Security Weaknesses of password-only authenticated key establishment protocol without public key cryptography

Mohsen Toorani
Department of Informatics
University of Bergen
Bergen, Norway
mohsen.toorani@ii.uib.no

Maryam Saeed
E-Learning Center
Iran University of Science and Technology
Tehran, Iran
m_saeed@vu.iust.ac.ir

Abstract

In 2005, Laih et al. proposed a password-based authentication key exchange protocol that is not based on public key cryptography but uses human ability to extract strings from distorted images. In this letter, it is shown that Laih et al.'s protocol is vulnerable to password compromise impersonation, malicious server, offline password guessing, undetectable online password guessing, stolen-verifier, and Unknown Key-Share (UKS) attacks and it does not provide forward secrecy and key confirmation.

Keywords: Cryptographic protocols, PAKE, CAPTCHA, Security analysis, Attacks.

1. Introduction

Password-Authenticated Key Exchange (PAKE) protocols enable entities to authenticate each other and share a cryptographic session key based on a low-entropy human memorable password. Before introduction of the first PAKE protocol in 1992 [1], it was hard to believe that a low-entropy password can be used for establishing a strong cryptographic key in such a way that is resistant to a dictionary attack from a list of most probable passwords. However, after introduction of the first PAKE protocol [1], many PAKE protocols were proposed but some of them were proved to have security problems.

This letter considers security analysis of Laih et al.'s protocol [2] that for the case of brevity will be referred to as the LDH protocol throughout this letter. In [3], it is proved that the LDH protocol is

vulnerable to offline dictionary attack. In [4], it is shown that the LDH protocol does not provide perfect forward secrecy. Security improvement of the LDH protocol is also considered in [5, 6] but in this letter, some new security problems are reported on the LDH protocol. This includes its vulnerability to password compromise impersonation, malicious server, undetectable online password guessing, stolen-verifier, and Unknown Key-Share (UKS) attacks.

2. A brief review on LDH protocol

The LDH protocol [2] is depicted in Fig. 1. It is a PAKE protocol that provides mutual authentication through three rounds of communication between user *U* and server *S* that have shared password *pw* in advance. Steps for the LDH protocol can be followed as:

(1) *U* selects a random number *t* and sends it to *S* together with its own identifier ID_U.

(2) *S* selects a random number *s* and a random string $r \in_R B_n$ in which B_n is a set of characters or digits of length *n*, and calculates $C_1 = E_{pw}(\varphi(r,s))$ and $C_2 = H(pw,r,t)$ where $E_{pw}(.)$ is a symmetric encryption algorithm with secret key *pw*, $\varphi(r,s)$ outputs a distorted random picture/audio based on *r* and *s*, and $H(.)$ is a collision-resistant one-way hash function. φ is so that people have ability to recognize *r* from the distorted picture/audio but a machine cannot. The CAPTCHA is an example of φ. *S* sends C_1 and C_2 to *U*.

(3) *U* uses its password *pw* as the key for decrypting C_1 as $I = D_{pw}(C_1)$. If *I* does not contain a meaningful string r', *U* halts the protocol run. Otherwise, it uses r' for calculating $H(pw,r',t)$ and checks if $C_2 = H(pw,r',t)$ or not. If C_2 is not verified, it stops the protocol run. Otherwise, *U* calculates $C_3 = H(1, pw, r', t)$ and session key $SK_U = H(2, pw, r', t)$, and sends C_3 to *S*.

(4) *S* verifies if $C_3 = H(1, pw, r, t)$ or not. If the equality is satisfied, *S* generates the session key as $SK_S = H(2, pw, r, t)$. Otherwise, the authentication fails.

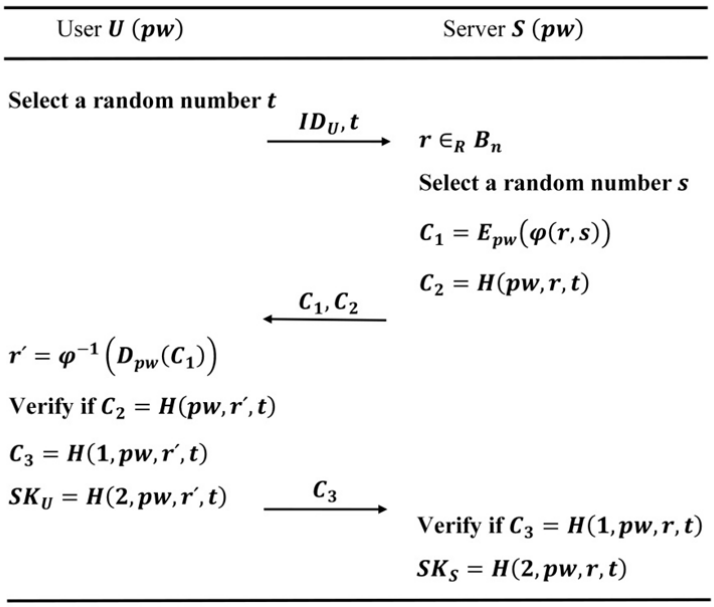


Fig. 1. The LDH protocol

3. Security Problems of the LDH protocol

In this section, we show that the LDH protocol not only is vulnerable to password compromise impersonation, malicious server, offline and undetectable online password guessing, stolen-verifier, and Unknown Key-Share (UKS) attacks but it does not provide forward secrecy and key confirmation.

3.1. Vulnerability to password compromise impersonation attack

When the password of user *U* is compromised, the adversary *E* can impersonate the legitimate server *S* and share a session key with *U*. This is a password compromise impersonation attack and its steps can be followed as:

(1) *E* intercepts the connection between *U* and *S*, and receives *t* and ID_U from *U*.

(2) *E* selects string *r* and number *s*, computes $C_1 = E_{pw}(\varphi(r,s))$ and $C_2 = H(pw,r,t)$, and sends C_1 and C_2 to user *U*.

(3) *U* decrypts C_1 using password *pw* to obtain $r' = \varphi^{-1}(D_{pw}(C_1))$. *U* verifies C_2 as $C_2 = H(pw,r',t)$, calculates $C_3 = H(1, pw, r', t)$ and $SK_U = H(2, pw, r', t)$, and sends C_3 to *E*.

(4) *E* computes the session key as $SK_S = H(2, pw, r, t)$.

3.2. Vulnerability to malicious server attack

In the malicious server attack, an adversary that runs on a malicious server, tempts people to register with that server and uses acquired passwords from users to masquerade as them in login to legitimate servers. The LDH protocol is vulnerable to this attack because the server stores users' passwords in clear. A malicious server $\tilde{S}$ can then use passwords for impersonating users to another benign server *S* and sharing a session key with *S*. Steps for this attack are as follows:

(1) $\tilde{S}$ starts the protocol run with a legitimate server *S*. $\tilde{S}$ generates number *t* and sends *S* the user *U*'s identifier ID_U together with *t*.

(2) *S* selects random string *r* and random number *s*, calculates $C_1 = E_{pw}(\varphi(r, s))$ and $C_2 = H(pw, r, t)$, and sends C_1 and C_2 to $\tilde{S}$.

(3) $\tilde{S}$ obtains r' as $r' = \varphi^{-1}(D_{pw}(C_1))$, computes $C_3 = H(1, pw, r', t)$ and $SK_U = H(2, pw, r', t)$, and sends C_3 to *S*.

(4) *S* verifies C_3 as $C_3 = H(1, pw, r, t)$ and generates the session key as $SK_S = H(2, pw, r, t)$.

3.3. Vulnerability to offline/ undetectable online password guessing attack

The LDH protocol is vulnerable to both undetectable online and offline password guessing attacks:

- For an undetectable online attack, the adversary *E* generates number *t* and sends it together with identifier of the designated user ID_U to the corresponding server *S*. After receiving C_1 and C_2 from *S*, *E* halts the communication and proceeds with trying different possible passwords pw' to compute $r' = \varphi^{-1}(D_{pw'}(C_1))$. *E* then checks if $C_2 = H(pw', r', t)$ or not. If the equality holds, the guessed password pw' is the correct password of the designated user. Otherwise, *E* tries other passwords.
- Steps are the same for an offline password guessing attack except that at beginning of the attack, the attacker just eavesdrops the communication between *U* and *S* and uses the eavesdropped values of *t*, C_1 and C_2 for checking the guessed password. Another scenario is also presented in [3] and [5] for an offline dictionary attack that is based on the fact that when $pw' \neq pw$, $D_{pw'}(C_1)$

resembles a random image. Then, hypothetical software may be used for checking if $D_{pw'}(C_1)$ is a distorted image or a random pattern. If such software is available, the required computations can be greatly decreased. However, this attack is skeptical because when the exploited CAPTCHA is really based on a hard Artificial Intelligence (AI) problem as proposed in [7], such problem does not exist and the attack described in [3] and [5] will not work but it does not have any influence on undetectable online/offline password guessing attack described in this paper.

3.4. Vulnerability to stolen-verifier attack

The stolen-verifier attack means that an adversary that could steal the verification table from a server uses it to impersonate a legitimate user. The LDH protocol is vulnerable to stolen-verifier attack because *pw* is clearly stored on *S*. Then, an attacker that has stolen the verification table from a server can use users' passwords for masquerading as them. Here is how the attack works. An adversary that has *pw* masquerade as *U* and starts the LDH protocol by selecting number *t*, and sending ID_U and *t* to *S*. *S* selects *r* and *s*, computes C_1 and C_2, and sends C_1 and C_2 to the adversary. The adversary then decrypts C_1 and obtains $r' = \varphi^{-1}(D_{pw}(C_1))$. He/she then computes $C_3 = H(1, pw, r', t)$ and $SK_U = H(2, pw, r', t)$, and sends C_3 to *S*. *S* authenticates the adversary by verifying C_3, and computes $SK_S = H(2, pw, r, t)$.

3.5. Vulnerability to Unknown Key-Share (UKS) attack

In the UKS attack, a client coerces into sharing a session key with an entity while it believes that the session key has been shared with another entity. This attack is applicable to the LDH protocol since identifiers of user and server are not involved in session key derivation function or authentications. As it is somewhat hard to remember different passwords for several accounts, most users select the same password for their accounts on different servers and this can be used for a UKS attack to initiator of the LDH protocol. For a simple case that user *U* shared the same password on two servers S_1 and S_2, the only effort to accomplish a UKS attack is to intercept the communication between *U* and S_1 and forwarding all the messages from *U* to S_2 and response of S_2 to *U*. It may be necessary to convert ID_U

to the identifier of U on S_2. Consequently, U will mistakenly believe that it has shared a session key with S_1 while it has really shared the session key with S_2**,** and S_2 correctly believes that the session key is shared with U. This is a UKS attack against the initiator.

3.6. Further shortcomings

- No forward secrecy: The LDH protocol does not provide forward secrecy because when a user's password is revealed; all the previous session keys can be easily calculated. The attacker that has eavesdropped t and C_1 from previous sessions, obtains the random string r' as $r' = \varphi^{-1}(D_{pw}(C_1))$ and calculates the corresponding session key as $H(2, pw, r', t)$. Absence of "*perfect*" forward secrecy is considered in [4] as the only security problem of the LDH protocol.

- No key confirmation: As it is clear from the protocol description, there is not any provision in the LDH protocol for the key confirmation.

4. Conclusion

In this letter, it is proved that the LDH protocol [2] is vulnerable to password compromise impersonation, malicious server, offline password guessing, undetectable online password guessing, stolen-verifier, and Unknown Key-Share (UKS) attacks while it does not provide forward secrecy and key confirmation attributes.

References

[1] S. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks," in *1992 IEEE Symposium on Security and Privacy*, 1992, pp. 72–84.

[2] C. S. Laih, L. Ding, and Y. M. Huang, "Password-only authenticated key establishment protocol without public key cryptography," *Electronics letters*, vol. 41, pp. 185-186, 2005.

[3] Q. Tang and C. J. Mitchell, "Enhanced password-based key establishment protocol," 2005.

[4] E. J. Yoon and K. Y. Yoo, "An Improvement of Password-Only Authenticated Key Establishment Protocol without Public Key Cryptography," in *International Conference on Multimedia and Ubiquitous Engineering*, Busan, 2008, pp. 87-91.

[5] I. Erguler and E. Anarim, "A password-based key establishment protocol with symmetric key cryptography," in *IEEE International Conference on Wireless and Mobile Computing (WIMOB'08)*, 2008, pp. 543-548.

[6] Y. J. Fan, Q. Y. Wen, and H. Zhang, "Smart card-based authenticated key exchange protocol with CAPTCHA for wireless mobile network," in *2nd International Conference on Future Computer and Communication*, 2010, pp. 119-123.
[7] L. V. Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security," in *Advances in Cryptology - EUROCRYPT 2003*, 2003, pp. 294-311.