Journal of the Nigerian Association of Mathematical Physics
Volume **19 (November, 2011), pp 487 – 492**
© *J. of NAMP*

A Key Generation Model for Improving the Security of Cryptographic Keys

Annie. O. Egwali
Department of Computer Science,
University of Benin, P.O. Box 1154, Benin City. Nigeria.
egwali.annie@yahoo.com; 07033247730

Abstract

Cryptography is a mathematical technique that plays an important role in information security techniques for addressing authentication, interactive proofs, data origination, sender/receiver identity, non-repudiation, secure computation, data integrity and confidentiality, message integrity checking and digital signatures. In public key cryptography, the security of private keys is very importance, for if ever compromised, it can be used to decrypt secret messages. Conventional methods that use textual passwords, graphical passwords and single modal biometric systems that are used to encryption and protect private keys do not provide adequate security due to uses password practices and the very low entropy of chosen passwords. To improve the security of private keys, we propose a novel multifactor key generation algorithm that dynamically regenerate the private key of a user with a significant high overhead that exceeds what an identity attacker can contend with, that meet the current security requirements of any public key algorithm. The proposed encryption algorithm proved to be more secured at protecting private keys than using textual password-based techniques.

1.0 Introduction

The goal of cryptography and encryption system is to embed a secret with sensitive data in a way that can only be decrypted with the right private key from the legitimate user. Because of the large size of a cryptographically strong key, it is not possible for a user to remember the private key and enter each time it is required for a cryptographic application. Instead, the private key is usually stored encrypted with a user chosen password. Under usability studies however, findings from earlier studies on users' textual password selection, revealed that users choose poor or weak textual passwords [1, 2]. Using a weak password transforms the cryptographic algorithm into a weak one.
From research studies it is established that a typical password should be 8-bit ASCII characters (256) or 64 bits, this generates a key length of size $64^8 = 2.8\text{x}10^{14}$ [3, 4]. But even this is not as good as it might appear because the 128 possible combinations of 8-bit per character are not equally likely; users usually do not use control characters, non-alphanumeric characters or password with high entropy. For instance, a textual password of length n from a character set of size c, will have a key length of size $k_p = c^n$ [5]. Users usually utilize passwords that are 3 to 6 characters long [6]. By increasing the length of the password to 8 digits, a key length of $10^8 = 100000000$ is obtained, whereas keeping the same length but increasing the allowed input characters to include all lowercase alphabetic characters and digits yields a key length of $(26+10)^8 = 2821109907456$, which is larger.
To enhance the security of cryptographic keys, other approaches exist which includes the use of biometric and graphical passwords. However, a drawback of biometric-based password and graphical password is the length of the keys that can be generated. Monrose et al [7] proposed a cryptographic key from voice characteristics of a user, which generated a 46-bit key length generated from a roughly two second spoken password. Feng and Wah [8] posited the idea of using on-line handwritten signatures for private key generation which can be used to generate a 160-bit key. Fabian et al [9] propose the use of voice characteristics for generating a cryptographic key with a key size of 46-bit key. Hao and Chan [10] proposed an on-line handwritten signature for private key generation which efficiently generates a 160-bit key. Uludag et al [11] convert fingerprint templates (minutiae data) into point lists in 2D space, which implicitly hide a given secret key of 128-bit length.
Graphical password generated keys is sometimes lower [12] than that of textual passwords. For example [13] yield $6561 \approx 2^{13}$ passwords. Déjà Vu [5, 14] model yields a key size of N!/K!(N-K)! (Where N denotes total set of images and K subset for authentication) which yields a $53130 \approx 2^{16}$ key size with a search time less than 0.5 seconds [15]. Face model [16] has a

Corresponding authors: E-mail: egwali@yahoo.com, Tel. +2347033247730

key size of $9^4 = 6.5 \times 10^3$. Man et al [17] password generated a key size of $400^5 \approx 1.0\mathrm{x}10^{13}$, which takes a little more than 40.5 days search time to compromise. However, [18] generated a key size of $\binom{N}{M}$, where N = 80 and N = 30, which yields a key size of 2^{73}, with a search time more than 570, 776 years [15].
All the aforementioned techniques have a limited key length, with 2^{73} bits being the largest key produced. However, in public key algorithms higher key lengths that exceed 2^{73} bits are required to keep pace with current computational power.

2.0 Proposed Model

Using the concept of textual passwords, graphical passwords and fingerprint biometric models, we developed a multifactor key generation algorithm (MKGA) and concentrated mainly on improving the different factors performance. Regenerating private keys involves a registration and key regeneration phase.

2.1 Registration Phase

The fingerprint image registration is carried out using a feature extractor and encryption software driver designed by [19], which uses the orientation field of the ridge map of the fingerprint as a biometric feature denoted as $\boldsymbol{O_r}$. The system interacts with FingerAuth, an Add-on Extension for Mozilla Firefox to derive minataeu data, ***Mui*** from the registered finger using the software driver. We denote a legal user by *Us*, user's full name by *Fullname*, the selected graphical password codes by *Gpas*, the existing private key by *FINkey*, user's fingerprint by F_{in}, the system by *Sys*, database by *Dbas*, displayed fingerprint window by FIN_{WIN}, finger by *F* and fingerprint sensor device by *Io*. These are the steps in the registration process:

(i) *Us* enters full name *Fullname* and not less than ten diagonally selected graphical image codes *Gpas* and submits to *Sys*.

(ii) *Sys* then
- a. stores *Fullname* and *Gpas* in the database *Dbas*.
- b. displays the fingerprint windows FIN_{WIN} to retrieve user's fingerptint data via an input device vector *Io*.

(iii) ***Us*** presents finger ***F*** to the input vector ***Io***

(iv) ***Sys*** then
- a. interacts with FingerAuth, an Add-on Extension for Mozilla Firefox to derive $\boldsymbol{F_{in}}$
- b. acquire ***FINkey*** using the algorithm proposed in [19]**.**
- c. compute ***Mui*** and $\boldsymbol{O_r}$ from $\boldsymbol{F_{in}}$, where:

$\boldsymbol{O_r} = \{\theta_r : 1 \le \theta_r \le 180$ degrees and $1 \le r \le 180\}$.

- d. store ***Mui*** in $\mathbf{Dbas}_{Mui}$.
- e. for each $\theta_r \in \boldsymbol{O_r}$, define θ_{rs}, where $W = r = 180$ and form set P*i* ($1 \le i \le 41$) consisting of seven successive θ_{rs}.

$\theta_{rs} = \{\theta_r, \theta_r \pm q\}_{1 \le q \le b}, 1 \le v \le (2b + 1)$ (1)

where *b* is the system parameter defining the number of shadow angles, which accommodate *Sys* tolerance to errors.

- f. define the set *T* containing the T_i segments of ***FINkey*** i.e. $T = \{T_1, T_2, ..., T_{41}\}$, where every consecutive $T_i \in T$ is taken as a secret and an (*r*-*d*) secret sharing system where *r* is the minimum number of shares required to retrieve the secret and *d* is the total number of shares. For our experiment, *r*=2 and *d*=5 x (2b+1).
- g. compute random, collision free hashes $Coll_{fr}$ using the following equation:

$Hash_{com} = Hash_1(\theta_{rs} + Hash_1(Gpas + \mathrm{z}))$ (2)

$Coll_{fr} = f_o(Hash_{com}, u, \S)$ (3)

where $Hash_{com}$ generates 160-bit hashes (see equation 2), $Hash_1$ is a function that uses the SHA-1 algorithm to derive 160-bit random unique hashes, z is a public value selected arbitrary for every *W*, $Coll_{fr}$ maps the 160-bit hashes generated into smaller collision free hashes, which are used to index the secret shares. *u* defines the maximum range of $Coll_{fr}$ and *§* is a random seed value chosen for every *W* to forestall collision in the values of $Coll_{fr}$.

- h. For each *FINkey* segment, store the secret shares generated in step (e) above in Dbas_{ec} using the $Coll_{fr}$ equivalent index pointer. Thus for *d* shares of T_1, the first 5 x (2*b*+1*)* $Coll_{fr}$ values will be used where, $1 \le W \le 5$ and $1 \le v \le (2b + 1)$.
- i. Encrypt *u* values,
- j. Encrypt Dbas_{Mui} and Dbas_{ec} with *Gpas* to derive EDbas_{Mui} and EDbas_{ec} respectively and store.

2.2 Key Regeneration Phase

Only legitimate users can regenerate private keys. All ten fingers of a user can be regenerated as a private key (i.e $\boldsymbol{FINkey_1}$, ... , $\boldsymbol{FINkey_{10}}$) and can be used with the public key for authentication. The steps in key regeneration are as follows:

(i) *Us* enters *Fullname* and not less than ten diagonally selected graphical image codes *Gpas* to *Sys*.

(ii) *Us* submits *Fullname* and *Gpas* to *Sys* by clicking the 'Register' command button.

(iii) *Sys* then

a. stores *Fullname* and *Gpas* in *Dbas*.

b. displays the fingerprint windows $\boldsymbol{FIN_{WIN}}$ to retrieve user's fingerptint data via an input device vector ***Io***.

(iv) ***Us*** presents finger ***F*** to the input vector ***Io***

(v) ***Sys*** then

a. interacts with FingerAuth, an Add-on Extension for Mozilla Firefox to derive $\boldsymbol{F_{in}}$ and ***Mui*** using the algorithm proposed in [19] to acquire ***FINkey*****.**

b. reads from the files EDbas_{Mui} and EDbas_{ec}

k. decrypt EDbas_{Mui} with *Gpas*

l. compute $\boldsymbol{O_r}$ from $\boldsymbol{F_{in}}$, where:

$\boldsymbol{O_g} = \{\theta_r: 1 \le \theta_r \le 180$ degrees and $1 \le m \le 180\}$.

m. for each $\theta_r \in \boldsymbol{O_r}$, define θ_{gs}, where $W = r = 180$ and form groups P*i* ($1 \le i \le 36$) consisting of five successive θ_{rs}.

$$\theta_{gs} = \{\theta_r - 1, \theta_r, \theta_r + 1\}, 1 \le x \le 3 \qquad (4)$$

n. calculate $Hash_{reg} = Hash_1\,(\theta_s + Hash_1(Gpas + z))$ (5)

$$Coll_{reg} = f_0\,(Hash_{regN}, u, \S) \qquad (6)$$

o. compute the segments of *FINkey* using polynomial interpolation. Each segment secret is considered valid if after calculating three secrets, any two secrets calculated within θ_{gs} matches and produces the same secret value. For reg = val, this is expressed as:

$Hash_{regN} \in Hash_{valN}$

$Coll_{regN} \in Coll_{valN}$

The regenerated private key of U_s is then formed by the concatenation of the segments T_1 to T_{36} and can be used with the public key for authentication.

3.0 Security Analysis

Design structure of the MKGA is a novel one because it incorporates the unique characteristics of graphical image codes and fingerprint biometric. When compared with the conventional alphanumeric password model, MKGA offers more security in an on-line attack because to compromise the system the attacker has to generate the legitimate user's fullname, graphical image codes and fingerprint and to dynamically regenerate the private key. In an off-line attack, it is assumed that the attacker captures the login registry files and then uses them in an off-line effort to regenerate the private key.

In an off-line attack, we assume the attacker has been able to access the files $EDbas_{Mui}$ and $EDbas_{ec}$ from the registry files and have by some means decrypted the files to retrieve the files $Dbas_{Mui}$ and $Dbas_{ec}$. However, because $Dbas_{Mui}$ contains only the ***Mui*** of U_s, no information of ***FINkey*** will be revealed. Also if the attacker decides to exploit $Dbas_{ec}$ for an offline attack and recreate the private key, the attacker has to make a choice on five authentic shares from all the shares in $Dbas_{ec}$ to create a distinct segment of the private key. And to check the secret authenticity, the attacker needs a minimum of seven authentic shares. To generate the right key, the attacker needs to execute 41! Permutations and derive 3.3×10^{49} number of iterations and in each iteration concatenate the secrets in the exact order.

A brute force attack of this nature is very time consuming and the search will be inexhaustible. For instance, an analysis of the number of keys possible with various textual password lengths using the best character set constraints, which is 8-bit ASCII characters (256) (contrasting lowercase letters (26), lowercase letters/digits (36), all alphanumeric characters (62), printable characters (95), 7-bit ASCII characters (128)) revealed that at 1 million keys/second, the amount of time to search all possible keys of a six character password, yielded $2.8\text{x}10^{14}$ number of iterations at 8.9 years, seven character password yielded $7.2\text{x}10^{16}$ number of iterations at 2,283 years, eight character password yielded $1.8\text{x}10^{19}$ number of iterations at 570,776 years and even a twenty character password yielded $1.4\text{x}10^{48}$ number of iterations at an almost inexhaustible time frame [15]. This is a significant high overhead, but is still not comparable with what the attacker has to contend with for the derived key size of 3.3×10^{49}.

4.0 Testing Framework

In the system performance testing experiment, we employed a modified testing methodology similar to that proposed by Phillips, et al [20, 21]. The framework which was extended to accommodate a multifactor authentication system involves the following seven steps:

a. *Factoid Selection:* Collect two sample sets from all authentication factors (i.e. in this case, graphical password via alphanumeric codes and fingerprint biometric codes) to get a target set (*T)* and a query set (*Q*). *T* contains the set of

b. credentials known to the system while *Q* contains the set of credentials that are to be compared against the target set. Separate instances of graphical password via alphanumeric codes and fingerprint biometric codes are used for users in both sets. For practical tests the intersection of any of these two sets should not be null, instead the intersection of any of these two sets contains the users to be found in the database.
c. *Score Allocation*: For each pair of query and target sets from all authentication factors obtain a similarity score matrix whose size is query set size by target set size. The score for graphical passwords and fingerprint biometric codes is a similarity score.
d. *Score Extraction*: Gallery and probe subsets are extracted from the target/query similarity matrix, respectively, to perform "virtual" experiments on a subset of the population, where a gallery is any arbitrary subset of the target set and the probe is any arbitrary subset of the query set.
e. Steps 1-3 are repeated for each factor.
f. *Data Operation*: The similarity matrices from step 2 is assemble and align; this includes converting data to a common format, forming subsets to obtain matrices of the same size, and mating each data to create real or virtual subjects. The result is a set of similarity matrices of equal size representing match-score data for mated subjects in a common format convenient for processing.
g. *Score Normalization:* Normalize the assembled similarity matrices to a common number range without reducing the dimensionality of the data.
h. *Score Fusion:* Fuse the set of normalized similarity matrices into a single fusion similarity matrix. A fusion function, *f(x1,…xn)*, defines a mapping from *n-space*, where each factor represents one of the *n* dimensions, into a single fused dimension. A threshold divides this range into an *accept* and *reject* part. Operationally, the threshold or boundary is derived from an estimate of the Receiver Operating Characteristic (ROC) curve developed in step 8.
i. *Performance Statistics*: Performance Statistics for verification are computed from the genuine and imposter scores. Genuine scores are those that result from comparing elements in the target and query sets of the same subject. Imposter scores are those resulting from comparisons of different subjects. Use each fusion score as a threshold and compute the false-accept rate (FAR) and false-reject rate (FRR) by selecting those imposter scores and genuine scores, respectively, on the wrong side of this threshold and divide by the total number of scores used in the test. A mapping table of the threshold values and the corresponding error rates (FAR and FRR) are stored. The complement of the FRR (1 – FRR) is the genuine accept-rate (GAR). The GAR and the FAR are plotted against each other to yield a ROC curve, a common system performance measure. In practice, one chooses a desired operational point on the ROC curve and uses the FAR of that point to determine the corresponding threshold from the mapping table.

5.0 Experimental Results

Based on steps a-e, the database consisted of 52 non-habituated users. For the graphical password, the image set comprised of 15 x 15 single unique images. To create a graphical password, a minimum of 6 images and a maximum of 10 images were selected. Users choose any of the choices of images upon their preference. For the fingerprint registration, the SeCugen scanner was used and each finger of choice needs to be left in the fingerprint sensor device for the system to scan the finger four times in order for the registration to be established.

In the first login phase, the users were asked to repeat the operation three subsequent times at a time interval of 1 week for the first two logins and 2 weeks for the second two logins under supervised conditions. The initial login graphical password scores was used as the target set known to the system while the subsequent login scores became the query set of subjects to be compared against the target set. Using the third login scores as the application query scores, we obtained a match-score. From the fingerprints scanned, each fingerprint from the test set (T) is tested with the same fingerprint from the target set (Q) (three different capture prints of the same finger in which any can be stored in the database) and used to generate scores. The system matches between two fingerprints and computes the Euclidean Distance. From these, 13 gallery and probe subsets were selected respectively to perform the 'virtual experimentations.

To normalize scores, the min-max score normalization technique was utilized because it is suitable when the bounds of the scores produced are known. The assembled similarity matrices scores were normalize to a common number range without reducing the dimensionality of the data. The min-max normalized score for the test score s_{ik} is given by

$$S^{I}_{ik} = \frac{S_{ik} - min(\{S_i\})}{max(\{S_i\}) - min(\{S_i\})}, \text{ where } \{S_i\} = \{S_{i1}, S_{i2}, \ldots S_{ig}\}. \quad (7)$$

ss_{ij} denote the j^{th} matching score output by the i^{th} factor, where $i = 1, 2, \ldots, f$ and $j = 1, 2, \ldots, g$ (f is the number of factors and g is the number of scores available in the set). For the graphical password similarity scores, normalizing the scores derive the same set of initial scores. While for the fingerprint scores after normalizing the scores, next the scores are transformed into similarity scores by subtracting the normalized score from 1.
To fuse the set of normalized similarity matrices into a single fusion similarity matrix, a fusion function, *f(x1,… xn)*, defined mapping from n-space, where each factor represented one of the n dimensions to a single fused dimension. The simple sum rule, which generally performs well over the range of normalization technique, was applied at step g [22].
The performance statistics for verification revealed a very high genuine acceptance rate for legitimate users and illegitimate users were rejected.

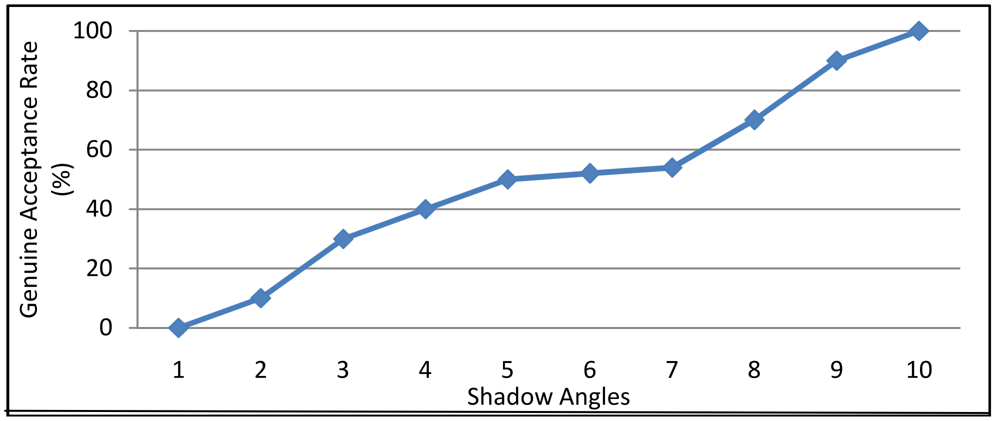

Figure 2: Genuine Acceptance Rate

Figure 2 shows the results obtained using the genuine fingerprint similarity score for different shadow angles. This resulted in a 100% genuine acceptance rate, which is the complement of the FRR (1 – FRR). And as the genuine acceptance rate increases, the false acceptance rate reduces. Users' credentials were never the same because MKGA accurately distinguishes them via the random seed value, unique characteristics of graphical image codes and fingerprint biometric. This facilitates the regeneration of an invariable key value for a legitimate user.

6.0 References

[1]. Birget J., Hong D. and Memon N. (2005). Graphical Passwords Based on Robust Discretization. Available at: clam.rutgers.edu/~birget/grPssw/robDiscr.pdf

[2]. Perrig, A. and Song, D. (1999): Hash Visualization: A New Technique to Improve Real World Security. In International Workshop on Cryptographic Techniques and E-Commerce. 131–138.

[3]. Miller, G. A. (1996): The magical number seven, plus or minus two: Some limits on our capacity for processing information. The Psychological Review, 63, 81-97.

[4]. Ives, B., Walsh, K., and Schneider, H. (2004). The domino effect of password reuse. Communications of the ACM, 47 (4): 75-78.

[5]. Sobrado, L., & Birget, J. C. (2002). Graphical passwords, The Rutgers Scholar, An Electronic Bulletin for Undergraduate Research, 4: 12-18.

[6]. O'GORMAN L. (2003): Comparing Passwords, Tokens, and Biometrics for User Authentication. Proceedings of the IEEE, 91(12): 67 - 69.

[7]. Monrose F. and Reiter M. (2005): Graphical passwords, in Security and Usability: Designing Secure Systems That People Can Use, L. Cranor and S. Garfinkel, Eds. O'Reilly Media, 2005, ch. 9: 157–174.

[8]. Feng H. and Wah C. (2002). "Private Key Generation from On-Line Handwritten Signatures", *Information Management & Computer Security*, 10 (4): 159 – 164.

[9]. Fabian M. and Aviel R..(1997): Keystroke Dynamics as a Biometric for Authentication. Available at: http://www.cs.jhu.edu/~fabian/papers/fgcs.pdf

[10] Hao F. and Chan Choong W. (2002). "Private Key Generation from On-Line Handwritten Signatures", *Information Management & Computer Security*, 10 (4): 159 – 164.

[11] Uludag U., Pankanti S. and Jain A. K. (2005). "Fuzzy vault for fingerprints," *To appear in Proc. Audio- and Videobased Biometric Person Authentication (AVBPA)*, (Rye Brook, NY), July 2005.

[12] Suo X., Zhu Y. and Owen G.S. (2005). Graphical passwords: A survey, 21st Annual Computer Security Applications Conference (ACSAC'05). 463-472.

[10]. Real User Corporation (2001). The Science Behind Passfaces, Document Revision 2, Real User Corporation, September 2001. Available at: http://www.realuser.com/published/ScienceBehindPassfaces.pdf

[11]. Dhamija R. and Perrig A. (2000). Deja Vu: A User Study Using Images for Authentication, in Proceedings of 9th USENIX Security Symposium. Denver, CO, 2000. 45–58. Available at: http://www.usenix.org/publications/library/proceedings/sec2000/dhamija.html.

[12]. Schneier, B. (1999). Inside risks: the uses and abuses of biometrics, August 1999 Communications of the ACM, 42 (8): 22 - 25.

[13]. Davis, D. Monrose, F. and Reiter, M.K. (2004). On user choice in graphical password schemes. In Thirteenth Usenix Security Symposium. San Diego, CA, USA. Available at: http://www.usenix.org/events/sec04/tech/davis.html.

[14]. Man S., Hong D. and Mathews M. (2003). A shouldersurfing resistant graphical password scheme, in Proceedings of International conference on security and management. Las Vegas, NV. 35-38.

[15]. Weinshall, D. and Kirkpatrick, S. (2004). Passwords You'll Never Forget, but Can't Recall, in Proceedings of Conference on Human Factors in Computing Systems (CHI). Vienna, Austria: ACM, 1399-1402.

[16]. SecuGen Corporation (2007). Available at: www.seugen.omfingerprint

[17]. Phillips, J., et al. (2003). "Face Recognition Vendor Test 2002. Evaluation Report". NISTIR 6965. Available at: http://www.frvt.org.

[18]. Phillips, P.J., Rauss, P.J. and Der S. (1996). "FERET (Face Recognition Technology) Recognition Algorithm Development and Test Results," Army Research Laboratory technical report, ARL-TR-995. Available at: http://www.frvt.org.

[19]. Kittler, J., Hatef, M. Duin, R. and Matas, J.G. (1998). "On Combining Classifiers", IEEE Transactions on PAMI 20 (3): 226-239.