Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **70**, *4*, pp. 585–590, Bucarest, 2025

WIRELESS AUTHENTICATION SYSTEM FOR INTERNET OF THINGS USING FREERADIUS AND BLOCKCHAIN

CEZAR-GABRIEL DUMITRACHE[1], CONSTANTIN VIOREL MARIAN[2], GABRIEL PREDUSCA[3], FLAVIA MARIA BARBU[1], MIHNEA NEFERU[1]

Keywords: Internet of Things (IoT); FreeRADIUS; Private blockchain; Network security.

Wi-Fi networks are essential for wireless connectivity in homes, businesses, and public areas. However, many people still rely on the outdated Wi-Fi Protected Access 2 (WPA2) protocol, which is well-known for its security weaknesses. These allow attackers to intercept and steal data or gain unauthorized access. While WPA3 offers improved security through stronger encryption and authentication, its adoption has been limited due to the fact that many older devices do not support it. To address this challenge, we propose a decentralized authentication system that integrates a FreeRADIUS server with a locally hosted validation mechanism. We implemented a private blockchain using Ganache to verify whether a device's certificate hash is registered. While the system is designed to function without reliance on a single backend, our prototype hosts all components on the same machine; a production deployment would require distributed nodes for improved resilience. Our results show that the blockchain-based verification adds approximately 300 milliseconds to the authentication process. However, once authenticated, communication remains stable, indicating that the proposed approach is both practical and secure for device verification in such networks.

1. INTRODUCTION

In recent years, the Internet of Things (IoT) has expanded with applications in healthcare, industry, agriculture, and smart cities [1]. While this technological expansion offers notable advantages, it also raises serious concerns, particularly regarding the security of communication between devices. One of the most critical issues is authentication. Many IoT devices rely on Wi-Fi connectivity, yet still use weak or unencrypted methods [2], leaving them vulnerable to unauthorized access and data breaches.

In response to these concerns, FreeRADIUS has emerged as a widely adopted **A**uthentication, **A**uthorization, and **A**ccounting (AAA) server, offering a reliable framework for managing device identities. However, as network infrastructures grow in complexity, the limitations of centralized authentication become more evident: these systems, while effective at smaller scales, can become bottlenecks, creating single points of failure that compromise security and efficiency (see §II). Fortunately, recent research suggests that blockchain-based key agreement schemes can offer more reliable alternatives, thereby enhancing security against threats like **M**an-**i**n-**t**he-**M**iddle (MITM) attacks [3]. This decentralization is especially valuable in the context of cross-domain Wi-Fi authentication, where blockchain has been shown to improve both scalability and privacy compared to traditional hierarchy-based systems [4]. As a result, it leads to greater resilience, interoperability, and transparency in authentication infrastructures. Moreover, blockchain technology enables the creation of secure and self-sufficient identity systems [5].

To address these challenges, we propose a decentralized approach: integrating a local blockchain into Wi-Fi authentication by combining FreeRADIUS with Ganache for testing, while Hyperledger Fabric is considered for production deployment, as it is designed to be implemented in larger networks and integrated across multiple nodes, ensuring scalability and enterprise-grade reliability.

We deployed our solution on a single Raspberry Pi 5 (RPi 5) with 8 GB RAM hosting Ganache, the **C**ertificate **A**uthority (CA), and FreeRADIUS to create a self-contained, scalable environment. One of the main contributions of this work is the integration of a local blockchain, which adds an extra layer of security to the wireless authentication process. Our system authenticates device identities without relying on a central authority, thus enhancing security and transparency.

To support this framework, we established a local and dedicated CA that generates and signs digital certificates for client devices. These certificates form the basis of our secure identity verification, with hashes being checked against the blockchain to ensure integrity and prevent unauthorized access.

In addition, we built a network bridge using a second Raspberry Pi to connect IoT devices without native Wi-Fi integration. This enables a wider range of devices to join the network securely (Fig. 1). It is important to note that this bridge can be configured either at Layer 3, by enabling NAT, in which case the IoT devices will be on a different network, or at Layer 2, where the devices will remain on the same network. To streamline identity verification, we implemented a Python script that queries the blockchain to validate each client during authentication.

The rest of the paper is divided as follows: Section II, we provide background and technical context; in Section III, we outline the overview of our architecture, whereas we focus on the application of distributed ledger technologies (DLTs) in Section IV and present our hybrid approach for a secure IoT authentication Section V. In Section VI, we present the experimental results, and Section VII discuss our conclusions and potential research directions.

Fig. 1 – The implemented setup.

[1] Doctoral School of Electronics, Telecommunications & Information Technology, Pitesti center, National University of Science and Technology POLITEHNICA Bucharest, Romania.
[2] National University of Science and Technology POLITEHNICA Bucharest, Romania (corresponding author).
[3] Faculty of Telecommunications and Energy Engineering, Valahia University of Târgovişte, Romania.
E-mails: cezar.dumitrache_433@student.upit.ro, constantin.marian@upb.ro, flavia_maria.barbu@upb.ro, gabriel.predusca@valahia.ro, mihnea.neferu@upb.ro

DOI: 10.59277/RRST-EE.2025.70.4.26

2. BACKGROUND AND TECHNICAL CONTEXT

As IoT grows, decentralized methods gain attention. DLTs provide a strong foundation for secure device identity, with two notable platforms highlighted below.

Hyperledger: A primary software platform that implements DLT is Hyperledger, a collaborative open-source project hosted by the Linux Foundation [6], designed to develop blockchain technologies for enterprise applications. It offers a modular, flexible, and scalable framework, adaptable to various use cases including supply chain traceability, identity management, and smart contracts [7].

Ethereum: On the other hand, there is Ethereum, a public and decentralized blockchain network known for supporting smart contracts and decentralized applications [8]. As one of the most adopted blockchain platforms, it is closely associated with cryptocurrencies and the broader Web3 ecosystem [9].

Ganache: A personal blockchain for Ethereum development, is used as a local simulator that provides a safe and controllable environment for testing our DLT-based authentication setup before deployment on a real network.

We chose to focus on the Ganache and Hyperledger Fabric frameworks, as they form the foundation of our implementation and testing, respectively. As shown in Fig. 2, we initially set up a Certificate Authority (CA) that issues and signs digital certificates for devices attempting to join the network. We firmly assert that this design significantly enhances transparency and security by enabling decentralized identity validation, thereby mitigating the risks associated with single points of failure.

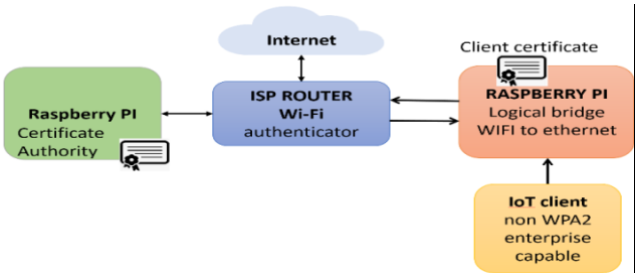

Fig. 2 – Topology of our setup.

2.1 WPA2 AUTHENTICATION WITH FREERADIUS: A BASELINE FOR COMPARISON TO WPA3

Wi-Fi security primarily relies on WPA2, with WPA3 introduced as a more secure alternative. However, when WPA2 is paired with a RADIUS server, it can offer a level of security comparable to WPA3, making it a practical solution in environments where upgrading the whole infrastructure, including all devices, is not feasible from an economic point of view. Both standards use the **A**dvanced **E**ncryption **S**tandard (AES), but WPA3 includes several key enhancements, which are mentioned below:

Individualized Data Encryption: WPA3 provides unique encryption for each session, improving privacy on open networks, a feature known as **O**pportunistic **W**ireless **E**ncryption (OWE).

Authentication Mechanism: WPA2 uses a four-way handshake, which is vulnerable to KRACK attacks. For stronger protection against offline dictionary attacks, WPA3 uses **S**imultaneous **A**uthentication of **E**quals (SAE).

Forward Secrecy: SAE in WPA3 also provides forward secrecy, meaning that if a password is compromised in the future, past session data remains secure.

Stronger Password-Based Authentication: WPA3 improves resilience even when users choose weak passwords.

However, the adoption of WPA3 is limited by hardware incompatibility, which means many networks continue to rely on older devices. As an alternative, we propose enhancing existing WPA2 infrastructures, which can provide WPA3-like security at a lower cost. This can be achieved through a FreeRADIUS server, enabling flexible and efficient authentication while maintaining a high level of security. Instead of shared WPA2 passwords, smart contracts securely authenticate users, removing centralized password storage and reducing credential theft. Blockchain's immutable ledger also prevents tampering and improves access traceability.

2.1.1 CHALLENGES AND LIMITATIONS

The following challenges were considered during our implementation:

Complexity: Integrating blockchain into an existing Wi-Fi infrastructure undoubtedly adds complexity to the system.

Privacy: While blockchain offers immutability and transparency, it also raises privacy concerns. Storing authentication-related data on a public or semi-public ledger could expose sensitive user information.

Interoperability and Standardization: Wi-Fi authentication needs to function across a wide variety of devices, network infrastructures, and service providers. The lack of standardized protocols for blockchain-based authentication limits interoperability and contributes to system fragmentation.

Our approach was tested for small business environments. Compared to WPA3, which would require replacing numerous devices, such as access points, we keep costs minimal by implementing this authentication method.

2.1.2 POTENTIAL WORKFLOW

For implementation and testing, we used Ganache on a Raspberry Pi, with Hyperledger Fabric planned for future deployment. This setup enabled simulation of digital identity registration and validation on a simplified private blockchain as follows:

User registration: Performed via Remix Ethereum, where the administrator records the hash on the deployed smart contract (Fig. 3).

Fig. 3 – Using Remix Ethereum to record a hash in the blockchain.

Authentication request: The authorized section within FreeRADIUS was modified so that, after the client is authenticated via **T**ransport **L**ayer **S**ecurity (TLS), an additional verification step is performed. At this stage, a script is triggered to compare the hash of the client's certificate with a previously stored value, ensuring the certificate's integrity and authenticity.

Credential validation: A smart contract (Fig. 4) verifies the request, ensuring the credentials match the stored records.

Access control: If authentication is successful, the user is granted access with 0; if not, 1 is returned.

```
w3 = Web3(Web3.HTTPProvider("http://127.0.0.1:8545"))
if not w3.is_connected():
    print("Error: Cannot connect to block chain")
    sys.exit(1)

contract_address = Web3.to_checksum_address("0x4AC06803B0fD12692166fA601d8393ae7Bf931cC")

abi = [
    {
        "inputs": [{"internalType": "bytes32", "name": "certHash", "type": "bytes32"}],
        "name": "isCertAllowed",
        "outputs": [{"internalType": "bool", "name": "", "type": "bool"}],
        "stateMutability": "view",
        "type": "function"
    }
]

contract = w3.eth.contract(address=contract_address, abi=abi)

try:
    is_allowed = contract.functions.isCertAllowed(hash_bytes32).call()
except Exception as e:
    print(f"Blockchain call failed: {e}")
    sys.exit(1)

sys.exit(0 if is_allowed else 1)
```

Fig. 4 – Validation of the stored hash from the blockchain.

3. THE OVERVIEW OF OUR ARCHITECTURE

To establish a distributed, secure, and autonomous authentication process, we propose an architecture that relies on four main components:

- the IoT device;
- the access point (AP);
- the FreeRADIUS server;
- the Ganache blockchain platform.

Each component is essential to authentication and access control. The setup in Fig. 5 links FreeRADIUS to Ganache. For stronger security, we added a second verification step after the initial TLS authentication: a smart contract checks whether the client's certificate hash is already registered on the blockchain.

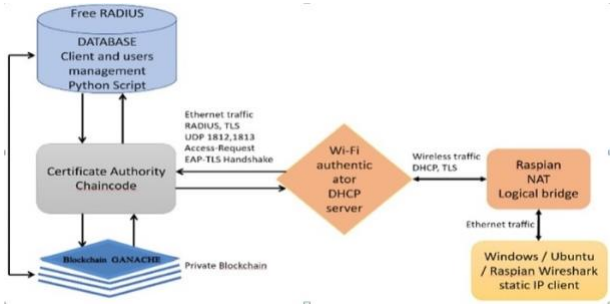

Fig. 5 – Logical setup.

3.1 ROLE OF THE WIRELESS AP

The AP acts as a proxy, relaying communications from IoT devices to the authentication infrastructure. Although it does not perform local validation, it ensures the transparent transfer of **E**xtensible **A**uthentication **P**rotocol (EAP) messages between the client and the backend.

3.2 FREERADIUS, CA, GANACHE

This section presents the logical and security core of an Enterprise Wi-Fi infrastructure, which combines digital authentication with blockchain validation via FreeRADIUS and Ganache. While local authentication is supported, this implementation focuses on blockchain-based identity validation. The server receives Wi-Fi authentication requests and runs a Python script that:

- Calculates the hash of the presented certificate.
- Queries the blockchain to check if the hash is already registered and decides whether the authentication is valid.

A CA is a trusted entity that issues, validates, and manages digital certificates, confirming an entity's identity [10]. Depending on the certificate type, the verification process may be automated or involve manual validation steps [11]. Once validated, the CA signs the certificate, which contains the applicant's public key, identity, validity period, and the CA's signature. Its main stages are outlined as follows:

Certificate usage: Digital certificates are used for authentication and encryption. For instance, when a website presents its digital certificate to a browser, it proves legitimacy and establishes an encrypted connection, ensuring secure communication between the user and the server.

Certificate examination: Any entity receiving the certificate can verify its validity through the CA's digital signature. The identity is positively verified if the signature is valid and if the certificate is not expired or revoked.

Certificate revocation: In case of compromise or expiration, the CA revokes the certificate by including it in a **C**ertificate **R**evocation **L**ist or by enabling real-time status verification via the **O**nline **C**ertificate **S**tatus **P**rotocol.

Role of the CA in digital security: CAs are essential for establishing trust in digital environments. They support secure online communications, protect financial transactions, and ensure the authentication of users and devices [12]. Without a trusted CA, online identities cannot be verified, increasing the risk of attacks such as man-in-the-middle.

We used Ganache on a Raspberry Pi to record certificate hashes when they are issued, thereby providing a compact, single-node solution for role and identity management [13]. Selecting the appropriate hardware platform is essential for such implementations, as prior studies highlight the importance of flexible hardware capable of working efficiently with heterogeneous devices [14, 15]. In our setup, Ganache supports smart contracts (chaincode) and is optimized for high performance in private networks, which makes it particularly suitable for local deployments [16].

The blockchain module performs three critical functions:

- verify the authentication token presented by a device;
- validate the digital signature via the device's public key;
- enforce authorization policies via smart contracts.

3.3 RASPBERRY PI – LOGICAL BRIDGE

The Raspberry Pi creates a logical bridge and forwards packets bidirectionally between *wlan0* and *eth0*, providing network access to the IoT device as if it were connected to a wired network [17]. Instead of creating a true **layer 2 bridge**, we configured a layer 3 router with NAT, allowing Ethernet devices to access the internet via Wi-Fi while masking and routing IP packets through NAT.

4. APPLYING BLOCKCHAIN WITHIN THE SYSTEM

To enhance our Wi-Fi infrastructure, we integrated blockchain validation, leveraging its decentralized approach to data storage and verification [18]. For this, we relied on Ganache to allow participants to replicate and synchronize data within our server environment [19], thus ensuring transparent recordkeeping without depending on a central authority [20]. We primarily employed it for its transparency, decentralization, and resistance to threats [21].

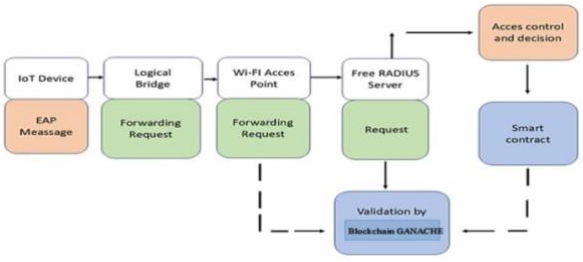

Fig. 6 – IoT connection process.

Figure 6 illustrates the process by which an IoT device connects to the network. The following steps provide a more detailed description:

Connection Initiation: The IoT device initiates an EAP-TLS access request, which is relayed by the AP to the FreeRADIUS server. The server subsequently validates the device's certificate hash against the blockchain.

Response and IP Assignment: Upon successful validation, an *Access-Accept* message is sent. Although our IoT devices use static IPs, the Raspberry Pi handles NAT for routing and address resolution.

5. A HYBRID ARCHITECTURE FOR SECURE IOT AUTHENTICATION

Our system enables secure IoT Wi-Fi authentication using a hybrid architecture [22] with a Wi-Fi AP, a FreeRADIUS server, and a Ganache blockchain, all hosted locally on a Raspberry Pi. The authentication description:

Token Generation by the IoT Device: Upon connection, the IoT device creates a digitally signed token containing identity details, a timestamp, and a nonce to prevent replay attacks, ensuring integrity and authenticity via the device's private key.

Authentication Request Forwarding: The AP receives the **E**xtensible **A**uthentication **P**rotocol (EAP) message with the signed token and forwards it to the FreeRADIUS server, acting solely as a bridge without performing cryptographic verification.

FreeRADIUS Script Execution: The server runs our custom script via the *radmin* module, extracting the certificate hash and interacting with Hyperledger Fabric through a local SDK application.

Blockchain Verification via Smart Contract: A smart contract on the Ganache blockchain verifies the device's digital signature and hash against registered entries, returning a success response if valid.

Access Decision: The FreeRADIUS script receives the blockchain response. If verification succeeds, FreeRADIUS sends an *Access-Accept* message to the AP, granting network access to the IoT device. If verification fails, the device is denied access using an *Access-Reject* message toward the AP.

5.1 ADVANTAGES

The authentication uses FreeRADIUS and Ganache for quick deployment on existing hardware. Blockchain-based certificate validation enables decentralized Wi-Fi identity management, offering several key advantages:

- Fully local infrastructure with no external services;
- Improved security and complete control over authentication data;
- Minimal operational costs due to low-cost hardware.
- Potential for integrating biometric user identification.
- EAP-TLS for strong security and no password transfer.
- Tamper-resistant identity validation.
- Clear modular separation between system components (AP, RADIUS, blockchain).
- Enterprise-level features using affordable devices.

6. RESULTS

Ganache is the most resource-intensive component of our system. It can run on a Raspberry Pi 4 with 8 GB RAM, this setup is suitable primarily for small networks, such as those operated by small to medium-sized businesses with a few hundred employees. In such environments, authentication requests typically amount to only a few dozen per minute even under peak load, which our Ganache-based solution can handle effectively. Larger infrastructures (hotels, conference centers, large enterprise environments, etc.) where hundreds of users may attempt to authenticate simultaneously during events or group activities, require a more robust DLT system.

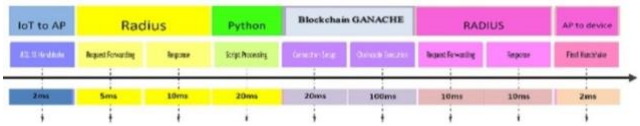

Fig. 6 – Results obtained when the device is authenticated in an Wi-Fi network.

Figure 7 shows the authentication workflow for an IoT device, including the EAP-TLS handshake and blockchain-based certificate validation. The 802.1X handshake with the AP takes 2 ms, followed by initial verification on FreeRADIUS. Then, a custom script connects to Ganache, where a smart contract validates the certificate hash, requiring about 100 milliseconds. The result is returned via RADIUS, concluding with a final AP-device handshake.

```
cez@raspberrypi:~ $ iperf3 -c 192.168.50.250
Connecting to host 192.168.50.250, port 5201
[  5] local 192.168.50.147 port 40264 connected to 192.168.50.250 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  28.9 MBytes   242 Mbits/sec    0    676 KBytes
[  5]   1.00-2.00   sec  21.2 MBytes   178 Mbits/sec    0    754 KBytes
[  5]   2.00-3.00   sec  26.2 MBytes   220 Mbits/sec    0    994 KBytes
[  5]   3.00-4.00   sec  25.0 MBytes   210 Mbits/sec    0   1.12 MBytes
[  5]   4.00-5.00   sec  25.0 MBytes   210 Mbits/sec    0   1.25 MBytes
[  5]   5.00-6.00   sec  27.5 MBytes   231 Mbits/sec    0   1.31 MBytes
[  5]   6.00-7.00   sec  27.5 MBytes   231 Mbits/sec    0   1.31 MBytes
[  5]   7.00-8.00   sec  26.2 MBytes   220 Mbits/sec    0   1.31 MBytes
[  5]   8.00-9.00   sec  26.2 MBytes   220 Mbits/sec    0   1.31 MBytes
[  5]   9.00-10.00  sec  27.5 MBytes   231 Mbits/sec    0   1.31 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec   261 MBytes   219 Mbits/sec    0             sender
[  5]   0.00-10.01  sec   258 MBytes   216 Mbits/sec                  receiver
```

Fig. 7 – Iperf3 test to server situated on the RaspberryPi with Freeradius and Hyperledger using TCP.

```
cez@raspberrypi:~ $ iperf3 -c 192.168.50.250 -u
Connecting to host 192.168.50.250, port 5201
[  5] local 192.168.50.147 port 35289 connected to 192.168.50.250 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   1.00-2.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   2.00-3.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   3.00-4.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   4.00-5.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   5.00-6.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   6.00-7.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   7.00-8.00   sec   129 KBytes  1.05 Mbits/sec  91
[  5]   8.00-9.00   sec   127 KBytes  1.04 Mbits/sec  90
[  5]   9.00-10.00  sec   129 KBytes  1.05 Mbits/sec  91
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-10.00  sec  1.25 MBytes  1.05 Mbits/sec  0.000 ms  0/906 (0%)  sender
[  5]   0.00-10.01  sec  1.25 MBytes  1.05 Mbits/sec  0.035 ms  0/906 (0%)  receiver
```

Fig. 8 – Iperf3 test from the client situated on the Raspberry Pi with a logical bridge using UDP.

Overall, authentication takes ~179 ms, mostly due to blockchain interactions. While this delay is noticeable, it ensures enhanced security. Using EAP-TLS with blockchain raises initial authentication latency to ~300 ms, but subsequent communications proceed without extra overhead once the device is registered. After completing the authentication process, we assessed network performance using iPerf3 (Fig. 8, Fig. 9). Once authenticated, devices communicate without noticeable latency or bottlenecks.

The difference in results does not reflect the inherent performance of TCP versus UDP. During the iPerf3 test, TCP adapts its transmission rate to maximize bandwidth use, while UDP maintains a constant sending rate. This explains the variations seen in Fig. 8 and Fig. 9.

6.1 A FREERADIUS AUTHENTICATION WORKFLOW

FreeRADIUS was executed in debug mode (freeradius -X), allowing successful authentications while providing detailed output for TLS and blockchain analysis.

```
(164) Received Access-Request Id 149 from 192.168.50.1:34925 to 192.168.50.250:1812 length 177
(164)   User-Name = "cezar"
(164)   NAS-IP-Address = 192.168.50.1
(164)   NAS-Identifier = "RalinkAP0"
(164)   NAS-Port = 17
(164)   Called-Station-Id = "E8-9C-25-09-0F-58"
(164)   Calling-Station-Id = "2C-CF-67-71-4D-0B"
(164)   WLAN-Pairwise-Cipher = 1027076
(164)   WLAN-Group-Cipher = 1027076
(164)   WLAN-AKM-Suite = 1027073
(164)   WLAN-Group-Mgmt-Cipher = 1027078
(164)   Framed-MTU = 1400
(164)   NAS-Port-Type = Wireless-802.11
(164)   EAP-Message = 0x020900060d00
(164)   HS20-AP-Version = 2
(164)   State = 0xd94f6aaede46677dc2c373c0eb3d378e
(164)   Message-Authenticator = 0x7794207cc98102bcff033facae751b62
(164) Restoring &session-state
(164)   &session-state:Framed-MTU = 994
(164)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(164)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(164)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(164)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(164)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(164)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(164)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Certificate"
(164)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"
(164)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, CertificateVerify"
(164)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"
(164)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"
(164)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"
(164)   &session-state:TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"
(164)   &session-state:TLS-Session-Version = "TLS 1.2"
(164) # Executing section authorize from file /etc/freeradius/3.0/sites-enabled/default
(164)   authorize {
(164)     [preprocess] = ok
```

Fig. 9 – First output from the FreeRADIUS –X command.

As shown in Fig. 10, the server first receives an *Access-Request* from the client. It then performs a TLS handshake to establish a secure communication channel. Afterward, the authorize function is called, triggering our custom script to verify whether the hash of the negotiated certificate is recorded on the blockchain, a process explained in more detail in Fig. 11.

```
(164) eap: Peer sent EAP Response (code 2) ID 9 length 6
(164) eap: No EAP Start, assuming it's an on-going EAP conversation
(164)     [eap] = updated
(164)     [files] = noop
(164)     [expiration] = noop
(164)     [logintime] = noop
(164)     [pap] = noop
(164)     [unix] = notfound
(164) extract_cert: Executing: /bin/sh:
(164) extract_cert: Program returned code (0) and output ''
(164) extract_cert: Program executed successfully
(164)     [extract_cert] = ok
(164) check_blockchain: Executing: /usr/bin/python3:
(164) check_blockchain: Program returned code (0) and output ''
(164) check_blockchain: Program executed successfully
(164)     [check_blockchain] = ok
(164)   } # authorize = updated
(164) Found Auth-Type = eap
(164) # Executing group from file /etc/freeradius/3.0/sites-enabled/default
(164)   authenticate {
(164) eap: Expiring EAP session with state 0xd94f6aaede46677d
(164) eap: Finished EAP session with state 0xd94f6aaede46677d
(164) eap: Previous EAP request found for state 0xd94f6aaede46677d, released from the list
(164) eap: Peer sent packet with method EAP TLS (13)
(164) eap: Calling submodule eap_tls to process data
(164) eap_tls: (TLS) Peer ACKed our handshake fragment.  handshake is finished
(164) eap: Sending EAP Success (code 3) ID 9 length 4
(164) eap: Freeing handler
(164)     [eap] = ok
(164)   } # authenticate = ok
```

Fig. 10 – Second output from the FreeRADIUS –X command.

Fig. 12 depicts the final stage of the authentication flow. Once all necessary validations are complete, FreeRADIUS sends an Access-Accept message to the authenticator. Upon receiving this message, the authenticator grants the network access, thereby completing the authentication process.

```
164) Sent Access-Accept Id 149 from 192.168.50.250:1812 to 192.168.50.1:34925 length 173
164)   MS-MPPE-Recv-Key = 0xf2595f1510f44e8cf4f408df5509900f5a91238617642a61fe34b14c7b264cab
164)   MS-MPPE-Send-Key = 0x3b099f1b22e2998d820b6cd7f8ebedb4fe535bb7746dffdf63238153c30d77dd
164)   EAP-Message = 0x03090004
164)   Message-Authenticator = 0x00000000000000000000000000000000
164)   User-Name = "cezar"
164)   Framed-MTU += 994
164) Finished request
Waking up in 4.2 seconds.
156) Cleaning up request packet ID 141 with timestamp +10026 due to cleanup_delay was reached
157) Cleaning up request packet ID 142 with timestamp +10026 due to cleanup_delay was reached
Waking up in 0.1 seconds.
158) Cleaning up request packet ID 143 with timestamp +10026 due to cleanup_delay was reached
Waking up in 0.1 seconds.
159) Cleaning up request packet ID 144 with timestamp +10026 due to cleanup_delay was reached
Waking up in 0.1 seconds.
160) Cleaning up request packet ID 145 with timestamp +10027 due to cleanup_delay was reached
Waking up in 0.1 seconds.
161) Cleaning up request packet ID 146 with timestamp +10027 due to cleanup_delay was reached
162) Cleaning up request packet ID 147 with timestamp +10027 due to cleanup_delay was reached
163) Cleaning up request packet ID 148 with timestamp +10027 due to cleanup_delay was reached
164) Cleaning up request packet ID 149 with timestamp +10027 due to cleanup_delay was reached
Ready to process requests
```

Fig. 11 – Third output from freeradius –X command.

6.2 EXTENDED NETWORK AUTHENTICATION: TESTING AND PERFORMANCE ANALYSIS

To gain further insight into the device authentication process, we conducted an additional experiment using a laptop running Ubuntu Linux, with network traffic captured via Wireshark (Fig. 13). This test allowed us to compare performance with the Raspberry Pi setup, showing how hardware differences and environmental factors influence authentication. The authentication exchange was completed in just 0.18 seconds and was significantly faster than on the Raspberry Pi due to the laptop's more powerful processor and larger antenna, which improve processing speed and signal reception. While the Raspberry Pi offers a low-cost solution, its limited processing power and smaller antenna may slightly reduce performance in comparison. For larger-scale deployments [23], more capable hardware is suitable to ensure enough computational resources.

1	0.000000000	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	EAPOL	18	Start
2	0.006716381	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAP	25	Request, Identity
3	0.006969314	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	EAP	29	Response, Identity
4	0.081210098	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAP	26	Request, TLS EAP (EAP-TLS)
5	0.082163663	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	TLSv1.2	214	Client Hello
6	0.102046411	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAP	1024	Request, TLS EAP (EAP-TLS)
7	0.102154729	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	EAP	24	Response, TLS EAP (EAP-TLS)
8	0.263838640	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAP	1024	Request, TLS EAP (EAP-TLS)
9	0.263949216	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	EAP	24	Response, TLS EAP (EAP-TLS)
10	0.342460807	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAP	1024	Request, TLS EAP (EAP-TLS)
11	0.342576415	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	EAP	24	Response, TLS EAP (EAP-TLS)
12	0.416048918	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	TLSv1.2	155	Server Hello, Certificate, Server Key Exchange,
13	0.422137334	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	EAP	1294	Response, TLS EAP (EAP-TLS)
14	0.518216480	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAP	26	Request, TLS EAP (EAP-TLS)
15	0.518333606	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	TLSv1.2	1290	Ignored Unknown Record
16	0.585576658	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAP	26	Request, TLS EAP (EAP-TLS)
17	0.585700387	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	TLSv1.2	414	Ignored Unknown Record
18	0.662249633	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	TLSv1.2	81	Change Cipher Spec, Encrypted Handshake Message
19	0.662573457	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	EAP	24	Response, TLS EAP (EAP-TLS)
20	0.732164079	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAP	24	Success
21	0.736695110	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAPOL	113	Key (Message 1 of 4)
22	0.736960577	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	EAPOL	141	Key (Message 2 of 4)
23	0.745594322	e8:9c:25:09:0f:58	0c:9a:3c:eb:7c:ce	EAPOL	209	Key (Message 3 of 4)
24	0.745716855	0c:9a:3c:eb:7c:ce	e8:9c:25:09:0f:58	EAPOL	113	Key (Message 4 of 4)
25	0.821476967	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x940e1da7
26	0.843767499	192.168.50.1	192.168.50.190	DHCP	354	DHCP ACK - Transaction ID 0x940e1da7

Fig. 12 – Authenticating an Ubuntu laptop in the topology.

6.3 IDENTIFIED LIMITATIONS

Integrating blockchain into Wi-Fi infrastructure is complex, requiring careful management of protocols and compatibility with existing authentication systems. While it enhances transparency and protects against spoofing and credential theft, implementation remains challenging.

Increased latency: The blockchain integration adds ~100–200 ms to the authentication process. This delay is generally acceptable; it may impact responsiveness or user experience in applications where low latency is critical.

Scalability and Overhead: While the system can be scaled by adding more Ganache peers and RADIUS nodes, this increases configuration and maintenance complexity, particularly on resource limited devices (e.g. Raspberry Pi).

Security: Although blockchain enhances resistance to data tampering, overall system security depends on additional factors, including private key management and securing client devices to prevent certificate compromise.

Configuration complexity: Integrating FreeRADIUS, a CA, and Ganache blockchain requires expertise in networking, cryptography, and blockchain, including secure protocols, certificate management, and DLT setup.

Hardware limitations: The Raspberry Pi 5 (8 GB RAM) is suitable for local or small-scale deployments; its limited resources may not be adequate in production environments with high volumes of multiple authentication requests.

7. CONCLUSIONS

We developed a blockchain-based IoT wireless system

designed to enhance the security of existing WPA2 Wi-Fi networks, aiming to achieve a protection level comparable to that provided by WPA3. Although our system is not optimized for ultra-low latency applications, it provides a well-balanced combination of security, transparency, and efficiency, thereby making it particularly suitable for research environments and personal use. While prior studies have explored blockchain-based network authentication, its implementation in digital identity management remains uncommon [24]. For instance, the approach presented in [25] uses standard user authentication methods without incorporating blockchain or decentralized identity management, indirectly reflecting a preference for conventional solutions. However, we chose to integrate blockchain as it enhances transparency, security, and decentralization (Sections 4 to 6).

As future work, we plan to replace Ganache with Hyperledger Fabric in the deployment phase to better adapt our solution to larger and more complex environments. We also aim to improve the usability of the authentication process to support broader adoption.

CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

Cezar Gabriel Dumitrache: conceptualization, methodology, manuscript structuring, investigation, checking references' validity, writing original draft, writing submitted version, implementation, validation.
Gabriel Predusca: investigation, writing, review, and editing.
Flavia Maria Barbu: manuscript structuring, checking references validity, writing-review and editing, writing-submitted version.
Mihnea Neferu: investigation, writing, review, and editing.
Constantin Viorel Marian (corresponding author): conceptualization, methodology, figure curation, writing-review and editing, supervising.

Received on 21 June 2025

REFERENCES

1. R. Almutairi, G. Bergami, and G. Morgan, *Advancements and Challenges in IoT Simulators: A Comprehensive Review*, Sensors, **24**, *5*, pp. 1511 (2024).
2. M. Almasre and A. Subahi, *Create a Realistic IoT Dataset Using Conditional Generative Adversarial Network*, Journal of Sensor and Actuator Networks, **13**, *5*, pp. 62 (2024).
3. Z. Liu, L. Meng, Q. Zhao, F. Li, M. Song, Y. Jian, and H. Tian, *Authenticated Key Agreement Scheme Based on Blockchain for AMI Communication Security*, Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg., **68**, *2*, pp. 218–223 (2023).
4. Z.N.P.K. Lang, *Blockchains in Public Administration: A RADIUS on Blockchain Framework for Public Administration*, Thesis, Goethe-Universität Frankfurt am Main (2024).
5. E.F. Hoffmann, C. Machado, and C.M. Westphall, *RadChain Connect: Integration Between Blockchain and FreeRADIUS for Secure Authentication in Wi-Fi Networks/Web Environment*, In Lecture Notes in Computer Science, 2025 International Conference on Information Networking (ICOIN), pp. 431–436 (2025).
6. H. Hu, J. Yu, Z. Lin, H. Wu, and C. Yang, *BlockLoader: A Comprehensive Evaluation Framework for Blockchain Performance Under Various Workload Patterns*, Mathematics, **12**, *21*, pp. 3403 (2024).
7. P. Oikonomou et al., *Prototyping a Hyperledger Fabric-based Security Architecture for IoMT-based Health Monitoring Systems*, Future Internet, **15**, *9*, pp. 308 (2023).
8. A. Wilczyński and G. Jasnosz, *Security Assessment of Smart Contract Integration and Wallet Interaction in Decentralized Applications: A Case Study of BlockScribe*, Appl. Sci., **15**, *15*, pp. 8473 (2025).
9. Y. Lai, J. Yang, M. Liu, Y. Li, and S. Li, *Web3: Exploring Decentralized Technologies and Applications for the Future of Empowerment and Ownership*, Blockchains, **1**, *2*, pp. 111–131 (2023).
10. Z. Zhang, Y. Liu, Y. Zhang, and J. Zhang, *A Secure and Efficient Authentication Scheme for Large-Scale IoT Networks*, Electronics, **13**, *18* (2024).
11. ***ManageEngine: *What is PKI? A Complete Guide to Public Key Infrastructure* (2025).
12. H. Li, X. Wang, and L. Zhang, *Enhancing Security and Flexibility in the Industrial Internet of Things*, Sensors, **24**, *3*, pp. 1035 (2025).
13. C. Liu, *HPCLS-BC: A Novel Blockchain Framework Using Heterogeneous Peer-Node and Cloud-Based Ledger Storage for Internet of Things Applications*, Future Generation Computer Systems, **150**, pp. 364–379 (2024).
14. G.C. Seriţan, B.A. Enache, I. Vilciu, S.D. Grigorescu, and V. Mladenov, *Comparison study of top development boards in the context of IoT*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **67**, *4*, pp. 483–486 (2022).
15. R.A. Crăciun, R.N. Pietraru, and M.A. Moisescu, *Internet of Things Platform Benchmark: An Artificial Intelligence Assessment*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **69**, *1*, pp. 97–102 (2024).
16. M. Gayathri Santhosh and T. Reshmi, *Enhancing PKI Security in Hyperledger Fabric with an Indigenous Certificate Authority*, In 2023 IEEE International Conference on Public Key Infrastructure and its Applications (PKIA), pp. 1–5 (2023).
17. N. Paliwal, *Setting up a WiFi Bridge Using Raspberry Pi 4* (2025).
18. A. Rahman, M.J. Islam, S.S. Band, G. Muhammad, K. Hasan, and P. Tiwari, *Towards a Blockchain-SDN-Based Secure Architecture for Cloud Computing in Smart Industrial IoT*, Digital Communications and Networks, **9**, *2*, pp. 411–421 (2023).
19. R. Alt and M. Gräser, *Distributed Ledger Technology*, Electronic Markets, **35**, *1*, pp. 53 (2025).
20. J. Sychowiec and Z. Zieliński, *A Collaborative Data Sharing Platform to Accelerate Translation of Biomedical Research*, Biomedicines, **12**, *9*, pp. 938 (2024).
21. M.S. Al Jasem, T. De Clark, and A.K. Shrestha, *Toward Decentralized Intelligence: A Systematic Literature Review of Blockchain-Enabled AI Systems*, Information, **16**, *9*, pp. 765 (2025).
22. M. Jarosz, K. Wrona, and Z. Zieliński, *Distributed Ledger-Based Authentication and Authorization of IoT Devices in Federated Environments*, Electronics, **13**, *19*, pp. 3932 (2024).
23. D. Penzes, Z. Vincze, and B. Varga, *A Performance Analysis of Security Protocols for Distributed Measurement Systems Based on Internet of Things with Constrained Hardware and Open Source Infrastructures*, Sensors, **24**, *9*, pp. 2781 (2024).
24. R.V. Brăcăcescu, *A proposal of digital identity management using blockchain*, Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg., **69**, *1*, pp. 85–90 (2024).
25. C.A. Iordache and C.V. Marian, *Project Management Expert System with Advanced Document Management for Public Institutions*, Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg., **69**, *2*, pp. 219–224 (2024).