

A note on LASSI: a lightweight authenticated key agreement protocol for fog-enabled IoT deployment


Zhengjun Cao (✉ caozhj@shu.edu.cn)
Shanghai University

Research Article

Keywords: Authentication, Key agreement, Key transfer, Salted password hashing, Symmetric key encryption

Posted Date: March 11th, 2023

DOI: <https://doi.org/10.21203/rs.3.rs-2658557/v1>

License:  This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: No competing interests reported.

A note on “LASSI: a lightweight authenticated key agreement protocol for fog-enabled IoT deployment”

Zhengjun Cao

Abstract We show that the scheme [Int. J. Inf. Sec., 21(6), 1373–1387] is flawed, in which the user encrypts the temporary ID using a symmetric key encryption in order to achieve the anonymity target. If the shared key for such a symmetric key encryption is really available, the scheme can be greatly simplified. We want to stress that the ultimate use of a key agreement scheme is just to establish a shared key for some symmetric key encryption, but not vice versa.

Keywords Authentication · Key agreement · Key transfer · Salted password hashing · Symmetric key encryption

1 Introduction

Very recently, Abdussami *et al.* [1] have presented a key agreement scheme for fog-enabled IoT scenario, in which there are four entities: IoT device, Fog node (FN), Cloud server (CS), and User. The user will first authenticate with the cloud server. The cloud server will store the data sensed by the IoT devices received via fog nodes and give access to the authorized users. IoT devices and user devices are not trusted entities. It assumes that the adversary can compromise the private credentials such as secret keys and session keys. Its security requirements include user anonymity, integrity, authentication, forward secrecy, and confidentiality.

Though the scheme is interesting, we find it is flawed because the user has to invoke a symmetric key encryption to securely transfer the temporary ID to CS. But symmetric key encryption is a heavy cryptographic

primitive, for which the final key derived from a key agreement scheme is just served as a shared key. The scheme has confused key transfer with key agreement.

We also find that the scheme is vulnerable to guessing password attack, because the password is not salted. It neglects the fact that an identifier is the characteristics that distinguish it from others, which should be public and easily available. We want to stress that an identifier can be hidden in a concrete session, but it is publicly accessible in the system, otherwise such an identifier loses its signification.

2 Preliminaries

Key agreement, key distribution, key exchange, and key transfer [3], are often confused, but their common target is to establish a shared key between users. The resulting key in a key agreement scheme is not preexisting. However, the resulting key in a key transfer scheme is preexisting, which should be recovered intactly.

The difference between key agreement and key transfer seems unfamiliar to some researchers. To illustrate it, we now review the popular Diffie-Hellman key exchange [2] and RSA [4] (see Table 1).

Apparently, RSA requires a complex system setup, which relies on Public Key Infrastructure (PKI) to enable Bob to invoke Alice’s true public key (n, e) . Its authentication originates directly from the reliance on PKI. Such reliance could be unavailable for some scenarios. Whereas, a lightweight key agreement scheme is more applicable to this case.

It’s worth noting that the usual size of RSA modulus is greater than 2048 bits. The corresponding modular exponentiation is expensive for limited power devices. So, RSA is not directly used for transferring data, instead transferring session keys.

Zhengjun Cao
Department of Mathematics, Shanghai University, Shangda Road 99, Shanghai 200444, China.
E-mail: caozhj@shu.edu.cn

Table 1: Diffie-Hellman key exchange versus RSA

Diffie-Hellman key exchange	RSA
<i>Setup.</i> A prime p , a generator $g \in \mathbb{F}_p^*$.	<i>Setup.</i> Alice picks two big primes p, q , computes $n = pq$. Pick e and compute d such that $ed \equiv 1 \pmod{\phi(n)}$. Set the public key as (n, e) , the private key as d .
$A \rightarrow B$. Alice picks an integer x_A to compute $y_A \equiv g^{x_A} \pmod{p}$. Send y_A to Bob.	
$A \leftarrow B$. Bob picks an integer x_B to compute the key $k \equiv y_A^{x_B} \pmod{p}$, and $y_B \equiv g^{x_B} \pmod{p}$. Send y_B to Alice.	$A \leftarrow B$. For $m \in \mathbb{Z}_n^*$, Bob checks the certification of public key (n, e) , and computes $c \equiv m^e \pmod{n}$. Send c to Alice.
$A \downarrow$. Alice computes the key $k \equiv y_B^{x_A} \pmod{p}$.	$A \downarrow$. Alice computes $m \equiv c^d \pmod{n}$. (Usually, m is a session key, not a concrete message)

3 Review of the scheme

Let ID_i, PW_i be the identity and password of i th user, respectively. $H(\cdot)$ is a hash function. A physically unclonable function (PUF) responses differ from one different PUF instance for the same challenge, but it gives the same response for the same challenge in an instance. The user registration with cloud server and mutual authentication between them can be depicted as follows (see Table 2).

4 A paradox in the scheme

In the scheme the user has to use a symmetric key encryption to transfer the new temporary identifier, T_{idnew} , i.e.,

$$B_i = E_{H(R_{i1} \| T_{id})}(A_i \| T_{id} \| T_U \| T_{idnew}) \quad (1)$$

The fingerprint $H(R_{i1} \| T_{id})$ acts as a session key.

We find the scheme tries to use the current session key to negotiate a new session key $H(T_{id} \| R_{i1} \| T_C)$. But there is no ultimate difference between

$$H(R_{i1} \| T_{id}) \quad \text{and} \quad H(T_{id} \| R_{i1} \| T_C),$$

when they are used for session keys. Both are random outputs of a same hash function corresponding to two different inputs.

As we know, a symmetric key encryption is rarely used for transferring session keys because it requires that both sides know a pre-agreed secret key. It becomes a paradox to *use a shared secret key to merely negotiate a new secret key*.

5 A possible revision

As we discussed before, the negotiated key is ultimately used for a subsequent symmetric key encryption to transfer data. So, it is unnecessary to separate the target of mutual authentication and that of data transfer. In the proposed scenario, we find, the user and cloud server can **concurrently** achieve the two targets. See the following Table 3.

In the revised scheme, the ciphertext is

$$\hat{C} = E_{H(R_{i1} \| T_{id})}(A_i \| T_{id} \| T_U \| T_{idnew} \| h \| m) \quad (2)$$

in which two more components h, m are simultaneously encrypted. Its confidentiality comes directly from the original scheme. Besides, the checking of

$$h = H(R_{i1} \| T_{idnew}) \quad (3)$$

suffices for mutual authentication. Any adversary cannot generate such a fingerprint corresponding to the random temporary identifier T_{idnew} , because the component R_{i1} are only known to the legal user and the cloud server.

6 On the signification of an identifier

ID-based encryption introduced by Shamir [5], is a type of public-key encryption in which the public key of a user is some unique information about the user's identity. Parties may encrypt messages with no prior distribution of keys between individual participants. This is very useful in cases where pre-distribution of authenticated keys is inconvenient or infeasible.

The discussed threat model assumes that user devices are not trusted entities. The data stored in user devices can be retrieved by using the power analysis attack (see §3.3, [1]). In order to protect the user's identity

Table 2: User registration and authentication with cloud server

User		Cloud server
Registration		
Select Id_i, PW_i , and compute $A_i = H(Id_i PW_i) \oplus r_i$, where r_i is a random number. Pick a temporary identity T_{id} . Generate challenge-response pairs $C_i = (C_{i1}, C_{i2}, \dots)$, $R_i = (R_{i1}, R_{i2}, \dots)$ by $PUF_i(\cdot)$. Store $\{A_i, r_i, T_{id}, C_i\}$.	$\xrightarrow[\text{secure channel}]{A_i, T_{id}, C_i, R_i}$	Store $\{A_i, T_{id}, C_i, R_i\}$ for the user.
Mutual authentication and key agreement		
Enter Id_i, PW_i . The user device checks if $A_i = H(Id_i PW_i) \oplus r_i$. Generate the response R_{i1} , and select $T_{id_{new}}$, the time-stamp T_U . Compute $B_i = E_{H(R_{i1} T_{id})}(A_i T_{id} T_U T_{id_{new}})$. Check if $T_U - T_C \leq \Delta T$. Decrypt D_i . Check the consistency of $A_i + 1$. If ok, compute $S_U = H(T_{id} R_{i1} T_C)$, and check if $q_i = H(S_U A_i)$.	$\xrightarrow[\text{open channel}]{B_i, T_U, C_{i1}, T_{id}}$ $\xleftarrow{D_i, T_C, q_i}$	Check if $T_C - T_U \leq \Delta T$. Decrypt B_i and check the consistency of A_i, T_{id} . Update T_{id} with $T_{id_{new}}$. Compute $S_C = H(T_{id} R_{i1} T_C)$, $q_i = H(S_C A_i)$, $D_i = E_{H(R_{i1} T_{id})}(A_i + 1 T_C)$.
Subsequent data transfer		
For a message m , compute the ciphertext $\hat{C} = E_{S_U}(m)$	$\xrightarrow{\hat{C}}$	Compute the plaintext $m = D_{S_C}(\hat{C})$

Table 3: A possible revision

User		Cloud server
Data transfer		
Enter Id_i, PW_i . The user device checks if $A_i = H(Id_i PW_i) \oplus r_i$. Generate the response R_{i1} , and select $T_{id_{new}}$, the time-stamp T_U . For a message m , compute $S_U = H(R_{i1} T_{id})$, $h = H(R_{i1} T_{id_{new}})$, $\hat{C} = E_{S_U}(A_i T_{id} T_U T_{id_{new}} h m)$.	$\xrightarrow[\text{open channel}]{\hat{C}, T_U, C_{i1}, T_{id}}$	Check if $T_C - T_U \leq \Delta T$. Decrypt \hat{C} and check the consistency of A_i, T_{id} . Verify that $h = H(R_{i1} T_{id_{new}})$. Update T_{id} with $T_{id_{new}}$.

and password, the user device only stores $\{A_i, T_{id}, r_i, C_i\}$. It claims that the adversary cannot get the true identity Id_i and password PW_i even if the device is compromised, under the assumption that guessing the password and identity of the user separately is possible, whereas guessing both parameters in polynomial time is impractical.

The assumption ignores a basic fact: *any identifier in the whole system, which is the characteristics that distinguish it from others, is public and easily available*. Notice that an identifier can be hidden in a concrete session, but it is publicly accessible in the system.

Some researchers have neglected the difference between session-invisible identifier and system-visible identifier.

In view of this fact, we find, the scheme is vulnerable to guessing password attack. Actually, according to the assumption a powerful adversary can access A_i, r_i which are stored in a target user device. The target Id_i is also accessible because it is a system-visible parameter, otherwise such an identifier loses its significance. So, the adversary can test password dictionaries to search for a *password* such that

$$H(Id_i || \text{password}) = A_i \oplus r_i$$

By the way, it is common knowledge to use salted password hashing to improve password security, in which the salt is a random string added to a password before it's hashed, making it difficult for an attacker to retrieve the original password without having access to both the fingerprint and the salt. The scheme has deviated from this usual route.

7 Conclusion

In this note, we show that the Abdussami *et al.*'s key agreement scheme is flawed. We clarify the difference between key transfer and key agreement. We also reiterate the signification of an identifier. The findings could be helpful for the future work on designing key agreement schemes.

Declarations

The author has no financial or proprietary interests in any material discussed in this article.

Data Availability Statements

All data generated or analysed during this study are included in this published article.

References

1. Abdussami, M., Amin, R., Vollala, S.: LASSI: a lightweight authenticated key agreement protocol for fog-enabled iot deployment. *Int. J. Inf. Sec.* **21**(6), 1373–1387 (2022)
2. Diffie, W., Hellman, M.: New directions in cryptography. *IEEE Trans. Inf. Theory* **22**(6), 644–654 (1976)
3. Menezes, A., Oorschot, P., Vanstone, S.: *Handbook of Applied Cryptography*. CRC Press, USA (1996)
4. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978)
5. Shamir, A.: Identity-based cryptosystems and signature schemes. In: *Proceedings of Annual Cryptology Conference, Advances in Cryptology (CRYPTO'84)*, pp. 47–53. Santa Barbara, California, USA (1984)