Formal Verification of CHAP PPP authentication Protocol for Smart City/Safe City Applications.

Pradeep R$^1$ and Dr. N R Sunitha$^2$

$^1,^2$Department of Computer Science and Engineering, Siddaganga Institute Of Technology, Tumkur - Karnataka, India 572103

E-mail: pradeepr@sit.ac.in

Abstract. A smart city is a technologically advanced metropolitan region with several connected devices that collects data using various electronic technologies, voice activation methods, and sensors. The information obtained from the data is utilised to efficiently manage assets, resources, and services; in turn, the data is used to enhance operations throughout the city. Achieving security for smart cities is one of the major challenges as the number of connected devices increases the vulnerability also increases. The security of a smart city system depends on the reliability of the security protocols used by the security systems. To design and develop a highly secure system for a smart city the security protocols used must be highly reliable. To prove the reliability of a security protocol the validation technique is not desirable because of its several drawbacks, these drawbacks can be overcome using the formal verification technique which provides the mathematical proof for its correctness. In this work, The Challenge-Handshake Authentication Protocol Point-to-Point (CHAP PPP) which is more commonly used in PPP authentication of smart cities is formally verified using the well-known verification technique known as the model checking technique. The Scyther model checker is the tool used to build the abstract security protocol model.

1. Introduction

Smart cities [1] is seen as the way of the future for the growth of a country’s economy. This notion has grown in popularity over the last two decades, and many academics have offered several definitions to better comprehend smart cities. A smart city is defined as a city with a strong technical database for linking people and exchanging information, as well as an improved transportation system and intelligent buildings to improve people’s quality of life. The smart city may be built by combining the newest communication and sensor technologies with smart energy systems, sophisticated construction processes, and advanced research and development systems for building infrastructures. A smart city’s effective structure can keep track of its infrastructure, including highways, railroads, airports, health centers, subways, and entertainment centers forming the Internet of Things (IoT)[2] through the Internet. Then we can integrate the IoT through sensors and cloud computing.

Smart city security has recently emerged as a critical problem in the cyber security arena. Almost every industrialised country today has access to a gadget that can connect to the internet. This was not always the case in the past. As a result, smart city security[4] has become a critical necessity that was previously unneeded. It was something that didn’t exist for them. Since smart cities have exploded all over the world in some shape or another. Any contemporary
city is governed by a smart city cyberinfrastructure (SCCI)[3]. The cyberinfrastructure, in particular, controls the city and its resources, such as water, transportation, and power usage. The following are some of the most critical security concerns with smart city security. The first of these concerns is whether the security of the technology that enables the notion of a smart city can be maintained in light of their fast expansion. Tablets, computers, cellphones, and other new gadgets have made it simpler for prospective attackers to identify security flaws in the cyberinfrastructure. In addition, with the development of city-wide Wi-Fi, which allows for continual internet access at any time in many places, the potential of an assault has only grown. Another challenge is ensuring that staff is properly trained in how to protect a smart city network. Because of the rapid growth and spread of smart cities, only a few security experts are qualified to manage and support a smart city security system. Another problem is the distribution of patches and security upgrades. For example, with each new upgrade, a new sort of cyber security flaw will emerge. The updates are also pushed out due to the rapid expansion of smart city programs. As a result, there will be opportunities for an attack like a SQL injection. When software is not adequately tested and released to the SCCI, it may be vulnerable to genuine attacks, posing a significant security risk.

To create a highly secure smart city application, the program’s security protocols must be extremely reliable, error-free, and resistant to attacks. Because of numerous flaws, the validation approach is insufficient to achieve high reliability of security protocols; it shows the presence of problems but never ensures their absence. So, to obtain high reliability and verify the correctness of security protocols mathematically, the formal verification approach[5] is the ideal choice. When a device design is exposed to particular formal conditions/properties, formal verification is used to determine if it fulfills all of the specifications/requirements.

Fig.1 shows an IoT network scenario where the IoT devices are connected via wifi. In the proposed work the device-to-device Challenge Handshake Authentication Protocol CHAP for IoT authentication protocol model is formally verified using the model checking technique.

1.1. CHAP- Challenge-Handshake Authentication Protocol for IOT
CHAP is a security protocol that allows an end-user or a network host to be verified by an authenticating authority. CHAP uses a continuously growing identity and a changing challenge-value to guard against replay attacks. In CHAP, both the client and the server must know the plaintext of the secret parameter, but it is never sent across the channel. As a result, CHAP outperforms Password Authentication Protocol in terms of security. MS-CHAP is a Microsoft version of CHAP that does not require any peer to know the plaintext and does not communicate it, but it has been cracked. The CHAP employs a three-way handshake to verify a peer’s identity.
A regular connection establishment is done when the connection is first established, and it can be repeated at any point thereafter.

1. After the connection setup procedure is done, the authenticator sends a "challenge" message to the IoT peer.
2. A value computed using a "one-way hash" function is returned by the peer.
3. The authenticator compares the answer to the predicted hash value calculated by itself. The authentication is accepted if the values match; otherwise, the link must be terminated.
4. At random intervals, the authenticator sends a fresh challenge to the peer, and steps 1 through 3 are repeated.

During the Link Establishment operation, each end of the point-to-point connection must first broadcast Link Control Packets (LCP) packets to set up the data link to establish communications over a point-to-point (PPP) link. After the connection has been established, PPP allows for an optional Authentication step.

CHAP defends against peer playback attacks by using a gradually developing identity and using a dynamically changing challenge value packet. The purpose of using repeating challenges is to minimize the length of time a person is exposed to a single attack. The number and timing of the challenges are determined by the IoT authenticator. The authenticator and the peer are the only ones who know the "code" used in this type of authentication. The relation is not used to send the password. Because the authentication is just one-way, the same secret set can be used for reciprocal authentication by negotiating CHAP in both ways. Because CHAP may be used to authenticate a variety of systems, the name fields can be used as an index to find the proper secret among a large number of them. This allows for many name/secret pairings per device, as well as the ability to alter the secret used at any point throughout the session.

The secret must be at least 1 octet long according to the CHAP algorithm. The key must be at least as long and difficult to guess as a strong password. It’s ideal if the secret is at least as long as the hash value for the hash method you’ve chosen (16 octets size for MD5 hash algorithm). This is to protect the confidentiality property from exhaustive search assaults. The one-way hash approach was selected because deducing the secret from the known challenge and answer values is computationally infeasible. Because an attacker can respond with a previously intercepted answer if a challenge value is repeated with the same secret, each challenge value must be unique. Because the same secret might be used to authenticate with servers all over the world, the challenge must be both globally and temporally unique. Each challenge value must be unexpected so that an attacker cannot mislead a peer into responding to a hypothetical future challenge and subsequently impersonate that peer in front of an IoT authenticator with that response. The CHAP begins with the Challenge package. The authenticator must send a CHAP packet with a Challenge value of 1 to the authenticator. A Challenge packet can be issued at any point throughout the Network-Layer Protocol protocol to confirm that the link has not been interfered with. During the Authentication and Network-Layer Protocol phases, the peer might expect Challenge packets. The peer must transmit a CHAP packet if a Challenge packet is received. When a Response packet is received, the authenticator must deliver a Success or Failure packet depending on the comparison. You’ll be allowed to go to the next stage when you’ve finished the authentication procedure. All Response packets received with the current Challenge Identifier must have the same reply Code that was previously returned for that Challenge to avoid alternative Names and Secrets being discovered. Any response packets received during another operation must be silently destroyed.

2. Literature Review

The creation of secure communication between communicative agents is aided by security protocols. The majority of digital communication networks rely on them, including encrypted internet conversations, bank deposits, and any electronic form of financial transfer. Such
Security protocols are tough to develop even with flawless cryptography. The most well-known authentication protocol Needam-Schroeder authentication protocol [8] was initially presented in 1978 and it’s been around for a long time to authenticate the communicating agents using a combined approach that includes a public key and symmetric key cryptosystems. In 1995 Galvin-Lowe [11] broke the Needam-Schroeder protocol using FDR model checker, and a new modified version of NS protocol called Needam-Schroeder-Lowe protocol was developed. This demonstrates the need for formal security protocol verification.

If design errors were introduced during the system development process of authentication security protocols, it is extremely difficult to identify and fix using testing methods. Formal verification is the most effective method for detecting and addressing design flaws, as well as ensuring dependability. It gives statistical verification that security mechanisms are functionally correct.

Figure 2 shows the types of Formal Verification techniques. A security protocol’s high-level description, as well as its security requirements and a specific instantiation, are transformed into CSP, which can be verified using a model checker that uses the Model checking approach. The attacker has total access to the network and is permitted to behave as one of the agents. The strand space solution, which is a Doley-Yao [9] model in essence that portrays a powerful attacker and the attacker capabilities. Several present research works describe the current state of the art for using the spin model checker to formally verify the NSPK and Denning Sacco DS protocol [10] describes how a model checker can be used to formally validate the protocol’s security properties.

To formally verify a security protocol there are several tools available which works based on the model checking technique such as Casper [11], Proverif [12], Cryptoverif [13], Span-Avispa [14] and Scyther model checker [15] etc. Out of these tools, the Scyther tool has the strongest adversary model and it has got better GUI and attack graphs which is very helpful in analysing the security protocol.

2.1. Scyther Model Checker
Scyther is a model checker for systematically analysing and verifying IoT authentication security methods. The Scyther tool validates IoT authentication security protocols using a black-box [16] analysis technique based on the perfect cryptography assumption, which states that an attacker can’t learn or break any data from the ciphertext unless he has the decryption key. The main objective of the tool is to find security loopholes in security protocol design. Attacks will be detected if the procedures are proven to be correct.

Scyther model checker incorporates the Dolev-Yao (DY) threat model [9] as the which is the
The DY attacker model is an active attacker who has got full control over the network through which the communication agents send and receive data, on which the agents interact and the adversary has complete control over the channel. This allows the attacker to listen in on any communication messages, execute reply attacks, and get information about the agents’ cryptographic primitives, he can delete the user sent messages and can analyse the data content, he can push his own created messages or data, or he can reroute the data traffic or simply he can retransmit messages that he has received and he can observe the traffic.

3. Proposed Methodology

Formal verification is a difficult process that necessitates a great deal of effort and experience required while developing a system, with help of automated tools verification of even a complex security system is made very easy. Fig.3 shows the architecture of verifying the CHAP Authentication protocols. The CHAP protocol specifications were obtained from its RFC [6], using those specifications an abstract protocol model is developed and the security properties were specified as claims and verified using the scyther model checker.

3.1. Challenge Handshake Authentication Protocol-CHAP PPP IOT Protocol verification model

CHAP [6] defends against peer replay attacks by using an incrementally developing identity and a changeable challenge-value. In CHAP, both the client and the server must know the plaintext of the secret, but it is never sent across the network. As a result, CHAP has a higher level of stability. Point-to-Point Protocol (PPP) servers use the CHAP authentication scheme to verify the identity of remote IoT peers. CHAP uses a three-way handshake to verify the client’s identity regularly. This happens when the initial connection (IUCP) is formed, and it can happen again at any time after that. A mutual secret (such as the client’s password) is used for authentication. The authenticator sends a "challenge" message to the peer after the link establishment process is completed. The peer responds with a value determined by combining the challenge and the secret with a one-way hash function. The authenticator compares the answer to the estimated hash value is calculated. If the values match, the authenticator accepts the authentication request otherwise, the relation must be terminated. The authenticator creates a new challenge for the client at random intervals, and steps 1 through 3 are repeated. The CHAP
Figure 4. IOT Device Challenge-Handshake-Authentication-Protocol.

PPP protocol formal specifications were obtained from the RFC1998 [6] which provides the complete specification of the CHAP PPP protocol that how the protocol must be implemented. The below Algorithm-3 shows the CHAP PPP Protocol abstract security model and it contains 2 communicating roles namely PEER and AUTHENTICATOR. This protocol has 3 message exchanges between the peers of the IoT network. In Step1. PEER sends the connection request to the AUTHENTICATOR by sending the request and PEER Identity encrypted with the public key of the Authenticator. In Step2 the Authenticator sends the PEER the hash(request) and challenge. In Step3 the PEER sends the Authenticator the has hash value of the challenge which authenticates the peer with the Authenticator.
Algorithm 1 CHAP PPP IOT Protocol Description.

Communicating Roles: PEER, AUTHENTICATOR

#Data from PEER
request - request of type Nonce
user-name - user name of type String
pswd - password of type String

Step1. PEER $\rightarrow$ AUTHENTICATOR: $\{\text{request}, \text{PEER}\} \text{pk(/Authenticator)}$

kir - Shared session key.
hash() - Hash function.

Step2. AUTHENTICATOR $\rightarrow$ PEER: $\{\text{hash(request)}, \text{challenge, Authenticator, kir}\} \text{k(PEER, Authenticator)}$

Step3. PEER $\rightarrow$ AUTHENTICATOR: $\{\text{hash(challenge)}, \text{user-name, pswd}\} \text{kir}$

#Security Claims by PEER role
Secret $\rightarrow$ kir
Secret $\rightarrow$ user-name
Secret $\rightarrow$ pswd
Niagree
Nisynch
Alive

#Security Claims by AUTHENTICATOR role
Secret $\rightarrow$ kir
Secret $\rightarrow$ user-name
Secret $\rightarrow$ pswd
Niagree
Nisynch
Alive

#Intruder Information
Intruder = EVE
IntruderKnowledge = PEER, AUTHENTICATOR, pk(PEER), pk(AUTHENTICATOR), pk(EVE), sk(EVE)

4. Verification Results

4.1. CHAP PPP protocol Verification Results

In the proposed work, the CHAP PPP protocol security protocol abstract model was built using its RFC specification document. Figure 5 shows the verification result of the CHAP PPP authentication protocol in which the proposed crypto for CHAPP PPP protocol for IoT is safe, secure, and free from crypto attacks corresponding to the security claims made in the protocol model. Hence the IOT peer can securely authenticate with the Authenticator by the challenge-response values and the CHAP PPP protocol satisfies the secrecy, non-injective agreement, non-injective synchronization, and aliveness security properties.
5. Conclusion
By giving mathematical proofs for the functional correctness of CHAP PPP used in smart city applications we can prove its reliability. The formal verification technique has the potential to improve the reliability of security protocols used in smart city networks. One of the most difficult aspects of formal verification of smart city applications is creating an abstract model for security protocols used by it and defining security properties to be satisfied by the protocol. The Scyther model checker was used to formally verify the CHAP authentication security protocol and the protocol is free from attacks for the respective security claims, and the CHAP can be used by smart city IoT devices for Peer-to-Peer authentication.

6. Future Work
The proposed CHAP protocol model must be further needs to be optimised by reducing the number of states in the state space. Most advanced central cloud-based authentication protocols such as SSH[17], TACACS[18], TACACS+[19], Kerberos[20], and other advanced smart city network authentication protocols must be formally verified using model checking techniques.

References
[1] Su, K, Li, J and Fu, H 2011 September Smart city and the applications international conference on electronics communications and control ICECC pp 1028-1031
[2] Lee, I and Lee, K 2015 The Internet of Things (IoT): Applications investments and challenges for enterprises Business Horizons 58 431-440
[3] Zhang, K, Ni, J, Yang, K, Liang, X, Ren, J and Shen, X.S 2017 Security and privacy in smart city applications: Challenges and solutions IEEE Communications Magazine, 55 122-129
[4] Wang, P, Ali, A and Kelly, W 2015 August Data security and threat modeling for smart city infrastructure international conference on cyber security of smart cities industrial control system and communications pp 1-6
[5] Meadows, C.A 1994 November Formal verification of cryptographic protocols: A survey. In International Conference on the Theory and Application of Cryptology Springer Berlin Heidelberg pp 133-150
[6] Housley, R. and Aboba, B 2007 Guidance for authentication authorization and accounting (AAA) key management BCP 132 RFC 4962
[7] Le Guernic, G 2008 Precise dynamic verification of confidentiality In 5th International Verification WorkshopVERIFY 08 p 82
[8] Lowe, G 1995 An Attack on the Needham Schroeder Public Key Authentication Protocol Information processing letters 56 3
[9] Rocchetto, M and Tippenhauer, N.O 2016 November CPDY: extending the Dolev-Yao attacker with physical-layer interactions In International Conference on Formal Engineering Methods Springer Cham pp 175-192
[10] Steiner, M, Tsudik, G. and Waidner, M 1995 Refinement and extension of encrypted key exchange ACM SIGOPS Operating Systems Review 29 22-30
[11] Lowe, G 1997 June. Casper: A compiler for the analysis of security protocols In Proceedings 10th Computer Security Foundations Workshop pp 18-30
[12] Blanchet, B 2013 Automatic verification of security protocols in the symbolic model: The verifier proverif In Foundations of security analysis and design Springer Cham pp 54-87
[13] Blanchet, B 2007 CryptoVerif: Computationally sound mechanized prover for cryptographic protocols In Dagstuhl seminar Formal Protocol Verification Applied 117 156
[14] Hernandez-Ardieta, J.L, Gonzalez-Tablas, A.I and Ramos, B 2009 Formal Validation of OFEPPSP+ with AVISPA In Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security Springer Berlin Heidelberg pp. 124-137
[15] Cremers, C.J 2008 The Scyther Tool: Verification falsification and analysis of security protocols In International conference on computer aided verification Springer Berlin Heidelberg pp 414-418
[16] Pelled, L.H, Eisenhardt, K.M and Xin, K.R 1999. Exploring the black box: An analysis of work group diversity conflict and performance Administrative science quarterly 44 1-28
[17] Ylonen, T. and Lonvick, C., 2006. The secure shell (SSH) protocol architecture.
[18] Finseth and C 1993 An access control protocol, sometimes called TACACS. RFC 1492
[19] Carrel and D 1997 TACACS+ Protocol Specification
[20] Neuman, B.C and Ts'o, T 1994 Kerberos: An authentication service for computer networks IEEE Communications magazine 32 33-38