A PUF based 3-Phase Adaptive Authentication Protocol with Ultralightweight Authentication Phase for IOT Devices

## INTRODUCTION

Authentication protocols form the basis to ensure security for all modern-day communications. The most common form of authentication is the two-factor authentication where a one-time password is shared to complete any financial transaction. But these transactions involve human interference to complete the process. In automated systems, especially where IOT devices interact with each other such forms of authentication are not feasible. For this various authentication protocols are designed especially for IOT devices which are beautifully highlighted in Mohamed et al [1].

Our protocol assumes the simple scenario of an EV charging station, where a user can feed in the data remotely. Given below is the architecture for EV and Charging Infrastructure:

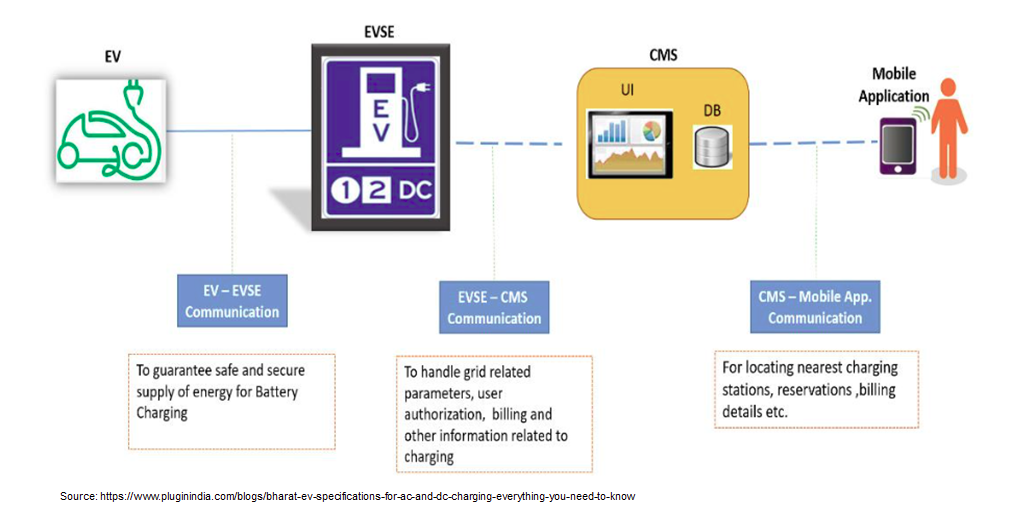


FIG 1. Architecture for EV and Charging Infrastructure

The protocol fits into the communication done by the user on the mobile application. The mobile application is used to feed in the data required for charging the EV, so that one can do the same without being physically present in the charging station. The charging station takes in the data and relays it to the microgrid control station, which enables the power flow to the charging station.

## LITERATURE REVIEW

While a number of researchers have worked on authentication protocols, the focus has not been primarily on an efficient authentication protocol. In [2], they have developed a key-exchanging protocol where it primarily addresses security challenges in the smart grid by protecting the embedded devices. One being the security breach during communication where an attacker acts as an authenticated or legitimate consumer act as an adversary to cut down bill cost. And the other by changing the power calculations data by leveraging the underlying physics of the meter. To solve this problem, they have used a PUF based smart meter. Also, another PUF based implementation [3], focuses on the physical access to IOT devices by an adversary leading to the launch of various physical and side channel attacks. It also eliminates the human requirement for passwords and smart cards. Although [2] and [3] solves some major challenges and has minimal hardware overheads, in both cases the PUF response is communicated over the network to the server which stands as a security vulnerability and the authentication protocol is computationally heavy and involves a greater number of message overheads which are not desired for resource constrained IOT devices.

In the field of IOT healthcare where anonymity and privacy are integral as health-related data are highly sensitive and personal, [4] has contributed to the security of data transferred through open public channels and prevention of DOS, impersonation, replay and man in the middle attacks. In [5], they have prevented Man in the Middle attacks, modification and replay attacks in scenarios where an adversary intercepts BLE at max range of 1000 meters using several publicly available tools. They have also reduced power consumption in transmitting sensor data using data signatures. But both [4],[5] fail to ensure security of the IOT devices from physical access and a fast authentication scheme.

In the field of Intelligent Transport Systems (ITS) and Vehicular Ad Hoc Network (VANET), Chistousov et al [6] proposed an adaptive authentication solution where the problem of key delivery between the parties was addressed with the help of zero knowledge proofs. Here, no symmetric or asymmetric encryptions have been used and the confidentiality of the vehicle’s route was varied with the traffic intensity so as to minimize authentication times. However, it failed to address possibility of the road side units being physically compromised.

Although various authentication mechanisms like Transport Layer Security (TLS) Protocol, Tunneled TLS (TTLS), Generalized Pre-Shared Key, Internet Key Exchange protocol version 2 (IKEv2) that work over the Extensible Authentication Protocol (EAP) and many others are used in traditional wireless networks, but these protocols are not resource constrained and hence not suitable for IOT devices.

## II. RESEARCH GAPS TARGETED to SOLVE in OUR SOLUTION

In most practical scenarios, where a device/user needs to be authenticated, it first needs to be enrolled at the server end. So, it is evident that for a device, it is enrolled only once but it may be authenticated any number of times in the future as the server deems necessary. With this backdrop, our authentication protocol is most beneficial for high frequency authentication and re-authentication requests, as the message transmissions are kept low and no encryption or decryption is used on either side. And all of this is done without compromising on security and eliminating all possibilities of man in the middle and replay attacks.

In cases where two users try to enroll simultaneously with the same credentials, the one that makes the request first usually gets enrolled rather than the intended user. To avoid such scenarios, we have performed a check in the enrollment phases so as to ensure that only the intended user is enrolled.

Unlike most other solutions, the PUF response at the user end is never communicated directly over the network which keeps the devices’ unique fingerprint hidden. This in turn solves the problem of the nodes being physically compromised as the PUF response will change whenever the device is subjected to any physical tampering.

## III. PROPOSED PROTOCOL

Our protocol assumes the simple scenario of an EV charging station, where a user can feed in data remotely. So, this scenario involves two device types - *Users* deployed as applications in the user mobile devices, and *Servers* at the administrative end. Users are typically resource constrained IOT devices, while servers are resource rich and computationally well equipped. Each user/IOT device is equipped with a physically unclonable function, PUF which acts as digital signature for that device and each server has its own unique identity Sid (basically a serial number)

Our protocol consists of 3-phases -

1. Pre-Shared Key Exchange
2. Enrollment phase
3. Authentication phase

Although the three phases are used simultaneously, it has been designed keeping in mind that an IOT device once enrolled with a server may require re-authentication as and when any suspicious activity is detected or when the server deems necessary. In that case the frequency of phases 1 and 2 will be once for every user (i.e. very minimal), while phase 3 will be once used after the device is enrolled and as many times later as necessary. So, this makes it important that the authentication phase is very light and requires minimal resources. And this is exactly what we have done. We have only used hashing in the user as well as server side, which enables faster authentication time as compared to other models. Also, the message exchanges occur in plaintext during this phase and at the same time ensures security.

The three phases of the protocol are discussed below:

**1. Pre-Shared Key Exchange:** This phase primarily focuses on the sharing of the master secret key (msk) with the user and server. The msk is used in the enrollment phase to generate the symmetric key (SKUS). While there exists a number of key sharing algorithms two notable ones that can be used are – one is the Transport Layer Security (TLS) that works over the Extensible Authentication Protocol (EAP). More details on EAP-TLS at RFC 3748. The other method can be using public key cryptography as shown below:

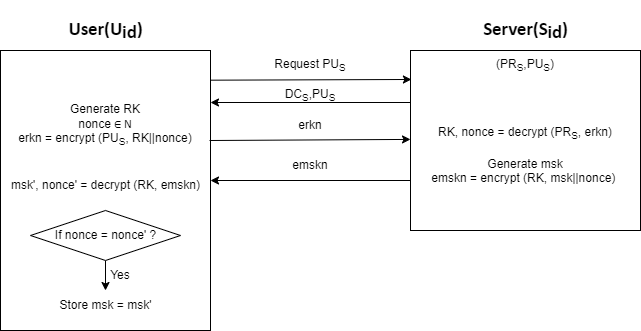


FIG 2. Master Secret Key (msk) shared using public key cryptography

In this method, initially a random key RK is generated by the user, encrypted using the public key PUS and sent to the server. A nonce is also sent along with the RK to prevent replay attack. The RK is decrypted at the server and used as the shared secret key for encryption and decryption to send the master secret key msk to the user. The two-step encryption and decryption are required to prevent MitM attacks at this stage.

**2. Enrollment Phase:** In this phase, both the user and server generate the shared secret keys (SKUS) from master secret key (msk) previously shared. This provides a secure and efficient way for the user and server to communicate securely over an untrusted network.

The user then sends an enrollment request with its user identity (uid). Once the server receives an enrollment request for a new user (i.e. user id already not in records), then the server samples a challenge *c* ∈R C, where C is the challenge space for the PUF instance embedded in the user/device, along with a seed randomly generated from the set of natural numbers. The challenge *c* is used as an input to generate a unique response from the user embedded with a PUF instance. The seed introduces complexity into the hash generated thereby ensuring randomness and unpredictability. As the inputs for the hash function are taken from multiple data sources, collision is avoided.

The server then encrypts and sends the challenge and seed to the user as

*cs* = encrypt (SKUS, (c || seed))

The user on receiving *cs*, decrypts the challenge *c* and seed and computes the following:

1. R = PUF(c) where R is the response generated by the PUF instance with challenge *c* as input.

2. α = H (R || seed || Sid) where α is the hash of the response, seed and Sid to ensure data integrity. It serves as the unique identifier for the user to be enrolled at the server. Here, Sid is also taken as input to the hash function as this ensures the servers

3. eac = encrypt (SKUS, (α || c))

It then sends the encrypted α and challenge *c* to the server. The challenge *c* is also sent back to the server so that the server can check if the challenge generated and the challenge received are same.

Once the server decrypts the alpha and challenge, it compares if the challenge received is the same as the challenge given. If the comparison results into true, it means that the user is the desired user that needs to be enrolled.

The significance of checking if the challenge generated and the challenge received is that it ensures only the intended user that started the enrollment process is enrolled. In case that simultaneous enrollment requests are made with the same credentials, it ensures that only the user to which the challenge was sent to, is enrolled.

An acknowledgement is also sent to the user and the data for that user is stored in the server as:

σUid = (c,seed, α’)

The challenge *c* and seed are required to be stored as they are inputs for the user to generate the hash α’’ that is sent to the server during the authentication phase discussed below. Now, to check if the user trying to authenticate is an authentic user, the server needs to match the hash value of the response. For this, the server needs to store the α’ at the enrollment phase so that it can compare the same response during the authentication phase.

At the server, the challenge *c*, seed and α’ are stored for the enrolled user. For every authentication and reauthentication of the user, the server needs to send the corresponding challenge and seed values in order to get the α from that user.

**3. Authentication Phase:** This phase is used to verify the authenticity of the users that are already enrolled with the server. The authentication request may be initiated by the user (usually done after enrollment) or as and when the server deems it necessary in which case the request is initiated by the server. Once, the server receives an authentication request or it decides to authenticate a user, it retrieves the corresponding entry for the user and parses σ = (c,seed, α’). It then generates a nonce and sends the challenge *c*, seed and nonce to the user. The nonce ensures that the β value is unique and unpredictable. It also prevents relay attacks.

The user on receiving the data computes the following:

First the PUF response is generated for the given challenge *c*, which is unique for each user/device. Following this α’’ is calculated so that it can be compared to the one stored at the server during the enrollment phase. Finally, β’ is calculated and sent to the server.

R’=PUF(c)

α’’ = H (R’ || seed || Sid)

β’ = H (α” || nonce)

The server compares it with the calculated β. If the comparison results true then the authentication is successful and the server allows further communication from this user.

**Re-authentication:** a re-authentication request is initiated only by the server for an already enrolled user as and when the server deems it necessary. In this case the server automatically fetches the sigma for the user it wants to authenticate and sends the challenge *c,* nonce and seed to that user. The next steps are similar to as discussed in the authentication phase above.

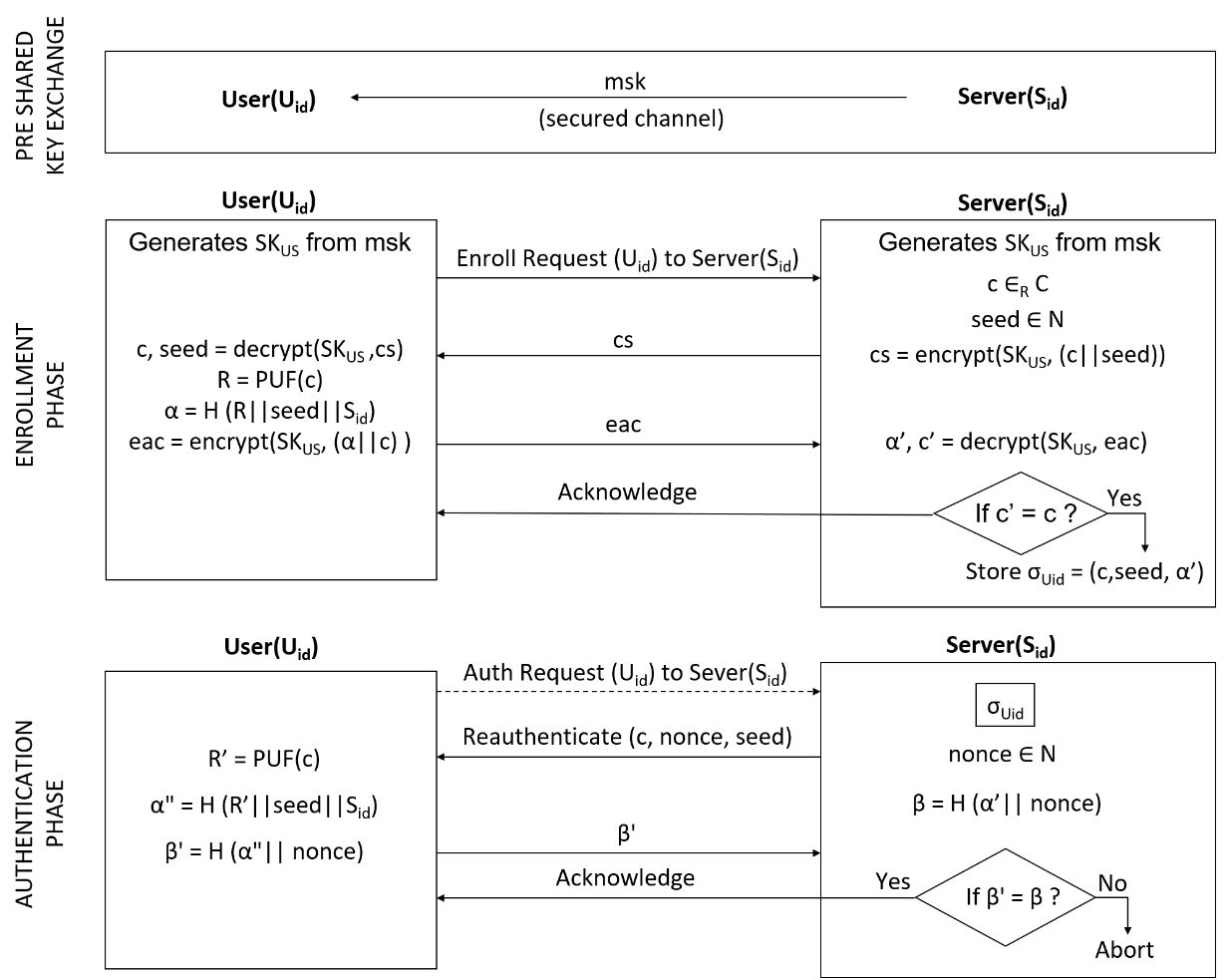


FIG 3. Proposed 3-phase authentication protocol with ultralightweight authentication phase for IOT devices.

In the above scenario, there is a possibility that the adversary might map the plaintext messages in authentication phase i.e c,nonce,seed → β mapping and may develop a machine learning model to generate the mapping. Although the probability of such a scenario is negligible, to completely eliminate this, we have used an adaptation scheme for the above protocol at the cost of execution time.

**Development of scheme for adaptation in the protocol:**

When this protocol is used along with an intrusion detection system (IDS) that continuously monitors the threat level. Our protocol continuously reads the threat score received from the IDS. Depending on this score, the authentication phase is modified such that in cases where this score is high or beyond a threshold value indicating a high threat factor. Then our protocol automatically uses encryption in the authentication phase.

Although this comes at the tradeoff between speed in our authentication phase, the additional security nullifies any probability of an adversary. This ensures that the data is not compromised even when there is a high threat score. The modified authentication phase is shown in Fig 4 below. It is to be noted here that there is no change in the enrollment or key sharing phase.

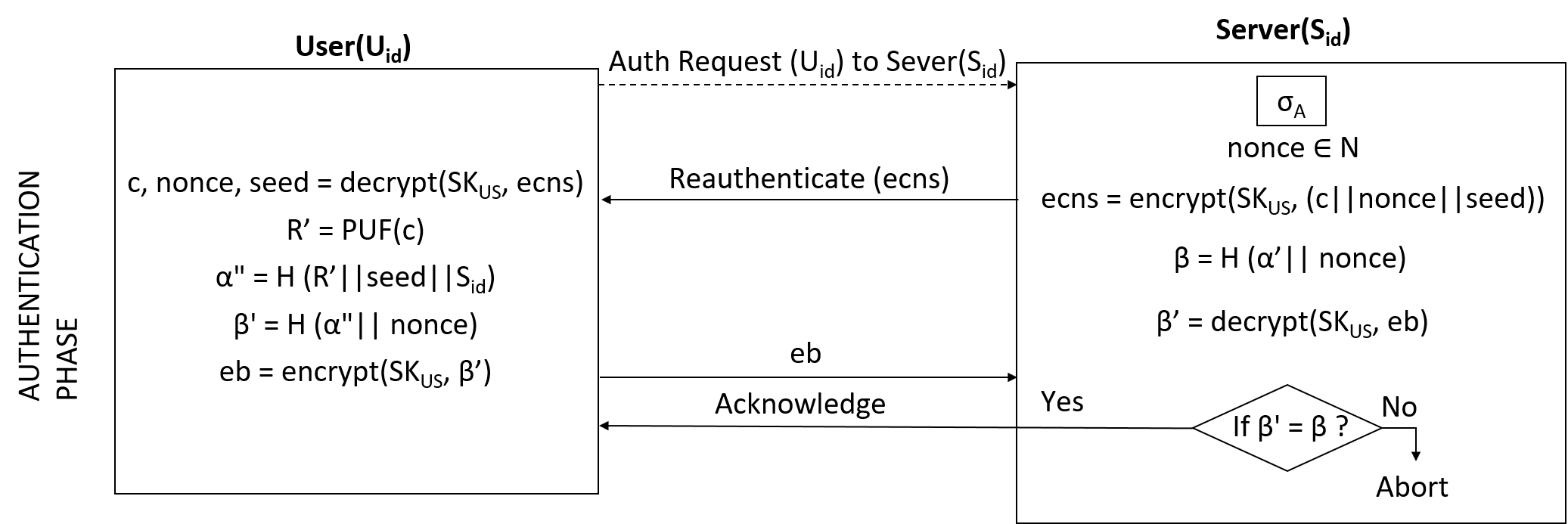


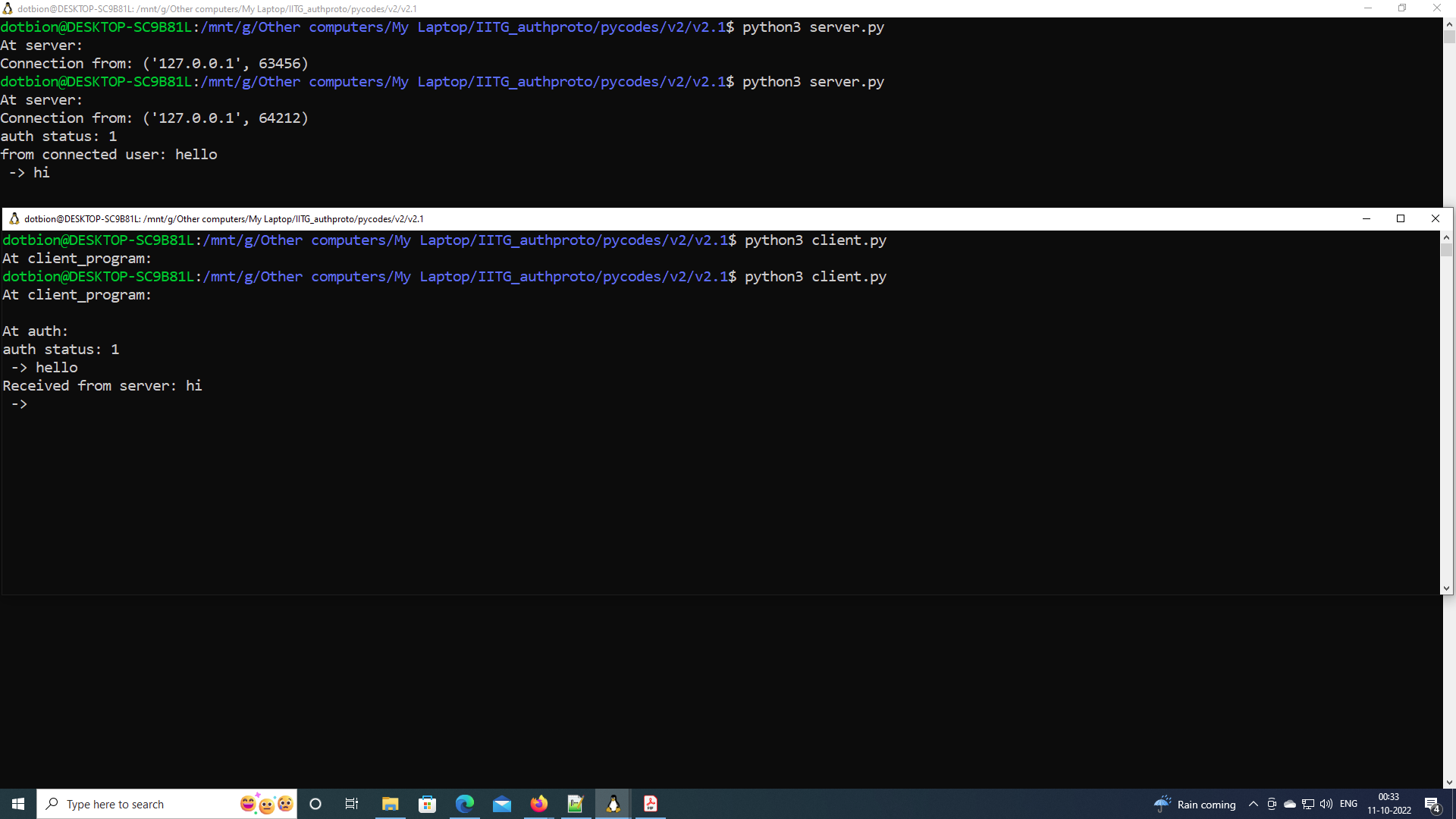
FIG 4. Implementation of adaptive nature in the proposed protocol with encryption enabled in the authentication phase.

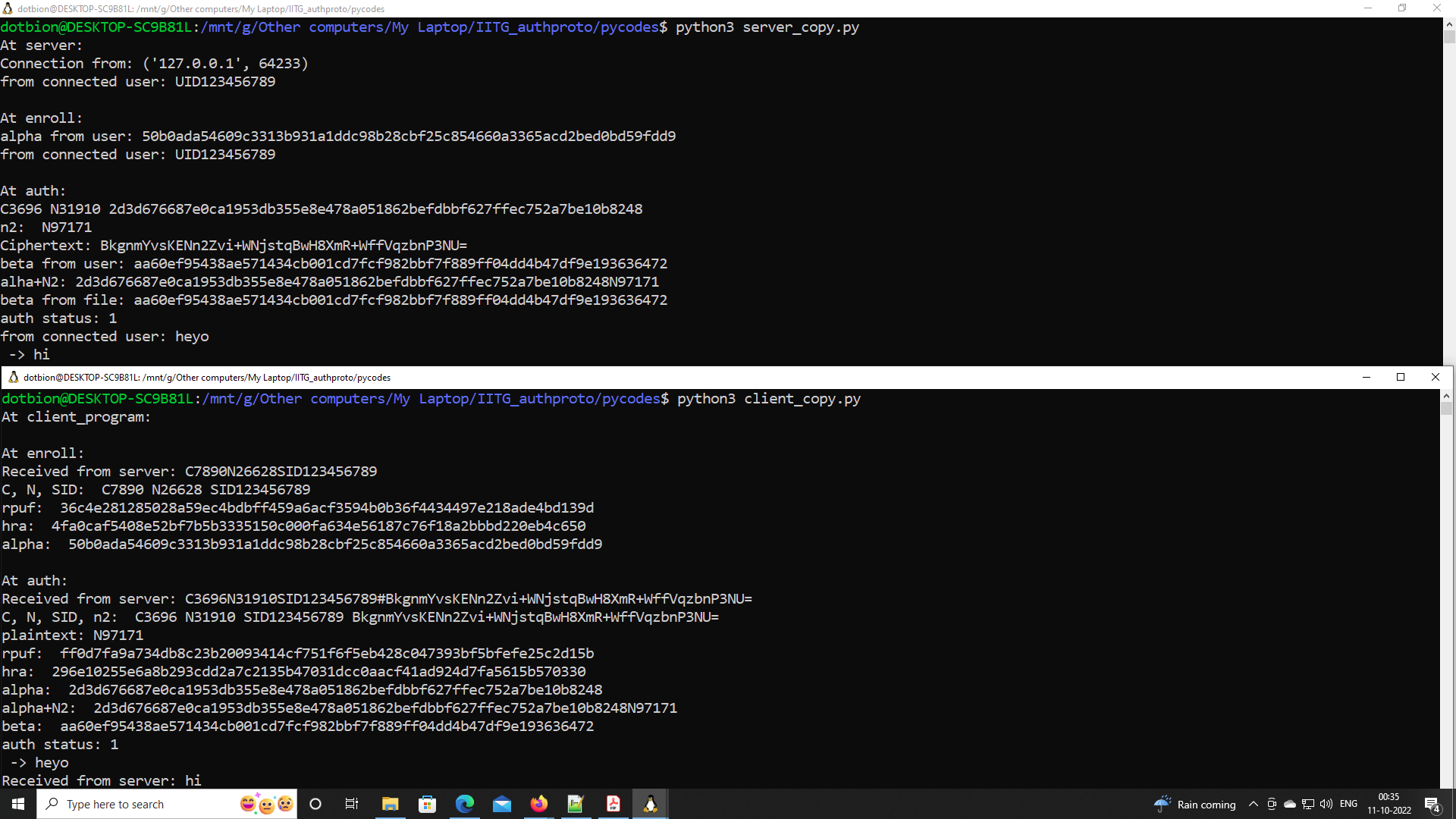
## IV. RESULTS

The above protocol has been designed in Python3 and executed to get the desired result. The same has been categorized into the following scenarios:

*Scenario 1: Legal User*

In this case, a new user is authenticated with whom the key has been shared, tries to authenticate and is successfully authenticated by the server.



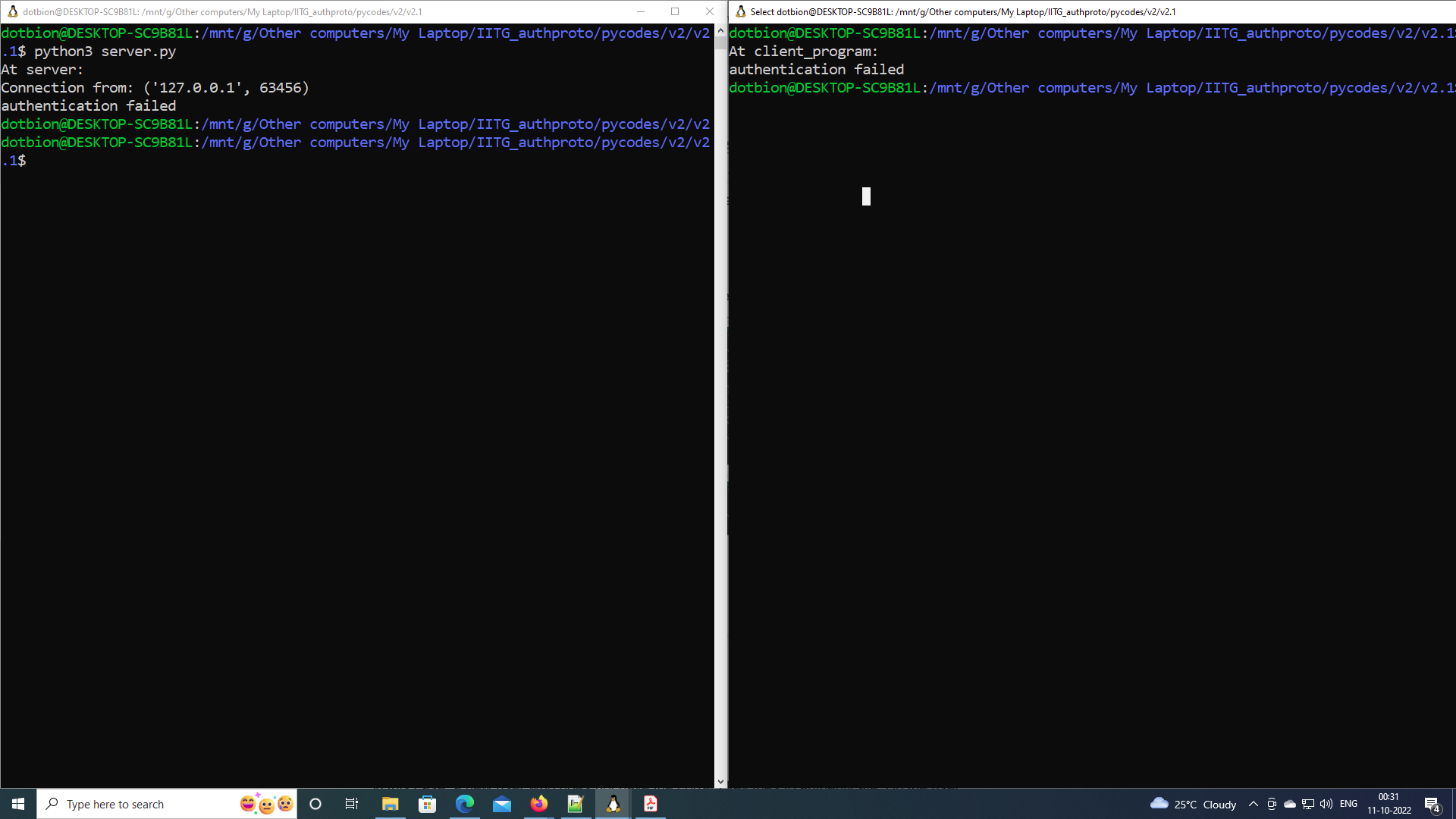


*Scenario 2: Illegal user*

In this case, a new user who does not have the correct key tries to authenticate. So, the user authentication fails.

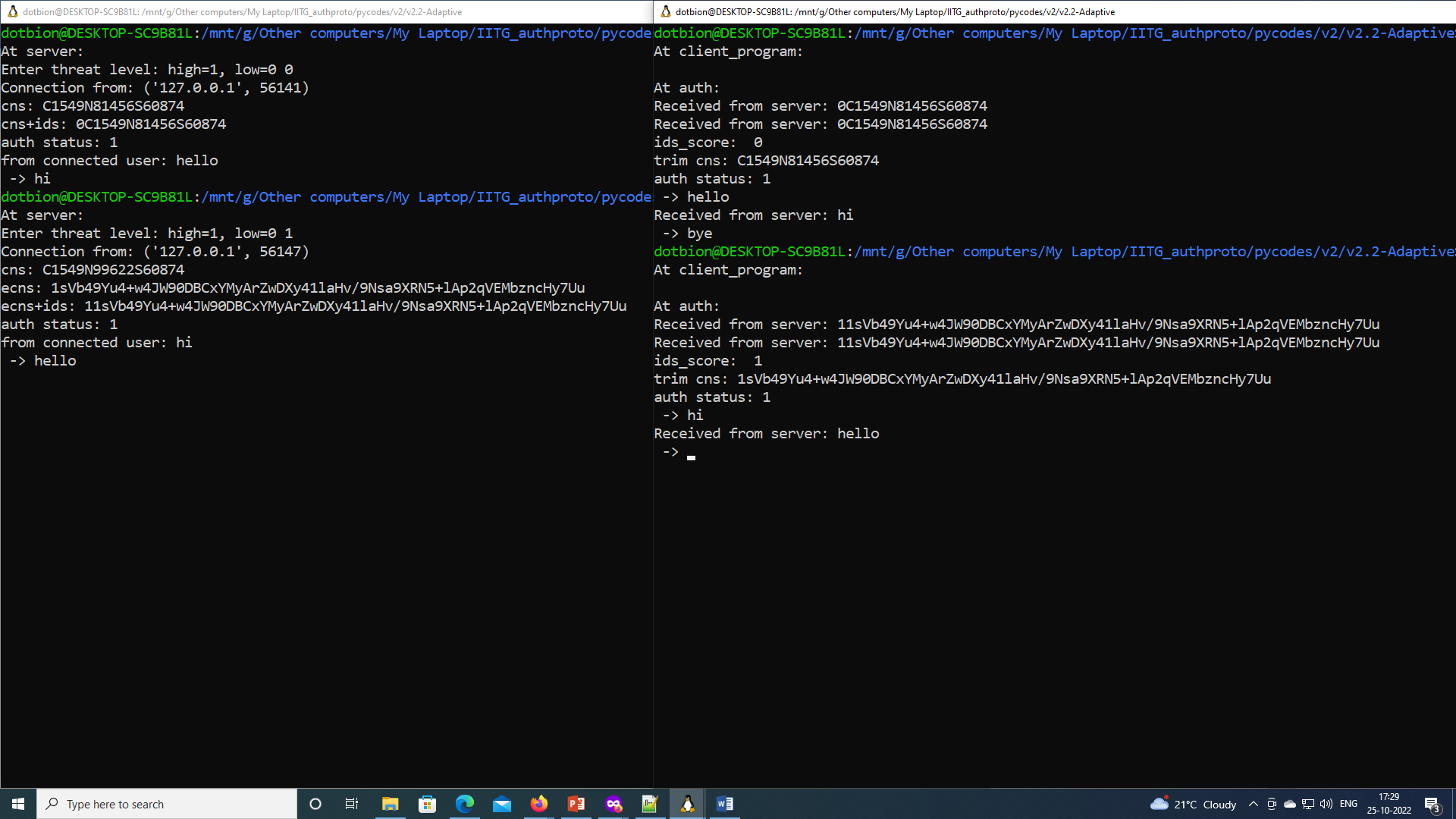
OR

A user which is not enrolled with the server tries to authenticate.



*Scenario 3: Adaptive nature*

In this case the protocol uses encryption-decryption at authentication phase if the threat score from the ids is high and otherwise does not use. For our convenience, at the moment we take the threat score as a boolean input from the user.



## V. DISCUSSION

**Prevention of Man in the Middle Attack in the proposed protocol:**

***Case 1:*** when Jeff sends its own user id (let Jid) for enrollment to the server.

The server does not find any pre-shared key for Jid hence the protocol does not proceed.

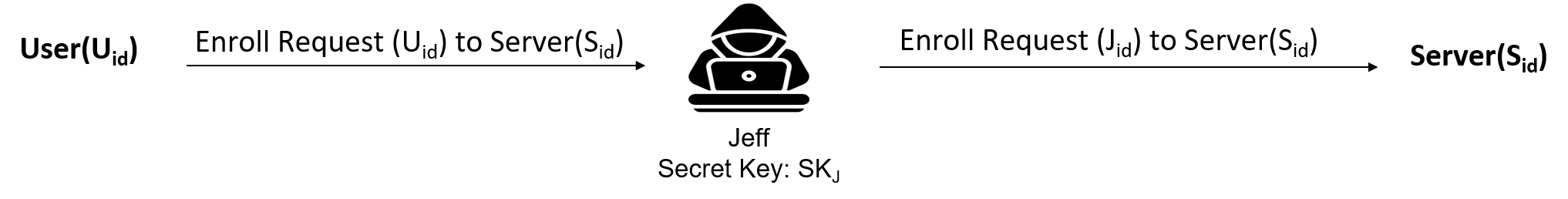


FIG 3. Prevention of Man In the Middle Attack in the Enrollment Phase by using the attackers own id.

***Case 2:*** when Jeff uses a legitimate user id but modifies the next set of data - *cs, eac*

In this case, Jeff uses a different key and encrypts its own challenge *cJ* and seed and sends it to the legitimate user and similarly encrypts its own alpha and challenge *cJ* and sends it back to the server. Since the encryption and decryption keys do not match the received challenge at server is never the same as the one sent. Hence, the user data is not stored in the server. This prevents Jeff from enrolling itself at the server and further authentication also fails. The same is shown in FIG 4 below.

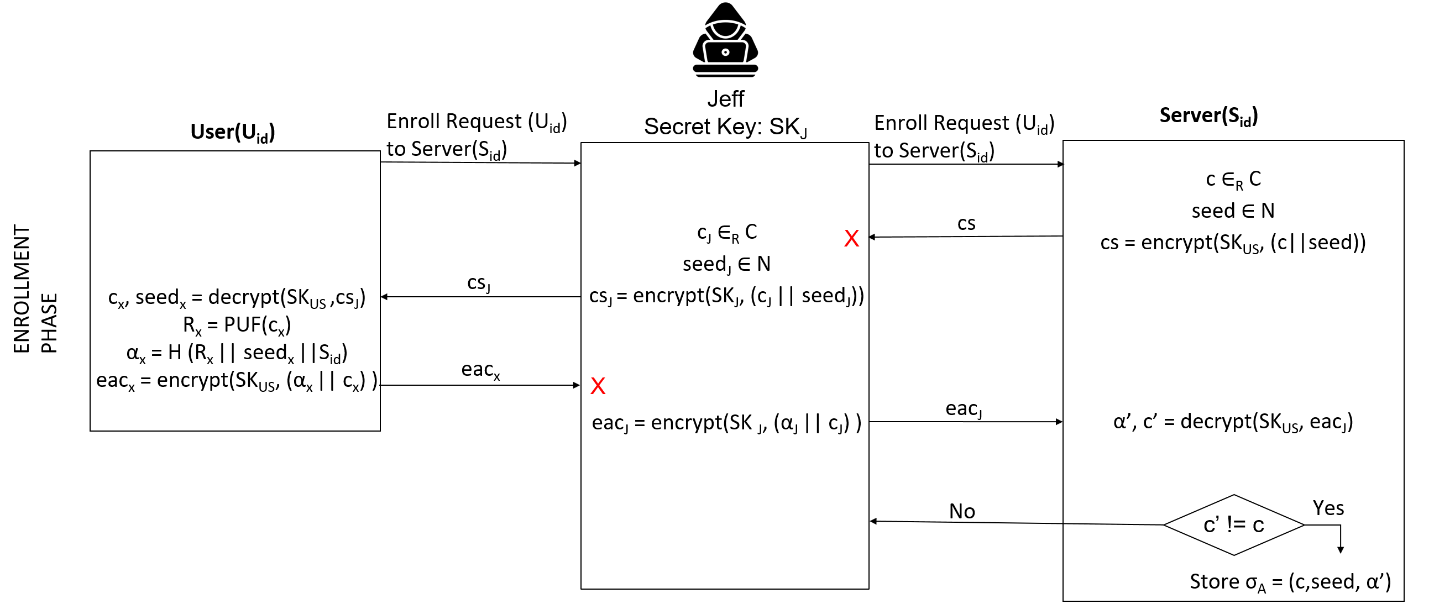


FIG 4. Prevention of Man in the Middle Attack in the Enrollment Phase

***Case 3:*** when Jeff acquires *cs* and *eac* in the enrollment phase and tries to perform man in the middle attack in the authentication phase.

In this case, Jeff sends its own challenge and seed or the ones received from the server, to the user. Alongside it calculates *β’J* and sends it to the server. Now since the PUF responses of Jeff will not be the same as the user, the alpha values generated by Jeff will not match the one already stored at the server. This will lead to the authentication failure. The same is depicted in FIG. 5 given below.

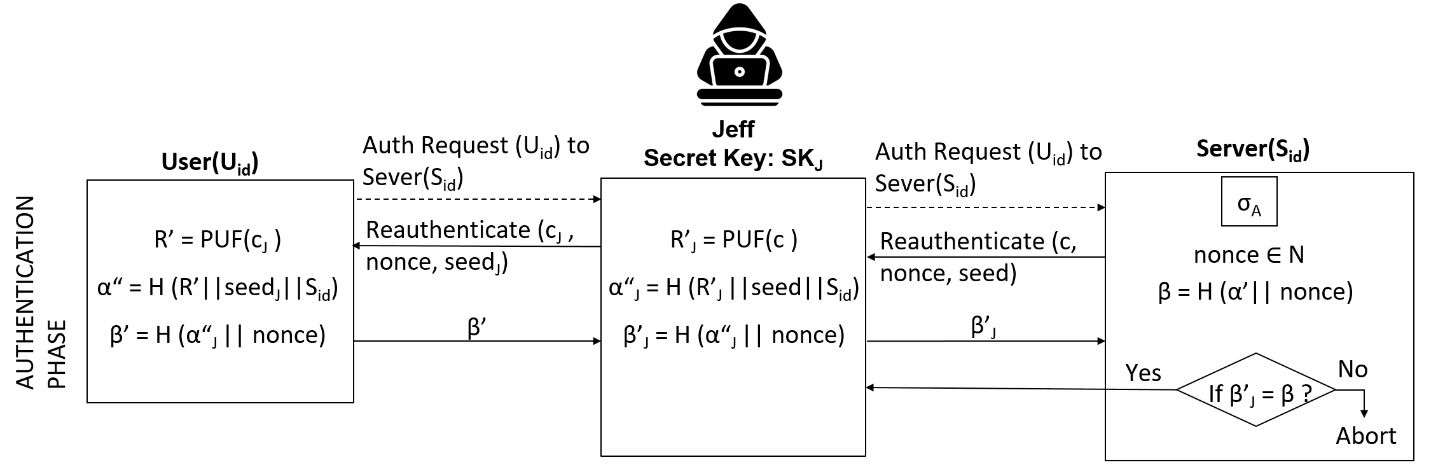


FIG. 5: Prevention of Man in the Middle in Authentication Phase

***Summary:***

In our proposed system, man in the middle attack is not possible as the key is shared in a secured environment. So, now even if Jeff intercepts messages from the user and server during the enrollment process it cannot decrypt the messages as Jeff does not have the key. Now, had there been no check involved at the server end before storing the user details then a different *eac* sent by an adversary would have been saved and consequently in the authentication phase the adversary would be able to authenticate. To prevent this from happening we have put a check on the challenge that was initially sent and the one received, so that only the intended user is able to enroll itself with the server.

Moreover, in the authentication phase all communication occurs in plaintext. Even if Jeff intercepts these messages, Jeff will never be able to generate the PUF response for the challenge and thereby the β' generated by Jeff will never be the one generated by the intended user. As a consequence, at the server, there will be a mismatch between the generated β and the received β' so the authentication will abort. This proves that the system is resilient against any MitM attacks.

In this scheme, although there is a possibility that the adversary might map the plaintext messages in authentication phase (c,nonce,seed → β) and may develop a machine learning model to generate the mapping but the probability of such a scenario is negligible as the keys, seed and α values for an enrolled user is made to change over a period of time. So, the data mapped so far gets invalidated. The adaptation scheme implemented, completely solves this problem.

**Prevention of Replay Attack in the proposed protocol:**

***Case 1:*** During Enrollment Phase

In this phase, if an adversary forwards the messages to either side, it gets no benefit as the messages to and from the server are encrypted. If an adversary tries to use the previous values to enroll another user, it will fail because the challenge and seed generated for every user will be different. Hence the previous values will not be valid.

***Case 2:*** During Authentication Phase

In this phase, the nonce generated by the server ensures freshness of the messages passed, so an adversary won't be able to use the same messages at a later point in time as the nonce value will change.

## VI. REFERENCES

[1] Mohamed Amine Ferrag, Leandros A. Maglaras, Helge Janicke, Jianmin Jiang, Lei Shu, "Authentication Protocols for Internet of Things: A Comprehensive Survey", *Security and Communication Networks*, vol. 2017, Article ID 6562953, 41 pages, 2017.

[2] Boyapally Harishma, Paulson Mathew, Sikhar Patranabis, Urbi Chatterjee, Umang Agarwal, Manu Maheshwari, Soumyajit Dey and Debdeep Mukhopadhyay, “Safe is the New Smart: PUF-Based Authentication for Load Modification-Resistant Smart Meters”, IEEE Trans. Dependable and Secure Computing, vol. 19, no. 1, pp 663-680, Jan 2022.

[3] Prosanta Gope and Biplab Sikdar, “Lightweight and Privacy Preserving Two-Factor Authentication Scheme for IoT Devices”, IEEE Internet of Things, vol. 6, no. 1, pp 580-589, Feb 2019.

[4] Mehedi Masud, Gurjot Singh Gaba, Karanjeet Choudhary, M. SHamim Hossain, Mohammed F. Alhamid and Ghulam Muhammad, “Lightweight and Anonymity Preserving User Authentication Scheme for IoT based Healthcare”, IEEE Internet of Things, vol. 9, no. 4, pp 2649-2656, Feb 2022.

[5] Osman Salem, Khalid Alsubhi, Aymen Shaafi, Mostafa Gheryani, Ahmed Mehaoua and Raouf Boutaba,”Man-in-the-Middle Attack Mitigation in Internet of Medical Things”, IEEE Trans. On Industrial Informatics, vol 18, no. 3, pp 2053-2062, Mar 2022

[6] Nikita Konstantinovich Chistousov, Igor Anatolyevich Kalmykov, Daniil Vyacheslavovich Dukhovnyj, Maksim Igorevich Kalmykov and Aleksandr Anatolyevich Olenev, “Adaptive Authentication Protocol Based on Zero-Knowledge Proof”, MDPI Algorithms, 2022

[7] J. Granjal, E. Monteiro and J. Sá Silva, "Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues," in IEEE Communications Surveys & Tutorials, vol. 17, no. 3, pp. 1294-1312, 2015.

[8] S. Sahoo, T. Dragičević and F. Blaabjerg, "Cyber Security in Control of Grid-Tied Power Electronic Converters—Challenges and Vulnerabilities," in IEEE Journal of Emerging and Selected Topics in Power Electronics, vol. 9, no. 5, pp. 5326-5340, Oct. 2021.

[9] Sowmya Ravidas, Alexios Lekidis, Federica Paci, Nicola Zannone, “Access control in Internet-of-Things: A survey,” Journal of Network and Computer Applications, Vol 144, pp 79-101, 2019

[10] J. Zhang, Z. Wang, Z. Yang and Q. Zhang, "Proximity based IoT device authentication," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1-9.

[11] Q. Zhou, M. Elbadry, F. Ye and Y. Yang, "Heracles: Scalable, Fine-Grained Access Control for Internet-of-Things in Enterprise Environments," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 1772-1780.

[12] A. Arfaoui, S. Cherkaoui, A. Kribeche, S. M. Senouci and M. Hamdi, "Context-Aware Adaptive Authentication and Authorization in Internet of Things," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1-6.

[13] Patricia Arias-Cabarcos, Christian Krupitzer, and Christian Becker. 2019. A Survey on Adaptive Authentication. ACM Comput. Surv. 52, 4, Article 80 (July 2020).

[14] A. P. Johnson, R. S. Chakraborty and D. Mukhopadhyay, "A PUF-Enabled Secure Architecture for FPGA-Based IoT Applications," in IEEE Transactions on Multi-Scale Computing Systems, vol. 1, no. 2, pp. 110-122, 1 April-June 2015.

[15] D. Mukhopadhyay, "PUFs as Promising Tools for Security in Internet of Things," in IEEE Design & Test, vol. 33, no. 3, pp. 103-115, June 2016.