(Strong) aPAKE Revisited:
Capturing Multi-User Security and Salting

Dennis Dayanikli Anja Lehmann

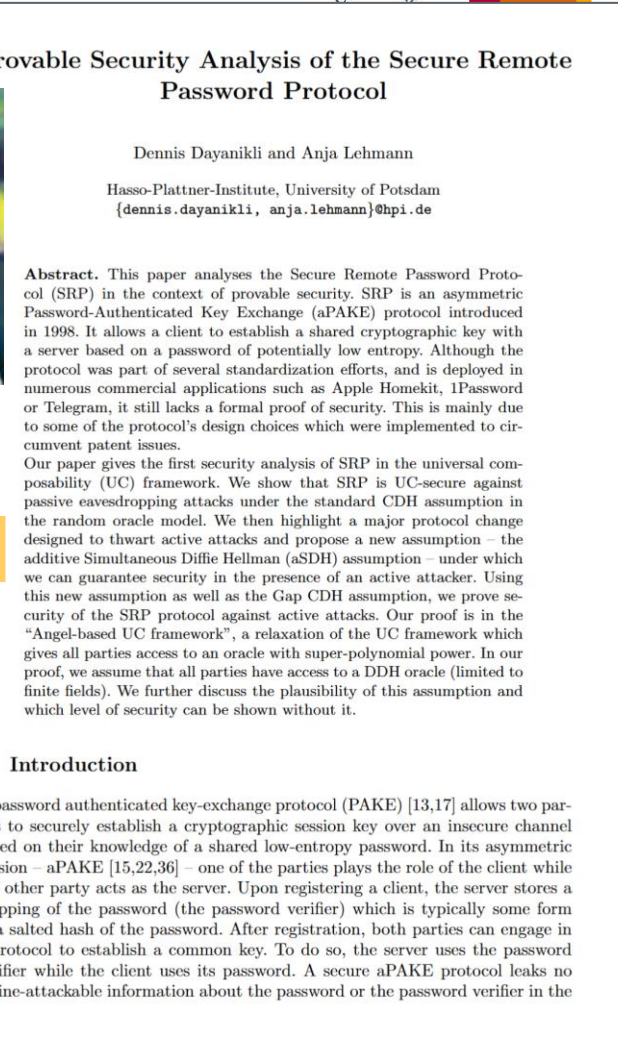
Hasso Plattner Institute | University of Potsdam

Conference Collision

- First security proof of SRP (v6a)
  Most widely deployed aPAKE protocol
  Apple Homekit, 1Password, Telegram, AWS
- Weird design from 1998 (to circumvent patents)
  complicated security analysis
- Security proof in state-of-the-art aPAKE model

This talk: current aPAKE model is too weak

37th IEEE Computer Security Foundations Symposium
July 8-12, 2024 - Enschede, The Netherlands



Provable Security Analysis of the Secure Remote Password Protocol

Dennis Dayanikli and Anja Lehmann

Hasso-Plattner-Institute, University of Potsdam
{dennis.dayanikli, anja.lehmann}@hpi.de

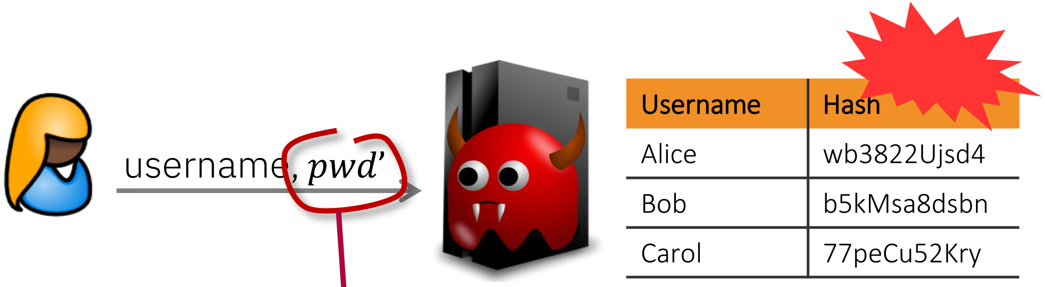
Abstract. This paper analyses the Secure Remote Password Protocol (SRP) in the context of provable security. SRP is an asymmetric Password-Authenticated Key Exchange (aPAKE) protocol introduced in 1998. It allows a client to establish a shared cryptographic key with a server based on a password of potentially low entropy. Although the protocol was part of several standardization efforts, and is deployed in numerous commercial applications such as Apple Homekit, 1Password or Telegram, it still lacks a formal proof of security. This is mainly due to some of the protocol's design choices which were implemented to circumvent patent issues.
Our paper gives the first security analysis of SRP in the universal composability (UC) framework. We show that SRP is UC-secure against passive eavesdropping attacks under the standard CDH assumption in the random oracle model. We then highlight a major protocol change designed to thwart active attacks and propose a new assumption – the additive Simultaneous Diffie Hellman (aSDH) assumption – under which we can guarantee security in the presence of an active attacker. Using this new assumption as well as the Gap CDH assumption, we prove security of the SRP protocol against active attacks. Our proof is in the "Angel-based UC framework", a relaxation of the UC framework which gives all parties access to an oracle with super-polynomial power. In our proof, we assume that all parties have access to a DDH oracle (limited to finite fields). We further discuss the plausibility of this assumption and which level of security can be shown without it.

1 Introduction

A password authenticated key-exchange protocol (PAKE) [13,17] allows two parties to securely establish a cryptographic session key over an insecure channel based on their knowledge of a shared low-entropy password. In its asymmetric version – aPAKE [15,22,36] – one of the parties plays the role of the client while the other party acts as the server. Upon registering a client, the server stores a mapping of the password (the password verifier) which is typically some form of a salted hash of the password. After registration, both parties can engage in a protocol to establish a common key. To do so, the server uses the password verifier while the client uses its password. A secure aPAKE protocol leaks no offline-attackable information about the password or the password verifier in the

Password-based Authentication (Status Quo)

- Passwords still predominant form of user authentication – convenient! No key needed



Username	Hash
Alice	wb3822Ujsd4
Bob	b5kMsa8dsbn
Carol	77peCu52Kry

stores only (salted) password hashes h

checks that $h_{Alice} == H(s, pwd')$?

- Storing hashes only, provides (some) protection in case of server compromise
- But: Passwords are send **in clear** to the server (via TLS) at every login!

LinkedIn, Twitter, Github lost millions of plaintext passwords due to accidental logging

85% of users re-use their passwords: a single bad server/ phishing attack is fatal

Ideally: password-based authentication without the need for plaintext passwords ...

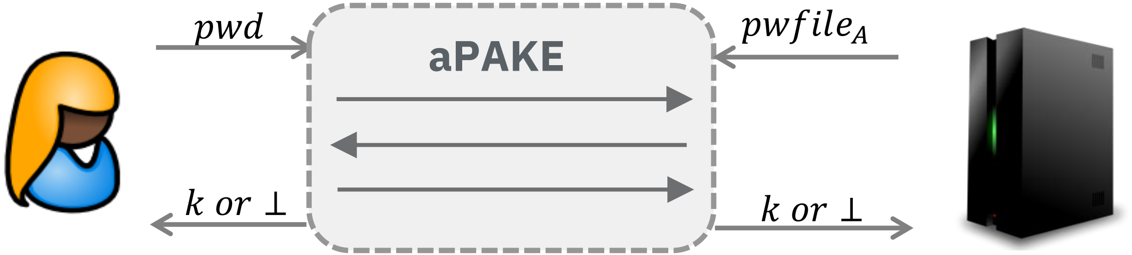
(Strong) Asymmetric PAKE

- Asymmetric Password Authenticated Key Exchange (aPAKE), invented in the 90s
- Enables secure pwd-based authentication between client and server

Main feature: client doesn't reveal pwd **during login**

or any offline-attackable information thereof

important as passwords have low entropy



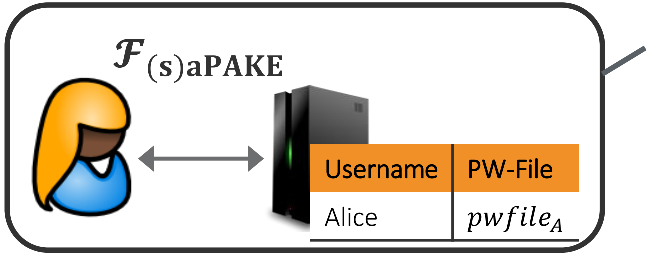
Username	PW-File
Alice	$pwfile_A$
Bob	$pwfile_B$
Carol	$pwfile_C$

If pwd correct, create fresh session key k, else return $\perp$

- Offline attacks only possible after compromise of pwd-file, precomputation is possible
- Strong aPAKE: no precomputation attacks before server compromise

OPAQUE: First Strong aPAKE [JKX'18]

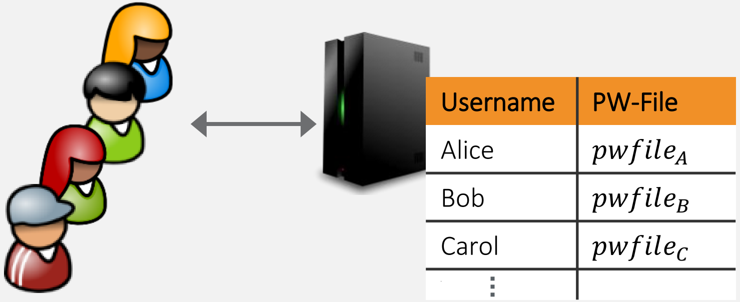
- Won the IETF (a)PAKE competition in 2020, currently undergoing standardization by IRTF
- Provably secure in Universal Composability framework … but only for a *single* user



Username	PW-File
Alice	$pwfile_A$

Ideal functionality $\mathcal{F}_{\text{(s)aPAKE}}$ assumes that server only manages a single user

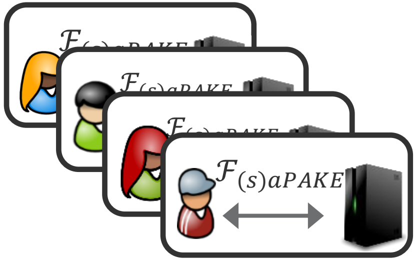
Real world: one server & *many* users



Username	PW-File
Alice	$pwfile_A$
Bob	$pwfile_B$
Carol	$pwfile_C$
⋮	

- Single-user setting simplifies analysis
- Multi-user version follows from UC Framework

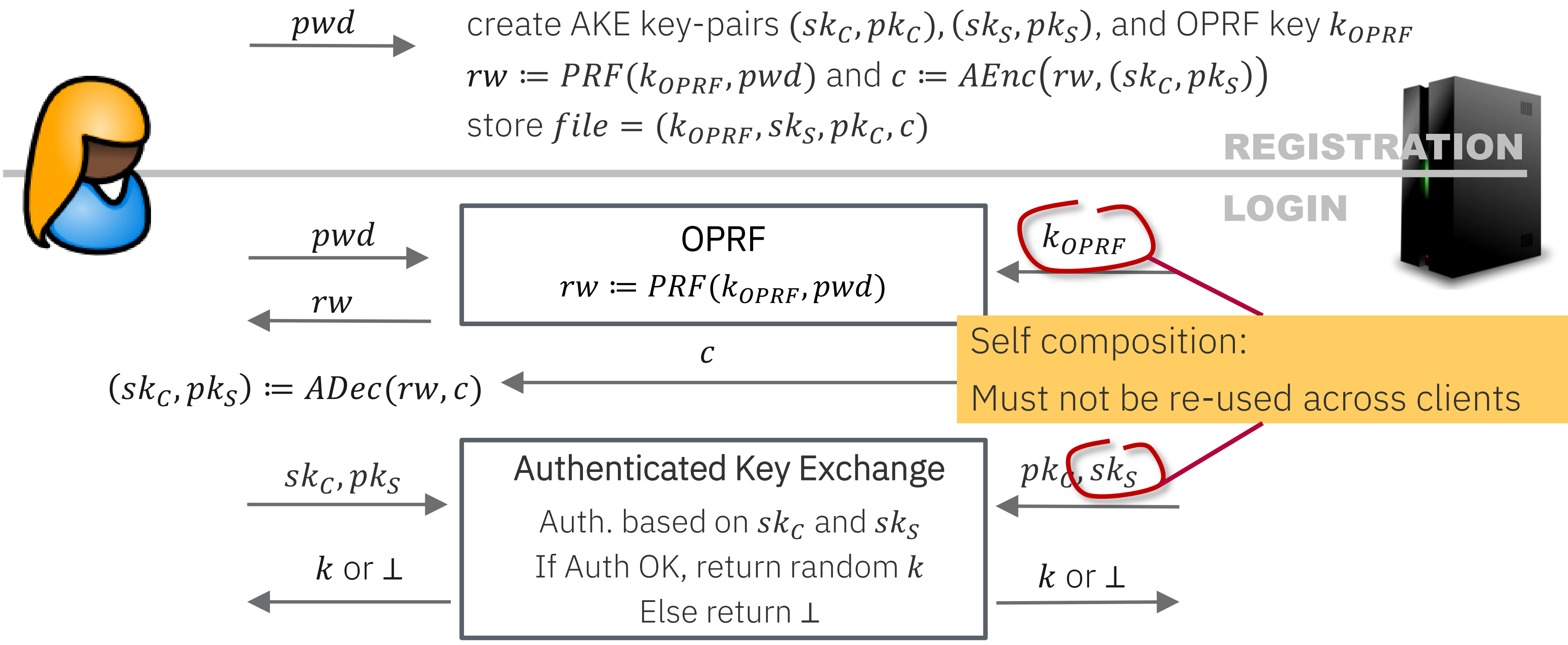
→ Protocol is secure under self-composition


$\mathcal{F}_{\text{(s)aPAKE}}$
$\mathcal{F}_{\text{(s)aPAKE}}$
$\mathcal{F}_{\text{(s)aPAKE}}$
$\mathcal{F}_{(s)aPAKE}$

But: only if all protocol copies are independent

→ server is not allowed to keep state across instances

OPAQUE | Single-User Protocol (Core Idea)



pwd

create AKE key-pairs $(sk_C, pk_C), (sk_S, pk_S)$, and OPRF key k_{OPRF}

$rw \coloneqq PRF(k_{OPRF}, pwd)$ and $c \coloneqq AEnc\big(rw, (sk_C, pk_S)\big)$

store $file = (k_{OPRF}, sk_S, pk_C, c)$

REGISTRATION

LOGIN

pwd

rw

OPRF

$rw \coloneqq PRF(k_{OPRF}, pwd)$

k_{OPRF}

c

$(sk_C, pk_S) \coloneqq ADec(rw, c)$

Self composition:

Must not be re-used across clients

sk_C, pk_S

k or $\perp$

Authenticated Key Exchange

Auth. based on sk_C and sk_S

If Auth OK, return random k

Else return $\perp$

pk_C, sk_S

k or $\perp$

Draft-OPAQUE | Multi-User Protocol for IRFT Standard

SETUP

Global Server Setup:

create static (=same for all users) AKE−key pair (sk_S, pk_S) and $seed_{OPRF}$

pwd

create AKE key-pairs $(sk_C, pk_C), (sk_S, pk_S)$, $k_{OPRF} := F(seed_{OPRF}, uid)$

$rw := PRF(k_{OPRF}, pwd)$ and $c := AEnc(rw, (sk_C, pk_S))$

store $file = (seed_{OPRF}, sk_S, pk_C, c)$

REGISTRATION

LOGIN

$k_{OPRF} := F(seed_{OPRF}, uid)$

$d)$

hange

sk_S

om k

Else return ⊥

pk_C, sk_S (long-term AKE key)

k or ⊥

IRFT: **“The oprf_seed value SHOULD be used for all clients”** (to prevent client enumeration attacks)

→ UC self composition no longer applies!

→ Is this OPAQUE version secure?

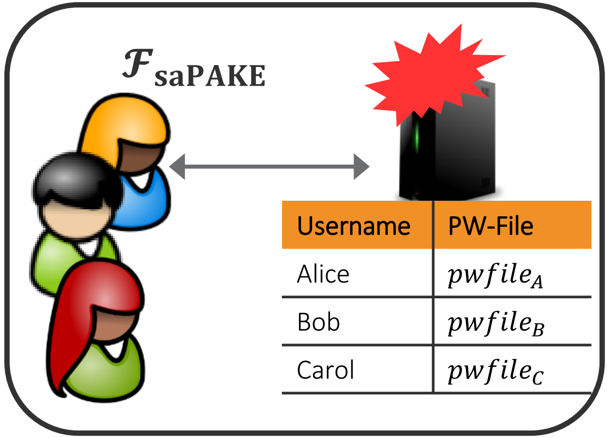
Our work:

1) New saPAKE security model for multi-user setting
2) Security analysis of Draft-OPAQUE in new model

(shared state modificatio only)

Security of Multi-User saPAKE

- Security model mostly straightforward extension of single-user functionality:



Username	PW-File
Alice	$pwfile_A$
Bob	$pwfile_B$
Carol	$pwfile_C$

No offline attacks on user's password during login

Precomputation & offline attacks *after* server compromise

New: impact of *partial* compromise across users:

Username	PW-File
Alice	$pwfile_A$
Bob	$pwfile_B$

Offline attacks on Alice's and Bob's pwd possible
... but not for Carol

No offline or precomputation attacks on uncompromised files

- Draft-OPAQUE: $file = (seed_{OPRF}, sk_S, pk_C, c)$ compromise leaks shared keys $seed_{OPRF}$ and sk_S
  - ✗ □ Re-use of $seed_{OPRF}$ allows to offline attacks all accounts after single file leak

    Server sends $c := AEnc(rw, (sk_C, pk_S))$ – Adv can compute rw^* via $seed_{OPRF}$ & test if $ADec(rw^*, c) \neq \perp$?
  - ✓ □ Re-use of sk_S does not harm security → security proof (assuming per-client k_{OPRF})

Security of Multi-User (strong) aPAKE

- Take away OPAQUE and strong aPAKE

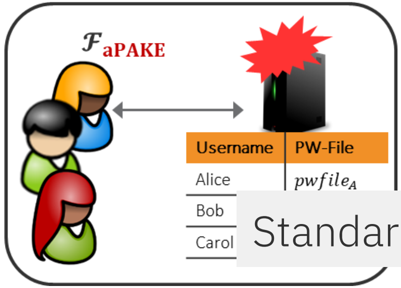
  Key-reuse *can* be secure (has advantages for server)

  Security model & protocol should already be for real-world setting

  (otherwise dangerous gap between proven vs. real-world protocol)

This was about strong aPAKE → what about standard one?

- Multi-user aPAKE model (and secure protocols therein) exist:



Username	PW-File
Alice	$pwfile_A$
Bob	
Carol	

No offline attacks on users password during login

~~Precomputation &~~ offline attacks only *after* file compromise

Standard aPAKE does not provide *any* security against precomputation attacks

- Here there is another gap between provable security guarantees and actual protocols:

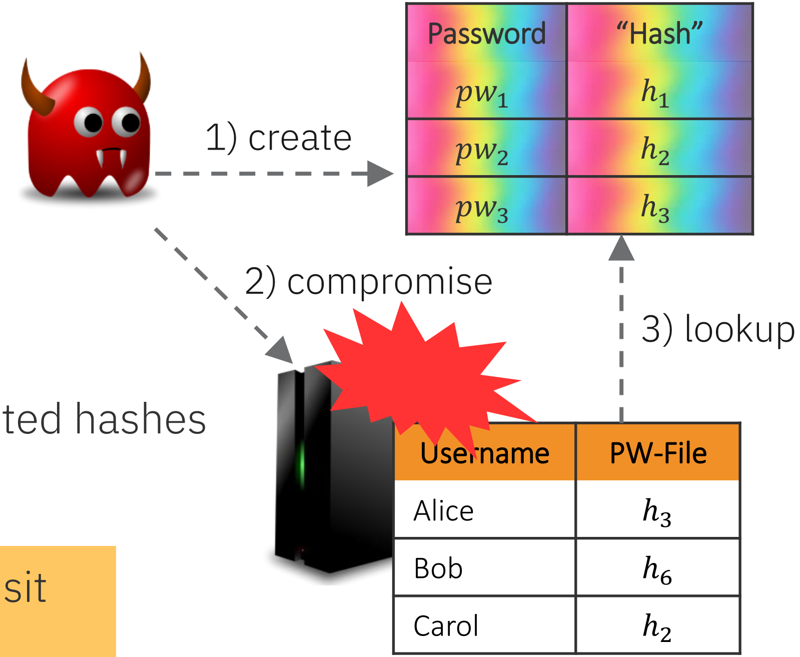
  Several protocols (such as SRP) are actually more secure than what is modelled!

aPAKE → No Security against Precomputation Attacks

- aPAKE Model

  Precomputation attacks possible at all time:
  - Adversary can create pwd-rainbow table
  - Lookup pwd after server is compromised
  - Equivalent to unsalted hashes

- Precomputation attacks are not possible with salted hashes in standard plaintext-pwd authentication!



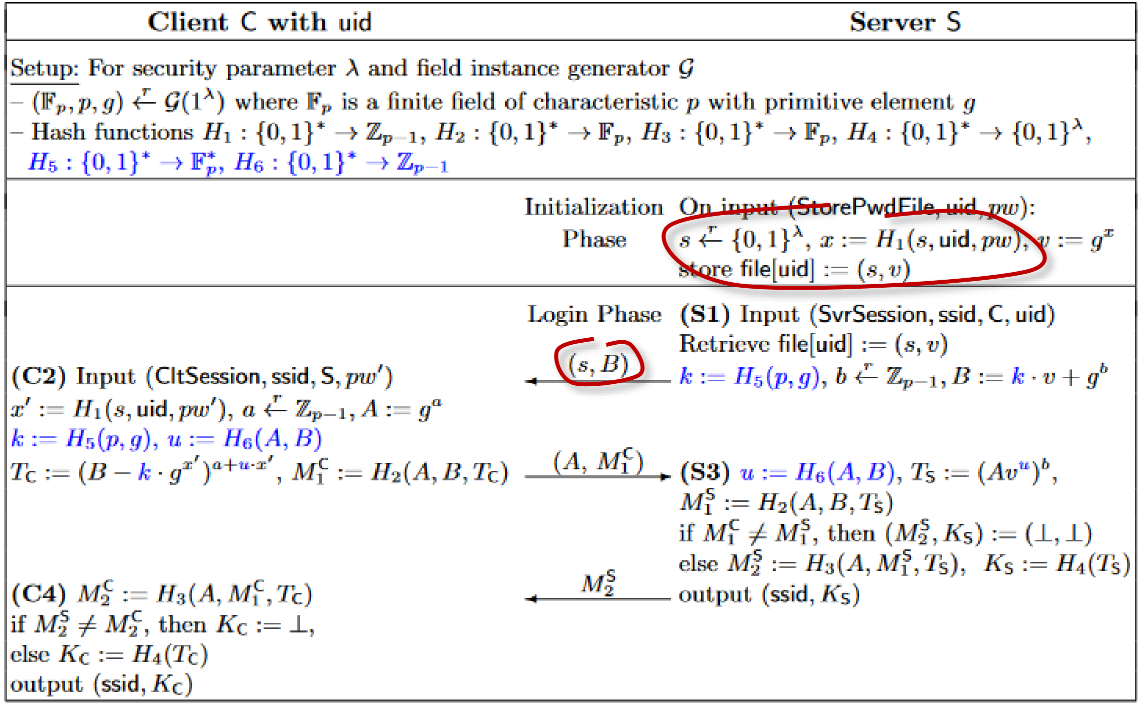
Password	"Hash"
pw_1	h_1
pw_2	h_2
pw_3	h_3

Username	PW-File
Alice	h_3
Bob	h_6
Carol	h_2

aPAKE guarantees strong security of passwords in transit but decreases security for stored passwords

...but some aPAKE protocols do have salt

Salting in aPAKE Protocols



Client C with uid | **Server S**

Setup: For security parameter λ and field instance generator $\mathcal{G}$

- $(\mathbb{F}_p, p, g) \xleftarrow{r} \mathcal{G}(1^\lambda)$ where $\mathbb{F}_p$ is a finite field of characteristic p with primitive element g
- Hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_{p-1}$, $H_2 : \{0,1\}^* \to \mathbb{F}_p$, $H_3 : \{0,1\}^* \to \mathbb{F}_p$, $H_4 : \{0,1\}^* \to \{0,1\}^\lambda$, $H_5 : \{0,1\}^* \to \mathbb{F}_p^*$, $H_6 : \{0,1\}^* \to \mathbb{Z}_{p-1}$

Initialization Phase

On input $(\mathsf{StorePwdFile}, \mathsf{uid}, pw)$:
$s \xleftarrow{r} \{0,1\}^\lambda$, $x := H_1(s, \mathsf{uid}, pw)$, $v := g^x$
store $\mathsf{file}[\mathsf{uid}] := (s, v)$

Login Phase

(S1) Input $(\mathsf{SvrSession}, \mathsf{ssid}, \mathsf{C}, \mathsf{uid})$
Retrieve $\mathsf{file}[\mathsf{uid}] := (s, v)$
$k := H_5(p, g)$, $b \xleftarrow{r} \mathbb{Z}_{p-1}$, $B := k \cdot v + g^b$

$\xleftarrow{(s, B)}$

(C2) Input $(\mathsf{CltSession}, \mathsf{ssid}, \mathsf{S}, pw')$
$x' := H_1(s, \mathsf{uid}, pw')$, $a \xleftarrow{r} \mathbb{Z}_{p-1}$, $A := g^a$
$k := H_5(p, g)$, $u := H_6(A, B)$
$T_\mathsf{C} := (B - k \cdot g^{x'})^{a + u \cdot x'}$, $M_1^\mathsf{C} := H_2(A, B, T_\mathsf{C})$

$\xrightarrow{(A,\ M_1^\mathsf{C})}$

(S3) $u := H_6(A, B)$, $T_\mathsf{S} := (Av^u)^b$,
$M_1^\mathsf{S} := H_2(A, B, T_\mathsf{S})$
if $M_1^\mathsf{C} \neq M_1^\mathsf{S}$, then $(M_2^\mathsf{S}, K_\mathsf{S}) := (\perp, \perp)$
else $M_2^\mathsf{S} := H_3(A, M_1^\mathsf{S}, T_\mathsf{S})$, $K_\mathsf{S} := H_4(T_\mathsf{S})$
output $(\mathsf{ssid}, K_\mathsf{S})$

$\xleftarrow{M_2^\mathsf{S}}$

(C4) $M_2^\mathsf{C} := H_3(A, M_1^\mathsf{C}, T_\mathsf{C})$
if $M_2^\mathsf{S} \neq M_2^\mathsf{C}$, then $K_\mathsf{C} := \perp$,
else $K_\mathsf{C} := H_4(T_\mathsf{C})$
output $(\mathsf{ssid}, K_\mathsf{C})$

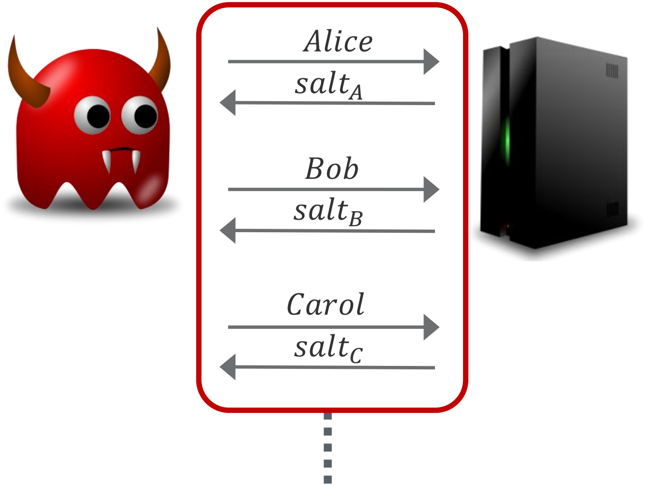
- Many existing UC-secure aPAKEs (e.g., SRP, OKAPE, AuCPace) use salting techniques
- Precomputation attacks impossible without knowing the salt
- But: salt is sent in clear to client during login

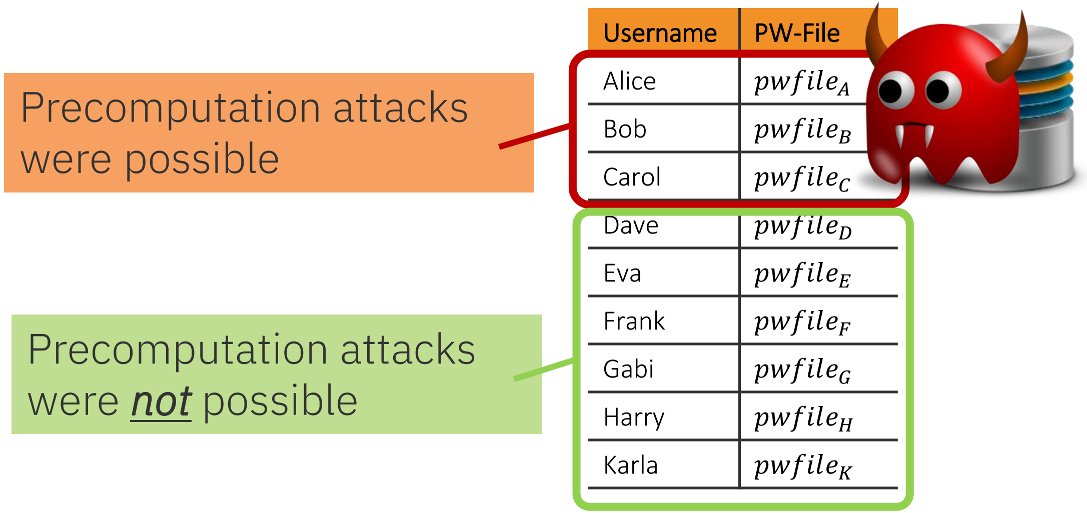
Benefits of „<u>Public</u>“ Salting?

- How can a salt help when it is revealed *before* authentication?

  Added security only becomes clear in multi-user setting

  Individual salt capture „attack“ vs. bulk compromise:





Precomputation attacks were possible

Precomputation attacks were ***not*** possible

Username	PW-File
Alice	$pwfile_A$
Bob	$pwfile_B$
Carol	$pwfile_C$
Dave	$pwfile_D$
Eva	$pwfile_E$
Frank	$pwfile_F$
Gabi	$pwfile_G$
Harry	$pwfile_H$
Karla	$pwfile_K$

Adversary can get salt, but has to start session for every user individually

Server compromise reveals bulk of user files
→ one attack often leaks millions of files

Strengthening Security of aPAKE

- Our Work: new model(s) to reflect the stronger security of protocols:
  - □ Precomputation attack possible only after Adv has initiated session for uid
  - □ When Adv compromises file for uid, but never initiated session before:

    → Same security for user as strong aPAKE

before compromise of file for user uid	aPAKE	New Models	Strong aPAKE
no offline attacks	🟩	🟩	🟩
no precomputation attacks	🟧	🟧🟩	🟩

- Several aPAKE protocols (almost) satisfy stronger model
  - □ Not fully – due to UC subtleties (assumes global network eavesdropper)
  - □ Simple transformation (encrypt salt under fresh pk) to yield stronger aPAKE security

Summary

- (Strong) aPAKE security model for multi-user security needed to:
  - □ Design optimal & secure protocol for real-world setting

    Server key-reuse has advantages, but not every key can be re-used → 🚫 OPRF-seed in OPAQUE
  - □ Understand cross-user impact of partial compromise

- Many aPAKE protocols provider better security than advertised
  - □ Many protocols have "public salt" → helps against precomputation attacks
  - □ Stronger security model(s) & transformations

Thank You!

Full paper at: https://eprint.iacr.org/2024/756