American Journal of Computer Science and Engineering
2014; 1(5): 39-42
Published online January 10, 2015 (http://www.openscienceonline.com/journal/ajcse)

An authentication scheme based on chaos, a TTP, and a DNA sequence

Ralph M. DeFrangesco

Drexel University, College of Computing and Informatics, Philadelphia, PA. 19104 USA

Email address

rd337@drexel.edu

To cite this article

Ralph M. DeFrangesco. An Authentication Scheme Based on Chaos, a TTP, and a DNA Sequence. *American Journal of Computer Science and Engineering*. Vol. 1, No. 5, 2014, pp. 39-42.

Abstract

As individuals, we count on authentication to protect the systems that store our most important information on. Critical systems include medical, financial, and systems that run our nation's infrastructure. This paper will discuss how chaos can be used in conjunction with a TTP and a DNA sequence to create a onetime credential that can be used to authenticate to a system.

Keywords

Chaos, TTP, Non-Linear Dynamics

1. Introduction

Authentication protocols are typically error-prone and difficult to implement (Muhammad, 2013). A strong authentication protocol is essential to any secure networked system because strong authentication instills trust (Muhammad, 2013).

Remote Authentication Dial-In Service (RADIUS) is a widely used service for authentication, authorization, and accounting services (Poznyakoff, 2008). Although RADIUS is widely used, the protocol has many known vulnerabilities.

Kerberos is considered a trusted third-party authentication protocol. It is based on the Needham and Schroeder encryption algorithm. Trust in this case is a matter of reference. The security part of the protocol works by giving complete trust to the Kerberos server, allowing it to determine if it trusts a client or not. The system uses tickets and timestamps and is based on the Needham and Schroeder algorithm for encryption. According to Wang & Feng (2013), Kerberos has suffered from several known vulnerabilities including: a replay attack, a dictionary attack, key storage problems, malware attacks, and the authentication forward problem.

Password authentication protocol (PAP) is a point-to-point protocol used with wireless networks. The basic protocol provides a methodology for a peer to establish a communications connection using a simple two-way handshake. The protocol has had several known flaws and there has been an attempt to fix the original release (Ma, McCrindle, & Cheng, 2006).

SSL has until recently been one of the more reliable protocols for encrypting information over the Internet. Even though it has been replaced by TLS, it is still used quite a bit.

However, recently the latest version has been broken. POODLE is an attack that hackers can user to cause a downgrade, and thereby breaking the cryptographic security of the protocol.

2. Current Methodology

Password based authentication is still the most commonly used method for authentication, but it provides minimal security to its users (Sood, Sarje & Singh, 2009). Today, basic authentication is still done with a login and password. In almost all organizations, a login is often assigned to an individual or server. This can be the first initial and last name of the individual or the system name. So, half of the equation is already known or can be guessed.

Although passwords still remain the primary method for securing data, they are susceptible to attacks. Jingbo & Pingping (2010) have found that user chosen passwords are

inherently insecure since they come from a limited universe. This limited universe makes them vulnerable to dictionary and brute force attacks. There are three types of credentials that use can use to authenticate; something you know, something you have, and something you are (Lakshmiraghavan, 2013). Two-factor authentication uses two of these options for authentication. Two-factor authentication can be added to increase security, but again it is susceptible to being broken or a side channel attack.

To compound the problem even further, many of the protocols that we use to protect us, can be broken or reveal information in plain text. For instance, Telnet and FTP send login credential information in plain text and SMTP sends mail messages in plain text yet they continue to be used.

3. Proposed Solution

This paper proposes an authentication scheme that uses chaos as a randomizer for a login and a Trusted-third Party (CA) DNA sequence as a password that creates a multi-factor, two-system, onetime credential used for authentication.

The one-time credential is based on the use of a one-time pad. Traditionally, a one-time pad uses a random sequence of numbers (key) that is combined mathematically with the message and then added together using modular addition. The result is a cipher that only the person receiving the message with the same key, can decode (Katz and Lindell, 2008).

A Trusted Third Party provides non-repudiation, equality, and fairness (Li, Wu & Li, 2013). In this solution, the Certificate Authority (CA) creates the random key. The individual user is given an application that creates the cipher text that is used as a one-time login and password, or credential as explained Figure 1.

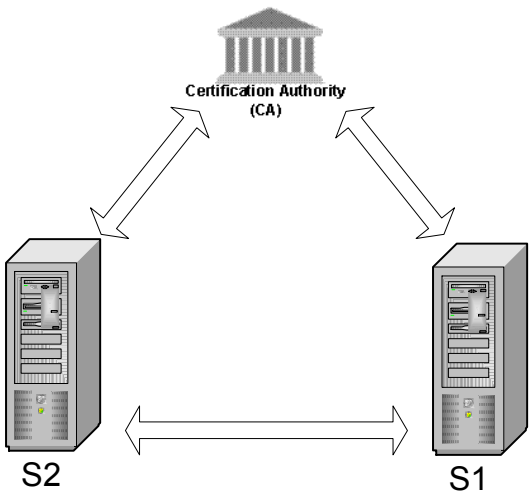

Figure 1. *A CA as a trusted third-party*

S1 is the server that starts the authentication sequence. S2 is the server that S1 wants to log into. The certificate authority is used to create two random values used as input into a chaos number generator. The login sequence is as follows:

1 S1 requests to log into S2 and sends a login request.
2 S1 also sends a request to the CA for two random values; the number of iterations the chaos generator uses and the initial input.
3 The CA sends the random values to both S1 and S2.
4 The CA, S1, and S2 use the random values as an input into the chaos number generator to come up with a chaos value.
5 Both S1 and S2 send a hash of the sequence to the CA for validation.
6 If all three values match, then the hashed value is used as the login.
7 S1 sends the login to S2.
8 The CA requests the IP address from S1.
9 The CA forwards the IP address to S2.
10 S2 requests the password. This is the DNA sequence based on the chaos value.
11 S2 validates the IP, the login, and password from S1.
12 If all three values match, then S2 allows S1 to authenticate.

4. Application Example

The Python random number generator creates two random numbers that will be used by the chaos generator. One is for the initial input value and the other is the number of iterations that the chaos generator will make. Although the Python random number is not truly random, it will suffice for the chaos generator.

The Certificate Authority runs this application to create the values that are used in the chaos calculator

```
import random
# Create a random value for the iteration input
iteration_val = random.randint(0,1000000)
print(iteration_val)
# Create a random value for the initial input
initial_val = random.random()
print(initial_val)
```

Output:

12000 (this is the number of iterations)

0.120118082049 (this is the initial input for the chaos generator)

Note: each time the random number runs, the values will be different)

The output of the random number generator is sent to both S1 and S2. Now, the CA, S1, and S2 all have the same random values. Each takes the two numbers from the Python random number generator and inputs them into the chaos generator. The chaos generator will create another random value.

This application takes the two inputs from the Python random generator and uses them to create a random number based on a chaos algorithm.

```
#Get the number of iterations
iterations = int(input("Enter the number of iterations: "))
```

```
# Gets the initial value
initial_value = float(input("Enter initial value: "))

# Prime the algorithm
temp_value = ((4 * initial_value) * (1 - initial_value))

# Loop through the algorithm by the number of iterations
for i in range(iterations):
    chaos_value = ((4 * temp_value) * (1 - temp_value))
    temp_value = chaos_value
    temp_string = str(temp_value)

# Print the number created by the chaos #generator
print("\nChaos value: ", temp_string)

# Open a file for writing
text_file = open("chaos_values.txt", "w")
text_file.write (temp_string)
text_file.close()

input("\nPress any key")
```

Now that there is a random number, based on chaos, the next step is to cut out the DNA sequence based on the chaos number that will be used as the password. The application loads the basic DNA sequence, gets the first two numbers from the random chaos value (minus the ".0") and uses it as a starting point. Next, the chaos value is loaded and used to get the random DNA sequence based on the individual values.

```
# This application loads the chaos value and
# creates a DNA sequence that will be used as a
# password for authentication


# Open the DNA file and load the DNA sequence
protein_file = open("Protein.txt", "r")
protein_line = protein_file.read()
protein_length = len(protein_line)
protein_file.close()


# Open the file with the chaos value in it to get the random
starting point
text_file = open("chaos_values.txt", "r")

# Strip off the leading "0." and get the random starting
point
strip_numbers = text_file.read(2)
temp_pos = text_file.read(2)
start_pos = int(temp_pos)
text_file.close()

# Open a the file with the chaos value in it again
text_file = open("chaos_values.txt", "r")

# Strip off the leading "0."
strip_numbers = text_file.read(2)

offset = []
position = []
i = 0

# Get the next 12 numbers and the random DNA sequence
starting at the random position
offset = text_file.read(12)
for i in range(12):
    position = offset[i]
    position = (start_pos + int(position))
    print(protein_line[position])

text_file.close()

input("\n\nPress any key")
```

The authentication scheme uses the chaos value as the login and essentially, the random number created by the chaos generator will be used to create a random sequence of DNA. This random sequence of DNA will be used as the password.

The strong point about this authentication scheme is that it is a onetime pad. This is to say that the login and password can only be used once. The sequence has to be rerun every time S1 needs to log into S2. This onetime pad makes it very difficult to guess the login and password. The use of a chaos generator based on an algorithm that is protected from public view, a DNA sequence that is changed on a regular basis, and a Certificate Authority that validates credentials are additional protection that make this authentication scheme a very viable option.

5. Conclusion

Authentication protocols are meant to protect information, however many are broken and do not provide protection like they should. This paper proposed a solution that uses a TTP, and a chaos generator to create a random DNA sequence that can be used as a onetime credential for authentication.

Appendix A

Contents of the Protein.txt file:

CCATAGCACGTTACAACGTGAAGGTAATTCCCGAG
GTTATATGGCCCACACTGTGGAACATTACCCATATCT
GCGTTGCCAGAA

References

[1] Jingbo, Y. & Pingping, S. (2010) A secure strong password authentication protocol. Second International Conference on Software Technology and Engineering, pgs 355-357.

[2] Katz, J., & Lindell, Y. (2008). Introduction to modern cryptography. Chapman and Hall: Boca Raton, FL.

[3] Lakshmiraghavan, B. (2013). Two-factor authentication. In *Pro ASP. NET Web API Security* (pp. 319-343). Apress.

[4] Li, Q., Wu, K. & Li, F. (2013) An optimistic non-repudiation protocol focused on transparent trusted third party IEEE Conference on High Performance Computing and Communications, pgs 682-689.

[5] Liu, D., & Coslow, M. (2008), Extensible authentication protocols for IEEE standards 802.11 and 802.16. The International Conference on Mobile Technology, Applications & Systems. September 10-12, 2008

[6] Ma, X., McCrindle, R. & Cheng, X. (2006). Verifying and fixing password authentication Protocol. Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel Distributed Computing.

[7] Muhammad, S (2013) Applying authentication tests to discover man-in-the-middle attack in security protocols. Eighth Conference on Digital Information Management, pgs 35-40.

[8] Poznyakoff, S. (2008) GNU Radius Reference Manual. GNU Press. Boston, MA.

[9] Sood, S, Sarje, A. & Singh, K. (2009) Cryptanalysis of password authentication schemes: Current status and key issues. International Conference on Methods and Models in Computer Science.

[10] Wang, C. & Feng, C. (2013) Security analysis and improvement for Kerberos based bynamic password and Diffie-Hellman algorithm. Fourth International Conference on Emerging Intelligent Data and Web Technologies, pgs 256-260.