Security Analysis of SPAKE2+

Victor Shoup

(NYU)
Traditional password authentication

\[ P \ (\text{client}) \]
secret: \( \pi \)

\[ Q \ (\text{server}) \]
secret: \( \text{salt}, h := H(\pi, id_P, id_Q, \text{salt}) \)

\[ \pi \rightarrow \text{Test } h \overset{?}{=} H(\pi, id_P, id_Q, \text{salt}) \]

... plus one-sided authenticated key exchange

- Client/server run a one-sided authenticated key exchange protocol, using server’s public key
- Client/server use established key to build a secure channel
  - Client “knows” he is talking securely to server
  - Server “knows” he is talking securely to “somebody”
- Client/server run traditional password authentication protocol over the secure channel
  - Now server “knows” who he is really talking to
Limitations of this traditional approach:

- Requires PKI
- Phishing attacks

**PAKE: Password Authenticated Key Exchange**

- introduced by [Bellovin, Merritt 1992]
- Eliminates need for PKI
- Prevents *offline* dictionary attacks
  - An adversary that actively interacts with client or server effectively gets just one guess at the password
  - An adversary that passively observes client and server effectively gets *no* information about the password
  - This holds even if adversary learns (information derived from) the session key
Protocol SPAKE0

shared secret password: $\pi$

\[
\begin{align*}
P & \quad Q \\
x & \leftarrow_R \mathcal{R} & \quad y & \leftarrow_R \mathcal{R} \\
\xrightarrow{\chi} & & \xleftarrow{y}
\end{align*}
\]

session key: $k := H(\pi, id_P, id_Q, x, y)$

**Problem:** Eavesdropper can mount an offline dictionary attack

- attacker sees $x, y$ and (say) $t = HMAC(k, m)$ for a known message $m$
- attacker tries all passwords $\pi' \in Dict$ and test if $t \overset?= HMAC(k', m)$, where $k' := H(\pi', id_P, id_Q, x, y)$
Protocol SPAKE1

shared secret password: \( \pi \)

\[ P \]
\[ \alpha \leftarrow_R \mathbb{Z}_q, u \leftarrow g^\alpha \]
\[ Q \]
\[ \beta \leftarrow_R \mathbb{Z}_q, \nu \leftarrow g^\beta \]
\[ u \]
\[ \nu \]
\[ w \leftarrow \nu^\alpha \]
\[ w \leftarrow u^\beta \]

session key: \( k := H(\pi, id_P, id_Q, u, \nu, w) \)

- CDH \( \implies \) eavesdropper cannot mount an offline dictionary attack
- active attacker can still mount an offline dictionary attack
- attacker runs protocol as \( Q \) against honest \( P \), so knows \( u, \nu, w \)
- attacker tries all passwords \( \pi' \in Dict \) and test if \( t ? = HMAC(k', m) \), where \( k' := H(\pi', id_P, id_Q, u, \nu, w) \)
Protocel SPAKE2

public system parameters: random $a, b \in \mathbb{G}$
shared secret password: $\pi \in \mathbb{Z}_q$

$P$
\[
\begin{align*}
\alpha &\leftarrow_R \mathbb{Z}_q, u \leftarrow g^\alpha a^\pi \\
\beta &\leftarrow_R \mathbb{Z}_q, v \leftarrow g^\beta b^\pi
\end{align*}
\]

$Q$
\[
\begin{align*}
u &\leftarrow_R \mathbb{Z}_q, v \leftarrow g^\beta b^\pi
\end{align*}
\]

\[
\begin{align*}
w &\leftarrow (v/b^\pi)^\alpha \\
w &\leftarrow (u/a^\pi)^\beta
\end{align*}
\]

session key: $k := H(\pi, id_P, id_Q, u, v, w)$

- From [Abdalla, Pointcheval 2005]
- CDH + Random Oracle $\implies$ no offline dictionary attacks
- only online dictionary attacks are possible — cannot be avoided
Limitation of SPAKE2: symmetry

Typical scenario:

- Client memorizes $\pi$
- Server stores $\pi$ in a password file

Password file compromised $\implies$ all passwords immediately compromised

**Asymmetric PAKE:** [Gentry, MacKenzie, Ramzan 2006]

Protection against password file compromise

In order to impersonate client to server, attacker must carry out an offline dictionary attack even if password file is compromised
**Protocol SPAKE2+**

public system parameters: random $a, b \in \mathbb{G}$
password: $\pi$, $(\phi_0, \phi_1) := F(\pi, id_P, id_Q)$

$P$ (client)
secret: $\phi_0, \phi_1$

$\alpha \leftarrow_R \mathbb{Z}_q, u \leftarrow g^\alpha a^{\phi_0}$

$Q$ (server)
secret: $\phi_0, c := g^{\phi_1}$

$\beta \leftarrow_R \mathbb{Z}_q, \nu \leftarrow g^\beta b^{\phi_0}$

$w \leftarrow (\nu/b^{\phi_0})^\alpha$

d $\leftarrow (\nu/b^{\phi_0})^{\phi_1}$

$w \leftarrow (u/a^{\phi_0})^\beta$

d $\leftarrow c^\beta$

session key: $k := H(\pi, id_P, id_Q, u, \nu, w, d)$

- From [Cash, Kiltz, Shoup 2008; Boneh, Shoup 2008]
- Currently being standardized
- Unproven claim: provides resilience against password file compromise
Limitation of SPAKE2+: pre-processing attacks

For a given pair of users $P$ and $Q$, attacker can precompute $(\phi'_0, \phi'_1) := F(\pi', id_P, id_Q)$ for all $\pi' \in Dict$

As soon as the attacker obtains $(\phi_0, c)$ from password file, attacker can perform a quick table lookup to determine $\pi$

We will not address this limitation here, but see:

Strong asymmetric PAKE: [Jarecki, Krawczyk, Xu 2018]

Protection against pre-processing attacks

In order to impersonate client to server, attacker must carry out an offline dictionary attack AFTER password file is compromised
Original goal of this work:

Prove the claim: CDH + Random Oracle $\implies$ SPAKE2+ is a secure asymmetric PAKE

Two popular security models for PAKE:

- **BPR model**: game based [Bellare, Pointcheval, Rogaway 2000]
  - But ... no extension to asymmetric PAKE :-(

- **UC (Universal Composability) model**: simulation based [Canetti, Halevi, Katz, Lindell, MacKenzie 2005]
  - Extends to asymmetric PAKE :-) [Gentry, MacKenzie, Ramzan 2006]
  - But ... SPAKE2 is not even secure in symmetric UC model :-(
  - For the same reason, SPAKE2+ cannot be secure in the asymmetric UC model :-(
Main results of this work:

- Define a new protocol
  \[ \text{kcSPAKE2}^+ \approx (\text{SPAKE2}^+) + (\text{key-confirmation}) \]

- Prove that kcSPAKE2+ is a secure asymmetric PAKE in the \textit{UC model} (assuming CDH + RO)

Along the way, we also:

- Prove that kcSPAKE2 is a secure symmetric PAKE in the \textit{UC model} (assuming CDH + RO)

- Prove that a variant of kcSPAKE2+ currently being standardized is a secure asymmetric PAKE in the \textit{UC model}

- Fix a few problems in the current definitions of UC secure symmetric and asymmetric PAKE
∀A ∃S ∀Z :  \text{Exec}[Π, A, Z] \approx \text{Exec}[\mathcal{F}, S, Z]
Interface for symmetric PAKE (both real and ideal)

- Many clients $P$, each associated with a unique server $Q$
- Many servers $Q$, each associated with a unique client $P$
- Each client-server pair $(P, Q)$ has a shared password $\pi$
  - $\mathcal{Z}$ chooses $\pi$ arbitrarily
- $\mathcal{Z}$ initiates many protocol instances of a client or server
- When a protocol instance terminates, it outputs either
  - abort, or
  - $(\textit{sid}, k)$, where
    - $\textit{sid}$ is a “session ID”
    - $k$ is a “session key”
- Intuition about session IDs:
  - For a given client-server pair $(P, Q)$ and a given $\textit{sid}$:
    - At most one instance of $P$ should hold $\textit{sid}$
    - At most one instance of $Q$ should hold $\textit{sid}$
    - Instances holding $\textit{sid}$ should hold same $k$
Ideal functionality for symmetric PAKE

- $S$ may make a single *password guess* on any protocol instance:
  - $S$ gives $\pi'$ to $F$
  - $F$ tells $S$ if $\pi' = \pi$

- $S$ instructs $F$ how to generate protocol instance $I$’s output:
  - **abort**: $I$ outputs abort
  - **(fresh-key, sid)**: no password guess allowed on $I$
    - $F$ chooses $k$ at random, and $I$ outputs $(sid, k)$
  - **(copy-key, sid)**: no password guess allowed on $I$, and there must be a unique compatible instance with the same sid, with a “fresh” key $k$
    - $I$ outputs $(sid, k)$
  - **(spoiled-key, sid, k)**: $S$ must have made a successful password guess on $I$
    - $I$ outputs $(sid, k)$
From symmetric to asymmetric PAKE

- New interface elements:
  - $Z$ can compromise a server $Q$
    - In the real world, $A$ obtains $Q$’s “password file”
    - In the ideal world, $S$ is allowed to assign “spoiled keys” to any instance of the corresponding client $P$
  - $Z$ can make explicit queries to a random oracle $F$ at inputs $(\pi', id_P, id_Q)$
    - Idea: queries to $F$ are “externally visible” events
    - In the real world, $A$ obtains $(\pi', id_P, id_Q)$ along with $F(\pi', id_P, id_Q)$
      - $A$ does not have direct access to $F$
    - In the ideal world, after a server is compromised, $S$ may make corresponding “offline password guesses”

- This repairs problems in previous work identified by [Hesse 2019]
Why isn’t SPAKE2 UC secure?

• “Theorem”: Protocol SPAKE2 is not UC secure (according to my definition — or any others in the literature)
  ◦ Details need to be worked out . . .

• More fundamentally: any secure-channels protocol layered directly on top of Protocol SPAKE2 is not UC secure either

• In concurrent work, [Abdalla, et al 2020] also observe that SPAKE2 is not UC secure
  ◦ They show that SPAKE2 is UC secure w/r to a much weaker ideal functionality: “lazy extraction security”
    · and they use a stronger and ”non-falsifiable” assumption: Gap CDH
  ◦ Fact: any secure-channels protocol layered on top of a “lazy extraction secure” PAKE protocol cannot be UC secure in any reasonable sense
    · so it’s not clear what the applications are
  ◦ They show that “lazy extraction secure” PAKE + key-confirmation = UC secure PAKE
    · so perhaps their security notion is useful for modular proofs
Why isn’t SPAKE2 UC secure?

\[ \alpha \leftarrow_R \mathbb{Z}_q, u \leftarrow g^\alpha a^{\pi'} \]

\[ \beta \leftarrow_R \mathbb{Z}_q, v \leftarrow g^\beta b^\pi \]

\[ w \leftarrow (u/a^\pi)^\beta \]

\[ k \leftarrow H(\pi, id_P, id_Q, u, v, w) \]

Q starts encrypting using \( k \)

Simulator must *immediately* decide if \( k \) is “fresh” or “spoiled”

\[ w' \leftarrow (v/b^{\pi'})^\alpha \]

\[ k' \leftarrow H(\pi', id_P, id_Q, u, v, w') \]

But only now can simulator test if \( \pi' = \pi \)
**Protocol kcSPAKE2+**

- **Public system parameter:** random $a \in \mathbb{G}$
- **Password:** $\pi$, $(\phi_0, \phi_1) := F(\pi, id_P, id_Q)$

---

**P (client)**

- **Secret:** $\phi_0, \phi_1$

  \[
  \alpha \leftarrow_R \mathbb{Z}_q, \ u \leftarrow g^{\alpha} a^{\phi_0}
  \]

**Q (server)**

- **Secret:** $\phi_0, c := g^{\phi_1}$

  \[
  \beta \leftarrow_R \mathbb{Z}_q, \ \nu \leftarrow g^{\beta}
  \]

  \[
  w \leftarrow (u / a^{\phi_0})^{\beta}, \ d \leftarrow c^{\beta}
  \]

  \[
  (k, k_1, k_2) \leftarrow H(\phi_0, id_P, id_Q, u, \nu, w, d)
  \]

  \[
  H(\phi_0, id_P, id_Q, u, \nu, w, d)
  \]

**Validate $k_1$**

\[
\]

**Validate $k_2$**

\[
\]

**Session key:** $k$