Crypto 14

Note: this material is not intended to replace the live lecture for students.

Contents

14.1 Simultaneous Authentication of Equals (SAE)

NOTE 14.1.1

Wi-Fi point access, WPA3 uses SAE.

As in the previous protocols two users share a password $\mathsf{P} = \textit{password}$ from which SAE is going to produce secure session keys.
SAE is peer-to-peer and Not Lock-step: no initiator or responder, nor client-server.

14.1.2 SAE parameters

a finite cyclic group Γ, e.g. DH or ECC, with r elements.

an ordering function L taking two users and returning the "greater".

a random oracle equally probable, one-way function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^s$.

a KDF that stretch an arbitrary string *smallstr* to a string *bigstr* of a given *length* :

$$bigstr = \mathsf{KDF}(smallstr, length)$$

a bijective function F from Γ to a set of numbers.

The protocol begins upon discovering of a peer.

Prior to sending any messages **Alice** and **Bob** select an element PWE, password element, from the group Γ.

14.1.3 PWE: SAE-ECC

$i = 0$
repeat
  $i = i + 1$
  if $L(Alice, Bob) = Alice$ **then**
    $pwdseed = H(Alice \mid Bob \mid password \mid i)$
  else
    $pwdseed = H(Bob \mid Alice \mid password \mid i)$
  end if
  $x = (KDF(pwdseed,\ len)) \bmod p$
  solve for y using the equation for the curve with x
  if $pwdseed$ is odd **then**
    $PWE = (x, -y)$
  else
    $PWE = (x, y)$
  end if
until PWE is on the curve

where p is the prime of the curve and len is the length of p.

14.1.4 PWE: SAE-DH

if $L(Alice, Bob) = Alice$ **then**
  $pwdseed = H(Alice \mid Bob \mid password)$
else
  $pwdseed = H(Bob \mid Alice \mid password)$
end if
$pwdvalue = (KDF(pwdseed,\ len)) \bmod p$
$PWE = pwdvalue^{((p-1)/r)} \bmod p$

where p is the group prime, r is the order, and len is the length of p.

14.1.5 SAE

Alice		**Bob**
generate a RND		generate a RND
$rand_A, mask_A$		$rand_B, mask_B$
$scal_A = (rand_A + mask_A) \pmod r$		$scal_B = (rand_B + mask_B) \pmod r$
$elem_A = (mask_A \cdot PWE)^{-1}$		$elem_B = (mask_B \cdot PWE)^{-1}$
	$\xrightarrow{scal_A, elem_A}$	
	$\xleftarrow{scal_B, elem_B}$	
$K = rand_A \cdot (scal_B \cdot PWE \cdot elem_B)$		$K = rand_B \cdot (scal_A \cdot PWE \cdot elem_A)$
$\mathsf{k} = \mathsf{F}(K)$		$\mathsf{k} = \mathsf{F}(K)$
$tok_A = \mathsf{H}(\mathsf{k}\|\|\mathsf{F}(elem_A)\|\|scal_A\|\|\mathsf{F}(elem_B)\|\|scal_B)$		
	$\xrightarrow{tok_A}$	
		$tok_B = \mathsf{H}(\mathsf{k}\|\|\mathsf{F}(elem_B)\|\|scal_B\|\|\mathsf{F}(elem_A)\|\|scal_A)$
	$\xleftarrow{tok_B}$	
token verification		token verification
$\mathsf{R} = \mathsf{H}(\mathsf{k}\|\|\mathsf{F}(elem_A \cdot elem_B)\|\|(scal_A + scal_B) \pmod r)$		$\mathsf{R} = idem$

14.2 Secure Remote Password (SRP)

SRP is a client-server authentication protocol.

NOTE 14.2.1

Authentication protocols are of two types:

plaintext-equivalent = requires the server to store a copy of P or something from which P is computationally feasible to obtain.

verifier-based = requires the server to store a V or something from which P is computationally infeasible to obtain. But from which P can be computationally verified.

SRP is verified-based which reduce the damage that a Trojan can inflict.

The client, to be regarded as a human, as the P stored in his brain.

The server store a *verifier* V that allows him to check P yet it is computationally infeasible to get P from the verifier V.

14.2.1 Asymmetric key exchange (AKE)

Alice is going to be the client.

AKE: parameters and primitives

A Alice's password.

S Server's password.

a one-way function $\mathsf{P}(x)$.

$\mathsf{Q}(x, y), \mathsf{R}(x, y)$ mixing functions.

$\mathsf{S}(x, y)$ the session key K generation function.

All this must satisfies for all $w, y, \mathsf{S}, \mathsf{A}$:

$$\mathsf{S}(\mathsf{R}(\mathsf{P}(w), \mathsf{P}(\mathsf{A})), \mathsf{Q}(y, \mathsf{S})) = \mathsf{S}(\mathsf{R}(\mathsf{P}(y), \mathsf{P}(\mathsf{S})), \mathsf{Q}(w, \mathsf{A}))$$

At the set up **Alice** computes $\mathsf{P}(\mathsf{A})$ and give it to the **Server**, and the **Server** computes $\mathsf{P}(\mathsf{S})$ and gives it to **Alice**.

14.2.2 AKE Session Key

Alice		Server
generate a RND w		
	$\xrightarrow{\mathsf{P}(w)}$	
		generate a RND y $\mathsf{K} = \mathsf{S}(\mathsf{R}(\mathsf{P}(w), \mathsf{P}(\mathsf{A})), \mathsf{Q}(y, \mathsf{S}))$
	$\xleftarrow{\mathsf{P}(y)}$	
$\mathsf{K} = \mathsf{S}(\mathsf{R}(\mathsf{P}(y), \mathsf{P}(\mathsf{S})), \mathsf{Q}(w, \mathsf{A}))$		

NOTE 14.2.3

To complete AKE authentication process **Alice** and **Server** can use a mutually agreeable method of key verification e.g. Challenge-Response see Crypto13 **Validation of Keys**.

Notice that AKE is Zero-knowledge password.

Instead EKE protocol use prearranged shared secret. More precisely, both parties keep exactly the same secret P.

14.2.2 SRP

14.2.4 SRP specifications

Computations performed in the Galois finite field GF(p), p a large prime number. That is to say, the arguments and values of $\mathsf{P}, \mathsf{Q}, \mathsf{R}, \mathsf{S}$ are integers between 0 and $p-1$.

Here P:

$$\mathsf{P}(x) = g^x$$

where g is a generator of GF(p).

Here the functions $\mathsf{Q}, \mathsf{R}, \mathsf{S}$:

$$\begin{cases} \mathsf{Q}(w,x) = w + u \cdot x \\ \mathsf{R}(w,x) = w \cdot x^u(w,x) \\ \mathsf{S}(w,x) = w^x \end{cases}$$

where $u(w,x)$ is a function to be explained later.

At set-up **Alice** generate a random salt s, his long term password A and compute $x = \mathsf{H}(s||\mathsf{A})$. Then she gives to the **Server in a secure way** s and $v = g^x$. The **Server** stores the verifier v and the salt s.

Exercise 14.2.5

Verify that:

$$\mathsf{S}(\mathsf{R}(\mathsf{P}(w), \mathsf{P}(\mathsf{A})), \mathsf{Q}(y, \mathsf{S})) = \mathsf{S}(\mathsf{R}(\mathsf{P}(y), \mathsf{P}(\mathsf{S})), \mathsf{Q}(w, \mathsf{A}))$$

NOTE 14.2.6

SRP parameters domain: the prime number p, the generator g and also a **Hash** function H.

14.2.7 SRP protocol

Alice		Server
	$\xrightarrow{\text{Alice}}$	
		looks for **Alice**'s s, v
	$\xleftarrow{s}$	
check $x = \textbf{Hash}(s, \mathsf{A})$ generate RND a		
	$\xrightarrow{g^{\mathsf{a}}}$	
		generate RND b, u $B = v + g^{\mathsf{b}}$ $S = (g^{\mathsf{a}} \cdot v^u)^{\mathsf{b}}$ $K = \textbf{Hash}(S)$
	$\xleftarrow{B,u}$	
$S = (B - g^x)^{\mathsf{a}+u\cdot x}$ $K = \textbf{Hash}(S)$ $M_1 = \textbf{Hash}(g^{\mathsf{a}}\|\|B\|\|K)$		
	$\xrightarrow{M_1}$	
		verify M_1 $M_2 = \textbf{Hash}(g^{\mathsf{a}}\|\|M_1\|\|K)$
	$\xleftarrow{M_2}$	
verify M_2		

Exercise 14.2.8

Explain:

1) Why just not make $B = g^{\mathsf{b}}$, instead of the sum of two exponentials $B = v + g^{\mathsf{b}}$, and simplify the protocol?

2) Partition attack: Why not make $B = v \oplus g^{\mathsf{b}}$?

3) How an intruder can gain access to the **Server** knowing 1) **Alice**'s verifier v and 2) how the **Server** generate u. Hint: the intruder send $g^{\mathsf{a}} \cdot v^{-u}$ to **Server**. In particular, this shows why the **Server** reveals u **after** he receives **Alice**'s g^{a}.

14.3 Bibliography

Books I used to prepare this note:

[Paar10] Paar, Christof, Pelzl, Jan, *Understanding Cryptography, A Textbook for Students and Practitioners*, Springer-Verlag, 2010.

Papers I used to prepare this note:

[BeMe92] S. Bellovin and M. Merritt; *Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks*, Proc of the Symposium on Security and Privacy, pages 72-84, IEEE, 1992 https://www.cs.columbia.edu/~smb/papers/neke.pdf

[Ha08] Harkins, Dan; *Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks*, 2008 Second International Conference on Sensor Technologies and Applications. IEEE, DOI 10.1109/SENSORCOMM.2008.131 https://ieeexplore.ieee.org/document/4622764

[Ha08] Harkins, Dan; *Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks*, 2008 Second International Conference on Sensor Technologies and Applications. IEEE, DOI 10.1109/SENSORCOMM.2008.131 https://pdfs.semanticscholar.org/cac2/887626aa79bc2c30ff09330e0920cadb24db.pdf

[Wu97] Wu, Thomas; *The Secure Remote Password Protocol*, 1998 Internet Society Symposium on Network and Distributed System Security. http://www.scs.stanford.edu/nyu/02sp/sched/srp.pdf

and some interesting links:

https://www.youtube.com/watch?v=iaH8UG2yMg4

https://en.wikipedia.org/wiki/Kerberos_(protocol)

https://en.wikipedia.org/wiki/Security_protocol_notation

https://en.wikipedia.org/wiki/Simultaneous_Authentication_of_Equals

http://homes.di.unimi.it/visconti/PBKDF2.pdf

https://en.wikipedia.org/wiki/Needham%E2%80%93Schroeder_protocol

http://www.cs.unc.edu/~fabian/course_papers/needham.pdf

http://pages.cs.wisc.edu/~remzi/Classes/736/Spring2005/Papers/data-encryption-denning.pdf

http://srp.stanford.edu/