Journal of the Nigerian Association of Mathematical Physics
Volume 47 (*July, 2018 Issue*), pp193 – 198

DESIGN AND IMPLEMENTATION OF A WEB BASED DATA PROTECTION SYSTEM FOR ESITM USEN, USING A COMPLEX KEY (ALPHANUMERIC) SCHEME AND HASHING TECHNIQUES

Ihama E.I., Izogie L. E. and Iyamu Iziegbe

Department of Computer Science and Information Technology, School of Applied Sciences, Edo State Polytechnic, Usen P.M.B 1104, Benin City, Nigeria.

Abstract

The research paper was designed and implemented using a complex key scheme approach and hashing techniques. The software was developed to help ensure that intruders into computers systems files and computer users involved in carrying out malicious acts on information system are prevented from gaining unauthorized access to the system. The system was developed with the right specification and authentication checks in place, the system provides security facilities for information system users. When information is properly managed and secured, it gives information system users the confidence to process and store their information in the computer system, and also while been transmitted via the internet. The system was designed using these approaches, the front-end was developed with PHP and Microsoft Visual Studio.net was integrated in development environment and Microsoft SQL Server was used at the database backend.

***Keywords*:** Hashing, complex key, authentication, secure, malicious, data

1.0 Introduction

For a very long time, different people have worked on how to protect information systems and computer security, if not more than fifty years or more. There is no accurate data about the cost of failures in computer security. On the other hand, most of them are never made public for fear of embarrassment. Of course, computer security is not just about computer systems. Like any security, it is only as strong as its weakest link, and the links include the people and the physical security of the system. Very often the easiest way to break into a system is to bribe an insider.

With computers, on the other hand, security is only a matter of software, which is cheap to manufacture, never wears out, and cannot be attacked with drills or explosives. This makes it easy to drift into thinking that computer security can be perfect, or nearly so. The fact that work on computer security has been dominated by the needs of national security has made this problem worse. In this context the stakes are much higher and there are no police or courts available to punish attackers, so it is more important not to make mistakes. Furthermore, computer security has been regarded as an offshoot of communication security, which is based on cryptography. Since cryptography can be nearly perfect, it is natural to think that computer security can be as well.

Cryptography is the science of writing in secret code and using mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient.

Recent work [1], show how to choose constants for a modified version of SHA-1 enabling a saboteur to help an attacker later find collisions of a certain form. Both of these examples are also examples of choosing weak constants, a strategy discussed below. With this weakness, exploitability hinges on knowledge of the cryptanalytic attack. While at first glance, this may seem to allow secrecy. History shows that other researchers often reproduce knowledge first developed behind closed doors.

A research work in [2] explore SETUP attacks for case of symmetric encryption, and rebrand SETUP attacks as algorithmic substitution attacks (ASAs). They give SETUP attacks against symmetric encryption, but also seek countermeasures, in

Corresponding Author**:** Ihama E.I., Email: eyoski@yahoo.com, Tel: +2347039404855

particular arguing that a symmetric encryption scheme is secure against sabotage if no attacker, even given a secret trapdoor, can distinguish between cipher texts generated by the trusted reference algorithm versus ones generated by a back doored version of it. They argue that deterministic, stateful schemes can be shown to meet this latter notion. However, it is important to note that this result is only meaningful if the algorithm underlying the reference implementation is assumed to be free of backdoors. More generally, the SETUP and ASA frameworks do not capture all relevant aspects of a cryptographic weakness, nor potential routes for a saboteur to achieve one.

Ease of use characterizes the computational and logistical difficulty of mounting an attack using the weakness. For example, the Dual EC backdoor can be tricky to exploit in certain situations [3]. There have been a string of high profile vulnerabilities of TLS certificate checking: Apple's double goto bug in [4], a bug in Open SSL [5], and the results of a number of research papers showing certificate checking bugs in numerous applications [6]. While we have no reason to expect that any of these bugs were maliciously inserted, similar bugs maliciously inserted would be devastating examples of sabotage.

Frameworks for understanding white box design weaknesses. Thus far the only formal frameworks aimed at understanding backdoors has been in substitution attack settings[7], the saboteur arranges for a subverted algorithm to replace a correct one. These attacks are assumed to be black-box, meaning defenders only have API access to the algorithm (whether correct or subverted). Yet, many of the confirmed examples of sabotage are directly built into public designs (e.g., EC DRBG). In these settings substitution attacks are inapplicable. We might refer to these as `whitebox design weaknesses' since they withstand knowledge and scrutiny of the cryptosystem design.

Robustness in provable-security design. Much of modern cryptographic design has focused on security definitions and proofs that constructions achieve them. This is a powerful approach, but does not always account for robustness when security definitions are narrow. The goal of semantic security for encryption provides, along one dimension, broad guarantees that nothing is leaked about plaintexts. But schemes proven secure relative to semantic security may suffer from padding oracle attacks [8], various side-channels [9], randomness failures, and implementation faults. In each case, attackers sidestep the strengths of achieving semantic security by taking advantage of something outside the model. That a model fails to account for all attack vectors is not really surprising, since it must be simple enough to admit formal reasoning.

Another issue is that provable security can in some cases encourage designs that are amenable to sabotage. Take for example the long line of work on building cryptographic protocols from a setup assumption such as common reference strings (CRS) [10]. This is seen as a way to increase security and simplicity of protocols, by avoiding more complex setup assumptions or the random oracle model. To support proofs, the CRS must often be generated in such a way that whoever chooses it can retain a trapdoor secret (analogously to the EC DRBG parameters). The resulting cryptographic protocol is ready-made for sabotage.

To deal with these issues, we advocate robustness as a general principle of provable-security cryptographic design. By this we mean that a scheme proposed for deployment should be (formally) analyzed relative to many security models, including orthogonal ones. By focusing on multiple models, one can keep each relatively simple, and yet provide more coverage of threats. Examples in this spirit can be found in [11]. In addition, theoreticians should evaluate each model with regards to its ability to prevent sabotage. Does it rely on subvertible global parameters? How many implicit assumptions are built in? What happens to security if one violates these assumptions? A scheme that relies on getting too many things `right' is worse than one that has fewer dependencies.

New cryptographic standardization processes. We need to improve our understanding of, and methodologies for, designing cryptographic standards. For widely used higher-level cryptographic protocols (e.g., TLS, WiMax, IPsec), we end up using standards that are designed by public committees such as those formed by the IETF. While we do not want to denigrate the hard work and expertise of those involved, it's clear that there are limitations to this approach and the results can sometimes be disappointing. Both TLS and IPsec are examples of painfully complex and difficult-to-implement-correctly standards [12]. There are many other examples of broken standards, and likely many others for which analysis would reveal problems.

We advocate research and experimentation on new design approaches for cryptographic standards. One approach that has seemingly worked well for low-level primitives is that of public design competitions such as those conducted by NIST. This forces public review, allows a number of small design teams unfettered during design by large committees, and ultimately appears to have yielded strong primitives such as AES and SHA-3. Whether this can work for the more nuanced setting of higher-level cryptographic systems is an interesting open question. A key challenge we foresee will be specifying sound requirements. Even with committee designs, we might begin including in existing standardization processes an explicit review step for resilience to sabotage (perhaps helped by using the outcomes of the hoped-for models).

Software engineering for cryptography. Implementing even good standards securely remains a monumental challenge. While some notable work has been done on implementation security, such as verified cryptographic implementations [13] and other tools for finding (benign) bugs or well-known problems [14], there seems too little relative to its importance.

We advocate building such a community. First steps might include workshops bringing together those who have done work in the area and other interested parties. It may also be worthwhile to engage funding agencies such as NSF, DARPA, the ERC, and others about introducing programs aimed at encouraging research in this area. We might also initiate educational

programs like the underhanded C contest [15] that would task participants to backdoor a reference piece of cryptographic code.

In the nearer term, we also relay the following pragmatic suggestions from [16]. First, vendors should make their encryption code public, including the specifications of protocols. This will allow others to examine the code for vulnerabilities. While it is true we won't know for sure if the code we're seeing is the code that's actually used in the application, it nevertheless forces saboteurs to surreptitiously substitute implementations.

This raises the bar and forces the owner of the system to outright lie about what implementation is being used. All this increases the number of people required for the sabotage conspiracy to work. The community should target creating independent compatible versions of cryptographic systems. This helps check that any individual one is operating properly.

Usability and deplorability of cryptography.

It is somewhat obvious, but still worthwhile, to say that we need better metrics and design principles for usability of cryptographic tools. This encompasses not only end-user issues, such as those covered in seminal studies on email encryption by [17], but also deployability for administrators and others. Setting up IPsec and TLS, for example, can be a hurdle too high for many. Like with software engineering for cryptography, we need a research community that focuses on ensuring software can be deployed and used.

The Encryption Process

The diagram below shows the encryption and decryption process using cryptography. The Readable text R is encrypted with an encryption algorithm system and an encryption key F. The resulting ciphertext,(C), is transmitted over the network. The receiver decrypts the ciphertext(C) with a decryption algorithm system and a decryption key G. The encryption and decryption algorithms are public information. However, at least one of the keys (F and G) are private information. The keys consist of a relatively short string of bytes (e.g., 128 bits). The longer the key, the more difficult to break the cipher.

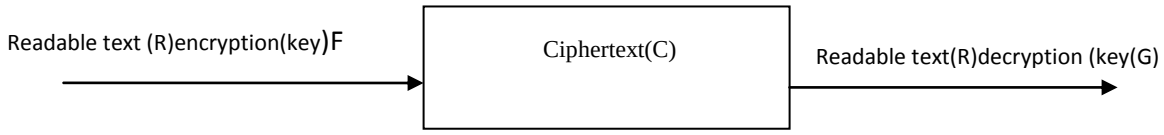

Fig. 1.The Process of Encryption.

This cryptography software is both online and offline based systems. That is they are applicable or useful in the protection of Internet files or systems within a network, and also protect an information systems both in storage and when transmitted across an unsecured network like the internet.

The Hashing Process

Hash functions, also called message digests and one-way encryption, are algorithms that, in some sense, use no key , Instead, a fixed-length hash value is computed based upon the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered. Hash algorithms are typically used to provide a digital fingerprint of a file's contents, often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords. Hash functions, then, provide a measure of the integrity of a file.

The key in public-key encryption is based on a hash value. This is a value that is computed from a base input number using a hashing algorithm. Essentially, the hash value is a summary of the original value. The important thing about a hash value is that it is nearly impossible to derive the original input number without knowing the data used to create the hash value.

Table 1.Hashing Process

Input number	**Hashing algorithm**	**Hash value**
10,667	**Input # x 143**	**1,525,381**

You can see how hard it would be to determine that the value 1,525,381 came from the multiplication of 10,667 and 143 from table 1 above. But if you knew that the multiplier was 143, then it would be very easy to calculate the value 10,667. Public-key encryption is actually much more complex than this example, but that's the basic idea.

The important thing about a hash value is that it is nearly impossible to derive the original input number without knowing the data used to create the hash value. In order to ensure that encrypted information are not easily decrypted by cryptanalyst, the proposed system will be using an N key system, which makes the system quite unique.

The system is user friendly. It designed to provides features, which provides users with input screen such that a user can enter his/her access code to login to the system. The system generally provides the following facilities:

i.) Provide facilities for storing user information;
ii.) Provide facilities for encrypting user information using N key system of 64-bits each.
iii.) Provide facilities for keeping store of information being encrypted within the system for future reference.
iv.) Provide facilities for protecting information from unauthorized or accidental discloser while the information is in transit (either electronically or physically) and while information is in storage.

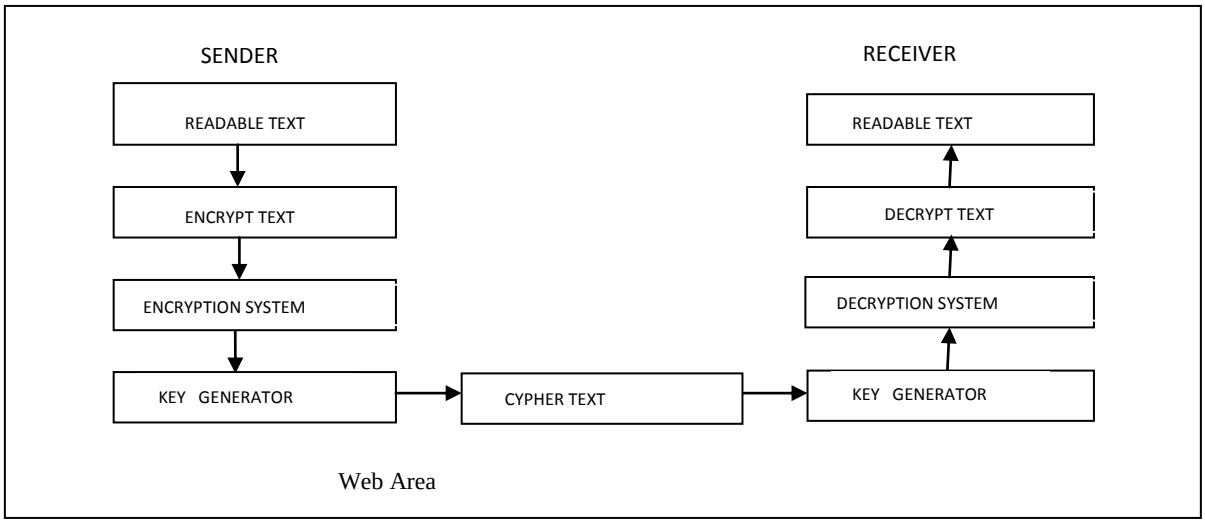

Fig2: The Architecture of the Data Security System Using Cryptography Methods

The Design Approach of the data protection system

The architecture above ensures that a sender uses an encryption system to encrypt their data(information).

The Encryption Process: in the encryption process the readabletext is encrypted by the sender, by supplying the system with the key(alphanumeric), the processing task of the encryption process, is a combination of transposition and substitution ciphers algorithms is adopted for the processing task of encryption of the users information (data), which transposes the readabletext into a ciphertext by the encryption system and the key generator. This will be done by supplying N (multiple) key/s of 256bits. These keys are used by the encryption system to encrypt the sender's information(data), this will render it into an unreadable form known as ciphertext. The Cyphertext: this is a transposed plaintext, using the encryption system and the key generator (which is a value that is computed from a base input number using a hashing algorithm)The Decryption Process: the ciphertext is transpose back into a readabletext, by the receiver, by supplying the key generator and the decryption system, which also uses a combination of transposition and substitution ciphers algorithms as used exactly by the sender, in other for the ciphertext to be transform back to an understandable format. (this is essentially a hash value, which is a summary of the original value).

The important thing about this system is that it uses a hash value system, which is nearly impossible to derived, without knowing the original input number of the data use to create the hash value, this ensure that encrypted information are not easily decrypted by cryptanalyst.

The system user interface: was designed with features, which provides users with input screen such that a user can enter his/her access code to login to the system. The user interface uses this approach, the front-end was developed with PHP and Microsoft Visual Studio.net was integrated in development environment while Microsoft SQL Server was used at the database backend.

Conclusion

Information system is a veritable tool to any organization, and due to advancement in technology, malicious users have developed more sophisticated means to gain access to unauthorized systems files. A strong data protection scheme system was designed and implemented for ESITM to prevent authorized and malicious attack from unauthorized users from gaining access to our systems files and information systems, which can be costly to the institute and other information systems users, this systems provides both online and offline protection measures, by visiting this site: www.esitmdencrypt.com.ng to download the software.

We encourage all information systems users in the Institute to user this software, in order to protect their files and information systems.

Software Applications

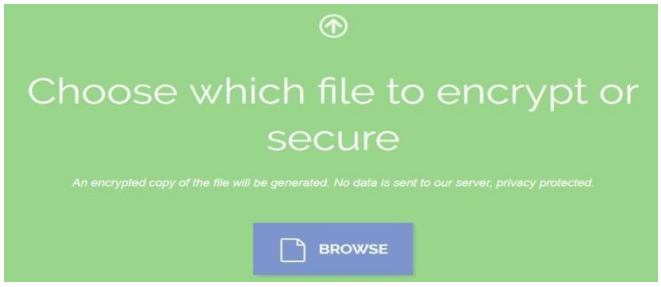

Fig3. Encryption process

Fig 4. Decryption process

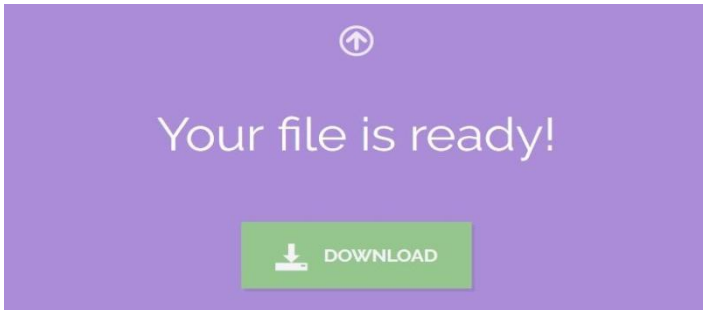

Fig 5. Downloading process

Fig 6. Encrypting your file

Fig 7. Decrypting your file

Reference

[1] A. Albertini, J. P. Aumasson, M. Eichlseder, F. Mendel, and M. Schl□a_er. Malicious Hashing: Eves Variant of SHA-1. In Selected Areas in Cryptography{SAC, 2014.

[2] MihirBellare, Kenneth G Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Advances in Cryptology{CRYPTO, pages 1{19. Springer, 2014.

[3] Stephen Checkoway, Matthew Fredrikson, Ruben Niederhagen, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, and HovavSchacham. On the practical exploitability of Dual EC in TLS implementations. In USENIX Security Symposium, 2014.

[4] NIST National Vulnerabilities Database. Vulnerability summary for CVE-2014-1266, 2014. Availableat http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-1266.

[5] NIST National Vulnerabilities Database. Vulnerability summary for CVE-2014-0224, 2014. Available at http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0224.

[6] Tyler Nichols, Joe Pletcher, Braden Hollembaek, Adam Bates, Dave Tian, Abdulrahman Alkhelai_, and Kevin Butler. CertShim: Securing SSL certi_cateveri_cation through dynamic linking. In ACM Conference on Computer and Communications Security { CCS. ACM, 2014.

[7] MihirBellare, Kenneth G Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Advances in Cryptology{CRYPTO, pages 1{19. Springer, 2014.

[8] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryptionstandard PKCS#1. In Advances in Cryptology{CRYPTO, pages 1{12. Springer, 1998.

[9] Dag Arne Osvik, Adi Shamir, and EranTromer. Cache attacks and countermeasures: the case of AES.In Topics in Cryptology{CT-RSA, pages 1{20. Springer, 2006.

[10] Ivan Damg_ard. E_cient concurrent zero-knowledge in the auxiliary string model. In Advances in Cryptology{EUROCRYPT, pages 418{430. Springer, 2000.

[11] MihirBellare, ZvikaBrakerski, MoniNaor, Thomas Ristenpart, Gil Segev, HovavShacham, andScott Yilek. Hedged public-key encryption: How to protect against bad randomness. In Advancesin Cryptology{ASIACRYPT, pages 232{249. Springer, 2009.

[12] Niels Ferguson and Bruce Schneier.A cryptographic evaluation of IPsec. 2003. Available at https://www.schneier.com/paper-ipsec.html.

[13] JoseBacelar Almeida, Manuel Barbosa, Gilles Barthe, and FrancoisDupressoir.Certified computeraided cryptography: Efficient provably secure machine code from high-level implementations. In ACMConference on Computer and Communications Security{CCS, pages 1217{1230. ACM, 2013.

[14] Lin-Shung Huang, Alex Rice, ErlingEllingsen, and Collin Jackson. Analyzing forged SSL certificatesin the wild. In IEEE Symposium on Security and Privacy.IEEE, 2014.

[15] Scott Craver. The underhanded C contest.http://underhanded.xcott.com/.22

[16] Bruce Schneier. How to Design|And Defend Against|The Perfect Security Backdoor. https://www.schneier.com/essays/archives/2013/10/how_to_design_and_de.html, 2013.

[17] Alma Whitten and J. Doug Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In USENIX Security Symposium, volume 1999, 1999.