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

Ahmed H. Aly¹, Atef Ghalwash², Mona M. Nasr³, Ahmed A. Abd El-Hafez⁴
¹,²,³Faculty of Computer Science and Artificial Intelligence, Helwan University, Cairo, Egypt
⁴National Telecom, Regulatory Authority (NTRA), Cairo, Egypt

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.

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

This is an open access article under the CC BY-SA license.

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
\Delta T Time range allowed for delay
i Index selected from R_{U} – ex: i = decimal value of iᵗʰ byte of R_{U}
j Index selected from R_{IoT} – ex: j = decimal value of jᵗʰ byte of R_{IoT}

Journal homepage: http://ijeecs.iaescore.com
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 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 (IDt)0, password (PWt)0, and a pre-shared key (PSKt)0 for each user (Ut) stored safely in the client application.
- The (NA) loads (IDt)0, (PWt)0, and pre-shared key (PSKt)0 in a tamper-proof memory for each IoT device.

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

---

**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMAC</td>
<td>Hash message authentication code</td>
</tr>
<tr>
<td>LFSR</td>
<td>Linear feed-back shift register</td>
</tr>
<tr>
<td>NFSR</td>
<td>Non-linear feed-back shift register</td>
</tr>
<tr>
<td>SK</td>
<td>Session key</td>
</tr>
</tbody>
</table>

**Symbols**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⊕</td>
<td>XoR Operation</td>
</tr>
</tbody>
</table>
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:
     - \(E_{PSK_U}(R_S || R_{U} || ID_{IoT} || T_{U2} || (HMAC_{U2}))\).

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 \(R_{U}\) (sent) = \(R_{U}\) (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:
     - \(ACK || HMAC_{S2}\).
   - 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:
     - \( E_{SK_{IoT}}(ID_s || ID_{IoT} || R_{S2} || T_{S2} || HMAC_{S2}) \)
   - The IoT device receives the message, checks if \( T_{S2} < \Delta T \), and verifies \( HMAC_{S2} \).
   - 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:
       - \( R_{IoT} = Hash(R_{S2} \oplus ID_s \oplus ID_{IoT}) \)
   - 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.
   - 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 \( K_{sIoT} \) 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 \( R_{Temp1} \)
   - \( E_{SK_{IoT}}(R_{Temp1} \parallel PW_{IoT} \parallel T_{S2} \parallel HMAC_{S2}) \)
   - The server receives the message, checks if \( T_{S2} < \Delta T \), and verify \( HMAC_{S2} \).
   - 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:
      - \( E_{SK_{IoT}}(R_{Temp2} \parallel ID_u) \parallel T_{S2} \parallel HMAC_{S2} \)
      - The IoT device receives the message, checks if \( T_{S2} < \Delta T \), and verifies \( HMAC_{S2} \).
      - 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:
    - \( E_{K_{Temp1}}(PW_{IoT}) || (ID_U) || T_{UJ} || MAC_{UJ} \)
    - The IoT device receives the message, checks if \( T_{UJ} < \Delta T \), and Verifies \( MAC_{UJ} \).
    - 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:
    - \( IV \) is Hash \( (R_{IoT}) \)
    - \( IV \) is Hash \( (R_{U}) \)
Formal security analysis of lightweight authenticated key agreement protocol for IoT...

Ahmed H. Aly

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.

\[ \text{SK} = F[\text{Shift (L1(i))}, \text{Shift (L2(j))}] \]

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](image)

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

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

Formal security analysis of lightweight authenticated key agreement protocol for IoT... (Ahmed H. Aly)
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 proof for each challenge and response used in (Light-AHAKA) 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:

- \( \Sigma \): 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.
- \( n \prec_n n' \): Denotes that the path from \( n \) to \( n' \) contains one or more edges in \( S \).
- \( n \prec_n n' \): 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 (HMC): 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 a transforming edge for \( a \).

Additionally, if:
1) \( a \) occurs only in \( t_1 \).
2) \( t_1 \) is a subterm of \( m' \).
3) \( K \not\in P \).

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.

![Light-AHAKA bundle](image-url)
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 $\Sigma$ space.

According to Figure and Table:

- The authentication server strand $S_{AS} \in AS$ is in bundle $C$ and its height is 9
- $K_{2/5}$ and $K_{3/7} \not\in P$.
- The random numbers $R_{3/I}, R_{2/D},$ and $R_U$ are fresh values and uniquely originating in $\Sigma$.
- There is user strand $S_U \in U$ in bundle $C$ and its height is 6 at least.
- There is IoT strand $S_I \in D$ in bundle $C$ and its height is 5 at least.

Lemma 1: If $H \not\in P$, then node $n = <S, 1^I>$ is an unsolicited test for $t = \{PW\}_H$ where the originating edge is $m = <U, I>$ and $a = PW$.

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

Lemma 2: If $K_{US} \not\in P$, then strand $S_{UL}$ is uniquely originating in $<S, 2>$ then the edge $<S, 2> \Rightarrow <S, 3>$ is an outgoing test for $R_{3/I}, t = \{R_{3/I}, ID_{3/I}\}_K_{US}$ is the test component and $a = R_{3/I}$.

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 $R_{3/I}$. In Figure, $m = <U, 2>$ in user strand $S_U$, and $m' = <U, 3 >$ in $S_I$.

Lemma 3: If $K_{US} \not\in P$, then strand $S_{UL}$ is uniquely originating in $<U, 3>$ then the edge $<U, 3> \Rightarrow <U, 4>$ is an outgoing test for $R_U$, $t = \{R_U, ID_U, ID_{UL}\}_K_{US}$ is the test component and $a = R_U$ and the response is in $\text{HMAC}_{3/2}$ using $R_U \not\in P$ as the key for the $\text{HMAC}_{3/2}$.

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 $t = \text{HMAC}_{3/2}$ is sub-term of the node $n$ and a new component of this node, while $R_U \not\in P$.

Then the positive node $m = <S, 4>$ such that $m \leq n$, where $t$ is uniquely originating at $m$. 

<table>
<thead>
<tr>
<th>Principal</th>
<th>Set of Strands</th>
<th>Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HMAC_{US, S1, S1, S1, S1, S1}</td>
<td>$- {R_{3/I}, R_S, ID_{AS}}<em>K</em>{US}, \text{HMAC}_{US}$</td>
</tr>
<tr>
<td></td>
<td>$- \text{ACK, HMAC}_{3/2}$</td>
<td>$+ ID_U, ID_{AS}, R_{3/I} K_{US}, \text{HMAC}_{3/2}$</td>
</tr>
<tr>
<td></td>
<td>$- \text{ACK, HMAC}_{3/3}$</td>
<td>$+ {R_{3/I}, PW_{3/I}}<em>K</em>{US}, \text{HMAC}_{3/3}$</td>
</tr>
<tr>
<td>IoT (D)</td>
<td>D [ID_{DS}, ID_{DU}, ID_{DS}, PW_{DS}, ID_{DU}, HMAC_{DS, DS, DS, DS}]</td>
<td>$&lt; - {ID_{DS}, PW_{DS}, R_{3/I}}<em>K</em>{US}, \text{HMAC}_{DS}$</td>
</tr>
<tr>
<td></td>
<td>$R_{3/I}, R_{3/I}, R_{3/I}, PW_{3/I}$</td>
<td>$+ \text{HMAC}_{DS, DS}$</td>
</tr>
<tr>
<td></td>
<td>$- \text{ACK, HMAC}_{3/3}$</td>
<td>$+ {R_{3/I}, PW_{3/I}}<em>K</em>{US}, \text{HMAC}_{3/3}$</td>
</tr>
<tr>
<td></td>
<td>$+ {PW_{US}}<em>K</em>{US}, ID_U, \text{HMAC}_{US}$</td>
<td>$- \text{ACK, HMAC}_{3/4}$</td>
</tr>
</tbody>
</table>

\[\text{HMAC}_{3/2}\]
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 $K_{IoT} \not\in P$, $R_{S2}$ is uniquely originating in $<S, 5>$ then the edge $<S, 5> \Rightarrow <S, 6>$ is an outgoing test for $R_{S2} = [ID_{IoT}, ID_{IoT}, R_{S2}]_{K_{IoT}}$ is the test component and $a = R_{S2}$, and the response is in $HMAC_{S2}$ using $R_{U} \not\in P$ 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 $S_0$, and $m'= <D, 2>$ of $S_0$.

2) According to Theorem 3 and Figure 3, $n= <S, 6>$ is the only negative server strand $S_{AS}$, and the test component $t= HMAC_{IoT}$ is sub-term of the node $n$ and a new component this node, while $K_{IoT} \not\in P$.

Then the positive node $m= <D, 2>$ such that $m \not\leq n$, 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 $R_{IoT} \not\in P$, $HMAC_{Sd}$ is uniquely originating in $m= <D, 3>$ then the node $n= <S, 7>$ is a test for $t= HMAC_{Sd}$ with $R_{IoT}$ as a key, and the test component is $a = ACK$.

Proof: According to Theorem 3 and Figure 3, $n= <S, 6>$ is the only negative node of bundle $C$, and $t = HMAC_{Sd} \subset \text{term (n)}$ is a new component of $n$ with $K \not\in P$. Then the positive node $m= <S, 7>$ such that $m \not\leq n$, where $t$ is uniquely originating at $m$.

Lemma 6: If $K_{AS} \not\in P$, then node $m= <U, 5>$ is an unsolicited test for $t = [R_{temp1}, PW_{IoT}]_{K_{AS}}$ 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 $t \subset m; t \not\in n$ for all $n$ such that $m \not\leq n$.

Lemma 7: If $K_{IoT} \not\in P$, then node $m= <D, 4>$ is an unsolicited test for $t = [R_{temp2}, ID_{IoT}]_{K_{IoT}}$ 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 $t \subset m; t \not\in n$ for all $n$ such that $m \not\leq n$.

Lemma 8: If $K_{temp} \not\in P$, then node $m= <D, 5>$ 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 $t \subset m; t \not\in n$ for all $n$ such that $m \not\leq n$.

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 (kus) 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).

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.

Figure 2. User role

Figure 4. User role

Figure 5. Server 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.
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 \(kis\).
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 $R_u$ 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 $K_u$, then $S$ authenticates $U$.
   b) The `authentication_on auth_2` represents the random number $R_s$ 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 $K_s$, then $U$ authenticates $S$.
   c) The `authentication_on auth_3` represents that the random number $R_s$ 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 $K_d$, then $D$ authenticates $S$.

2) Three secrecy goals:
   a) The `secrecy_of sec_4` denotes that the random number $R_u$ is kept secret only to $U$ and $S$.
   b) The `secrecy_of sec_5` denotes that the random number $R_s$ is kept secret only to $S$ and $U$.
   c) The `secrecy_of sec_6` denotes that the random number $R_s$ is kept secret only to $D$ and $S$.

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.
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


Formal security analysis of lightweight authenticated key agreement protocol for IoT... (Ahmed H. Aly)

**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.