REQ1: A PAKE scheme MUST clearly state its features regarding balanced/augmented versions.

SPEKE is a balanced PAKE but it can easily be made augmented (e.g. the B-SPEKE protocol).

REQ2: A PAKE scheme SHOULD come with a security proof and clearly state its assumptions and models.

A security proof for SPEKE in the random oracle model based on a new assumption, "the Decision Inverted-Additive Diffie-Hellman" was published in:

Philip MacKenzie, On the Security of the SPEKE Password-Authenticated Key Exchange Protocol,
Cryptology ePrint Archive, Report 2001/057,
2001, https://eprint.iacr.org/2001/057

REQ3: The authors SHOULD show how to protect their PAKE scheme implementation in hostile environments, particularly, how to implement their scheme in constant time to prevent timing attacks.

SPEKE uses a password-derived generator to perform a Diffie-Hellman key exchange.
Derivation of this generator can be done in constant time using a CFRG-approved
hash-to-element technique.

REQ4: If the PAKE scheme is intended to be used with ECC, the authors SHOULD discuss their requirements for a potential mapping or define a mapping to be used with the scheme.

SPEKE can be used with ECC and, as noted under REQ3, an appropriate hash-to-curve
algorithm can be used to obtain the password-derived generator.

REQ5: The authors of a PAKE scheme MAY discuss its design choice with regard to performance, i.e., its optimization goals.

In addition to the operations necessary to map the password to a generator, SPEKE
reuires two modular exponentiations.

Another benefit of SPEKE is that as a balanced PAKE it just requires the same
password to be provisioned on the two peers. There is no secure channel between
them that is assumed to exist or have existed and there are no storage requirements
placed on implementations.

REQ6: The authors of a scheme MAY discuss variations of their

> scheme
>   that allow the use in special application scenarios. In
>   particular, techniques that facilitate long-term (public)
> key
>   agreement are encouraged.

(Note: this is not being written by the author of the scheme).

Techniques to facilitate long-term public key agreement can be built on top of
SPEKE much in the same way that PKEX built on top of SPAKE2. Because there is
no requirement for a secure channel to provision SPEKE there is no catch-22.

> REQ7: Authors of a scheme MAY discuss special ideas and solutions
> on
>   privacy protection of its users.

SPEKE provides no privacy protection but it doesn't preclude it either. It could
be possible to implement a scheme similar to that used by TLS-PWD (RFC 8492)
at the cost of a secure channel at provisioning time.

> REQ8: The authors MUST follow the IRTF IPR policy
>   <https://irtf.org/ipr>.

SPEKE was patented with U.S. Patent 6,226,383. That patent expired in March 2007.
There are no other patents that are known to apply to SPEKE.

Additional questions from Stanislav Smyshlyaev on 5 July 2019

> How does it meet the "SHOULD" requirements of RFC 8125?

I do believe so, yes.

> Does it meet "crypto agility" requirements, not fixing any
> particular primitives
> and/or parameters?

Very much so.

> What setting is the PAKE suitable for, which applications does it
> have?
> "Peer communication" (where both ends share the same password)
> or "client-server"
> (where the server does not store the password but only a one-way
> function of the
> password)?

As a balanced PAKE, SPEKE is suitable for peer communications. It can also be implemented

in a true peer-to-peer protocol where there are no "initiators" and "responders".

> A nomination should specify for which use-cases the protocol is recommended ("PAKE
> as a more-secure replacement for a PSK on a machine-2-machine interface" or "PAKE
> for securely accessing a remote HMI interface server (e.g. a web server) without
> configured WEB-PKI certificates").

This is definitely a more secure replacement for a PSK. Any kind of deployment where each
side agrees on a passcode/passphrase, from establishing keys to secure routing messages
to IoT devices with a limited UI. Any application that traditionally was done with "enter
password here".

> Can two communicating parties initiate the key exchange process at the same time,
> or must it always be the case that only one party can initiate the process?

SPEKE could be adapted into such a key exchange, such as the (abandoned) IKEv3 proposal.

> Is it suitable to be considered as a standalone scheme?

Yes.

> Can it be integrated into IKEv2? TLS Handshake? Any other protocols?

IKEv2 definitely. As a balanced PAKE it is perfect for this. For TLS it could be made
to work but honestly an augmented PAKE would be better for TLS. For other protocols,
it would be ideal for LAKE or an ACE protocol.

> Is there a publicly available security proof? If yes,
> Are there known problems with the proof?

Not that I know of.

> Is the considered security model relevant for all applications that PAKE is intended
> for (e.g., if a PAKE is to be used in TLS Handshake, it is important that the TLS
> adversary model is considered for the PAKE)?

I believe so.

> Does it allow to be sure in sufficient level of security for

common values of
password lengths?

Yes.

Security assessment.
Does its security depend on any nontrivial implementation properties? Are they
clearly stated in the document?

No.

Does the PAKE have precomputation security (for augmented PAKEs)?

N/A

If the PAKE relies on the assumption of a trusted setup - more specifically, the
discrete logarithm relationship between the two group elements in the system setup
MUST be unknown - in anticipation of the worst but not impossible scenario, the
authors should clearly state the security implications when the discrete logarithm
relationship becomes known, and the subsequent mitigation measures.

N/A

Performance assessment.
What's with the “round efficiency” of the PAKE? In a standard two/multi-party secure
computation setting, the “round” is defined as a step in which all parties can
complete operations at the same time without depending on each other. In practice,
a 2-round protocol could be implemented as 2 flows or 3 flows depending on the
application context, but that’s more the implementation detail.

The SPEKE protocol is simply a Diffie-Hellman but to do a secure PAKE it is necessary to
confirm the established keys. So this would be a 3- or 4-message exchange.

How many operations of each type (scalar multiplications, inversions in finite
fields, hash calculations etc.) are made by each side?

Excluding the hash-to-element calculations to derive the shared generator there are
two modular exponentiations/point multiplications to derive the shared secret and

two keyed hash calls for key confirmation.

Which recommendations for secure usage can be given?
Is it defined how the explicit key confirmation is performed/must be performed
externally? Are there clear statements whether this procedure is optional or
mandatory?

Key confirmation is mandatory.

Can any recommendations on using iterated hashing (e.g., with Scrypt) with the
PAKE be given?

No.

Can any recommendations to avoid a user enumeration attack be given?

Throttle back on negotiations when unsuccessful authentication attempts exceed some threshold
and simulate the protocol with random data when an "invalid" username is presented.