Answers to IETF PAKE Selection Questions on the J-PAKE Nomination

Feng Hao

16 July, 2019

Expanded answers for all positions of RFC 8125 regarding their PAKEs

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

[Answer] J-PAKE is a balanced scheme. It's completely symmetric except that the identities used by the two communication parties must differ. It provides the following security features (see https://tools.ietf.org/html/rfc8236#section-6).

1. Off-line dictionary attack resistance – It does not leak any information that allows a passive/active attacker to perform off-line exhaustive search of the password.
2. Forward secrecy – It produces session keys that remain secure even when the password is later disclosed.
3. Known-session security – It prevents a disclosed session from affecting the security of other established session keys.
4. On-line dictionary attack resistance – It limits an active attacker to test only one password per each protocol execution.

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

[Answer] Security proofs were provided with the original publication of J-PAKE in [HR08] (the journal version of the paper published in [HR10]). They show the protocol satisfies the following four security requirements (defined above): offline dictionary attack resistance, forward secrecy, known-session security and lin-line dictionary attack resistance, under the Computational Diffie-Hellman or Decisional Diffie-Hellman assumption in the random oracle model (a summary of the proved results can be found in Table 1 in [HR10]). An independent formal analysis of J-PAKE that proves the security of the protocol in a formal model with algebraic adversaries and random oracles is presented by Abdalla, BenHamouda and MacKenzie in [ABM15].

[ABM15] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. "Security of the J-PAKE password-authenticated key exchange protocol." IEEE Symposium on Security and Privacy, 2015. https://www.normalesup.org/~fbenhamo/files/publications/SP_AbdBenMac15.pdf

[HR10] Feng Hao, Peter Ryan, "J-PAKE: Authenticated Key Exchange Without PKI," Springer Transactions on Computational Science XI, Special Issue on Security in Computing, Part II, Vol. 6480, pp. 192-206, 2010. http://eprint.iacr.org/2010/190.pdf

[HR08] Feng Hao, Peter Ryan, "Password Authenticated Key Exchange by Juggling," Proceedings of the 16th Workshop on Security Protocols (SPW'08), Cambridge, UK, LNCS 6615, pp. 159-171, 2008. https://www.dcs.warwick.ac.uk/~fenghao/files/pw.pdf

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.

[Answer] J-PAKE only involves basic group operations. Obviously, the implementation of such standard group operations should follow the best practice in secure coding and avoid any potential side-channel information leakage. In J-PAKE, the only operation that directly involves the shared secret (denoted s) is the multiplication of s with an ephemeral secret (see [HR10]). This operation should incur a negligible cost as compared to the rest of the operations such as exponentiation. It's highly unlikely that this trivial operation will leak side-channel information about the secret but for the risk-averse to deal with a hostile environment, one may use a hashed secret as s to ensure the constant-time computation even for this trial operation.

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.

[Answer] J-PAKE requires no mapping function.

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

[Answer] One could optimize the computation by re-using the square terms during the calculation of exponentiations. This is explained in the last paragraph of Section 5 in [HR10], and is copied below:

"...the protocol enumerates 14 exponentiations for each user, but actually many of the operations are merely repetitions. To explain this, let the bit length of the exponent be L = log2 q. Computing g^{x1} alone requires roughly 1.5L multiplications which include L square operations and 0.5L multiplications of the square terms. However, the same square operations need not be repeated for other items with the same base g (i.e., g^{x2}, etc). This provides plenty of room for efficiency optimization in a practical implementation."

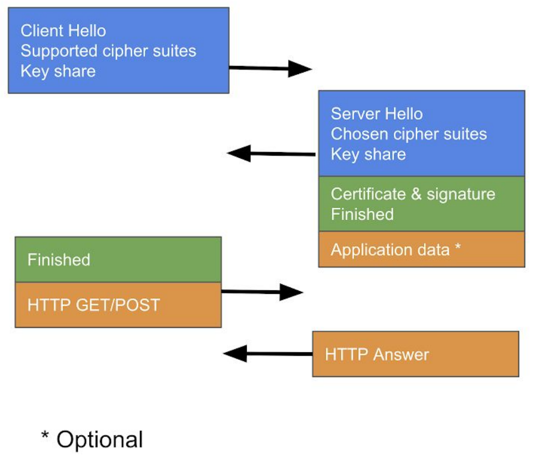
REQ6: The authors of a scheme MAY discuss variations of their scheme that allow the use in special application scenarios. In particular, techniques that facilitate long-term (public) key agreement are encouraged.

[Answer] There are two variations: J-PAKE can be implemented as either two rounds or three flows depends on the application requirements (see https://tools.ietf.org/html/rfc8236#section-4). The only difference is that in the former, the two parties can initiate the key exchange at the same time without having to wait for the other party, which can prove useful in some applications, e.g., mesh networks.

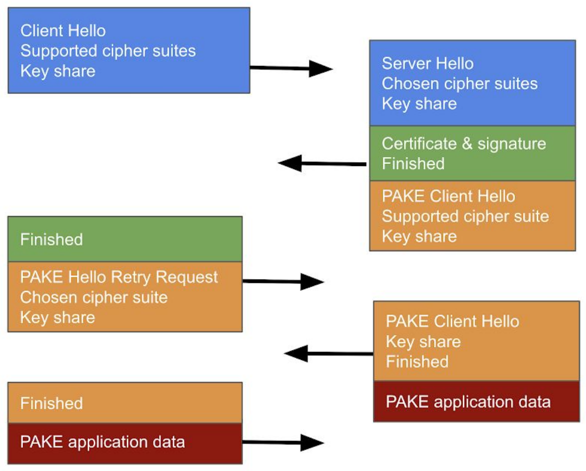
It's possible to use J-PAKE to bootstrap long-term (public) key agreement by first establishing a password-authenticated secret channel, and then through this channel exchange long-term public keys (a process called "personal registration" as opposed to "CA registration"; see Section III.A in https://eprint.iacr.org/2010/136.pdf). Subsequently, the two communicating parties use the received long-term public keys to establish future secure channels based on a public-key authenticated key exchange protocol. The advantages of this approach are: 1) it allows using a public-key authenticated key exchange protocol "without requiring a PKI"; 2) the use of a public-key authenticated key exchange protocol complements the security of PAKE by preventing online dictionary attacks in future sessions. I recommend pairing J-PAKE with YAK (https://en.wikipedia.org/wiki/YAK_(cryptography)), which is constructed based on the same design principle as J-PAKE, but uses public keys to achieve authentication rather than passwords. However, J-PAKE can flexibly pair with any secure public-key authenticated key exchange protocols.

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

[Answer] The privacy of the user identities can be protected through the password+certificate mode as presented below:



TLS 1.3 handshake



J-PAKE/TLS 1.3 handshake
(password + certificate)

In this mode, we use TLS 1.3 to bootstrap the J-PAKE process. All J-PAKE messages are transmitted within the TLS encrypted channels. Once the J-PAKE process is completed, all subsequent data will be encrypted by using the session key derived from J-PAKE. The benefits of combining the password and the server certificate in this way are: i) the protection of the user identities under TLS encrypted channel under normal cases that a PKI is not corrupted; ii) protection of the user password against phishing attacks as compared to sending the password over TLS; iii) strengthened guarantee of end-to-end security even under the exceptional case that a PKI is corrupted (e.g., a DigiNotar-like rogue or self-signed certificate is added to the client certificate store to enable man-in-the-middle interceptions of data at the enterprise or state levels); iv) strengthened protection of the client against impersonation attacks, e.g., through on-line dictionary search (since the server has to present a valid certificate).

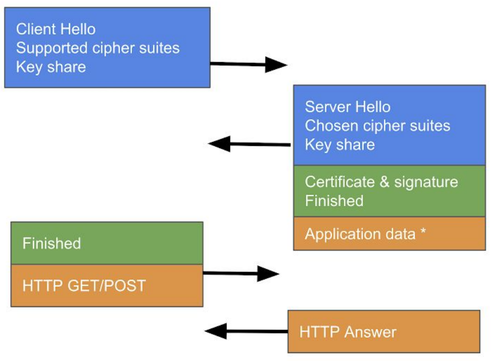
REQ8: The authors MUST follow the IRTF IPR policy <https://irtf.org/ipr>.

[Answer] J-PAKE is not encumbered by patents.

Opinions on the following questions

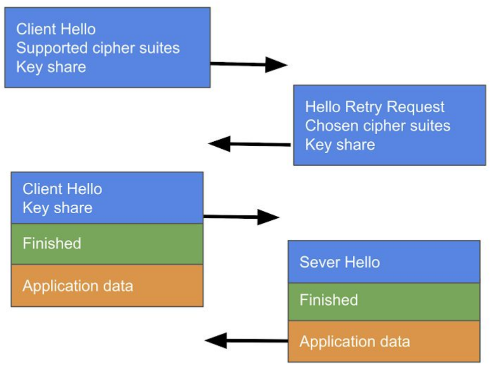
- How does it meet the "SHOULD" requirements of RFC 8125? [Answer] we believe it meets the requirements fully.
- Does it meet "crypto agility" requirements, not fixing any particular primitives and/or parameters? [Answer] Yes, the protocol flexibly works with any finite multiplicative or additive groups that are suitable for cryptography. The protocol only uses the most basic group operations (mul/exp in a finite field setting and +/* in an elliptic curve setting) without requiring any mapping function or trusted setup.
- 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)? [Answer] peer communication
- 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"). [Answer] It can be used as a more secure replacement for a PSK when the initial shared secret is weak. Once the J-PAKE key exchange is completed, the subsequent sessions may use the standard PSK (since the shared secret is a high-entropy key now).
- 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? [Answer] Yes, they can initiate the key exchange process at the same time without having to wait for the other.
- Is it suitable to be considered as a standalone scheme? [Answer] Yes.
- Can it be integrated into IKEv2? TLS Handshake? Any other protocols? [Answer] Yes, it can be integrated into TLS handshakes, e.g., as below (integration into IKEv2 and other protocols should also be possible but we haven't looked into those in detail; we can investigate if needed.)



* Optional

TLS 1.3 handshake



J-PAKE/TLS 1.3 handshake (password only)

- Is there a publicly available security proof? If yes, [Answer] Yes
  - Are there known problems with the proof? [Answer] No known problem.
  - 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)? [Answer] Yes, we use standard security models that consider both passive (with the forward secrecy) and active adversaries. It's worth noting that PAKE and TLS are complementary. The security model of TLS critically depends on

    the integrity of a PKI, while PAKE is designed to function even when the PKI is completely corrupted or unavailable.
  - Does it allow to be sure in sufficient level of security for common values of password lengths? [Answer] Yes. In fact, the level of security provided by J-PAKE does not depend on the password length except for the likelihood of a successful online dictionary attack.
- Security assessment.
  - Does its security depend on any nontrivial implementation properties? Are they clearly stated in the document? [Answer] J-PAKE doesn't require any mapping function, nor a trusted setup. It only uses basic group operations and a standard hash function.
  - Does the PAKE have precomputation security (for augmented PAKEs)? [Answer] N/A (J-PAKE is a balanced scheme)
  - 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. [Answer] J-PAKE doesn't assume a trusted setup.
- 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. [Answer] J-PAKE requires 2 rounds for achieving implicit key confirmation and an extra round for achieving explicit key confirmation.
  - How many operations of each type (scalar multiplications, inversions in finite fields, hash calculations etc.) are made by each side? [Answer] In the finite field setting, each party performs 14 modular exponentiations (https://tools.ietf.org/html/rfc8236#section-2.3). In the elliptic curve setting, each party performs 11 scale multiplications (https://tools.ietf.org/html/rfc8236#section-3.3). The rest operations including one inversion in the finite field, 6 standard hash calculations (3 used in generating the three zero-knowledge proofs and 3 in verifying these ZKPs) and 1 standard key derivation function (KDF) in the end for each party incur a negligible cost by comparison.
- 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? [Answer] J-PAKE flexibly works with any secure explicit key confirmation method. Two examples are provided in https://tools.ietf.org/html/rfc8236#section-5. This procedure is optional in J-PAKE.

- Can any recommendations on using iterated hashing (e.g., with Scrypt) with the PAKE be given? [Answer] No particular recommendations to offer (this question is most relevant to augmented PAKE)
- Can any recommendations to avoid a user enumeration attack be given? [Answer] Yes. We consider two types of enumeration attacks: against the password and against the user identity. Our main concern is the former (assuming the user identity is public), for which we have three recommendations. 1) set a limit for online authentication failures (see https://tools.ietf.org/html/rfc8236#section-6) and add increasing penalty delays for consecutively failed attempts. 2) running J-PAKE inside a TLS session (addressed in REQ7); 3) pair J-PAKE with a secure public-key authenticated key exchange protocol (addressed in REQ6); 4) pair J-PAKE with PSK (J-PAKE used to bootstrap the pre-shared keys). For avoiding the enumeration attack against the user identity (in applications where the user identity is considered private), our recommendation is that rather than immediately rejecting the connection with an invalid user identity error, the responding party uses a random value (taken from the range of [1, q] but extremely unlikely to be a matching password) as the password to engage with the rest of the protocol and communicates an authentication failure in the end to the other party. However, we should note that this may not be sufficient to ensure a constant-time response in a complex system as the operation of checking the user identity in the database may incur variation in time depending on whether the user identity exists or not. To obscure the variations, one may add a random delay for checking the user identity and increase the delay after the consecutively failed attempts.

Acknowledgements: I thank Joona Kannisto and Robert Cragie for many helpful discussions while writing this document.