Digital Communications and Networks 10 (2024) 989–1000

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

A novel hybrid authentication protocol utilizing lattice-based cryptography for IoT devices in fog networks

Kumar Sekhar Roy [a,*], Subhrajyoti Deb [b], Hemanta Kumar Kalita [c]

[a] Department of Computer Science and Engineering, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, 576104, Karnataka, India
[b] Department of Computer Science & Engineering, ICFAI University, Agartala, 799210, India
[c] Department of Computer Science & Engineering, Central Institute of Technology, Kokrajhar, 783370, India

ARTICLE INFO

Keywords:
Internet of things
Authentication
Post-quantum cryptography
Lattice-based cryptography
Cloud computing
Fog computing
Fail-safe

ABSTRACT

The Internet of Things (IoT) has taken the interconnected world by storm. Due to their immense applicability, IoT devices are being scaled at exponential proportions worldwide. But, very little focus has been given to securing such devices. As these devices are constrained in numerous aspects, it leaves network designers and administrators with no choice but to deploy them with minimal or no security at all. We have seen distributed denial-of-service attacks being raised using such devices during the infamous Mirai botnet attack in 2016. Therefore we propose a lightweight authentication protocol to provide proper access to such devices. We have considered several aspects while designing our authentication protocol, such as scalability, movement, user registration, device registration, etc. To define the architecture we used a three-layered model consisting of cloud, fog, and edge devices. We have also proposed several pre-existing cipher suites based on post-quantum cryptography for evaluation and usage. We also provide a fail-safe mechanism for a situation where an authenticating server might fail, and the deployed IoT devices can self-organize to keep providing services with no human intervention. We find that our protocol works the fastest when using ring learning with errors. We prove the safety of our authentication protocol using the automated validation of Internet security protocols and applications tool. In conclusion, we propose a safe, hybrid, and fast authentication protocol for authenticating IoT devices in a fog computing environment.

1. Introduction

Since integrating IoT in the home environment, industry, logistics, etc., lots of movable devices have come into action. Such devices may change their physical location consistently, and they may not have enough resources of their own to securely authenticate themselves or for users to verify the authenticity of such devices. For such devices, scalability should not be an issue; making time-sensitive decisions is also of importance, so that the transmission delay and propagation delay are as minimum as possible. Along with the forthcoming quantum computers, previously used popular cipher suites also cannot be used, and therefore new algorithms need to be evaluated that are not vulnerable to algorithms running on a quantum computer or traditional electronic computer. The security algorithms need to be lightweight in nature so that they are easily processed by constrained IoT devices. Through our paper, we propose a hybrid authentication protocol that addresses all these issues.

Fog computing is an extension of cloud computing towards the edge of a network. Introduced in 2012 by Bonomi et al., fog computing provides several advantages over a centralized cloud server [1]. Fog computing moves time-dependent computation towards the edge of the network, thus providing the faster response, better mobility, and greater scalability, reducing the computation overhead of a constrained IoT device. Fog servers are connected to a centralized cloud server on the back end for providing the further analysis, storage, and processing of data generated by IoT devices. The cloud server only provides its services when required by the device or fog server, the mostly but not limited to less time-sensitive data. Over the years, several cloud-fog-edge architectures have been proposed; most commonly used models include three-layered and four-layered architectures. For the purpose of our research, we use a three-layered architecture, as shown in the following section.

Post-Quantum Cryptography (PQC) is a rising issue in the field of cryptography. After the recent developments in the manufacturing of

* Corresponding author.
E-mail address: shekhar699@gmail.com (K.S. Roy).

https://doi.org/10.1016/j.dcan.2022.12.003
Received 18 February 2021; Received in revised form 21 September 2022; Accepted 2 December 2022
Available online 16 December 2022

quantum computers, several public key cryptographies based on integer factorization and the discrete logarithmic problem have been proven to be vulnerable to Shor's algorithm running on a sufficiently large quantum computer. Thus several organizations and communities have suggested several public-key cryptography based algorithms that are not vulnerable to algorithms running on a quantum computer or traditional electronic computer. One such cryptography family is lattice-based cryptography. Proposed by Miklos Ajtai in 1996 [2], lattice-based cryptography has been analyzed by several researchers over the years [3,4]. The two important members of this family of cryptography are Ring-Learning With Errors (Ring-LWE) [3] and NTRU [5]. Another such family of cryptography is code-based cryptography, firstly introduced by Robert McEliece in 1978 [6]. Code-based cryptography relies on error-correcting codes for providing secrecy. We focus our protocol on lattice-based cryptography for our research.

Purpose, motivation, and contribution

IoT networks will comprise billions of heterogeneous devices, and their interconnection and interoperability pose a big challenge to the proper functioning of these networks. Neither centralized nor distributed architecture can suffice the security requirements related to these networks. Also, IoT devices have several constraints regarding memory, computation, power consumption, etc. Availability of authentication servers also poses a big challenge as there can be insecurities in the network. As most users are not experts, mistakes will happen if the security mechanisms are not useable enough. Misconfigurations can be exploited by malicious adversaries to access personal data or even take control of that particular user-centric network. Therefore, to ensure that the authentication of entities can function in any crisis scenario, we propose an authentication protocol for such a hybrid model, which is lightweight in nature such that it consumes the minimal resources of a device and is easily configurable from a user point of view.

We organize our paper as follows. Section 2 addresses related work. In Section 3, we provide the preliminaries required to understand the cryptography involved in our protocol. In Section 4, we describe our system model and present our authentication protocol. In Section 5 we analyze our protocol and show results. Finally, Section 6 concludes the paper.

2. Literature survey

The study of authentication for constrained devices has been found in the literature. Several researchers have proposed feasible authentication protocols over the years. In Ref. [21], the authors proposed a two-way authentication scheme for IoT based on Datagram Transport Layer Security (DTLS) handshake and X.509 certificate [22] consisting of Rivest-Shamir-Adleman (RSA) keys [23]. They utilized 2048-bit RSA keys for sensor nodes equipped with Trusted Platform Module (TPM) chips, which might not always be possible. The computation overhead for such a large key set would be very high. Also, the RSA algorithm is based on the integer factorization problem, which is proven to be vulnerable to Shor's Algorithm running on a sufficiently large quantum computer [24].

The authors in Refs. [19,25] separately evaluated the feasibility of RSA and Elliptic Curve Cryptography (ECC) [26] for the fog. In Ref. [19], the authors analyzed the applications of ECC and RSA in a publish-subscribe network in a fog/IoT communication model. They found ECC to provide less overhead in terms of storage and better scalability than RSA when utilized in Secure Socket Layer (SSL)/Transport Layer Security (TLS) while guaranteeing the same level of security. On the other hand, the authors in Ref. [25] performed a detailed analysis and comparison of RSA and ECC on several hardware platforms. They found similar results to those in Ref. [19]. ECC proved to be better than RSA based on communication throughput as well as power efficiency. All the above references study and utilize public-key cryptography based on either ECC or RSA, which are dependent on the elliptic curve discrete logarithm problem and integer factorization problem, respectively. These problems can be solved on a sufficiently large quantum computer running Shor's algorithm [24].

PQC has not been extensively used by researchers for authentication as of the present day. Although, a handful of researchers have explored the applications of post-quantum cryptography in the realm of constrained devices [27–30]. Xu et al. recently demonstrated the use of lattice-based cryptography for constrained devices [28]. They demonstrated the exhaustive implementation of NTRUEncrypt [31], R-LWEenc [32], R-BIN-LWEenc [33], IBE [34], BLISS [32], and NewHope [35] on several microcontrollers. Chen et al. recently proposed a handover authentication scheme based on NTRU for wireless networks [29]. Several researchers have proposed different versions of code-based cryptography, but any literature on the implementation of code-based cryptography for constrained devices can hardly be found. In Ref. [36], the authors mentioned an authentication scheme based on quantum key distribution and ECC in the IoT infrastructure. They proposed a scheme using quantum cryptography for data exchange in the registration phase. Nevertheless they failed to mention the precise cryptography suite in the PQC realm used in this phase, neither did they suggest any. For the purpose of authentication, they utilized ECC, which has been proven vulnerable. In our paper, we focus our attention on lattice-based cryptography algorithms, thus providing secure authentication to fog/IoT infrastructure in the near future. A detailed survey of the related work is provided in Table 1.

3. Preliminaries

In this section, we provide a detailed description of the cryptography suites utilized in our research. Quantum computers were thought to be theoretical until 2015 when NASA, along with D-Wave demonstrated a fully functional quantum computer. The concept of quantum computers was firstly initiated by scientists such as Richard Feynman, Paul Benioff, and Yuri Manin. Quantum computers differ from classical electronic computers in that classical computers use bits, which can have two definitive states i.e., either 0 or 1; on another hand, quantum computers utilize qubits, which can be in a superposition of states. A quantum

Table 1
Related work.

Scheme	Architecture	Cipher Suite	Disadvantage
[7]	GWN based multiserver	Hash and XOR	Stolen verifier attack, Forgery attack, Replay attack
[8]	Multiserver	Hash and XOR	Offline password guessing attack, User impersonation attack, Mutual authentication
[9]	Multiserver	Hash and XOR	Offline password guessing attack, User impersonation attack, Mutual authentication, Replay attack
[10]	Multiserver	Hash and XOR	User impersonation attack
[11]	Multiserver	Hash and XOR	Replay attack
[12]	Cloud	Hash and XOR	User impersonation attack
[13]	GWN based multiserver	Two factor authentication	GWN bypassing attack, privileged-insider attack, Mutual authentication
[14]	GWN based multiserver	Two factor authentication	Stolen smart card attack, privileged-insider attack, Forgery attack
[15]	GWN based multiserver	Two factor authentication	Smart card attack, sensor node impersonation attack, Man-in-the-middle attack
[16]	Cloud	Elliptic curve cryptography	Time-dependent decision making, Shor's algorithm
[17]	Cloud	Elliptic curve cryptography	Time-dependent decision making, Shor's algorithm
[18]	Cloud	Elliptic curve cryptography	Time-dependent decision making, Shor's algorithm
[19]	Fog	Elliptic curve cryptography	Time-dependent decision making, Shor's algorithm
[20]	Fog	Lattice-based cryptography	–

computer can be in 2^k states simultaneously using k qubits. A sufficiently large quantum computer running Shor's algorithm [24] can break the encryption based on the integer factorization problem, discrete logarithm problem, and elliptic curve discrete logarithm problem. Thus several researchers and organizations proposed different cryptography suites under the name of PQC, that could replace RSA and ECC in the near future. Table 2 shows different cryptography suites suggested by different researchers in the PQC era [37].

Lattice-based cryptography resists attacks based on quantum computers. Lattice-based cryptography is found to be faster and more efficient in terms of execution time. Also, lattice-based cryptography, unlike code-based cryptography, does not suffer from large key sizes. The principles involved in both these families of public-key cryptography are mentioned in the following section.

In 1996 and 1998 both Ajtai and Hoffstein et al. proposed the cryptographic constructions based on lattices [2,5]. Although mathematicians such as Joseph Louis Lagrange and Carl Friedrich Gauss studied mathematics related to lattices a long time back, Ajtai showed in his result that a Short Integer Solution (SIS), which is an average case lattice problem, is at least as hard to solve as a worst-case lattice problem. Whereas, Hoffstein et al. proposed NTRU, a public key encryption scheme. The mathematical structure of a lattice can be defined as follows.

A lattice L is a discrete subgroup of R^y, where R^y is a y-dimensional Euclidean space. More precisely, L can be depicted as a set of points in R^y. Any element of L is uniquely represented as a linear grouping of basis vectors b_1, b_2, ……b_y with real integer coefficients.

$$L = \left\{ \sum_{i=1}^{y} d_i b_i : d_i \in \mathbb{Z} \right\} \tag{1}$$

(1) NTRU

Developed by mathematicians Jeffrey Hoffstein (de), Jill Pipher, and Joseph H. Silverman in 1996 [5], NTRU has been studied by several researchers since then. Its security is based on the difficulty of analyzing the result of polynomial arithmetic modulo two unrelated moduli, and its correctness is based on the clustering properties of the sums of random variables. The authors in Ref. [38] performed an attack based on lattice reduction. In later years several researchers performed several attacks successfully on NTRU [39,40]. In 2016 Daniel J. Bernstein proposed NTRU Prime based on a strong algebraic structure that could fend against previously known attacks on NTRU [41]. A detailed description is as follows.

NTRU Prime is parametrized by (p, q, t), where $p \geq max\{2t, 3\}$; $q \geq 48t+1$; $x^p - x - 1$ is irreducible in polynomial ring $(/q)[x]$. Here

$$R = [x]/(x^p - x - 1) \tag{2}$$

$$R/3 = (/3)[x]/(x^p - x - 1) \tag{3}$$

$$R/q = (/q)[x]/(x^p - x - 1) \tag{4}$$

An element of R is referred to as small if all its coefficients are in $\{-1, 0, 1\}$.

The algorithms involved in NTRU Prime are as shown in algorithms 1, 2, and 3 respectively.

Algorithm 1: Key Generation in NTRU

Input: Input to this function are: R and q
//This function describes the Key Generation using NTRU
Key Generation
begin
  Generate a uniform random small element $g \in R$. Repeat this step until g is invertible in R=3
  Generate a uniform random t-small element $f \in R$. (Note that f is nonzero and hence invertible in R/q, since $t \geq 1$.)
  Compute $h = g/(3f)$ in R/q
  Encode h as a string h'. The public key is h'
  Save the following secrets: f in R; and $1/g$ in $R/3$
end

Algorithm 2: Encryption in NTRU

Input: Input to this function are: h', R and q
//This function describes encryption using NTRU
encryption
begin
  Decode the public key h', obtaining $h \in R=q$
  Generate a uniform random t-small element $r \in R$
  Compute $hr \in R=q$
  Round each coefficient of hr, viewed as an integer between -(q - 1)/2 and (q - 1)/2, to the nearest multiple of 3, producing $c \in R$
  Encode c as a string c'
  Hash r, obtaining a left half C ("key confirmation") and a right half K
  The cipher-text is the concatenation Cc'. The session key is K
end

Algorithm 3: Decryption in NTRU

Input: Input to this function is: c'
//This function describes decryption using NTRU
decryption
begin
  Decode c', obtaining $c \in R$
  Multiply by $3f$ in R/q
  View each coefficient of $3fc$ in R/q as an integer between -(q - 1)=2 and (q - 1)/2, and then reduce modulo 3, obtaining a polynomial e in $R/3$
  Multiply by $1/g$ in $R/3$
  Lift e/g in $R/3$ to a small polynomial $r' \in R$
  Compute c', C', K' from r' as in encryption
  If r' is t-small, $c' = c$, and $C' = C$, then output K'
  Otherwise, output False
end
// If Cc' is a legitimate cipher-text then c is obtained by rounding the coefficients of hr to the nearest multiples of 3, i.e., $c = m + hr$ in $R=q$, where $m¢$ is small.

Table 2
Post-quantum cryptography.

Sl no.	Family	Algorithm
1	Lattice-based	NTRU Ring LWE BLISS
2	Multivariate	Rainbow
3	Hash-based	Lamport Signature Merkle Signature
4	Code-based	McEllice Niederreiter

(2) Ring learning with errors

In [3], the authors proposed Ring-LWE based on Regev's learning with errors assumption over polynomial rings [42]. LWE entails distinguishing random linear equations, unsettled by a small amount of noise, from truly uniform ones. Lyubashevsky et al. showed that the hardness of Ring-LWE can be reduced to the worst-case hardness of the Short Vector Problem

(SVP). Ring-LWE has been utilized by several researchers for key exchange, homomorphic encryption, digital signature, public key encryption, etc. In simple words, Ring-LWE can be explained as, given polynomially many samples over a particular ring of form (a_i+e_i), where a_i's are uniformly random, e_i is a small ring element, and a third party cannot differentiate among these sequence of samples from random pairs of ring elements. For our research, we utilize an efficient implementation of Ring-LWE found in Ref. [43]. The Ring-LWE problem works in the following manner.

Polynomials o and v are chosen from $R_q=_q[x]/(f)$ where f is an irreducible polynomial of degree $n-1$. Then e of degree n is sampled from χ, and the error distribution is Gaussian distribution χ_σ. Distribution $A_{v,\chi}$, over $R_q \times R_q$ comprises tuples (a, t), here $t = a.v + e \in R_q$. Therefore, given a polynomial number of sample pair ($a.t$) from $A_{v,\chi}$, it is very hard to find v. The algorithm for Ring-LWE encryption is as follows.

The algorithms involved in the Ring-LWE cryptosystem are shown in algorithms 4, 5, and 6.

Algorithm 4: Key Generation in Ring-LWE

Input: Input to this function are: r_1 and r_2
//This function describes the Key Generation using Ring-LWE
Key Generation
begin
    Two polynomials r_1 and r_2 are sampled from χ_σ using discrete Gaussian sampler
    $NTT(r_1) \rightarrow \bar{r}_1$
    $NTT(r_2) \rightarrow \bar{r}_2$
    $\bar{p}_1 \leftarrow \bar{r}_1 - \bar{a} * \bar{r}_2$
end

Algorithm 5: Encryption in Ring-LWE

Input: Input to this function are: m, $\bar{a}$, and $\bar{p}_1$
//This function describes encryption using Ring-LWE
encryption
begin
    Message m is encoded to a polynomial $\bar{m} \in \bar{R}_q$
    Using a discrete Gaussian sampler $e_1, e_2, e_3 \in \bar{R}_q$ are generated from χ_σ
    $NTT(e_1) \rightarrow \bar{e}_1$
    $NTT(e_2) \rightarrow \bar{e}_2$
    $(\bar{c}_1, \bar{c}_2) \leftarrow (\bar{a} * \bar{e}_1 + \bar{e}_2; \bar{p}_1 * \bar{e}_1 + NTT(e_3 + \bar{m}))$
end

Algorithm 6: Decryption in Ring-LWE

Input: Input to this function is: $\bar{m}$
//This function describes decryption using Ring-LWE
decryption
begin
    Performing inverse NTT on $\bar{m}$
    $\bar{m} = INTT(\bar{c}_1 * \bar{r}_2 + \bar{c}_2)$
    Decode $\bar{m}$ to m
end

4. Proposed protocol

4.1. System model

Our network model consists of a three-layered fog architecture, as shown in Fig. 1. According to this model, a cloud server exists at the top of the hierarchy, acting as the central server. As shown in Fig. 1, the fog server acts as a middleware between the edge devices and a central cloud server. The edge devices being connected to the fog server are consequently connected to the cloud server. The distributed fog servers are connected to this centralized cloud server for data analysis, processing, storage, etc. According to our model, the cloud server is periodically updated about the happenings in the fog network. Any decision that is time-independent is also forwarded to the cloud by the fog server. The fog servers, being close to the edge devices, analyze and process the data that are time-dependent and make a decision. The authentication of such devices can be performed by the fog server, which would reduce the overhead on the cloud server. Also, it would provide better scalability, mobility, and heterogeneity and reduce latency. Thus providing authentication to the edge devices through the means of fog servers would prove beneficial for the whole network. We take into account a scenario where a fog user/device moves from one fog to another. If such a situation arises, our model will provide authentication to the migrating device as well. The entire scenario is mentioned in Fig. 2. All the symbols and notations used with regard to our protocol are provided in Table 3.

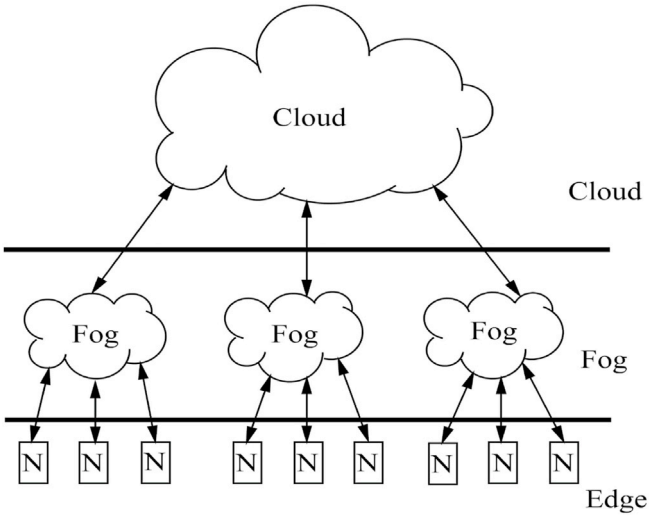


Fig. 1. Network model.

In our survey, we could not find any research that claims to provide authentication for devices moving within the fog (intra-fog) or among different fog (inter-fog); also no other protocol includes a fail-safe mechanism that includes a procedure to be followed after an authenticating fog server has failed. Although there is some research that uses inter-fog communications, it is for computation purposes only [44]. A

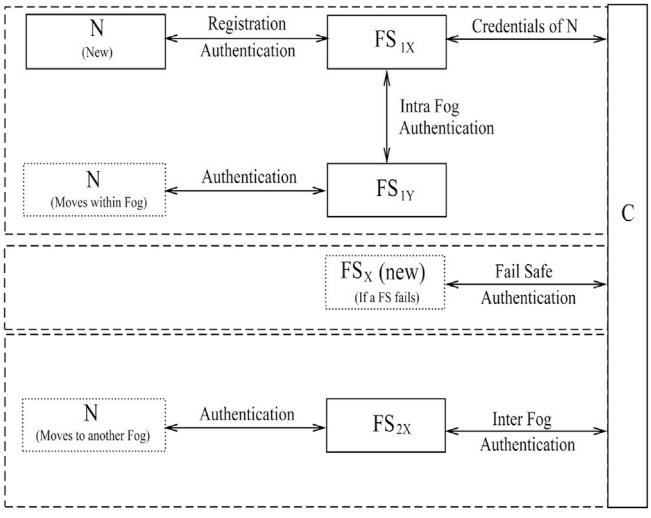


Fig. 2. A Holistic View of our Protocol.

Table 3
Notations.

Symbol	Representation
N	Node
FS	Fog server
$nonce$	Pseudo random number
C	Cloud server
C_{id}	Identity of cloud servers
N_{id}	Identity of fog user/device
FS_{id}	Identity of fog server
ACK	Acknowledgment (nonce+1)
ka	N's public key
Prv_N	N's private key
kb	FS's public key
kd	Another FS's public key
Prv_{FS}	FS's private key
pwd_N	N's password
kc	C's public key
pwd_{FS}	FS's password
$\mathbb{Z}$	Set of all integers
y	Integer
R	Euclidean space
p	Prime number
q	Prime number
t	A small element in R
g	Random small element in R
j	Random secret ring element.
NTT	Number Theoretic Transform
o	Polynomial
v	Polynomial
TS	Time stamp
SS	Shared Secret
R	Polynomial ring
e	Error polynomial
χ	Error distribution
σ	Standard deviation
r_2	Ring-LWE private key

holistic view of our proposed authentication protocol is given in Fig. 2. As mentioned in the figure, a new device N registers itself on the fog server FS_{1X}, which in turn shares the credentials to the cloud server so that the credentials can be used globally for authentication. If device N moves from the jurisdiction of FS_{1X} to a nearby fog server FS_{1Y} under the same distributed fog, the user/device would not have to register again as we include the intra-fog authentication phase. We include an inter-fog authentication phase as well, where if a device N moves from the jurisdiction of FS_{1X} to another fog server FS_{2X} in another fog, N would not have to register itself again for authentication. The inter-fog authentication is completed with the help of the cloud server. We also include a scenario where a fog server might fail. In such a case, a clustering algorithm is implemented where a cluster head will be chosen to act as a fog server FS_X temporarily and provide authentication services. Our hybrid authentication protocol is as follows.

Our proposed protocol consists of six phases. In the first phase, the user registers himself in the fog server. In the context of our protocol, a user implies any IoT device or the user personal device. The fog network comprises one or more such fog servers. Also, the fog servers have to be authenticated with the cloud server; therefore, they are registered with the cloud server, as shown in the second phase. The user-fog authentication is shown in the third phase, whereas the cloud-fog authentication is shown in the fourth phase. In the fifth and sixth phases, we provide the mechanism for intra-fog and inter-fog authentication, respectively. For the purpose of our protocol to work, we assume that at the initial state, both the user and fog server have a public and private key pair based on the lattice-based cryptography suite. We also provide a fail-safe phase which acts as an emergency response to an attack.

4.2. Node registration

This phase marks the registration of an IoT node N on a fog server FS. The precise details of this phase are provided in algorithm 7. Node N initiates this phase by sending an encrypted message $\{TS, FS_{id}, N_{id}, nonce\}\_kb$ to the fog-based authentication server. When the fog-based authentication server receives a registration request from a node, it verifies the node credentials and records the nonce along with the time stamp. Once the fog-based authentication server has verified the node credentials, it sends an acknowledgment message in the form of $\{ACK, FS_{id}, N_{id}\}\_ka$ to the node in its encrypted form. The node, after receiving the acknowledgment message, decrypts the message and verifies ACK. The node then sends its password to be linked with its identity, N_{id}, in the form of $\{TS, pwd_N, nonce\}\_kb$ to the fog-based authentication server. The mechanism of the Node registration phase is illustrated in algorithm 7.

Algorithm 7: Node registration

Input: Input to this function are: N_{id}, FS_{id}, pwd_N, ka, kb as described in section 4.1
Assumption: It is assumed that both parties know each other's public key
This algorithm describes the registration of a Node (N) on the fog server (FS)
begin
  Transition 1 //role played by N
  begin
    1: N encrypts $\{N_{id}, FS_{id}, TS, nonce\}$ with kb
    2: N sends this registration request to FS
  end
  Transition 2 //role played by FS
  begin
    1: Receive $\{N_{id}, FS_{id}, TS, nonce\}\_kb$ from N
    2: FS decrypts this message to extract N's identity
    3: FS verifies N_{id}, records nonce
    4: If verified, FS encrypts $\{N_{id}, FS_{id}, ACK\}$ by ka
    5: FS sends this acknowledgment message to N
  end
  Transition 3 //role played by N
  begin
    1: N decrypts $\{N_{id}, FS_{id}, ACK\}\_ka$ to find the acknowledgment
    2: N verifies the acknowledgment
    3: N encrypts $\{TS, pwd_N, nonce\}$ by kb
    4: N sends this message to FS to get itself registered
  end
  Transition 4 //role played by FS
  begin
    1: FS receives $\{TS, pwd_N, nonce\}\_kb$ from N
    2: FS decrypts it and verifies the nonce
    3: Stores pwd_N, to be used for authentication of N later on
  end
end

4.3. Fog server registration

This phase entails the registration of a fog server to the cloud-based authentication server, as mentioned in algorithm 8. The fog server starts the communications by sending $\{TS, FS_{id}, C_{id}, nonce\}\_kc$ to the cloud-based authentication server. The cloud-based authentication server, after receiving the message, decrypts the message using the

private key, verifies FS_{id}, and records the nonce. Thereafter, the cloud-based authentication server sends {*ACK*, FS_{id}, C_{id}}_*kb* as an acknowledgment message to the fog server. The fog server, after verifying the acknowledgment message, sends back {pwd_{FS}, *nonce*}_*kc* to the cloud server. The cloud server saves password pwd_{FS} to be linked with FS_{id} for authentication in later phases. The process of fog server registration is illustrated in algorithm 8.

Algorithm 8: Server registration

Input: Input to this function are: FS_{id}, C_{id}, pwd_{FS}, kb, kc as described in section 4.1
Assumption: It is assumed that both parties know each other's public key
This algorithm describes the registration of a fog server (FS) on the cloud-based authentication server (C)
begin
  Transition 1 //role played by FS
  begin
    1: FS encrypts $\{FS_{id}, C_{id}, TS, nonce\}$ with kc
    2: FS sends this registration request to C
  end
  Transition 2 //role played by C
  begin
    1: Receive $\{FS_{id}, C_{id}, TS, nonce\}\_kc$ from FS
    2: C decrypts this message to extract FS's identity
    3: C verifies FS_{id}, records nonce
    4: If verified, C encrypts $\{FS_{id}, C_{id}, ACK\}$ by kb
    5: C sends this acknowledgment message to FS
  end
  Transition 3 //role played by FS
  begin
    1: FS decrypts $\{FS_{id}, C_{id}, ACK\}\_kb$ to find the acknowledgment
    2: FS verifies the acknowledgment
    3: FS encrypts $\{TS, pwd_{FS}, nonce\}$ by kc
    4: FS sends this message to C to get itself registered
  end
  Transition 4 //role played by C
  begin
    1: C receives $\{TS, pwd_{FS}, nonce\}\_kc$ from FS
    2: C decrypts it and verifies the nonce
    3: Stores pwd_{FS}, to be used for authentication of FS later on
  end
end

4.4. Node authentication

In this phase, the actual authentication occurs between a node and a fog-based authentication server, as shown in algorithm 9. The node sends an access request using {*TS*, N_{id}, pwd_N, *nonce*}_*kb* to the fog-based authentication server. The fog-based authentication server, after receiving the access request from the node, verifies the node's identification (N_{id}) associated with its password (pwd_N) and records the nonce. If the password matches with the identification, an acknowledgment message {*ACK*, FS_{id}, N_{id}}_*ka* is sent to the node. The node gets access to the fog server after this stage. The node authentication process is also illustrated in algorithm 9.

Algorithm 9: Node authentication

Input: Input to this function are: N_{id}, FS_{id}, pwd_N, ka, kb as described in section 4.1
Assumption: It is assumed that both parties know each other's public key
This algorithm describes the authentication of a Node (N) on the fog server (FS)
begin
  Transition 1 //role played by N
  begin
    1: N encrypts $\{N_{id}, pwd_N, TS, nonce\}$ with kb
    2: N sends this authentication request to FS
  end
  Transition 2 //role played by FS
  begin
    1: Receive $\{N_{id}, pwd_N, TS, nonce\}\_kb$ from N
    2: FS decrypts this message to extract N's identity and password
    3: FS verifies N_{id} and pwd_N, records nonce
    4: FS encrypts $\{N_{id}, FS_{id}, ACK, S\}$ by ka, if verified
    5: FS sends this acknowledgment message to N
    6: N is now authenticated on FS
  end
  Transition 3 //role played by N
  begin
    1: N decrypts $\{N_{id}, FS_{id}, ACK, SS\}\_ka$ to find the acknowledgment
    2: N verifies the acknowledgment and stores SS to be used in fail-safe
    3: N is now authenticated on FS
  end
end

4.5. Fog server authentication

This phase shows the authentication process between a fog server and a cloud-based authentication server as mentioned in algorithm 10. The cloud server acts as a centralized authentication entity for all the fog servers included in the cloud-fog environment. The fog server begins the authentication process by sending {*TS*, FS_{id}, pwd_{FS}, *nonce*} to the cloud-based authentication server. The cloud-based authentication server verifies the fog server identification and the password associated with it, with the one stored in its database. It verifies and stores the nonce as well. If the fog server identification and password sent by the fog server match with that stored in the cloud server's database, the cloud-based authentication server sends an acknowledgment message {*ACK*, FS_{id}, C_{id}}_*kb* to the fog server. The fog server verifies the acknowledgment and hence gets associated with the cloud server. The whole process of fog server authentication is also illustrated in algorithm 10.

Algorithm 10: Server authentication

Input: Input to this function are: FS_{id}, C_{id}, pwd_{FS}, kb, kc as described in section 4.1
Assumption: It is assumed that both parties know each other's public key
This algorithm describes the Authentication of a Fog server (FS) on a cloud-based authentication server (C)
begin
  Transition 1 //role played by FS
  begin
    1: FS encrypts $\{FS_{id}, pwd_{FS}, TS, nonce\}$ with kc
    2: FS sends this authentication request to C
  end
  Transition 2 //role played by C
  begin
    1: Receive $\{FS_{id}, pwd_{FS}, TS, nonce\}\_kc$ from FS
    2: C decrypts this message to extract FS's identity and password
    3: C verifies FS_{id} and pwd_{FS}, records nonce
    4: If verified, C encrypts $\{FS_{id}, C_{id}, ACK, SS\}$ by kb
    5: C sends this acknowledgment message to FS
    6: FS is now authenticated on C
  end
  Transition 3 //role played by FS
  begin
    1: FS receives $\{FS_{id}, C_{id}, ACK, SS\}\_kb$ from C
    2: FS decrypts $\{FS_{id}, C_{id}, ACK, SS\}\_kb$ to find the acknowledgment
    3: FS verifies the acknowledgment and stores SS
    4: FS is now authenticated on C
  end
end

4.6. Intra-fog authentication

In this phase, we propose a mechanism assuming a scenario where a node moves from the jurisdiction of one fog server (FS_1) to another fog server (FS_2) in the same fog network. The benefit of this phase is that a node does not have to register itself again when moving under a different fog server and can be easily authenticated as mentioned in algorithm 11. The node initiates the authentication process by sending $\{TS, N_{id}, pwd_N, nonce, FS_1\}\_kb$ to FS_2. FS_2, after receiving the authentication request, simply forwards this message to FS_1. Now, FS_1, after receiving the message from FS_2, verifies node N's credentials with the ones stored in its database and records the nonce. If the verification is successful, FS_1 sends an acknowledgment message $\{ACK, N_{id}, pwd_N\}\_kd$, containing the node's credentials to FS_2. In turn, FS_2 sends an acknowledgment message $\{ACK, FS_2, N_{id}\}\_ka$ to N, hence providing access to itself. FS_2 also stores N's credentials for future reference. The intra-fog authentication mechanism is also illustrated in algorithm 11.

Algorithm 11: Intra-fog authentication

Input: Input to this function are: N_{id} FS_1, FS_2, pwd_N, ka, kb, kd as described in section 4.1
Assumption: It is assumed that all the parties involved know each other's public key
This algorithm describes the Authentication of a Node (N) when it moves from one Fog server (FS_1) to another Fog server (FS_2) in the same Fog network
begin
  Transition 1 //role played by N
  begin
    1: N encrypts $\{N_{id}, pwd_N, TS, nonce, FS_1\}$ with kb
    2: N sends this authentication request to FS_2
  end
  Transition 2 //role played by FS_2
  begin
    1: Receive $\{N_{id}, pwd_N, TS, nonce, FS_1\}\_kb$ from N
    2: Forward this message to FS_1
  end
  Transition 3 //role played by FS_1
  begin
    1: FS_1 receives $\{N_{id}, pwd_N, TS, nonce, FS_1\}\_kb$ from FS_2
    2: FS_1 decrypts $\{N_{id}, pwd_N, TS, nonce, FS_1\}\_kb$ to find N_{id} and pwd_N
    3: FS_1 verifies the credentials
    4: FS_1 encrypts $\{ACK, N_{id}, pwd_N\}$ by kd
    5: FS_1 sends this acknowledgment to FS_2
  end
  Transition 4 //role played by FS_2
  begin
    1: FS_2 receives $\{ACK, N_{id}, pwd_N\}\_kd$ from FS_1
    2: FS_2 decrypts it and stores N_{id} and pwd_N
    3: FS_2 sends $\{ACK, N_{id}, FS_2\}\_ka$ to N
  end
  Transition 5 //role played by N
  begin
    1: N receives $\{ACK, N_{id}, FS_2\}\_ka$ from FS_2
    2: N verifies this acknowledgment message and is thus authenticated
  end
end

4.7. Inter-fog authentication

This phase encompasses a scenario where a node moves from one fog network to another fog network, operating under the same cloud server. As the node is already registered under a fog network as well as the cloud-based authentication server, it does not have to register itself again to gain access to a different fog server under the same cloud, as mentioned in algorithm 12. In this phase, we assume that all the parties involved in the authentication process know each other's public keys. A node N moves from the jurisdiction of FS_1 to that of FS_2, which is under a different fog network.

The authentication process starts when a node sends an authentication request using $\{TS, N_{id}, pwd_N, nonce, FS_1\}_{pub_c}$ to FS_2. FS_2, in turn, simply forwards this message to C. The cloud server maintains a database recording all the node credentials operating under different fog servers in its jurisdiction. The cloud-based authentication server then checks and verifies the node credentials. If the verification is successful, the cloud-based authentication server stores the nonce and sends back an acknowledgment message $\{ACK, N_{id}, pwd_N\}_{pub_{FS_2}}$ to FS_2. The fog-based authentication server FS_2, after storing node N's credentials, sends an acknowledgment message $\{ACK, FS_{id}, N_{id}\}_{pub_N}$, granting access to the node. The node verifies the acknowledgment message and then, can access the fog server. The complete mechanism utilized in this phase is illustrated in algorithm 12.

Algorithm 12: Inter-Fog Authentication

Input: Input to this function are: N_{id} C, FS_2, pwd_N, ka, kc, kd as described in section 4.1
Assumption: It is assumed that all the parties involved know each other's public key
This algorithm describes the Authentication of a Node (N) when it moves from one Fog server (FS_1) to another Fog server (FS_2) in a different Fog network but under the same cloud server (C)
begin
  Transition 1 //role played by N
  begin
    1: N encrypts $\{N_{id}, pwd_N, TS, nonce, FS_1\}$ with kc
    2: N sends this authentication request to FS_2
  end
  Transition 2 //role played by FS_2
  begin
    1: Receive $\{N_{id}, pwd_N, TS, nonce, FS_1\}\_kc$ from N
    2: Forward this message to C
  end
  Transition 3 //role played by C
  begin
    1: C receives $\{N_{id}, pwd_N, TS, nonce, FS_1\}\_kc$ from FS_2
    2: C decrypts $\{N_{id}, pwd_N, TS, nonce, FS_1\}\_kc$ to find N_{id} and pwd_N
    3: C verifies the credentials
    4: C encrypts $\{ACK, N_{id}, pwd_N\}$ by kd
    5: C sends this acknowledgment to FS_2
  end
  Transition 4 //role played by FS_2
  begin
    1: FS_2 receives $\{ACK, N_{id}, pwd_N\}\_kd$ from C
    2: FS_2 decrypts it and stores N_{id} and pwd_N
    3: FS_2 sends $\{ACK, N_{id}, FS_2\}\_ka$ to N
  end
  Transition 5 //role played by N
  begin
    1: N receives $\{ACK, N_{id}, FS_2\}\_ka$ from FS_2
    2: N verifies this acknowledgment message and is thus authenticated
  end
end

4.8. Fail-safe authentication

Our protocol also takes into account a fail-safe mechanism if and when a fog-based authentication server fails. Hypothetically, a fog server might fail for many different reasons, such as physical damage, succumbing to some unforeseen attacks and power failures. Therefore, we devise our authentication protocol to work and authenticate devices if such a scenario arises.When a fog server fails, all the devices connected to it are scattered randomly. The first task would be to reorganize them efficiently. Therefore, clustering algorithms should come into action. The design and development of routing protocols for constrained devices can be a challenging task. Clustering provides an elegant solution to such challenges by reducing the consumed energy as a consequence of reduced communications. Clustering algorithms mostly work by forming a cluster and selecting a cluster head or vice versa. We utilize the services of Hy-IoT, proposed in 2018 by Sadek [45]. As shown in Fig. 3 Hy-IoT is a hybrid energy-aware clustering protocol for IoT heterogeneous networks. It combines two different protocols together i.e., LEACH [46] and SEP [56].

Algorithm 13: Fail-Safe Authentication

Input: Input to this function are: N_{id} C, FS, SS, ka, kb, kc as described in section 4.1
Assumption: It is assumed that all the parties involved know each other's public key
This algorithm describes the Authentication of a Node (N) when a Fog server fails and clustering happens and a CH is elected to be the new Fog server (FS) under the cloud server (C)
begin
  Transition 1 //played by N
  begin
    1: N encrypts $\{N_{id}, SS, TS, nonce\}$ with kc
    2: N sends this authentication request to FS
  end
  Transition 2 //played by FS
  begin
    1: Receive $\{N_{id}, SS, TS, nonce\}\_kc$ from N
    2: Forward this message to C
  end
  Transition 3 //played by C
  begin
    1: C receives $\{N_{id}, SS, TS, nonce\}\_kc$ from FS
    2: C decrypts $\{N_{id}, SS, TS, nonce\}\_kc$ to find N_{id} and SS
    3: C verifies the credentials
    4: C encrypts $\{ACK, N_{id}, SS\}$ by kb
    5: C sends this acknowledgment to FS
  end
  Transition 4 //played by FS
  begin
    1: FS receives $\{ACK, N_{id}, SS\}\_kb$ from C
    2: FS decrypts it and stores N_{id} and SS
    3: FS sends $\{ACK, N_{id}, FS\}\_ka$ to N
  end
  Transition 5 //played by N
  begin
    1: N receives $\{ACK, N_{id}, FS\}\_ka$ from FS
    2: N verifies this acknowledgment message and is thus authenticated
  end
end

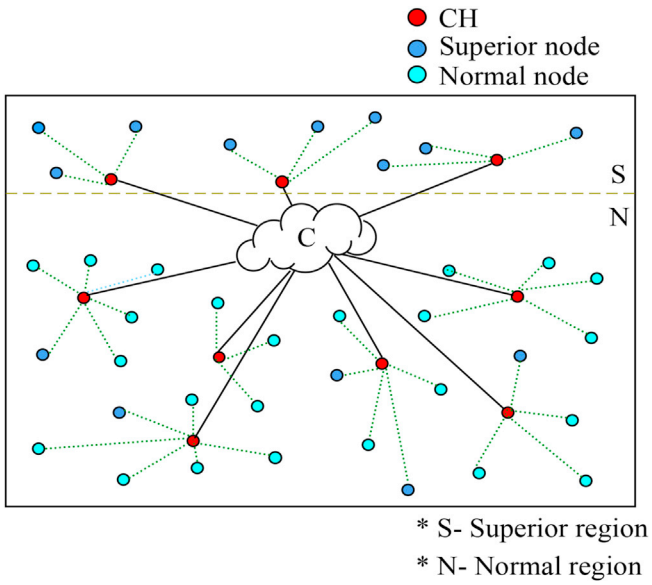


Fig. 3. Hy-IoT clustering architecture.

Table 4
Execution time.

Cryptosystem	Encryption (sec.)	Decryption (sec.)
NTRU Prime	0.008285	0.0011
Ring-LWE	0.000785	0.000597
Niederreiter	0.001007	0.018580
ECC	0.357316	0.182518
RSA1024	0.49276	11.197080
RSA (using small prime)	0.000026	0.018580

This phase marks the authentication of several randomly scattered devices with the cluster heads; these cluster heads act as the fog server. Since these constrained devices are low in resources, a new registration phase with the fog server would seem inefficient, therefore we utilize the shared secret for authentication. Sharing of previously used and permanent passwords would be ill-advised, as the new fog server would be temporary; therefore, the shared secret provides for the primary password. As mentioned in algorithm 13, the node sends its credentials comprising of its identity and a shared secret along with the time stamp and nonce to the fog server, encrypted by the cloud server's public key. The fog server, upon receiving this message, simply forwards the message to the cloud-based authentication server along with its own identity, password, time stamp, and nonce. The cloud-based authentication server, upon receiving this message, verifies the fog server's identity as well as the node identity from its database. Once it is verified, the cloud-based authentication server replies to the fog server with the shared secret and an acknowledgment, hence authenticating the node device trying to connect with the fog server. The fog server upon receiving the shared secret replies the node device with the shared secret and an acknowledgment, therefore establishing mutual authentication.

5. Result and analysis

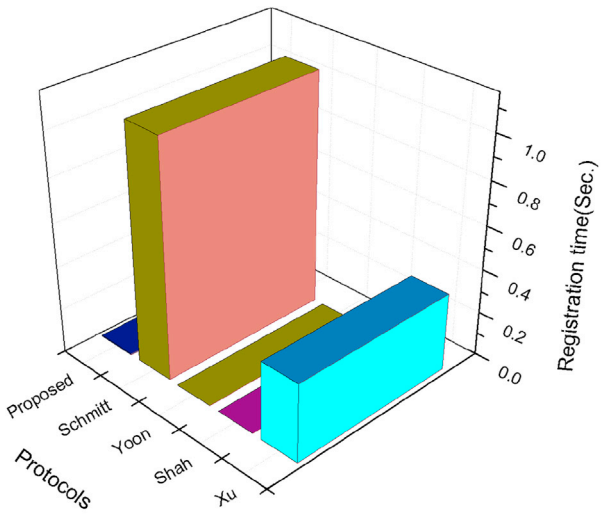
In this section, we show that our authentication protocol is secure against several attacks. As evident from the figures mentioned above in the previous sections, in most of the instances, we have used nonces and time stamps to avoid replay attacks. We also make sure that all the parties involved in the authentication process are mutually authenticated. As we have utilized the services of public key cryptography from the realm of PQC, it is evident that the cipher suites cannot be broken using traditional electronic computers or quantum computers. We propose lattice-based cryptography to be used in our protocol, i.e., NTRU and Ring-LWE. Based on the user's need, the user can use either of the public-key cryptography for encryption and decryption. As we can see from Table 4, Ring-LWE takes the least time for encryption and decryption; we suggest the usage of Ring-LWE. The values of the execution time vary based on the conditions chosen such as the curve chosen for ECC or the computational configuration of the machine. The prime focus here is the ratio by which the execution times differ and not the exact value. Evidently, there is a small difference in the execution time between lattice-based cryptography and code-based cryptography. Also, it has to be mentioned that code-based cryptography suffers from large key sizes, but little research work can be found which reduces the key size significantly [47]. We also suggest that a fog server authenticates a limited number of devices (based on its computational capabilities) at a time and does not take any more requests to avoid distributed denial of service attacks. The fog server will block an IP if it receives multiple authentication requests, thus avoiding resource exhaustion attacks. A comparison between several authentication protocols is mentioned in Table 5. It can be seen that several authentication protocols are secure against most of the known attacks except the attacks based on quantum computers. Our protocol stands out from the rest in that matter.

We verify our authentication protocol to be secure by using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool. We specify our protocol in AVISPA using High Level Protocol Specification Language (HLPSL) and verify their safety using On-the-Fly Model Checker (OFMC) and Constraint Logic-based ATtack SEarcher (CL-AtSe) modes. The OFMC mode combines two different ways for the analysis of Internet security protocols in a lazy and demand-driven way. It incorporates a lazy Dolev-Yao intruder model using symbolic techniques and optimizations, generating actions in a demand-driven way. It also incorporates the use of lazy data types for building an efficient on-the-fly model checker for protocols with very large state spaces. The CL-AtSe mode is an effective and efficient automatic analyzer designed for cryptographic protocols. It is a constrained logic based attack searcher that models all reachable states of the participants in the protocol in accordance with the Dolev-Yao intruder model. It can model any state-based security property such as fairness, authentication, and privacy. It can also model the algebraic properties of several operators such as the eXclusive OR (XOR) operation and exponentiation. It can also effectively analyze constraints such as inequalities and typing. Therefore, we prove the validity of our authentication protocol in these two modes.

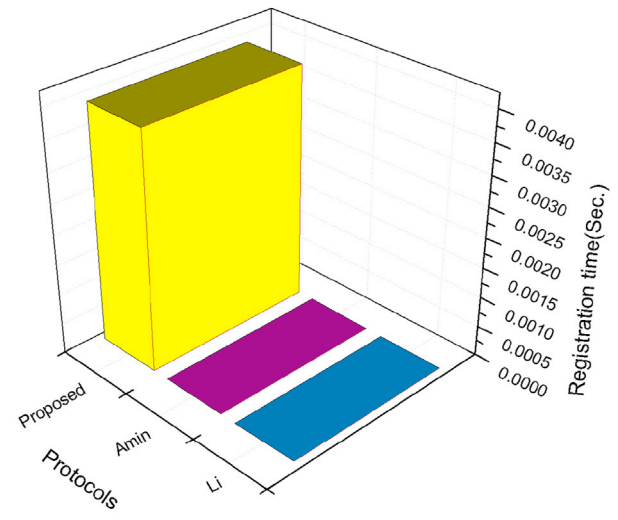
We also compare our protocol with several other protocols in the context of the computation time, as mentioned in Figs. 4a and 6b. Our protocol proves to be much faster than the protocols utilizing public-key cryptography such as Xu et al. [51], Yoon et al. [52], Shah et al. [53], and

Table 5
Attack comparison.

Attacks	[8]	[9]	[48]	[10]	[12]	[11]	[49]	[50]	Proposed
Offline password guessing	N	N	Y	Y	Y	Y	Y	Y	Y
User impersonation	N	N	Y	N	–	Y	–	Y	Y
Resist replay	Y	N	N	Y	–	N	–	Y	Y
Mutual Authentication	N	N	Y	Y	Y	N	Y	Y	Y
Man in the middle	–	–	–	–	Y	–	–	Y	Y
Quantum Computer	–	–	–	–	–	–	N	–	Y
Year	2014	2013	2013	2014	2016	2011	2015	2018	

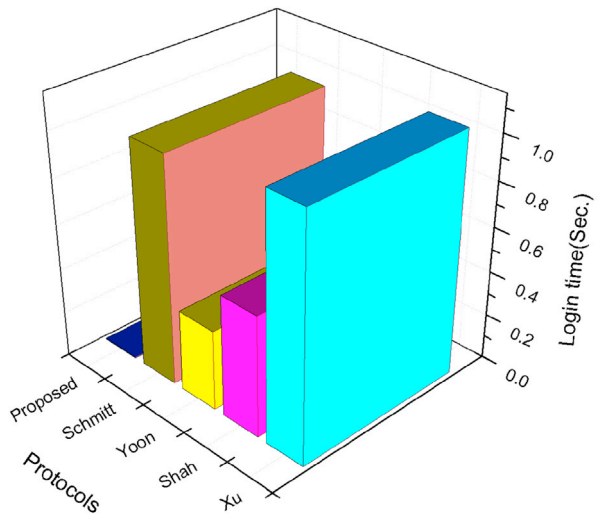


(a) With protocols using PKC

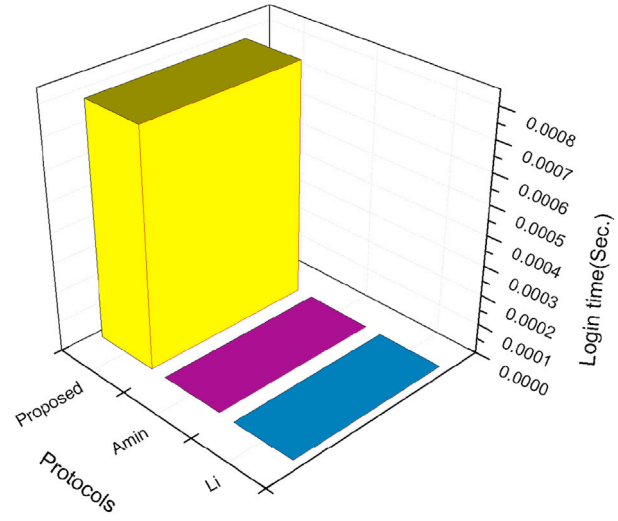


(b) With protocols using Hash algorithms

Fig. 4. Registration time comparison.



(a) With protocols using PKC



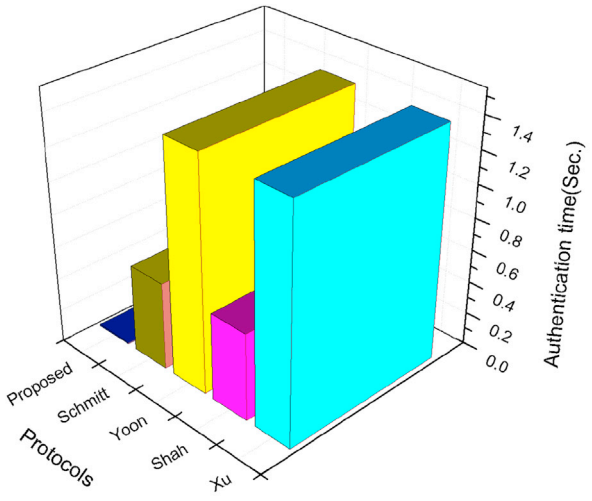
(b) With protocols using Hash algorithms

Fig. 5. Login time comparison.

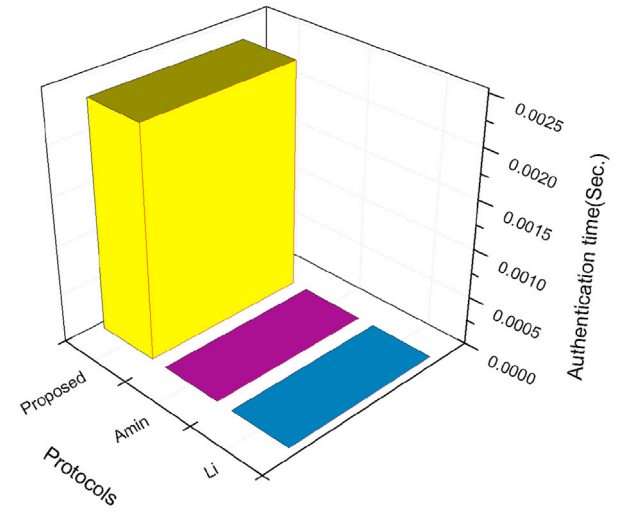
Schmitt [54], as shown in Figs. 4a, 5a and 6a. We can see from Figs. 4b, 5b and 6b that some of the authentication protocols utilizing hash-based authentication have less computation time than our protocol, almost approaching zero, such as Li et al. [55] and Amin et al. [50]. This trade-off between the computation cost and provable security against attacks based on quantum computers had to be made to prove our protocol secure against algorithms running on quantum computers as well as traditional electronic computers, which other protocols fail to provide. A comparison between the execution time taken by several public key cryptographies is also mentioned in Table 4. The execution time for encrypting and decrypting 128-bits is attained in the Linux environment using an Intel Core i5 (4210U, 1.7 GHz) and a 4 GB RAM. Although a recent report suggested that it is unlikely that a sufficiently large quantum computer would be developed in the next decade that could solve RSA-2048.[1] However, in the context of the IoT, RSA takes too long in terms of execution time and therefore cannot be considered for usage in the IoT environment. Therefore, even though RSA-2048 can be used for other networks, it is not suitable for IoT networks and hence raises the need for the applicability of PQC. An informal analysis of our authentication protocol is mentioned as follows.

1. **Offline password guessing:** We suggest that the user provide a strong password, to resist brute force password guessing attack that includes alphanumeric characters and punctuation marks.
2. **Resist replay:** A replay attack is when an attacker tries to impersonate a legitimate user by re-sending a previously sent message from the legitimate user that the attacker has previously captured. The authentication protocol should provide resistance against such an

[1] 'The U.S. National Academies Reports on the Prospects for Quantum Computing' (https://spectrum.ieee.org/tech-talk/computing/hardware/the-us-nationalacademies-reports-on-the -prospects -for-quantum-computing).



(a) With protocols using PKC



(b) With protocols using Hash algorithms

Fig. 6. Authentication time comparison.

attack. Therefore, we utilize nonces and time stamps in most instances to avoid such an attack.

3. **Mutual authentication:** It is the form of authentication where both the communicating parties participate in the process. Each party authenticates the other and makes sure that the other communicating party is legitimate. It is an important feature because an attacker can impersonate a legitimate user to gain access to a network with malicious intentions. And therefore, we make sure that all the parties are mutually authenticated.
4. **Privileged insider attack:** A privileged insider attack is when a user with enough privileges collects data about another user and tries to impersonate as that user in another network. The authentication protocol should resist such attacks. Therefore, we suggest that the password and user ID be stored in its encrypted form.
5. **Man in the middle attack:** It is a classical attack where an eavesdropper in the middle of two communicating parties acts as a legitimate user (other communicating party) to each user, i.e., acts as a sender to the receiver and receiver to the sender. A secure authentication protocol should resist such an attack. We verify the resistance of our protocol against this attack by using the AVISPA tool.
6. **Resource exhaustion attack:** It is when an adversary sends multiple requests to the server demanding the server requests. The adversary would try to send the same message again and again within quick succession. The server would check for the cipher-text with the previously received cipher-text within a few seconds. If found to be the same, the user's IP address will be blocked.

6. Conclusion

In this paper, we provide an authentication protocol for the IoT in a fog computing environment. Fog computing provides several advantages over cloud computing as fog servers are nearer to the edge devices and decrease scalability, latency, and communication costs. Therefore fog servers are ideal for IoT networks. PQC involves cryptosystems that are immune to the attacks based on quantum computers. Although quantum computers are not a threat to network security as of now, the same cannot be said for the recent future. We provide an authentication protocol with six different phases and prove each phase to be secure in OFMC mode and CL-AtSe mode using HLPSL in the AVISPA tool. While analyzing different public-key cryptosystems, including NTRU, Ring-LWE, Niedderieter, RSA-1024, and ECC, we find that NTRU, Ring-LWE, and Niedderieter outperform RSA and ECC in terms of execution time, Ring-LWE being the best among them. We also compare our authentication protocol for the IoT with others and find that our protocol is secure against most known attacks. Future works would include creating a fail-safe phase that could dynamically create a fog network if and when a fog server fails.

References

[1] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things characterization of fog computing, Proc. MCC, ACM (2016) 13–17.
[2] M. Ajtai, Generating hard instances of lattice problems, in: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, ACM, 1996, pp. 99–108.
[3] V. Lyubashevsky, C. Peikert, O. Regev, On ideal lattices and learning with errors over rings, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2010, pp. 1–23.
[4] C. Gentry, C. Peikert, V. Vaikuntanathan, Trapdoors for hard lattices and new cryptographic constructions, in: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, 2008, pp. 197–206.
[5] J. Hoffstein, J. Pipher, J.H. Silverman, Ntru: a ring-based public key cryptosystem, in: International Algorithmic Number Theory Symposium, Springer, 1998, pp. 267–288.
[6] R.J. McEliece, A public-key cryptosystem based on algebraic, Coding Thv 4244 (1978) 114–116.
[7] K.H. Wong, Y. Zheng, J. Cao, S. Wang, A dynamic user authentication scheme for wireless sensor networks, in: IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06), vol. 1, IEEE, 2006, pp. 8–16.
[8] K. Xue, P. Hong, C. Ma, A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture, J. Comput. Syst. Sci. 80 (1) (2014) 195–206.
[9] B. Wang, M. Ma, A smart card based efficient and secured multi-server authentication scheme, Wireless Pers. Commun. 68 (2) (2013) 361–378.
[10] M.-C. Chuang, M.C. Chen, An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics, Expert Syst. Appl. 41 (4) (2014) 1411–1418.
[11] S.K. Sood, A.K. Sarje, K. Singh, A secure dynamic identity based authentication protocol for multi-server architecture, J. Netw. Comput. Appl. 34 (2) (2011) 609–618.
[12] J.-J. Huang, W.-S. Juang, C.-I. Fan, Y.-F. Tseng, H. Kikuchi, Lightweight authentication scheme with dynamic group members in iot environments, in: Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services, 2016, pp. 88–93.
[13] M.L. Das, Two-factor user authentication in wireless sensor networks, IEEE Trans. Wireless Commun. 8 (3) (2009) 1086–1090.
[14] M.K. Khan, K. Alghathbar, Cryptanalysis and security improvements of 'two-factor user authentication in wireless sensor networks, Sensors 10 (3) (2010) 2450–2459.
[15] M. Turkanović, B. Brumen, M. Hölbl, A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion, Ad Hoc Netw. 20 (2014) 96–112.
[16] S. Kalra, S.K. Sood, Secure authentication scheme for iot and cloud servers, Pervasive Mob. Comput. 24 (2015) 210–223.
[17] P.K. Dhillon, S. Kalra, A secure multi-factor ecc based authentication scheme for cloud-iot based healthcare services, J. Ambient Intell. Smart Environ. 11 (2) (2019) 149–164.

[18] A. Tewari, B. Gupta, A lightweight mutual authentication protocol based on elliptic curve cryptography for iot devices, Int. J. Adv. Intell. Paradigms 9 (2–3) (2017) 111–121.
[19] A.A. Diro, N. Chilamkurti, N. Kumar, Lightweight cybersecurity schemes using elliptic curve cryptography in publish-subscribe fog computing, Mobile Network. Appl. 22 (5) (2017) 848–858.
[20] Z. Liu, K.-K.R. Choo, J. Grossschadl, Securing edge devices in the post-quantum internet of things using lattice-based cryptography, IEEE Commun. Mag. 56 (2) (2018) 158–162.
[21] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, G. Carle, Dtls based security and two-way authentication for the internet of things, Ad Hoc Netw. 11 (8) (2013) 2710–2723.
[22] R. Housley, W. Ford, W. Polk, D. Solo, x. Internet, 509 public key infrastructure certificate and crl profile, Tech. rep., RFC 2459 (January) (1999).
[23] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Commun. ACM 21 (2) (1978) 120–126.
[24] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35th Annual Symposium on Foundations of Computer Science, IEEE, 1994, pp. 124–134.
[25] M. Suárez-Albela, T.M. Fernández-Caramés, P. Fraga-Lamas, L. Castedo, A practical evaluation of a high-security energy-efficient gateway for iot fog computing applications, Sensors 17 (9) (2017) 1978.
[26] N. Koblitz, Elliptic curve cryptosystems, Math. Comput. 48 (177) (1987) 203–209.
[27] K.S. Roy, H.K. Kalita, A quantum safe user authentication protocol for the internet of things, Int. J.Next Generate. Comput. (2019) 178–192.
[28] R. Xu, C. Cheng, Y. Qin, T. Jiang, Lighting the Way to a Smart World: Lattice-Based Cryptography for Internet of Things, arXiv preprint arXiv:1805.04880.
[29] R. Chen, D. Peng, A novel ntru-based handover authentication scheme for wireless networks, IEEE Commun. Lett. 22 (3) (2017) 586–589.
[30] K.S. Roy, H.K. Kalita, A survey on post-quantum cryptography for constrained devices, Int. J. Appl. Eng. Res. 14 (11) (2019) 2608–2615.
[31] O.M. Guillen, T. Pöppelmann, J.M.B. Mera, E.F. Bongenaar, G. Sigl, J. Sepulveda, Towards post-quantum security for iot endpoints with ntru, in: Design, Automation & Test in Europe Conference & Exhibition (DATE) 2017, IEEE, 2017, pp. 698–703.
[32] Z. Liu, T. Pöppelmann, T. Oder, H. Seo, S.S. Roy, T. Güneysu, J. Großschädl, H. Kim, I. Verbauwhede, High-performance ideal lattice-based cryptography on 8-bit avr microcontrollers, ACM Trans. Embed. Comput. Syst. 16 (4) (2017) 1–24.
[33] J. Buchmann, F. Göpfert, T. Güneysu, T. Oder, T. Pöppelmann, High-performance and lightweight lattice-based public-key encryption, in: Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security, 2016, pp. 2–9.
[34] T. Güneysu, T. Oder, Towards lightweight identity-based encryption for the post-quantum-secure internet of things, in: 2017 18th International Symposium on Quality Electronic Design (ISQED), IEEE, 2017, pp. 319–324.
[35] E. Alkim, P. Jakubeit, P. Schwabe, Newhope on arm cortex-m, in: International Conference on Security, Privacy, and Applied Cryptography Engineering, Springer, 2016, pp. 332–349.
[36] A. Lohachab, et al., Using quantum key distribution and ecc for secure inter-device authentication and communication in iot infrastructure, in: Proceedings of 3rd International Conference on Internet of Things and Connected Technologies, ICIoTCT, 2018, pp. 26–27.
[37] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, D. Smith-Tone, Report on Post-quantum Cryptography, vol. 12, US Department of Commerce, National Institute of Standards and Technology, 2016.
[38] D. Coppersmith, A. Shamir, Lattice attacks on ntru, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1997, pp. 52–61.
[39] N. Howgrave-Graham, A hybrid lattice-reduction and meet-in-the-middle attack against ntru, in: Annual International Cryptology Conference, Springer, 2007, pp. 150–169.
[40] C. van Vredendaal, Reduced memory meet-in-the-middle attack against the ntru private key, LMS J. Comput. Math. 19 (A) (2016) 43–57.
[41] D.J. Bernstein, C. Chuengsatiansup, T. Lange, C.v. Vredendaal, Ntru prime: reducing attack surface at low cost, in: International Conference on Selected Areas in Cryptography, Springer, 2017, pp. 235–260.
[42] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, J. ACM 56 (6) (2009) 1–40.
[43] R. De Clercq, S.S. Roy, F. Vercauteren, I. Verbauwhede, Efficient software implementation of ring-lwe encryption, in: 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2015, pp. 339–344.
[44] M. Mukherjee, M. Guo, J. Lloret, R. Iqbal, Q. Zhang, Deadline-aware fair scheduling for offloaded tasks in fog computing with inter-fog dependency, IEEE Commun. Lett. 24 (2) (2019) 307–311.
[45] R.A. Sadek, Hybrid energy aware clustered protocol for iot heterogeneous network, Future Compute Informatic J. 3 (2) (2018) 166–177.
[46] S. Tyagi, N. Kumar, A systematic review on clustering and routing techniques based upon leach protocol for wireless sensor networks, J. Netw. Comput. Appl. 36 (2) (2013) 623–645.
[47] C. Wieschebrink, Cryptanalysis of the niederreiter public key scheme based on grs subcodes, in: International Workshop on Post-Quantum Cryptography, Springer, 2010, pp. 61–72.
[48] D. He, S. Wu, Security flaws in a smart card based authentication scheme for multi-server environment, Wireless Pers. Commun. 70 (1) (2013) 323–329.
[49] V. Shivraj, M. Rajan, M. Singh, P. Balamuralidhar, One time password authentication scheme based on elliptic curves for internet of things (iot), in: 2015 5th National Symposium on Information Technology: towards New Smart World (NSITNSW), IEEE, 2015, pp. 1–6.
[50] R. Amin, N. Kumar, G. Biswas, R. Iqbal, V. Chang, A light weight authentication protocol for iot-enabled devices in distributed cloud computing environment, Future Generat. Comput. Syst. 78 (2018) 1005–1019.
[51] J. Xu, W.-T. Zhu, D.-G. Feng, Improvement of a fingerprint-based remote user authentication scheme, in: 2008 International Conference on Information Security and Assurance (Isa 2008), IEEE, 2008, pp. 87–92.
[52] E.-J. Yoon, K.-Y. Yoo, Robust biometrics-based multi-server authentication with key agreement scheme for smart cards on elliptic curve cryptosystem, J. Supercomput. 63 (1) (2013) 235–255.
[53] T. Shah, S. Venkatesan, Authentication of iot device and iot server using secure vaults, in: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), IEEE, 2018, pp. 819–824.
[54] C. Schmitt, M. Noack, B. Stiller, Tinyto: two-way authentication for constrained devices in the internet of things, in: Internet of Things, Elsevier, 2016, pp. 239–258.
[55] C.-T. Li, M.-S. Hwang, An efficient biometrics-based remote user authentication scheme using smart cards, J. Netw. Comput. Appl. 33 (1) (2010) 1–5.
[56] Smaragdakis, Georgios, Ibrahim Matta, and Azer Bestavros. "SEP: A stable election protocol for clustered heterogeneous wireless sensor networks." In Second international workshop on sensor and actor network protocols and applications (SANPA 2004), (3)(2004) 1–10