: Password-Authenticated Key Exchange by Juggling", RFC 8236, DOI 10.17487/RFC8236, September 2017, <https://www.rfc-editor.org/info/rfc8236>.

[I-D.irtf-cfrg-spake2]
Ladd, W. and B. Kaduk, "SPAKE2, a PAKE", draft-irtf-cfrg-spake2-08 (work in progress), March 2019.

[I-D.krawczyk-cfrg-opaque]
Krawczyk, H., "The OPAQUE Asymmetric PAKE Protocol", draft-krawczyk-cfrg-opaque-02 (work in progress), July 2019.

[SPEKE] Hao, F. and S. F. Shahandashti, "The SPEKE Protocol Revisited", 2014, <https://eprint.iacr.org/2014/585>.

[CPace-AuCPace]
Haase, B. and B. Labrique, "AuCPace: Efficient verifier-based PAKE protocol tailored for the IIoT", 2019, <https://eprint.iacr.org/2018/286>.

[VTBPEKE] Pointcheval, D. and G. Wang, "VTBPEKE: Verifier-based Two-Basis Password Exponential Key Exchange", 2017, <https://www.di.ens.fr/david.pointcheval/Documents/Papers/2017_asiaccsB.pdf>.

[BSPAKE] "BSPAKE", <https://gist.github.com/Sc00bz/ef0951ab98e8e1bac4810f65a42eab1a>.

[IKEv2-IANA]
"Internet Key Exchange Version 2 (IKEv2) Parameters", <http://www.iana.org/assignments/ikev2-parameters>.

[EAP-IANA]
"Extensible Authentication Protocol (EAP) Registry", <https://www.iana.org/assignments/eap-numbers/eap-numbers.xhtml>.

[PAKE-ALT]
, <https://mailarchive.ietf.org/arch/msg/ipsec/LCSDy33kfr2X9m7NsErmR4eKhkw>.

[RFC7619] Smyslov, V. and P. Wouters, "The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7619, DOI 10.17487/RFC7619, August 2015, <https://www.rfc-editor.org/info/rfc7619>.

[I-D.ietf-ipsecme-ikev2-intermediate]
Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-ietf-ipsecme-ikev2-intermediate-02 (work in progress), July 2019.

Author's Address

Valery Smyslov
ELVIS-PLUS

Email: svan@elvis.ru