Australian Journal of Science and Technology

ISSN Number (2208-6404)
Volume 3; Issue 2; June 2019

Original Article

An enhanced web security for cloud-based password management

Ridwan Olayinka Oladipupo, Ajayi Olusola Olajide*

Department of Computer Science, Adekunle Ajasin University, Akungba-Akoko, Ondo, Nigeria

ABSTRACT

Password is a security mechanism for securing application and its contents by preventing unauthorized users' access, using secure means. Breaches of access by unauthorized users have, however, become a subject of concern to many web developers and application owners. Cloud-based password management system represents the storage and access of web passwords through the "Cloud." The study examines and analyzes the current password management status of web application (Adekunle Ajasin University, Akungba-Akoko) with a view to detect the flaws/insecurity in the site. This paper uncovers the vulnerabilities of existing web application and analyzes how they can be exploited by attackers to crack users' saved passwords. The study proposes a novel cloud-based password management design to achieve a high level of security with the desired confidentiality, integrity, and availability properties. The study, which employed cryptographic hash function (Secure Hash Algorithm-256) and Diffie–Hellman key exchange algorithm, was designed using penetration testing technique and implemented a highly secure cryptography (i.e., a zero-knowledge protocol) for making the site more tightly secured, thereby ensuring a secure channel through the data flows.

Keywords: Availability, breaches of access, cloud, confidentiality, cryptographic, integrity, password management, password, secure, security

Submitted: 29-04-2019, **Accepted:** 16-05-2019, **Published:** 29-06-2019

INTRODUCTION

Since the internet is open systems and the web applications are increasingly used to deliver critical services, they become a valuable target for security attacks. The security of the web applications become the main concern to many users of the web applications, especially when the web application is interactive and requires the exchange of sensitive information such as financial, health, or credit card numbers. If these web applications were not secured, then the entire database of sensitive information is at serious risk. Therefore, there was great effort in both the research and industry communities to provide secure communication services to web applications. A great deal of attention has been given to network-level security, such as port scanning, and great achievements have been accomplished at this level as well. However, it was found that about 75% of attacks were targeted to application level, such as web servers.[1]

According to Ajayi and Fanala,[2] it was observed that web application introduced in 1990 was a general, delivery mechanism. It transformed from a static hypertext document to a complete dynamic runtime environment for multiparty and distributed applications. The emerging trend was popular in peer-to-peer web applications and multiple applications. However, the transformation of the web application from the server-centric model creates a significant and numerous challenges in web applications security.[3]

Password compromise is still the root cause behind many cyber breaches. In 2014, two of three breaches involved attackers using stolen or misused credentials.[4] Yet, the majority of internet users still do not follow secure password management practices.[5] Password is the most common method for users to authenticate themselves when entering computer systems or websites. It acts as the first line of defense against unauthorized access, and it is, therefore, critical to maintain the effectiveness of this line of defense by rigorously practicing a good password management policy.

According to Adams and Sasse[6] study on password habits, it was found that insecure password practices can, in general,

Address for correspondence: Ajayi Olusola Olajide, Department of Computer Science, Adekunle Ajasin University, Akungba-Akoko, Ondo State, Nigeria. Tel: +234(70)56433798. E-mail: olusola.ajayi@aaua.edu.ng

not be caused by user carelessness, but on the inadequacy of policies under which users have to manage passwords. However, a cloud-based synchronization across devices and password managers promise tremendous security and usability benefits at minimal deployability costs.[7] Secure password practices result in numerous cryptic passwords which are very difficult to keep track of. It is impossible for most people to consistently remember more than just a few of them. Yet, according to a recent survey conducted by Google, over 50% of users reported that they rely on memory alone to keep track of their passwords. The fact that they rely solely on memory is a clear indication that they are not following secure password practices because if they can remember all of their passwords, then they must be creating simple passwords, or reusing passwords for multiple accounts, or both.[8]

Furthermore, due to more customer data going online by adapting to online banking or fund transfer practices, user's accounts and other information have become vulnerable to fraud and other attacks. Furthermore, hackers in recent years are increasingly targeting web applications since most networks are closely monitored through Intrusion Detection Systems and firewalls. Therefore, the web application layer needs to be secured from unauthorized users by building across the software development lifecycle security mechanism.[9] This ensures that it is not an afterthought issue, only considered in the end[10] as in many software development processes, where as a result, attackers continue to explore areas of vulnerability to undermine the integrity of applications. In recognition of this problem, developers have to incorporate security during the development to produce vulnerability-free software systems since the existence of flaws at the design or coding phase of the development lifecycle can open web applications to a wide range of attacks.[11]

Hence, one of the important sectors that exploit the web technology in their services is the education sector such as research institutions, universities, and training organizations. Web application and websites are heavily used in education for information dissemination, lectures, assignments, collaborations, discussions, conferences, grading, training, distance learning, research activities, e-voting, and many others. Web applications in education sector usually hold sensitive information, such as faculty member researches, student results, and staffs account. These data or information need to be secured from non-authorized users. Unfortunately, the sense and awareness of securing these data have not received great attention from academicians. While securing enterprise data are usually focused on financial, military, or demographic organizations, it is often neglected in education organizations.

The intentions behind this research work are in two-fold: First, to observe and analyze the current password management status of web application (Adekunle Ajasin University, Akungba-Akoko [AAUA] AVERS) with a view to detect the flaws/insecurity in the site and second, to design a highly secure cryptography (i.e., a zero-knowledge protocol) for making the site/service more tightly secured and to ensure a secure channel through with the data flows.

According to Scott and Kent,[12] cloud-based password managers have become popular and ease the burdens of password management on users that are willing to trust a third party to store their passwords. Li *et al*.[13] and Silver *et al*.[14] in their recent security evaluations of password managers revealed security flaws in popular password managers that can result in the compromise of user's passwords. Password managers are designed to ease the burden of password management.

Password management applications are recommended by the vast majority of IT security experts. Yet, the vast majority of non-experts do not use them to manage their passwords. A recent survey of both security experts and non-expert web users conducted by Google revealed some interesting differences between the two groups. According to the Google survey, 73% of security experts use a password manager themselves, compared to only 24% of non-experts. Both experts and non-experts agree that it is very important to follow secure password practices, but they disagree on the benefits of using a password management application for this purpose. About 48% of the security experts polled ranked the use of password management applications as one of the top things people can do to stay safe on the internet, while only 3% of non-experts thought that this was an important practice to follow. It seems that the average web user is either unaware of the benefits gained using a password management application or that they do not trust them to keep their passwords secure.[8]

Text-based passwords still occupy the dominant position in online user authentication.[7,15] They protect online accounts with valuable assets and thus have been continuously targeted by various cracking and harvesting attacks. Password security heavily depends on creating strong passwords and protecting them from being stolen. However, researchers have demonstrated that strong passwords that are sufficiently long, random, and hard to crack by attackers are often difficult to remember by users.[16] Meanwhile, no matter how strong they are, online passwords are also vulnerable to harvesting attacks such as phishing.[17-19] These hard problems have been further aggravated by the fact that web users have more online accounts than before and are forced to create and remember more and more usernames and passwords probably using insecure practices such as sharing passwords across websites.[20] Password manager, particularly browser-based password manager (BPM), is one of the most popular approaches that can potentially best address the online user authentication and password management problems. Browser integration enables BPMs to easily save users' login information including usernames and passwords into a database, and later automatically fill the login forms on behalf of users. Therefore, users do not need to remember a large number of strong passwords; meanwhile, BPMs will only fill the passwords on the login forms of the corresponding websites and thus can potentially protect against phishing attacks. Fortunately, all the five most popular browsers Internet Explorer, Firefox, Google Chrome, Safari, and Opera have provided password managers as a useful built-in feature. Unfortunately, the designs of all those BPMs have severe security vulnerabilities. In essence, our key observation is that the encrypted passwords stored by those BPMs are very weakly protected – they can be trivially decrypted by attackers for logging into victim's accounts on the corresponding websites. We have developed tools to demonstrate that once stolen, the encrypted website login information saved by the five browsers (without using a master password in Firefox and Opera) can all be easily decrypted by attackers. When a master password is used in Firefox or Opera, even though decrypting a user's login information becomes harder, brute force attacks and phishing attacks against the master password are still quite possible.

This paper uncovers the vulnerabilities of existing web application and analyzes how they can be exploited by attackers to crack users' saved passwords. Moreover, we propose a novel cloud-based password management design to achieve a high level of security with the desired confidentiality, integrity, and availability properties. Cloud-based password management is cloud-based storage in the sense that the protected data will be completely stored in the cloud – nothing needs to be stored on a user's computer. This study intends to move the storage into the cloud for two main reasons. One is that in the long run, trustworthy storage services in the cloud[21] can better protect a regular user's data than a local computer. The other is that the stored data can be easily accessible to the user across different operating system accounts and different computers. It is believed that cloud-based management is a rational design that can also be integrated into other popular browsers to make the online experience of web users more secure, convenient, and enjoyable.

Statement of Problem

Although several researches have been carried out in the area of web security among which include[22] – Surviving the web: A journey into web session security;[12] End-to-end passwords;[23] and All your browser-saved passwords could belong to us: A security analysis and a cloud-based new design, this work observes and analyzes the current password management status of web application (AAUA AVERS) with a view to detecting the flaws/insecurities in the site. Recent studies have explored password managers' usability. Chiasson *et al*.[24] conducted a 20.6-person user study comparing two password managers: PwdHash and Password Multiplier. Ambarish *et al*.[25] conducted a usability study of three password managers. The users in their study preferred local-based password managers to a cloud-based manager. They credited the finding to the reluctance of users to trust an online password manager. This study in its own view is set to design a highly secure cryptography (i.e., a zero-knowledge protocol) for making the site/service more tightly secured.

Related Works

In 1999, Provos and Mazieres[26] proposed ways of building systems in which password security keeps up with hardware speeds. They formalized the properties desirable in a good password system and showed that the computational cost of any secure password scheme must increase as hardware improves. They presented two algorithms with adaptable cost Eksblowfish, a block cipher with a purposefully expensive key schedule and BCrypt, a related hash function. The key setup begins with a modified form of the standard Blowfish key setup, in which both the salt and password are used to set all subkeys. There are a number of rounds in which the standard Blowfish keying algorithm is applied, using alternately the salt and the password as the key, each round starting with the subkeys state from the previous round. Cryptotheoretically, this is no stronger than the standard Blowfish key schedule, but the number of rekeying rounds is configurable; this process can, therefore, be made arbitrarily slow, which helps deter brute-force attacks on the hash or salt. Failing a major breakthrough in complexity theory, these algorithms should allow password-based systems to adapt to hardware improvements and remain secure well into the future.

In 2009, Colin[27] introduced SCrypt which is an improvement on BCrypt. As noted, the main threat against BCrypt in 1999 was application-specific integrated circuit (ASICs) with low gate counts, but today, the threat is field-programmable gate array (FPGAs) and BCrypt was not designed to protect against that threat. In the above cases, a function is being provided that developers can put passwords into to get an encoded result. The solutions are improvements on Morris and Thompson's work for protecting passwords in UNIX systems.

Thulasimani and Madheswaran[28] proposed hash functions that are the most widespread among all cryptographic primitives and are currently used in multiple cryptographic schemes and in security protocols. The basic design of Secure Hash Algorithm (SHA)-192 is to have the output length of 192. The SHA-192 has been designed to satisfy the different levels of enhanced security and to resist the advanced SHA attacks. The security analysis of the SHA-192 is compared to the old one given by National Institute of Standards and Technology and gives more security. The SHA-192 can be used in many applications such as public key cryptosystem, digital sign encryption, message authentication code, random generator, and in security architecture of upcoming wireless devices such as software-defined radio.

In Richa *et al*.,[29] the authors proposed a new hash function algorithm that includes 64 bits key as an ingredient to the function. It produced 128 bits digest with a secure and simple technique as compared to many other existing techniques. The author submitted that the use of key adds the source integration facility while creating digest just for integrity purpose.

Selva and Anuja[30] in their paper secured password management technique using one-time protocol in smartphone. They proposed a user authentication technique that can be used to prevent from password stealing and password reuse attacks by installing the applications on the Android smartphone. Deepika *et al*. proposed user security in cloud using password authentication. The study implements password authentication technique to enhance the security of cloud by creating an algorithm based on the selection of username and generates a password to strengthen the security in cloud.

In Disha,[31] the paper justified MD algorithm as one that enhances security of data by generating digital signature. He proposed an algorithm that has five phases: Key generation, digital signing, encryption, decryption, and signature. The evaluation of the result shows that the algorithm would be a high security algorithm for data transfer.

The paper, Sahni[32] discussed common cryptographic hashing algorithms and compared their relative performance. Majoring on MD5 and SHA-1, the evaluation result shows MD5 produced a hash value of 128 bits, whereas SHA-1 produced 160 bits. While submitting that SHA-1 proved more secure than MD5, he, however, pointed out that in terms of ease of implementation, MD5 is more suitable.

In Sriramya and Karthika,[33] the study focuses on providing security to user's data using salted password hashing technique. To ensure this and enforce a tight security procedure, BCrypt algorithm was implemented to secure user's privacy when shopping online. BCrypt is currently the secure standard for password hashing. It is derived from the Blowfish block cipher which, to generate the hash, uses lookup tables which are initiated in memory. This means a certain amount of memory space needs to be used before a hash can be generated. This can be done on central processing unit, but when using the power of graphics processing unit (GPU), it will become a lot more cumbersome due to memory restrictions. BCrypt has been around for 14 years, based on a cipher which has been around for over 20 years. It has been well vetted and tested and hence considered the standard for password hashing. When BCrypt was originally developed, its main threat was custom ASICs specifically built to attack hash functions. These days those ASICs would be GPUs (password brute forcing can actually still run on GPU, but not in full parallelism) which are cheap to purchase and are ideal for multithreaded processes such as password brute forcing. FPGAs are similar to GPUs, but the memory management is very different. On these chips, brute-forcing BCrypt can be done more efficiently than on GPUs, but if you have a long enough password, it will still be unfeasible. The iteration count is a power of two, which is an input to the algorithm.

Tivkaa *et al*.[34] researched an enhanced Password-Username Authentication System Using Cryptographic Hashing and Recognition-based Graphical password. The research work presented an authentication solution that addresses the issue of SQL injection (SQLI) and online password guessing attacks on login form as implemented in generic web applications. The solution combined the use of plain text credentials that are cryptographically hashed at runtime with recognition-based graphical login credentials. The goal is to always guarantee access to a user account even when such account is under attack while at the same time ensuring convenient and secure login experience by legitimate users. The goal was achieved by blocking the Internet Protocol addresses from which there are unsuccessful login attempts. However, security test shows that the solution is not vulnerable to SQLI and online password guessing attacks. In other to mitigate online guessing and SQLI attack on authentication functionality, the researcher combines a technique which combines the use of cryptographic hashing algorithm, recognition-based graphical password, and parameterized queries.

In Ajayi and Fanala,[2] the study carried out an empirical evaluation of data hashing algorithms for password checks in PHP Web Applications taking SHA1, MD5, BCrypt, SCrypt, Salt, and SHA256 into consideration and implementing a comparative analysis of the specified algorithms to prove their vulnerabilities and strengths. The research work helps to evaluate the different hashing algorithms. MD5, SHA1, and SHA256 appear so weak, MD5 allows attackers to create multiple, differing input sources that, when this algorithm is used, result in the same output fingerprint. On the other hand, SHA1-2 levels are vulnerable to length-extension and partial-message collision attacks. The salt and crypt hashing algorithm appear strong, they protect against rainbow table attacks and also help to deter brute-force attacks on the hash.

Rituraj *et al*.[35] researched An Effectual Hybrid Approach Using Data Encryption Standard (DES) and SHA for image steganography. The paper presents and discusses Least Significant Bit (LSB)-based image steganography with DES SHA algorithm so as to provide an extra layer of security. The proposed approach deals with the hybrid approach of encryption and cryptography using DES which is popularly known as Data Encryption Scheme and also the SHA-512 algorithm which will be the robust approach to perform the steganography process. For evaluating purpose of the proposed hybrid approach, statistical technique mean square error and peak signal-to-noise ratio was used.

Sanjeev *et al*.[36] presented a secure transfer of university question paper using image steganography. This research discusses on how the security of question paper delivery can be enhanced by image steganography. There are a lot of cases of university question paper leakage. Due to this, university has undergone various problems to tackle this event. The current system that the university uses for question paper transfer is based on face detection which is not very reliable and can be exploited. Moreover, this transfer process is not very secured and is a complex process. However, the system was developed to enhance the security of the transfer process. The system adopts LSB embedding which provides the 1st layer of security, and the embedded data are encrypted by DES algorithm which again a layer of encryption to the embedded data. Audio and video steganography was equally used to make the process of transfer more challenging to crack.

MATERIALS AND METHODS

Methodology

This study adopts the use of cryptographic hash function (SHA-256) and Diffie–Hellman (D-H) key exchange algorithm. Security of password can be done by a technique called cryptography. Cryptography is the science of using mathematics to encrypt and decrypt data. It enables you to store sensitive information or transmit it across insecure communication channels so that it cannot be read by anyone except the intended recipient. Cryptic writing is a method to protect the data from a third party to keep the data in a confidential manner; the technique cryptography is very essential in securing the content over a network. However, there are possibilities for an attack in the communication channel by the attacker who will trace out the key that is used for the encryption as well as the decryption. Hence, a strict method or algorithm known as D-H key exchange algorithm will be used for the secure exchanging of the key.

In the pursuit of a successful implementation of this work, the following procedure shall be followed, as illustrated in Figure 1:

i. Application review: This step performs a penetration testing on the already existing web applications (AAUA AVERS), to deduce the vulnerabilities and flaws/insecurities on the site. Furthermore, the weakness of the type of password management system as well as the type of storage management in place will be checked. However, this will be achieved using a penetration testing tool on Kali Linux.
ii. Application prototype: This step develops a prototype application with same functionality as the existing application. In other to achieve the application prototype development, Html, CSS, JavaScript, and Bootstrap should be used for the frontend development while Python/Django and PostgreSQL will be used for the backend development.
iii. Cryptosystem implementation: This step embeds a zero-knowledge cryptography in the prototype application. SHA-256 will be used to secure user's access information, while D-H key exchange algorithm will be used for securing the data/channel flow.
iv. Cloud storage: The application should be deployed in the cloud storage. This will be achieved by considering deployment in Amazon (cloud storage provider).

Model

A model can come in many shapes, sizes, and styles. It is important to emphasize that a model is not the real world but merely a human construct to help us better understand real-world systems. The model illustrated in Figure 2 is deployed as a guide to successful implementation of this research work.

The model consists of the following components,
i. Flaw detection
ii. Cryptographic hash function (SHA 256)
iii. D-H key exchange algorithm

Flaw Detection

Penetration testing is an investigation conducted to provide stakeholders with information about the quality of the application under test. Penetration testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding flaws (errors or other defects). In other to detect the flaws in the existing application, an inbuilt pen testing tools on Kali Linux will be used.

Cryptographic Hash Function (SHA 256)

Cryptographic hash functions, also called message digests and one-way encryption, are algorithms that, in some sense, use no key. It is an algorithm that can be run on data such as an individual file or a password to produce a value called a checksum. The main use of a cryptographic hash function is to verify the authenticity of a piece of data. Two files can be assumed to be identical only if the checksums generated

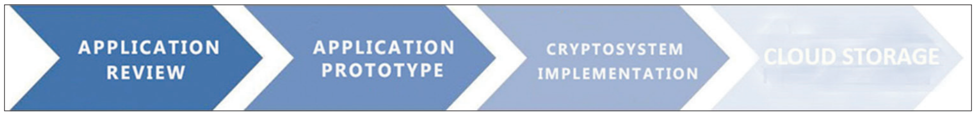


Figure 1: Diagram showing the research methodology

from each file, using the same cryptographic hash function, are identical. Some of the commonly used cryptographic hash functions include MD5, SHA-1, and SHA-2.

An SHA is considered to be cryptographically secure. The original data, once hashed by an SHA, typically cannot be reconstructed with a feasible amount of computing power.[37] SHA-256 generates an almost-unique 256-bit (32-byte) signature for a text. See below for the source code. Figure 3 shows the block diagram of Hash function. A hash function H accepts a variable-length block of data M as input and produces a fixed-size hash value h = H(M). In general terms, the principal object of a hash function is data integrity. A change to any bit or bits in results, with high probability, in a change to the hash code. Virtually, all cryptographic hash functions involve the iterative use of a compression function. The compression function used in SHAs falls into one of two categories: A function specifically designed for the hash function or an algorithm based on a symmetric block cipher. SHA and Whirlpool are examples of these two approaches, respectively. The hash algorithm involves repeated use of a compression function, f, that takes two inputs (an – bit input from the previous step, called the chaining variable and a – bit block) and produces an – bit output.

Secure Hash Algorithm (SHA) is a family of cryptographic hash functions. In pursuit of the study, SHA-256 will be used to secure user's access information. However, comparison of SHA Parameters is shown in the Table 1. All sizes are measured in bits.

D-H Key Exchange Algorithm

A simple public-key algorithm is D-H key exchange. This protocol enables two users to establish a secret key using a public-key scheme based on discrete logarithms. The protocol is secure only if the authenticity of the two participants can be established. D-H is used for secret-key key exchange only and not for authentication or digital signatures.

Algorithm is as follows:

i. Select two global public elements: A prime number p and an integer α that is a primitive root of p.

Table 1: Comparison of SHA parameters

Algorithm	Message digest size	Message Size	Block size	Word size	Number of step
SHA-1	160	$<2^{64}$	512	32	80
SHA-224	224	$<2^{64}$	512	32	64
SHA-256	256	$<2^{64}$	512	32	64
SHA-384	384	$<2^{12k}$	1024	64	80
SHA-512	512	$<2^{12k}$	1024	64	80

SHA: Secure hash algorithm

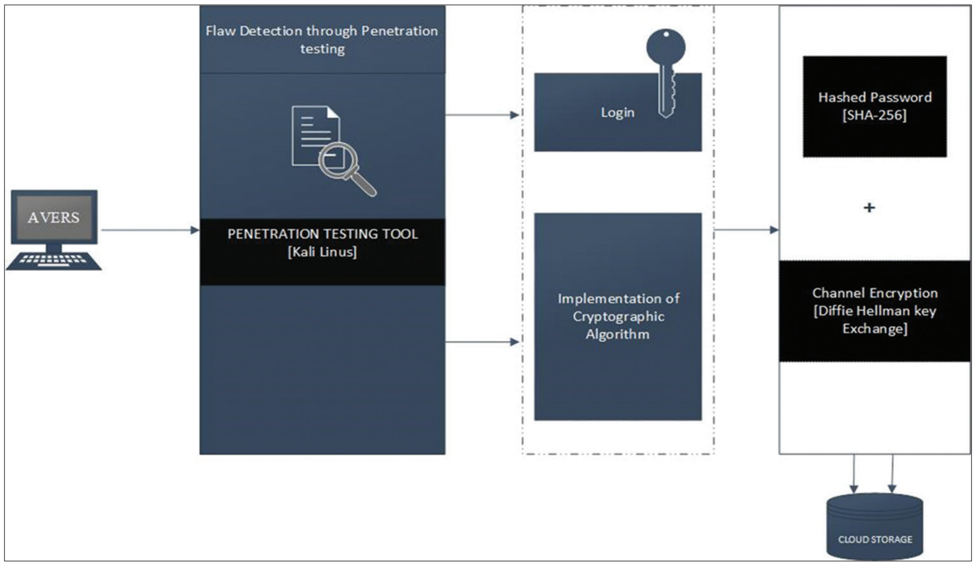

Flaw Detection through Penetration testing
AVERS
PENETRATION TESTING TOOL [Kali Linus]
Login
Implementation of Cryptographic Algorithm
Hashed Password [SHA-256]
+
Channel Encryption [Diffie Hellman key Exchange]
CLOUD STORAGE

Figure 2: Architectural model for the study

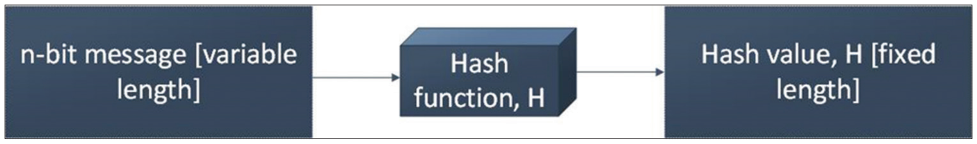


Figure 3: Block diagram of hash function

ii. Sender key generation: Sender selects a random integer XA < p which is private and computes YA = α XA mod p, which is public.
iii. Receiver key generation: Receiver selects a random integer XB < p which is private and computes YB = α XB mod p, which is public.
iv. Sender calculates secret key: K = (YB) XA mod p
v. Receiver calculates secret key which is identical to sender secret key. K = (YA) XB mod p.

Mathematical Model

1. The first i actions or test cases detect M_i defects and the testing process during the time interval (0, *t*) detects M (*t*) defects; that is,

$$M_i = \sum_{k=1}^{i} Z_k, \qquad with\, M_0 = 0;$$

$$M(t) = \sum_{k=1}^{H(t)} Z_k, \qquad with\, M(0) = 0, Z_0 = 0.$$

2. Upon a failure being revealed, the execution of the current test case terminates; at most one, failure causing defect is removed immediately from the software under test, and a new defect may or may be introduced; more specifically, it holds

$$N_k = \begin{cases} N_{k-1}, & with\ probability\, 1-p-q \\ N_{k-1}+1, & with\ probability\ p \\ N_{k-1}, & with\ probability\, q \end{cases}$$

$$0 \le p, q \le 1, 0 \le p+q \le 1.$$

However, N_k denotes the number of defects remaining in the software after the kth test

RESULTS AND DISCUSSION

Experimental Evaluation

The experimental work considers the password check of AAUA AVERS, testing with SHA256 and D-H key exchange algorithm.

Procedure for Evaluation

1. The password was encrypted with PBKDF2 and SHA256 hash
2. The hashed password was, however, secured over an unsecured communication channel with D-H key exchange algorithm.

Properties of the Component Evaluated

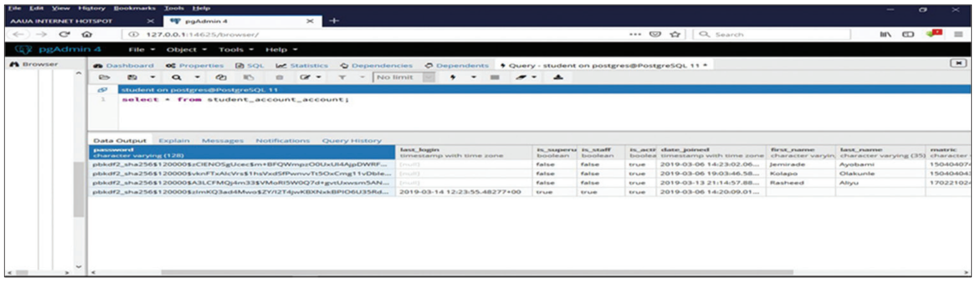
A cryptographic hash (sometimes called "digest") is a kind of "signature" for a text or a data file. SHA-256 generates an almost-unique 256-bit (32-byte) signature for a text. In particular, cryptographic hash functions exhibit three properties

1. They are "collision free." In simple words, no two input hashes should map to the same output hash
2. They can be hidden. In simple words, it should be difficult to guess the input value for a hash function from its output
3. They should be puzzle-friendly. That is to say, it should be difficult to select an input that provides a predefined output. Thus, the input should be selected from a distribution that is as wide as possible

Furthermore, the D-H key exchange is a secure method for exchanging cryptographic keys. This method allows two parties which have no prior knowledge of each other to establish a shared, secret key, even over an insecure channel. The concept uses multiplicative group of integers modulo, which without knowledge of the private keys of any of the parties, would present a mathematically overwhelming task to a code breaker. The general idea of the D-H key exchange involves two parties exchanging numbers and doing simple calculations to get a common number which serves as a secret key. Both parties may not know beforehand what the final secret number is, but after some calculations, both are left with a value that only they know about which they can use for various purposes like identification and as a secret key for other cryptographic methods.

Evaluation Result/Output

Figure 4 shows that adding Diffie Hellman Key exchange algorithm to a hashed password (with SHA 256) proves a more secured cryptography for preserving and protecting password in the cloud storage.



```
select * from student_account_account;
```

password character varying (128)	last_login timestamp with time zone	is_superu boolean	is_staff boolean	is_acti boolea	date_joined timestamp with time zone	first_name character varyin	last_name character varying (35)	matric character
pbkdf2_sha256$120000$zClENOSgUcec$m+BFQWmpzO0UxUl4AjpDWRF...	[null]	false	false	true	2019-03-06 14:23:02.06...	Jemirade	Ayobami	15040407
pbkdf2_sha256$120000$vknFTxAlcVrs$1hsVxdSfPwnvvTt5OxCmg11vDble...	[null]	false	false	true	2019-03-06 19:03:46.58...	Kolapo	Olakunle	15040404
pbkdf2_sha256$120000$A3LCFMQj4m33$VMoRI5W0Q7d+gvtUxwsm5AN...	[null]	false	false	true	2019-03-13 21:14:57.88...	Rasheed	Aliyu	17022102
pbkdf2_sha256$120000$zlmKQ3ad4Mwo$ZY/I2T4jwKBXNxkBPIO6U35Rd...	2019-03-14 12:23:55.48277+00	true	true	true	2019-03-06 14:20:09.01...			

Figure 4: Evaluation output

CONCLUSION

Password management is a set of principles and best practices to be followed by users while storing and managing passwords in an efficient manner to secure passwords as much as they can to prevent unauthorized access. Finding shows the reluctance of users in trusting an online password manager. In essence, password managers are a great double-whammy: not only do they drastically increase your security, but they also simplify your life.

This study in its own view, is set to design a highly secure cryptography (i.e. a zero-knowledge protocol) for making the site /service more tightly secured. Consequently, this work presented an enhanced web security for cloud-based password management using SHA-256 and Diffie Hellman key exchange algorithm. The methodology and model of the system were expatiated upon. The system structure and the visible activities that take place within the system were also presented using appropriate UML design. For the implementation, OWASP ZAP, a penetration tool on Kali LinuX was used to penetrate the existing application, Html, CSS and Bootstrap was used for the Frontend Development of the prototype application, Python/Django was used for the interaction with the server, and PostgreSQL for the database.

The result of the evaluation carried out shows that Django endeavors to provide a secure and flexible set of tools for managing user passwords.

Future Direction

The designed architecture presented in this study is simplified such that it can easily be modified to enable adaptation and application to other research domains such as mobile application and desktop application.

REFERENCES

1. Livshits B, Lam M. Finding Security Vulnerabilities in Java Applications with Static Analysis, Proceedings of the 14th Conference on USENIX Security Symposium, 14, 2005. Available from: http://www.portal.acm.org. [Last accessed on 2018 Jun 15].
2. Ajayi OO, Fanala TF. Empirical evaluation of data hashing algorithms for password checks in PHP webapps using salt and pepper. Comput Inf Syst Dev Inf Allied Res J 2017;8:21-8.
3. Alanazi F, Sarrab M. The history of web application security risks. Int J Comput Sci Inf Secur 2011;9:40-7.
4. Higgins K. Stolen Passwords Used In Most Data Breaches; 2014. from http://www.darkreading.com/stolen-passwords-used-in-most-data-breaches/d/d-id/1204615. [Last accessed on 2018 Jun 15].
5. Rubenking N. Survey: Hardly Anybody Uses a Password Manager; 2015. Available from: http://www.pcmag.com/article2/0,2817,2407168,00.asp. [Last accessed on 2018 Jun 15].
6. Adams A, Sasse MA. Users are not the enemy. Commun ACM 1999;42:40-6.
7. Bonneau J, Herley C, Oorschot PC, Stajano F. The quest to replace passwords: A Framework for Comparative Evaluation of Web Authentication schemes. In Proceedings Of IEEE Symp. On Security and Privacy; 2012.
8. Ion L, Reeder R, Consolvo S. No one can hack my mind: Comparing Expert and Non-Expert Security Practices; 2015. Available from: https://www.usenix.org/system/files/conference/soups2015/soups15- paper-ion.pdf. [Last accessed on 2015 Dec 13].
9. Ge X, Paige RF, Polack FA, Chivers H, Brooke PJ. Agile Development of Secure Web Applications. Proceedings of the 6th International Conference on Web Engineering. Palo Alto: ICWE; 2006. p. 305-12.
10. Norwawi NM, Selamat MH. Secure e-commerce web development framework. Inf Technol J 2011;10:769-78.
11. McGraw G, Viega J. Building Secure Software. In RTO/NATO Real-Time Intrusion Detection Symp; 2002.
12. Scott R, Kent S. In Proceedings of New Security Paradigm Workshop'17, Islamorada, Florida, USA: 2017.
13. Li Z, He W, Akhawe D, Song D. The Emperor's New Password Manager: Security Analysis of Web-based Password Managers. In: USENIX Security Symposium. Berkeley: USENIX; 2014. p. 465-79. Available from: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-li-zhiwei.pdf. [Last accessed on 2018 Jun 15].
14. Silver D, Jana S, Boneh D, Chen E, Jackson C. Password Managers: Attacks and Defenses, (Usenix Security) 2014. Available from: https://www.crypto.stanford.edu/dabo/pubs/abstracts/pwdmgrBrowser.html. [Last accessed on 2018 Jun 20].
15. Herley C, Van Oorschot PC. A research agenda acknowledging the persistence of passwords. IEEE Secur Priv 2012;10:28-36.
16. Komanduri S, Shay R, Kelley PG, Mazurek ML, Bauer L, Christin N, *et al*. Of Passwords and People: Measuring the Effect of Password-Composition Policies. In Proceedings of CHI; 2011.
17. Jakobsson M, Myers S. Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. New York, NY, USA: Wiley-Interscience; 2006.
18. Rachna D, Tygar JD, Marti H. Why Phishing Works. In Proceedings of CHI; 2006.
19. Yue C. Preventing the Revealing of Online Passwords to Inappropriate Websites with Login Inspector. In Proceedings of USENIX LISA; 2012.
20. Stone-Gross B, Cova M, Cavallaro L, Gilbert B, Szydlowski M, Kemmerer RA, *et al*. Your Botnet is my Botnet: Analysis of a Botnet Takeover. In: Proceedings of Conference on Computer and Communications; 2009.
21. Popa RA, Lorch J, Molnar D, Wang HJ, Zhuang L. Enabling Security in Cloud Storage SLAs with Cloud Proof. In Proceedings of USENIX ATC; 2011.
22. Stefano C, Riccardo F, Marco S, Mauro T. Surviving the web: A journey into web session security. ACM Comput Surv 2017;5:1-34.
23. Rui Z, Chuan Y. In CODASPY'13, Proceedings of the third ACM Conference on Data and Application Security and Privacy. San Antonio, Texas, USA: 2013.
24. Chiasson S, Van Oorschot PC, Biddle R. A Usability Study and Critique of Two Password Managers. In 15th USENIX Security Symposium. Vancouver, Canada: 2006. p. 1-16.

25. Ambarish K, Nitesh S, Nicolas C. A comparative usability evaluation of traditional password managers. In: International Conference on Information Security and Cryptology. Heidelberg: Springer; 2010. p. 233-51.
26. Provos N, Mazieres, D. A Future-Adaptable Password Scheme. Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference, Monterey, California: 1999.
27. Colin P. Stronger Key Derivation Via Sequential Memory-Hard Functions Conference; 2009.
28. Thulasimani L, Madheswaran M. A novel secure hash algorithm for public key-digital signature schemes. Int Arab J Inf Technol 2012;9:262-7.
29. Richa P, Upenda M, Abhay B. Design and analysis of a new hash algorithm with key integration. Int J Comput Appl 2013;81:33-8.
30. Selva R, Anuja P. Secured password management technique using One-Time-Protocol (OTP) in smartphone. Int J Comput Sci Mob Comput 2014;3:976-81.
31. Disha S. Digital security using cryptographic message digest algorithm. Int J Acad Res Comput Sci Manage Stud 2015;3:215-9.
32. Sahni N. A review on cryptographic hashing algorithms for message authentication. Int J Comput Appl 2015;120:29-32.
33. Sriramya P, Karthika RA. Providing password security by salted password hashing using B crypt algorithm. ARPN J Eng Appl Sci 2015;10:5551-6.
34. Tivkaa M, Choji D, Agaji I, Atsaam D. An enhanced password-username authentication system using cryptographic hashing and recognition based graphical password. IOSR J Comput Eng 2016;18:54-8. Available from: http://www.iosrjournals.org. [Last accessed on 2018 Jul 10].
35. Rituraj G, Rekha V, Amanpreet K. An effectual hybrid approach using data encryption standard (DES) and secured hash algorithm (SHA) for image steganography. Int J Recent Innov Trends Comput Commun 2018;6:44-53.
36. Sanjeev B, Sai K, Ajay C, Aditya C. Secure transfer of university question paper using image steganography. IOSR J Eng 2018;6:89-92.
37. Madhuravani B, Murthy DS. Cryptographic hash functions: SHA family. Int J Innov Technol Explor Eng 2013;2:326-9.