# Security Analysis of SPAKE2+

Victor Shoup

(NYU)

# Traditional password authentication

| $P$ (client) | | $Q$ (server) |
|---|---|---|
| secret: $\pi$ | | secret: $salt, h := H(\pi, id_P, id_Q, salt)$ |
| | $\xrightarrow{\pi}$ | Test $h \stackrel{?}{=} H(\pi, id_P, id_Q, salt)$ |

# ... 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]*
- Eliminiates 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: $\pi$

| $P$ | | $Q$ |
|---|---|---|
| $x \leftarrow_R \mathcal{R}$ | | $y \leftarrow_R \mathcal{R}$ |
| | $\xrightarrow{\quad x \quad}$ | |
| | $\xleftarrow{\quad y \quad}$ | |

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 $t \stackrel{?}{=} HMAC(k', m)$, where $k' := H(\pi', id_P, id_Q, x, y)$

# Protocol SPAKE1

shared secret password: $\pi$

| $P$ | | $Q$ |
|---|---|---|
| $\alpha \leftarrow_R \mathbb{Z}_q, u \leftarrow g^\alpha$ | | $\beta \leftarrow_R \mathbb{Z}_q, v \leftarrow g^\beta$ |
| | $\xrightarrow{\ u\ }$ | |
| | $\xleftarrow{\ v\ }$ | |
| $w \leftarrow v^\alpha$ | | $w \leftarrow u^\beta$ |

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

- 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 $u, v, w$
- attacker tries all passwords $\pi' \in Dict$ and test if $t \stackrel{?}{=} HMAC(k', m)$, where $k' := H(\pi', id_P, id_Q, u, v, w)$

# Protocol SPAKE2

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

| $P$ | | $Q$ |
|---|---|---|
| $\alpha \leftarrow_R \mathbb{Z}_q, u \leftarrow g^\alpha a^\pi$ | | $\beta \leftarrow_R \mathbb{Z}_q, v \leftarrow g^\beta b^\pi$ |
| | $\xrightarrow{\quad u \quad}$ | |
| | $\xleftarrow{\quad v \quad}$ | |
| $w \leftarrow (v/b^\pi)^\alpha$ | | $w \leftarrow (u/a^\pi)^\beta$ |

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

- From *[Abdalla, Pointcheval 2005]*
- CDH + Random Oracle $\Longrightarrow$ 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 $\Longrightarrow$ 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) | | $Q$ (server) |
|---|---|---|
| secret: $\phi_0, \phi_1$ | | secret: $\phi_0, c := g^{\phi_1}$ |
| $\alpha \leftarrow_R \mathbb{Z}_q, u \leftarrow g^{\alpha} a^{\phi_0}$ | | $\beta \leftarrow_R \mathbb{Z}_q, v \leftarrow g^{\beta} b^{\phi_0}$ |
| | $\xrightarrow{\quad u \quad}$ | |
| | $\xleftarrow{\quad v \quad}$ | |
| $w \leftarrow (v/b^{\phi_0})^{\alpha}$ | | $w \leftarrow (u/a^{\phi_0})^{\beta}$ |
| $d \leftarrow (v/b^{\phi_0})^{\phi_1}$ | | $d \leftarrow c^{\beta}$ |

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

- 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 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 $\Longrightarrow$ 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 (*U*niversal *C*omposability) 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

  kcSPAKE2+ $\approx$ (SPAKE2+) $+$ (key-confirmation)

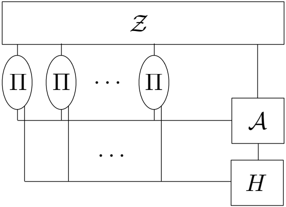
- Prove that kcSPAKE2+ is a secure *asymmetric PAKE* in the *UC model* (assuming CDH + RO)

**Along the way, we also:**

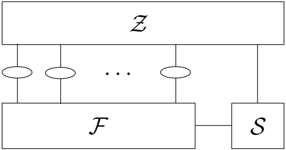
- Prove that kcSPAKE2 is a secure *symmetric PAKE* in the *UC model* (assuming CDH + RO)
- Prove that a variant of kcSPAKE2+ currently being standardized is a secure *asymmetric PAKE* in the *UC model*
- Fix a few problems in the current definitions of UC secure symmetric and asymmetric PAKE

# UC framework

Real world




Ideal world




$$\forall \mathcal{A} \;\; \exists \mathcal{S} \;\; \forall \mathcal{Z} : \quad \text{Exec}[\Pi, \mathcal{A}, \mathcal{Z}] \approx \text{Exec}[\mathcal{F}, \mathcal{S}, \mathcal{Z}]$$

## 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
  - $(sid, k)$, where
    - $sid$ is a"session ID"
    - $k$ is a "session key"
- Intuition about session IDs:
  - For a given client server pair $(P, Q)$ and a given $sid$:
    - At most one instance of $P$ should hold $sid$
    - At most one instance of $Q$ should hold $sid$
    - Instances holding $sid$ should hold same $k$

# Ideal functionality for symmetric PAKE

- $\mathcal{S}$ may make a single *password guess* on any protocol instance:
  - $\mathcal{S}$ gives $\pi'$ to $\mathcal{F}$
  - $\mathcal{F}$ tells $\mathcal{S}$ if $\pi' = \pi$
- $\mathcal{S}$ instructs $\mathcal{F}$ how to generate protocol instance $I$'s output:
  - `abort`: $I$ outputs `abort`
  - (`fresh-key`, *sid*): *no password guess allowed on I*

    $\mathcal{F}$ 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*): *$\mathcal{S}$ must have made a successful password guess on I*

    $I$ outputs $(sid, k)$

## From symmetric to asymmetric PAKE

- New interface elements:
  - $\mathcal{Z}$ can *compromise* a server $Q$
    - In the real world, $\mathcal{A}$ obtains $Q$'s "password file"
    - In the ideal world, $\mathcal{S}$ is allowed to assign "spoiled keys" to any instance of the corresponding client $P$
  - $\mathcal{Z}$ 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, $\mathcal{A}$ obtains $(\pi', id_P, id_Q)$ along with $F(\pi', id_P, id_Q)$
      - $\mathcal{A}$ does not have direct access to $F$
    - In the ideal world, after a server is compromised, $\mathcal{S}$ 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?

$\widehat{P}$ ( $\pi'$ ) | $Q$ ( $\pi$ )

$\alpha \leftarrow_R \mathbb{Z}_q, u \leftarrow g^{\alpha} a^{\pi'}$ | $\beta \leftarrow_R \mathbb{Z}_q, v \leftarrow g^{\beta} b^{\pi}$

$\xrightarrow{\quad u \quad}$

$\xleftarrow{\quad v \quad}$

$w \leftarrow (u/a^{\pi})^{\beta}$
$k \leftarrow H(\pi, id_P, id_Q, u, v, w)$
$Q$ starts encrypting using $k$
Simulator must *immediately* decide
if $k$ is "fresh" or "spoiled"

$\vdots$

$w' \leftarrow (v/b^{\pi'})^{\alpha}$
$k' \leftarrow H(\pi', id_P, id_Q, u, v, w')$
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) | | $Q$ (server) |
|---|---|---|
| secret: $\phi_0, \phi_1$ | | secret: $\phi_0, c := g^{\phi_1}$ |
| $\alpha \leftarrow_R \mathbb{Z}_q, u \leftarrow g^{\alpha} a^{\phi_0}$ | $\xrightarrow{u}$ | $\beta \leftarrow_R \mathbb{Z}_q$, $v \leftarrow g^{\beta}$<br>$w \leftarrow (u/a^{\phi_0})^{\beta}$, $d \leftarrow c^{\beta}$<br>$(k, k_1, k_2) \leftarrow$<br>$H(\phi_0, id_P, id_Q, u, v, w, d)$ |
| $w \leftarrow v^{\alpha}$, $d \leftarrow v^{\phi_1}$<br>$(k, k_1, k_2) \leftarrow$<br>$H(\phi_0, id_P, id_Q, u, v, w, d)$ | $\xleftarrow{v, k_1}$ | |
| validate $k_1$ | | |
| | $\xrightarrow{k_2}$ | validate $k_2$ |

session key: $k$