A note on "a multi-instance cancelable fingerprint biometric based secure session key agreement protocol employing elliptic curve cryptography and a double hash function"

Zhengjun Cao[1], Lihua Liu[2]

Abstract. We show that the key agreement scheme [Multim. Tools Appl. 80:799-829, 2021] is flawed. (1) The scheme is a hybrid which piles up various tools such as public key encryption, signature, symmetric key encryption, hash function, cancelable templates from thumb fingerprints, and elliptic curve cryptography. These tools are excessively used because key agreement is just a simple cryptographic primitive in contrast to public key encryption. (2) The involved reliance is very intricate. Especially, the requirement for a secure channel between two parties is generally unavailable.
Keywords: Public key encryption; Key agreement; Key transfer; Mutual authentication; Digital signature; Symmetric key encryption.

1 Introduction

Identification schemes in which a claimed identity is either corroborated or rejected immediately, are simpler than digital signature schemes, which involve a variable message and typically provide a non-repudiation feature. Entity authentication techniques can be divided into three categories, depending on which of the following the security is based [4]: something known (password, personal identification number, private key, etc), something possessed (chipcard, hand-held customized calculator, etc), and something inherent (handwritten signature, fingerprint, voice, etc).

A key agreement protocol is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by each of these, such that no party can predetermine the resulting value. Many key agreement protocols involve a trusted party, which is referred to trusted third party, trusted server, authentication server, key distribution center, or certification authority.

Recently, Sarkar and Singh [6] have presented a multi-instance cancelable fingerprint biometric-based key agreement scheme without saving the primary biometric data in the database. The scheme combines various tools such as public key encryption, signature, symmetric key encryption, hash function, cancelable templates from thumb fingerprints, and elliptic curve cryptography. But we find they have misused the reliance of Public Key Infrastructure (PKI) and secure channels. The hybrid scheme can be greatly simplified if such reliance is available. We also notice that the scheme has some notations/descriptions, incompatible with general conventions.

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

2 Preliminary

Key agreement, key distribution, key exchange, and key transfer, 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 the difference, we now review Diffie-Hellman key exchange [1] and RSA public key encryption [5](see Table 1).

Table 1: Diffie-Hellman key exchange versus RSA

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

Note that RSA requires a complex system setup, which relies on PKI to enable Bob to invoke Alice's true public key (n, e). Its authentication originates directly from the reliance on PKI. If such moderate reliance is unavailable, a lightweight key agreement scheme could be chosen.

3 Review of the Sarkar-Singh scheme

In the considered scenario, there are two parties A and B. Let ID_A, ID_B be the registered IDs, K_{privA}, K_{pubA}, K_{privB}, K_{pubB} be the private keys and public keys, CT_A, CT_B be the cancelable templates generated from the right thumb impressions, HT_{PartyA}, HT_{PartyB} be the hash values generated from the left thumb impressions, CT_{AXOR}, CT_{BXOR} be the XOR values of cancelable template, of user A and B, respectively. The scheme consists of three phases: pre-computation, session key generation, and secure communication. It can be described as follows (see Table 2).

4 Intricate requirements for reliance

As we know, any cryptographic protocol requires a certain reliance. For example, in the Diffie-Hellman scheme the system parameters p and g must be credible. While, in RSA the invoked public key (n, e) must be authentic. In the Sarkar-Singh scheme, the required reliance is very intricate (see Table 3). It requires a secure channel between party A and party B for Pre-computation. The secure channel must be private and authentic so as to ensure the confidentiality of the transferred parameters p, x, and the integrities of signatures $(p \| ID_B)K_{privA}$, $(x \| ID_A)K_{privB}$ and public key certificates

Table 2: The Sarkar-Singh key agreement scheme

Party A	Pre-computation	Party B
Make a cancelable template CT_A from his right thumb fingerprint template. Calculate a hash table HT_{PartyA} from his left thumb fingerprint. Generate the secret value of p.		Verify the signature. If true, make a cancelable template CT_B
Generate the signature $(p\|ID_B)K_{privA}$.	$\xRightarrow[\text{[secure channel]}]{(p\|ID_B)K_{privA},\ p,\ Certi_A}$	from his fingerprint template.
where $Certi_A$ is his public key certificate.		Calculate a hash table HT_{PartyB} from his left thumb fingerprint. Pick a value $x\,(0 < x < (p-1)$.
Verify the signature. If true,	$\xLeftarrow{(x\|ID_A)K_{privB},\ x,\ Certi_B}$	Generate the signature $(x\|ID_A)K_{privB}$.
compute y such that $y^2 = x^3 + x + 1 \bmod p$. Store the point $Q(x, y)$.		Compute y such that $y^2 = x^3 + x + 1 \bmod p$. Store the point $Q(x, y)$.
Party A	**Session key generation**	**Party B**
Compute a XOR operation on every points of his cancelable template CT_A, i.e., $CT_AXOR = (x'_1 \oplus y'_1 \oplus x'_2 \oplus y'_2 \cdots x'_n \oplus y'_n)$. Pick a number n_1, compute $x_1 = n_1 + CT_AXOR \bmod p$, $y_1^2 = x_1^3 + x_1 + 1 \bmod p$. Get a point $P_1(x_1, y_1)$. Compute $R_1 = P_1 + Q$. Generate the signature $(Hash - valueof - R_1)_{PrivatekeyA}$.		
Compute the ciphertext $C_1 =$ $[R_1, (Hash - valueof - R_1)_{PrivatekeyA}]_{publickeyB}$	$\xrightarrow[\text{[insecure channel]}]{C_1}$	Decrypt C_1 to get R_1. Verify the signature to accept R_1. Compute a XOR operation on every points of his cancelable template, i.e., $CT_BXOR = (x''_1 \oplus y''_1 \oplus x''_2 \oplus y''_2 \cdots x''_n \oplus y''_n)$. Pick a number n_2, compute $x_3 = n_2 + CT_BXOR \bmod p$, $y_3^2 = x_3^3 + x_3 + 1 \bmod p$. Get a point $P_2(x_3, y_3)$. Compute $R_2 = P_2 + Q$. Generate the signature $(Hash - valueof - R_2)_{PrivatekeyB}$. Compute the ciphertext $C_2 =$ $[R_2, (Hash - valueof - R_2)_{PrivatekeyB}]_{publickeyA}$. Compute $S(x_5, y_5) = P_2 + R_1$. Generate the signature σ_1 for $ID_A\|R_1\|R_2$ with the private key K_{privB}.
Decrypt C_2 to get R_2. Verify the signature to accept R_2. Compute $S(x_5, y_5) = P_1 + R_2$.	$\xleftarrow{C_2, C_3}$	Encrypt the signature with secret value S to get the ciphertext C_3.
Decrypt C_3 with the value S to get the signature σ_1. Verify the signature. If true, generate the signature σ_2 for $ID_B\|R_1\|R_2$ with the key K_{privA}. Encrypt the signature with value S to get the ciphertext C_4.	$\xrightarrow{C_4}$	Decrypt C_4 to get σ_2. Verify the signature. If true,
Decrypt C_5 to get HT_{PartyB}.	$\xleftarrow{C_5}$	encrypt HT_{PartyB} to get the ciphertext C_5.
Encrypt HT_{PartyA} to get the ciphertext C_6.	$\xrightarrow{C_6}$	Decrypt C_6 to get HT_{PartyA}.
Set the key $K = (x_5\|y_5\|HT_{PartyA}\|HT_{PartyB})$.		Set the key $K = (x_5\|y_5\|HT_{PartyA}\|HT_{PartyB})$.

Remarks: The notations $C_1, \cdots, C_6$, σ_1, σ_2 are not used in the original description. Instead, it is vaguely described such as the sentences "Party B now sends the signature to party A. This signature is encrypted with secret value S". "Party A now sends the signature to party B. This signature is encrypted with secret value $S(x_5, y_5)$". (see page 813-814, [6])

$Certi_A, Certi_B$. Otherwise, the parameters p, x can be eavesdropped, and $Certi_A, (p\|ID_B)K_{privA}$ could be falsified by an adversary, which results in the failure of signature verification.

Table 3: Different reliance required in three key agreement/transfer schemes

Diffie-Hellman	RSA	Sarkar-Singh
① system parameters p and g — weak reliance	① PKI to authenticate the public key (n, e) — moderate reliance	① a secure channel for Pre-computation — strong reliance ② PKI to authenticate the public keys $publickeyB$ and $publickeyA$. — moderate reliance ③ public verification algorithms for verifying the signatures σ_1, σ_2 — moderate reliance ④ a system's hash function $h(\cdot)$ for computing $(Hash - valueof - R_1)$ and $(Hash - valueof - R_2)$. — weak reliance ⑤ a symmetric key encryption algorithm for transferring the signatures σ_1, σ_2, and the hash tables HT_{PartyA}, HT_{PartyB}. — weak reliance

Table 4: A simplification

Party A	Pre-computation	Party B
Pick a 256-bit nonce α.	$\xrightarrow{\alpha,\ ID_A}$ [secure channel]	
Store $\{\alpha, \beta, ID_B\}$.	$\xleftarrow{\beta,\ ID_B}$	Pick a 256-bit nonce β. Store $\{\alpha, \beta, ID_A\}$.
Party A	**session key agreement**	**Party B**
Compute $\gamma_1 = h(\alpha\|ID_A\|ID_B)$, where $h(\cdot)$ is a 256-bit hash function.	$\xrightarrow{\gamma_1}$ [insecure channel]	Invoke the stored data to compute $\gamma_1' = h(\alpha\|ID_A\|ID_B)$. Check that $\gamma_1 = \gamma_1'$. If true,
Invoke the stored data to compute $\gamma_2' = h(\beta\|ID_A\|ID_B)$. Check that $\gamma_2 = \gamma_2'$. If true, set the session key $K = h(\alpha\|\beta\|ID_A\|ID_B)$.	$\xleftarrow{\gamma_2}$	compute $\gamma_2 = h(\beta\|ID_A\|ID_B)$, and set the session key $K = h(\alpha\|\beta\|ID_A\|ID_B)$.
Pick a 256-bit nonce α_{new}, compute $\lambda_1 = K \oplus \alpha_{new} \oplus h(\alpha\|ID_A)$, $\psi_1 = h(\alpha_{new}\|ID_A)$.	$\xrightarrow{\lambda_1,\psi_1}$	Compute $\alpha'_{new} = \lambda_1 \oplus K \oplus h(\alpha\|ID_A)$ Check that $\psi_1 = h(\alpha'_{new}\|ID_A)$. If true, pick a 256-bit nonce β_{new}, compute
Compute $\beta'_{new} = \lambda_2 \oplus K \oplus h(\beta\|ID_B)$. Check that $\psi_2 = h(\beta'_{new}\|ID_B)$. If true, update $\alpha \leftarrow \alpha_{new}, \beta \leftarrow \beta'_{new}$.	$\xleftarrow{\lambda_2,\psi_2}$	$\lambda_2 = K \oplus \beta_{new} \oplus h(\beta\|ID_B)$, $\psi_2 = h(\beta_{new}\|ID_B)$, and update $\alpha \leftarrow \alpha'_{new}, \beta \leftarrow \beta_{new}$.

We want to stress that the scheme can be greatly simplified if such strong reliance for a secure channel is available. Here is a simplification (see Table 4).

Notice that party B authenticates party A by checking $\gamma_1 = \gamma_1'$. Likewise, party A authenticates party B by checking $\gamma_2 = \gamma_2'$. After the mutual authentication is completed, they use the session key K to securely transfer $\alpha_{new}, \beta_{new}$ and update $\alpha \leftarrow \alpha_{new}, \beta \leftarrow \beta_{new}$, in order to provide forward secrecy. An adversary who captures γ_1 which equals to $h(\alpha \| ID_A \| ID_B)$, and γ_2 which equals to $h(\beta \| ID_A \| ID_B)$, cannot derive the session key $K = h(\alpha \| \beta \| ID_A \| ID_B)$ due to the one-way and collision-free properties of $h(\cdot)$. As for its security proof, we refer to Ref.[3], in which the proposed scheme was very similar to the above simplification.

5 Other flaws

The scheme is not explicitly described, because there are many unwonted notations, descriptions, and reiterations of general knowledge. It takes almost 4 pages (see page 807-810, Ref.[6]) to reiterate the points addition on an elliptic curve, which is common knowledge in any textbook on elliptic curve theory, like Ref.[2]. The reiterations distract attention heavily. Some notations are tedious and incompatible with general conventions. Here are some unwonted notations/descriptions (see Table 5).

Table 5: Unwonted notations/descriptions

Sarkar-Singh notation/description	Conventional notation/description
Party A concatenates party B's identity ID_B and calculated p value, signs the result with his private key K_{privA}, and conveys the value of p, the signature, and his individual public key certificate to party B.	Let $Sig(\cdot,\cdot)$ be the signing algorithm. Party A generates the signature $\sigma = Sig(K_{privA}, p \| ID_B)$. Send $\{p, \sigma, Certi_A\}$ to party B.
$(Hash - valueof - R_1)$	Let $h(\cdot)$ be the hash function. Compute $h(R_1)$.
$(Hash - valueof - R_1)_{PrivatekeyA}$	Let sk_A be the party A's secret key. Compute the signature $\sigma' = Sig(sk_A, h(R_1))$.
$[R_1, (Hash - valueof - R_1)_{PrivatekeyA}]_{publickeyB}$	Let pk_B be the party B's public key, $Enc(\cdot,\cdot)$ be the encrypting algorithm. Compute $\nu = Enc(pk_B, R_1 \| Sig(sk_A, h(R_1)))$.

By the way, the final key is simply set as $K = (x_5 \| y_5 \| HT_{PartyA} \| HT_{PartyB})$. But the randomness of the concatenated string is generally insufficient for practical applications. It is better to set the key as a hash value, i.e., $K = h(x_5 \| y_5 \| HT_{PartyA} \| HT_{PartyB})$.

6 Further discussions

The step 6 (see page 814, Ref.[6]) is described as:

> *Party A now sends the signature to party B. This signature is encrypted with secret value* $S(x_5, y_5)$.

Actually, the point $S(x_5, y_5)$ is used as the key for a symmetric key encryption, though which has not been explicitly specified. Note that the final key $K = (x_5 \| y_5 \| HT_{PartyA} \| HT_{PartyB})$ is also used as a symmetric key for the later data transfer process. So, it makes use of the intermediate symmetric

key $S(x_5, y_5)$ to negotiate another symmetric key $(x_5 \| y_5 \| HT_{PartyA} \| HT_{PartyB})$. But there is no any virtual difference between the two keys when their hash values are used for symmetric key encryption. It seems that Sarkar and Singh have neglected the vicious circle hidden in their presentation.

7 Conclusion

We show that the Sarkar-Singh key agreement scheme is flawed because of the excessively used tools, intricate reliance, and cumbersome notations. We hope this note could be helpful for the future work on designing such schemes.

References

[1] Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory 22(6), 644-654 (1976)

[2] Hankerson D., Vanstone S., Menezes A.: Guide to Elliptic Curve Cryptography. Springer New York, USA (2006)

[3] Khan H., Dowling B., Martin K.: Pragmatic authenticated key agreement for IEEE Std 802.15.6. Int. J. Inf. Sec. 21(3): 577-595 (2022)

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

[5] Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120-126 (1978)

[6] Sarkar A., Singh B.: A multi-instance cancelable fingerprint biometric based secure session key agreement protocol employing elliptic curve cryptography and a double hash function. Multim. Tools Appl. 80(1): 799-829 (2021)