Defending Industry 4.0: An Enhanced Authentication Scheme for IoT Devices

Nasour Bagheri, Saru Kumari, Carmen Camara, and Pedro Peris-Lopez

***Abstract*—To address the security concerns of Industry 4.0, recently, Garg *et al.* proposed a lightweight authentication protocol, and Akram *et al.* showed some of its security drawbacks. We continue this line by exposing how Garg *et al.*'s protocol suffers from noninvasive and invasive attacks. First, we explain that a passive attacker can trace any two communicating nodes to compromise their location privacy. Next, we show that an active though noninvasive adversary can compromise the integrity of the exchanged messages without being detected and run a de-synchronization attack. Besides, the adversary can extract any shared session key from any pair of nodes in the protocol. We named this attack a pandemic session key disclosure attack, and its consequences are more harmful than the impersonation of a compromised node. Finally, we disclose how the proposed scheme does not guarantee the privacy protection for the keys when we assume an honest but curious server. To overcome those existing security flaws, we finally propose a revised protocol called TARDIGRADE. First, our informal analysis, and then, our formal security analysis using the real-or-random model shows that TARDIGRADE provides the desired security, and likewise, our performance analysis confirms a reasonable cost compared with Garg *et al.*'s protocol.**

***Index Terms*—Authentication, Industry 4.0, Internet of Things (IoT), noninvasive adversary, pandemic session key-disclosure attack, privacy, security, traceability.**

I. Introduction

INDUSTRY 4.0 is a subset of the fourth industrial revolution that is more dealing with smart technologies that are related to the industry, in which the Internet of Things (IoT) can also play a relevant role and everything will be smart and connected. However, there are several challenges to deploy IoT technology in the Industry 4.0 setting, and one vital challenge is the different cyber threats, physical attacks, or both, that are targeting the IoT devices. To dealing with these concerns, as an example, Esfahani *et al.* [1] proposed a web authentication mechanism to prevent man-in-the-middle attacks in Industry 4.0 supply chains. Radanliev *et al.* [2] presented a systematic synthesis of the literature related to the impact of IoT-based supply chains and their related cyber risks. Sengupta *et al.* [3] proposed a fog-based architecture to provide the desired security for Industrial IoT (IIoT) and Industry 4.0. Chamikara *et al.* [4] devoted their study to introduce a framework for reaching privacy and reliability in the IIoT. Zhang *et al.* [5] proposed an anonymous batch authentication scheme for smart vehicular networks, as a type of IIoT, and Zhao and Dong [6] proposed an entropy-based feature selection method for IIOT. Some research articles, e.g., Lins and Oliveira [7], also suggested the use of software-defined networks (SDN) in Industries 4.0. However, the SDN has its drawbacks when it comes to security [8]. To address the security concerns of employing IoT-based smart devices in Industry 4.0, Garg *et al.* [9] recently proposed a lightweight mutual authentication and key agreement protocol, which is more efficient compared to the previous related works. Unfortunately, later Akram *et al.* [10] have shown that the scheme does not provide the desired security under a (semi-)invasive adversarial model. In this article, we continue this line by presenting several new powerful attacks (including some noninvasive) against Garg *et al.* protocol. Then, we propose an enhanced version, called TARDIGRADE, to remedy the known and harmful attacks against the original protocol.

A. Motivation

To provide desired security for different applications, designers always propose new solutions, including cryptography protocols. For example, Wei *et al.* [11] introduce a hierarchical attribute-based access scheme for e-health; in [12], Erdem and Sandikkaya use the one-time password as a service to achieve desired security for cloud users; Han *et al.* [13] propose an approach to help the criminal investigators to recognize the roles of online illegal gambling participants. On the other hand, it is widely accepted that any new security proposal solution should not be trusted exclude it has been enough evaluated by independent third parties. On the other hand, the previous security analysis on Garg *et al.* [9], conducted by Akram *et al.* [10] did not shed light on all security flaws of this protocol. All

Manuscript received 6 March 2021; revised 1 July 2021, 5 October 2021, and 11 November 2021; accepted 25 November 2021. Date of publication 24 December 2021; date of current version 26 August 2022. This work was supported in part by Leonardo Grant for Researchers and Cultural Creators, BBVA Foundation under Grant P2019-CARDIOSEC, in part by the Spanish Ministry of Science, Innovation and Universities under Grant PID2019-111429RBC21(ODIO), in part by the Comunidad de Madrid (Spain) under Grant CYNAMON (P2018/TCS-4566), and in part by European Structural Funds (ESF and FEDER). The work of Nasour Bagheri was supported in part by the Iran National Science Foundation (INSF) under Grant 98010674. *(Corresponding author: Saru Kumari.)*

Nasour Bagheri is with Electrical Engineering Department, Shahid Rajaee Teacher Training University (SRTTU), Tehran 16788-15811, Iran, and also with the School of Computer Science (SCS), Institute for Research in Fundamental Sciences (IPM), Farmanieh Campus, Tehran 19538-33511, Iran (e-mail: nbagheri@sru.ac.ir).

Saru Kumari is with the Department of Mathematics, Chaudhary Charan Singh University, Meerut 250004, India (e-mail: saryusiirohi@gmail.com).

Carmen Camara and Pedro Peris-Lopez are with Computer Science Department, University Carlos III of Madrid, 28911 Madrid, Spain (e-mail: macamara@pa.uc3m.es; pperis@inf.uc3m.es).

Digital Object Identifier 10.1109/JSYST.2021.3131689

of the aforementioned studies motivated us to do a more detailed analysis of this protocol.

In addition, any cryptographic protocol should not leakage any information if one switches from a client to another client. The aforementioned property is essential when considering a case where the adversary can control a client by introducing it as a malicious client to the network or even compromising it. Hence, it is a realistic assumption that the adversary can control a node. However, this control should not give any advantage to the adversary to compromise the security of other clients in that network. Although the security target, in this case, could be different from a protocol to another, if the protocol is ideally secure, the adversary should not gain any advantage, excluding the controlled client. This property motivated us to analyze the security of Garg *et al.* [9] when the adversary can control a client node. In our adversarial model, the adversary aims to reveal the established session key between two uncompromised nodes assuming that it has control over a node in the network.

Following our analysis and also related literature, e.g., [10], relay on the stored secret parameter on the client side does not provide a high level of security. Hence, several multifactor authentication schemes and key establishment protocols have been proposed so far for different applications, e.g., [14] and [15]. However, in the case of industry 4.0, we cannot simply accommodate a multifactor authentication solution since there are many uncontrolled devices in the network. In this case, a promising approach could be to introduce the device fingerprint into the authentication process as a security factor. This fingerprint could be an embedded physically unclonable function (PUF) in the devices. Hence, to provide reasonable security against compromised devices, we use a PUF in the proposed protocol.

B. Our Contribution

Our contribution in this article is threefold, as described in the following.

1) We provide a more detailed security analysis of Garg *et al.*'s protocol than that presented by Akram *et al.* For example, we show how the proposed protocol does not protect privacy (data location) either messages integrity.
2) We introduce an attack model that we name as pandemic session key-disclosure attack. We show how the adversary can extract any shared session key of any nodes pair in the protocol under an adversary pandemic approach.
3) To overcome the security flaws of Garg *et al.*'s protocol, we propose a revised protocol (named `TARDIGRADE`) and prove its security under the real or random model.

C. Paper Organization

The rest of this article is organized as follows. In Section II, we introduce the required preliminaries, including the adversarial model and a brief description of Garg *et al.*'s protocol; in Section III, we present a variety of attacks against the original scheme; then, we propose an enhanced protocol, `TARDIGRADE`, in Section IV, and provide its security and cost analysis in Section VI. Finally, Section VII concludes this article.

TABLE I
LIST OF USED NOTATIONS

Symbol	Description
E	Elliptic curve under predefined parameters
$\mathbb{G}$	A finite Prime Field
P	Generator point of a large group G
q	A large prime number
N_i	i-th IoT node
S	A trusted server
ID_i	The unique identifier of N_i
d_S/d_i	Private key of S/N_i
Q_S/Q_i	Public key of S/N_i
r_i	A random number
$<\mathcal{C},\mathcal{R}>$	A Challenge and Response Pairs (CRPs)
$H(.)$	One-way hash function
$H^r(.)$	Employing $H(.)$ in random number generation mode to generate a random sequence of desired length
$\oplus$	Bitwise XOR operation
$a.P$	Multiplying a point P on the elliptic curve $\mathcal{E}$ by natural number (scalar) a, results in another point on the curve
$\|$	Concatenation
TS_x	A timestamp generated by x
$A \stackrel{?}{=} B$	Determine whether $\mathcal{A}$ and B are equal
SK_{ij}	The shared session key between N_i and N_j
$\|X\|$	Cardinality of the set/variable X
$Auth_y^x$	A token generated by the party x to be verified by the party y.

II. Preliminaries

A. Garg et al.'s Protocol

The Garg *et al.*'s protocol includes three phases: initialization; registration; and mutual authentication and key agreement phases, respectively [9]. To describe this protocol, we use the list of notations represented in Table I. In the initialization phase, each node counts with a PUF, and the trusted server discloses the protocol's parameters, including the elliptic curve $\mathcal{E}$ and its parameters $P, q, \mathbb{G}, a$, and b. It also selects its privet key $d_s \in Z_q^*$ and computes its public key, i.e., $Q_s = d_s.P$. In the registration phase, the node N_i generates a unique identity ID_i for itself and sends it to S.

Then, S generates a pair $(d_i, Q_i = d_i.P)$ as the N_i's private and public keys. It also generates two pairs $<\mathcal{C}_{i1}, \mathcal{R}_{i1}>$ and $<\mathcal{C}_{i2}, \mathcal{R}_{i2}>$ as challenge and response pairs (CRPs) and shares the token $(<\mathcal{C}_{i1}, \mathcal{R}_{i1}>, <\mathcal{C}_{i2}, \mathcal{R}_{i2}>, d_i)$ with N_i.

The mutual authentication and key agreement phase of the protocol executes as follows, via the trusted sever S.

1) N_i generates a random number $r_i \in Z_q^*$, computes $R_i = r_i.P$, generates the timestamp TS_i and sends $M_1 =< \text{ID}_i, \text{ID}_j, \text{TS}_i, R_i >$ to S.
2) Once received M_1, S verifies TS_i, selects two random CRPs for N_i and N_j, e.g., $<\mathcal{C}_{i1}, \mathcal{R}_{i1}>$ and $<\mathcal{C}_{i2}, \mathcal{R}_{i2}>$ for N_i and $<\mathcal{C}_{j1}, \mathcal{R}_{j1}>$ and $<\mathcal{C}_{j2}, \mathcal{R}_{j2}>$ for N_j. Next, it generates a random number $r_s \in Z_q^*$ and its timestamp TS_s, calculates $R_s = r_s.$

P, $\text{TK}_s = d_S.R_i$, $\text{TK}_i^* = r_s.Q_i$, $\text{Auth}_s = H(\mathcal{C}_{i1}\|\mathcal{C}_{i2}\|\text{TS}_i\|\text{TS}_s\|\text{TK}_s)$, $C_{i2}^* = \mathcal{C}_{i2} \oplus H(\mathcal{R}_{i1}\|\text{ID}_i\|\text{ID}_j\|\text{TS}_s\|\text{TK}_i^*)$, and $\text{SK}_{\text{info}}^i = \mathcal{C}_{i2} \oplus H(\mathcal{C}_{i2}\|\mathcal{C}_{j2}\|r_s\|d_s)$. Finally, S sends $M_2 =< \mathcal{C}_{i1}, C_{i2}^*, R_s, \text{Auth}_s, \text{TS}_s, \text{SK}_{\text{info}}^i >$ to N_i.

3) Once received M_2, N_i verifies TS_s, extracts $\mathcal{C}_{i2}$ from C_{i2}^* and $\mathcal{R}_{i1}$ from $\mathcal{C}_{i1}$, computes Auth_s^* based on the received data and verifies whether $\text{Auth}_s^* \stackrel{?}{=} \text{Auth}_s$ to authenticate S. It also derives its session key as $\text{SK}_{ij} = H(\text{ID}_i\|\text{ID}_j\|\text{TS}_s\|(\text{SK}_{\text{info}}^i \oplus \mathcal{C}_{i2}))$, which is used only once the mutual authentication between N_i and N_j is established and confirmed. Next, N_i generates its current timestamp TS_i^*, computes $\text{Auth}_i = H(\mathcal{R}_{i1}\|\mathcal{R}_{i2}\|\text{TS}_i^*\|\text{TS}_s\|\text{TK}_i)$ and sends $M_3 =< \text{TS}_i^*, \text{Auth}_i >$ to S.

4) S verifies TS_i^*, calculates Auth_i^* based on the received data and verifies whether $\text{Auth}_i^* \stackrel{?}{=} \text{Auth}_i$ to authenticate N_i. Assuming N_i is legitimate and has been authenticated by S, the server initiates the process to check the N_j's authenticity.

5) S generates its current timestamp TS_s^*, computes $\text{TK}_j^* = r_s.Q_j$, $C_{j2}^* = \mathcal{C}_{j2} \oplus H(\mathcal{R}_{j1}\|\text{ID}_j\|\text{ID}_i\|\text{TS}_s^*\|TK_j^*)$ and $\text{SK}_{\text{info}}^j = \mathcal{C}_{j2} \oplus H(\mathcal{C}_{i2}\|\mathcal{C}_{j2}\|r_s\|d_s)$. Then, S sends $M_4 =< \mathcal{C}_{j1}, C_{j2}^*, R_s, \text{TS}_s^*, \text{SK}_{\text{info}}^j, \text{ID}_i, TS_S >$ to N_j.

6) Once received M_4, N_j verifies TS_s^*, extracts $\mathcal{C}_{j2}$ from C_{j2}^* and $\mathcal{R}_{j1}$ from $\mathcal{C}_{j1}$, and derives its session key as $\text{SK}_{ij} = H(\text{ID}_i\|\text{ID}_j\|\text{TS}_s\|(\text{SK}_{\text{info}}^j \oplus \mathcal{C}_{j2}))$, which is used only once the mutual authentication between N_i and N_j is established and confirmed. Next, N_j generates its current timestamp TS_j and a random number $r_j \in Z_q^*$, computes $\mathcal{R}_j = r_j.P$ and $\text{Auth}_j = H(\mathcal{R}_{j1}\|\mathcal{R}_{j2}\|\text{TS}_j^*\|\text{TS}_s^*\|\text{TK}_j)$ and sends $M_5 =< \text{TS}_j, R_j, \text{Auth}_j >$ to S.

7) S verifies TS_j, regenerates Auth_j^* based on the received data and verifies whether $\text{Auth}_j^* \stackrel{?}{=} \text{Auth}_j$ to authenticate N_j. Assuming N_j is legitimate and has been authenticated by S, the server generates the current timestamp TS_s^{**}, computes $\text{TK}_s = d_s.R_j$ and $\text{Auth}_s = H(\mathcal{C}_{j1}\|\mathcal{C}_{j2}\|\text{TS}_s^{**}\|\text{TK}_s)$ and sends $M_6 =< \text{TS}_s^{**}, \text{Auth}_s >$ to N_j.

8) Once received M_6, N_j verifies TS_s^{**} and checks whether $\text{Auth}_s^* \stackrel{?}{=} \text{Auth}_s$ to authenticate S.

B. Adversary Model

Through our study, we assume a probabilistic polynomial time active adversary with complete access to the transmitted messages passed over the public channels by the protocol parties. As a result, the attacker can eavesdrop on the exchanged messages, modify them, store and replay them later, or attempt to impersonate any of the protocol parties. Furthermore, the attacker has access to the public parameters of the protocol, such as the participants' public keys. This adversary model is based on the Dolev–Yao (DY) adversarial model [16]. Besides, given that the adversary may access the nodes and read their memory, we suppose that the attacker can compromise a target client in offline mode (nonactive session) and disclose the stored information in its nonvolatile memory, including the secret key. In an active session, however, the attacker has no access to the internal values. As a result, the attacker can only access the temporary values of a legitimate session. For the privacy model, in this article, we use Phan's traceability model [17]. It is a reformulated version of the seminal model that was initially proposed by Juels and Weis [18].

1) Semantic Security in the Real-or-Random Model: To share a session key in a three-party authenticated key agreement scheme, instances use their long-term secrets to share a session key sk, where a protocol's party could be either a client $N \in \mathcal{N}$ or a trusted server $S \in \mathcal{S}$. A client N, honest or malicious, holds a long-lived key sk_N. For each client N, the server S holds a transformation of sk_N, e.g., $sk_S[U]$, in a vector $sk_S =< sk_S[N] >_{N\in\mathcal{N}}$. If two clients N_i and N_j share the same session data we call them *partner*.

To determine the adversary's ability to distinguish a real session key agreement from a random one, at the beginning of the experiment, a bit b is chosen uniformly at random where $b = 0$ defines the random world (RW) and $b = 1$ represents the real word (target scheme). Following the DY adversary model [16], $\mathcal{A}$ can run the following query types [19] to distinguish the real world from the random world.

1) $\mathrm{Execute}$. It models a passive adversary $\mathcal{A}$, who eavesdrops on the channel and gets read access to the transferred messages between S and the involved nodes.
2) Send. It models an active adversary who may intercept a message, and then, either modifies it, creates a new one, or forwards it to S.
3) $\mathrm{Reveal}(N_i)$ query. It outputs the session key held by the node N_i, when a session key is assigned to N_i and Test query was not requested from either N_i or its partner.
4) $\mathrm{Test}(N_i)$. Depending on the model, its target could be determining the session key or distinguishing/tracing the scheme used or its instances. In the former case, if no session key for the node N_i is defined or if a Reveal query was asked to either N_i or to its partner, then it returns the undefined symbol $\perp$. Otherwise, it returns the session key for the node N_i if $b = 1$ or a random of key of the same size if $b = 0$. For the latter case, the tokens $\{M_1^{ij}, M_2^{ij}, M_3^{ij}, \ldots\}$ are returned if $b = 1$ and a random sequence of the same size if $b = 0$.

Let us assume a protocol $\mathcal{P}$, in which $\mathcal{A}$ has access to the $\mathrm{Execute}$, Send, and Test oracles, and outputs a guess bit b_0. The adversary wins the game, defining the semantic security in the real-or-random (RoR) sense, if $b_0 = b$, where b is the hidden bit used in the Test oracle. The adversary's advantage to win this game, $\text{Adv}_{\mathcal{D},\mathcal{P}}^{\text{RoR}}(t, R)$, is defined as follows:

$$\text{Adv}_{\mathcal{D},\mathcal{P}}^{\text{RoR}}(t, R) = ((\Pr(\mathcal{A} \to b_0 = 1 : b = 1) \\ (\Pr(\mathcal{A} \to b_0 = 1 : b = 0)))$$

$\mathcal{P}$ offers RoR semantic security if the aforementioned advantage is insignificant. Mathematically

$$\text{Adv}_{\mathcal{D},\mathcal{P}(t,R)}^{\text{RoR}} < \varepsilon(.)$$

and $\varepsilon(.)$ being some negligible function.

In supplement to the aforementioned, the privacy of the keys in a three-party key exchange protocol is critical. Following the privacy model proposed by Abdola *et al.* [19], the privacy of the shared key concerning the server must be guaranteed. More precisely, we want to trust as little as possible on the third parties, and the server should be considered an honest but curious entity. Hence, although the server's participation is required to establish a session key between two nodes in the underlying IoT system, the server should not be able to achieve any information on the value of that session key.

2) Pandemic Session Key Disclosure Attack: In pandemic session key disclosure attack, we assume the adversary compromises a client $N_i \in \mathcal{N}$ and aims to establish a session key with $N_j \neq N_i$ as $N_f \notin \{N_i, N_j\}$. The access to the information related to N_i could be granted by $\mathrm{Reveal}(N_i)$ query type. Hence, our adversary has access to the $\mathrm{Execute}$, Send, and Test oracles, can also request a single Reveal, and aims to establish a session key $\mathrm{SK}_A j$ with N_j. The adversary's advantage to win this game $\mathrm{Adv}^{\mathrm{Pand}}_{N_i,\mathrm{SK}_{N_j-N_f},\mathcal{P}}(t,R)$, is defined as follows:

$$\mathrm{Adv}^{\mathrm{Pand}}_{N_i,\mathrm{SK}_{N_j-N_f},\mathcal{P}}(t,R) = \big((\Pr(\mathcal{A}^{N_i} \to \mathrm{SK}_{N_j-N_f}) \\ (\Pr(\mathcal{A} \to \mathrm{SK}_{N_j-N_f}))\big).$$

$\mathcal{P}$ offers RoR semantic security if the aforementioned advantage is insignificant. Mathematically

$$\mathrm{Adv}^{\mathrm{Pand}}_{N_i,\mathrm{SK}_{N_j-N_f},\mathcal{P}}(t,R) < \varepsilon(.)$$

and $\varepsilon(.)$ being some negligible function, which means that compromising N_i does not help the adversary to establish a session key with N_f as N_j.

III. Security Analysis of Garg *et al.*

We first throw a little light on Akram *et al.*'s comment on the security of Garg *et al.*'s protocol. Akram *et al.* [10] showed that in a (semi-)invasive adversarial model in which the adversary can access the N_i's memory, the attacker could disclose the secret value d_i and also its CRPs. Given that information, then it is straightforward to impersonate N_i at any time. They also suggested a remedy to fix this security hole, which could also be a solution for any other similar protocol in which the secret values are directly stored in the node's memory. The solution consists of holding a randomized version of the private data in the memory of the node. These values, when necessary, can be reordered to their correct form by a dedicated assembly code. Unfortunately, if the adversary can access the program data of the N_i's processor (e.g., a microcontroller), the adversary could compromise the assembly instructions, and consequently, discloses the secret values. Besides, we want to highlight that Akram *et al.* [10] do not claim a full-proof solution and only gave some indications.

To continue in this vein, we highlight other security pitfalls of Garg *et al.*'s protocol as follows.

1) Similarly to [10], we assume the adversary can compromise N_i, and consequently, achieve its memory records (i.e., $(<\mathcal{C}_{i1}, \mathcal{R}_{i1}>, <\mathcal{C}_{i2}, \mathcal{R}_{i2}>, d_i)$). We also assume that the server S uses constant and preshared CRPs for each node; otherwise, the protocol does not work as was already mentioned in [10]. Then, we extend the Akram *et al.*'s (semi-)invasive attack to what we name as a pandemic session key disclosure attack. Under this pandemic approach, the adversary can disclose the session key even if the protocol runs between noncompromised nodes.
2) We show how the protocol presents security pitfalls even under a noninvasive adversarial model. The adversary succeeds in a traceability attack and can compromise the integrity of the messages exchanged in the protocol.

The proposed attacks are mainly based on the observations described as follows.

1) *Observation-1:* In a key agreement session between N_i and N_j, the S server sends $\mathrm{SK}^i_{\mathrm{info}} = \mathcal{C}_{i2} \oplus H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s)$ to N_i and $\mathrm{SK}^j_{\mathrm{info}} = \mathcal{C}_{j2} \oplus H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s)$ to N_j. Assuming that the adversary has compromised N_i, he can extract $H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s)$ from $\mathrm{SK}^i_{\mathrm{info}}$, and therefore, $\mathcal{C}_{j2}$ from $\mathrm{SK}^j_{\mathrm{info}}$.
2) *Observation-2:* In step 1 of the protocol, the N_i node sends its identifier and the identifier of N_j in plain text over a public and insecure channel.
3) *Observation-3:* In steps 2 and 3 of the protocol, the S server sends the messages related to the shared key to the N_i and N_j nodes, respectively. However, the integrity of these messages is not guaranteed.

A. Pandemic Session Key Disclosure Attack

Following the Akram *et al.* node impersonation attack, we assume that N_i node is connected to S server to establish a session key with N_j node. In the authentication procedure, S sends $\mathrm{SK}^i_{\mathrm{info}} = \mathcal{C}_{i2} \oplus H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s)$ to N_i as part of M_2 message, and $\mathrm{SK}^j_{\mathrm{info}} = \mathcal{C}_{j2} \oplus H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s)$ to N_j as part of M_4 message. Following the observation-1 given in step 1, the adversary can extract $H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s)$ as $\mathcal{C}_{i2} \oplus \mathrm{SK}^i_{\mathrm{info}}$ and $\mathcal{C}_{2j}$ as $H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s) \oplus \mathrm{SK}^j_{\mathrm{info}}$. Next, assuming that N_f node ($N_f \neq N_i$) aims to share a session key with N_j node, the mutual authentication and key agreement phase of the protocol process is as follows, via S server.

1) N_f generates a random number $r_f \in Z^*_q$, calculates $R_f = r_f.P$, generates the timestamp TS_f and sends the tuple $M_1 =< \mathrm{ID}_f, \mathrm{ID}_j, \mathrm{TS}_f, R_f >$ to S.
2) Once received M_1, S verifies TS_f, selects two random CRPs for N_f and N_j, e.g., $< \mathcal{C}_{f1}, \mathcal{R}_{f1} >$ and $< \mathcal{C}_{f2}, \mathcal{R}_{f2} >$ for N_f and $< \mathcal{C}_{j1}, \mathcal{R}_{j1} >$ and $< \mathcal{C}_{j2}, \mathcal{R}_{j2} >$ for N_j. Next, it generates a random number $r_s \in Z^*_q$ and its timestamp TS_s, computes $R_s = r_s.P$, $\mathrm{TK}_s = d_S.R_f$, $\mathrm{TK}_f = r_s.Q_f$, $\mathrm{Auth}_s = H(\mathcal{C}_{f1} \| \mathcal{C}_{f2} \| \mathrm{TS}_f \| \mathrm{TS}_s \| \mathrm{TK}_s)$, $C^*_{f2} = \mathcal{C}_{f2} \oplus H(\mathcal{R}_{f1} \| \mathrm{ID}_f \| \mathrm{ID}_j \| \mathrm{TS}_s \| \mathrm{TK}^*_f)$ and $\mathrm{SK}^f_{\mathrm{info}} = \mathcal{C}_{f2} \oplus H(\mathcal{C}_{f2} \| \mathcal{C}_{j2} \| r_s \| d_s)$. Finally, S sends $M_2 =< \mathcal{C}_{f1}, C^*_{f2}, R_s, \mathrm{Auth}_s, \mathrm{TS}_s, \mathrm{SK}^i_{\mathrm{info}} >$ to N_f.
3) Once received M_2, N_f verifies TS_s, extracts $\mathcal{C}_{f2}$ from C^*_{f2} and $\mathcal{R}_{f1}$ from $\mathcal{C}_{f1}$. It also calculates Auth^*_s based on the received data and verifies whether $\mathrm{Auth}^*_s \stackrel{?}{=}$

Auth_s to authenticate S. Besides, it derives its session key as $\mathrm{SK}_{fj} = H(\mathrm{ID}_f \| \mathrm{ID}_j \| \mathrm{TS}_s \| (\mathrm{SK}^f_{\text{info}} \oplus \mathcal{C}_{f2}))$, which is used only once the mutual authentication between N_f and N_j is established and confirmed. Finally, N_f generates its current timestamp TS^*_f, computes $\mathrm{Auth}_f = H(\mathcal{R}_{f1} \| \mathcal{R}_{f2} \| \mathrm{TS}^*_f \| \mathrm{TS}_s \| \mathrm{TK}_f)$ and sends $M_3 =< \mathrm{TS}^*_f, \mathrm{Auth}_f >$ to S.

4) S checks the validity of TS^*_f, computes Auth^*_f based on the received data and verifies whether $\mathrm{Auth}^*_f \stackrel{?}{=} \mathrm{Auth}_f$ to authenticate N_f. Assuming N_f is legitimate and has been authenticated by S, the server initiates the process to check the N_j's authenticity.
5) S generates its current timestamp TS^*_s, computes $\mathrm{TK}^*_j = r_s.Q_j$, $C^*_{j2} = \mathcal{C}_{j2} \oplus H(\mathcal{R}_{j1} \| \mathrm{ID}_j \| \mathrm{ID}_f \| \mathrm{TS}^*_s \| TK^*_j)$ and $\mathrm{SK}^j_{\text{info}} = \mathcal{C}_{j2} \oplus H(\mathcal{C}_{f2} \| \mathcal{C}_{j2} \| r_s \| d_s)$. Next, S sends $M_4 =< \mathcal{C}_{j1}, C^*_{j2}, R_s, \mathrm{TS}^*_s, \mathrm{SK}^j_{\text{info}}, \mathrm{ID}_f, \mathrm{TS}_S >$ to N_j.
6) Once received M_4, N_j verifies TS^*_s, extracts $\mathcal{C}_{j2}$ from C^*_{j2} and $\mathcal{R}_{j1}$ from $\mathcal{C}_{j1}$, and derives its session key as $\mathrm{SK}_{fj} = H(\mathrm{ID}_f \| \mathrm{ID}_j \| \mathrm{TS}_s \| (\mathrm{SK}^j_{\text{info}} \oplus \mathcal{C}_{f2}))$, which is used only once the mutual authentication between N_f and N_j is established and confirmed. Then, N_j generates its current timestamp TS_j and a random number $r_j \in Z^*_q$, computes $\mathcal{R}_j$ and $\mathrm{Auth}_j = H(\mathcal{R}_{j1} \| \mathcal{R}_{j2} \| \mathrm{TS}^*_j \| \mathrm{TS}^*_s \| \mathrm{TK}_j)$, and finally, sends $M_5 =< \mathrm{TS}_j, R_j, \mathrm{Auth}_j >$ to S.
7) S checks the validity of TS_j, calculates Auth^*_j based on the received data, and verifies whether $\mathrm{Auth}^*_j \stackrel{?}{=} \mathrm{Auth}_j$ to authenticate N_j. Assuming N_j is legitimate and authenticated by S, the server generates the current timestamp TS^{**}_s, computes $\mathrm{TK}_s = d_s.R_j$ and $\mathrm{Auth}_s = H(\mathcal{C}_{j1} \| \mathcal{C}_{j2} \| \mathrm{TS}^{**}_s \| \mathrm{TK}_s)$, and finally, sends $M_6 =< \mathrm{TS}^{**}_s, \mathrm{Auth}_s >$ to N_j.
8) Once received M_6, N_j verifies TS^{**}_s, computes Auth^*_s based on the received data and verifies whether $\mathrm{Auth}^*_s \stackrel{?}{=} \mathrm{Auth}_s$ to authenticate S.
9) Assuming that S is also legitimate and authenticated, the mutual authentication and key agreement process end successfully.
10) Given M_1, M_2, and M_4, the adversary can retrieve $\mathrm{ID}_f, \mathrm{ID}_j, \mathcal{C}_{f1}, \mathcal{C}_{j1}, \mathrm{TS}_s, \mathrm{SK}^f_{\text{info}} = \mathcal{C}_{f2} \oplus H(\mathcal{C}_{f2} \| \mathcal{C}_{j2} \| r_s \| d_s)$, and $\mathrm{SK}^j_{\text{info}} = \mathcal{C}_{j2} \oplus H(\mathcal{C}_{f2} \| \mathcal{C}_{j2} \| r_s \| d_s)$. Besides, given $\mathcal{C}_{j2}$ from the node impersonation attack between N_i and N_j, the adversary extracts $H(\mathcal{C}_{f2} \| \mathcal{C}_{j2} \| r_s \| d_s) = \mathrm{SK}^j_{\text{info}} \oplus \mathcal{C}_{j2}$, $\mathcal{C}_{f2} = \mathrm{SK}^f_{\text{info}} \oplus H(\mathcal{C}_{f2} \| \mathcal{C}_{j2} \| r_s \| d_s)$ and $\mathrm{SK}_{fj} = H(\mathrm{ID}_f \| \mathrm{ID}_j \| \mathrm{TS}_s \| H(\mathcal{C}_{f2} \| \mathcal{C}_{j2} \| r_s \| d_s))$.

Following the aforementioned attack, the adversary could retrieve the session key of two noncompromised nodes successfully, i.e., $\mathrm{SK}_{fj} = H(\mathrm{ID}_f \| \mathrm{ID}_j \| \mathrm{TS}_s \| H(\mathcal{C}_{f2} \| \mathcal{C}_{j2} \| r_s \| d_s))$. More interestingly, the adversary could also disclose $\mathcal{C}_{f2}$ that allows extracting the session key between N_f and any other node (i.e., N_*). Consequently, the adversary can gain the agreed session key between any pair of nodes in the IoT network by compromising a single node of the network. In this way, the node impersonation attack turns out to be a pandemic attack.

B. Traceability Attack

Suppose that the N_i node communicates with the S server. Following the security model given in Section II-B1, if an adversary can link the messages transferred between the aforementioned two entities, over the public channel and in different sessions, with a nonnegligible probability p, we can claim that the target protocol is vulnerable to traceability. It is a paramount concern because it compromises the privacy location of the protocol's parties. Next, we show how Garg *et al.*'s protocol puts at risk the privacy (location) of IoT nodes since an adversary can track them with probability "1" (maximum adversary advantage), as described in the following steps.

1) Phase 1 (Learning): $\mathcal{A}$ sends an $\mathrm{Execute}(S, N_0, t)$ query and acquires the public message $M_1 =< \mathrm{ID}_0, \mathrm{ID}_j, \mathrm{TS}_0, R_0 >$ passed over the insecure channel. Then, $\mathcal{A}$ stores ID_0 as a static search index, which is linked to N_0.
2) Phase 2 (Challenge): $\mathcal{A}$ chooses two fresh nodes $\{N_0, N_1\}$ whose associated identifiers are ID_0 and ID_1, respectively. Next, he sends a $\mathrm{Test}(t', N_0, N_1)$ query. As a result, and depending on a chosen random bit $b \in \{0, 1\}$, $\mathcal{A}$ is given a static search index $Y =< \mathrm{ID}_b, \mathrm{ID}_{f/g}, \mathrm{TS}_b, R_b >$ from the set $\{< \mathrm{ID}_0, \mathrm{ID}_f, \mathrm{TS}_0, R_0 >, < \mathrm{ID}_1, \mathrm{ID}_g, \mathrm{TS}_1, R_1 >\}$.
3) Phase 3 (Guessing): $\mathcal{A}$ finishes $\mathcal{G}$ and outputs a bit $\widetilde{b}$ as its conjecture of the value b. In particular, $\mathcal{A}$ utilizes the following simple but effective decision rule:

$$\begin{cases} \text{if } \mathrm{ID}_b == \mathrm{ID}_0 & \widetilde{b} = 0 \\ \text{if } \mathrm{ID}_b \neq \mathrm{ID}_0 & \widetilde{b} = 1. \end{cases} \tag{1}$$

It is clear that the following equation gives the adversary advantage:

$$\mathrm{Adv}^{\mathsf{UNT}}_{\mathcal{A}}(1,1) = \left| Pr[\widetilde{b} = b] - \frac{1}{2} \right| = 1 - 0.5 = 0.5$$

which is the maximum advantage that an adversary can get in this traceability model. We can use a similar approach to track any other N_* node. Therefore, the proposed protocol by Garg *et al.* provides the worse security concerning the traceability attack and should not be used in an application in which the nodes' location privacy is essential. Besides the aforementioned, we should note that through the pandemic attack, $\mathcal{C}_{i1}$ and $\mathcal{C}_{i2}$ values are disclosed and can be the sources for a traceability attack. Moreover, the fixed R_s in M_2 and M_4 is also a source of traceability to distinguish two communicating nodes.

C. Desynchronization and Integrity Attacks

Assume that an active adversary $\mathcal{A}$ can control the messages transferred between S, N_i, and N_j. $\mathcal{A}$ replaces the $\mathrm{SK}^i_{\text{info}}$ value sent from S to N_i by $\mathrm{SK}^i_{\text{info}} \oplus \Delta$, for any arbitrary $\Delta \neq 0$. This act by the adversary is undetectable during the authentication between S and N_j, then S, N_i, and N_j believe that the mutual authentication and key agreement process is completed successfully. However, at the end of this process, the session key in the N_j side is $\mathrm{SK}_{ij} = H(\mathrm{ID}_i \| \mathrm{ID}_j \| \mathrm{TS}_s \| H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s))$, while N_i computes it as

$\text{SK}_{ij} = H(\text{ID}_i \| \text{ID}_j \| \text{TS}_s \| (H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s) \oplus \Delta))$. Hence, the involved nodes could not communicate properly and the attacker successes in a desynchronization attack. Also, the attacker could maintain the synchronization between both nodes, but the integrity of the keys continues compromised. For this purpose, the attacker $\mathcal{A}$, apart from the aforementioned, can also change the message sent to N_j in step 3 accordingly, such that N_i and N_j both agree on $\text{SK}_{ij} = H(\text{ID}_i \| \text{ID}_j \| \text{TS}_s \| (H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s) \oplus \Delta)))$. The attack mentioned previously is serviceable to distinguish the Garg *et al.*'s protocol from an ideal protocol in which no message reveals any information. The adversary follows the procedure described as follows to distinguish Garg *et al.*'s protocol ($\mathcal{T}_P$) from a secure protocol ($\mathcal{S}_P$).

1) When S, respectively, sends M_2 to N_i (that includes $\text{SK}^i_{\text{info}}$) and M_4 (that includes $\text{SK}^j_{\text{info}}$) to N_j, the adversary replaces their $\text{SK}^i_{\text{info}}$ and $\text{SK}^j_{\text{info}}$ values by $\text{SK}^i_{\text{info}} \oplus \Delta$ and $\text{SK}^j_{\text{info}} \oplus \Delta$, respectively.
2) If the protocol finishes without any error and the communication between N_i and N_j ends successfully, the adversary concludes that he is observing the Garg *et al.*'s protocol; otherwise adversary concludes that it is a secure protocol ($\mathcal{S}_P$).

To determine the adversary's advantage, we have to consider two possible scenarios. First, it is clear that if the adversary communicates with Garg *et al.*'s protocol, then with a probability of "1," the procedure returns $\mathcal{T}_\mathcal{P}$ in step 2. Second, if the adversary communicates with $\mathcal{S}_P$, then the procedure returns $\mathcal{T}_\mathcal{P}$ with a probability of "2^{-l}," where l is the security parameter, e.g., output-length of the $H(.)$ hash function. Therefore, the advantage of the adversary is as follows:

$$\begin{aligned}\text{Adv}^{\mathsf{DIS}}_{\mathcal{A}} &= \left|\Pr\left[D^{\mathcal{T}_\mathcal{P}} = 1\right] - \Pr\left[D^{\mathcal{S}_P} = 1\right]\right| \\ &= 1 - 2^{-l}.\end{aligned}$$

Since the advantage is not negligible, it means that Garg *et al.*'s protocol has hazardous security flaws (information disclosure concerns) in the RoR model.

D. Shared Key Privacy

In Garg *et al.*'s protocol, the server generates $\text{SK}^i_{\text{info}} = \mathcal{C}_{i2} \oplus H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s)$ and $\text{SK}^j_{\text{info}} = \mathcal{C}_{j2} \oplus H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s)$, and then, shares these values with N_i and N_j, respectively. Using these tokens, N_i computes the shared key as $\text{SK}_{ij} = H(\text{ID}_i \| \text{ID}_j \| \text{TS}_s \| (\text{SK}^i_{\text{info}} \oplus \mathcal{C}_{i2}))$ and N_j computes it as $\text{SK}_{ji} = H(\text{ID}_i \| \text{ID}_j \| \text{TS}_s \| (\text{SK}^j_{\text{info}} \oplus \mathcal{C}_{j2}))$. Consequently, $\text{SK}_{ij} = \text{SK}_{ji} = H(\text{ID}_i \| \text{ID}_j \| \text{TS}_s \| H(\mathcal{C}_{i2} \| \mathcal{C}_{j2} \| r_s \| d_s))$. Unfortunately for the protocol designers, a closer look at the shared key shows that a curious server can also derive it straightforwardly since it has access to all the information pieces required to compute the mentioned key (SK_{ij}). Thus, an insider adversary on the server side can compromise the key shared between the existing nodes.

IV. TARDIGRADE, THE REVISED PROTOCOL

In this section, we propose TARDIGRADE as a revised version of the Garg *et al.*'s protocol to remedy its security flaws. In a nutshell and as a significant difference with the original protocol, in our proposed solution, we require that nodes can generate CRPs. We also assume that each node is equipped with a reliable PUF(.). It is worth noting that, in our design, we use a $H^r(\cdot)$ to denote a hash function in random number generation mode. For example, Sponge-based structures [20], [21], such as KECCAK [22] function, support this feature. Following this assumption, when computing $A \oplus H^r(B)$, it is possible to adapt the output length of $H^r(B)$ to mask the string A properly. Similar to Garg *et al.*'s protocol, TARDIGRADE includes three phases, i.e., initialization, registration, and mutual authentication and key agreement phases, respectively.

A. Initialization Phase

We keep the initialization phase intact, i.e., identical to the Garg *et al.*'s protocol.

B. Registration Phase

The registration phase of the protocol occurs over a secure channel. N_i generates an identity ID_i for itself and sends it to S. The server S accepts the identifier if it has not been already used by another node. Then, S generates a pair (d_i, $Q_i = d_i.P$) as the N_i's private and public keys, respectively. It also generates a sequence of t random challenges $\mathcal{C}_{i1}, \ldots, \mathcal{C}_{it}$ and sends the message $< \mathcal{C}_{i1}, \ldots, \mathcal{C}_{it}, d_i >$ to the N_i node. Once received the message, N_i stores the token $< \text{ID}_i, d_i >$ in its local memory, computes $\{\mathcal{R}_{iw} = \text{PUF}(\mathcal{C}_{iw})\}_{w=\{1,\ldots,t\}}$, and finally, sends $< \mathcal{R}_{i1}, \ldots, \mathcal{R}_{it} >$ to S. The server stores $(\text{ID}_i, Q_i, < \mathcal{C}_{i1}, \mathcal{R}_{i1} >, \ldots, < \mathcal{C}_{it}, \mathcal{R}_{it} >)$ in its encrypted database. (ID_i, Q_i) is also stored in a public database, accessible by any instance.

C. Mutual Authentication and Key Agreement Phase

We change the messages flow of the mutual authentication and key agreement phase of the protocol, between the N_i and N_j nodes via the S sever. We provide mutual authentication between the communicating nodes and facilitates the key agreement between them. This phase of the protocol consists of the following steps.

1) N_i generates its timestamp TS_i and calculates $r_i = \text{PUF}(\text{TS}_i) \bmod q$. If $r_i \neq 0$, it computes $R_i = r_i.P$ and $\text{TK}_s = r_i.Q_s$, and finally, sends the tuple $M_1 =< (\text{ID}_i, \text{ID}_j) \oplus H^r(\text{TK}_S, \text{TS}_i, R_i), \text{TS}_i, R_i >$ to the S server.
2) Once received M_1, S verifies TS_i and computes $\text{TK}_s = d_s.R_i$ and $H^r(\text{TK}_S, \text{TS}_i, R_i)$. Next, it extracts $(\text{ID}_i, \text{ID}_j)$, retrieves randomly a CRP pair $< \mathcal{C}_i, \mathcal{R}_i >$ for N_i and a CRP pair $< \mathcal{C}_j, \mathcal{R}_j >$ for N_j, respectively, from their records in the secure database. After this, it generates a random number $r_s \in Z^*_q$ and its timestamp TS_s and calculates $R_s = r_s.P$, $\text{TK}_j = r_s.Q_j$, $\text{SK}^j_{\text{info}} = \mathcal{R}_j \oplus H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s)$, $\text{Auth}^j_s = H(\mathcal{R}_j \| R_i \| \text{TS}_s \| \text{ID}_i \| \text{ID}_j \| \text{SK}^i_{\text{info}} \| \text{TK}_j)$, and $M_2 =< (\mathcal{C}_j, \text{Auth}^j_s, \text{SK}^j_{\text{info}}, \text{ID}_i, R_i) \oplus H^r(\text{TK}_j \| R_S \| \text{TS}_s), R_s, \text{TS}_s >$. Finally, the S server sends M_2 to the N_j node.

3) N_j receives M_2, verifies TS_s, and computes $\mathrm{TK}_j = d_j.R_s$ and $H^r(\mathrm{TK}_j \| R_S \| \mathrm{TS}_s)$. It also extracts $\mathcal{C}_j$, Auth_s^j, $\mathrm{SK}_{\text{info}}^j$, ID_i, and R_i and calculates $\mathcal{R}_j = \mathrm{PUF}(\mathcal{C}_j)$. Then, it computes Auth_s^{j*} based on the received data and verifies whether $\mathrm{Auth}_s^{j*} \stackrel{?}{=} \mathrm{Auth}_s^j$. If so, N_j generates its timestamp TS_j and calculates $r_j = \mathrm{PUF}(\mathrm{TS}_j) \bmod q$. If $r_j \neq 0$, it computes $R_j = r_j.P$ and derives its session key as $\mathrm{SK}_{ji} = H(r_j.R_i \| (\mathrm{SK}_{\text{info}}^j \oplus \mathcal{R}_j))$, which is used only once the mutual authentication between N_i and N_j is established and confirmed. Finally, it computes $\mathrm{Auth}_j^s = H(R_j \| \mathrm{TS}_j \| \mathrm{TS}_s \| (\mathrm{SK}_{\text{info}}^j \oplus \mathcal{R}_j))$, $\mathrm{Auth}_j^i = H(R_j \| \mathrm{ID}_j \| \mathrm{ID}_i \| \mathrm{SK}_{ji})$, and $M_3 =< \mathrm{Auth}_j^s, \mathrm{Auth}_j^i, R_j, \mathrm{TS}_j >$. Next, N_j sends M_3 to the S server.
4) Once received M_3, S verifies TS_j, calculates Auth_j^{s*} based on the received data and verifies whether $\mathrm{Auth}_j^{s*} \stackrel{?}{=} \mathrm{Auth}_j^s$ to authenticate N_j. Assuming N_j is legitimate and has been authenticated by S, the server generates the current timestamp TS_s^* and computes $\mathrm{SK}_{\text{info}}^i = \mathcal{R}_i \oplus H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s)$, $\mathrm{TK}_i' = H^r(\mathrm{TS}_s^* \| \mathrm{TK}_s \| \mathrm{TS}_i)$, and $\mathrm{Auth}_s^i = H(\mathrm{Auth}_j^i \| \mathrm{SK}_{\text{info}}^i \| \mathrm{TK}_i' \| \mathcal{R}_i)$. Finally, it sends $M_4 =< (\mathcal{C}_i, \mathrm{Auth}_j^i, \mathrm{Auth}_s^i, \mathrm{SK}_{\text{info}}^i, R_j) \oplus \mathrm{TK}_i', \mathrm{TS}_s^* >$ to the N_i node.
5) Upon receiving M_4, N_i verifies TS_s^* and computes $\mathrm{TK}_i'^* = H^r(\mathrm{TS}_s \| \mathrm{TS}_s^* \| \mathrm{TS}_i)$. It also extracts $\mathcal{C}_i$, Auth_j^i, Auth_s^i, $\mathrm{SK}_{\text{info}}^i$, and R_j and calculates $\mathcal{R}_i = \mathrm{PUF}(\mathcal{C}_i)$. Next, it computes the shared key as $\mathrm{SK}_{ij} = H(r_i.R_j \| (\mathrm{SK}_{\text{info}}^i \oplus \mathcal{R}_i))$ and verifies the extracted Auth_j^i and Auth_s^i to authenticate S and N_j. Assuming they are legitimate and have been authenticated, N_i generates the current timestamp TS_i^* and computes $\mathrm{Auth}_i^s = H(R_i \| \mathrm{TS}_i^* \| \mathrm{TS}_s^* \| (\mathrm{SK}_{\text{info}}^j \oplus \mathcal{R}_i))$ and $\mathrm{Auth}_i^j = H(R_i \| \mathrm{ID}_i \| \mathrm{ID}_j \| \mathrm{SK}_{ij})$. Finally, it sends $M_5 =< \mathrm{Auth}_i^s, \mathrm{Auth}_i^j, \mathrm{TS}_i^* >$ to the S server.
6) Once received M_5, S verifies TS_i^*, calculates Auth_i^{s*} based on the received data, and checks whether $\mathrm{Auth}_i^{s*} \stackrel{?}{=} \mathrm{Auth}_i^s$ to authenticate N_i. Assuming N_i is legitimate and has been authenticated by S, the server generates the current timestamp TS_s^{**} and computes $\mathrm{Auth}_s'^j = H(\mathrm{Auth}_i^j \| \mathrm{ID}_j \| \mathrm{TS}_s^{**} \| \mathcal{R}_j)$ and $M_6 =< \mathrm{Auth}_s'^j, \mathrm{TS}_s^{**} >$. Finally, it sends M_6 to the N_j node.
7) Upon receiving M_6, N_j verifies TS_s^{**} and $\mathrm{Auth}_s'^{j*}$ to authenticate S and N_i. Assuming that they are legitimate and have been authenticated, the mutual authentication and key agreement process is completed and the shared key will be $\mathrm{SK}_{ij} = \mathrm{SK}_{ji} = H(r_i.r_j.P \| H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s))$.

V. Security Proof of TARDIGRADE

To show the security soundness of TARDIGRADE against various attacks, we provide our formal and informal security reasoning in this section. The formal security proof is conducted on the real or random model, and informal security analysis against various attacks, including pandemic, replay, impersonation, and desynchronization attacks, is also provided. Table II represents a security comparison between TARDIGRADE and the most relevant PUF-based works, including the Garg *et al.*'s scheme.

TABLE II
Security Comparison

Protocol	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
[23]	✓	✓	×	✓	×	×	×	×	×	✓
[24]	✓	×	×	✓	✓	✓	×	✓	×	✓
[25]	✓	×	×	✓	-	×	✓	×	-	✓
[9]	✓	×	×	✓	×	✓	✓	×	✓	×
Our	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Here, $P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9$, and P_{10}, respectively, denote security against replay attack, impersonation attack, traceability and anonymity, secret disclosure attack, session key security, desynchronization attack, man-in-the-middle attack, insider adversary, forward secrecy, and pandemic attack.

A. Informal Security Analysis

Informal security proof methods are used using the analyst's knowledge and reasoning to prove that the security protocol is weak or the scheme lacks security pitfalls and is resistant the attack in question .

1) Replay Attack: In a replay attack, the adversary may eavesdrop on a protocol session, and then, aims to impersonate a protocol party by rebroadcasting the eavesdropped messages. Fortunately, in the TARDIGRADE protocol, each session is randomized by timestamps, and the correctness and integrity of the current timestamp is verified by the receiver. For example, in the first step of the protocol, N_i calculates $r_i = \mathrm{PUF}(\mathrm{TS}_i) \bmod q$, $R_i = r_i.P$, and $\mathrm{TK}_s = r_i.Q_s$, and finally, sends the tuple $M_1 =< (\mathrm{ID}_i, \mathrm{ID}_j) \oplus H^r(\mathrm{TK}_S, \mathrm{TS}_i, R_i), \mathrm{TS}_i, R_i >$ to the S and S verifies TS_i at the first. Undoubtedly, eavesdropping on the messages does not help the adversary in a later session. In detail, the adversary will be rejected by S due to the TS_i. In addition, the attacker has no significant chance to adapt TS_i to the attack time because a hash function guarantees its integrity. A similar argument can be conducted for other messages as well. Hence, TARDIGRADE provides security against replay attacks.

2) Impersonation Attack: To impersonate a protocol party, the attacker should do either a successful replay attack or be able to generate acceptable messages for the verifier. On the one hand, we already have discussed the security of TARDIGRADE against replay attacks. On the other hand, the adversary cannot produce expected messages without complete control of a protocol party.

More precisely, to impersonate N_i, the adversary should generate a valid $\mathrm{Auth}_i^s = H(R_i \| \mathrm{TS}_i^* \| \mathrm{TS}_s^* \| (\mathrm{SK}_{\text{info}}^j \oplus \mathcal{R}_i))$ and $\mathrm{Auth}_i^j = H(R_i \| \mathrm{ID}_i \| \mathrm{ID}_j \| \mathrm{SK}_{ij})$, where $\mathrm{SK}_{ij} = H(r_i.R_j \| (\mathrm{SK}_{\text{info}}^i \oplus \mathcal{R}_i))$ and $\mathcal{R}_i = \mathrm{PUF}(\mathcal{C}_i)$. Assuming that the used PUF is unclonable, the adversary has no chance to reproduce it. In addition, $\mathcal{C}_i$ has been masked in the message sent by S through the computation of $\mathrm{SK}_{\text{info}}^i = \mathcal{R}_i \oplus H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s)$, $\mathrm{TK}_i' = H^r(\mathrm{TS}_s^* \| \mathrm{TK}_s \| \mathrm{TS}_i)$, $\mathrm{Auth}_s^i = H(\mathrm{Auth}_j^i \| \mathrm{SK}_{\text{info}}^i \| \mathrm{TK}_i' \| \mathcal{R}_i)$, and $M_4 =< (\mathcal{C}_i, \mathrm{Auth}_j^i, \mathrm{Auth}_s^i, \mathrm{SK}_{\text{info}}^i, R_j) \oplus \mathrm{TK}_i', \mathrm{TS}_s^* >$. Hence, the adversary has no significant chance to impersonate N_i.

To impersonate S, the adversary should use $R_i = r_i.P$ and its secret key to compute TK_s and extract $(\mathrm{ID}_i, \mathrm{ID}_j)$ from M_1. Besides, the adversary needs the access to a valid $< \mathcal{C}_i, \mathcal{R}_i >$ token to calculate valid $\mathrm{SK}_{\text{info}}^i = \mathcal{R}_i \oplus H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s)$, $\mathrm{TK}_i' = H^r(\mathrm{TS}_s^* \| \mathrm{TK}_s \| \mathrm{TS}_i)$ and $\mathrm{Auth}_s^i = H(\mathrm{Auth}_j^i \| \mathrm{SK}_{\text{info}}^i \| \mathrm{TK}_i' \| \mathcal{R}_i)$

values. Hence, to impersonate S to N_i, the adversary needs the secret key of S and also a valid $< \mathcal{C}_i, \mathcal{R}_i >$, which is only shared with a legitimate server in the registration phase. All of the aforementioned confirms that the adversary has no significant chance of impersonating S.

To impersonate N_j, the adversary should be able to compute $\text{TK}_j = d_j.R_s$ for which he needs access to the node's private key. Besides, to compute $\text{Auth}_j^s = H(R_j \| \text{TS}_j \| \text{TS}_s \| (\text{SK}_{\text{info}}^j \oplus \mathcal{R}_j))$, $\text{Auth}_j^i = H(R_j \| \text{ID}_j \| \text{ID}_i \| \text{SK}_{ji})$, the adversary needs to clone its PUF, which is impractical under an ideal PUF model.

In short, an attacker could not impersonate any of the participating entities.

3) Traceability and Anonymity: Transferred messages over the public channel are $M_1 =< (\text{ID}_i, \text{ID}_j) \oplus H^r(\text{TK}_S, \text{TS}_i, R_i), \text{TS}_i, R_i >$, $M_2 = < (\mathcal{C}_j, \text{Auth}_s^j, \text{SK}_{\text{info}}^j, \text{ID}_i, R_i) \oplus H^r(\text{TK}_j \| R_S \| \text{TS}_s), R_s, \text{TS}_s >$, $M_3 =< \text{Auth}_j^s, \text{Auth}_j^i, R_j, \text{TS}_j >$, $M_4 =< (\mathcal{C}_i, \text{Auth}_j^i, \text{Auth}_s^i, \text{SK}_{\text{info}}^i, R_j) \oplus \text{TK}_i', \text{TS}_s^* >$, $M_5 =< \text{Auth}_i^s, \text{Auth}_i^j, \text{TS}_i^* >$, and $M_6 =< \text{Auth}_s'^j, \text{TS}_s^{**} >$. In these messages, timestamps (TS) could not be used to trace any party. R_i, R_s, and R_j are also random per session. The rest of the message components are masked by randomized values. Hence, the adversary cannot connect transferred messages over different sessions to compromise a protocol party's anonymity or trace it.

4) Secret Disclosure Attack: Following the given argument in the previous subsection, any transferred message over the public channel contains a timestamp, a random value, or a masked parameter. For masking, we use session-dependent parameters that include secret parameters. Therefore, the adversary will not be able to remove the mask without access to a protocol party's secret parameter. For example, given $M_1 =< (\text{ID}_i, \text{ID}_j) \oplus H^r(\text{TK}_S, \text{TS}_i, R_i), \text{TS}_i, R_i >$ the adversary needs d_s as the private key of the server to compute TK_S and extract ID_i and ID_j. Apart from this, the private keys are also masked by error correction code (ECC) point multiplications, e.g., $\text{TK}_s = d_s.R_i$, which does not allow the adversary to access the private keys without compromising the elliptic-curve Diffie–Hellman (ECDH) paradigm. Consequently, the adversary success probability for extracting any secret parameter from the transferred messages over the public channel is negligible.

5) Session Key Security: The session key is ephemeral and computed as

$$\text{SK}_{ij} = \text{SK}_{ji} = H(r_i.r_j.P \| H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s))$$

where r_i and r_j are session-dependent random values that are, respectively, generated by N_i and N_j and $\mathcal{R}_i$ and $\mathcal{R}_j$ are computed by the embedded PUFs on-board N_i and N_j, respectively. To compute $r_i.r_j.P$, given $R_i = r_i.P$ and $R_j = r_j.P$, the adversary should solve the ECDH problem, which is considered as a hard problem. Similarly, to disclose $\mathcal{R}_i$ and $\mathcal{R}_j$, the adversary should extract $\mathcal{C}_i$ and $\mathcal{C}_j$ at first, and then, predict the PUFs' responses, which is not practical. Therefore, the session key is sufficiently secure against any adversary controlling the messages transferred through the public channel.

6) Permanent Desynchronization Attack: Given that shared parameters are not updated after each session, the adversary cannot then desynchronize a protocol party by forcing it to update its shared parameters to different values compared with the related records in the other protocol parties. Thus, the only source of the desynchronization could be the unstable behavior of the used PUFs. This undesirable behavior has been ruled out by assuming that any used PUF is sufficiently stable over time. Hence, the proposed protocol is secure against permanent desynchronization attacks.

7) Man-in-the-Middle Attack: The integrity of any transferred message is guaranteed by secure hash functions and the usage of secret parameters that are part(s) of their inputs. More precisely, if we look at the transferred messages, i.e., M_1 to M_6, we can observe that, for example, $H^r(\text{TK}_S, \text{TS}_i, R_i)$ guarantee the integrity of M_1 and TK_S is a secret parameter unknown to the adversary or $\text{Auth}_i^s = H(R_i \| \text{TS}_i^* \| \text{TS}_s^* \| (\text{SK}_{\text{info}}^j \oplus \mathcal{R}_i))$ and $\text{Auth}_i^j = H(R_i \| \text{ID}_i \| \text{ID}_j \| \text{SK}_{ij})$ guarantee the integrity of M_6 and $\mathcal{R}_i$ and SK_{ij} are secrets unknown to the adversary. Therefore, any modification to the transferred messages will be detected by the receiver with high probability, showing that TARDIGRADE is secure against man-in-the-middle attacks.

8) Insider Adversary: In addition to the transferred messages over a public channel, which are accessible to any adversary, a privileged insider adversary could also access the exchanged messages in the registration phase or access the stored parameters in the S memory. However, such an adversary has no access to the server's private key. Given that to impersonate S, its private key is required, for example, to extract $(\text{ID}_i, \text{ID}_j)$ from the received M_1, the adversary will not be able to impersonate S or extract its private key. In addition, to extract the shared session key, the insider needs to compute $r_i.r_j.P$, given $R_i = r_i.P$ and $R_j = r_j.P$ and will not be possible without solving the ECDH problem. Nevertheless, the insider has the advantage of accessing $\mathcal{R}_i$ and $\mathcal{R}_j$ from the server's memory, compared to a naive adversary. Hence, although the insider adversary has some advantages over other adversaries, it is not yet feasible to impersonate the server or extract the session key, which could be the adversary's goal.

9) Forward Secrecy: Given that the session key at time t is computed as $\text{SK}_{ij} = \text{SK}_{ji} = H(r_i.r_j.P \| H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s))$, where r_i and r_j are the session-dependent random values, assuming the adversary compromised the session key of any other session t', e.g., $\text{SK}'_{ij} = \text{SK}'_{ji} = H(r'_i.r'_j.P \| H(\mathcal{R}'_i \| \mathcal{R}'_j \| r'_s \| d'_s))$, and the access to the transferred messages at time t and even $H(\mathcal{R}'_i \| \mathcal{R}'_j \| r'_s \| d_s))$ and $H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s))$ tokens, all this does not help the adversary to determine SK_{ij} without solving the ECDH problem. Hence, the proposed protocol provides forward secrecy.

10) Pandemic Attack: Assuming that the adversary compromised N_i and even emulated its PUF, it does not help to compromise the security of any other node N_j. The reason comes from the fact that we revised the structure of the messages such that N_i has no access to PUF responses of N_j and could only determine $H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s)$, which does not help to reveal $\mathcal{R}_j$. A similar argument holds for N_j. Consequently, the proposed protocol is secure against pandemic attacks.

B. Formal Security Analysis of TARDIGRADE in the RoR Model

In cryptography, mathematical proofs, also known as provable security, are commonly used to officially validate a protocol's security. The attacker's capabilities are specified by an adversarial model in this type of security proof. The goal of the proof is to show that the attacker must solve the underlying hard problem to breach the modeled system's security; for example, Li *et al.* [26] aim to bypass the assumptions of the random oracle model in the proposed password-authenticated key exchange scheme. As a result, the described adversarial model has an essential impact on the security assertions that have been undertaken. Because side-channel attacks and other implementation-specific vulnerabilities, such as node capture attacks [27], are challenging to model without building the system, such a proof frequently excludes them and does not guarantee the protocol security against such attacks. Furthermore, the primitives are deemed secure under a proven security. As a result, any attack that relies on nonideal primitive behavior will go undetected in this proof. However, this form of security proof is vital since it ensures the proposed scheme's structural soundness. Although many suggested schemes in the literature with provable security are defective in reality, each new scheme should be backed up by such evidence to assure a structural security and to explicitly reflect the opponent's capabilities and potential attack methods [28].

In this section, following [19], we formally evaluate the security of TARDIGRADE in the RoR model. We have calculated the adversary's advantage in distinguishing the real world of TARDIGRADE from the random world (RW). From here and for simplicity, we denote TARDIGRADE by RP.

Theorem 1: Let q_{exe}, q_{send}, and q_{test}, respectively, represent the number of queries to Execute, Send, and Test oracles on RP/RW, then

$$\begin{aligned}&\text{Adv}^{\text{RoR}}_{\mathcal{D},\text{RP}}(t, q_{\text{exe}}; q_{\text{test}}; q_{\text{send}}) -\\ &\text{Adv}^{\text{RoR}}_{\mathcal{D},\text{RW}}(t, q_{\text{exe}}; q_{\text{test}}; q_{\text{send}}) \leq\\ &5.q.\varepsilon_{\text{ECC}} + 10.q.\varepsilon_H + 4.q.\varepsilon_{\text{PUF}}\end{aligned}$$

where ε_{ECC} denotes the maximum advantage of solving ECDLP or EC-CDHP by the adversary on each query and ε_H represents the maximum advantage of contradicting the collision resistance property of $H(.)$. Besides, ε_{PUF} denotes the maximum advantage of distinguishing the output of PUF(.) from a random sequence, and q represents the total amount of queries (i.e., $q = q_{\text{exe}} + q_{\text{test}} + q_{\text{send}}$).

Proof: We assume two nodes (N_i and N_j) that communicate through a S server to share a session key. We also consider an $\mathcal{A}$ adversary who aims to compromise the semantic security of RP in the the RoR model. Under this setting, we use a game-based approach to prove the aforementioned theorem. For this, we pass through a series of games $\mathcal{G}$, starting from random world RW and ended in real-world (RP). For each game $\mathcal{G}_n$, we define an event $\text{Adv}^{\text{RoR}-\mathcal{G}_n}_{\mathcal{D},\mathcal{P}}(t, R)$, which corresponds to the adversary's advantage to correctly guess the hidden bit b involved in the Test queries. It should be noted that the structure of messages are identical in both RW and RP to rule out any trivial advantage for the adversary, e.g., we preserve the structure of the timestamps in both worlds.

1) **Game** $\mathcal{G}_0$. It defines RW and $\text{Adv}^{\text{RoR}-\mathcal{G}0}_{\mathcal{D},\text{RW}}(t, R) = 0$
2) **Game** $\mathcal{G}_1$. Compared to $\mathcal{G}_0$, in this game, any instance follows the structure of the transferred messages in RP. Nevertheless, all messages are selected completely random. It is clear $\text{Adv}^{\text{RoR}-\mathcal{G}0}_{\mathcal{D},\text{RW}}(t, R) - \text{Adv}^{\text{RoR}-\mathcal{G}1}_{\mathcal{D},\text{RW}}(t, R) = 0$.
3) **Game** $\mathcal{G}_2$. In this game, R_i, R_j, and R_s are calculated using ECC point multiplication. Given that r_i, r_j, and r_s are fresh random numbers, the adversary's advantage to distinguish $\mathcal{G}_2$ from $\mathcal{G}_1$ is as follows:
$$\text{Adv}^{\text{RoR}-\mathcal{G}_2}_{\mathcal{D},\text{RW}}(t, R) \leq \text{Adv}^{\text{RoR}-\mathcal{G}_1}_{\mathcal{D},\text{RW}}(t, R) + 3.q.\varepsilon_{\text{ECC}}$$
where $q = q_{\text{exe}} + q_{\text{send}} + q_{\text{test}}$.
4) **Game** $\mathcal{G}_3$. In this game, as a part of the transferred messages, the values of $(\text{ID}_i, \text{ID}_j) \oplus R_1, (\mathcal{C}_j, \text{Auth}^j_s, \text{SK}^j_{\text{info}}, \text{ID}_i, R_i) \oplus R_2$ and $(\mathcal{C}_i, \text{Auth}^i_j, \text{Auth}^i_s,$ $\text{SK}^i_{\text{info}}, R_j) \oplus R_3$ are used, respectively, in M_1, M_2, and M_4. Note that R_1, R_2, and R_3 are random values of the required length. It is obvious that this modification does not affect the adversary's advantage
$$\text{Adv}^{\text{RoR}-\mathcal{G}_3}_{\mathcal{D},\text{RW}}(t, R) = \text{Adv}^{\text{RoR}-\mathcal{G}_2}_{\mathcal{D},\text{RW}}(t, R).$$
5) **Game** $\mathcal{G}_4$. In this game, $R_1, R_2, R_3, \text{TK}_S, \text{TK}_j, \text{SK}^i_{\text{info}}$, and $\text{SK}^j_{\text{info}}$ are, respectively, replaced by $H^r(\text{TK}_S, \text{TS}_i, R_i)$, $H^r(\text{TK}_j \| R_S \| \text{TS}_s)$, $H^r(\text{TS}^*_s \| \text{TS}^*_s \| \text{TS}_i)$, $r_i.Q_s$, $r_s.Q_j$, $\mathcal{R}_i \oplus H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s)$, and $\mathcal{R}_j \oplus H(\mathcal{R}_i \| \mathcal{R}_j \| r_s \| d_s)$. Given that r_i, r_j, and r_s are the fresh random numbers and timestamps are generated incrementally, the adversary's advantage to distinguish $\mathcal{G}_4$ from $\mathcal{G}_3$ comes from discerning the output of H^r from a random sequence or dealing with the ECC hard problems. Hence, the $\mathcal{A}$'s advantage is determined as follows:
$$\text{Adv}^{\text{RoR}-\mathcal{G}_4}_{\mathcal{D},\text{RW}}(t, R) \leq \text{Adv}^{\text{RoR}-\mathcal{G}_3}_{\mathcal{D},\text{RW}}(t, R) + 2.q.\varepsilon_{\text{ECC}} + 5.q.\varepsilon_H.$$
6) **Game** $\mathcal{G}_5$. This game is identical to G_4, except that Auths values are calculated using $H(.)$. However, any Auth, e.g., Auth^j_s, is randomized by a nonce or a timestamp. Therefore, $\mathcal{A}$'s advantage comes from the unmasked Auth tokens
$$\text{Adv}^{\text{RoR}-\mathcal{G}_5}_{\mathcal{D},\text{RW}}(t, R) \leq \text{Adv}^{\text{RoR}-\mathcal{G}_4}_{\mathcal{D},\text{RW}}(t, R) + 4.q.\varepsilon_H.$$
7) **Game** $\mathcal{G}_6$. This game is identical to G_5, excluding that r_i and r_j are calculated using a PUF. Consequently
$$\text{Adv}^{\text{RoR}-\mathcal{G}_6}_{\mathcal{D},\text{RW}}(t, R) \leq \text{Adv}^{\text{RoR}-\mathcal{G}_5}_{\mathcal{D},\text{RW}}(t, R) + 2.q.\varepsilon_{\text{PUF}}.$$
8) **Game** $\mathcal{G}_7$. This game is identical to G_6, except that $\mathcal{R}_i$ and R_j are calculated using a PUF. Therefore
$$\text{Adv}^{\text{RoR}-\mathcal{G}_7}_{\mathcal{D},\text{RW}}(t, R) \leq \text{Adv}^{\text{RoR}-\mathcal{G}_6}_{\mathcal{D},\text{RW}}(t, R) + 2.q.\varepsilon_{\text{PUF}}.$$
9) **Game** $\mathcal{G}_8$. This game is identical to G_7 with the exception that the session key is calculated using a hash function (i.e., $\text{SK}_{ji} = H(r_j.R_i \| (\text{SK}^j_{\text{info}} \oplus \mathcal{R}_j)))$. Given that input

TABLE III
REQUIRED PRIMITIVES

Protocol	N_i	N_j	S
Garg *et al.*'s	ECC, H, RNG,PUF	ECC, H, RNG,PUF	ECC, H, RNG
TARDIGRADE	ECC, H, PUF	ECC, H, PUF	ECC, H, RNG

values for SK_{ij} are randomized by nonces and timestamps, consequently

$$\text{Adv}_{\mathcal{D},\text{RW}}^{\text{RoR}-\mathcal{G}_8}(t,R) \leq \text{Adv}_{\mathcal{D},\text{RW}}^{\text{RoR}-\mathcal{G}_7}(t,R) + q.\varepsilon_H.$$

Finally, it is straightforward that $\mathcal{G}_8$ represents the implementation of RP. Therefore

$$\text{Adv}_{\mathcal{D},\text{RP}}^{\text{RoR}}(t,R) - \text{Adv}_{\mathcal{D},\text{RW}}^{\text{RoR}}(t,R) \leq$$

$$\text{Adv}_{\mathcal{D},\text{RW}}^{\text{RoR}-\mathcal{G}_8}(t,R) - \text{Adv}_{\mathcal{D},\text{RW}}^{RoR-\mathcal{G}_0}(t,R) \leq$$

$$5.q.\varepsilon_{\text{ECC}} + 10.q.\varepsilon_H + 4.q.\varepsilon_{\text{PUF}}$$

which completes the proof.

VI. Cost Analysis of TARDIGRADE

Garg *et al.* compared their proposal with the state-of-the-art of related works, i.e., Chatterjee *et al.* [23], Braeken [29], and Aman *et al.* [30] protocols, and showed how their scheme outperforms the existing solutions in terms of security and efficiency. Therefore, for the sake of avoiding repetition, we only compare TARDIGRADE with Garg *et al.*'s protocol in terms of performance.

A. Performance Analysis

In terms of computational complexity, N_i and N_j nodes in TARDIGRADE perform, respectively, seven and six calls to the hash function, two PUF invocations and three ECC point multiplications. These calculations are slightly higher than in Garg *et al.*'s protocol, in which each node generates a random number, makes four calls to the hash function and three ECC point multiplications. Concerning the server, in Garg *et al.*'s protocol, it does six ECC point multiplications plus eight calls to the hash function. Similarly, in TARDIGRADE, the server performs only three ECC point multiplications plus eight calls to the hash function.

Regarding the primitives supported on-board, Garg *et al.*'s protocol nodes require a random number generator and a PUF function. In contrast, in TARDIGRADE, nodes only do PUF(.) computations. The aforementioned peculiarity is possible because the new scheme uses the embedded PUF(.) function, with a timestamp as the seed, to generate the required random numbers. Using this approach, we reduce the hardware overhead.

We present the comparison between the required primitives, computational costs, and communication overheads of Garg *et al.*'s protocol and TARDIGRADE scheme in Tables III–V. For the comparison, the bit lengths of a timestamp, a node's identifier, a challenge, a response, a hash function and an ECC point are 32, 160, 160, 160, 160, and 320 bits, respectively. We consider SHA-256 (truncating its output to 160-bit when

TABLE IV
COMPUTATIONAL OVERHEAD

Protocol	N_i	N_j	S
Garg *et al.*'s	$3T_{mn} + 4T_{hn} \approx 75\ ms$	$3T_{mn} + 4T_{hn} \approx 75\ ms$	$6T_{ms} + 8T_{hs} \approx 15.345\ ms$
TARDIGRADE	$3T_{mn} + 7T_{hn} + 2T_{PUFn} \approx 90\ ms$	$3T_{mn} + 6T_{hn} + 2T_{PUFn} \approx 87\ ms$	$3T_{ms} + 8T_{hs} \approx 7.832\ ms$

TABLE V
COMMUNICATION OVERHEAD OF EACH TRANSFERRED MESSAGE (BITS)

Protocol	M_1	M_2	M_3	M_4	M_5	M_6
Garg *et al.*'s	672	992	192	1024	512	192
TARDIGRADE	672	1312	672	992	352	192

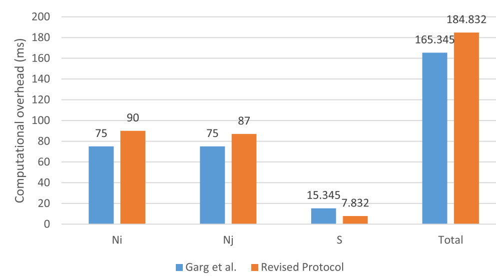

Fig. 1. TARDIGRADE versus Garg *et al.*'s protocol, computation comparison.

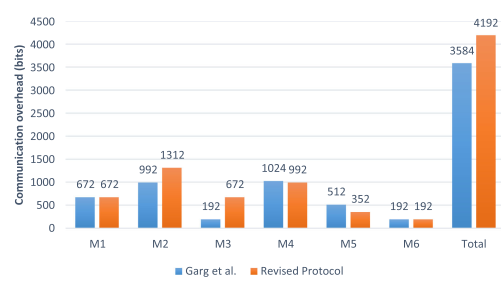

Fig. 2. TARDIGRADE versus Garg *et al.*'s protocol, per-message communications comparison.

required) to avoid the recent security flaws of SHA-1 [31]. For experimental evaluation, we used an Intel Xeon CPU E5-2650V2 with 2.60-GHz frequency and 8-GB RAM as the server and an Arduino UNO R3 board with an ATmega328P microcontroller as the sensor node. Under this platform, we achieve $T_{ms} \approx 2.5044$ ms, $T_{hs} \approx 0.03993$ ms, $T_{mn} \approx 21$ ms, and $T_{hn} \approx 3$ ms, which denote the computation times for ECC point multiplication and one-way hash function on the server and the node, respectively. We also consider the consuming time of a PUF invocation ($T_{\text{PUF}n}$) equals to T_{hn}.

Based on the results, the performance of TARDIGRADE is comparable with that of Garg *et al.*'s protocol. More precisely, the computation time of the nodes increased by 18%, but the computation time of the server decreased by 49%, as shown in Fig. 1. The communication cost of TARDIGRADE is slightly higher with an increment of 18%, as displayed in Fig. 2. However, it is worth noting that the enhanced protocol provides mutual authentication between N_i and N_j, besides extra security features and a higher security level. In contrast, Garg *et al.*'s protocol only targeted mutual authentication for the server node, does not provide node–node mutual authentication, and presents critical security holes.

B. On the Security and Reliability

Throughout our analysis, we assume the PUF model used by Garg *et al.* [9, Sec. III.A.1]. That is, given challenges $\mathcal{C} \neq \mathcal{C}'$ then PUF($\mathcal{C}$) and PUF($\mathcal{C}'$) are entirely different. Note that a particular PUF returns the same output every time a user test it with the same input. Likewise, different PUFs output distinct responses for the same challenge. However, the current PUFs on the shelves may not behave exactly in that way. More precisely, a significant drawback of PUF technology is the dependence of their output to device ageing and operating conditions, leading to instability of the returned response to the given challenge. A commonly used technique to overcome this instability is to use fuzzy-extractor modules, and helper data [32], [33]. These modules convert noisy PUF responses to reliable responses using proper error correction codes. Depending on the kind of PUF used, other solutions could be applied, e.g., Schaub *et al.* [34] introduce some enrolment techniques to enhance the responses of PUFs in hostile scenarios and Wallrabensteing [35] aims to mitigate the ageing effects of the PUF. On the other hand, many researchers have concentrated on predicting the PUF output by modeling it to compromise its security. Among different approaches, machine-learning-based techniques are more promising in this direction [36]. Although all those details are valuable and very important at the application level, it is worth noting that designing such a PUF function is an active research area itself and out of the scope of this article. Therefore, we urge interested readers to consult [37]–[39] for the latest advances and challenges for designing a reliable PUF or refer to [40] and [41] for the latest advances on designing secure PUFs against modeling attacks.

VII. Conclusion

In this article, we further analyze the security of a protocol, which was recently proposed by Garg *et al.* We show that besides the node impersonation attack presented by Akram *et al.*, the scheme has critical security faults. More precisely, we offer how the protocol does not guarantee the privacy protection of localization, and a passive adversary can easily track any node in this protocol. The protocol is also vulnerable to desynchronization and integrity attacks. Besides, following the Akram *et al.* adversarial model, we presented an attack named as pandemic session key disclosure attack, for which the adversary can disclose the agreed session key between any pair of nodes in the IoT network by compromising a single node. Using the available resources in Garg *et al.*'s protocol (e.g., PUF functions), we revise the protocol to fix its security flaws efficiently and provide new security features such as node–node mutual authentication. First, an informal security analysis, and then, a formal security analysis of the enhanced protocol in the real or random model highlights its security futures compared with the Garg *et al.*' protocol. Finally, in terms of performance, the new scheme is similar to its predecessor.

While we investigated the security of the proposed protocol by using the DY adversarial model, other adversarial models could be used depending on the application, such as the Canetti–Krawczyk model and the extended Canetti–Krawczyk. In those models, the adversary can expose the secret information of any protocol participant, including the server—note that, in this article, the attacker can only access the secret information of the clients. Under this new scenario, the adversary's goal might be different, such as the security of the temporary key. While we ensure client security through the careful use of PUFs in our architecture, full server security may require the use of the user credentials such as biometrics. However, we will leave this as a topic for future research.

Through our analysis, similar to many other related works, we assume that the used PUF is ideal and also our security analysis is conducted under this assumption. Although this hypothesis is widely accepted and many researchers try to design a reliable and unpredictable PUF, however, we still do not have a PUF circuit that behaves like an ideal one, as mentioned in Section VI-B. Therefore, designing a secure protocol that relaxes the PUF model and achieves the security level of `TARDIGRADE` could be challenging, but worth investigating.

Last but not the least, we believe the application of the proposed adversarial model is not limited to just Garg *et al.*'s protocol, and some other protocols could be victims of this attack. Hence, we suggest that protocol designers examine the security of their proposals against this new attack. For instance, our analysis concludes that the recent proposal by Nikooghadam *et al.* [42] is also vulnerable to this attack.

References

[1] A. Esfahani *et al.*, "An efficient web authentication mechanism preventing man-in-the-middle attacks in Industry 4.0 supply chain," *IEEE Access*, vol. 7, pp. 58981–58989, 2019.

[2] P. Radanliev *et al.*, "Cyber risk at the edge: Current and future trends on cyber risk analytics and artificial intelligence in the Industrial Internet of Things and Industry 4.0 supply chains," *Cybersecurity*, vol. 3, no. 1, pp. 1–21, 2020.

[3] J. Sengupta, S. Ruj, and S. D. Bit, "A secure fog-based architecture for industrial Internet of Things and Industry 4.0," *IEEE Trans. Ind. Inform.*, vol. 17, no. 4, pp. 2316–2324, Apr. 2021.

[4] M. A. P. Chamikara, P. Bertók, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "A trustworthy privacy preserving framework for machine learning in industrial IoT systems," *IEEE Trans. Ind. Inform.*, vol. 16, no. 9, pp. 6092–6102, Sep. 2020.

[5] J. Zhang, H. Zhong, J. Cui, Y. Xu, and L. Liu, "An extensible and effective anonymous batch authentication scheme for smart vehicular networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3462–3473, Apr. 2020.

[6] L. Zhao and X. Dong, "An industrial Internet of Things feature selection method based on potential entropy evaluation criteria," *IEEE Access*, vol. 6, pp. 4608–4617, 2018.

[7] T. Lins and R. A. R. Oliveira, "Cyber-physical production systems retrofitting in context of Industry 4.0," *Comput. Ind. Eng.*, vol. 139, 2020, Art. no. 106193.

[8] K. Benzekki, A. E. Fergougui, and A. E. Elalaoui, "Software-defined networking (SDN): A survey," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5803–5833, 2016.

[9] S. Garg, K. Kaur, G. Kaddoum, and K.-K. R. Choo, "Towards secure and provable authentication for Internet of Things: Realizing industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4598–4606, May 2019.

[10] M. A. Akram, K. Mahmood, S. Kumari, and H. Xiong, "Comment on "Towards secure and provable authentication for Internet of Things: Realizing Industry 4.0"," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4676–4681, May 2020.

[11] J. Wei, X. Chen, X. Huang, X. Hu, and W. Susilo, "RS-HABE: Revocable-storage and hierarchical attribute-based access scheme for secure sharing of e-Health records in public cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2301–2315, Sep./Oct. 2021.

[12] E. Erdem and M. T. Sandikkaya, "OTPaaS—One time password as a service," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 3, pp. 743–756, Mar. 2019.

[13] X. Han, L. Wang, S. Xu, D. Zhao, and G. Liu, "Recognizing roles of online illegal gambling participants: An ensemble learning approach," *Comput. Secur.*, vol. 87, 2019 Art. no. 101588.

[14] Q. Jiang, X. Zhang, N. Zhang, Y. Tian, X. Ma, and J. Ma, "Three-factor authentication protocol using physical unclonable function for IoV," *Comput. Commun.*, vol. 173, pp. 45–55, 2021.

[15] Q. Jiang, N. Zhang, J. Ni, J. Ma, X. Ma, and K. R. Choo, "Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9390–9401, Sep. 2020.

[16] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.

[17] R. C.-W. Phan, "Cryptanalysis of a new ultralightweight RFID authentication protocol-SASI," *IEEE Trans. Dependable Secure Comput.*, vol. 6, no. 4, pp. 316–320, Oct.–Dec. 2008.

[18] A. Juels and S. A. Weis, "Defining strong privacy for RFID," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, pp. 1–23, 2009.

[19] M. Abdalla, P. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 3386, S. Vaudenay, Ed. Berlin, Germany: Springer, 2005, pp. 65–84.

[20] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "On the indifferentiability of the sponge construction," in *Advances in Cryptology—EUROCRYPT 2008* (Lecture Notes in Computer Science), vol. 4965, N. P. Smart, Ed. Berlin, Germany: Springer, 2008, pp. 181–197.

[21] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "Sponge-based pseudo-random number generators," in *Cryptographic Hardware and Embedded Systems, CHES 2010* (Lecture Notes in Computer Science), vol. 6225, S. Mangard and F. Standaert, Eds. Berlin, Germany: Springer, 2010, pp. 33–47.

[22] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The making of KECCAK," *Cryptologia*, vol. 38, no. 1, pp. 26–60, 2014.

[23] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-based secure communication protocol for IoT," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 3, pp. 1– 25, 2017.

[24] U. Chatterjee *et al.*, "Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database," *IEEE Trans. Dependable Secur. Comput.*, vol. 16, no. 3, pp. 424–437, May/Jun. 2019.

[25] U. Chatterjee, D. Mukhopadhyay, and R. S. Chakraborty, "3PAA: A private PUF protocol for anonymous authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 756–769, Sep. 2020, doi: 10.1109/TIFS.2020.3021917.

[26] Z. Li, D. Wang, and E. Morais, "Quantum-safe round-optimal password authentication for mobile devices," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2020.3040776.

[27] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, "Understanding node capture attacks in user authentication schemes for wireless sensor networks," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2020.2974220.

[28] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy for Industry 4.0," *Sci. China Inf. Sci.*, vol. 65, no. 1, pp. 1–15, 2022.

[29] A. Braeken, "PUF based authentication protocol for IoT," *Symmetry*, vol. 10, no. 8, pp. 1–15, 2018.

[30] M. N. Aman, K. C. Chua, and B. Sikdar, "Mutual authentication in IoT systems using physical unclonable functions," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1327–1340, Oct. 2017.

[31] G. Leurent and T. Peyrin, "From collisions to chosen-prefix collisions application to full SHA-1," in *EUROCRYPT* (Lecture Notes in Computer Science), vol. 11478, Y. Ishai and V. Rijmen, Eds. Berlin, Germany: Springer, 2019, pp. 527–555.

[32] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 889–902, Jun. 2015.

[33] C. Herder, L. Ren, M. van Dijk, M.-D. Yu, and S. Devadas, "Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 1, pp. 65–82, Jan./Feb. 2017.

[34] A. Schaub, J. Danger, S. Guilley, and O. Rioul, "An improved analysis of reliability and entropy for delay PUFs," in *Proc. 21st Euromicro Conf. Digit. Syst. Des.*, Prague, Czech Republic, Aug. 29–31, 2018, pp. 553–560.

[35] J. Wallrabensteing, "Remote re-enrollment of physical unclonable functions," U.S. Patent 2019/0138753, 2019.

[36] P. Santikellur and R. S. Chakraborty, "A computationally efficient tensor regression network-based modeling attack on XOR arbiter PUF and its variants," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 40, no. 6, pp. 1197–1206, Jun. 2021.

[37] K. Choi, S. Baek, J. Heo, and J. Hong, "A 100% stable sense-amplifier-based physically unclonable function with individually embedded non-volatile memory," *IEEE Access*, vol. 8, pp. 21857–21865, 2020.

[38] D. Jeon, J. Baek, Y. Kim, J. Lee, D. K. Kim, and B. Choi, "A physical unclonable function with bit error rate$\times 10^{-8}$ based on contact formation probability without error correction code," *IEEE J. Solid State Circuits*, vol. 55, no. 3, pp. 805–816, Mar. 2020.

[39] S. Lee, M.-K. Oh, Y. Kang, and D. Choi, "Design of resistor-capacitor physically unclonable function for resource-constrained IoT devices," *Sensors*, vol. 20, no. 2, pp. 1–16,, 2020.

[40] A. Wang, W. Tan, Y. Wen, and Y. Lao, "NoPUF: A novel PUF design framework toward modeling attack resistant PUFs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 6, pp. 2508–2521, Jun. 2021.

[41] C. Gu, C. Chang, W. Liu, S. Yu, Y. Wang, and M. O'Neill, "A modeling attack resistant deception technique for securing lightweight-PUF-based authentication," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 40, no. 6, pp. 1183–1196, Jun. 2021.

[42] M. Nikooghadam, H. Amintoosi, S. H. Islam, and M. F. Moghadam, "A provably secure and lightweight authentication scheme for internet of drones for smart city surveillance," *J. Syst. Archit.*, vol. 115, 2021, Art. no. 101955.