Computer Systems Science & Engineering
DOI:10.32604/csse.2022.020799
Article

Tech Science Press

Analysis of the Desynchronization Attack Impact on the E2EA Scheme

Shadi Nashwan*

Computer Science Department, College of Computer and Information Sciences, Jouf University, Sakaka, 42421, Saudi Arabia
*Corresponding Author: Shadi Nashwan. Email: shadi_nashwan@ju.edu.sa
Received: 08 June 2021; Accepted: 26 July 2021

Abstract: The healthcare IoT system is considered to be a significant and modern medical system. There is broad consensus that these systems will play a vital role in the achievement of economic growth in numerous growth countries. Among the major challenges preventing the fast and widespread adoption of such systems is the failure to maintain the data privacy of patients and the integrity of remote clinical diagnostics. Recently, the author proposed an end-to-end authentication scheme for healthcare IoT systems (E2EA), to provide a mutual authentication with a high data rate between the communication nodes of the healthcare IoT systems. Although the E2EA authentication scheme supports numerous attractive security services to resist various types of attack, there is an ambiguous view of the impact of the desynchronization attack on the E2EA authentication scheme. In general, the performance of the authentication scheme is considered a critical issue when evaluating the applicability of such schemes, along with the security services that can be achieved. Therefore, this paper discusses how the E2EA authentication scheme can resist the desynchronization attack through all possible attack scenarios. Additionally, the effect of the desynchronization attack on the E2EA scheme performance is analyzed in terms of its computation and communication costs, based on a comparison with the recently related authentication schemes that can prevent such attack. Moreover, this research paper finds that the E2EA authentication scheme can not only prevent the desynchronization attack, but also offers a low cost in terms of computations and communications, and can maintain consistency and synchronization between the communication nodes of the healthcare IoT systems during the next authentication sessions.

Keywords: Desynchronization attack; healthcare IoT systems; E2EA scheme; mutual authentication; anonymity; perfect forward secrecy

1 Introduction

The healthcare IoT system is one of the most important medical systems. There is a broad consensus that these systems will play an essential function in achieving economic growth for several growth countries in terms of the health of their societies [1–4]. The core goal of the healthcare IoT system is to remotely monitor the medical state of the patients. Moreover, the healthcare IoT system can offer other benefits, such as reducing the number of healthcare centers and the expenditures they incur. It can cover also the shortfall

in the number of health care centers located in distant areas, and patients can be easily treated by professional doctors and experts from around the world [5,6].

The fundamental process of the healthcare IoT systems can be summarized easily and simply. The patients' vital signs are collected through specific sensor devices, such as body temperature, blood sugar, blood pressure, pedometer, etc. Then, these collected data are sent to the healthcare service provider by wireless medical sensor network technology (WMSN), using internet communication channels [7]. After this, professional doctors monitor the patient's medical status using the physiological data of the patients that periodically arrive at the healthcare service provider [8–10].

Among the major challenges limiting the fast and widespread adoption of the healthcare IoT system are maintain patients' data privacy and the integrity of remote clinical diagnostics. Attackers can access and collect patients' vital signs data, and this issue can lead to passive consequences such as patients losing their jobs and their health insurance. Moreover, attackers also can tamper with communication between the patient and doctors, which could cause incorrect clinical diagnosis and lead to the wrong orders or advice being given to the patients [11].

With the growing demand for healthcare IoT systems, many healthcare IoT authentication schemes have been proposed to resolve the security weaknesses and to prevent different types of attack that target the patient's privacy and the integrity of remote clinical diagnostics [12–20]. These attacks can be summarized as the password table attack, man-in-the-middle attack, wrong login information attack, replay attack, impersonate attack, insider attack, smart card loss attack, and desynchronization attack [21–25]. Most healthcare IoT authentication schemes that have been proposed are unable to prevent all these attacks, especially the desynchronization attack [26,27].

Usually, to maintain consistency and synchronization between the communication nodes in such systems, the authentication schemes use different synchronization techniques, such as random numbers, pseudonym identities, timestamps, serial numbers, and hash functions [11–24]. Therefore, the improper use of such techniques to renew the nodes' identities and the session keys that will be used in the next authentication sessions may lead to extensive computational costs to prevent desynchronization attacks [25,28]. Unfortunately, healthcare IoT systems suffer from restrictions due to the limitations of the memory space and the low computation abilities of the sensor nodes that collect the patients' vital signs data [29]. Therefore, the performance of the authentication scheme is considered a critical issue when evaluating the applicability of the authentication schemes and the security services that can be achieved [30–33].

Recently, the author has proposed an end-to-end authentication scheme for healthcare IoT systems (E2EA) [26]. The E2EA authentication scheme can satisfy a set of attractive security services such as mutual authentication, anonymity, and perfect forward services in all stages of the authentication scheme. The performance analysis has proved that the E2EA authentication scheme has a low cost in terms of communications, computations, and storage space compared with other recently proposed authentication schemes. Moreover, the E2EA authentication scheme can prevent numerous related-attack types, including desynchronization attacks.

There is an ambiguous view of the impact of desynchronization attacks on the E2EA scheme. Therefore, this paper discusses how the E2EA authentication scheme can resist desynchronization attacks throughout different attack scenarios, as well as analyzing the effect of the desynchronization attack on the E2EA authentication scheme's performance.

This paper is organized as follows: The E2EA authentication scheme is reviewed in Section 2. Section 3 offers an analysis of the E2EA authentication scheme of resisting desynchronization attacks under all possible scenarios. An analysis of the desynchronization attack's impact on the E2EA authentication scheme performance is presented in Section 4. Finally, the conclusion of the paper is given in Section 5.

2 Review of E2EA Authentication Scheme

The stages of the E2EA authentication scheme are classified into four categories. The first category is the registration stages for the physician, patient, and WMSN nodes. The second category is the login stages for physician and patient nodes. The third category is the password change stages for physician and patient nodes. The last category is the authentication stages, which are called the long-term, short-term, and WMSN node authentication stages [26].

E2EA authentication scheme is executed between four authentication entities: the physician node (Pi) represents the professional doctor, the gateway node (GWN) node represents the healthcare service provider, the smart device node (SDj) represents the participant's patient, and the WMSN node (Sk) represents the sensors that will collect the patient's vital signs and be accessed by the physician. Tab. 1 summarizes the notations of the E2EA authentication scheme [26].

Table 1: Notation of the E2EA scheme

Notation	Description
PIDi	Pi node's identity number.
PPWi	Pi node's password.
PSCi	Pi node's security code.
SNi	Session number that calculated by Pi node and GWN node.
SCi	Pi node's smart card.
IDi	Pi node's identity that used in the GWN side.
Xi	GWN node's secret key for Pi node.
SIDj	SDj node's identity number of.
SPWj	Password of SDj node.
SSCj	SDj node's security Code.
SNj	Session number that calculated by SDj node and GWN node.
SCj	Smart card of smart device SDj node.
IDj	SDj node's identity that used in the GWN side.
Xj	GWN node's secret key for SDj node.
Sk	Sensor node.
SIDk	Sk node's identity number.
SSk0, SSk1	Sk node's sequence numbers.
ST	Type of WMSN node.
PSij	Subsequent Key that calculated by GWN node.
E/D	Encryption/Decryption functions.
h	One-way hash function.
Φ	Empty value
$\oplus$	Xor operation
{}	Transmitted message

This section reviews the registration and authentication stages, where the other stages are not affected or will be affected by the desynchronization attack scenarios discussed in the next section. It should be noted that the registration stages are executed by authentication entities through secure communication channels, while the authentication stages are executed through public communication channels.

2.1 Physician Registration Stage

When a physician (Pi) wants to access a patient's node (SDk), the Pi needs to register in the GWN. Then, the GWN issues a smart card (SCi) to the Pi as a response to the registration request message that was sent from Pi to the GWN, as shown in Fig. 1. This procedure is summarized as follows:

(1) Using a secure channel, the Pi sends the registration request message {M1: PIDi, Ci, PSCi} to the GWN. The identity number (PIDi), and security code (PSCi) are selected by the Pi. Additionally, the value of (Ci) is computed as Ci = h2 (PIDi || PPWi || R0), where both the password (PPWi) and the random number (R0) are also generated by the Pi;

(2) Upon receipt of the {M1} message, the GWN verifies whether the PIDi has registered previously. If it exists in the physician's table, the GWN rejects {M1} and tells the Pi to choose another PIDi. Otherwise, GWN computes SNi = h0 (R1), PKi = h1 (PIDi || Xi), PFi = (PKi $\oplus$ PSCi), and PVi = h1 ((SNi || PSCi) $\oplus$ (Ci || PKi)), where the (R1) is a random number and (Xi) is a secret key that was generated by the GWN. Then, the GWN initiates the values of the IDip = h1 (PIDi || SNi) and IDis = Φ, where (Φ) is a null value. After that, the GWN node inserts the Pi's data into the physicians' table [PIDi, IDip, IDis, and SNi]. Finally, the GWN stores [SNi, PFi, and PVi] into SCi, initiates the session counter (C0ij = 0), and returns the SCi to Pi securely;

(3) Upon receiving the SCi from the GWN, the Pi sets up the SCi and stores the R0 in the SCi.

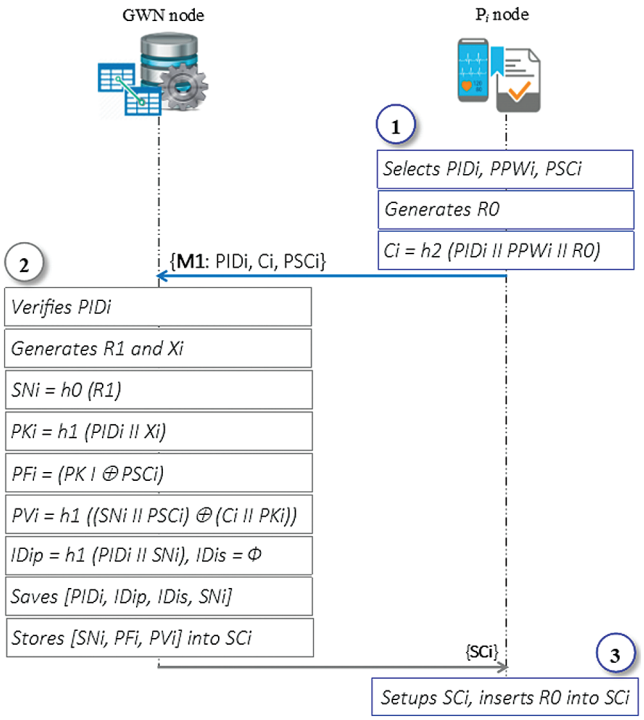

Figure 1: Physician registration stage

2.2 Patient Registration Stage

To register in the healthcare IoT system, the patient's smart device (SDj) sends a registration request message to the GWN. After that, GWN issues the smart card (SCj) to the SDj as a response to the registration request message that was sent from SDj to the GWN, as shown in Fig. 2. This procedure is summarized as follows:

(1) Using a secure channel, the SDj sends the registration message {M1: SIDj, Cj, SSCj} to the GWN. The identity number (SIDj) and security code (SSCj) are selected by the SDj. Additionally, the value of (Cj) is computed as Cj = h2 (SIDj || SPWj || R2), where both the password (SPWj) and the random number (R2) are also generated by the SDj;

(2) Upon receipt of the {M1} message, the GWN verifies whether the SIDj has registered previously. If it exists in the patients' table, the GWN rejects the {M1} message and tells the SDj to choose another SIDj. Otherwise, GWN computes SNj = h0 (R3), SN = (SSCj $\oplus$ SNj), SKj = h1 (SIDj || Xj), SFj = (SKj $\oplus$ SSCj), and SVj = h1 ((SNj || SSCj) $\oplus$ (Cj || SKj)), where the random number (R3) and a secret key (Xj) were generated by the GWN. Then, the GWN initiates both the IDj = h1 (SIDj || SNj and IDjp = IDjs = Φ, where Φ is a null value. After that, the GWN node saves the SCj's data into the patients' table [SIDj, IDj, IDjp, IDjs, and SNj]. Finally, the GWN stores [SN, SFj, and SVj] in SCj and securely returns the SCj to the SDj;

(3) Upon receiving SCj from the GWN, the SDj node sets up the SCj, stores the (R2) and securely initiates the session counter as (C1j = 0).

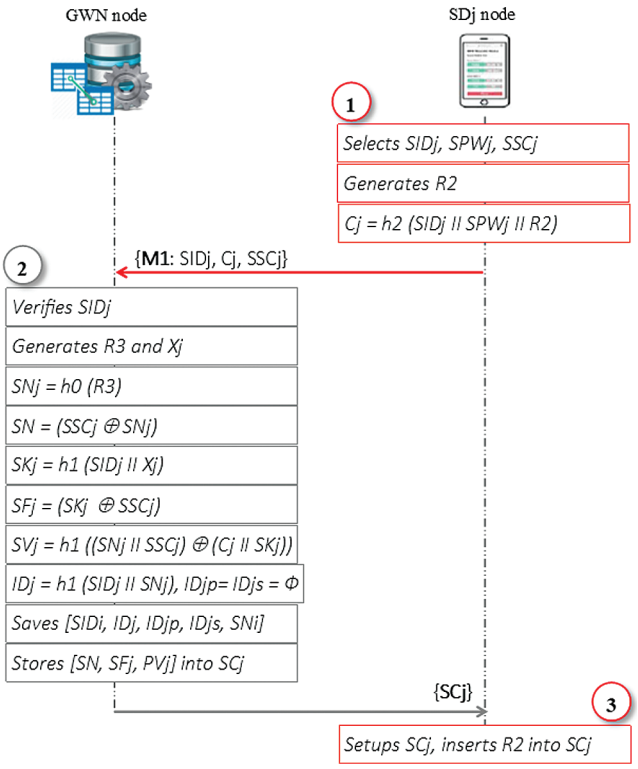

Figure 2: Patient registration stage

2.3 WMSN Node Registration Stage

To securely communicate with the SDj, the sensor node (Sk) sends a registration request message to the SDj. After that, the SDj issues a set of authentication parameters, such as the SSk1 and SNk0, as a response to the request message that was sent from Sk, as shown in Fig. 3. This procedure is summarized as follows:

(1) Using a secure channel, the Sk sends the registration request message {M1: SIDk} to the SDj node, where the SIDk value is specified, where the Sk was designed for use by a specific healthcare service provider.

(2) Upon receipt of the {M1} message, the SDj creates the session number SNk0 = (R4) randomly, initiates two sequence numbers, SSk0 = SSk1 = 0, stores Sk's record in the sensors' table as [SIDk, SSk0, and SNk0], and securely transmits {M2: SSk1, SNk0} message to the Sk;

(3) The Sk securely saves [SSk1, SNk0] into its memory.

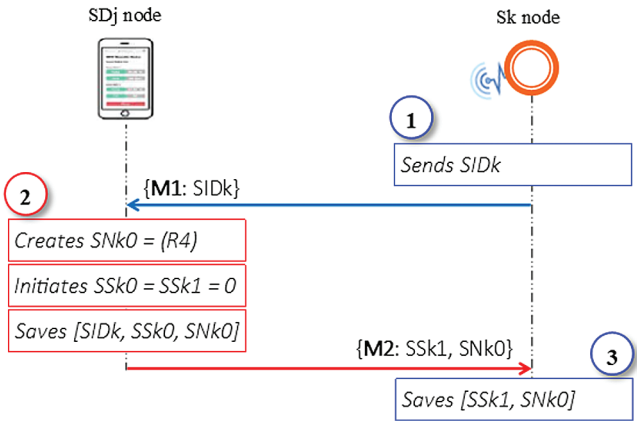

Figure 3: Sensor registration stage

2.4 Long-Term Authentication Stage

To monitor and diagnose the patient's medical status, the Pi indirectly communicates with SDj through the GWN. Therefore, the Pi needs to satisfy a mutual authentication with both GWN and SDj, as well as create the shared keys that will be used during the direct sub-authentication sessions with the SDj, as shown in Fig. 4. This procedure is summarized as follows:

(1) The Pi transmits the long-term request message {M1: IDi, CTi0, and Vi0} as a challenge message to the GWN via a public channel. The value of (IDi) was computed as IDi = h1(PIDi || SNi), while the value of (CTi0) was computed as CTi0 = ETPKi (TP0 || R5 || SIDj), where the (TP0) is a timestamp of Pi and (R5) was generated randomly. Finally, the value of (Vi0) was computed as Vi0 = h3 (TP0 || TPKi || SNi || IDi || R5), where the (TPKi) was computed as TPKi = (IDi $\oplus$ PKi) using the value of (PKi), which was extracted during the previous login physician node stage.

(2) Upon receipt of the {M1} message from the Pi, the GWN finds the values of the (IDip) and (IDis) using the value of (IDi), and operates according to the following conditions [10,17,31]:

If ((IDi = IDip) and (IDis $\neq \Phi$)), then the GWN verifies whether the Pi can diagnose the medical state of a patient remotely, by extracting the value of (SIDj) as follows: The Pi renews the (SNi) value as SNi = h0 (SNi), and calculates the PKi = h1 (PIDi || Xi) and TPKi = (IDi $\oplus$ PKi). After that, the GWN node decrypts the value of (CTi0) using <TP0 || R5 || SIDj> = DTPKi(CTi0). If the SIDj does not match, the GWN rejects

the {M1} message and ends the long-term authentication stage. Otherwise, the GWN node verifies the value of (TP0). If it does not match, the GWN rejects the {M1} message and ends the long-term authentication stage. Otherwise, GWN calculates the (XVi0) value as XVi0 = h3 (TP0 || TPKi || SNi || IDi || R5) to verify whether the received value of (Vi0) matches with the computed (XVi0) value. If it matches, the GWN renews the values of (IDis) and (IDip) as IDis = IDip, (IDip) = h1 (IDi || R5), respectively. Otherwise, the GWN rejects the {M1} message and ends the long-term authentication stage.

However, if ((IDi = IDip) and (IDis = Φ)), then the GWN verifies whether the Pi can diagnose the medical state of a patient remotely by extracting the value of (SIDj), as follows: The Pi calculates the PKi = h1 (PIDi || Xi) and TPKi = (IDi ⊕ PKi). After that, the GWN node decrypts the value of (CTi0) using the <TP0 || R5 || SIDj> = DTPKi(CTi0) function. If SIDj does not match, the GWN rejects the {M1} message and ends the long-term authentication stage. Otherwise, the GWN node also verifies the value of (TP0). If it does not match, the GWN rejects the {M1} message and ends the long-term authentication stage. Otherwise, GWN calculates the (XVi0) value as XVi0 = h3 (TP0 || TPKi || SNi || IDi || R5) to verify whether the value of (Vi0) that was received matches with the computed (XVi0) value. If it matches, the GWN renews the values of (IDis) and (IDip) as IDis = IDip and IDip = h1 (IDi ||R5), respectively. Otherwise, the GWN rejects the {M1} message and ends the long-term authentication stage.

If (IDi = IDis), then the GWN verifies whether the Pi can diagnose the medical state of a patient remotely by extracting the value of (SIDj), as follows: the Pi calculates the PKi = h1 (PIDi || Xi) and TPKi = (IDi ⊕ PKi). After that, the GWN node decrypts the value of (CTi0) using <TP0 || R5 || SIDj> = DTPKi(CTi0). If SIDj does not match, the GWN rejects the {M1} message and ends the long-term authentication stage. Otherwise, the GWN node verifies the value of (TP0). If it also does not match, the GWN rejects the {M1} message and ends the long-term authentication stage. Otherwise, GWN calculates the (XVi0) value as XVi0 = h3 (TP0 || TPKi || SNi || IDi || R5) to verify whether the received value of (Vi0) matches the computed (XVi0) value. If it matches, the GWN renews the values of (IDip) as IDip = h1 (IDi||R5). Otherwise, the GWN rejects the {M1} message and ends the long-term authentication stage.

If ((IDi ≠ IDip) and (IDi ≠ IDis)), then the GWN rejects the {M1} message and ends the long-term authentication stage;

(3) Using the (PIDi) and (SIDj) values, GWN calculates the key of the next direct sub-authentication sessions (PSij) between the Pi and SDj as PSij = h2 ((PIDi ⊕ Xi) || (SIDj ⊕ Xj) || SQij), where the value of (SQij) represents the serial number of the long-term authentication stage. Then, GWN renews the (C0j) counter as C0j = (C0j +1), the (SNj) as SNj = h0(SNj), and the (IDj) as IDj = h1(SIDj || SNj). After that, GWN transmits the long-term request message {M2: C0j, CTj0, and Vj0} to SDj through a public channel. The value of (CTj0) was computed as CTj0 = ETSKj (TGWN0 || R6 || PSij), where the value of (SKj) was computed as SKj = h1 (SIDj || Xj), the (TGWN0) is a present timestamp for GWN, and (R6) was generated randomly. The value of (Vj0) was computed as Vj0 = h3 (TGWN0 || PSij ||I IDjp || SIDj ||R6,) where the (IDjp) was previously computed as IDjp = h1 (SIDj || IDjp);

(4) Upon receipt of the {M2} message from GWN, the SDj verifies whether the (ΔCj) value falls within the (1 ≤ ΔCj ≤ μ2) range, where the value of the ΔCj is calculated as ΔCj = (C0j – C1j) and (μ2) is determined based on the system specifications. If this does not hold, the SDj rejects {M2} message and ends the long-term authentication stage. Otherwise, the SDj sets the initial value of the (SNj) as SNj = (SSCj ⊕ SN) and re-computes the SNj value using the SNj = h0 (SNj) function (ΔCj − 1) times, until the value of (ΔCj − 1) = 1. Then, the SDj sets the values of the (SN) and (IDj) as SN = (SSCj ⊕ SNj) and IDj = h1 (SIDj || SNj), respectively. After that, the SDj decrypts the (CTj0) value as DTSKj (CTj0) = <TGWN0 || R6 || PSij>, where the key (TSKj) was calculated as TSKj = (IDj ⊕ SKj). It should be noted that the value of (SKj) was calculated during the previous patient login authentication stage. The value of (TGWN0) is directly checked; if it does not hold, the SDj rejects the {M2} message and ends

the long-term authentication stage. Otherwise, the SDj sets the initial value of the (IDjs) as IDjs = IDjp and re-computes the IDjs value using the IDjs = h1 (SIDj || IDjs) function (ΔCj − 1) times, until the value of (ΔCj – 1) = 1. The SDj calculates the (XVj0) value as XVj0 = h3 (TGWN0 || PSij || IDjs || SIDj || C0j) to verify whether the value of (Vj0) that was received matches the computed (XVi0) value. If it does not match, the SDj rejects the {M2} message and ends the long-term authentication stage. Otherwise, GWN is considered to be a valid node for SDj. Then, SDj sets C1ij = C0ij. After that, SDj transmits the long-term response message {M3: IDjs, CTj1, and Vj1} as a response message to the GWN via a public channel. The value of (CTj1) was computed as CTj1 = ETSKj (TSD || R7 || C1j), where the (TSD) is a current timestamp of SDj and the (R7) was generated randomly. The value of (Vj1) was calculated as Vj1 = h3 (TSD || TSKj || PSij || IDjs || R7);

(5) Upon receipt of the {M3} message, GWN decrypts the value of (CTj1) using the <TSD || R7 || C1j> = DTSKj (CTj1) function. Then, the value of (TSD) is checked. If it does not hold, GWN rejects the {M3} message and ends the long-term authentication stage. Otherwise, the GWN calculates (XVj1) as XVj1 = h3 (TSD || TSKj || PSij || IDjs || R7) to verify whether the received value of (Vj1) matches the computed (XVj1) value. If it does not match, GWN rejects the {M3} message and ends the long-term authentication stage. Otherwise, the SDj is considered a legal node for the GWN. After that, the GWN transmits the long-term response message {M4: CTi1, and Vi1} to the Pi through a public channel. The value of (CTi1) was calculated as CTi1 = ETPKi (R8 || PSij || TGWN1), where the (R8) is a random number and (TGWN1) is a present timestamp of GWN. The (Vi1) was calculated as Vi1 = h3 (PIDi || PSij || R8 || SNi || TGWN1);

(6) Upon receipt of the {M4} message, Pi decrypts the value of (CTi1) using the <R7 || PSij || TGWN1> = DTPKi (CTi1) function. Then, the (TGWN1) is checked. If it does not hold, the Pi rejects the {M4} message and ends the long-term authentication stage. Otherwise, the GWN calculates (XVi1) as XVi1 = h3 (PIDi || PSij || R8 || SNi || TGWN1) to verify whether the value of (Vi1) that was received matches the computed (XVi1) value. If it does not match, the Pi rejects the {M4} message and ends the Long-term authentication stage. Otherwise, the pi updates the SNi = h0(SNi) and sets the IDi = IDip = h1 (IDi || R5), and the GWN is considered a legal node. After that, Pi transmits the acknowledgment message {M5: IDi, Vix} to the GWN through a public channel. The value of (Vix) was calculated as Vix = ((TP1 ⊕ TGWN1) || Vi2), where the (TP1) is a present timestamp of Pi. Additionally, the value of (Vi2) was calculated as Vi2 = h3 (PIDi || PSij || R8 || SNi || (TP1 − TGWN1));

(7) Upon receipt of the {M5} message, the GWN calculates the values of (TP1) and (ΔTP) as TP1 = ((TP1 ⊕ TGWN1) ⊕ TGWN1)) and ΔTP = (TP1 – TGWN1), respectively. Then, the value of ΔTP is checked. If it does not hold, the GWN resends the {M4} message to refresh the TGWN1 value. Otherwise, the GWN calculates (XVi2) as XVi2 = h3 (IDip || PSij || R7 || SNi || ΔTP) to verify whether the received value of (Vi2) matches the computed (XVi2) value. If it does not match, the GWN node rejects message {M5} and ends the long-term authentication stage. Otherwise, the Pi is considered a legal node. Finally, the GWN renews the values of the SNi = h0 (SNi), the IDis = Φ and the SQij = (SQij +1).

2.5 Short-Term Authentication Stage

In this stage, the Pi can directly communicate with the SDj, without reverting to the GWN. Therefore, the Pi can monitor and diagnose the medical status of the SDj based on real-time data during emergencies [20]. In this case, the Pi achieves mutual authentication with the SDj to protect the direct sub-authentication sessions from unauthorized access, as shown in Fig. 5. This procedure is summarized as follows:

(1) Pi transmits the short-term request message {M1: C0ij, CTi2, Vi3} as a challenge message to the SDj via a public channel. The value of (C0ij) was initiated as C0ij = (C0ij + 1). The value of (CTi2) was calculated as CTi2 = EPSij (TPi || R9 || C0ij), where the (TPi) is a present timestamp of Pi, and (R9) was generated randomly. It should be noted that the value (PSij) was renewed as PSij = h1(PSij || ID0ij), where the value of (ID0ij) was also renewed, as ID0ij = h1(SIDj || ID0ij). Finally, the value of (Vi3) was computed as Vi3 = h3(TPi || SIDj || PSij || ID0ij ||R9);

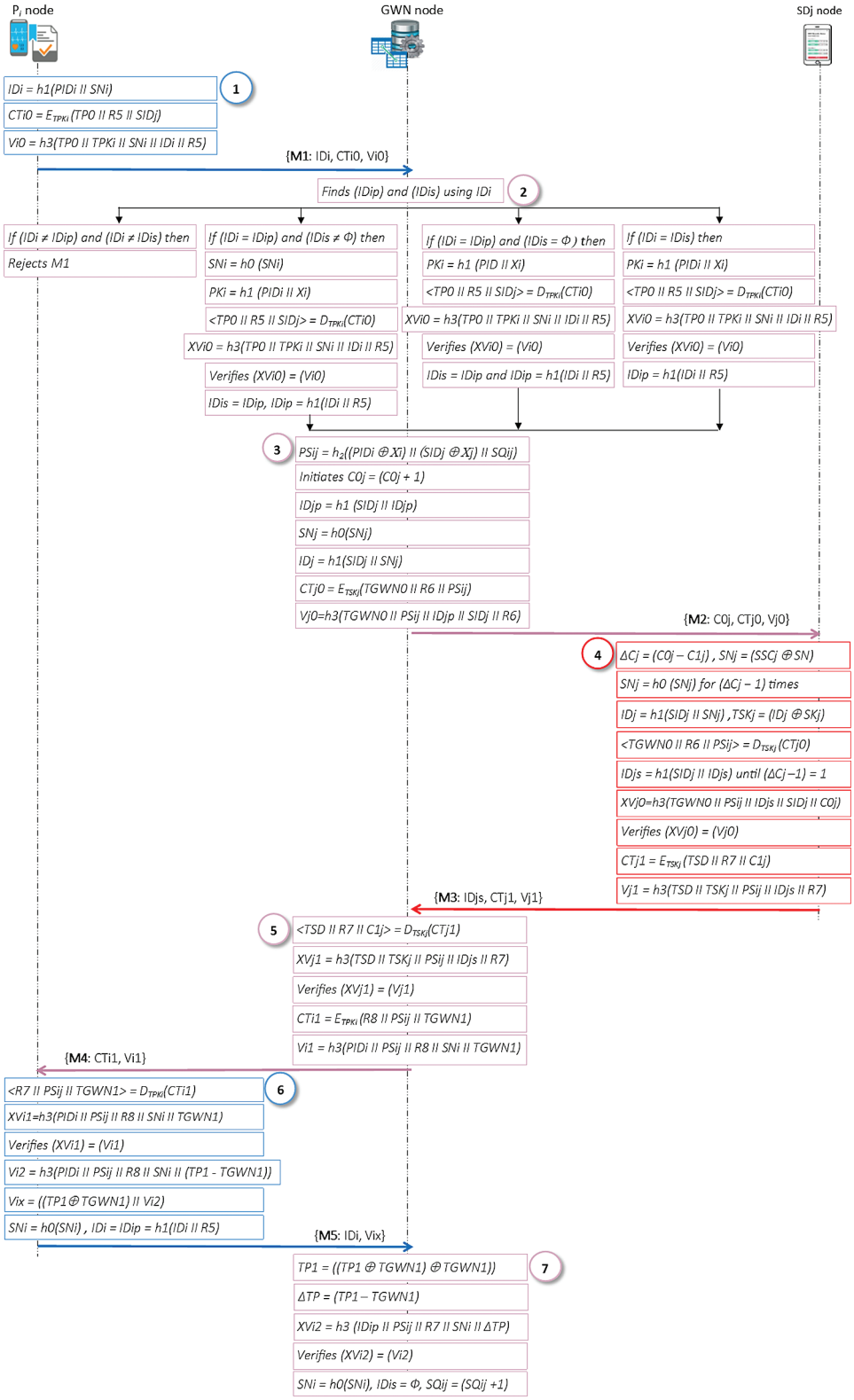

Figure 4: Long-term authentication stage

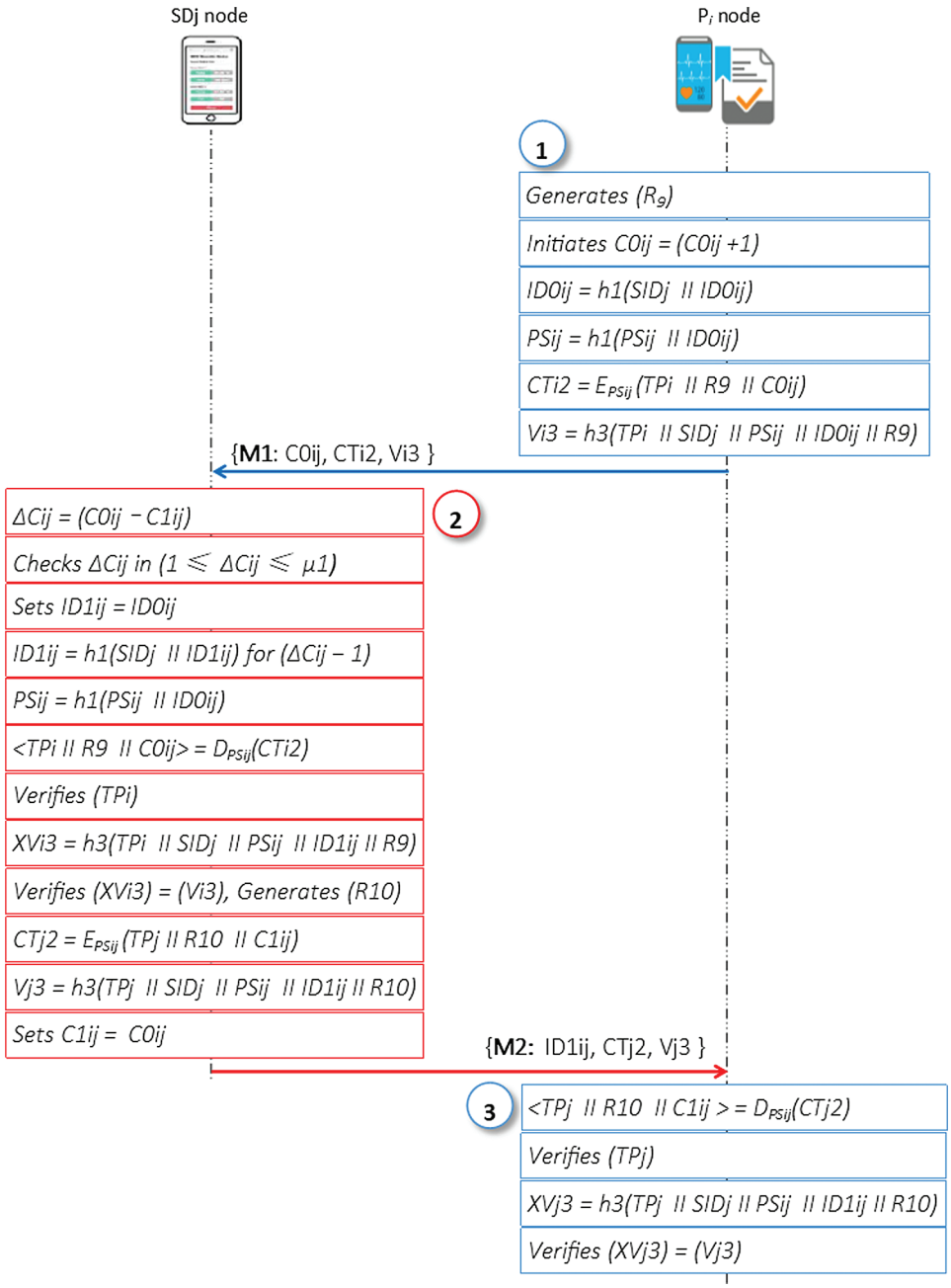

Figure 5: Short-term authentication stage

(2) Upon receipt of the {M1} message, the SDj verifies whether the (ΔCij) value falls within the (1 ≤ ΔCij ≤ μ1) range, where the value of the ΔCij is calculated using ΔCij = (C0ij – C1ij) and (μ1) was determined based on the system specifications. If it does not hold, the SDj rejects the {M1} message and ends the authentication session. Otherwise, SDj initiates the value of (ID1ij) as ID1ij = ID0ij, and then re-computes the ID1ij using the ID1ij = h1(SIDj ‖ ID1ij) function (ΔCij − 1) times, until the value of (ΔCij − 1) = 1. Then, the SDj decrypts the value of (CTi2) using the <TPi ‖ R9 ‖ C0ij> = DPSij(CTi2) function, where the value of the PSij was calculated as PSij = h1(PSij ‖ ID0ij). Next, the value of (TPi) is checked. If it does not hold, SDj rejects the {M1} message and ends the short-term authentication

stage. Otherwise, the SDj calculates the value of (XVi3) as XVi3 = h3(TPi || SIDj || PSij || ID1ij || R9) to verify whether the received value of (Vi3) matches the computed value of (XVi3). If this does not match, SDj rejects the {M1} message and ends the short-term authentication stage. Otherwise, the Pi is considered a valid node. After that, the SDj transmits the short-term response message {M2: ID1ij, CTj2, Vj3} as a response message to the Pi via public channel. The value of (CTj2) was calculated using the CTj2 = EPSij (TPj || R10 || C1ij) function, where the (TPj) is a present timestamp of the SDj and (R10) was generated randomly. The value of (Vj3) has calculated as Vj3 = h3(TPj || SIDj || PSij || ID1ij || R10);

(3) Upon receipt of the {M2} message, Pi decrypts the value of (CTj2) using the < TPj || R10 || C1ij > = DPSij (CTj2) function, where the key value of (PSij) was retrieved based on the fact of ID1ij = ID0ij. Then, the value of (TPj) is checked. If it does not hold, the Pi rejects the {M2} message and ends the short-term authentication stage. Otherwise, the Pi calculates (XVj3) as XVj3 = h3(TPj || SIDj || PSij || ID1ij || R10) to verify whether the received value of (Vj3) that matches the computed value of (XVj3). If it does not hold, Pi rejects the {M2} message and ends the short-term authentication stage. Otherwise, the SDj is considered a valid node.

2.6 WMSN Node Authentication Stage

This stage is accomplished between the Sk and the SDj to receive the medical instructions that were transmitted by the Pi, as well as to exchange the physiological data that were sensed by the Sk. Therefore, the mutual authentication procedure is performed whenever the SDj communicates with its connected Sk, as shown in Fig. 6. This procedure is summarized as follows:

(1) The SDj transmits the sensor request message {M1: CTk, Vk0, and SSk0} as a challenge message via a public channel to the Sk. The value of (CTk) has computed as CTk = ((SKk || ST) $\bigoplus$ h2(SNk0 || SIDk || SSk0)) where the (SKk) is generated randomly, the value of (SNk0) has computed as SNk0 = h1 (SNk0||SIDk), and (ST) has been specified according to the system requirements. While the value of (Vk0) has computed as Vk0 = h3(ST || SIDk || SKk || SNk0 ||SSk0). Finally, the value of (SSk0) has updated as SSk0 = SSk0 + 1. It should be noted that the value of pseudonym identity (IDk) was also renewed as IDk = h1(SKk || SIDk).

(2) Upon receipt of the {M1} message, the Sk verifies whether the (ΔSSk) value falls within the ($1 \leq \Delta SSk \leq \mu 0$) range, where the value of the ΔSSk was calculated using ΔSSk = (SSk0 − SSk1) and (μ0) was determined based on the system specifications. If it does not hold, the Sk rejects the {M1} message and ends the sensor authentication session. Otherwise, Sk initiates the value of (SNk1) as SNk1 = SNk0 and re-computes the SNk1 value using the SNk1 = h1(SNk1 || SIDk) function ΔSSk times, until the value of (ΔSSk − 1) = 1. After that, Sk calculates the value of (Vk1) as Vk1 = h3(ST || SIDk || SKk || SNk1 || SSk0 − 1), where the value of (SKk) was extracted as (SKk || ST) = CTk $\bigoplus$ h2(SNk0 || SIDk || SSk0). Next, Sk verifies whether the received value of (Vk0) matches the computed value of (Vk1). If it does not match, the Sk rejects the message {M1} and ends the sensor authentication stage. Otherwise, the Sk updates the value of (SSk1) to SSk1 = SSk0. Then, the Sk transmits the sensor response message {M2: IDk, and Vk2} as a response message to the SDj, via a public communication channel. The value of (IDk) was computed as IDk = h1(SKk || SIDk). The value of (Vk2) was computed as Vk2 = h3(ST || SIDk || SKk || SNk0 || SSk0), where the value of (SNk0) was calculated as SNk0 = h1(SNk1 || SIDk).

(3) Upon receipt of the {M2} message, the Sk computes (Vk3) as Vk3 = h3(ST || SIDk || SKk || SNk0 || SSk0), where the value (SNk0) was calculated as SNk0 = h1(SNk0 || SIDk), to verify whether the received value of (Vk2) matches the computed (Vk3) value. If it does not match, the SDj node rejects the {M2} message and terminates the sensor authentication session stage. Otherwise, the Sk node is considered a legal sensor.

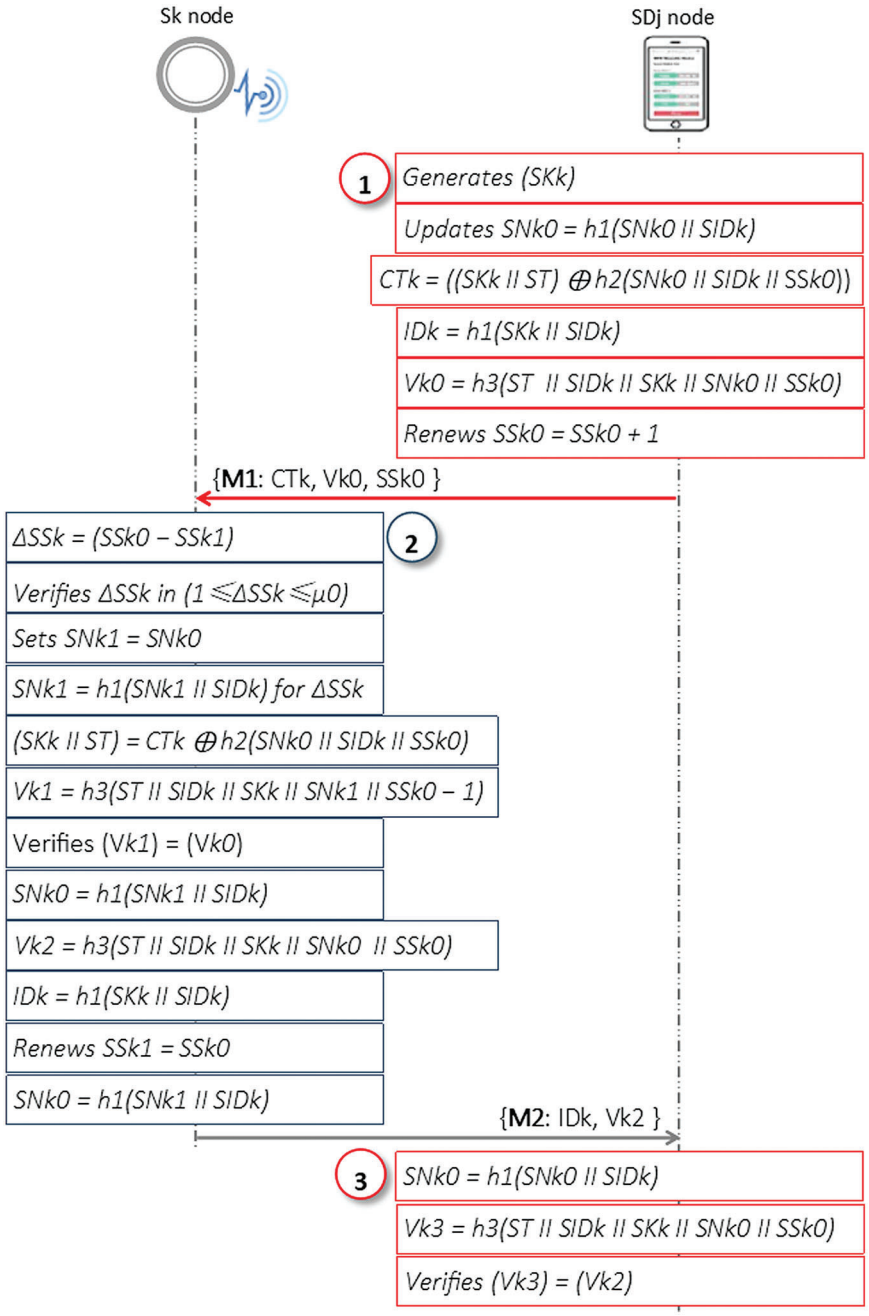

Figure 6: WMSN node authentication stage

3 Resistance Analysis of Desynchronization Attack

Usually, the pseudonym identity, one-way hash function, serial number, timestamp, and encryption techniques are used to assign and update the communication nodes' identities, as well as to generate the session keys that will be used during the next sessions in the authentication schemes. The improper use of these techniques leads to inconsistencies in the identities or the values of the shared keys between the communication nodes. Therefore, if the adversary can block the authentication messages, this may result in a desynchronization attack [26].

To illustrate how the E2EA authentication scheme can resist desynchronization attacks and preserve consistency in the shared values of the identities and keys of communication nodes during the next authentication session, a set of attack scenarios are discussed in the following subsections.

3.1 Attack Scenarios of Long-Term Authentication Stage

To ensure the long-term authentication stage can resist desynchronization attacks, suppose the following attack scenarios, as shown in Fig. 7.

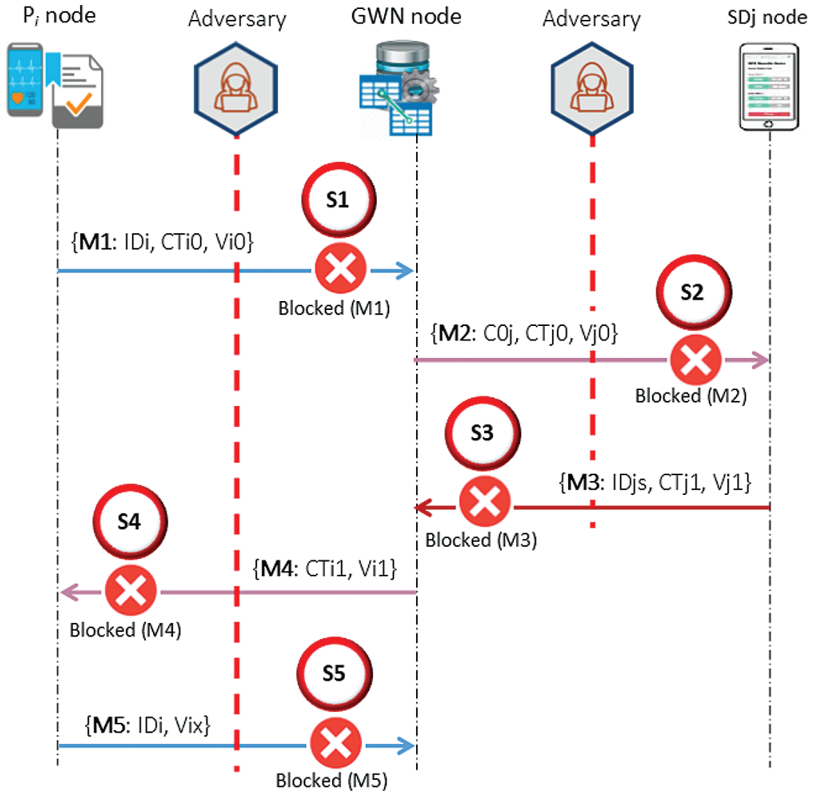

Figure 7: Desynchronization attack scenarios of the long-term authentication stage

(S1) Suppose an adversary has blocked the long-term request message {M1: IDi, CTi0, and Vi0} flow. The adversary cannot impact the synchronization between the Pi and the GWN during the next authentication session. In this scenario, the hash values of (SNi) and (IDi) have not yet been updated on both sides. However, the long-term response message (M4) has also not been received by the Pi. Therefore, the Pi can initiate a new authentication session and resend the (M1) message using different values for the (TP0) and (R5). As a result, the Pi can resynchronize the authentication session with GWN.

(S2) Suppose an adversary has blocked the long-term request message {M2: C0j, CTj0, and Vj0} flow. This scenario will not affect the synchronization between the Pi and GWN. Furthermore, the adversary cannot impact the synchronization between the GWN and the SDj during future authentication sessions. In this case, the hash values of (SNj) and (IDj) have only been updated on the GWN side. However, the long-term response message (M3) has not been received by the GWN. Therefore, the GWN can resend the {M2} message again, based on a new (C0j) value, with different values for the (TGWN0) and (R6). Alternatively, the SDj using the new value of (ΔCj) can resynchronize the one-way hash functions to

update the hash values of (SNj) and (IDj), as in the GWN side. Therefore, GWN can resynchronize the authentication session with the SDj.

(S3) Suppose an adversary has blocked the long-term response message {M3: IDjs, CTj1, and Vj1}. Such a scenario will not affect the synchronization between Pi and the GWN. This scenario is similar to the pervious scenario; the adversary also cannot impact the synchronization between the SDj and the GWN during the future authentication session. In this case, the hash values of the (SNj) and (IDj) have been updated on both sides. However, the long-term response message (M3) has not been received by the GWN. Therefore, the GWN can compute and resend the {M2} message using different values for (TGWN0), and (R6). After that, the SDj node using the new value of (ΔCj) can resynchronize the one-way hash functions to update the hash values of (SNj) and (IDj), as in the GWN side. Next, the SDj can compute and resend the {M3} message again, using the new values for the (TSD), and (R7). Therefore, the GWN can resynchronize the authentication session with SDj.

(S4) Suppose an adversary has blocked the long-term response message (M4: CTi1, Vj1) flow. This attack will not affect the synchronization between the GWN and SDj. Moreover, this attack cannot impact the synchronization between the Pi and the GWN during future authentication sessions. In this case, the hash values of (SNi) and (IDis) have not yet been updated, and only the synchronization is required for the hash value of the pseudonym identity (IDi). However, the long-term response message (M4) has not been received by the Pi. Therefore, the Pi can initiate and resend the long-term request message (M1) using different values of (TP0), and (R5). Since the value of (IDi) for the GWN is equal to (IDi = IDis), the GWN can compute and resend the long-term response message {M4} using the new values for (TGWN1) and (R8). Therefore, the Pi can resynchronize the authentication session with GWN.

(S5) Suppose an adversary has blocked the acknowledgment message {M5: IDi, Vix} flow. This scenario also will not affect the synchronization between the GWN and SDj. Furthermore, this attack cannot impact the synchronization between the Pi and GWN during future authentication sessions. In this case, the hash values of the (IDi) and (IDip) have been updated on both sides as (IDi = IDip), and the hash value of the (SNi) has been updated on the Pi side, but has not yet been updated on the GWN side. However, Pi will not receive any response notification from the GWN. Therefore, Pi can initiate a new authentication session and resend the {M1} message using different values of (TP0), and (R5). Since the value of (IDi) for the GWN node will be equal to ((IDi = IDip) and (IDis ≠ Φ)), the same session key value of (TPKi) will be computed by both the Pi and the GWN. After that, the GWN will resend (M4) to the Pi, using different values for (TGWN1) and (R8). Therefore, the Pi can resend (M5) and resynchronize the authentication session with GWN.

3.2 Attack Scenarios of Short-Term Authentication Stage

To ensure that the short-term authentication stage can resist desynchronization attacks, suppose the following attack scenarios, as shown in Fig. 8.

(S6) Suppose an adversary has blocked the short-term request message {M1: C0ij, CTi2, and Vi3} flow. In this attack, the adversary cannot impact the synchronization between the Pi and SDj during the future authentication session. In this case, the hash values of (ID0ij), and (PSij) have only been updated on the Pi side. However, the response authentication message (M2) has not been received by the Pi. Therefore, the Pi can initiate and resend the {M1} message using new values for (R9), (TPi), and (C0ij). On the other side, the SDj, using the new value of (ΔCij), can resynchronize the one-way hash functions to update the hash values of (ID1ij) and (PSij), as in the Pi side. Therefore, the Pi can resynchronize the authentication session with the SDj.

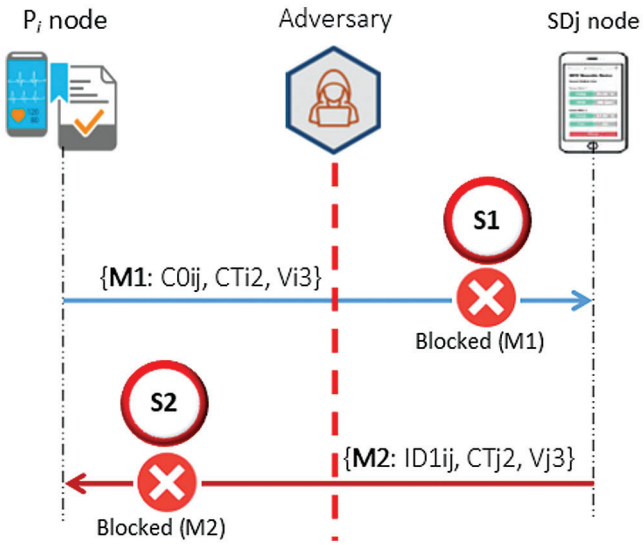

Figure 8: Desynchronization attack scenarios in the short-term authentication stage

(S7) Suppose an adversary has blocked the short-term response message {M2: ID1ij, CTj2, and Vj3} flow. This scenario is similar to the previous scenario; the adversary cannot impact the synchronization between the Pi and SDj during future authentication sessions. In this case, the hash values of (ID0ij), and (PSij) have been updated on both sides. However, the short-term response message (M2) has also not been received by the Pi. Therefore, the Pi can resend the {M1} message using new values for (R9), (TPi), and (C0ij). On the other side, the SDj, using the new value of (ΔCij), can resynchronize the one-way hash functions to update the hash values of (ID1ij) and (PSij), as shown on the Pi side. Next, the SDj can compute and resend the {M2} message, using the new values for (TPj), and (R10). Therefore, the Pi can resynchronize the authentication session with SDj.

3.3 Attack Scenarios of WMSN Node Authentication Stage

Similarly, to ensure that the WMSN node authentication stage can resist desynchronization attacks, suppose an adversary node can perform the following attack scenarios, as shown in Fig. 9.

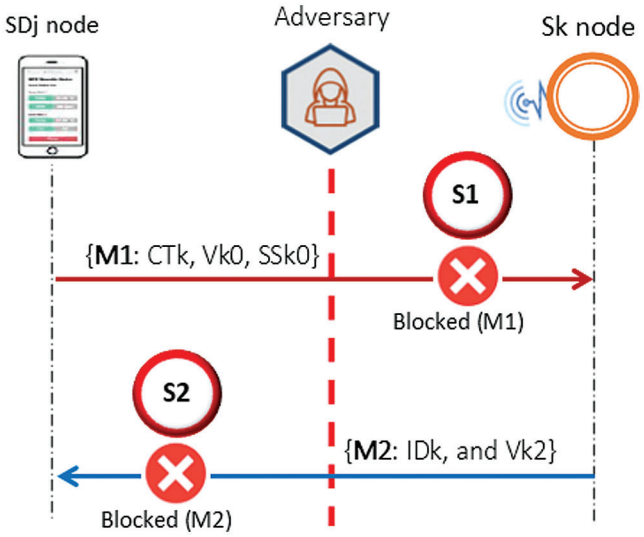

Figure 9: Desynchronization attack scenarios of the WMSN node authentication stage

(S8) Suppose an adversary has blocked the sensor request message {M1: CTk, Vk0, and SSk0} flow. In this attack, the adversary cannot impact the synchronization between the Sk and SDj during a future authentication session. In this case, the value of (CTk), the hash value of (IDk), and the hash value of (SNk0) have only been updated in the SDj side. However, the sensor response message (M2) has not been received by the SDj. Therefore, the SDj can initiate and resend the {M1} message, using new values for the (SKk) and (SSk0). On the other side, the Sk, using the new value of (ΔSSk), can resynchronize the one-way hash functions to update the hash values of the (IDk), (SSk1), and (SNk1), as in the SDj side. Therefore, the SDj can resynchronize the authentication session with Sk.

(S9) Suppose an adversary has blocked the sensor response message {M2: IDk, Vk2} flow. This scenario is somewhat similar to the case in scenario (S8). In this attack, the adversary also cannot impact the synchronization between the Sk and SDj node during future authentication sessions. In this case, the value of (CTk), the hash value of (IDk), and the hash value of (SNk0) have been updated on both sides. However, the sensor response message (M2) has not been received by the SDj. Therefore, the SDj can initiate and resend {M1}, using new values for (SKk), and (SSk0). On another side, the Sk, using the new value of (ΔSSk), can resynchronize the one-way hash functions to update the hash values of (IDk), (SSk1), and (SNk1), as shown in the SDj side. After that, the Sk can compute and resend the {M2} message again. Therefore, the SDj can resynchronize the authentication session with Sk.

4 Performance Analysis

This section offers a performance analysis of the desynchronization attack scenarios' impact on the E2EA authentication scheme in the terms of communication and computation costs. In addition, the E2EA authentication scheme is compared with a set of recently related authentication schemes that resisted the desynchronization attack [19,20]. Throughout the previous desynchronization attack scenarios, the E2EA authentication scheme showed the ability resist desynchronization attacks when executing the authentication sessions, during all authentication stages. It should be noted that the desynchronization attack may be able to make the E2EA authentication scheme suspend temporarily but cannot impact resuming the authentication sessions.

To facilitate analysis, and a reasonable comparison with the recently related authentication schemes, the performance analysis of the effect of all possible desynchronization attack scenarios will be based on the following assumptions: the size of all authentication parameters (identities, serial number, counters passwords, random numbers, timestamps, etc.) is equal to (128) bits; the block size of both encryption and decryption functions are multiples of (128) bits; the size of outputs for all hash functions is equal to (160) bits; the computation time for both encryption and decryption processes using the AES algorithm is equal (TE/D) ≅ 0.0056s [11,19,26,29]; the computation time needed to obtain all hash values using SHA-1 algorithm is equal to (Th) ≅ 0.00032s [11,19,26,29].

4.1 Computation Cost Analysis

In this section, the E2EA authentication scheme will be compared with the other proposed authentication schemes [19,20] in terms of the impact of the desynchronization attack on the computation costs. The costs are computed based on the total execution time of the cryptographic functions in each attack scenario. Tab. 2 shows the cryptographic functions that should be re-executed in each node to maintain consistency and synchronization between the communication nodes, with scenario (S5) having the highest number of re-executed functions.

Tab. 3 shows the total computation costs for the recently related authentication schemes that resist desynchronization attacks [19,20], as well as for the E2EA authentication scheme. It should be noted that scenario (S5) causes the highest computation cost among all possible attack scenarios in [19], as well as

for the E2EA scheme, while scenario (S4) represents the highest computation cost in [20]. The results indicate that the E2EA authentication scheme has lower computation costs than the authentication scheme in [19], which used both one-way hash and cryptographic functions in attack scenarios (S1-S5). However, the computation costs of E2EA authentication are higher than the computation costs of the authentication scheme in [20], which only uses one-way hash functions in the attack scenarios (S1-S4).

Table 2: Total cryptographic functions and their computation costs for each attack scenario

Scenario	Pi node	GWN node	SDj node	Sk node	Total cryptographic functions
S1	TE/D + Th	–	–	–	TE/D + Th
S2	–	TE/D + 4Th	2Th	–	TE/D + 6Th
S3	–	TE/D + 4Th	2TE/D + 5Th	–	3TE/D + 9Th
S4	TE/D + Th	2TE/D + 4Th	N/A	–	3TE/D + 5Th
S5	2TE/D + 5Th	2TE/D + 5Th	N/A	–	4TE/D + 10Th
S6	TE/D + 3Th	–	N/A	–	TE/D + 3Th
S7	TE/D + 3Th	–	2TE/D + 5Th	–	3TE/D + 8Th
S8	–	–	4Th	–	4Th
S9	–	–	4Th	8Th	12Th

Table 3: The comparisons of total computation costs for each attack scenario

Scenario	[19]	[20]	E2EA
S1	0.01998s	0.00128s	0.00592s
S2	0.02254s	0.00320s	0.00752s
S3	0.02478s	0.00512s	0.01968s
S4	0.02638s	0.00896	0.01840s
S5	0.02798s	N/A	0.02560s
S6	N/A	N/A	0.00656s
S7	N/A	N/A	0.01006s
S8	N/A	N/A	0.00128s
S9	N/A	N/A	0.00288s

4.2 Communication Costs Analysis

In this section, the E2EA authentication scheme will be compared with the recently related authentication schemes [19,20] in terms of the desynchronization attack's impact on the communication costs. These costs are computed based on the total bit size of the authentication messages in each attack scenario. Tab. 4 shows the total bit size of the authentication messages that should be retransmitted to maintain consistency and synchronization between the communication nodes. The results indicate that the authentication messages that would be transmitted to resist scenario (S5) have the highest communication costs.

Table 4: The total size of the retransmitted authentication messages for each attack scenario

Scenario	Stage[1]					Stage[2]		Stage[3]		Total/bits
	M1	M2	M3	M4	M5	M1	M2	M1	M2	
S1	672	–	–	–	–	–	–	–	–	672
S2	–	672	–	–	–	–	–	–	–	672
S3	–	672	672	–	–	–	–	–	–	1344
S4	672	–	–	544	–	–	–	–	–	1216
S5	672	–	–	544	288	–	–	–	–	1504
S6	–	–	–	–	–	672	–	–	–	672
S7	–	–	–	–	–	672	672	–	–	1344
S8	–	–	–	–	–	–	–	448	–	448
S9	–	–	–	–	–	–	–	448	288	736

Notes:[1]Long-term authentication stage, [2]Short-term authentication stage, [3]WMSN authentication phase.

Tab. 5 shows the total communication costs for the recent-related authentication schemes that are resisted the desynchronization attack [19,20] as well as for the E2EA authentication scheme. It should be noted that scenario (S5) leads to the highest communication cost among all the possible attack scenarios in [19], as well as for the E2EA scheme, causing five authentication messages to be resent, while scenario (S4) represents the highest communication cost in [20], in which four authentication messages are sent. The results indicate that the E2EA authentication scheme has the minimum required communication costs.

Table 5: The total communication costs for each attack scenario

Scenario	[19]	[20]	E2EA
S1	704 bits	768 bits	672 bits
S2	1504 bits	1728 bits	672 bits
S3	1952 bits	2336 bits	1344 bits
S4	2272 bits	3136 bits	1216 bits
S5	2560 bits	N/A	1504 bits
S6	N/A	N/A	672 bits
S7	N/A	N/A	1344 bits
S8	N/A	N/A	448 bits
S9	N/A	N/A	736 bits

5 Conclusion

This paper eliminates the ambiguity in the behavior of the end-to-end authentication scheme for healthcare IoT systems (E2EA) when resisting desynchronization attacks. The performance of the authentication scheme is considered critical to evaluating the applicability of such schemes, along with the security services that can be achieved. Therefore, this paper discusses how the E2EA authentication

scheme can resist desynchronization attacks, looking at all possible attack scenarios in all authentication stages. Furthermore, the effect of desynchronization attacks on E2EA authentication scheme performance is analyzed based on a comparison with recently related authentication schemes. Finally, this paper finds that the E2EA authentication scheme not only prevents desynchronization attacks, it also offers low computation and communication costs to maintain consistency and synchronization between the system's communication nodes in future authentication sessions.

Acknowledgement: The author would like to express his gratitude to all the members of the Computer and Information Sciences College at Jouf University for their support.

Funding Statement: The author received no specific funding for this study.

Conflicts of Interest: The author declares that has no conflicts of interest to report regarding the present study.

References

[1] S. R. Patil, D. R. Gawade and S. N. Divekar, "Remote wireless patient monitoring system," *Int. Journal of Electronics & Communication Technology (IJECT)*, vol. 6, no. 1, pp. 9–13, 2015.

[2] C. Assaba and S. Gite, "IOT based health care remote monitoring and context-aware appointment system," *Int. Journal of Current Engineering and Technology*, vol. 7, no. 6, pp. 2347–5161, 2017.

[3] M. A. Uddin, A. Stranieri, I. Gondal and V. Balasubramanian, "Continuous patient monitoring with a patient centric agent: A block architecture," *IEEE Access*, vol. 6, pp. 32700–32726, 2018.

[4] P. H. Waghmare and A. N. Bhute, "Healthcare monitoring system using smartphone," *Int. Journal of Innovative Research in Science (IJIRSET)*, vol. 6, no. 6, pp. 12407–12413, 2017.

[5] M. M. Janet and R. Dharmalingam, "Enhanced IoT system in healthcare application using wireless body sensor networks," *Int. Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)*, vol. 24, no. 2, pp. 6–9, 2017.

[6] M. D. Babakerkhell and N. Pandey, "Analysis of different IoT based healthcare monitoring systems," *Int. Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 6S2, pp. 61, 2019.

[7] A. Hamarsheh, Y. Abdalaziz and S. Nashwan, "Recent impediments in deploying IPv6," *Advances in Science, Technology and Engineering Systems Journal (ASTES)*, vol. 6, no. 1, pp. 336–341, 2021.

[8] G. Farzaneh and A. Rahnamaei, "Authentication in health care application using wireless medical sensor network: A survey," *Int. Journal of Research in Computer Applications and Robotics (IJRCAR)*, vol. 4, no. 4, pp. 59–69, 2016.

[9] K. Dhakal, A. Alsadoon, P. W. Prasad, R. S. Ali, L. Pham *et al.,* "A novel solution for a wireless body sensor network: Telehealth elderly people monitoring," *Egyptian Informatics Journal*, vol. 21, no. 2, pp. 91–103, 2020.

[10] A. Al-Qerem, F. Kharbat, S. Nashwan, S. Ashraf and K. Blaou, "General model for best feature extraction of EEG using discrete wavelet transform wavelet family and differential evolution," *Int. Journal of Distributed Sensor Networks*, vol. 16, no. 3, pp. 1–21, 2020.

[11] S. Nashwan, "AAA-WSN: Anonymous access authentication scheme for wireless sensor networks in big data environment," *Egyptian Informatics Journal*, vol. 22, no. 1, pp. 15–26, 2021.

[12] P. Kumar, S. Lee and J. Lee, "E-SAP: Efficient-strong authentication protocol for healthcare applications using wireless medical sensor networks," *Sensors*, vol. 12, no. 2, pp. 1625–1647, 2012.

[13] D. He, K. Kumar, J. Chen, C. Lee, N. Chilamkurti *et al.,* "Robust anonymous authentication protocol for healthcare applications using wireless medical sensor networks," *Multimedia Systems*, vol. 21, no. 1, pp. 49–60, 2015.

[14] X. Li, J. Niu, S. Kumari, J. Liao, W. Liang *et al.,* "A new authentication protocol for healthcare applications using wireless medical sensor networks with user anonymity," *Security and Communication Networks*, vol. 9, no. 15, pp. 2643–2655, 2016.

[15] J. Srinivas, D. Mishra and S. Mukhopadhyay, "A mutual authentication framework for wireless medical sensor networks," *Journal of Medical Systems*, vol. 41, no. 5, pp. 857, 2017.

[16] F. Wu, X. Li, A. K. Sangaiah, L. Xu, S. Kumari *et al.,* "A lightweight and robust two-factor authentication scheme for personalized healthcare systems using wireless medical sensor networks," *Future Generation Computer Systems*, vol. 82, no. 1, pp. 727–737, 2018.

[17] R. Amin, S. H. Islam, G. P. Biswas, M. K. Khan and N. Kumar, "A robust and anonymous patient monitoring system using wireless medical sensor networks," *Future Generation Computer Systems*, vol. 80, no. 4, pp. 483–495, 2018.

[18] R. Ali, A. K. Pal, S. Kumari, A. K. Sangaiah, X. Li *et al.,* "An enhanced three factor based authentication protocol using wireless medical sensor networks for healthcare monitoring," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 1, pp. 74, 2018.

[19] M. Shuai, B. Liu, N. Yu and X. Xiong, "Lightweight and secure three-factor authentication scheme for remote patient monitoring using on-body wireless networks," *Security and Communication Networks*, vol. 2019, no. 12, pp. 1–14, 2019.

[20] M. Fotouhi, M. Bayat, A. K. Das, H. A. Far, S. M. Pournaghi *et al.,* "A lightweight and secure two-factor authentication scheme for wireless body area networks in health-care IoT," *Computer Networks*, vol. 177, no. 1, pp. 107333, 2020.

[21] S. Nashwan, "SAK-AKA: A secure anonymity key of authentication and key agreement protocol for LTE network," *Int. Arab Journal of Information Technology (IAJIT)*, vol. 14, no. 5, pp. 790–801, 2017.

[22] S. Nashwan, "Secure authentication protocol for NFC mobile payment systems," *Int. Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 8, pp. 256–263, 2017.

[23] S. Nashwan, "Synchronous authentication key management scheme for Inter-eNB handover over LTE networks," *Int. Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 8, pp. 100–107, 2017.

[24] M. Al-Fayoumi and S. Nashwan, "Performance analysis of SAP-NFC protocol," *Int. Journal of Communication Networks and Information Security (IJCNIS)*, vol. 10, no. 1, pp. 125–130, 2018.

[25] S. Nashwan, "SE-H: Secure and efficient hash protocol for RFID system," *Int. Journal of Communication Networks and Information Security (IJCNIS)*, vol. 9, no. 3, pp. 358–366, 2017.

[26] S. Nashwan, "An end-to-end authentication scheme for healthcare IoT systems using WMSN," *Computers, Materials & Continua*, vol. 68, no. 1, pp. 607–642, 2021.

[27] N. Almrezeq, L. Almadhoor, T. Alrasheed, A. A. Abd El-Aziz, S. Nashwan *et al.,* "Design a secure IoT architecture using smart wireless networks," *Int. Journal of Communication Networks and Information Security (IJCNIS)*, vol. 12, no. 3, pp. 401–410, 2020.

[28] S. Nashwan and B. Alshammari, "Formal analysis of MCAP protocol against replay attack," *British Journal of Mathematics & Computer Science (BJMCS)*, vol. 22, no. 1, pp. 1–14, 2017.

[29] Y. Lu, L. Li, H. Peng and Y. Yang, "An energy efficient mutual authentication and key agreement scheme preserving anonymity for wireless sensor networks," *Sensors*, vol. 16, no. 6, pp. 837, 2016.

[30] L. Xiong, T. Peng, H. Liang and Z. Liu, "A lightweight anonymous authentication protocol with perfect forward secrecy for wireless sensor networks," *Sensors*, vol. 17, no. 11, pp. 2681, 2017.

[31] M. Wazid, A. K. Das and A. V. Vasilakos, "Authenticated key management protocol for cloud-assisted body area sensor networks," *Journal of Network and Computer Applications*, vol. 123, no. 2, pp. 112–126, 2018.

[32] Y. Chen, Y. Ge, Y. Wang and Z. Zeng, "An improved three-factor user authentication and key agreement scheme for wireless medical sensor networks," *IEEE Access*, vol. 7, pp. 85440–85451, 2019.

[33] M. K. Hasan, M. Shahjalal, M. Z. Chowdhury and Y. M. Jang, "Real-time healthcare data transmission for remote patient monitoring in patch-based hybrid OCC/BLE networks," *Sensors*, vol. 19, no. 5, pp. 1208, 2019.