A new keyed hash function based on latin squares and error-correcting codes to authenticate users in smart home environments

Hussain Ahmad$^{1[0000-0001-6959-8352]}$ and
Carolin Hannusch$^{1[0000-0002-0098-7293]}$

Department of Computer Science
Faculty of Informatics
University of Debrecen
Kassai út 26
4028 Debrecen, Hungary
hussain.ahmad@inf.unideb.hu
hannusch.carolin@inf.unideb.hu

Abstract. We introduce the scheme of a keyed hash function, also called message authentication code (MAC) based on previously given latin squares and binary linear error-correcting codes. We investigate the properties of the introduced scheme regarding its security and applicability. Further, we give a possible application in a smart home environment, especially for opening the entrance door to a house by using a mobile device.

Keywords: Keyed hash function · message authentication code · smart home · latin square · error-correcting codes

1 Introduction

1.1 Motivation

Smart homes have become more and more present in our lives. Parallel to the development of smart home environments, their security also needs to be considered [11]. The aim of the current paper is to introduce a keyed hash function (message authentication code MAC) based on a given latin square and a fixed binary linear error-correcting code. We want to apply the introduced hash function to authenticate users in a smart home environment.

1.2 Preliminaries

Latin squares have a wide literature, as well as error-correcting codes have. We direct the reader for some fundamental literature about latin squares to [12] and about linear codes to [18]. Recently, a program package named Torch was introduced [9]. This program package can be used for the fast generation of linear codes and it can be implemented in a software. Hash functions have played an important role in cryptography for the last decades. There exists a huge literature about hash functions. We direct the reader for some fundamental literature about hash functions to [13] and [15]. An overview of cryptographic non-keyed hash functions is given in [23] and some literature about cryptographic keyed hash functions (message authentication codes) can be found in [19], [20]. Several approaches for unkeyed hash functions have been made [6], [7], [8], [24] as well as for keyed hash functions [10], [21], [25], for identification in small devices [1], using latin squares in hash functions [5], [17], even regarding critical sets of latin squares [2] and applying error-correcting codes in hash functions [4]; to mention some results of the last years without claim of completeness.

2 Definitions and Notations

Latin squares A **latin square** L is an $n \times n$ square containing n distinct symbols exactly once in each row and each column. Then each row and each column of L determines a **permutation** of the **symmetric group** S_n. We denote the permutations determined by the rows by $\sigma_1, \ldots, \sigma_n$ and the permutations determined by the columns by $\tau_1, \ldots, \tau_n$. The cardinality of S_n is $n! = 1 \cdot 2 \cdot \ldots \cdot n$. The number of latin squares is known for $n \leq 11$ (A002860 in [16]).

Error-correcting codes A **binary linear code** of length N is a subspace of the N-dimensional vector space over the finite field with two elements $\mathbb{F}_2$. A linear code is called **error-correcting code** if it can correct at least 1 error, which occured in the communication channel. The distance of two codewords is the number of distinct coordinates. The **minimum distance** of a linear code is the minimum of all occuring distances between its codewords. It is well known, that if the minimum distance of a linear code is d, then the code can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors. The number of all k-dimensional subspaces of $\mathbb{F}_2$ is $\binom{N}{k}_2$, where $\binom{N}{k}_2$ denotes the Gaussian polynomial coefficient ([14], p. 698).

3 Protocol of the hash function

In this section, we give the protocol for a new hash function based on a given latin square and a given binary linear code.

Input Let L be a latin square of order n. Let $m = (m_1, \ldots, m_k)$ be a binary string of length k. Then let C be a binary linear code of length N and dimension k and let G be a generator matrix for C.

Computing the hash value Choose a set $I_1 \times I_2 \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$ and compute

$$\rho := \prod_{i \in I_1} \sigma_i \prod_{j \in I_2} \tau_j .$$

Then compute $(mG)^\rho$ (which means multiply m with G and apply the permutation ρ on the columns of the product).

Output Finally, determine its value in the binary number system. Thus

$$hash(m) = (mG)^{\rho}_{BINARY}.$$

4 Properties of the hash function

According to Section 9.2.2 in [15] a good hash function has to fulfill the following properties: deterministic, pre-image resistance, second pre-image resistance and strong collision resistance. In this section, we investigate our hash function introduced in Section 3 regarding these properties:

Deterministic Since G and ρ are fixed in the protocol, it follows that $m_1 = m_2$ implies $hash(m_1) = hash(m_2)$.

Pre-image resistance Let us assume $y = hash(m)$ is given. In order to find m, we need to find ρ, for which we have $n!$ possibilities and G, for which there exist $\binom{N}{k}_2$ possibilities. Thus, if n is large enough, then it is computational infeasible to compute m from y.

Second pre-image resistance Given m_1 and $hash(m_1)$, we want to find m_2, such that $hash(m_1) = hash(m_2)$. We denote $m_1 = (a_1, \ldots, a_k)$ and $m_2 = (b_1, \ldots, b_k)$ and

$$G = \begin{pmatrix} g_{1,1} & \cdots & g_{1,N} \\ g_{2,1} & \cdots & g_{2,N} \\ \vdots & & \\ g_{k,1} & \cdots & g_{k,N} \end{pmatrix}.$$

Thus we need to solve

$$(m_1G)^{\rho} = (m_2G)^{\rho}$$

and therefore we have to solve the following system of binary equations.

$$\sum_{i=1}^{k} a_i g_{i,j} = \sum_{i=1}^{k} b_i g_{i,j} \ \forall j = 1, \ldots, N \tag{1}$$

Since $g_{i,j}$ is fix and $g_{i,j} \in \{0,1\}$ for each $1 \le i \le k$ and $1 \le j \le N$, there is no relation between a_i and b_i which is true for each $1 \le i \le k$. Thus it is computational infeasible to determine m_2 from m_1.

Strong collision resistance In order to find a pair of messages m_1 and m_2, such that $hash(m_1) = hash(m_2)$, we need to solve the system of equations 1. Again, there is no relation between two distinct solutions of these equations, therefore it is computational infeasible to find such a pair if n is large enough.

5 Application in Smart Home environment

Imagine, Bob arrives home, but he forgot his key at his workplace. Unfortunately, Alice is also out for the whole evening, so Bob cannot enter the house. Bob just grabs his smartphone in order to call Alice, when realizing that they built in a smart door system. Bob turns on the display, which will show a latin square. Then Bob will open the corresponding software (app) on his mobile device and scan the latin square. The software knows the necessary keys, so it can compute Bob's hash value. Since Bob is an authorized person for entering the house, the hash value will be accepted by the smart door system and the system will finally open the door (see Figure 1).

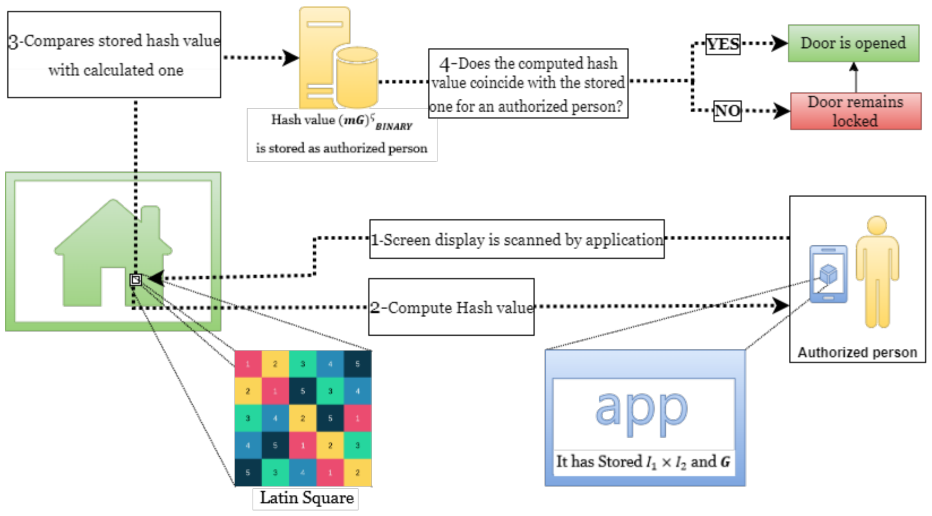
3-Compares stored hash value with calculated one
Hash value $(mG)^{\varsigma}_{BINARY}$ is stored as authorized person
4-Does the computed hash value coincide with the stored one for an authorized person?
YES
NO
Door is opened
Door remains locked
1-Screen display is scanned by application
2-Compute Hash value
Authorized person
app
It has Stored $I_1 \times I_2$ and G
Latin Square

Fig. 1. Application in smart door system

Keysize The matrix G contains $n \cdot k$ entries, the cardinality of the set $I_1 \times I_2$ is at most n^2, the permutation ρ can be stored as a product of cycles moving at most n elements.

Storage For a message (identifier) m we have $1 \leq hash(m) \leq 2^N - 1$, which has to be stored for each authorized person t-times, if each authorized person has t identifiers. Furthermore, we need to store latin squares, sets $I_1 \times I_2$ and the generator matrices. How many we store of each of these, is actually due to the implementation and can be customized for individual needs.

Computational capability The mobile device has to compute ρ from the latin square, i.e. it must be able to compute the product of $2n$ permutations in acceptable time. Further, it has to compute mG, which is a vector-matrix multiplication, and apply the permutation ρ to mG, which means mixing up the coordinates of mG. Finally it has to compute the value of $(mG)^{\rho}$ in binary number system, which means it must be able to compute $\sum_{i=0}^{N-1} a_i \cdot 2^i$ in acceptable time, where $a_i \in \{0,1\}$.

6 Security of the smart door system

Let us assume that each authorized person has $m_1, \ldots, m_t$ identifiers of length k. We denote the hash values of the identifiers by

$$H_1 = (m_1G)^{\rho}_{BINARY}$$

$$\vdots$$

$$H_t = (m_tG)^{\rho}_{BINARY}.$$

We have

$$1 \leq H_i \leq 2^N - 1$$

for each H_i, where $i = 1, \ldots, t$. Then the security of the smart door system can be scaled by increasing t and k.

Every linear code has a minimum distance. Within the known bounds [3] we can choose the minimum distance d. Since we want C to be an error-correcting code, we need choose $d \geq 4$. The used linear code C, as well as its generator matrix G should be kept as a secret, which implies that d is also kept as a secret. Therefore, an unauthorized person who wants to attack the hash function has to find the generator matrix G. The number of possible matrices G is $\binom{N}{k}_2 \cdot n! \cdot k!$, since $\binom{N}{k}_2$ is the number of all k-dimensional subspaces of $\mathbb{F}_2^N$ and in each generator matrix permuting the rows and columns changes the matrix.

Actually, the number of all possible latin squares is not known for $n > 11$. Therefore if $n > 11$ and the latin square L of order n is kept as a secret, then it is computational infeasible for an unauthorized person to find L. In fact, for the protocol of the hash function, we do not need to know L, but just the permutation ρ. A permutation can be tried in brute-force attack and since $\rho \in S_n$, we have $n!$ possibilities for ρ. Since the latin square L can be seen on the display also by unauthorized persons, $I_1 \times I_2$ has to be kept as a secret necessarily.

Note that the security of the introduced protocol can be customized according to computational capability of devices, since we can increase n and N theoretically to infinity.

7 Conclusion and future work

We introduced a protocol for a keyed hash function (message authentication code), whose security is based on well-known mathematical facts. The introduced hash function fulfills all required properties for a good and useful hash function. Its strength is its security, which can be customized to actual needs for protection against brute-force attacks in accordance with computational capability of mobile and small devices. It's the task of future work to create a software in order to realize an application for mobile devices in combination with a smart home environment. Using the Torch program package [9] we can create generator matrices for binary linear codes. The scanning of latin squares can work similarly to the scanning of QR codes [22]. It's up to further research work to combine these results efficiently and finalize a software of the introduced smart door system.

References

1. Ayebie, E. B., Assidi, H., Souidi, E. M.: An efficient identification scheme based on rank metric. In International Symposium on Foundations and Practice of Security pp. 273-289, Springer, Cham. (2020)
2. Chum, C. S., Zhang, X.: Applying Hash Functions in the Latin Square Based Secret Sharing Schemes. In Security and Management, pp. 197-203 (2010)
3. Conway, J. H., Sloane, N. J. A.: A new upper bound on the minimal distance of self-dual codes. IEEE Transactions on Information Theory, 36(6), pp. 1319-1333 (1990)
4. Cramer, R., Damgård, I. B., Döttling, N., Fehr, S., Spini, G.: Linear secret sharing schemes from error correcting codes and universal hash functions. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 313-336, Springer, Berlin, Heidelberg (2015)
5. Ghosh, R., Verma, S., Kumar, R., Kumar, S., Ram, S.: Design of hash algorithm using Latin square. Procedia Computer Science, 46, pp. 759-765 (2015)
6. Gnatyuk, S., Kinzeryavyy, V., Kyrychenko, K., Yubuzova, K., Aleksander, M., Odarchenko, R.: Secure hash function constructing for future communication systems and networks. In International Conference of Artificial Intelligence, Medical Engineering, Education pp. 561-569, Springer, Cham. (2018)
7. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In 30th USENIX Security Symposium (USENIX Security 21), pp. 519-535 (2021)
8. Hannusch, C., Horváth, G.: Properties of Hash Functions based on Gluškov Product of Automata. J. Autom. Lang. Comb., 26(1-2), pp. 55–65 (2021).
9. Hannusch, C., Major, S. R.: Torch: Software Package For The Search Of Linear Binary Codes. In 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS) pp. 103-106, IEEE (2022)
10. Hussain, S. S., Farooq, S. M., Ustun, T. S.: Analysis and implementation of message authentication code (MAC) algorithms for GOOSE message security. IEEE Access, 7, pp. 80980-80984 (2019)
11. Huszti, A., Kovács, S., Oláh, N.: Scalable, password-based and threshold authentication for smart homes. International Journal of Information Security, pp. 1-17 (2022)
12. Keedwell, A. D., Dénes, J.: Latin squares and their applications. Elsevier (2015)
13. Knuth, D. E.: The art of computer programming, Vol. 3. Sorting and searching (1973)
14. MacWilliams, F. J., Sloane, N. J. A.: The theory of error correcting codes (Vol. 16). Elsevier (1977)

15. Menezes, A. J., Van Oorschot, P. C., Vanstone, S. A.: Handbook of applied cryptography. CRC press (2018)
16. OEIS Foundation Inc. (2022), The On-Line Encyclopedia of Integer Sequences, Published electronically at http://oeis.org
17. Pal, S. K., Kapoor, S., Arora, A., Chaudhary, R., Khurana, J.: Design of strong cryptographic schemes based on latin squares. Journal of Discrete Mathematical Sciences and Cryptography, 13(3), pp. 233-256 (2010)
18. Pless, V., Brualdi, R. A., Huffman, W. C.: Handbook of coding theory. Elsevier Science Inc. (1998)
19. Preneel, B.: Cryptanalysis of message authentication codes. In International Workshop on Information Security, pp. 55-65, Springer, Berlin, Heidelberg (1997)
20. Preneel, B., Van Oorschot, P. C.: On the security of iterated message authentication codes. IEEE Transactions on Information theory, 45(1), pp. 188-199 (1999)
21. Ramadhani, F., Ramadhani, U., Basit, L.: Combination of Hybrid Cryptography In One Time Pad (OTP) Algorithm And Keyed-Hash Message Authentication Code (HMAC) In Securing The Whatsapp Communication Application. Journal of Computer Science, Information Technology and Telecommunication Engineering, 1(1), pp. 31-36 (2020)
22. Rouillard, J.: Contextual QR codes. In 2008 The Third International Multi-Conference on Computing in the Global Information Technology (iccgi 2008) (pp. 50-55). IEEE (2008)
23. Sobti, R., Geetha, G.: Cryptographic hash functions: a review. International Journal of Computer Science Issues (IJCSI), 9(2), 461 (2012)
24. Teh, J. S., Alawida, M., Ho, J. J.: Unkeyed hash function based on chaotic sponge construction and fixed-point arithmetic. Nonlinear Dynamics, 100(1), pp. 713-729 (2020)
25. Wang, Y., Chen, L., Wang, X., Wu, G., Yu, K., Lu, T.: The design of keyed hash function based on CNN-MD structure. Chaos, Solitons and Fractals, 152, 111443 (2021)