American Journal of Applied Sciences 5 (7): 777-782, 2008
ISSN 1546-9239

A Password-Based Key Derivation Algorithm Using the KBRP Method

[1]Shakir M. Hussain and [2]Hussein Al-Bahadili
[1]Faculty of CSIT, Applied Science University, P.O. Box 22, Amman 11931, Jordan
[2]Faculty of Information Systems and Technology,
Arab Academy for Banking and Financial Sciences, P.O. Box 13190, Amman 11942, Jordan

Abstract: This study presents a new efficient password-based strong key derivation algorithm using the key based random permutation the KBRP method. The algorithm consists of five steps, the first three steps are similar to those formed the KBRP method. The last two steps are added to derive a key and to ensure that the derived key has all the characteristics of a strong key. In order to demonstrate the efficiency of the algorithm, a number of keys are derived using various passwords of different content and length. The features of the derived keys show a good agreement with all characteristics of strong keys. In addition, they are compared with features of keys generated using the WLAN strong key generator v2.2 by Warewolf Labs.

Key words: Key derivation, key generation, strong key, random permutation, Key Based Random Permutation (KBRP), key authentication, password authentication, key exchange

INTRODUCTION

Key derivation is the process of generating one or more keys for cryptography and authentication, where a key is a sequence of characters that controls the process of a cryptographic or authentication algorithm[1,2]. Strong key derivation has become a primary concern in wireless network in order to provide secure communication in a hostile environment. For example, Diffie-Hellman based key exchanges establish a secure communication channel between two parties by securely negotiating a large random element in a given cyclic group, called master secret. Then, this secret is used to derive keys for encrypting and authenticating data[3]. These keys must be bit-strings of some specific length uniformly distributed and used as input parameters to symmetric ciphers (for privacy), message authentication codes (for authentication) and pseudo-random functions (for expansion of a seed into a longer bit-string)[4].

There are a number of approaches that have been developed for strong key derivation such as: functional based[5,6,2], biometric based[7,8,9] and voice based[10,11,12]. The functional based key derivation approach is often used to derive one or more keys from a common secret value (password); therefore, it may be referred to as a password-based key derivation[2].

For strong cryptography, keys need to be carefully selected and must acquire some special characteristics, such as[2,7,13]:

- The key size must be long enough so that it can not be easily broken
- The key should have the highest possible entropy (close to unity)
- Run length should be less than eight bits (a character word length)
- Bits are randomly distributed throughout the key sequence
- No particular pattern can be recognized throughout the key sequence

Hash function and pseudorandom function are widely used for key derivation. These two functional approaches are used in PBKDF1 and PBKDF2, respectively. PBKDF2 is recommended for new applications while PBKDF1 is included only for compatibility with existing applications and is not recommended for new applications (Kal 00). A wired equivalent privacy or wireless encryption protocol (WEP) is another functional approach that is developed to provide security for wireless networks. However, a number of techniques and programs are now available that can crack a WEP key in less than a minute. Despite that a wireless strong key generator has been developed to enhance wireless security such as the WLAN strong key generator v2.2 by Warewolf Labs[14].

In this study a new efficient password-based key derivation algorithm is proposed for a strong key derivation, regardless of the password elements and

Corresponding Author: Shakir M. Hussain, Faculty of CSIT, Applied Science University, P.O. Box 22, Amman 11931, Jordan

length. The algorithm utilizes the Key Based Random Permutation (KBRP) method in[15]. In order to ensure that the binary sequence generated is hard to predict, first, an important modification is introduced to the last step in KBRP, namely the fill () step. Second, two further steps are added to the KBRP method to derive a strong key.

The modified KBRP method assures that the key derived is of highest possible entropy (close to unity, since the number of 0's and 1's are only differing by 1). But the derived key may not provide an adequate run length for 0's and 1's. Therefore, in order to satisfy a suitable run length (RL) of 0's and 1's, we march through the derived key and every time the run length is greater than RL, then the last bit in RL is replaced with the subsequent bit that represents its complement.

This algorithm ensures even with weak password, a strong key can be derived to satisfy the characteristics mentioned above. So that user can still use their favorite password set without affecting the key strength.

The features of the keys generated using the proposed algorithm are evaluated, analyzed and compared with the results obtained from the WLAN strong key generator v2.2 by Wareworlf Labs. The features evaluation process is done by comparing the entropy of the derived key, maximum run length of 0's and 1's and the average run length.

LITERATURE REVIEW

A number of key derivation approaches have been developed throughout the years, such as: functional based[2,5,6], biometric based[7,8,9] and voice based[10,11,12].

B. Kaliski[2] proposed a functional based approach to derive a key from a password and other parameters such as a salt value and an iteration count. He developed two functions for key derivation, namely, PBKDF1 and PBKDF2. PBKDF2 is recommended for new applications; PBKDF1 is included only for compatibility with existing applications and is not recommended for new applications.

He defined four steps in his key derivation functions. These are: (1) select a salt value and an iteration count, (2) select a length in octets for the derived key, (3) apply the key derivation function to the password, the salt, the iteration count and the key length to produce a derived key and (4) output the derived key. In this approach, many numbers of keys may be derived from a password by varying the salt.

Costanzo[7] proposed a biometric key derivation approach for generating a cryptographic key from an individual's biometric for use in proven symmetric cipher algorithms. The proposed approach uses a method referred to as Biometric Aggregation. In this approach, the encryption process begins with the acquisition of the required biometric samples. Features and parameters are extracted from these samples and used to derive a biometric key that can be used to encrypt a plaintext message and its header information. During the decryption process, acquisition of additional biometric samples from which the same features and parameters are extracted and used to produce a "noisy" key as done in the encryption process. Next, a small set of permutations of the "noisy" key are computed. These keys are used to decrypt the header information and determine the validity of the key. If the header is determined to be valid, then the rest of the message is decrypted. The proposed approach eliminates the need for biometric matching algorithms, reduces the cost associated with lost keys and addresses non-repudiation issues.

Monrose *et al.*[11,12] developed a technique to reliably generate a cryptographic key from a user's voice while speaking a password. The key resists cryptanalysis even against an attacker who captures all system information related to generating or verifying the cryptographic key. Moreover, the technique is sufficiently robust to enable the user to reliably regenerate the key by uttering the password again. They described an empirical evaluation of their technique using 250 utterances recorded from 50 users.

Teoh[9] proposed a novel two-stage technique to generate personalized cryptographic keys from the face biometric, which offers the inextricably link to its owner. At the first stage, integral transform of biometric input is to discretise to produce a set of bit representation with a set of tokenised pseudorandom number, coined as FaceHash. In the second stage, FaceHash is then securely reduced to a single cryptographic key via Shamir secret-sharing. Tokenised FaceHashing is rigorously protective of the face data, with security comparable to cryptographic hashing of token and knowledge key-factor. The key is constructed to resist cryptanalysis even against an adversary who captures the user device or the feature descriptor.

Peyravian *et al.*[16] used either a 'user Identifier' (userID) or information that may be used to identify the individual such as a user's biometric data. These data can be the user's fingerprints, hand geometry, iris pattern, or any other suitable biometric identification. Then such user ID-based or biometric-based key or random number generated based on their algorithms may be used in asymmetric-key cryptographic systems (such as the RSA) or the symmetric-key cryptographic systems (such as the DES).

THE PROPOSED KEY DERIVATION ALGORITHM

Here we present a description of the proposed key derivation algorithm. It utilizes the KBRP method[15]; therefore, we first provide a brief description of this method followed by a detail implementation of the proposed algorithm.

KBRP METHOD

Key Based Random Permutation (KBRP) is a method that generates one permutation of size N out of N! permutations[15]. This permutation is generated from a certain key (password) by considering all elements of the password in the generation process. The permutation is stored in one-dimensional array, P, of size N. The KBRP method consists of three consecutive steps: init (), eliminate () and fill ().

The first step, init (), initializes an array of size N with elements from the given password. It takes the ASCII code of each element in the password and storing them in the array consecutively. If the password provides less than N elements, the unfilled elements are derived by adding two consecutive values, starting at the first element of the array until all unfilled elements are set to values. Finally, the mod operation with respect to N+1 is calculated for each element, so that all values are set within the range 1 to N.

The second step, eliminate (), is to eliminate the similar values by replacing them with zero and keep only one value out of these similar values. Last step, fill (), is to replace all zero values with values in the range 1 to N that are not exist in the array. The resulted array is now representing the permutation.

THE PASSWORD-BASED KEY DERIVATION ALGORITHM

The proposed algorithm consists of five steps. These are: init (), eliminate (), fill (), derive () and certify (). The first three steps are similar to those in the KBRP method explained above.

The forth step, derive (), derives the key from the permutation P generated from the KBRP method by performing mod 2 operation for each element in P. This ensures that each element in P will have either 0 or 1 and the number of 0's and 1's are even, since N is always an even number. To satisfy one of the requirements of a strong key by do not use an equal number of 0's and 1's, we invert one or more bits of the same value. These bits can be selected randomly or according to a particular procedure. However, in our algorithm, in order to obtain the highest acceptable entropy (close to unity), only one bit is complimented. This bit is chosen according to the value of the element number one in the permutation array, so that the derived key depends on the password elements entered and different keys can be derived depending on the position of inverted bit. The drive () step is performed to ensure that bits are randomly distributed throughout the binary key sequence.

```
P: array holds permutation with
Values 1 to N
KEY: key (string of bits) of size N
For (i = 1 to N)
   KEY[i] = P[i] MOD 2
KEY[P[1]] = NOT KEY [P[1]]
```

Fig. 1: Algorithm for the derive() step

```
KEY: key (string of bits) of size N.
RL: run length set to 4
Bits0: counter holds consecutive 0 bits
Bits1: counter holds consecutive 1 bits
while KEY contain run length > RL
      set Bits0, Bits1 to 0
      count Bits0 in KEY
      count Bits1 in KEY
      if (Bits0 > RL)
            swap one 0 bit with the first following
                        1 bit in KEY
            set Bits0 to 0
      if (Bits1 > RL)
            swap one 1 bit with the first following
                        0 bit in KEY
            set Bits1 to 0
```

Fig. 2: Algorithm for the certify() step

Figure 1 outlines the procedure of the key derivation step. However, this step will not ensure that the derived key satisfies the run length requirement for a strong key. For this reason, we use step 5 to certify the key.

The fifth step, certify (), takes the outcome of the forth step and check the run length for the two binary digits 0 and 1 to make sure that the run length for each of them is not more than a particular value, r. To achieve this objective without varying the entropy, the binary sequence is checked and once the run length equal to r, the r^{th}+1 bit is swapped with its next complement bit (here we use r = 4). Fig. 2, shows the algorithm for the certify () step.

Table 1: 40-bit derived keys for various passwords

Password	40-bit derived key	
aaaa	A892236DB6	6927545F71
KBRP	63276B3867	2B313E2650
1234	0DAF1EEF10	5A7934265E
computer	C4B0891DDE	614D756356
success246	E2645E4C96	376930787E

Table 2: 104-bit derived keys for various passwords

Password	104-bit key
aaaa	0CCA7BB4F6C91ECD9B3712C224
	28296B476E5A5B39744B47657D
KBRP	631176F42B6CC6312B4927B776
	246370512A673F264A7C635649
1234	ED3DB0896112C22584769EEC3C
	4438206C2D7840522237672F5A
computer	E9611286E4CAC75DB7878448D7
	772575313A7A4C45592E7D3773
cuccess246	F5BAC35137BA5132EC8C2A6566
	7E5E2341315E5628313768734A

Table 3: 128-bit derived keys for various passwords

Password	128-bit key
aaaa	49DA7BB4F776EC9EE6762264B0896110
	7A7B365F4E523A2A3E2032242B427746
KBRP	63223AE4753350A599772170E332EB5B
	3A6977547247717A575B623A20656835
1234	12DC2113152D3C4DA93DDBD34E9EF190
	55506D39522B2726673C237032375E3A
computer	F4B0891DD556112C2E99C4B09BCED3DD
	2E312536443351573 85D2D5A69302252
success246	EEBA5275B663512C21A3730C2D13ADAF
	675065577247645371276645 5F7A7D66

Table 4: 232-bit derived keys for various passwords

Password	232-bit key
aaaa	ED89D8791EC223C869EEDB8761627
	13D3B1B85E5844B0B8F113AEC3784
	525A72613B5D445F75597C6F57247
	3205D5F2451243A4771735F31623E
KBRP	632F32E72939EE8727263AF48F6A8
	EBD9C48E30DA786943846622E8D8C
	2C314C5A5A2C4A216B58466C55726
	35540245D6131437A584A7125652F
1234	EEE258422896113CB0896110A6D3D
	DE32D3DDA7BB7B31CC242F27B27BB
	4170242D665145287839484227476
	36A202B5E4E5330493C7A5F617749
computer	12225769EED3DDA7BB4F769EED3DD
	88E1371ECD8F4B0896112C2258448
	33346D41526F746B2B61493378512
	6655769474D6171335E3E2A733D5D
success246	F5BB7AC58BC31849D657A61911553
	573DE584498C8630CF76DACC642F3
	2246757C483C5F3C5B7A40767D3A7
	84E6E6D4D472D58366664294F5E74

ILLUSTRATIVE EXAMPLES

The five steps forming the proposed algorithm are implemented in a small code using C++ language. The code takes as an input any character set (password) and output a binary key that has all features of a strong key; such a highest possible entropy, a run length of both 0's and 1's is less than or equal to r (4<r<8). In addition, no pattern can be recognized throughout the entire key and a minimum number of repetitions are allowed for any sub set of the binary sequence within the key. In particular, the code calculates a number of parameters such as: entropy of the key, the number of 0's and 1's and the minimum, maximum and average run length for both 0's and 1's.

In order to demonstrate the efficiency of the new algorithm in deriving strong keys, we present the results for the key derived for five different initial passwords, these are:

- Password of 4 similar letters (aaaa)
- Password of 4 different letters (KBRP)
- Password of 4 numeric values (1234)
- Password of 8 different letters (computer)
- Password of 10 different alphanumeric characters (success246)

The results for 40, 104, 128 and 232 bits key lengths are presented in Table 1-4, respectively. The derived keys are compared with those obtained from the WLAN strong key generator program v2.2 by Warewolf Labs[14] in Table 5-8.

The results show that the keys derived using the proposed algorithm always have the maximum acceptable entropy, a controlled run length for both 0's and 1's of not more than a particular value (in this case r=4) for all key lengths and an acceptable average run length. On the other hand, the tests show that for the Warewolf program, the run length may be more than 8 for long keys.

CONCLUSION

This study presents a new functional password-based strong key derivation algorithm using the key based random permutation (KBRP) method. The KBRP method consists of three computational steps (init (), eliminate (0 and fill () steps) to generate a permutation P of size N out of N! possible permutations. Two other computational steps (derive () and certify () steps) are added to process the outcome of the KBRP method to derive a key that meets all the features of a strong key. In particular, the derived key has a maximum

Table 5: A comparison between the proposed algorithm and the WLAN strong key generator v2.2 by Warewolf Labs for 40-bit key length

Password	Entropy	No of 0's/1's	Max RL 0's/1's	Ave RL 0's/1's
aaaa	0.9982	21/19	3/2	1.62/1.46
	0.9982	19/21	3/5	1.58/1.75
KBRP	0.9982	19/21	4/3	1.90/2.10
	0.9837	23/17	4/5	2.09/1.70
1234	0.9982	19/21	4/4	2.11/2.63
	1.0000	20/20	4/4	1.67/1.82
computer	0.9982	21/19	4/4	2.10/1.90
	1.0000	20/20	4/3	1.43/1.54
success246	0.9982	21/19	3/4	1.91/1.73
	0.9982	19/21	5/6	2.11/2.63

Table 6: A comparison between the proposed algorithm and the WLAN strong key generator v2.2 by Warewolf Labs for 104-bit key length

Password	Entropy	No of 0's/1's	Max RL 0's/1's	Ave RL 0's/1's
aaaa	0.9997	51/53	4/4	1.82/1.96
	0.9989	50/54	5/5	1.56/1.69
KBRP	0.9997	51/53	4/4	1.70/1.83
	0.9957	56/48	5/6	2.00/1.71
1234	0.9997	53/51	4/4	1.96/1.82
	0.9783	61/43	7/4	2.35/1.72
computer	0.9997	53/51	4/4	1.96/1.82
	0.9957	48/56	3/5	1.60/1.87
success246	0.9997	51/53	4/4	1.65/1.71
	0.9976	55/49	5/6	1.96/1.82

Table 7: A comparison between the proposed algorithm and the WLAN strong key generator v2.2 by Warewolf Labs for 232-bit key length

Password	Entropy	No. of 0's/1's	Max RL 0's/1's	Ave RL 0's/1's
aaaa	0.9998	51/53	4/4	1.85/1.97
	0.9984	67/61	7/5	1.97/1.85
KBRP	0.9998	65/63	4/4	1.81/1.75
	0.9998	65/63	6/4	1.76/1.70
1234	0.9998	65/63	4/4	1.88/1.85
	0.9984	67/61	6/4	1.86/1.74
computer	0.9998	63/65	4/4	1.82/1.84
	0.9857	73/55	6/3	1.87/1.45
success246	0.9998	63/65	4/4	1.75/1.76
	0.9984	61/67	5/5	1.65/1.86

Table 8: A comparison between the proposed algorithm and the WLAN strong key generator v2.2 by Warewolf Labs for 232-bit key length

Password	Entropy	No. of 0's/1's	Max RL 0's/1's	Ave RL 0's/1's
aaaa	0.99995	117/115	4/4	2.17/2.13
	0.99995	115/117	6/5	1.80/1.86
KBRP	0.99995	117/115	4/4	1.98/1.98
	0.98623	132/100	8/4	1.91/1.45
1234	0.99995	115/117	4/4	1.98/1.98
	0.98790	131/101	7/5	2.15/1.66
computer	0.99995	115/117	4/4	1.89/1.95
	1.00000	116/116	5/5	1.71/1.71
success246	0.99995	115/117	4/4	1.89/1.89
	0.99979	114/118	7/5	1.90/2.00

acceptable entropy, the number of 0's and 1's may be set to differ by a specific value (in this work it is set to differ by 1 only, a controlled maximum run length for both 0's and 1's and a satisfactory average run length. These key features are validated against key features generated using a standard WLAN strong key generator v2.2 from Warewolf Labs.

ACKNOWLEDGEMENTS

This study received financial support towards the cost of its publication from the Deanship of Research and Graduate Studies at Applied Science University, Amman - Jordan.

REFERENCES

1. Chevassut, O., P.A. Fouque, P. Gaudry and D. Pointcheval, 2005. Key Derivation and Randomness Extraction.
2. Kaliski, B., 2000. Password-Based Cryptography Specification Version 2. Network Working Group, RFC 2898-PKCS#5, http://www.faqs.org/rfcs/rfc2898.html
3. Diffie, W. and M.E. Hellman, 1976. New Directions in Cryptography. IEEE Transactions on Information Theory, 22: 644-654.
4. Bellare, M., R. Canetti and H. Krawczyk, 1996. Keying Hash Functions for Message Authentication. In Crypto '96, LNCS 1109, Springer-Verlag, Berlin, pp: 1-15.
5. Dodis, W., R. Gennaro, J. Hastad, H. Krawczyk and T. Rabin, 2004. Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. Proceedings of Crypto '04, LNCS, Springer-Verlag, Berlin, pp: 494-510.
6. Zorn, G., 2001. Deriving Keys for Use with Microsoft Point-to-Point Encryption (MPPE). Network Working Group.
7. Costanzo, C.R., 2004. Biometric Cryptography: Key Generation Using Feature and Parametric Aggregation. School of Engineering and Applied Sciences, Department of Computer Science, George Washington University.
8. Uludag, U., S. Pankanti, S. Prabhakar and A. Jain, 2004. Biometric Cryptosystems: Issues and Challenges. Proceedings of the IEEE, Vol. 92.
9. Teoh, A.B., D.C. Ngo and A. Goh, 2004. Personalised Cryptographic Key Generation Based on FaceHashing. Computers and Security, 23: 606-614.
10. Monrose, F., M.K. Reiter, Q. Li, D. Lopresti and C. Shih, 2002. Towards Voice Generated Cryptographic Keys on Resource Constrained Devices. Proceedings of the 11th USENIX Security Symposium.
11. Monrose, F., M.K. Reiter, Q. Li and S. Wetzel, 2001. Cryptographic Key Generation From Voice. Proceedings of the IEEE Conference on Security and Privacy, USA.
12. Monrose, F., M.K. Reiter, Q. Li and S. Wetzel, 2001. Using Voice to Generate Cryptographic Keys. Speech Recognition Workshop, Greece.
13. Roginsky, A., 2004. A New Method for Generating RSA Keys. International Business Machines Consulting Group.
14. Elliott, C., 2005. WLAN Strong Key Generator v2.2 by Warewolf Labs. Warewolf Labs, Warewolf Website.
15. Hussain, S.M. and N.M. A-Ajloni, 2006. Key Base Random Permutation (KBRP). J. Computer Sci., 2: 419-421.
16. Peyravian, M., S.M. Matyas, A. Roginsky and N. Zunic, 1999. Generating User-Based Cryptographic Keys and Random Numbers. Computers and Security, 18: 619-626.