Context-Aware Multifactor Authentication Based on Dynamic Pin

Yair Diaz-Tellez, Eliane Bodanese, Theo Dimitrakos, Michael Turner

CONTEXT-AWARE MULTIFACTOR AUTHENTICATION BASED ON DYNAMIC PIN

Yair Diaz-Tellez[1], Eliane L. Bodanese[1], Theo Dimitrakos[2], Michael Turner[2]

[1]School of Electronic Engineering and Computer Science, Queen Mary University of London, London, United Kingdom
{y.diaz-tellez, eliane.bodanese}@qmul.ac.uk
[2]Security Futures Practice, BT Innovate and Design, British Telecommunications Ipswich, United Kingdom
{theo.dimitrakos, michael.turner}@bt.com

Abstract. An innovative context-aware multi-factor authentication scheme based on a dynamic PIN is presented. The scheme is based on graphical passwords where a challenge is dynamically produced based on contextual factors and client device constraints while balancing security assurance and usability. The approach utilizes a new methodology where the cryptographic transformation used to produce the Dynamic PIN changes dynamically based on the user input, history of authentications, and available authentication factors at the client device.

Keywords: authentication, dual ciphers, context-aware, dynamic PIN

1 Introduction

User authentication is a means of identifying a user and verifying his identity. Different authentication methods exist, e.g. token-based, biometric-based, and knowledge-based. Each method has its own properties, (dis)advantages, and applications. Text passwords are a widely used method because of convenience and usability; however, they are vulnerable to key logging, shoulder-surfing, dictionary, and social engineering attacks. Graphical passwords are an alternative as they can mitigate the abovementioned attacks. One approach to increase assurance is multi-factor authentication. However, not all transactions require the same assurance level. An adequate level depends on criticality, sensitivity, context, and the risk involved. Additionally, there are trade-offs among variables such as assurance, performance, and usability.

This work proposes an innovative context-aware multi-factor authentication scheme based on a Dynamic PIN. The scheme produces a graphical challenge based on context, client device constraints, and risk associated, while balancing assurance and usability. Also, a methodology is proposed where the crypto-function used to produce the Dynamic PIN changes dynamically. A PIN is generated without any predictable backward and forward correlation making practically infeasible for an attacker to predict the next PIN. The approach leverages on the fact that users commonly

use various types of client devices that incorporate authentication factors (e.g. SIM cards, biometric readers, etc.), sensors, and APIs, which can be integrated in the authentication process to modulate security assurance, and to optimize it using context.

Section 2 presents related work. The scheme consists of two functional phases: **registration and setup**: the user creates an account and registers different information (section 3); and **challenge and dynamic PIN**: a challenge and Dynamic PIN are generated (section 4). Section 5 presents the conclusions and future work.

2 Related Work

The proposed scheme considers how the properties of graphical passwords[1] can be adjusted at runtime balancing assurance vs. usability. In [2], a PIN-based mechanism is presented that uses a secret sequence of objects to analyze security vs. usability. This work does not consider the use of contextual information to influence the generation of the challenge. Several frameworks have been proposed that make use of context [3, 4]. [5] introduces the notion of implicit authentication that consists in authenticating users based on behavioral patterns. [6] presents a framework that combines passive factors (e.g. location) and active factors (e.g. tokens) in a probabilistic model for selecting an authentication scheme that satisfies security requirements; however, it does not consider client device constraints.

Security tokens, implementable on hardware[7] and software[8], typically generate a one-time-password (OTP)[9] in response to the user typing a PIN. Several token-based systems feature a fixed function that outputs the OTP [10-14]. The proposed system is effectively a software-based security token that produces an OTP, i.e. the Dynamic PIN; and additionally, the crypto-function used changes itself dynamically improving the pseudo-randomness of the scheme.

3 Registration and Setup

Registering authentication factors. The scheme uses authentication factors that may be available on the client device, e.g. SIM number, or fingerprint. For each factor, the server creates a record and associates a secret seed generated randomly, $\left(af_{name_i}, af_{value_i}, af_{seed_i} \right)$ with af_i an authentication factor. For example, (IMSI, 464989052765867, 4596). A vector of secret seeds is pushes into the device. An authentication token is computed by $AuthNToken = PBKDF2(concat(\ (\ af_{value_i} + af_{seed_i}\ )))$. PBKDF2, a Key Derivation Function, is used to derive a key strong against brute force attacks. A token vector is defined as $AuthNTokenVector = AuthNToken_1, AuthNToken_2, \ldots AuthNToken_m$, where m is the number of authentication factors registered for a device. The client must recreate the same token calculated at the server side. Notice that the value af_{value_i} is obtained at runtime.

Registering the image-based password(s). The user is presented with a selection of image categories, $\mathbb{C} = \{C_1, C_2, C_3, \ldots\}$. Each category consists of image objects

grouped by common characteristics easy to understand, e.g. icons. Let category C= $(o_1, o_2, o_3, .. o_n) \in \mathbb{C}$ be a set of n objects. A seed is randomly generated for each object o_i. Let $seedsVector = (seed_1, seed_2, seed_3, .. seed_n)$ be the vector of seeds with $(o_i, seed_i) \in (C \times seedsVector)$. Each $seed_i$ is of 2 bytes length and represented as 4 hexadecimal digits (0-F). The user selects a subset of objects as his secret password: $secretSequence \subset C$. For $|n| \gg |secretSequence|$ a brute force attack can become increasingly difficult as the number of combinations and permutations increases.

Registering device parameters. This includes form factor parameters about the device(s): type of device, display size, authentication interfaces – e.g. biometrics. These are used to specify at runtime an adequate customization of the image challenge.

4 Challenge and Dynamic PIN Generation

Overview. Steps: (i) the server generates a random pin string (RPS) and the graphical challenge. The RPS is used as part of the dynamic pin generation algorithm. The challenge is constructed by combining a subset of secret and non-secret images based on device constraints, context, and level of assurance required. Due to lack of space, it is assumed the communication channel between client and server is secured. A key exchange protocol, e.g. Diffie Hellman, can be easily incorporated in the scheme. (2) the user is asked to recognize the subset of secret images; (3) a crypto-function is computed dynamically based on different variable elements of information including user input, authentication factors, and history of authentication attempts; (4) the crypto-function is then used to generate the dynamic pin; (5) the client device sends the dynamic pin to the server for validation.

4.1 Generation of the RPS and the context-based image challenge**.**

In this section, first the random pin string (RPS) and the image challenge are introduced. Then, a rules-based mechanism used to parameterize and generate the challenge based on runtime context information, device constraints, and risk is presented.

Random pin string (RPS). The RPS is a synchronization value used during the computation of the dynamic pin. The RPS is a pseudo-randomly generated string of 160 bytes in length: $RPS = RPB_1 RPB_2 RPB_3 \ldots RPB_{159}, \ where\ RPB_i\ is\ a\ byte.$

Image challenge. The image challenge is the set $ChallengeImages = (SecretImages \cup NonSecretImages), with\ cardinality\ |q + p|$ where: (i) $SecretImages \subset secretSequence, with\ cardinality\ |SecretImages| = q,$ a subset of images selected from the $secretSequence \subset C$ that contains the image password objects selected by the user at registration; and (ii) $NonSecretImages = C \setminus SecretSequence, with\ cardinality\ |NonSecretIamges| = p,$, where the relative complement of $secretSequence$ in C is the set of elements in C, but not in $secretSequence$: $C \setminus secretSequence = \{o \in C \dashv |o \notin secretSequence\}$.

***Security strength of the challenge and usability*. Fig. 1**(a) shows an example of a challenge represented as a grid of icons where $q = 5, p = 20$, and $n = 25$ (the greyed images represent the secret).

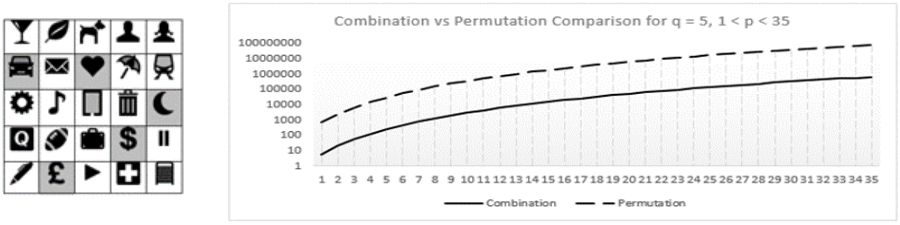

Fig. 1. (a) Example of an image challenge, (b) Combination vs. Permutation Functions

The security strength of the challenge depends on the values of p and q, and on the mode in which the user is asked to recognize the secret images. In the combination mode images are recognized in any order: $n!/(n-q)!\,(q)!$. In permutation mode images are recognized according to the sequence registered: $n!/((n-q)!)$. In both cases, $n = q + p$. **Fig. 2**(b) illustrates, in logarithmic scale, the speed at which the number of possible combinations (and therefore the security strength) for a challenge with $q = 5$ increases for different values of p ($1 < p < 35$) for the permutation and combination modes. As illustrated, permutation mode provides higher security over combination mode. However, such increase in security is inversely proportional to usability since it is easier for the user to recognize the 5 secret images without having to recall the exact sequence. **Table 1** compares the modes for different p and q.

Table 1. Comparison combination vs. permutation for different p and q

q	p	n	*Combination mode*	*Permutation mode*
5	20	24	10626	255024
5	20	25	53130	6375600
6	20	26	230230	1.66E+08
4	30	34	46376	1113024
5	30	35	324632	38955840
6	30	36	1947792	1.4E+09

Rules-based challenge generation, usability vs security assurance. A challenge is generated at runtime taking into account: client device constraints, contextual factors and risk associated, the level of authentication assurance required, and usability.

Client device constraints. A client device constraint is defined as a tuple $(device\_id, parameter, p)$ where p (see **Table 1**) is an estimate of the size of the challenge. For instance, a laptop has a larger screen than a smartphone and can display a challenge with a larger number of image objects $n = q + p$,

1. $(smartphone_{abc},\ screen_{size} = 5 \text{ inch}, 20)$

Contextual rules. This refers to contextual factors that carry a level of risk. A contextual rule is defined as a tuple $(context_{parameter_1}, \dots, context_{parameter_x}, risk_{level})$

where $risk_{level}$ is a value between 0 and 1. For example, consider an employee authenticating to a corporate server and the following contextual rules:

2. $(location = HOME, time = ANY, risk_{level} = MEDIUM = 0.5)$

In the proposed system, the level of risk defines the strength of the challenge – *challenge rules*; and the number of authentication factors required –*authentication factor rules*.

Challenge rules. A challenge rule is defined as a tuple $(q, p, challenge_{mode}, assurance_{level})$. where $assurance_{level}$ is a value between 0 and 1 For example consider the following rules for $p = 20$ (see **Table 1**):

3. $(5, 20, mode = unordered, assurance_{level} = 0.3)$
4. $(6, 20, mode = unordered, assurance_{level} = 0.5)$
5. $(4, 20, mode = ordered, assurance_{level} = 0.5)$

Notice that rules 4 and 5 are given the same $assurance_{level}$. This is because the number of possible combinations and permutations for these two rules are of similar order of magnitude, 230230 and 255024(see **Table 1**).

Authentication factor rules. An authentication factor rule is defined as a tuple $\left(number_{authentication_{factors}}, risk_{level}\right)$ and indicates the number of authentication factors required given some risk level. For instance:

6. $\left(authentication_{factors} = 1,\ risk_{level} = LOW = 0.2\right)$
7. $\left(authentication_{factors} = 2,\ risk_{level} = MEDIUM = 0.5\right)$

Example. Assume rules (1) and (2) are applicable. In such case, then rules (4) and (5) would hold true, i.e. p=20 and $assurance_{level} \geq risk_{level}$. And between (4) and (5), (4) would be the optimal choice since it mitigates the present level of risk and provides the best option in terms of usability, i.e. unordered recognition mode. Rule (4) enforces p=20 and q=6 to generate the challenge. Similarly, rule (7) would hold true, i.e. $assurance_{level} \geq risk_{level}$, and enforces the use of at least two authentication factors, i.e. the challenge itself, and an additional factor, e.g. IMSI.

4.2 User response to the challenge**.**

The user responds to the challenge by selecting the secret images. The algorithm then retrieves the secret images' seeds and performs an XOR operation over them whose result is a pin of 4 hexadecimal digits (hex_i) in length: $UserPIN = seed_{1'} \oplus seed_{2'} \oplus seed_{3'} \oplus \ldots \oplus seed_{|secretImages|} = hex_1 hex_2 hex_3 hex_4$, where each element $seed_{i'}$ corresponds to an element $seed_i \in seedsVector$ (i.e. $seed_{i'} \xrightarrow{f} seed_i$). The value $UserPIN$ along with the value RPS are taken as input parameters to the cryptographic transformation that calculates the dynamic pin.

4.3 Computation of the cryptographic transformation function**.**

A Substitution Box (S-Box) is a component used in cryptosystems to perform substitutions in a way that relations between output and input bits are highly non-linear. This protects against cryptanalysis. An S-Box designed to be resistant to linear and differential cryptanalysis is the Rijndael S-Box[15]. The design was made balancing security and computational efficiency. The security strength of crypto-algorithms based on S-Boxes can be improved by changing the S-Box dynamically (e.g. Blowfish). This makes more difficult to carry out an attacks without knowing what S-Box to associate to a given output. To increase the pseudo-randomness of the dynamic PIN, it is proposed to use an S-Box that can be obtained dynamically, complies with strong security design criteria and crypto-properties, and that can be generated using a deterministic technique based on parameters known to both client and server. Barkan et al. [16] show that by replacing the irreducible polynomial and the affine transformation in the Rijndael S-Box it is possible to produce dual ciphers with the same cryptographic properties of the original S-Box. This result is used here to propose an indexing technique that allows selecting different dual ciphers, i.e. S-Boxes. In the next subsections the mathematical definitions that support the formulation of the indexing technique are presented along with the proposed indexing function(s).

Rijndael S-box[15]. The Rijndael S-box is an algebraic operation that takes in an element of the Galois Field $GF(2^8)$ and outputs another element of $GF(2^8)$, where $GF(2^8)$ is viewed as the finite field $\frac{GF(2)[X]}{(X^8+X^4+X^3+X+1)}$ of polynomials over the finite field $GF(2)$ reduced modulo by the polynomial $X^8+X^4+X^3+X+1$. The operation has 2 steps: (i) find the multiplicative inverse of the input over $GF(2^8)$ (0 is sent to 0); and (ii) apply the affine transformation $Ax + b$ where x is the result of the first step, (in Rijndael) A is a specific 8×8 matrix with entries in $GF(2)$ and b is a specific vector with 8 entries in $GF(2)$, both specifically chosen to make it resistant to linear and differential cryptanalysis. Elements in $GF(2^8)$ are represented as bytes and transformations can be pre-computed and represented as a lookup matrix.

Dual Ciphers[16]. Two ciphers E and E' are called Dual Ciphers if they are isomorphic, that is to say there exists three invertible transformations f, g, h such that $E_K(P) = f(E'_{g(K)}(h(P)))\ \ \forall P, K$, where P is the plain text and K is the key. Different cipher can be created from an original cipher while keeping the original's algebraic properties because of the isomorphism.

Square Dual Cipher of the Rijndael S-box[16]. If the constants of the Rijndael S-box (denote the Rijndael S-box E) are replaced such that: (i) it is replaced A with A^2 where A^2 is not simply the square of the matrix, it is equal to QAQ^{-1} where Q is an 8×8 matrix chosen such that $Qx = x^2$ for all x. As a side result this also means that $QAQ^{-1}x = (Ax)^2$; and (ii) b is replaced with b^2. Hence it can be shown that these transformations result in a dual cipher (let it be denoted E^2). It can be seen that $A^2x + b^2 = QAQ^{-1}x + Qb = Q(A(Q^{-1}x) + b)$. Hence making these transformations (and creating the square dual cipher) is equivalent to applying a pre and post matrix multiplication on the original Rijndael S-box. This same transformation can be applied to E^2 to obtain E^4 and similarly for $E^8, E^{16}, E^{32}, E^{64}$ and E^{128} ($E^{256} = E$).

Modifying the polynomial of the Rijndael S-box[16]. Recall that the first operation of the Rijndael S-box is to find the multiplicative inverse of the input over $GF(2^8)$. There are a total of 30 irreducible polynomials of degree 8, of which the Rijndael selected $X^8+X^4+X^3+X+1$. As different fields $GF(2)[X]/f(X)$ for different irreducible polynomials $f(X)$ of the same degree are isomorphic, there exist a linear transformation which can be represented as a binary matrix R such that R takes an element of the Rijndael case and outputs an element of the new case with the changed polynomial. The matrix R is of the form $R = (1,\ a\ ,\ a^2, a^3,\ a^4,\ a^5,\ a^6, a^7\ )$ where $a^{i'}s$ are computed modulo the new irreducible polynomial. Hence R can be used in the same way as Q was used in the previous: applying a pre and post matrix multiplication on the original Rijndael S-box $R(S(R^{-1}x))$. As there are 30 irreducible polynomials, each of which has the 8 squared ciphers this totals $8 \times 30 = 240$ different dual ciphers. In the book of Rijndaels [17] the 240 dual ciphers of Rijndael are presented including the matrices Rs and the $R^{-1}s$. Here they are used in the following way on the original Rijndael S-box to create a new S-box: $S_{new}(X) = R \times SRijndael(R^{-1} \times X)$, with $SRijndael$ the original Rijndael S-Box matrix.

Indexing the dual ciphers of the Rijndael S-box. In the proposed scheme, an indexing technique is used for the 240 distinct dual ciphers of Rijndael (in [18] the number of possible dual ciphers has been extended to 9120). An indexing function is defined, i.e. $index_{new}$, to determine what dual cipher, out of the 240, to use to generate a new S-Box, i.e. $S_{new}(X)$. $index_{new}$ takes as input authentication tokens $AuthNToken_i$, the seeds associated to $SecretImages$ and $NonSecretImages$, and the index value of the last successful authentication. The proposed indexing function has two variants dependent recognition mode: permutation or combination.

Let $DualCipherMatrices = \big((R_1, R_1^{-1}), (R_2, R_2^{-1}), \dots, (R_{240}, R_{240}^{-1})\big)$ be the vector of Dual Ciphers' matrices $(R_{index}, R_{index}^{-1})$ where $0 \leq index < 240$. The two indexing function are defined as follows: (combination mode) $index_{new} = (index_{current} + \sum_{i=1}^{l} AuthNToken_i + \sum_{SecretImages} seeds + \sum_{NonSecretImages} seeds)\ mod\ 240$; and (permutation mode) $index_{new} = (index_{current} + \sum_{i=1}^{l} AuthNToken_i + \sum_{k=1}^{q=|SecretImages|} k \times seed_k + \sum_{NonSecretImages} seeds)\ mod\ 240$, with $t_{index_{current}} < t_{index_{new}}, with\ time\ t$.

In permutation mode each $seed_k$ is multiplied by the index k forcing the result to depend on order. Both variants of $index_{new}$ output an integer between 0 and 239 that is used to select $(R_{index_{new}}, R_{index_{new}}^{-1}) \in DualCipherMatrices$, and to determine the new S-Box transformation: $S_{new}(X) = R_{index_{new}} \times SRijndael(R_{index_{new}}^{-1} \times X)$. $S_{new}(X)$ takes as input a byte and outputs another byte.

4.4 Generation of Dynamic PIN (***DynPIN***).

The dynamic pin, $DynPIN$, is generated by performing iterative substitutions through the transformation function, $S_{new}(X)$, using the values $UserPIN$ and RPS.

Recall that $UserPIN = hex_1 hex_2 hex_3 hex_4$, where$0 \ll hex_i \ll F$, is 2 bytes long, that is, each hexadecimal digit hex_i is a nibble (half byte), and that $RPS = RPB_1 RPB_2 RPB_3 \dots RPB_{159}$, is 160 bytes long. Each byte RPB_i is composed by a

more significant and a less significant part, $nibble\ RPB_i{}^H\ and\ nibble\ RPB_i{}^L$. The dynamic pin is defined $DynPIN = byte_1 byte_2 byte_3 byte_4$.

Each $byte_i$ digit is computed as the result of a chain of 7 substitutions between $UserPIN$ (2 bytes long) and 2 bytes of RPS, and 7 iterations through the s-box $S_{new}(X)$. **Fig. 2** shows the sequence of substitutions to produce $byte_1$. In the diagram each arrow indicates one iteration through $S_{new}(X)$. During each iteration, $S_{new}(X)$ takes as input one byte consisting of two nibbles: a hexadecimal digit of $UserPIN$ and a nibble of RPB_i; and outputs a new byte, hereafter S_i. The following are the 7 iterations performed to generate $byte_1$:

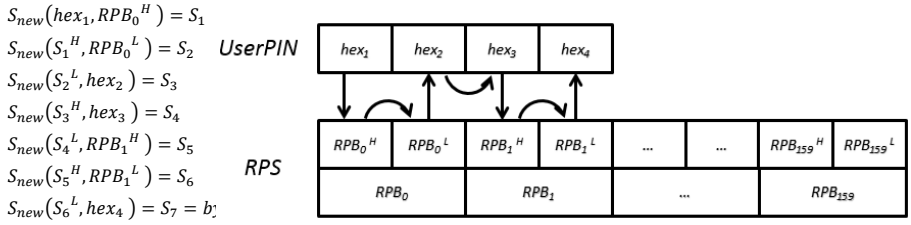

Fig. 2. Chain of substitutions and S-Box iterations to generate $byte_1$ of $DynPIN$

The same process is repeated for the other digits, $byte_2, byte_3, and\ byte_4$, but using a different starting byte of RPS for each digit. To achieve this, the RPS is split into 4 quarters (each one 40 bytes long), and the previous digit ($byte_{i-1}$) of $DynPIN$ viewed as an integer, 0 to 255, and reduced modulo 40, is used to determine a starting byte in RPS. The starting byte in RPS for $byte_i$ is calculated as follows: $Z = (40 * j) + Y$, where $Y = |(byte_{i-1})_{10}|_{40}$. Z is the starting byte in the RPS used to calculate $byte_2\ (j = 1), byte_3 (j = 2), and\ byte_4 (j = 3)$. Thus the starting byte in the RPS is randomized within each quarter block of the RPS. Once all the digits are computed, the dynamic pin is sent to the server for validation. The server executes the same algorithm and verifies the dynamic pin sent by the user.

5 Conclusions and Future Work

This paper presented a context-based multi-factor authentication system based on a dynamic PIN. A novel crypto-function has been proposed that changes dynamically based on the user input, history of authentications, and authentication factors at the client device. The dynamic aspect of the crypto-function increases the pseudo-randomness of the scheme and provides strong protection against cryptanalysis. The scheme generates a challenge based on context, takes into account risk and tunes assurance vs. usability criteria. In addition to the standard client-server workflow scenario, the proposed scheme has been implemented as a software security token that displays the Dynamic PIN and the user types it on a web interface. The synchronization between client and server via the web interface uses QR-Codes and the client device's camera. The prototype is being validated.

References

1. Xiaoyuan, S., Ying, Z., Owen, G.S.: Graphical passwords: a survey. In: Computer Security Applications Conference, 21st Annual, pp. 10 pp.-472. (Year)
2. Catuogno, L., Galdi, C.: A Graphical PIN Authentication Mechanism with Applications to Smart Cards and Low-Cost Devices. In: Onieva, J., Sauveron, D., Chaumette, S., Gollmann, D., Markantonakis, K. (eds.) Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks, vol. 5019, pp. 16-35. Springer Berlin Heidelberg (2008)
3. Bardram, J., Kjær, R., Pedersen, M.: Context-Aware User Authentication – Supporting Proximity-Based Login in Pervasive Computing. In: Dey, A., Schmidt, A., McCarthy, J. (eds.) UbiComp 2003: Ubiquitous Computing, vol. 2864, pp. 107-123. Springer Berlin Heidelberg (2003)
4. Corner, M.D., Noble, B.D.: Protecting applications with transient authentication. Proceedings of the 1st international conference on Mobile systems, applications and services, pp. 57-70. ACM, San Francisco, California (2003)
5. Jakobsson, M., Shi, E., Golle, P., Chow, R.: Implicit authentication for mobile devices. Proceedings of the 4th USENIX conference on Hot topics in security, pp. 9-9. USENIX Association, Montreal, Canada (2009)
6. Hayashi, E., Das, S., Amini, S., Hong, J., Oakley, I.: CASA: context-aware scalable authentication. Proceedings of the Ninth Symposium on Usable Privacy and Security, pp. 1-10. ACM, Newcastle, United Kingdom (2013)
7. http://www.safenet-inc.com/products/data-protection/two-factor-authentication/gold-challenge-response/
8. Aloul, F., Zahidi, S., El-Hajj, W.: Two factor authentication using mobile phones. In: Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on, pp. 641-644. (Year)
9. Lamport, L.: Password authentication with insecure communication. Commun. ACM 24, 770-772 (1981)
10. Dodson, B., Sengupta, D., Boneh, D., Lam, M.: Secure, Consumer-Friendly Web Authentication and Payments with a Phone. In: Gris, M., Yang, G. (eds.) Mobile Computing, Applications, and Services, vol. 76, pp. 17-38. Springer Berlin Heidelberg (2012)
11. Gianluigi, M., Pirro, D., Sarrecchia, R.: A mobile based approach to strong authentication on Web. In: Computing in the Global Information Technology, 2006. ICCGI '06. International Multi-Conference on, pp. 67-67. (Year)
12. Wen-Bin, H., Jenq-Shiou, L.: Design of a time and location based One-Time Password authentication scheme. In: Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International, pp. 201-206. (Year)
13. Soare, C.A.: Internet Banking Two-Factor Authentication using Smartphones (2012)
14. Eldefrawy, M.H., Khan, M.K., Alghathbar, K., Kim, T.-H., Elkamchouchi, H.: Mobile one-time passwords: two-factor authentication using mobile phones. Security and Communication Networks 5, 508-516 (2012)
15. Daemen, J., Rijmen, V.: AES Proposal: Rijndael. submitted to the Advanced Encryption Standard (AES) contest (1998)
16. Barkan, E., Biham, E.: In How Many Ways Can You Write Rijndael? In: Zheng, Y. (ed.) Advances in Cryptology — ASIACRYPT 2002, vol. 2501, pp. 160-175. Springer Berlin Heidelberg (2002)
17. Barkan, E., Biham, E.: The book of Rijndaels. Cryptology ePrint Archive, Report 2002/158 (2002), http://eprint.iacr.org/2002/158
18. Raddum, H.: More Dual Rijndaels. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) Advanced Encryption Standard – AES, vol. 3373, pp. 142-147. Springer Berlin Heidelberg (2005)