Overview of existing PAKEs and PAKE selection criteria

Stanislav V. Smyshlyaev, Ph.D.
CISO, CryptoPro LLC

CFRG
IETF 104, March 2019, Prague

PAKE selection process: IETF 103

IETF 103, chairs' slides

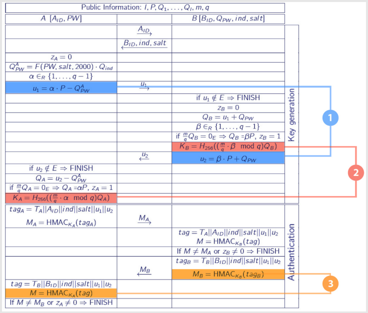
- After receiving several PAKE proposals recently and seeing several CFRG documents complete, chairs want to announce PAKE selection process.
- The aim is to select one or more PAKE to recommend to the wider IETF community.
- All submissions need to satisfy RFC 8125 (Requirements for PAKE Schemes).
- Suggestions on how to pick the best PAKE are very welcome! Which questions should CFRG chairs ask?
- Involving Crypto Review Panel to come up with recommendations.

PAKE selection process: IETF 103

Opinions

- Support of the process at the CFRG session (“and please do it soon”) and later at the TLS and IPSECME sessions.
- Maybe changing “one or more” with “zero or more”?
- Better to select one without a variety of options.
- Need to think what can be the “formal” result (a separate document with “we think that this one is the best one”?..).
- Parallels with the selection process for elliptic curves.

A very general scheme of a typical (balanced) PAKE



Public Information: $I, P, Q_1, \ldots, Q_l, m, q$

$A\ [A_{ID}, PW]$		$B\ [B_{ID}, Q_{PW}, ind, salt]$
	$\xrightarrow{A_{ID}}$	
	$\xleftarrow{B_{ID}, ind, salt}$	
$z_A = 0$		
$Q^A_{PW} = F(PW, salt, 2000) \cdot Q_{ind}$		
$\alpha \in_R \{1, \ldots, q-1\}$		
$u_1 = \alpha \cdot P - Q^A_{PW}$	$\xrightarrow{u_1}$	
		if $u_1 \notin E \Rightarrow$ FINISH
		$z_B = 0$
		$Q_B = u_1 + Q_{PW}$
		$\beta \in_R \{1, \ldots, q-1\}$
		if $\frac{m}{q} Q_B = 0_E \Rightarrow Q_B = \beta P,\ z_B = 1$
		$K_B = H_{256}((\frac{m}{q} \cdot \beta \mod q) Q_B)$
	$\xleftarrow{u_2}$	$u_2 = \beta \cdot P + Q_{PW}$
if $u_2 \notin E \Rightarrow$ FINISH		
$Q_A = u_2 - Q^A_{PW}$		
if $\frac{m}{q} Q_A = 0_E \Rightarrow Q_A = \alpha P,\ z_A = 1$		
$K_A = H_{256}((\frac{m}{q} \cdot \alpha \mod q) Q_A)$		
$tag_A = T_A \Vert A_{ID} \Vert ind \Vert salt \Vert u_1 \Vert u_2$ $M_A = \mathrm{HMAC}_{K_A}(tag_A)$	$\xrightarrow{M_A}$	
		$tag = T_A \Vert A_{ID} \Vert ind \Vert salt \Vert u_1 \Vert u_2$ $M = \mathrm{HMAC}_{K_B}(tag)$
		If $M \neq M_A$ or $z_B \neq 0 \Rightarrow$ FINISH
	$\xleftarrow{M_B}$	$tag_B = T_B \Vert B_{ID} \Vert ind \Vert salt \Vert u_1 \Vert u_2$ $M_B = \mathrm{HMAC}_{K_B}(tag_B)$
$tag = T_B \Vert B_{ID} \Vert ind \Vert salt \Vert u_1 \Vert u_2$		
$M = \mathrm{HMAC}_{K_A}(tag)$		
If $M \neq M_B$ or $z_A \neq 0 \Rightarrow$ FINISH		

- 1: sending key shares protected under passwords
- 2: computing a shared key
- 3 (optional): explicit key confirmation

- Balanced: both sides store the same representation of password.
- Augmented: one side maintains a transform of the password and the other maintains the raw password.

Selection process: general questions

Possible usage of PAKEs: TLS, IPsec, messengers, IoT etc.
One PAKE for all applications? Or distinct sets of requirements?

Examples

1. An augmented (and secure against attacks involving precomputations) PAKE is good for client-server protocols — but may be redundant for one-to-one communications (messengers? Wi-Fi DPP?).
2. Explicit key confirmation stage may be good for usage a PAKE „by itself“, but may be redundant for usage in IKEv2 and TLS Handshake.

Selection process: general questions

“Usage of PAKE with TLS 1.3”, draft-barnes-tls-pake-04

For usage with TLS 1.3 PAKE must be:

- Possible to execute in one round-trip, with the client speaking first.
- The Finished MAC must provide sufficient key confirmation for the protocol, taking into account the contents of the handshake messages.
- Providing forward secrecy.

Examples: SPAKE+, SPEKE, DragonFly, OPAQUE, SRP.

- For key establishment in messengers?
- For M2M/IoT?
- ...

Possible selection criteria: questions for each PAKE (1)

1. Does it meet the requirements of RFC 8125?
2. How does it meet the „SHOULD“ requirements of RFC 8125?
3. Is there a publicly available security proof? If yes,
    1. are there known problems with the proof?
    2. is the considered security model relevant for IKEv2 and TLS Handshake cases?
    3. does it allow to be sure in sufficient level of security for common values of password lengths?

Possible selection criteria: questions for each PAKE (2)

4. Does it include elements, which might be redundant for usage in IKEv2 and TLS Handshake?
5. Does its security depend on any nontrivial implementation properties? Are they clearly stated in the document?
6. Does it meet „crypto agility“ requirements, not fixing any particular primitives and/or parameters?
7. How many round trips are needed by the protocol?
8. How many operations of each type (scalar multiplications, inversions in finite fields, hash calculations etc.) are made by each side?

PAKEs in CFRG (starting from 2013)

- Dragonfly, RFC 7664
- SPAKE2, draft-irtf-cfrg-spake2-07
- SPAKE2+, draft-irtf-cfrg-spake2-07
- AugPAKE, draft-irtf-cfrg-augpake-09
- SESPAKE, RFC 8133
- J-PAKE, RFC 8236
- PKEX, draft-harkins-pkex-06
- VTBPEKE, a talk at IETF 101
- OPAQUE, draft-krawczyk-cfrg-opaque-01

PAKEs elsewhere

- SRP, RFC 2945
- IEEE P1363.2: balanced PAKEs: PAK, PPK, SPEKE
- IEEE P1363.2: augmented PAKEs: AMP, BSPEKE2, PAKZ, WSPEKE, SRP
- EKE
- Augmented-EKE
- PAK-X
- ITU-T X.1035

Sorry if I forgot your favorite PAKE!

Dragonfly, RFC 7664

- Balanced
- Requires mapping of passwords to points on an elliptic curve.
- Security proof: Lancrenon, J. and M. Skrobot, "On the Provable Security of the Dragonfly Protocol".
- "Elliptic curve groups used with Dragonfly authentication MUST have a cofactor of one".
- A method of deterministically mapping a secret string into an element in a selected group is required.
- RFC 8492, "Secure Password Ciphersuites for TLS".
- In a nutshell:

$$\begin{aligned}
&\mathrm{A, B : P_{pw} = F(pw) \in E} \\
&\mathrm{A \rightarrow B : x_A + y_A; -y_A \cdot P_{pw}} \\
&\mathrm{A \leftarrow B : x_B + y_B; -y_B \cdot P_{pw}} \\
&\mathrm{A, B : SK = KDF(x_A \cdot x_B \cdot P_{pw})} \\
&\text{Mutual key confirmation for SK}
\end{aligned}$$

SPAKE2, draft-irtf-cfrg-spake2-08

- Balanced
- The discrete logarithms of the public role-specific elements must be unknown, determining them must be computationally infeasible.
- Security proof: Abdalla, M. and D. Pointcheval, "Simple Password-Based Encrypted Key Exchange Protocols."
- Ongoing discussions on addressing cofactors in DH etc.
- In a nutshell:

$$\begin{aligned} \mathrm{A} \rightarrow \mathrm{B} &: \mathrm{x_A} \cdot \mathrm{P} + \mathrm{pw} \cdot \mathrm{P_1} \\ \mathrm{A} \leftarrow \mathrm{B} &: \mathrm{x_B} \cdot \mathrm{P} + \mathrm{pw} \cdot \mathrm{P_2} \\ \mathrm{A}, \mathrm{B} &: \mathrm{SK} = \mathrm{KDF}(\mathrm{x_A} \cdot \mathrm{x_B} \cdot \mathrm{P}) \end{aligned}$$

SPAKE2+, draft-irtf-cfrg-spake2-08

- Augmented
- The discrete logarithms of the public role-specific elements must be unknown, determining them must be computationally infeasible.
- draft-barnes-tls-pake-04, “Usage of PAKE with TLS 1.3”, SPAKE2+ in TLS 1.3 has been addressed.
- No separate security proof for SPAKE2+, elements of security assessment in: Cash, D., Kiltz, E., and V. Shoup, “The Twin-Diffie Hellman Problem and Applications”
- Ongoing discussions on addressing cofactors in DH etc.
- In a nutshell:

$$
\begin{aligned}
&(\mathrm{w}_0, \mathrm{w}_1) = \mathrm{KDF}(\mathrm{pw}),\ \mathrm{B\ stores\ only\ w}_0\ \mathrm{and}\ \mathrm{w}_1 \cdot \mathrm{P} \\
&\mathrm{A} \rightarrow \mathrm{B} : \mathrm{x}_\mathrm{A} \cdot \mathrm{P} + \mathrm{w}_0 \cdot \mathrm{P}_1 \\
&\mathrm{A} \leftarrow \mathrm{B} : \mathrm{x}_\mathrm{B} \cdot \mathrm{P} + \mathrm{w}_0 \cdot \mathrm{P}_2 \\
&\mathrm{A}, \mathrm{B} : \mathrm{SK}_0 = \mathrm{KDF}(\mathrm{x}_\mathrm{A} \cdot \mathrm{x}_\mathrm{B} \cdot \mathrm{P}) \\
&\mathrm{A}, \mathrm{B} : \mathrm{SK}_1 = \mathrm{KDF}(\mathrm{w}_1 \cdot \mathrm{x}_\mathrm{B} \cdot \mathrm{P})
\end{aligned}
$$

SESPAKE, RFC 8133

- Balanced
- The discrete logarithms of the public role-specific elements must be unknown, determining them must be computationally infeasible.
- Security proof: Smyshlyaev, S., Oshkin, I., Alekseev, E. Ahmetzyanova, L., "On the Security of One Password Authenticated Key Exchange Protocol"
- Can be thought of as slightly modified SPAKE2 with explicit key confirmation.
- In a nutshell:

$$\begin{aligned}
&\mathrm{A} \rightarrow \mathrm{B} : \mathrm{x_A} \cdot \mathrm{P} - \mathrm{pw} \cdot \mathrm{P}' \\
&\mathrm{A} \leftarrow \mathrm{B} : \mathrm{x_B} \cdot \mathrm{P} + \mathrm{pw} \cdot \mathrm{P}' \\
&\mathrm{A}, \mathrm{B} : \mathrm{SK} = \mathrm{KDF}(\mathrm{x_A} \cdot \mathrm{x_B} \cdot \mathrm{P}) \\
&\text{Mutual key confirmation for SK}
\end{aligned}$$

AugPAKE, draft-irtf-cfrg-augpake-09 (expired)

- Augmented
- Security proof: Shin, S., Kobara, K., Imai, H., "Security Proof of AugPAKE", incomplete security proof.
- Pre-shared key is generated completely by B.
- In a nutshell:

$$\begin{aligned}
&\mathrm{A} \rightarrow \mathrm{B} : \mathrm{y_A} \cdot \mathrm{P} \\
&\mathrm{A} \leftarrow \mathrm{B} : \mathrm{x_B} \cdot (\mathrm{y_A} \cdot \mathrm{P} + \mathrm{r} \cdot \{\mathrm{pw} \cdot \mathrm{P}\}) \\
&\mathrm{A}, \mathrm{B} : \mathrm{SK} = \mathrm{KDF}(\mathrm{x_B} \cdot \mathrm{P}) \\
&\text{Mutual key confirmation for pre-shared key SK}
\end{aligned}$$

J-PAKE, RFC 8236

- Balanced
- Follows a completely different design approach from all other PAKE protocols, and is built upon a Zero Knowledge Proof (ZKP) primitive: Schnorr NIZK proof.
- Security proof: Abdalla, M., Benhamouda, F., MacKenize, P. "Security of the J-PAKE Password-Authenticated Key Exchange Protocol"
- Does not require trusted setup. Recommendations for key confirmation are given.
- Requires more computations than the other considered protocols.
- In a nutshell:

$$\begin{aligned}
&\mathrm{A} \rightarrow \mathrm{B} : \mathrm{x}_\mathrm{A}^1 \cdot \mathrm{P}, \mathrm{x}_\mathrm{A}^2 \cdot \mathrm{P}, \mathrm{ZKP}(\mathrm{x}_\mathrm{A}^1), \mathrm{ZKP}(\mathrm{x}_\mathrm{A}^2)\\
&\mathrm{A} \leftarrow \mathrm{B} : \mathrm{x}_\mathrm{B}^1 \cdot \mathrm{P}, \mathrm{x}_\mathrm{B}^2 \cdot \mathrm{P}, \mathrm{ZKP}(\mathrm{x}_\mathrm{B}^1), \mathrm{ZKP}(\mathrm{x}_\mathrm{B}^2)\\
&\mathrm{A} \rightarrow \mathrm{B} : (\mathrm{x}_\mathrm{A}^1 + \mathrm{x}_\mathrm{B}^1 + \mathrm{x}_\mathrm{B}^2) \cdot \mathrm{s} \cdot \mathrm{x}_\mathrm{A}^2 \cdot \mathrm{P}, \mathrm{ZRP}(\mathrm{s} \cdot \mathrm{x}_\mathrm{A}^2)\\
&\mathrm{B} \rightarrow \mathrm{A} : (\mathrm{x}_\mathrm{B}^1 + \mathrm{x}_\mathrm{A}^1 + \mathrm{x}_\mathrm{A}^2) \cdot \mathrm{s} \cdot \mathrm{x}_\mathrm{B}^2 \cdot \mathrm{P}, \mathrm{ZRP}(\mathrm{s} \cdot \mathrm{x}_\mathrm{B}^2)\\
&\mathrm{A}, \mathrm{B} : \mathrm{SK} = \mathrm{KDF}((\mathrm{x}_\mathrm{A}^1 + \mathrm{x}_\mathrm{B}^1) \cdot \mathrm{x}_\mathrm{A}^2 \cdot \mathrm{x}_\mathrm{B}^2 \cdot \mathrm{P})
\end{aligned}$$

PKEX, draft-harkins-pkex-06

- Balanced
- The discrete logarithms of the public role-specific elements must be unknown, determining them must be computationally infeasible.
- No security proof (but the first half is based on SPAKE2).
- Can be thought of as: a two-phase protocol with SPAKE2 at phase 1 and expilcit key confirmation with exchanging new public keys with binding to identities at phase 2.
- Privacy: binding public keys to identities, providing anonymity in subsequent communications.
- In a nutshell:

$$
\begin{aligned}
&\mathrm{A} \rightarrow \mathrm{B} : \mathrm{x_A} \cdot \mathrm{P} + \mathrm{pw} \cdot \mathrm{P_1} \\
&\mathrm{A} \leftarrow \mathrm{B} : \mathrm{x_B} \cdot \mathrm{P} + \mathrm{pw} \cdot \mathrm{P_2} \\
&\mathrm{A, B} : \mathrm{SK} = \mathrm{KDF}(\mathrm{x_A} \cdot \mathrm{x_B} \cdot \mathrm{P}) \\
&\mathrm{A} \rightarrow \mathrm{B} : \mathrm{E_{SK}}(\mathrm{z_A} \cdot \mathrm{P}, \mathrm{HMAC}_{\mathrm{z_A} \cdot \mathrm{x_B} \cdot \mathrm{P}}(\mathrm{z_A} \cdot \mathrm{P}, \mathrm{x_A} \cdot \mathrm{P}, \mathrm{x_B} \cdot \mathrm{P})) \\
&\mathrm{A} \leftarrow \mathrm{B} : \mathrm{E_{SK}}(\mathrm{z_B} \cdot \mathrm{P}, \mathrm{HMAC}_{\mathrm{z_B} \cdot \mathrm{x_A} \cdot \mathrm{P}}(\mathrm{z_B} \cdot \mathrm{P}, \mathrm{x_A} \cdot \mathrm{P}, \mathrm{x_B} \cdot \mathrm{P}))
\end{aligned}
$$

TBPEKE, VTBPEKE

- TBPEKE: balanced; VTBPEKE: augmented.
- The discrete logarithms of the public role-specific elements must be unknown, determining them must be computationally infeasible.
- Security proof: Pointcheval, D., Wang, G., “VTBPEKE: Verifier-based Two-Basis Password Exponential Key Exchange”.
- TBPEKE: a generalization of SPEKE on arbitrary cyclic groups.
- TBPEKE in a nutshell:

$$\begin{aligned}
&\mathrm{A},\mathrm{B}:\mathrm{G}=\mathrm{U}+\mathrm{pw}\cdot\mathrm{V}\\
&\mathrm{A}\rightarrow\mathrm{B}:\mathrm{x_A}\cdot\mathrm{G}\\
&\mathrm{A}\leftarrow\mathrm{B}:\mathrm{x_B}\cdot\mathrm{G}\\
&\mathrm{A},\mathrm{B}:\mathrm{SK}=\mathrm{KDF}(\mathrm{x_A}\cdot\mathrm{x_B}\cdot\mathrm{G})
\end{aligned}$$

- VTBPEKE: augmented version of TBPEKE.

OPAQUE, draft-krawczyk-cfrg-opaque-01

- Augmented, secure against precomputations.
- Security proof: Jarecki, S. et al., ”OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-Computation Attacks”.
- Proof is modular: applies to the composition of any OPRF with any KCI-secure KE.
- draft-sullivan-tls-opaque-00, “Usage of OPAQUE with TLS 1.3”.
- Recommendations on privacy.
- “Compilation” of a secure AKE to augmented PAKE using OPRF.
- In a nutshell: registration phase:

$$\mathrm{A}, \mathrm{B} : \mathrm{RW} = \mathrm{OPRF}(\mathrm{K_B}, \mathrm{pw})$$
$$\mathrm{A} : \mathrm{Envelope_A} = \mathrm{AuthEnc_{RW}}(\mathrm{Priv_A}, \mathrm{Pub_A}, \mathrm{Pub_B})$$
$$\mathrm{A} \rightarrow \mathrm{B} : \mathrm{Envelope_A}, \mathrm{Pub_A}$$

- Authentication phase:

$$\mathrm{A}, \mathrm{B} : \mathrm{RW} = \mathrm{OPRF}(\mathrm{K_B}, \mathrm{pw})$$
$$\mathrm{A} \leftarrow \mathrm{B} : \mathrm{Envelope_A}$$
$$\mathrm{A}, \mathrm{B} : \mathrm{AKE\ on\ keys\ Priv_A}, \mathrm{Priv_B}$$

PAKE requirements (RFC 8125)

«R1: A PAKE scheme MUST clearly state its features regarding balanced/augmented versions.»

- Dragonfly: balanced
- SPAKE2: balanced
- SPAKE2+: augmented
- AugPAKE: augmented
- SESPAKE: balanced
- J-PAKE: balanced
- PKEX: balanced
- VTBPEKE: augmented
- OPAQUE: augmented (with security against precomputations)

«R2: A PAKE scheme SHOULD come with a security proof and clearly state its assumptions and models.»

Important notice: credibility and clearness of the security proofs for the candidates — to be carefully verified by Crypto Review Panel.

- Dragonfly: yes
- SPAKE2: yes
- SPAKE2+: partial (H. Krawczyk: "... no security proof")
- AugPAKE: yes (S. Smyshlyaev: "incomplete"; H. Krawczyk: "... no security proof")
- SESPAKE: yes
- J-PAKE: yes
- PKEX: no (according to draft-harkins-pkex-06)
- VTBPEKE: yes (H. Krawczyk: "... in a weak model that allows for precomputation attacks")
- OPAQUE: yes

R3–R8

- R3: recommendations for protection in hostile environments.
- R4: for ECC: mappings to be used.
- R5: optimization goals.
- R6: comments on special application scenarios.
- R7: privacy considerations.
- R8: status with respect to patents.

— more about a document, not a protocol itself.

Further steps after we select one (or more)

- Selection for usage in IETF protocols is not the same as selection of one PAKE for usage “by itself”.
- Recommendations for usage in protocols should be given (e.g., key confirmation, handling the counters of failed attempts of authentication, handling errors, etc.).
- If we create a new CFRG document (RFC on one or more PAKEs with additional blessing(s) from CFRG? “Recommendations for usage of PAKEs in IETF protocols”?), the recommendations should be given there.
- Recommendations for generation of parameters should be given: e.g., SPAKE, SESPAKE and PKEX need that the discrete logarithms of the public role-specific elements are unknown, and determining them is computationally infeasible.

Questions to consider

- We have several PAKEs.
- We have some criteria to compare them.
- We can give some recommendations for usage in protocols.

Questions to consider

- We have several PAKEs.
  — Suggestions to add/remove PAKEs from the list of candidates?
- We have some criteria to compare them.
  — Suggestions to add/remove criteria?
- We can give some recommendations for usage in protocols.
  — What else should we address in some future document?

Backup slides

What do we want from one PAKE?

General PAKE requirements

- Conformance to RFC 8125, «Requirements for PAKE schemes».
- Reasonable efficiency.
- Crypto agility (choice of elliptic curves, hash functions).

What do we want from one PAKE?

PAKE requirements: security

- Impossibility for an active adversary to obtain criteria for password.
- Clear statements about the security properties with credible security assessments provided.
- Recommendations for security in hostile environments (protection against side-channel attacks etc.)

SPAKE2

The SPAKE2 protocol contains a key agreement step only. During the key agreement step the parties exchange keys using Diffie-Hellman with public components masked by element that depends on the password - one of the predefined elliptic curve points multiplied by the password-based coefficient. This approach provides an implicit key authentication, which means that after this step one party is assured that no other party aside from a specifically identified second party may gain access to the generated secret key.

SPAKE2

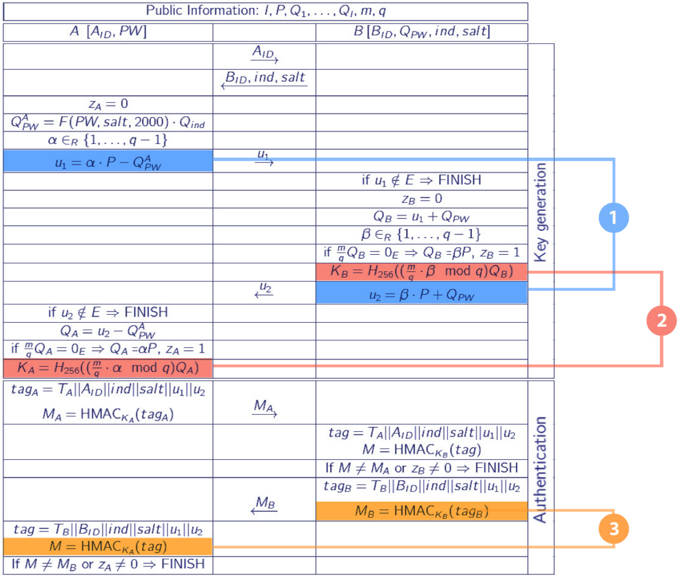
Public Information: $\mathbb{E}, P, M, N, q, \mathcal{D}$		
$A: \ ID_A, pw$		$B: \ ID_B, pw$
$x_A \in_{\mathcal{U}} \{1, \dots, q-1\}$		$x_B \in_{\mathcal{U}} \{1, \dots, q-1\}$
$X_A = x_A \cdot P + pw \cdot M$		$X_B = x_B \cdot P + pw \cdot N$
	$\xrightarrow{ID_A, X_A}$	$X_A \overset{?}{\in} \mathbb{E} \setminus \{0\}$
$X_B \overset{?}{\in} \mathbb{E} \setminus \{0\}$	$\xleftarrow{ID_B, X_B}$	
$SK_A = x_A \cdot (X_A - pw \cdot N)$		$SK_B = x_B \cdot (X_B - pw \cdot M)$
$K_A = H(X_A \| X_B \| pw \| K_A)$		$K_B = H(X_A \| X_B \| pw \| K_B)$

The SPAKE2 protocol for EC Group

SESPAKE

The SESPAKE protocol consists of two steps: the key agreement step and the key confirmation step. The first step is identical to SPAKE2. During the key confirmation step the parties exchange strings that strongly depend on the generated key. After this step the parties are assured that a legitimate party and no one else actually has possession of the secret key.

SESPAKE



Public Information: $I, P, Q_1, \ldots, Q_l, m, q$

$A\ [A_{ID}, PW]$		$B\ [B_{ID}, Q_{PW}, ind, salt]$	
	$\xrightarrow{A_{ID}}$		Key generation
	$\xleftarrow{B_{ID}, ind, salt}$		
$z_A = 0$			
$Q^A_{PW} = F(PW, salt, 2000) \cdot Q_{ind}$			
$\alpha \in_R \{1, \ldots, q-1\}$			
$u_1 = \alpha \cdot P - Q^A_{PW}$	$\xrightarrow{u_1}$		
		if $u_1 \notin E \Rightarrow$ FINISH	
		$z_B = 0$	
		$Q_B = u_1 + Q_{PW}$	
		$\beta \in_R \{1, \ldots, q-1\}$	
		if $\frac{m}{q} Q_B = 0_E \Rightarrow Q_B = \beta P,\ z_B = 1$	
		$K_B = H_{256}((\frac{m}{q} \cdot \beta \mod q) Q_B)$	
	$\xleftarrow{u_2}$	$u_2 = \beta \cdot P + Q_{PW}$	
if $u_2 \notin E \Rightarrow$ FINISH			
$Q_A = u_2 - Q^A_{PW}$			
if $\frac{m}{q} Q_A = 0_E \Rightarrow Q_A = \alpha P,\ z_A = 1$			
$K_A = H_{256}((\frac{m}{q} \cdot \alpha \mod q) Q_A)$			
$tag_A = T_A \Vert A_{ID} \Vert ind \Vert salt \Vert u_1 \Vert u_2$ $M_A = \mathrm{HMAC}_{K_A}(tag_A)$	$\xrightarrow{M_A}$		Authentication
		$tag = T_A \Vert A_{ID} \Vert ind \Vert salt \Vert u_1 \Vert u_2$ $M = \mathrm{HMAC}_{K_B}(tag)$	
		If $M \neq M_A$ or $z_B \neq 0 \Rightarrow$ FINISH	
		$tag_B = T_B \Vert B_{ID} \Vert ind \Vert salt \Vert u_1 \Vert u_2$	
	$\xleftarrow{M_B}$	$M_B = \mathrm{HMAC}_{K_B}(tag_B)$	
$tag = T_B \Vert B_{ID} \Vert ind \Vert salt \Vert u_1 \Vert u_2$			
$M = \mathrm{HMAC}_{K_A}(tag)$			
If $M \neq M_B$ or $z_A \neq 0 \Rightarrow$ FINISH			

Dragonfly and AugPAKE

The Dragonfly and AugPAKE protocols consist of two steps too. Both protocols use the additional key diversification technique: the parties exchange strings for the key confirmation step which are obtained using the common secret generated on the key agreement step with one one-way function. The target session key is obtained from the same secret with another one-way function.

Dragonfly and AugPAKE

In the Dragonfly protocol a special mapping from the password set to the EC point group is used: every password corresponds to a group generator. During the key agreement step the parties exchange Diffie-Hellman keys and their masked multiplicities with respect to the password-based generator.
In the AugPAKE protocol the server computes a secret value and sends it to the client masking with the password-based element and the client's point. In order to demask the obtained value we should know password and multiplicity of the client's point.

Public Information: $\mathbb{E}, q, \mathcal{D}$		
$A:\ ID_A, pw$		$B:\ ID_B, pw$
$P_{pw} = F(pw) \in \mathbb{E}$		$P_{pw} = F(pw) \in \mathbb{E}$
$x_A, y_A \in_{\mathcal{U}} \{1, \ldots, q-1\}$		$x_B, y_B \in_{\mathcal{U}} \{1, \ldots, q-1\}$
$s_A = x_A + y_A \bmod q$		$s_B = x_B + y_B \bmod q$
$X_A = -x_A \cdot P_{pw}$		$X_B = -x_B \cdot P_{pw}$
	$\xrightarrow{ID_A, X_A, s_A}$	$X_A \overset{?}{\in} \mathbb{E} \setminus \{0\}$
$X_B \overset{?}{\in} \mathbb{E} \setminus \{0\}$	$\xleftarrow{ID_B, X_B, s_B}$	
$SK_A = y_A \cdot (s_B \cdot P_{pw} + X_B)$		$SK_B = y_B \cdot (s_A \cdot P_{pw} + X_A)$
$K_A = H(SK_A \Vert X_A \Vert X_B \Vert s_A \Vert s_B \Vert 1)$		$K_A = H(SK_B \Vert X_A \Vert X_B \Vert s_A \Vert s_B \Vert 1)$

$T_A = H(SK_A \Vert X_A \Vert X_B \Vert s_A \Vert s_B \Vert 2)$	$\xrightarrow{T_A}$	
		$T_A \overset{?}{=} H(SK_B \Vert X_A \Vert X_B \Vert s_A \Vert s_B \Vert 2)$
	$\xleftarrow{T_B}$	$T_B = H(SK_B \Vert X_A \Vert X_B \Vert s_A \Vert s_B \Vert 3)$
$T_B \overset{?}{=} H(SK_A \Vert X_A \Vert X_B \Vert s_A \Vert s_B \Vert 3)$		

The Dragonfly protocol for EC Group

Public Information: $\mathbb{E}, P, q, \mathcal{D}$

$A:\ ID_A, pw$		$B:\ ID_B, W = pw \cdot P$
$x_A \in_{\mathcal{U}} \{1, \dots, q-1\}$		$x_B \in_{\mathcal{U}} \{1, \dots, q-1\}$
$X_A = x_A \cdot P$		$SK_B = x_B \cdot P$
	$\xrightarrow{ID_A, X_A}$	$X_A \overset{?}{\in} \mathbb{E}\backslash\{0\}$
$r = H(ID_A \Vert ID_B \Vert X_A)$		$r = H(ID_A \Vert ID_B \Vert X_A)$
$X_B \overset{?}{\in} \mathbb{E}\backslash\{0\}$	$\xleftarrow{ID_B, X_B}$	$X_B = x_B \cdot (X_A + r \cdot W)$
$SK_A = \frac{1}{x_A + pw \cdot r} \cdot X_B$		
$K_A = H(ID_A \Vert ID_B \Vert X_A \Vert X_B \Vert SK_A \Vert 1)$		$K_B = H(ID_A \Vert ID_B \Vert X_A \Vert X_B \Vert SK_B \Vert$
$T_A = H(ID_A \Vert ID_B \Vert X_A \Vert X_B \Vert SK_A \Vert 2)$	$\xrightarrow{T_A}$	
		$T_A \overset{?}{=} H(ID_A \Vert ID_B \Vert X_A \Vert X_B \Vert SK_B \Vert$
	$\xleftarrow{T_B}$	$T_B = H(ID_A \Vert ID_B \Vert X_A \Vert X_B \Vert SK_B \Vert$
$T_B \overset{?}{=} H_2(ID_A \Vert ID_B \Vert X_A \Vert X_B \Vert SK_A \Vert 3)$		

The AugPAKE protocol for EC Group

«R3: The authors SHOULD show how to protect an implementation of their PAKE scheme in hostile environments, particularly, how to implement their scheme in constant time to prevent timing attacks»

One example scenario that should be taken into account: if temporary points can become (due to some specific MitM attack) zero points, a fake calculation scenario preventing timing attacks should be described. For most PAKEs careful handling of the counters of false attempts (incrementing them before the start of the protocol) should be enough. Should be carefully verified by Crypto Review Panel for main canditates.

«R4: In case 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.»

All "PAKEs in CFRG (starting from 2013)" support usage with ECC. Clearness and completeness of requirements for mappings for main candidates should be verified by Crypto Review Panel.

«R5: A PAKE scheme MAY discuss its design choice with regard to performance, i.e., its optimization goals.»

An independent performance evaluation regarding target protocols (at least, TLS 1.3 and IKE) should be done. Special reviews from TLS and IPSECME on performance?

«R6: The authors of a scheme MAY discuss variations of their scheme that allow the use in special application scenarios. In particular, techniques that allow agreeing on a long-term (public) key are encouraged.»

The protocols can be used for secure channel establishment — the core issue here is the allowed security level. These questions are discussed in the paper with security proofs and can be added to the documents.

«R7: A scheme MAY discuss special ideas and solutions on privacy protection of its users.»

A discussion of privacy protection is present in OPAQUE draft and in PKEX draft.

«R8: The authors MUST declare the status of their scheme with respect to patents.»

Availability for free use seems to be a necessary condition for selecting a PAKE as a recommended one.

1: sending key shares protected under passwords

Core part of a PAKE — deriving ephemeral public keys, encrypting and exchanging them.

2: computing a shared key

Computations needed to obtain a shared key (or a master key that is used for further key derivation).

3 (optional): explicit key confirmation

Additional messages needed to make both parties sure that the key establishment phase succeeded.

A complete PAKE with explicit key confirmation may be useful by itself, but may not be convenient for integration in other protocols — e.g., consider ClientFinished and ServerFinished.

Explicit key confirmation is a part of some PAKEs (e.g., SESPAKE) and is explicitly discussed in other ones (e.g., J-PAKE).

Question for discussion: Should we select a “minimalistic” PAKE without explicit key confirmation in the end?

Augmentation = additional tasks for an adversary in two attack scenarios:
The attacker gets the stored user password-related information from the server, doesn't spend his resources to brute-force pw from f(pw) and impersonates the user on ...

1. ... other servers, where the user used the same pw.
2. ... the same compromised server.

H. Krawczyk: "it should only be open to inevitable attacks: online impersonation attempts with guessed user passwords and offline dictionary attacks upon the compromise of a server and leakage of its password file"
Better to have it in client-server protocols to add protection in case of compromised servers.
Question for discussion: are we always ready to use augmented protocols, despite the fact we don't need the augmentation in many cases? Or select two PAKEs: one balanced and one augmented?

Something about each PAKE

- Dragonfly — seems to be redundant for integration to existing protocols: internal key confirmation.
- SPAKE2 — balanced; solid construction with security proof, no redudancy.
- SPAKE2+ — in contrast to OPAQUE, does not accomodate secret salt.
- AugPAKE — issues with security proof, seems to be redundant for integration to existing protocols: internal key confirmation.
- SESPAKE — has explicit key confirmation stage, redundant for usage in existing protocols.
- J-PAKE — completely different, notably slower.
- PKEX — based on a solid and well-studied construction, but a security proof still needs to be provided.
- VTBPEKE — no draft, augmented without protection against precomputations.
- OPAQUE — a -01 draft with a skeleton; more details needed.