A Note on "A Lightweight and Privacy-Preserving Mutual Authentication and Key Agreement Protocol for Internet of Drones Environment"

Zhengjun Cao[1], Lihua Liu[2]

Abstract. We show that the key agreement scheme [IEEE Internet Things J., 9(12), 2022, 9918–9933] is flawed. In order to authenticate each other, all participants use message authentication code (MAC) to generate tags for exchanged data. But MAC is a cryptographic technique which requires that the sender and receiver share a symmetric key. The scheme tries to establish a new shared key by using an old shared key, which results in a vicious circle. To the best of our knowledge, it is the first time to discuss such a flaw in the related literatures.

Keywords: Key agreement, Internet of drones, Random shuffle, Message authentication code, Key update.

1 Introduction

Recently, Pu *et al.* [1] have presented a key agreement scheme for Internet of Drones system. The scheme makes use of many cryptographic techniques, such as physical unclonable function (PUF), random shuffling, MAC, and chaotic system, to construct session keys among drones and zone service providers (ZSP). Its security goals include entity authentication, data confidentiality, integrity, session key agreement, user anonymity, and immune against various attacks. In this note, we remark that the scheme is flawed because there is a vicious circle. We want to stress that a simpler key update mechanism suffices for the considered scenario.

2 Review of the scheme

Let Z_s be the identity of the sth ZSP, ID_i, PID_i^t be the real identity and pseudonym of the ith drone, respectively, (C_i^t, R_i^t) be the PUF challenge-response pair of ID_i, $S(\cdot)_{(C_i^t, R_i^t)}$ be the random shuffling with (C_i^t, R_i^t), and $S^{-1}(\cdot)_{(C_i^t, R_i^t)}$ be the reverse process of random shuffling. $C(\cdot)$ is a MAC function, and $H(\cdot)$ is a secure one-way hash function. A PUF is represented as a function P, i.e., $R = P(C)$, where C and R are the input challenge and the output response of PUF, respectively.

We now only describe the key-agreement process between a drone and a ZSP as follows (see Table 1, [1]).

[1]Department of Mathematics, Shanghai University, Shanghai, 200444, China
[2]Department of Mathematics, Shanghai Maritime University, Shanghai, 201306, China.
Email: liulh@shmtu.edu.cn

Table 1: The Pu *et al.*'s key agreement scheme

Drone $ID_i : \{C_i^t\}$		ZSP Z_s
Compute $R_i^t = P(C_i^t)$, $PID_i^t = H(ID_i \Vert R_i^t)$.		
Pick a nonce N_i^t, to compute		Locate PID_i^t to fetch $[ID_i, PID_i^t, (C_i^t, R_i^t)]$.
$M_1 = S(PID_i^t \Vert Z_s \Vert N_i^t)_{(C_i^t, R_i^t)}$,	$\xrightarrow[\text{[open channel]}]{M_1, MAC_1}$	Compute $PID_i^t \Vert Z_s \Vert N_i^t = S^{-1}(M_1)_{(C_i^t, R_i^t)}$.
$MAC_1 = C(M_1 \Vert N_i^t)$.		Check if $MAC_1 = C(M_1 \Vert N_i^t)$. If so, pick a nonce N_s^{t+1} to compute
Compute $PID_i^t \Vert Z_s \Vert N_i^t \Vert N_s^{t+1} = S^{-1}(M_2)_{(C_i^t, R_i^t)}$.	$\xleftarrow{M_2, MAC_2}$	$M_2 = S(PID_i^t \Vert Z_s \Vert N_i^t \Vert N_s^{t+1})_{(C_i^t, R_i^t)}$,
Check if $MAC_2 = C(M_2 \Vert N_i^t \Vert N_s^{t+1})$.		$MAC_2 = C(M_2 \Vert N_i^t \Vert N_s^{t+1})$.
If so, pick a nonce N_i^{t+1} to compute		
$C_i^{t+1} = S(N_s^{t+1} \Vert N_i^{t+1})_{(C_i^t, R_i^t)}$, $R_i^{t+1} = P(C_i^{t+1})$,		
$M_3 = S(PID_i^t \Vert Z_s \Vert N_s^{t+1} \Vert N_i^{t+1})_{(C_i^t, R_i^t)}$,		Compute $PID_i^t \Vert Z_s \Vert N_s^{t+1} \Vert N_i^{t+1} = S^{-1}(M_3)_{(C_i^t, R_i^t)}$,
$M_4 = S(PID_i^t \Vert Z_S \Vert N_S^{t+1} \Vert N_i^{t+1} \Vert R_i^{t+1})_{(C_i^t, R_i^t)}$,	$\xrightarrow{M_3, M_4, MAC_{34}}$	$PID_i^t \Vert Z_S \Vert N_S^{t+1} \Vert N_i^{t+1} \Vert R_i^{t+1} = S^{-1}(M_4)_{(C_i^t, R_i^t)}$.
$MAC_{34} = C(M_3 \Vert M_4 \Vert N_i^{t+1} \Vert R_i^{t+1})$,		Check if $MAC_{34} = C(M_3 \Vert M_4 \Vert N_i^{t+1} \Vert R_i^{t+1})$.
$SK_{i,s} = H(N_i^{t+1}) \oplus H(N_S^{t+1})$.		If so, compute $C_i^{t+1} = S(N_s^{t+1} \Vert N_i^{t+1})_{(C_i^t, R_i^t)}$,
Update $\{C_i^{t+1}\}$.		$PID_i^{t+1} = H(ID_i \Vert R_i^{t+1})$,
		$SK_{s,i} = H(N_s^{t+1}) \oplus H(N_i^{t+1})$.
		Update $[ID_i, PID_i^{t+1}, (C_i^{t+1}, R_i^{t+1})]$.

Table 2: The revision

Drone $ID_i : \{C_i^t\}$		ZSP Z_s
Compute $R_i^t = P(C_i^t)$, $PID_i^t = H(ID_i \Vert R_i^t)$, $s_i^t = H(PID_i^t \Vert C_i^t \Vert R_i^t)$. Pick a nonce N_i^t, to		
compute $C_i^{t+1} = H(ID_i \Vert N_i^t)$, $R_i^{t+1} = P(C_i^{t+1})$,		Locate PID_i^t to fetch $[ID_i, PID_i^t, (C_i^t, R_i^t)]$.
$M = Enc_{s_i^t}(PID_i^t \Vert N_i^t \Vert C_i^{t+1} \Vert R_i^{t+1})$.	$\xrightarrow{PID_i^t,\ M}$	Compute $s_i^t = H(PID_i^t \Vert C_i^t \Vert R_i^t)$,
Update $\{C_i^{t+1}\}$.		$\widetilde{PID_i^t} \Vert \widetilde{N_i^t} \Vert \widetilde{C_i^{t+1}} \Vert \widetilde{R_i^{t+1}} = Dec_{s_i^t}(M)$. Check if $PID_i^t = \widetilde{PID_i^t}$, and $\widetilde{C_i^{t+1}} = H(ID_i \Vert \widetilde{N_i^t})$. If so, compute $PID_i^{t+1} = H(ID_i \Vert R_i^{t+1})$.
		Update $[ID_i, PID_i^{t+1}, (C_i^{t+1}, R_i^{t+1})]$.

3 Analysis

The scheme piles many techniques and seems hard to comprehend. We now only show some clear flaws.

3.1 A self-contradictory specification

The zone service provider Z_s cannot retrieve the drone's pseudonym PID_i^t by the received data $\{M_1, MAC_1\}$. The original specification is self-contradictory:

> *ZSP Z_s first tries to locate PID_i^t in the database. If PID_i^t is not found, the authentication request is rejected. Otherwise, it fetches the entry $[ID_i, PID_i^t, (C_i^t, R_i^t)]$ for drone ID_i. Then, it retrieves N_i^t from M_1 through $S^{-1}(M_1)$, $\cdots$*

In fact, the pseudonym PID_i^t is hidden in the ciphertext

$$M_1 = S(PID_i^t \Vert Z_s \Vert N_i^t)_{(C_i^t, R_i^t)}$$

To decrypt the ciphertext, Z_s needs to use the pair (C_i^t, R_i^t) at first. But Z_s cannot determine which is the target entry $[ID_i, PID_i^t, (C_i^t, R_i^t)]$.

To fix, the pseudonym PID_i^t should be directly transferred. Since it is updated in each session, an adversary cannot use it to reveal the true identity ID_i.

3.2 A vicious circle

Message authentication code, also referred to as a tag, is used to authenticate the origin and nature of a message. MAC ensures that the message is coming from the correct sender, has not been changed, and that the data transferred over a network or stored in or outside a system is legitimate and does not contain harmful code [2].

Given a message, the MAC system uses a symmetric key algorithm to generate an authentication tag by processing the message. The resulting computation is the message's MAC, which is then appended to the message and transmitted to the receiver. The receiver computes the MAC using the same algorithm. If the resulting MAC the receiver arrives at equals the one sent by the sender, the message is verified as authentic, and legitimate.

Notice that *MAC uses a secure key only known to the sender and the recipient* to process the target message. In the proposed scheme, the secure key is not specified. Concretely,

$$\begin{aligned} MAC_1 &= C(M_1 \| N_i^t), \quad MAC_2 = C(M_2 \| N_i^t \| N_s^{t+1}), \\ MAC_{34} &= C(M_3 \| M_4 \| N_i^{t+1} \| R_i^{t+1}) \end{aligned}$$

In these computations, the essential key for MAC is not mentioned at all. For convenience, we now denote it by sk_{MAC}.

To securely transfer data, it is usual to utilize symmetric key encryption, such as AES. But the scheme only uses random shuffle to encrypt data, with the shared key (C_i^t, R_i^t). Concretely,

$$\begin{aligned} PID_i^t \| Z_s \| N_i^t &\xrightarrow{S(\cdot)_{(C_i^t, R_i^t)}} M_1, \\ M_1 &\xrightarrow{S^{-1}(\cdot)_{(C_i^t, R_i^t)}} PID_i^t \| Z_s \| N_i^t, \\ PID_i^t \| Z_s \| N_i^t \| N_s^{t+1} &\xrightarrow{S(\cdot)_{(C_i^t, R_i^t)}} M_2, \\ M_2 &\xrightarrow{S^{-1}(\cdot)_{(C_i^t, R_i^t)}} PID_i^t \| Z_s \| N_i^t \| N_s^{t+1}, \\ &\cdots \end{aligned}$$

In nature, the two parties use the current shared parameters $[sk_{MAC}, PID_i^t, (C_i^t, R_i^t)]$ to negotiate new shared parameters $[sk_{MAC}, SK_{s,i}, PID_i^{t+1}, (C_i^{t+1}, R_i^{t+1})]$. Besides, the mechanism for updating the key sk_{MAC} is not specified at all.

We want to stress that there is no essential difference between the shared keys $[sk_{MAC}, PID_i^t, (C_i^t, R_i^t)]$ and $[sk_{MAC}, SK_{s,i}, PID_i^{t+1}, (C_i^{t+1}, R_i^{t+1})]$, when they are used for encryption. It becomes a vicious circle to exchange the current shared key only for the new shared key. Actually, a simpler key update mechanism suffices for the considered scenario.

3.3 A possible revision

Let $Enc_{(\cdot)}(\cdot)$ be the AES encryption algorithm, and $Dec_{(\cdot)}(\cdot)$ be the AES decryption algorithm. The possible revision can be described as follows (see Table 2).

In the revision, the hash value $s_i^t = H(PID_i^t \| C_i^t \| R_i^t)$ is shared by the two parties. It acts as either the session key or the authentication credential. The consistency-checking of

$$PID_i^t = \widetilde{PID_i^t} \text{ and } \widetilde{C_i^{t+1}} = H(ID_i \| \widetilde{N_i^t})$$

ensures that the sender is a legitimate drone, and the new challenge-response pair (C_i^{t+1}, R_i^{t+1}) is properly generated.

4 Conclusion

We show that the Pu *et al.*'s key agreement scheme is flawed. We want to stress that in some cases key agreement and key update can be concurrently obtained, depending on whether there is some shared credential.

References

[1] C. Pu, et al.: A Lightweight and Privacy-Preserving Mutual Authentication and Key Agreement Protocol for Internet of Drones Environment. IEEE Internet Things J. 9(12): 9918–9933 (2022)

[2] A. Menezes, P. Oorschot, S. Vanstone: Handbook of Applied Cryptography. CRC Press, USA (1996)