The Libyan Journal of Science (International Journal): Volume 18, 2014/2015

Adding Robustness to Libyan National Identification Number (LNIN) System

Abdulraheem A. Beraam* and Lutfi S. El-Lhweje

Department of Computer, Faculty of Science, University of Tripoli
*E-mail: a_beraam@tripoliUniv.edu.ly

Abstract

Errors can be introduced into data through a variety of means. Libyan National Identification Number (LNIN) which lately has been introduced and became the source of identifying Libyan citizen is no exception. However, the LNIN was constructed without any base for validation. The other vulnerability is the weak authentication process when individuals inquire their LNIN especially via web site. One can increase reliability by introducing check digits to the right end of the LNIN. Check digit is generally designed to detect and to capture human transcription errors and to protect against corruption in the number itself (i.e., a single mistyped digit or some permutations of two successive digits). The introduction of check digits does not mean secrecy. Our goal is twofold; review the concept and the benefit of check digit, where several well known algorithms will be presented and propose a strategy to smoothly fix the LNIN by adapting an algorithm which satisfies the requirements of the LNIN, and strengthen the authentication process before revealing sensitive information via web site.

المستخلص

إدخال أخطاء ممكن في البيانات من خلال مجموعة متنوعة الوسائل ورقم التعريف الوطني والذي تم اخيرا إدراجه كوسيلة ومصدر لتحديد هوية المواطن الليبي ليس استثناء. هذا الرقم تم تمصميمه دون أي وسيلة لاكتشاف الاخطاء والتي قد تحدث أثناء التعامل معه أو أي قاعدة للتحقق من صدقيته خلال إدراجه بالعديد من المعاملات التي تهم المواطن. أضف إلى ذلك ضعف الحماية في الحصول على البيانات الشخصية خصوصا عن طريق الموقع الالكتروني المخصص لذلك. بالامكان تحصين رقم التعريف الوطني بإضافة خانة رقمية تعرف برقم المطابقة وهو رقم رقابي للتحقق من صحة الرقم الوطني ويمنع ويسهل ويسرع من عملية المطابقة للرقم دون الحاجة لقاعدة البيانات المخزن بها بيانات المواطنين. إضافة رقم المطابقة لرقم التعريف الوطني لايعني السرية. الهدف من هذه الورقة مزدوج: استعراض مفهوم رقم المطابقة وكيفية الاستفادة منه بإدراجه ضمن رقم التعريف الوطني واقتراح استراتيجة لاصلاح هذا الرقم عن طريق تسخير خوارزمية والتي تمكن من الاستفادة منه وتلبي المتطلبات وتعزَز عملية التحقق للتأكد من

Accepted for Publication: 27/12/2015

هوية الاشخاص قبل الكشف عن معلومات حساسة من خلال الموقع الالكتروني على شبكة الانترنت.

Keywords**:** Libyan national identification number (LNIN); check digit; transcription errors; algorithm; validation.

Introduction

Lately we have been hearing of Libyan National Identification Number (LNIN), and how such a number can improve services to the Libyan citizens, reduce redundancy and help the government against corruption in the welfare system, child benefit, and multi-salary abusers. With the LNIN, services like identity cards, passports, family booklets, driving licenses, voting, banking and many other services can be carried out more quickly, rationally and reliably. However the current LNIN structure has a design fault that may hinder the ambition of the government to integrate the LNIN for a variety of services. Taking into consideration that most government agencies, to be optimistic, if not all, are still paper-based systems, even agencies that use information technologies are still lagging behind in their computerized systems to be fully integrated. This is further complicated by the bad infrastructural state such as communications. To improve or to overcome such deficiency, a check-digit must be added to the LNIN without requiring data base access for validation.

The Structure of the LNIN

The LNIN is a fixed format, unique identification number, that consists of 12 numeric digits containing the following consecutive components, from left to right:

1. Gender code: 1 digit {1 = male | 2 = female}
2. Birth year: 4 digits (e.g. 2011)
3. Sequential no.: 7 digits.

Modern identification numbers such as LNIN should serve at least three functions:

1. The number should be unique and unambiguously identify the person with whom it is associated.
2. The number should not reveal any personal information about the citizen (i.e., the number generated will be devoid of any classification based on caste, creed, religion, gender or geography).
3. The number should have a “self-checking” aspect.

Unfortunately the LNIN satisfies only the first function. Besides, when inserting the LNIN into any requesting digital form, no validation can be done against corruption in the number itself, because the design of LNIN format does not provide such service to capture such errors.

On the other hand, aquiring the LINN via web site [2] shows weakness in the authentication process when requesting sensitive personal information (breach of confidentiality). The web site form (http://info.nid.gov.ly/InfoNid/reqID.aspx) requests only registration number and birth year. Authentication is the process of determining whether someone is in fact, who is declared to be or not [3]. This authentication weakness allows amateurs, crackers, and malicious hackers [4] to exploit this weakness and obtain the LINN of other citizens.

Check Digit Background

When human beings use numbers (e.g., typing them on a keyboard, dialing them on telephones, or reading them and telling them to others), they tend to make certain kinds of mistakes more often. According to Hamming [5], the two most common human errors are:

1. Interchanging adjacent digits of numbers (xy → yx): 54 become45.
2. Doubling the wrong one of a triple of digits (xxy → xyy): 667 become 677.

Kirtland [6] cited errors which include the following:

1. Single digit errors (x → y): 1 becomes 2.
2. Twin errors (xx → yy): 11 become 22.
3. Jump transpositions errors (xzy → yzx): 132 become 231.
4. Jump twin errors (xzx → yzy): 131 become 232.
5. Similar pronunciation (e.g., “sixty" to "sixteen"): 60 become 16.

To overcome or to reduce such errors, a check digit [7], also known as a checksum character has been in use for many years in variety of identification systems (e.g., Bank Account Numbers, Credit Card, ISBN, and UPC). It consists of a single digit (sometimes more than one) computed by an algorithm from the other digits (or letters) in the sequence input. If a number n has the digits $d_k, d_{k-1}, \ldots, d_2, d_1$ (i.e., $n = \sum_{i=1}^{k} d_i 10^{i-1}$), the secured number then, which has the digit representation $d_k, d_{k-1}, \ldots, d_2, d_1, p$ where p is the computed check-digit.

Notable Generic Algorithms

Various check digit methods have been designed. Each method is able to detect all “single-error mistakes”, but fail to detect all transposition errors. At least one

erroneous transposition is undetectable with these methods [8]. In choosing check digit algorithm, a high probability of catching errors is traded off against implementation difficulty. In this paper we will review only notable simple generic algorithms (e.g., Luhn-1954 [9]), which do not catch as many errors as complex ones, and those which require sophisticated algorithms (e.g., Verhoeff [10]; Damm [11]), that catch all single digit substitution, transposition errors, and many complex errors but not all. To reduce this failure rate, it is necessary to use more than one check digit [12] (e.g., International Bank Account Number IBAN uses modulo 97-10 two digits). However, several studies have shown that the number of errors roughly doubles when more than one check-digit is used [13]. Therefore, we will review only one of the check digit schemes mentioned above since it is good enough for our purpose (see appendix A for details of these schemes).

In addition, all these schemes calculate the check digit by performing a series of mathematical operations on the digits that precede the check-digit, and can be validated offline or as JavaScript embedded in client side of a web application (i.e., no online database needed for validation). For example, the barcode reader's decoder in the supermarket calculates the check digit by performing a series of mathematical operations on the digits that precede the check-digit, and compares the result of the calculation to the value of the check-digit. Typically, if the check-digit does not match the result of the calculation, the reader emits a signal (such as a beep) or warning to acknowledge that the results do not match.

Strengths and Weaknesses of the Algorithms

Luhn algorithm [9] is easily understood and implemented. The algorithm will detect any single digit error, as well as almost all transposition of adjacent digits. Luhn algorithm suffers from detecting adjacent transposition errors (x0 → 0x); 90 become 09. However, it is well known and used in major credit cards such as VISA card, automatic teller machines cash withdrawal cards, and more.

The Verhoeff's algorithm [10] detects all occurrences of what Hamming [5] noted. Additionally, the algorithm detects most occurrences of what Kirtland [6], cited, but the scheme does not catch most jump twin errors involving digits with difference of 5 such as 050 vs. 505, 161 vs. 616, 272 vs. 727, 494 vs. 949, but it catches 383 vs. 838. The main weakness of the algorithm is its complexity. Unlike Luhn algorithm, the calculations required for the Verhoeff's check-digit cannot readily be performed by hand from memory.

The Damm's algorithm [11] has desirable features similar to Verhoeff algorithm. However, the scheme does not catch 383 vs. 838. Despite its desirable Adding

properties, the algorithm also has the benefit of using one table. Its main weakness is largely unknown as it is newly published in 2004.

Check digit algorithm adds the self validation attribute to any validation functions.

Proposed Solution to LNIN

The LNIN system can be strengthened by including a check-digit to minimize human transcription errors. To reduce errors rate, fix the vulnerability to LNIN design format, and fix authentication process when revealing personal information via web site (it is very important to authenticate people in a secure way). We propose a road map which includes:

1. A check-digit must be appended to the right of the LNIN which does not change the LNIN previously issued to the public.
2. No change to LNIN database is required since the check-digit is a calculated value.
3. The choosen check digit algorithm must be made public (availabile), so all government, public agencies, and financial institutions can benefit by incorporating the algorithm to validate and reduce the common human transcription errors when requesting the LNIN.
4. Alert the public to obtain the LNIN with a check-digit via SMS or the web site. However the previous methods of informing the public with their LNIN must be changed to incorporate more stringent authentication by requesting more information about what has been previously asked for privacy reasons (e.g. first name, surname or family reference number) because registration numbers are sequential, it is very easy to infer (guess) others registration number and birth year from your own.
5. Once a check digit algorithm is adopted, government, public agencies and financial institutions can benefit by incorporating the check-digit algorithm in their computerized system to validate the LNIN even if the LNIN database is offline. Validating LNIN by the client can improve application responsiveness, especially if the user interface (UI) is remote from the server. Users do not have to wait for the server to respond when they enter or submit invalid LINN, and the server does not need to attempt to process data that it will later reject. However, even if the LNIN is validated by the client or in the UI, it should always be revalidated on the server or in the receiving service. This protects against malicious users who may circumvent client-side validation and submit invalid data.

Conclusion

Check digit algorithms are generally designed to capture human transcription errors. It is not intended to be a cryptographically secure hash function; it was designed to protect against *typing errors* during data entry, not malicious attacks. The LNIN format in use now must be fixed to include a check-digit to provide validation against corruption when typing the LNIN into electronic forms. Government sectors, public agencies and financial institutions can benefit by incorporating public check digit algorithm without requiring data base access for validation in their computerized system. However, it is cost effective to provide robustness and eliminate the large number of duplicate and fake identities in government and private databases having LNIN database online when the governmental infrastructure permits. Finally we would recommend the Damm's algorithm over the Verhoeff's algorithm to be adapted for LNIN for several reasons (reliable, low errors rate, single check-digit, employs one table, simple to implement, and is quite efficient). On the other hand, incorporation of stringent authentication prevents the weaknesses from being exploited and increases confidentiality when information is online.

References

[1] Cole, E., Krutz, R., Conley, J., Reisman, B., Ruebush, M., Gollmann, D. and Reese, R. (2008). Network Security Fundamentals. John Wiley, p. 7.
[2] Civil Status Bureau, Libyan Interim Gov., http://info.nid.gov.ly/InfoNid/reqID.aspx
[3] Pfleeger, C. P. and Pfleeger S. L. (2007). Security in Computing, 4th ed., Prentice-Hall, p. 21-23.
[4] Salomon, D. (2005). Coding for Data and Computer Communications, Springer Verlag, p. 56.
[5] Hamming, R. (1986). Coding and Information Theory, 2nd ed., Prentice-Hall, p. 27.
[6] Kirtland, J. (2001). Identification numbers and check digit schemes, Mathematical Association of America, p. 153.
[7] Wagner, N. and Putter, P. (1989). Error detecting decimal digits, CACM, **32(1)**, 106-110.
[8] Chu, C. K. (1981). A note on multiple error detection in ASCII numeric data communication, J. Ass. Comput. Mach., **28**, 265-269.
[9] Luhn, P. H. (1960). Computer for Verifying Numbers, U.S. Patent 2,950,048.
[10] Verhoeff, J. (1969). Error detecting decimal codes, Mathematical Centre Tract 29, the Mathematical Centre, Amsterdam,
[11] Damm, H. M. (2007). Totally anti-symmetric quasigroups for all orders $n \neq 2,6$, Discrete Mathematics, **307(6)**: 715–729.

[12] Sethel, A. S., Rajaraman, V. and Kenjale, P. (1978). An error correcting scheme for alphanumeric data, Inform. Process. Lett., vol 7, pp. inform. theory, vol. **IT-13**, 102-105.

[13] Gumm, P. H. (1985). A new class of check digit methods for arbitrary number system. IEEE Trans. On Information Theory, **IT-13**, 102-105.

[14] Damm, H. M. (2004). Total anti-symmetrische quasigruppen. Philipps-Universität Marburg, urn:nbn:de:hebis:04-z2004-05162.

Appendix A

A.1 Luhn algorithm

The algorithm also known as mod 10 algorithm, was created by Hans Peter Luhn [9], a scientist at IBM. The algorithm is in the public domain and is in wide use today. It is a simple checksum formula used to validate a variety of identification numbers, such as credit card numbers and many government identification numbers. The algorithm operates on the digits in a right-to-left manner. Assume a number ***N*** has the digits $d_k, d_{k-1}, \ldots, d_2, d_1$. Here is the calculation of the check-digit algorithm:

1. For all odd position *i* From right to left
   Set $d_i = d_i * d_i$
   If $d_i > 9$ set$d_i = d_i - 9$
2. Set$checksum = (\sum_{i=1}^{k} d_i)\ mod\ 10$
3. *check-digit* = If *checksum* $\neq 0$ then$10 - checksum$ else *0*

A.2 Verhoeff algorithm

Verhoeff [10] proposed a scheme that avoids the weakness of the widely used Luhn algorithm. The solution is based on multiplication in the dihedral group D5 (the dihedral group of order 10), which is not commutative (i.e., a*b is not always equal to b*a). In practice, however, the scheme would normally be implemented using pre-computed lookup tables (multiplication table D, permutation table P, and an inverse table INV).

D =
((0,1,2,3,4,5,6,7,8,9),(1,2,3,4,0,6,7,8,9,5),(2,3,4,0,1,7,8,9,5,6),(3,4,0,1,2,8,9,5,6,7),
(4,0,1,2,3,9,5,6,7,8),(5,9,8,7,6,0,4,3,2,1),
(6,5,9,8,7,1,0,4,3,2),(7,6,5,9,8,2,1,0,4,3),
(8,7,6,5,9,3,2,1,0,4), (9,8,7,6,5,4,3,2,1,0))

P = ((0,1,2,3,4,5,6,7,8,9),(1,5,7,6,2,8,3,0,9,4),(5,8,0,3,7,9,6,1,4,2),(8,9,1,6,0,4,3,5,2,7), (9,4,5,3,1,2,6,8,7,0),(4,2,8,6,5,7,3,9,0,1),(2,7,9,3,8,0,6,4,1,5),(7,0,4,6,9,1,3,2,5,8))
INV = (0,4,3,2,1,5,6,7,8,9)
Assume a number ***N*** has the digits $d_k, d_{k-1}, \ldots, d_2, d_1$. The following algorithm will perform the Verhoeff's check-digit calculation.

1. Set *checksum* to *zero*
2. For *all* d_i From right to left :*checksum* = *D*[*checksum*][*P*[(*i*+*1*) *mod 8*][d_i]
3. Set *check-digit* = *checksum*

A.3 Damm Algorithm

The Damm's algorithm is similar to Verhoeffs' algorithm. It was presented by Damm in 2004 [14]. It is essential part is a quasigroup of order 10 with the special feature of being totally anti-symmetric. Damm revealed several methods to create such TA-quasigroups of order 10 and gave some examples in his doctoral dissertation [14]. In practice, however, the scheme would normally be implemented using pre-computed lookup table DMMtbl.

DMMtbl=((0,3,1,7,5,9,8,6,4,2),(7,0,9,2,1,5,4,8,6,3),(4,2,0,6,8,7,1,3,5,9),(1,7,5,0,9, 8,3,4,2,6),

(6,1,2,3,0,4,5,9,7,8),(3,6,7,4,2,0,9,5,8,1),(5,8,6,9,7,2,0,1,3,4),(8,9,4,5,3,6,2,0,1,7), (9,4,3,8,6,1,7,2,0,5),(2,5,8,1,4,3,6,7,9,0))

Assume a number ***N*** has the digits $d_1, d_2, \ldots, d_{k-1}, d_k$. Using DMMtbl table, the following algorithm will perform the Damm's check-digit calculation.

1. Set *checksum* to *zero*
2. For all d_i from left to right:*checksum* = *DAMMtbl*[*checksum*][d_i]
3. Set *check-digit* = *checksum*