Indonesian Journal of Electrical Engineering and Computer Science
Vol. 24, No. 1, October 2021, pp. 621~636
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v24.i1.pp621-636

Formal security analysis of lightweight authenticated key agreement protocol for IoT in cloud computing

Ahmed H. Aly[1], Atef Ghalwash[2], Mona M. Nasr[3], Ahmed A. Abd El-Hafez[4]
[1,2,3]Faculty of Computer Science and Artificial Intelligence, Helwan University, Cairo, Egypt
[4]National Telecom, Regulatory Authority (NTRA), Cairo, Egypt

Article Info

Article history:

Received Jun 24, 2021
Revised Aug 29, 2021
Accepted Aug 30, 2021

Keywords:

AVISPA
Formal verification
Internet security protocol
Internet of things
Lightweight authentication
Strand space model

ABSTRACT

The internet of things (IoT) and cloud computing are evolving technologies in the information technology field. Merging the pervasive IoT technology with cloud computing is an innovative solution for better analytics and decision-making. Deployed IoT devices offload different types of data to the cloud, while cloud computing converges the infrastructure, links up the servers, analyzes information obtained from the IoT devices, reinforces processing power, and offers huge storage capacity. However, this merging is prone to various cyber threats that affect the IoT-Cloud environment. Mutual authentication is considered as the forefront mechanism for cyber-attacks as the IoT-Cloud participants have to ensure the authenticity of each other and generate a session key for securing the exchanged traffic. While designing these mechanisms, the constrained nature of the IoT devices must be taken into consideration. We proposed a novel lightweight protocol (Light-AHAKA) for authenticating IoT-Cloud elements and establishing a key agreement for encrypting the exchanged sensitive data was proposed. In this paper, the formal verification of (Light-AHAKA) was presented to prove and verify the correctness of our proposed protocol to ensure that the protocol is free from design flaws before the deployment phase. The verification is performed based on two different approaches, the strand space model and the automated validation of internet security protocols and applications (AVISPA) tool.

Corresponding Author:

Ahmed Hassan Aly
Faculty of Computer Science and Artificial Intelligence
Helwan University, Cairo, Egypt
Email: ahmed71.aly@gmail.com

Nomenclatures

C	Bundle in strand space model
$E(.)$	Symmetric lightweight encryption
F	Dobbertain function
ID_{IoT}	IoT identity
ID_S	Authentication server identity
ID_U	User identity
$H(.)$	A lightweight collision-free one-way hash function
PW_{IoT}, PW_U	The password of the IoT device, user respectively
PSK_{IoT}	The pre-shared key between the user and the IoT device
PSK_U	The pre-shared key between the user and the server
R_S, R_U, R_{IoT}	Random numbers of server, user, IoT device respectively
T_S, T_U, T_{IoT}	Time-stamps of server, user, IoT device respectively
ΔT	Time range allowed for delay
i	Index selected from R_U – ex: i= decimal value of i^{th} byte of R_U
j	Index selected from R_{IoT} – ex: j= decimal value of j^{th} byte of R_{IoT}

$\|\|$	Concatenation
$\oplus$	XoR Operation

Abbreviations

HMAC	Hash message authentication code
LFSR	Linear feed- back shift register
NFSR	Non-linear feed-back shift register
SK	Session key

1. INTRODUCTION

The internet of things (IoT) has become an emerging technology in recent years, the number of connected devices is anticipated to reach 75 billion devices by the end of 2025 [1]. IoT technology covers different applications, smart cities, healthcare, and smart traffic. Based on the application used, IoT devices send a plethora of data, real-time videos, and photos to the servers. The traditional platforms storage systems and computing platforms can not effectively handle the massive data generated by deployed IoT devices. As a result, imperative attention is required to find suitable mechanisms that rely on large resource pools such as cloud computing to handle the offloaded data. In many use cases, authorized users need to access real-time data directly from the IoT devices for instant and critical decisions (i.e., Healthcare, Fire ignition, and Traffic Congestion) [3]. In this use case, cloud servers are responsible for the mutual authentication of the user and the intended IoT device before granting authorized users the right to access the real-time data. Due to the constrained nature of IoT devices, these devices are susceptible to different vulnerabilities and security breaches [4], [5]. Meanwhile, hackers are developing new methods to exploit poor security mechanisms implemented in IoT devices. A robust authentication protocol is substantial to prevent unauthorized access, protect sensitive data, and maintain user privacy.

In this context, we proposed a new lightweight authentication and key agreement protocol (Light-AHAKA) [6]. Our proposed protocol is based on the challenge-response mechanism to achieve mutual authentication, taking into consideration the constrained nature of IoT devices. The cryptographic functions used are symmetric-key cryptography, hash function, and hash message authentication code (HMAC) [7]. Every session, the protocol generates a new key for encrypting the traffic. A different session key for each session allows a limited number of messages to be encrypted with one session key, making it very difficult for attackers to find the generated session keys. Moreover, if the attacker succeeds in finding the session key by any means, this key is related to a specific session and has nothing to do with the upcoming sessions. For enhancing the resiliency of the IoT-Cloud network, the (Light-AHAKA) updates the pre-shared keys, passwords, and participant identities. All the previous parameters are valid only for one session, making the breakthrough to our protocol very difficult.

The design and formal analysis of security protocols is a challenging problem. Serious security flaws in protocols were discovered in several cases, many years after they were first published or deployed. Attacks on these protocols generally avoid targeting the mathematical cryptographic primitives, but rather focus on exploiting the protocol's design flaws. Formal verification provides rigorous and thorough methods of evaluating the correctness of the security protocols to discover subtle flaws. Researchers have developed numerous approaches and formal methods that could be utilized for the verification of security protocols. In this context, we present the formal verification of (Light-AHAKA) to verify the correctness of our proposed protocol. The formal verification is conducted based on two different approaches, the strand space model and AVISPA.

The upcoming sections of the paper are structured in the following way. Section 2 of the paper reviews the (Light-AHAKA) protocol. The formal analysis of (Light-AHAKA) is introduced thoroughly in section 3 based on two different approaches. Section 4, discusses the results of the formal analysis of the (Light-AHAKA). Finally, section 5 concludes the paper.

2. REVIEW OF (LIGHT-AHAKA)

The (Light-AHAKA) is a lightweight authentication protocol based on lightweight symmetric key cryptography, lightweight hash function, lightweight hash-based message authentication code (HMAC) [8], and exclusive-or operation. A review of the (Light-AHAKA) will be illustrated in the upcoming steps. The IoT-Cloud network will be initialized according to the following:

- The network administrator (*NA*) assigns a unique identity (ID_{Ui}), password (PW_{Ui}), and a pre-shared key (PSK_{Ui}) for each *user* (U_i) stored safely in the client application.
- The (*NA*) loads (ID_{IoTi}), (PW_{IoTi}), and pre-shared key (PSK_{IoTi}) in a tamper-proof memory for each *IoT* device.

The (Light-AHAKA) procedure Figure 1 will be reviewed in the following steps:

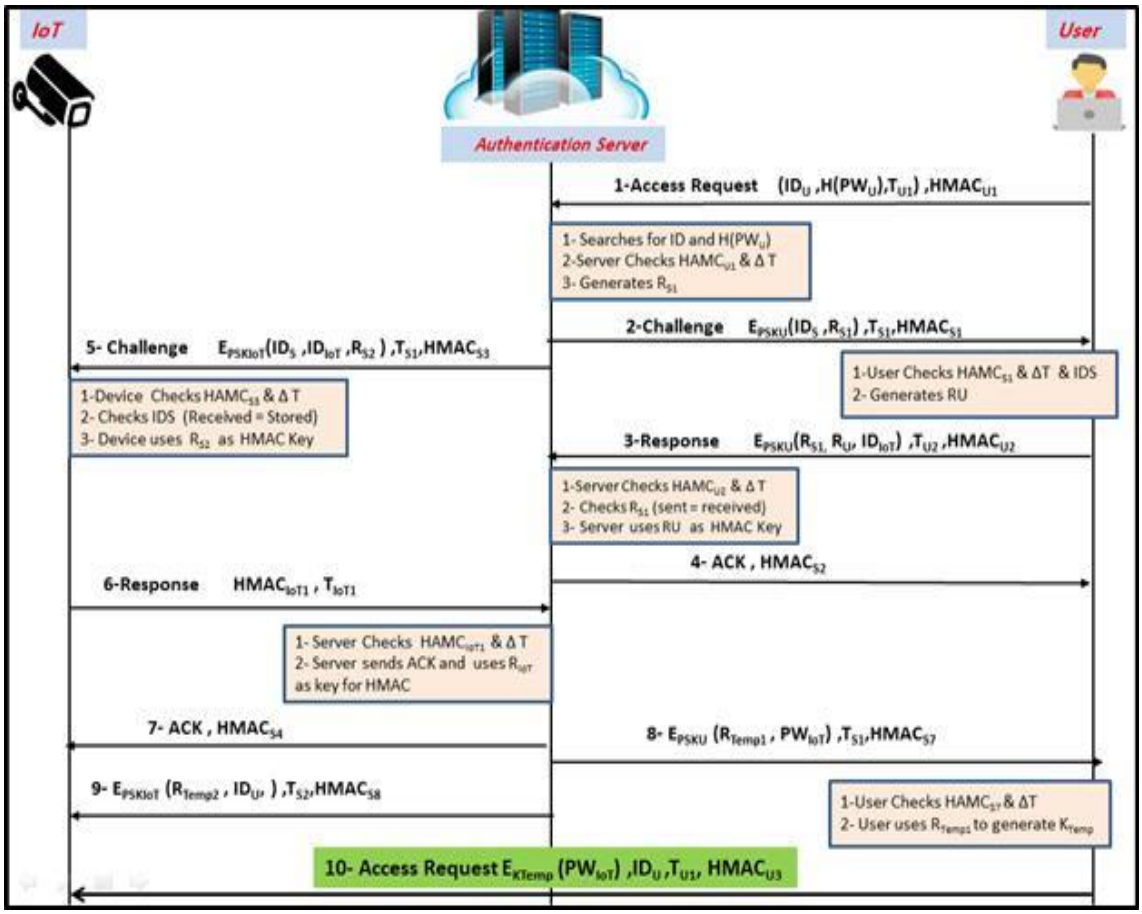

Figure 1. Light-AHAKA procedure

1) The *user* sends an access request to the *authentication server*:
   - $(ID_U \| H(PW_U) \| T_{U1}) \| (HMAC_{U1})$.
2) The *authentication server* will do the following:
   - Checks if ($T_{U1} < \Delta T$), calculates $(HMAC_{U1})$, and compares it with the received one.
   - Searches in the database for (ID_U) and the hash of the password.
   - Generates a random number (R_{S1}).
   - Sends the following message encrypted with the pre-shared key (PSK_U) as a challenge for the *user*:
     - $E_{PSK_U} (ID_S \| R_{S1}) \| T_{S1} \| (HMAC_{S1})$.
3) The *user* receives the message and will do the following:
   - Checks if ($T_{S1} < \Delta T$) and verifies $(HMAC_{S1})$.
   - Decrypts the message and checks ID_S (received) $= ID_S$ (stored).
   - Extracts (R_{S1}).
   - From the previous steps, the *user* authenticates the *server*.
   - The user will generate a random number (R_U) and send the following message as a response to the *server*:
     - $E_{PSK_U} (R_{S1} \| R_U \| ID_{IoT}) \| T_{U2} \| (HMAC_{U2})$.
4) The *authentication server* receives the message and will do the following:
   - Checks if ($T_{U2} < \Delta T$) and verifies $(HMAC_{U2})$.
   - Decrypts the message and checks if R_{S1} (sent) = R_{S1} (received).
   - From the previous step, the *server* authenticates the *user*.
   - Stores R_U and searches for the (ID_{IoT}).
   - The *server* sends the *user* acknowledgment message and the *HMAC* of the acknowledgment using (R_U) as a key for the *HMAC* as a response for the *user*:
     - *ACK* || *HMACS2*
   - The *user* receives the messages and calculates the *HMAC* using (R_U) as a key and compares the result with the received one.

5) The *server* will start the mutual authentication with the intended *IoT* device:
   - The *server* generates a random number (R_{S2}) and sends a challenge message to the intended *IoT* device:
     - $E_{PSK_{IoT}} (ID_S || ID_{IoT} || R_{S2}) || T_{S1} || HMAC_{S3}$
   - The *IoT* receives the message, checks if ($T_{S1} < \Delta T$), and verifies ($HMAC_{S3}$).
   - Decrypts the message and checks if ID_S (received) = ID_S (Stored).
   - From the previous steps, the IoT device authenticates the server.
   - The IoT device calculates the response message for the challenge of the server:
     - $HMAC_{IoT1} (T_{IoT1} \oplus R_{S2} \oplus PW_{IoT})$.
   - Calculates the nonce R_{IoT} as follows:
     - $R_{IoT} = Hash (R_{S2} \oplus ID_S \oplus ID_{IoT})$
6) The *IoT* device sends the following response message to the server:
   - $(HMAC_{IoT1} || T_{IoT1})$.
   - The *server* receives the message, checks if ($T_{IoT1} < \Delta$ T), and verifies $(HMAC_{IoT1})$.
   - Calculates R_{IoT} as follows:
     - $R_{IoT} = Hash (R_{S2} \oplus ID_S \oplus ID_{IoT})$

     From the previous steps, the *server* authenticates the *IoT* device.
7) The *server* sends the *IoT* device acknowledgment message and the *HMAC* of the acknowledgment using (R_{IoT}) as a key for the *HMAC* as a response for the *IoT* device:
   - $ACK || HMAC_{S4}$
8) After the *authentication server* has finished authenticating the *user* and the *IoT* device, the server sends the *user* and the *IoT* device temporary parameters to establish a temporary key. The *user* will make use of this key to access the *IoT* device. In the initial stages of designing (Light-AHAKA), it was designed to make the authentication server send K_{IoT} to the user, but after studying the Strand Space Model presented in section 4.1, we found that to achieve maximum security, it is not recommended to transmit pre-shared keys.
9) The *server* sends to the *user* the following data:
   - $R_{Temp1} = (R_{IoT} \oplus ID_S)$
   - *IoT* device password (PW_{IoT}), and R_{Temp1}
     - $E_{PSK_U} (R_{Temp1} || PW_{IoT}) || T_{S1} || HMAC_{S5}$
   - The *user* receives the message, checks if ($T_{S1} < \Delta T$), and verify $HMAC_{S5}$.
   - Calculates $R_{IoT} = R_{Temp1} \oplus ID_S$.
   - Calculates $K_{Temp1} = R_{IoT} \oplus R_U$.
10) The *server* sends to the *IoT* device the following data:
   - $R_{Temp2} = R_U \oplus ID_S$.
   - R_{Temp2} and *user* ID encrypted with *IoT* pre-shared key (PSK_{IoT}) as follows:
     - $E_{PSK_{IoT}} (R_{Temp2} || ID_U) || T_{S2} || HMAC_{S6}$
   - The *IoT* device receives the message, checks if ($T_{S2} < \Delta$ T), and verifies $HMAC_{S6}$.
   - Calculates $R_U = R_{Temp2} \oplus ID_S$
   - Stores (R_U) and (ID_U).
   - Calculates $K_{Temp1} = R_{IoT} \oplus R_U$.
11) The *user* sends an access request to the *IoT* device as follows:
   - $E_{K_{Temp1}} (PW_{IoT}) || ID_U || T_{U1} || MAC_{U3}$
   - The *IoT* device receives the message, checks if ($T_{U1} < \Delta T$), and Verifies $HMAC_{U3}$.
   - Checks if ID_U (received) = ID_U (stored).
   - Decrypts the message using (K_{Temp1}).
   - Extracts the password and checks that it is a valid password.
   - Sends acknowledgment to the *user*.
12) The (Light-AHAKA) participants (*server- IoT* device *-user*) store two shift registers. The first shift register is a linear feedback shift register (*L1*) connected to a primitive polynomial to produce a well-balanced sequence of streams. The second is a non-linear feedback shift register (*L2*) connected to a non-linear Boolean function to increase the non-linearity of the output sequence. The output of the two registers is connected to a vectorial dobbertain function (*F*) as a combiner function [9], which is an almost perfect non-linear function characterized by high resistance to linear and differential attacks.
13) The *IoT* device and the *user* will do the following for generating the session key:
   - IV_1=Hash (R_{IoT}).
   - IV_2=Hash (R_U).

– *IV1* and *IV2* will be used to fill (*L1*) and (*L2*), respectively.

The session key (*SK*) will be calculated using the key agreement and parameters update module as shown in Figure 2.

- *SK= F [Shift (L1(i)), Shift (L2(j))]*

– As an example, let's assume that *i* is predetermined as the 20th byte of *RU*, and the decimal value of *i* is (126), so the *L1* will be shifted (126) shifts.

14) The passwords and the identities of (*User- IoT*) will be updated every session based upon the method illustrated in step (13). More details about (Light-AHAKA) are provided in [1].

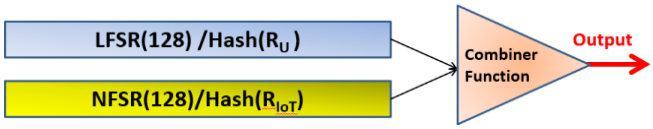

Figure 1. Key agreement and parameters update module

3. FORMAL SECURITY ANALYSIS OF (LIGHT-AHAKA)

In this section, we review the security analysis of (Light-AHAKA), which is summarized in Table 1. We also present the formal analysis of the protocol based on two different approaches, the Strand Space Model and AVISPA. Table 2 presents the adversary capabilities which are based on the Dolev-Yao model [10] and the Canetti-Krawczyk threat model [11].

Table 1. Security analysis of light-AHAKA

Security attack	Countermeasures
Impersonation attacks	Updating the following parameters every session: 1- (User-IoT device) passwords. 2- (User-IoT) Identities. 3- Pre-shared Keys. 4- Random Numbers
Privileged insider attacks	1- Storing the password Hashed 2- Sending the hash of the password
Man in the middle attacks (MITM)	1- Encrypting all challenges and responses 2- Fresh Random Numbers 3- Using Timestamps
DOS Attacks	Updating the following parameters every session: 1- (User-IoT device) passwords 2- (User-IoT) Identities 3- Pre-shared Keys
Replay Attacks	1- Fresh Random Numbers 2- Using Timestamps
Offline guessing attacks	1- Sending the password hashed 2- Updating the password every session
Data integrity attacks	1- Using HMAC function 2- Using fresh random numbers
Parallel session attack	1- Using the hash function 2- Using the HMAC function
Session key discloser attack	Using the following: 1- Fresh Random Numbers (RU - RIoT) 2- Collision-free hash function 3- The Primitive Polynomial 4- The Nonlinear Boolean Function 5- The shifting values of the two LSFR
User anonymity and untraceability	Updating the following every session: 1- (User- IoT) Identities 2- Fresh Random Numbers

Table 2. Adversary capabilities

Capabilities	Definition
Capability 1	The adversary can intercept, replay and modify any message exchanged in the network.
Capability 2	The adversary is a legitimate participant in the network, s/he can initiate a session with any other participant.
Capability 3	The adversary can send/receive messages.
Capability 4	The adversary can perform a man in the middle, impersonation, replay attacks on any run of the protocol.
Capability 5	The adversary can obtain an expired session key.
Capability 6	The adversary can obtain the pre-shared keys of the network participants.
Capability 7	initiate an unlimited number of parallel protocol runs with network participants

3.1. Strand space model

The strand space model [12]-[15] is a formal analysis method vastly used to prove the correctness of authentication protocols. We have selected the strand space model to prove the correctness of (Light-AHAKA) as the authentication test idea is well-suited to authentication protocols based on the challenge-response mechanism. The authentication tests [14], [16]-[18] provide rigorous poof for each challenge and response used in (Light-AHKA) and ensure that each response follows the constraints of the authentication tests. Before proving the correctness of (Light-AHAKA) using the strand space model, we will discuss the basic notions of strand space and the authentication test idea.

3.1.1. Basic notions of the strand space model

The basic notions of the Strand Space Model are as follows:

- Σ: is the set of strand space comprising all strands of the protocol participant in addition to the penetrator strands.
- A: The set of all elements that are exchanged between the protocol participants.
- t: are the elements of set A.
- $+t/-t$: The positive sign means the term t is sent while the minus sign means received.
- $n_1 \rightarrow n_2$: Denotes that the message is sent from node n_1 and received in n_2.
- $n_1 \rightarrow n_2$: Denotes that n_1 and n_2 belong to the same strand and n_1 precedes n_2 on the graph.
- S: Set of all edges in the graph.
- $n \prec_S n'$: Denotes that the path from n to n' contains one or more edges in S.
- $n \preceq_S n'$: Denotes that path from n to n' contains zero or more edges in S.
- T: Denotes the set of atomic messages exchanged in the protocol.
- K: Denotes the set of cryptographic keys of regular strands.
- $\{m\}_K$: Denotes that the participant used the cryptographic key K in encrypting the message m.

3.1.2. Penetrator strands

In this section, the capabilities of the penetrator are presented, which depend on two factors. The first one is the set of keys K_P owned by the penetrator, the second is the ability of the penetrator to generate a new message from the intercepted messages. The strands of the penetrator are illustrated in the following points:

- Text message (M): The penetrator can send a message $<+m>$.
- Flushing (F): The penetrator receives a message from a legitimate participant $<-m>$.
- Tee (T): The penetrator receives the message m and sends it.
- Concatenation (C): The penetrator receives the messages m and t, the penetrator joins them to get (mt), then sends (mt).
- Component separation (S)*:* The penetrator receives the message (mt) and can separate the components (m) and (t) and sends them.
- Key (K): The penetrator sends a key $< + Kp>$ which is from the list of the penetrator keys.
- Encryption (E): The penetrator receives a legitimate key K and a message m, then encrypts m using K, then sends $\{m\}_K$
- Decryption (D): The penetrator receives a private key K^{-1} and a ciphertext $\{m\}_K$, then decrypts $\{m\}_K$ using K^{-1}, and extracts the message m, then sends it.
- Hash message authentication code (HAMC): The penetrator receives K and a message m, and obtains the HMAC function. The penetrator calculates the HMAC value of $K||m$, then sends $HMAC\ \{K\ ||M\}$ (additional penetrator strand).

3.1.3. Authentication test idea

Based on the strand space, Thayer and Guttman [14], [16]-[18] proposed the concept of authentication tests. It formalizes the challenge-response method used in structuring many authentication protocols. In the authentication tests, a protocol participant transmits a test component (e.g. Nonce), and later receives back the test component in another transformed form, then only a regular participant, not a penetrator, performed this transformation. Accordingly, mutual authentication can be achieved based on the idea of authentication tests. For proving a security protocol correctness, one or more of the following authentication tests must be examined:

a) Outgoing Test: a challenge (nonce) is sent in an encrypted form by a protocol participant. The receiver, as a regular participant, is challenged to decrypt it and extract the nonce and send it back to the sender in another encrypted form. i.e. the encrypted form is going out of the edge.
b) Incoming Test: a challenge (nonce) is sent by a protocol participant. The receiver is challenged to encrypt the nonce and send it back to the sender to prove its legitimacy. i.e. the encrypted form is incoming to the edge.
c) Unsolicited Test: a participant receives a message without a prior request. If the message form shows that it can only be produced by a legitimate participant, we can deduce that the regular node that originated the message is preceding the receiving node. It is frequently used in the case of a server requesting a client to send its authentication parameters.

The security analysis of (Light-AHAK) is based on the outgoing test and the unsolicited test, which are considered as the first and third authentication tests respectively. The theorems for these two tests are formalized as follows**:**

Theorem 1: Let n and $n' \in C$, if $n \Rightarrow^{+} n'$ is an outgoing test for a in t then:

1) The nodes m, $m' \in C$ exit.
2) t is a component of m.
3) $m \Rightarrow^{+} m'$ is transforming edge for a.

Additionally, if *:*

1) a occurs only in t_1.
2) $t1$ subterm of m'.
3) $K^{-1} \notin P$.

Then there exists a regular negative node that receives t_1 as a component which is n'.

Theorem 2: For a test component $t=\{h\}_K$, n is considered an unsolicited test for t, if there exists a positive regular node m, t is a component of m *and* m is preceding n such that $m \preceq_C n$.

The strand space introduced tests for testing the encrypted components, but other cryptographic functions like the *HMAC* are not represented. When the *HMAC* cryptographic function is used in authentication protocols, the formal analysis becomes more sophisticated. To reinforce the formal analysis of the authentication protocols, the test theorem of *HMAC* is proposed in [19] as follows:

Theorem 3: Let $t= (h)_{HMAC\_K}$, $t \subset term\ (n)$, t is a new component of n and n is a negative node, assuming K is safe. Then, there must be a regular node m preceding n, $m \preceq_C n$, and t and h are uniquely originating at m.

3.1.4. Light-AHAKA formal analysis

Figure 3 shows Light-AHAKA executive bundle, which contains 3 sets: *authentication server* strands, *user* strands, *IoT* strands. The trace of strands in the Light-AHAKA is presented in Table 3.

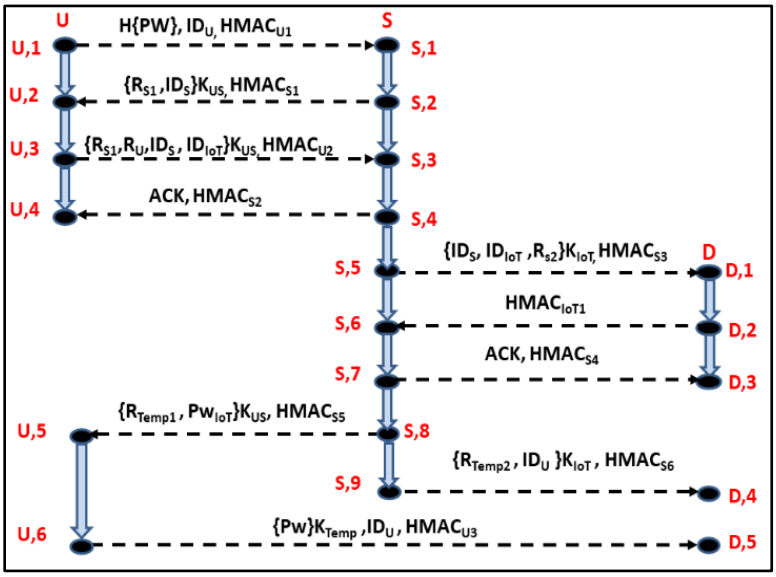

Figure 3. Light-AHAKA bundle

Table 3. Set of strands

Principal	Set of Strands	Trace
Authentication Server (*AS*)	AS [$H(PW_U)$, ID_U, ID_S, ID_{IoT}, R_{S1}, R_{S2}, R_U, R_{Temp1}, R_{Temp2}, PW_{IoT}, $HMAC_{U1-3}$, $HMAC_{S1-6}$, $HMAC_{IoT1}$]	< - ID_U, $H(PW_U)$, $HMAC_{U1}$ + $\{R_{S1}, ID_S\}_{K_{US}}$, $HMAC_{S1}$ - $\{R_{S1}, R_U, ID_S, ID_{IoT}\}_{K_{US}}$, $HMAC_{U2}$ + ACK, $HMAC_{S2}$ + $\{ID_S, ID_{IoT}, R_{S2}\}_{K_{IoT}}$, $HMAC_{S3}$ - $HMAC_{IoT1}$ + ACK, $HMAC_{S4}$ + $\{R_{Temp1}, PW_{IoT}\}_{K_{US}}$, $HMAC_{S5}$ + $\{R_{Temp2}, ID_U\}_{K_{IoT}}$, $HMAC_{S6}$ >
User(*U*)	U[$H(PW)$, ID_U, ID_S, ID_{IoT}, R_{S1}, R_U, R_{Temp1}, PW_{IoT}, $HMAC_{U1-3}$, $HMAC_{S1,2,5}$]	< - ID_U, $H(PW_U)$, $HMAC_{U1}$ - $\{R_{S1}, ID_S\}_{K_{US}}$, $HMAC_{S1}$ + $\{R_{S1}, R_U, ID_S, ID_{IoT}\}_{K_{US}}$, $HMAC_{U2}$ - ACK, $HMAC_{S2}$ - $\{R_{Temp1}, PW_{IoT}\}_{K_{US}}$, $HMAC_{S5}$ + $\{PW_{IoT}\}_{K_{Temp}}$, ID_U, $HMAC_{U3}$ >
IoT(*D*)	D [ID_{IoT}, ID_U, ID_S, PW_{IoT}, R_{S2}, R_{Temp2}, R_U, $HMAC_{IoT1}$, $HMAC_{S3,4,6}$, $HMAC_{U3}$]	< - $\{ID_S, ID_{IoT}, R_{S2}\}_{K_{IoT}}$, $HMAC_{S3}$ + $HMAC_{IoT1}$ - ACK, $HMAC_{S4}$ - $\{R_{Temp2}, ID_U\}_{K_{IoT}}$, $HMAC_{S6}$ - $\{PW_{IoT}\}_{K_{Temp}}$, ID_U, $HMAC_{U3}$ >

4.1.5. Strand space proof

In this section, the correctness of (Light-AHAKA) will be proved by proposing eight lemmas for bundle *C* which is in Σ space.

According to Figure and Table:

- The *authentication server* strand $S_{AS} \in AS$ is in bundle *C* and its height is 9
- K_{US} and $K_{IoT} \notin P$.
- The random numbers R_{S1}, R_{S2}, *and* R_U are fresh values and uniquely originating in Σ.
- There is *user* strand $S_U \in U$ in bundle *C* and its height is 6 at least.
- There is *IoT* strand $S_D \in D$ in bundle C and its height is 5 at least.

Lemma 1: If $H \notin P$, then node $n = <S, 1>$ is an unsolicited test for $t = \{PW\}_H$ where the originating edge is $m=<U,1>$ and $a = PW$.

Proof: According to Theorem 2, $m =<U,1>$ is the only positive regular node where $t \sqsubset m$; $t \not\sqsubset n$ for all n such that $m \preceq_C n$.

Lemma 2: If $K_{US} \notin P$, R_{S1} is uniquely originating in $<S, 2>$ then the edge $<S, 2> \Rightarrow^+ <S, 3>$ is an outgoing test for R_{S1}, $t= \{R_{S1}, ID_S\}_{K_{US}}$ is the test component and $a= R_{S1}$.

Proof: According to Theorem 1, proving the lemma is achieved by finding two regular nodes (m, m') in bundle *C* and $m \Rightarrow^+ m'$ is transforming edge for R_{S1}. In Figure, $m =<U,2>$ in *user* strand S_U, and $m' =<U,3>$ in S_U.

Lemma 3: If $K_{US} \notin P$, R_U is uniquely originating in $<U, 3>$then the edge $<U, 3> \Rightarrow^+ <U, 4>$ is an outgoing test for R_U, $t= \{R_{S1}, R_U, ID_S, ID_{IoT}\}_{K_{US}}$ is the test component and $a= R_U$ and the response is in $HMAC_{S2}$ using $R_U \notin P$ as the key for the $HMAC_{S2}$.

Proof:

1) According to Theorem 1, proving the lemma is achieved by finding two regular nodes (m, m') in bundle *C* and $m \Rightarrow^+ m'$ is transforming edge for R_U. In Figure, $m =<S,3>$ in *server* strand $S_{AS,}$ and $m'=<S,4>$ of S_{AS}.
2) According to Theorem 3 and Figure, $n=<U, 4>$ is the only negative node of *user* strand S_U, and the test component $t= HMAC_{S3}$ is sub-term of the node n and a new component of this node, while $R_U \notin P$. Then the positive node $m=<S, 4>$ such that $m \preceq_C n$, where t is uniquely originating at m.

From the proof of the three aforementioned lemmas, we prove the correctness of the mutual authentication between the authentication server *S* and the user *U*; moreover, the random numbers (R_{S1} & R_U) are freshly generated from regular strands.

Lemma 4: If $K_{IoT} \notin P$, R_{S2} is uniquely originating in <*S, 5*> then the edge <*S, 5*> $\Rightarrow^+$ <*S, 6*> is an outgoing test for R_{S2}, $t = \{ID_S, ID_{IoT}, R_{S2}\}_{K_{IoT}}$ is the test component and $a = R_{S2}$, and the response is in $HMAC_{S2}$ using $R_U \notin P$ as the key for the $HMAC_{S2}$.

Proof:

1) According to Theorem 1, proving the lemma is achieved by finding two regular nodes (*m*, *m'*) in bundle *C* and $m \Rightarrow^+ m'$ is transforming edge for R_{S2}. In Figure 3, *m*=<*D,1*> in *D* strand S_D, and *m'*=<*D,2*> of S_D.
2) According to Theorem 3 and Figure 3, *n*=<*S, 6*> is the only negative server strand S_{AS}, and the test component $t = HMAC_{IoT1}$ is sub-term of the node *n* and a new component this node, while $K_{ds} \notin P$. Then the positive node *m*=<*D, 2*> such that $m \preceq_C n$, where *t* is uniquely originating at *m*.

From the proof of the previous lemma, we prove the correctness of the mutual authentication between the authentication server *S* and the *IoT* device. Furthermore, the random numbers (R_{S2}) are freshly generated from a regular strand.

Lemma 5: If $R_{IoT} \notin P$, $HMAC_{S4}$ is uniquely originating in *m*=<*D, 3*> then the node *n*= <*S, 7*> is a test for $t = HMAC_{S4}$ with R_{IoT} as a key, and the test component is *a*= *ACK*.

Proof: According to Theorem 3 and Figure 3, *n* is the only negative node of bundle *C*, and $t = HMAC_{S4} \subset term(n)$ is a new component of *n* with $K \notin P$. Then the positive node *m*=<*S, 7*> such that $m \preceq_C n$, where *t* is uniquely originating at *m*.

Lemma 6: If $K_{US} \notin P$, then node *n*= <*U, 5*> is an unsolicited test for $t = \{R_{Temp1}, Pw_{IoT}\}_{K_{US}}$ where *m*=<*S, 8*> and $a = R_{Temp1}$.

Proof: According to Theorem 2 and Figure 3, *m*=<*S,8*> is the only positive node in *server* strand S_{AS} where $t \subset m$; $t \not\subset n$ for all *n* such that $m \preceq_C n$.

Lemma 7: If $K_{IoT} \notin P$, then node *n*= <*D, 4*> is an unsolicited test for $t = \{R_{Temp2}, ID_U\}_{K_{IoT}}$ where *m*=<*S, 9*> and $a = R_{Temp2}$.

Proof: According to Theorem 2 and Figure 3, *m*=<*S,9*> is the only positive node in *server* strand S_{AS} where $t \subset m$; $t \not\subset n$ for all *n* such that $m \preceq_C n$.

Lemma 8: If $K_{Temp} \notin P$, then node *n*= <*D, 5*> is an unsolicited test for $t = \{Pw\}_{K_{Temp}}$ where *m*=<*U, 6*> and $a = Pw$.

Proof: According to Theorem 2 and Figure 3, *m*=<*U,6*> is the only positive node in user strand S_U where $t \subset m$; $t \not\subset n$ for all *n* such that $m \preceq_C n$.

4.2. Simulation for formal security verification using AVISPA tool

To ensure the correctness of (Light-AHAKA), we used automated validation of internet security protocols and applications (AVISPA) as a tool for simulating (Light-AHAKA).

4.2.1. AVISPA

AVISPA is an automated tool used for validating security protocols and cryptographic applications [20]-[22]. AVISPA is used for analyzing the security properties of the investigated protocols by searching for possible attacks in different scenarios. We specified (Light-AHAKA) in high-level protocol specification language (HLPSL), which is a role-based language developed for AVISPA.

The HLPSL language specifies the different roles in the authentication protocols. Each role represents a protocol participant, and then all the roles are composed to represent the interacting behavior of the participants. Each role specified in HLPSL is independent of the other roles, setting the initial knowledge of each role and communicating with the other roles via data transfer channels. HLPSL is a role-based language, in such a way that the sequence of actions of each protocol participant is specified in a separate module, which is called a basic role. After completing the step of specifying all the roles of the protocol participants, these roles will be instantiated by one or more agents playing the given role and describing how the participants interact with each other by concatenating all the basic roles together into one composed role.

AVISPA uses the HLPSL2IF tool for transforming a specification written in the HLPSL language into a low-level specification in the IF language. This tool compiles the specification of a protocol given as a parameter in a file with the extension (.hlpsl), and either lists the errors found in the specification or generates a file with the same name but with a new extension (.if) containing the specification that will be analyzed later on. In the field of designing lightweight authentication protocols for IoT-Cloud computing, numerous researchers [23]-[28] relied on the AVISPA tool for simulating their proposals and verifying the correctness of the proposed lightweight protocols, as AVISPA presents four protocol analyzer tools, on-the-fly model-

checker (OFMC) [29], SAT-based model-checker (SATMC) [30], constraint logic-based attack searcher (CL-AtSe) [31], and tree automata based on automatic approximations for the analysis of security protocols (TA4SP) [32].

AVISPA implementation consists of the users' roles, session role, environment role, and finally the security goals. The user roles comprise all agents in the protocol, the symmetric keys, and finally the send/receive channels. All the messages exchanged via the channels are subjected to the control of the Dolev-Yao *(dy)* intruder model. According to this model, it is assumed that the intruder has full power over the communication network, such that all messages exchanged by the agents are intercepted by the intruder. In addition, the intruder has the power to analyze, modify, and compose new messages to other protocol participants, pretending that these messages were initiated by a legitimate agent. The local section defines all the local variables used by each role.

4.2.1. User role

Figure 4 shows the role of the user *(U)*, which is considered as the initiator of the protocol, played by *U*. The knowledge of *U* comprises all agents in the protocol (*U*, authentication server *(S)*, IoT device *(D)*, and the symmetric pre-shared key *(k_{us})* between *U* and *S*.

"Rcv(start)" is sent to *U* as a trigger signal to initiate the protocol run. *U* starts the registration phase with the authentication server *S* (step 1) and after successful registration, *U* starts the mutual authentication process with *S* (steps 2 & 3). Finally, *U* receives an acknowledgment *(ACK)* from the server (step 4). In step (5), *U* receives the credentials of *D* from the server and sends an access request to *D* (step 6).

```
role role_U(S:agent,U:agent,D:agent,Kus:symmetric_key,SND,RCV:channel(dy))
played_by U
def=
        local
                State:nat,H:hash_func,PWu:text,Rs1:text,Ru:text,ACK1:text,
                Rtemp1:text,T:text,Ktemp1:symmetric_key,PWd:text,HMAC:hash_func

        init
                State := 0
        transition
                1.  State=0   /\ RCV(start) =|>
        %STEP(1):USER REGISTERATION
                    State':=1 /\ T':=new() /\ PWu':=new()
                              /\ SND(U.H(PWu').T'.HMAC(U.H(PWu').T'))

        %STEP(2):USER VERIFIES SERVER CHALLENGE
                2. State=1    /\ RCV({S.Rs1'}_Kus.T.HMAC({S.Rs1'}_Kus.T)) =|>
                   State':=2  /\ request(U,S,auth_1,Rs1')

        %STEP(3):USER GENERATES ITS  CHALLENGE  AND SEND IT TO THE SERVER
                              /\ Ru':=new() /\ secret(Ru',sec_4,{U,S})
                              /\ SND({Rs1'.Ru'.D}_Kus.T.HMAC({Rs1'.Ru'.D}_Kus.T))
                                        /\ witness(U,S,auth_2,Ru')

        %STEP(4):USER RECEIVES ACK FROM THE SERVER
                4. State=2    /\ RCV(ACK1'.HMAC(ACK1')) =|> State':=3

        %STEP(5):USER RECEIVES IOT PASSSWORD
                8. State=3    /\ RCV({Rtemp1'.PWd'}_Kus.T.HMAC({Rtemp1'.PWd'}_Kus.T)) =|>

        %STEP(6):USER SENDS ACCESS RREQUEST TO IOT
                    State':=4  /\ Ktemp1':=new()
                                        /\ SND({PWd'}_Ktemp1'.D.T.HMAC({PWd'}_Ktemp1'.D.T))
end role
```

Figure 2. User role

4.2.2. Server role

Figure 5 shows the role of the server *(S)*. *S* receives a registration request from *U* (step 1). Then *S* starts the mutual authentication process with *U* (steps 2 & 3) and after passing the mutual authentication, *S* will send *U* the *ACK* message (step 4). Then, S will start authenticating the intended IoT device *D* (steps 5 and 6). After successful authentication, *S* sends *D* an *ACK* message (step 7). Finally, *S* sends the credentials of *D* to U (step 8), and the identity of *U* to *D* (step 9).

4.2.3. Device role

Figure 6 shows the role of the device *(D)*. *D* receives the challenge from the server to start the mutual authentication process (step 1). *D* responds with the *HMAC* of *S* challenge concatenated with its password (step 2). *S* verifies the *HMAC* of *D*, then *S* replies with the acknowledgment *ACK* (step 3). In (step 4), *S* sends *D* the identity of *U* that is requesting access to the data from the *IoT* device. Finally, in (step 5) D receives the access request from *U*.

```
role role_S(S:agent,U:agent,D:agent,Kus:symmetric_key,Kds:symmetric_key,SND,RCV:channel(dy))
played_by S
def=
        local
                State:nat,H:hash_func,PWu:text,Rs1:text,Ru:text,ACK1:text,Xor:hash_func,
                Rs2:text,ACK2:text,PWd:text,Rtemp1:text,T:text,Rtemp2:text,HMAC:hash_func

        init
                State := 0
        transition
        %STEP(1):SERVER RECEIVES USER REGISTERATION REQUEST
                1. State=0   /\ RCV(U.H(PWu').T'.HMAC(U.H(PWu').T')) =|>

        %STEP(2):SERVER GENERATES THE CHALLENGE FOR THE USER
                  State':=1  /\ Rs1':=new() /\ secret(Rs1',sec_5,{U,S})
                             /\ SND({S.Rs1'}_Kus.T'.HMAC({S.Rs1'}_Kus.T'))
                                    /\ witness(S,U,auth_1,Rs1')

        %STEP(3):SERVER RECEIVES THE RESPONSE FROM THE USER
                3. State=1   /\ RCV({Rs1.Ru'.D}_Kus.T.HMAC({Rs1.Ru'.D}_Kus.T)) =|>
                   State':=2 /\ request(S,U,auth_2,Ru')

        %STEP(4):SERVER SENDS ACK TO THE USER
                             /\ ACK1':=new() /\ SND(ACK1'.HMAC(ACK1'))
        %STEP(5):SERVER GENERATES CHALLENGE TO THE IOT
                             /\ Rs2':=new() /\ secret(Rs2',sec_6,{S,D})
                             /\ SND({S.D.Rs2'}_Kds.T.HMAC({S.D.Rs2'}_Kds.T))
                                      /\ witness(S,D,auth_3,Rs2')

        %STEP(6):SERVER RECEIVES THE RESPONSE FROM THE IOT
                6. State=2   /\ RCV(HMAC(T.Xor(T.Rs2.PWd')).T) =|>

        %STEP(7):SERVER SENDS ACK TO THE IOT
                   State':=3 /\  ACK2':=new() /\ SND(ACK2'.HMAC(ACK2'))
        %STEP(8):SERVER SENDS IOT PASSSWORD TO THE USER
                             /\ Rtemp1':=new()
                                      /\ SND({Rtemp1'.PWd'}_Kus.T.HMAC({Rtemp1'.PWd'}_Kus.T))

        %STEP(9):SERVER SENDS USER ID TO THE IOT
                             /\ Rtemp2':=new()
                                      /\ SND({Rtemp2'.D}_Kds.T.HMAC({Rtemp2'.D}_Kds.T))
end role
```

Figure 5. Server role

```
role role_D(S:agent,D:agent,U:agent,Kds:symmetric_key,SND,RCV:channel(dy))
played_by D
def=
        local
                State:nat,Xor:hash_func,Rs2:text,ACK2:text,Rtemp2:text,
                T:text,Ktemp1:symmetric_key,PWd:text,HMAC:hash_func


        init
                State := 0
        transition
        %STEP(1):IOT RECEIVES SERVER CHALLENGE
                5. State=0    /\ RCV({S.D.Rs2'}_Kds.T'.HMAC({S.D.Rs2'}_Kds.T')) =|>
                   State':=1  /\ witness(D,S,auth_3,Rs2')

        %STEP(2):IOT SENDS RESPONSE AND CREDENTIALS TO THE SERVER
                              /\ PWd':=new() /\ SND(HMAC(T'.Xor(T'.Rs2'.PWd')).T')

        %STEP(3):IOT RECEIVES ACK FROM THE SERVER
                7. State=1    /\ RCV(ACK2'.HMAC(ACK2')) =|> State':=2

        %STEP(4):IOT RECEIVES USER ID FROM THE SERVER
                9. State=2    /\ RCV({Rtemp2'.D}_Kds.T.HMAC({Rtemp2'.D}_Kds.T)) =|> State':=3

        %STEP(5):IOT RECEIVES ACCESS REQUEST FROM THE USER
                10. State=3   /\ RCV({PWd}_Ktemp1'.D.T.HMAC({PWd}_Ktemp1'.D.T)) =|> State':=4
end role
```

Figure 6. Device role

4.2.4. Session role

Figure 7 shows the role of the session of the Light-AHAKA in HLPSL. The roles of the three agents, *U*, *S*, and *D*, are combined by defining a role for the session. The three roles are instantiated with the arguments of each role and combined using the keyword *composition*. In the composition role, the sessions are illustrated by specifying how the agents interact in the (Light-AHAKA) protocol.

```
role session1(S:agent,U:agent,D:agent,Kus:symmetric_key,Kds:symmetric_key)
def=
        local
                SND3,RCV3,SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_S(S,U,D,Kus,Kds,SND3,RCV3)
                /\ role_D(S,D,U,Kds,SND2,RCV2)
                /\ role_U(S,U,D,Kus,SND1,RCV1)
end role

role session2(S:agent,U:agent,D:agent,Kus:symmetric_key,Kds:symmetric_key)
def=
        local
                SND3,RCV3,SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_S(S,U,D,Kus,Kds,SND3,RCV3)
                /\ role_D(S,D,U,Kds,SND2,RCV2)
                /\ role_U(S,U,D,Kus,SND1,RCV1)
end role

role session3(S:agent,U:agent,D:agent,Kus:symmetric_key,Kds:symmetric_key)
def=
        local
                SND3,RCV3,SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_S(S,U,D,Kus,Kds,SND3,RCV3)
                /\ role_D(S,D,U,Kds,SND2,RCV2)
                /\ role_U(S,U,D,Kus,SND1,RCV1)
end role

role session4(S:agent,U:agent,D:agent,Kus:symmetric_key,Kds:symmetric_key)
def=
        local
                SND3,RCV3,SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_S(S,U,D,Kus,Kds,SND3,RCV3)
                /\ role_D(S,D,U,Kds,SND2,RCV2)
                /\ role_U(S,U,D,Kus,SND1,RCV1)
end role

role session5(S:agent,U:agent,D:agent,Kus:symmetric_key,Kds:symmetric_key)
def=
        local
                SND3,RCV3,SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_S(S,U,D,Kus,Kds,SND3,RCV3)
                /\ role_D(S,D,U,Kds,SND2,RCV2)
                /\ role_U(S,U,D,Kus,SND1,RCV1)
end role
```

Figure 7. Session role

4.2.5. Environment role

The environment role is the top-level role that specifies the global constants and forms the composition of one or more sessions. In Figure 8, five sessions are instantiated and the intruder is represented and his/her knowledge is defined. The first session is the normal session with the three agents of (Light-AHAKA). In the second session, the intruder represents the user *U* with the knowledge of the pre-shared key *kus*, which is represented as *kis*. In the third session, the intruder presents the server *S* with the knowledge of the pre-shared key *Kus*, which is presented as *Kis*. In the third session, the intruder presents also the server *S*, but in this session the knowledge is different, the intruder knows the pre-shared key *kus*, which is presented as *kis*. In the fourth session, the intruder presents the authentication server *S* with the knowledge of the pre-shared key *kds*, which is presented as *kis*. Finally, in the fifth session, the intruder presents the *IoT* device *D* with the knowledge of the pre-shared key *kds* as *kis*.

```
role environment()
def=
        const
                kus:symmetric_key,server:agent,hash_0:hash_func,kds:symmetric_key,
                bob:agent,alice:agent,kis:symmetric_key,
                auth_1,auth_2,auth_3,sec_4,sec_5,sec_6: protocol_id

        intruder_knowledge = {alice,bob,server,kis}
        composition
                session5(server,alice,i,kus,kis)
                /\ session4(i,alice,bob,kus,kis)
                /\ session3(i,alice,bob,kis,kds)
                /\ session2(server,i,bob,kis,kds)
                /\ session1(server,alice,bob,kus,kds)
end role
```

Figure 8. Environment role

4.2.6. Security goals

AVISPA achieves two goals, which are secrecy and authentication. Secrecy is verified via the goal predicate *secret*, while authentication is verified utilizing the goal predicates *witness and request.* In (Light-AHAKA) implementation Figure 9, the following secrecy and authentications goals are examined and verified:

1) Two authentication goals:
   a) The authentication_on auth_1 represents that the random number *Ru* is generated by *U* and only known to *U*. If *S* verifies that this random number is generated by *U* and encrypted by the pre-shared key *Kus*, then *S* authenticates *U*.
   b) The authentication_on auth_2 represents the random number *Rs1* is generated by *S* and only known to *S*. If *U* verifies that this nonce is generated by *S* and encrypted by the pre-shared key *kus*, then *U* authenticates *S*.
   c) The authentication_on auth_3 represents that the random number *Rs2* is generated by *S* and only known to *S*. If *D* verifies that this nonce is generated by *S* and encrypted by the pre-shared key *Kds*, then *D* authenticates *S*.
2) Three secrecy goals:
   a) The secrecy_of sec_4 denotes that the random number *Ru* is kept secret only to *U* and *S*.
   b) The secrecy_of sec_5 denotes that the random number *Rs1* is kept secret only to *S* and *U*.
   c) The secrecy_of sec_6 denotes that the random number *Rs2* is kept secret only to *D* and *S*.

```
goal
        authentication_on auth_1
        authentication_on auth_2
        authentication_on auth_3
        secrecy_of sec_4
        secrecy_of sec_5
        secrecy_of sec_6

end goal
```

Figure 9. Light-AHAKA goals

2.4.7. Simulation results

In this section, we presented the simulation results of our (Light-AHAKA) protocol using the back-ends OFMC and CL-AtSe using AVISPA.

Figure 10 and Figure 11 ensure that the simulation of (Light-AHAKA) is considered SAFE under the two back-ends, OFMC and CL-AtSe, respectively. The (Light-AHAKA) achieves mutual authentication between all protocol participants. All the random numbers generated in the protocol procedure are kept secret, and finally, no attacks were found.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/LightWeight_V9.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 108.41s
  visitedNodes: 5020 nodes
  depth: 18 plies
```

Figure 10. OFMC result

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/LightWeight_V9.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 3615 states
  Reachable  : 1139 states
  Translation: 0.88 seconds
  Computation: 1.30 seconds
```

Figure 11. ATSE result

4. RESULTS AND DISCUSSION

In IoT-Cloud computing networks, users may require to access real-time data from IoT devices directly rather than accessing offloaded data to the cloud. This use-case is for instant and critical decisions. In this case, the user and the IoT device have to be authenticated before granting the user the right to access the data. In this context, we presented (Light-AHAKA) as a new authentication and key agreement protocol for IoT in cloud computing.

The main advantages of (Light-AHAKA) are as follows:

- (Light-AHAKA) is a challenge-response protocol based on lightweight cryptographic functions.
- (Light-AHAKA) authenticates the user and the IoT device (Mutual Authentication).
- Considering the constrained nature of IoT devices, the response in the authentication of the IoT device is formed of the HMAC function only.
- (Light-AHAKA) provides perfect forward secrecy.
- (Light-AHAKA) provides key agreement to encrypt the exchanged traffic.
- (Light-AHAKA) updates the passwords and identities of the user and the IoT device every session.
- Unprecedented key agreement and parameters update module Figure 2.
- (Light-AHAKA) is immune to the attacks summarized in Table 1.

In this paper, we conducted the formal analysis of (Light-AHAKA) based on two approaches, the Strand Space Model, and the AVISPA simulation tool. The results of the formal analysis were as follows:

1) By proving eight lemmas based on the Strand Space Model:
   - All challenge-response messages were tested according to the authentication test idea.
   - No extra/ missing messages were found.
   - It was proved that all messages were initiated by regular strands.
   - (Light-AHAKA) achieves mutual authentication.
   - All random numbers are fresh and generated by regular strands.
2) By Simulating (Light-AHAKA) using two back ends of the AVISPA tool :
   - No attacks were found on (Light-AHAKA).
   - The authentication server authenticates the user.
   - The authentication server authenticates the IoT device.
   - All random numbers are fresh and kept secret.

According to the aforementioned results, it was necessary to conduct a formal analysis of (Light-AHAKA) based on the aforementioned approaches. The first approach verifies the correctness of challenge-response-based protocols, checks the right sequences of the exchanged messages, and the honesty of the strands. The second approach searches for possible attacks on (Light-AHAKA) using two back ends.

5. CONCLUSION

We proposed (Light-AHAKA) as a new lightweight authenticated key agreement protocol for IoT in cloud computing. The (Light-AHAKA) authenticates the user and the IoT device before granting the user the right to access the IoT device directly for critical and instant decisions. Additionally, the user and the IoT device generate a session key to encrypt the exchanged traffic. To enhance the resiliency of IoT-Cloud networks, after each session, the passwords and the identities of (IoT-User) are updated. The security analysis was conducted to ensure that Light-AHAKA is immune to potential attacks. In this paper, formal verification of the (Light-AHAKA) was conducted to ensure that the protocol is secure against known security attacks. We conducted the formal verification based on two different approaches: the Strand Space Model and the AVISPA. The Strand Space Model tested all challenges and responses forming the protocol and proved that each challenge was succeeded by the appropriate response. In AVISPA we used two back-ends, OFMC and CL-AtSe. The results of the two approaches show that (Light-AHAKA) achieved mutual authentication, all random numbers generated in the protocol procedure were kept secret, and no attacks were found. In future work, we plan to work on the practical implementation of the (Light-AHAKA). A test-bed network will be constructed to test and evaluate the communication cost, computational cost, execution time, and storage cost.

REFERENCES

[1] "HIS, Internet of Things (IoT) Connected Devices Installed BaseWorldwide from 2015 to 2025," 2021. Accessed: 25 May. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[2] M. Wazid, A. K. Das, R. Hussain, G. Succi, and J. J. Rodrigues, "Authentication in cloud-driven IoT-based big data environment: Survey and outlook," *Journal of Systems Architecture*, vol. 97, pp. 185-196, 2019, doi: 10.1016/j.sysarc.2018.12.005.

[3] M. Medwed, "IoT security challenges and ways forward," in *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices*, 2016, pp. 55-55, doi: 10.1145/2995289.2995298.
[4] Y. H. Hwang, "IoT security & privacy: threats and challenges," in *Proceedings of the 1st ACM workshop on IoT privacy, trust, and security*, 2015, pp. 1-1, doi: 10.1145/2732209.2732216.
[5] M. Abomhara and G. M. Køien, "Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks," *Journal of Cyber Security and Mobility*, pp. 65-88, 2015, doi: 10.13052/jcsm2245-1439.414.
[6] A. Aly, G. Atef, N. Mona, and A. E.-H. Ahmed, "A New Lightweight Authenticated Key Agreement Protocol For IoT In Cloud Computing," *Journal of Engineering Science and Technology (JESTEC)*, vol. 16, no. 5, 2021.
[7] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Annual international cryptology conference*, Springer, pp. 1-15, 1996, doi: 10.1007/3-540-68697-5_1.
[8] K. McKay, L. Bassham, M. Sönmez Turan, and N. Mouha, "Report on lightweight cryptography," *National Institute of Standards and Technology*, 2016, doi: 10.6028/NIST.IR.8114.
[9] H. Dobbertin, "Almost perfect nonlinear power functions on GF (2n): the Niho case," *Information and Computation*, vol. 151, no. 1-2, pp. 57-72, 1999, doi: 10.1006/inco.1998.2764.
[10] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198-208, 1983, doi: 10.1109/TIT.1983.1056650.
[11] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2002, pp. 337-351, doi: 10.1007/3-540-46035-7_22.
[12] F. J. T. Fabrega, J. C. Herzog and J. D. Guttman, "Strand spaces: why is a security protocol correct?," Proceedings. 1998 *IEEE Symposium on Security and Privacy (Cat. No.98CB36186)*, 1998, pp. 160-171, doi: 10.1109/SECPRI.1998.674832.
[13] F. J. T. Fábrega, J. C. Herzog, and J. D. Guttman, "Strand spaces: Proving security protocols correct," *Journal of computer security*, vol. 7, no. 2/3, pp. 191-230, 1999, doi: 10.3233/JCS-1999-72-304.
[14] Guttman and F. J. Thayer, "Authentication tests," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, 2000: IEEE, pp. 96-109, doi: 10.1109/SECPRI.2000.848448.
[15] S. Doghmi, J. Guttman, and F. J. Thayer, "Skeletons and the shapes of bundles," in *Proc. 7th Int. Workshop on Issues in the Theory of Security*, 2006: Citeseer, pp. 24-25.
[16] J. D. Guttman, "Key compromise, strand spaces, and the authentication tests," *Electronic Notes in Theoretical Computer Science*, vol. 45, pp. 1-21, 2001, doi: 10.1016/S1571-0661(04)80960-5.
[17] J. D. Guttman and F. J. Thayer, "Authentication tests and the structure of bundles," *Theoretical computer science*, vol. 283, no. 2, pp. 333-380, 2002, doi: 10.1016/S0304-3975(01)00139-6.
[18] J. D. Guttman, "Authentication tests and disjoint encryption: a design method for security protocols," *Journal of Computer Security*, vol. 12, no. 3-4, pp. 409-433, 2004, doi: 10.3233/JCS-2004-123-405.
[19] M. Yao, D. Zhou, R. Deng, and M. Liu, "A Security Protocol for Access to Sensitive Data in Trusted Cloud Server," in *International Conference on Cloud Computing and Security*, 2018: Springer, pp. 531-542, doi: 10.1007/978-3-030-00009-7_48.
[20] A. Armando *et al.*, "The AVISPA tool for the automated validation of internet security protocols and applications," presented at the International conference on computer aided verification, 2005, doi: 10.1007/11513988_27.
[21] "AVISPA v1.0 User Manual." Accessed: 25 May, 2021. [Online]. Available: http://avispa-project.org
[22] L. Vigano, "Automated security protocol analysis with the AVISPA tool," *Electronic Notes in Theoretical Computer Science*, vol. 155, pp. 61-86, 2006, doi: 10.1016/j.entcs.2005.11.052.
[23] H. Khalid, S. J. Hashim, S. M. S. Ahmad, F. Hashim, and M. A. Chaudhary, "A New Secure and Lightweight Multi-Factor Authentication Scheme for Cross-Platform Industrial IoT Systems," *Sensors*, vol. 21, no. 4, p. 1428, 2021, doi: 10.3390/s21041428.
[24] H. Khalid, S. J. Hashim, S. M. Syed Ahmad, F. Hashim, and M. A. Chaudhary, "Cross-SN: A Lightweight Authentication Scheme for a Multi-Server Platform Using IoT-Based Wireless Medical Sensor Network," *Electronics*, vol. 10, no. 7, p. 790, 2021, doi: 10.3390/electronics10070790.
[25] R. Amin, N. Kumar, G. Biswas, R. Iqbal, and V. Chang, "A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment," *Future Generation Computer Systems*, vol. 78, pp. 1005-1019, 2018, doi: 10.1016/j.future.2016.12.028.
[26] G. Sharma and S. Kalra, "A lightweight user authentication scheme for cloud-IoT based healthcare services," *Iranian Journal of ScienceTechnology, Transactions of Electrical Engineering* vol. 43, no. 1, pp. 619-636, 2019, doi: 10.1007/s40998-018-0146-5.
[27] S. Zargar, A. Shahidinejad, and M. Ghobaei-Arani, "A lightweight authentication protocol for IoT-based cloud environment," *International Journal of Communication Systems*, vol. 34, no. 11, p. e4849, 2021, doi: 10.1002/dac.4849.
[28] Y. Ben Slimane, K. J. I. J. o. E. Ben Ahmed, and C. Engineering, "Efficient End-to-End Secure Key Management Protocol for Internet of Things," vol. 7, no. 6, 2017, doi: 10.11591/ijece.v7i6.pp3622-3631.
[29] D. Basin, S. Mödersheim, and L. Vigano, "An on-the-fly model-checker for security protocol analysis," in *European Symposium on Research in Computer Security*, 2003: Springer, pp. 253-270, doi: 10.1007/978-3-540-39650-5_15.
[30] K. L. McMillan, "Interpolation and SAT-based model checking," in *International Conference on Computer Aided Verification*, 2003: Springer, pp. 1-13, doi: 10.1007/978-3-540-45069-6_1.
[31] M. Turuani, "The CL-Atse protocol analyser," in *International Conference on Rewriting Techniques and Applications*, 2006: Springer, pp. 277-286, doi: 10.1007/11805618_21.

[32] Y. Boichut, P.-C. Héam, O. Kouchnarenko, and F. Oehl, "Improvements on the Genet and Klay technique to automatically verify security protocols," in *Proc. AVIS*, 2004, vol. 4, p. 84.

BIOGRAPHIES OF AUTHORS

Ahmed H. Aly received his M.Sc. in Electrical Engineering from Arab Academy for Science, Technology and Maritime Transport (AASTMT) in 2014. Currently, he is Ph.D. student Faculty of Computers & AI, Helwan University. He is an expert in Cryptography and Network Security.

Prof Atef Z. Ghalwash, received his Ph.D. degree from the Faculty of Engineering, Maryland university- USA. Currently, he is a full professor in the Faculty of Computers & AI, Helwan University. Artificial Intelligence, Security & SWE are part of his field of interest.

Prof. Mona M. Nasr. Chief Information Officer, CIO for Helwan University Manager of Scientific Computing Center (SCC) for Helwan University. Head of Information Systems Department.

Dr. Ahmed Ali Abdel-Hafez received his Ph.D. from the School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Canada in 2003. Currently, he is Chief Expert; National Telecom. Regulatory Authority (NTRA), Egypt.