International Journal of Security and Its Applications
Vol. 10, No. 3 (2016), pp.139-150
http://dx.doi.org/10.14257/ijsia.2016.10.3.13

A Secure Score Report Implemented in a Spreadsheet without Privacy Concerns

Dongchang Yoo[1], Minyoung Bae [1] and Yongjin Yeom[1, 2]

[1]Department of Mathematics, [2]Department of Financial Information Security
Kookmin University, 77 Jeongneung-Ro, Seongbuk-Gu, Seoul, 136-702, Korea
{yoodc8167, mypear, salt}@kookmin.ac.kr

Abstract

Students' educational records such as exam score and academic achievement should be considered as sensitive information. According to the federal student privacy law, individual scores should not be posted publicly. If a secure online management system is well-equipped in the school, then it is easy to distribute individual data in a privacy preserving way. However, in general, such a centralized system costs high and is not flexible enough to be used for instant score report after each exam. In this paper, we propose a practical way for an instructor to post students' individual exam scores online in a single file. By implementing a cryptographic hash function together with score data in an MS Excel file, we demonstrate a score report from which allows each student to retrieve his/her score with his/her own password. Based on our worksheet as a template, it is easy for instructors to write their own score reports without relying on any heavy management systems. Since our score report is implemented in MS Excel worksheets, students find out their scores with Excel program or viewers even in their mobile phones without installing any other programs or apps. Also, it is cryptographically as secure as underlying hash function SHA-256.

***Keywords**: privacy preserving, hash function, SHA-256, Excel built-in function*

1. Introduction

The importance of privacy is increasing rapidly and cannot be emphasized too much [14, 15]. That is true even for reporting exam scores or sending individual messages on achievement in the school. Students' education records should be handled with care as sensitive information and the administration department of a school or college has to protect the records from unexpected disclosures so that only approved people can access them. According to the Family Educational Rights and Privacy Act (FERPA) [5] in the United States, student's grade is very sensitive personal information. For instance, an instructor may not provide a verbal or written reference for a student that discusses the student's educational achievement unless the instructor has written permission from the student. On the other hand, students tend to expect instant feedbacks from the instructors particularly after an exam. It is recommended to have a system which holds educational data securely and readily provides each student with his/her data in a privacy preserving way. In Korea, since 2003, the National Education Information System, very huge system called NEIS [6], has kept records on all students up to high school and has being managed by the government. Each college has its own educational system mostly operated by the password based access control. Such a centralized system is expensive to adopt and manage and not so flexible that we have to learn the system as designed at the beginning. Additionally, we concern about privacy violations in IT outsourcing environment that someone from the system maintenance company may access sensitive data without any legal permissions.

In this paper, our main interest is how to report or post score records in a privacy preserving way. Instead of relying on the heavy system, it is better for instructors to design and write score sheets by themselves and post the results on their blogs or on the course web site without privacy concerns. We are going to describe how it can be done with a single Excel file so that each student can download the score file then pick up his/her score with his/her own password as Figure 4. In fact, the file from the bulletin on the web site contains score reports for all students but each record is encrypted with the individual passwords. Consequently, students can decrypt their own data only. In order to do that, we implement a cryptographic hash function within an Excel worksheet and enable the worksheet to identify and authenticate users.

According to an article on the Washington Post [13], data walls in public schools may violate the federal student privacy law, FERPA. In fact, data walls usually include lists of students and their test scores which show academic achievement. Even in the college, after exams, it is common to post score reports on the bulletin board of the department or on the wall of instructor's office. Then privacy violations may occur by taking photos and spreading them via social network services. Impatient students are willing to readily obtain their scores after each exam. In that case, our proposal shows a clear solution. Since the method we propose is implemented using MS Excel, students can find out their exam scores confidentially even in their mobile devices with Excel or viewers without installing any additional applications.

In addition, our implementation of cryptographic algorithms with built-in functions of Excel is an illustration to show every step of cryptographic algorithms without any cumbersome programming. It can be used for educational purpose to teach how the hashing algorithms works. As for block ciphers, Chok *et al*. [4] present how to implement DES [8] and AES [7] algorithms using spreadsheets.

The structure of the remaining part of the paper is organized as follows: In Section 2, we introduce a hash function SHA-256 and describe how to implement this within an Excel file. An example score report will be constructed in Section 3 and the security of our scheme is considered in Section 4. In the conclusion, the summary of our result is provided and future works are suggested.

2. Secure Hash Algorithm SHA-256 in an Excel Worksheet

2.1 Secure Hash Algorithm SHA-256

A cryptographic hash function is an algorithm which transforms an input message of any length into the output of fixed size. For a given output, it is not feasible that one can find its corresponding input. SHA-256 [1] is one of the most popular hash functions as a standard algorithm published by the NIST. The hashing algorithm can be divided into two stages: preprocessing and hashing computation. In the preprocessing stage, the input message of arbitrary length is transformed into 512 bit blocks by padding and parsing. The computing stage refers to the compression function which consists of 64 iterations of the round function depicted in Figure 1.

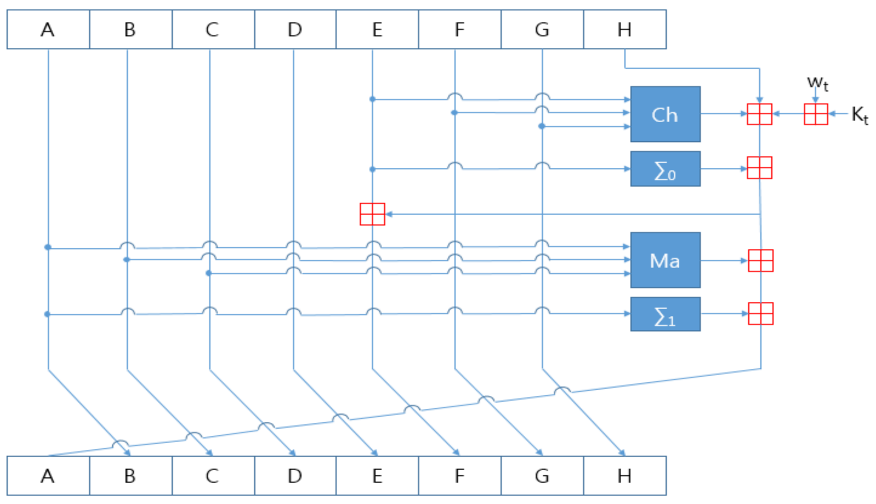


Figure 1. One Iteration in a SHA-256 Compression Function

The blue components in Figure 1 perform the following operations, equation (1), (2), (3) and (4). And the red square symbol represents addition modulo 2^{32}.

$$\mathrm{Ch}(\mathrm{E},\mathrm{F},\mathrm{G}) = (\mathrm{E} \wedge \mathrm{F}) \oplus (\neg \mathrm{E} \wedge \mathrm{G}) \tag{1}$$

$$\mathrm{Ma}(\mathrm{A},\mathrm{B},\mathrm{C}) = (A \wedge B) \oplus (\mathrm{A} \wedge \mathrm{C}) \oplus (\mathrm{B} \wedge \mathrm{C}) \tag{2}$$

$$\sum 0\ (\mathrm{A}) = (\mathrm{A} >>> 2)\ \oplus (\mathrm{A} >>> 13) \oplus (\mathrm{A} >>> 22) \tag{3}$$

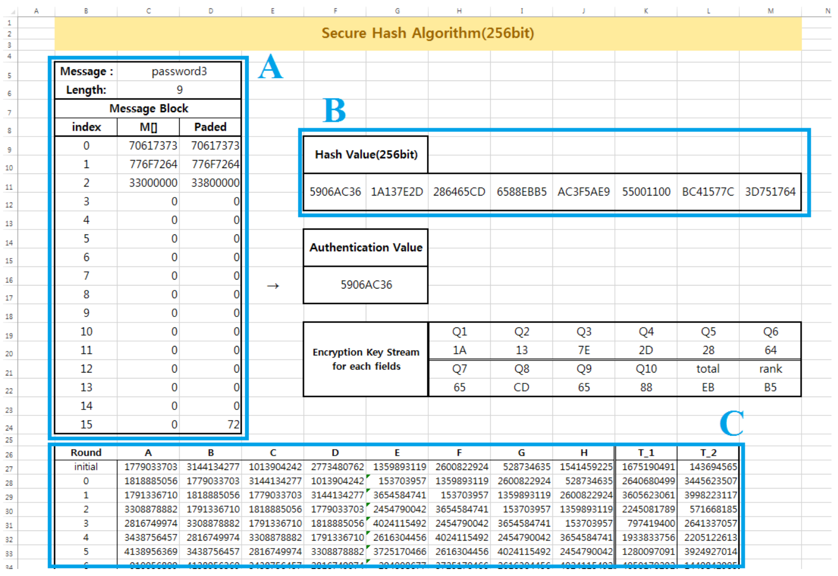
$$\sum 1(\mathrm{E}) = (\mathrm{E} >>> 6)\ \oplus (\mathrm{E} >>> 11) \oplus (\mathrm{E} >>> 25) \tag{4}$$

When a password is hashed by SHA-256, then it is not feasible for an attacker to recover the password from the 256 bit hash value. If the server holds hash values instead of users' passwords, as in Linux systems, the hacker who steals the stored hash values of passwords cannot extract any information on the passwords by reversing the process of the hash function. The only way to attack them is the exhaustive search of possible passwords, called dictionary attack. If users choose strong passwords for themselves, the server login procedure is cryptographically as secure as its hash function.

2.2 Implementing SHA-256 in an MS Excel

We implement SHA-256 algorithm within an Excel worksheet as Figure 2. The input message is entered in the cell D5 and the 256 bit hash value can be written in the eight consecutive cells F15:M15. The message padding and expansion is performed in the box A of Figure 2. We can express the 64 steps of the compression function in the box C.

Each operation in the compression function is implemented merely using Excel built-in functions such as IF(), MOD() and VLOOKUP(). If you use MS Excel version 2013 or later, you can implement it easily using bitwise operation functions such as BITAND(), BITXOR(), and BITLSHIFT(), referenced by [12]. The required constants in each step can be directly referred to the cells located on the right of the box C. If we enter a new message in D5, then all the procedures for SHA-256 are automatically recalculated by Excel so that the cells F15:M15 holds the new hash value. Here are some examples of implementations in MS Excel 2013.



Secure Hash Algorithm(256bit)

Message :	password3
Length:	9

Message Block

index	M[]	Paded
0	70617373	70617373
1	776F7264	776F7264
2	33000000	33800000
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	72

Hash Value(256bit)

5906AC36	1A137E2D	286465CD	6588EBB5	AC3F5AE9	55001100	BC41577C	3D751764

Authentication Value
5906AC36

Encryption Key Stream for each fields						
	Q1	Q2	Q3	Q4	Q5	Q6
	1A	13	7E	2D	28	64
	Q7	Q8	Q9	Q10	total	rank
	65	CD	65	88	EB	B5

Round	A	B	C	D	E	F	G	H	T_1	T_2
initial	1779033703	3144134277	1013904242	2773480762	1359893119	2600822924	528734635	1541459225	1675190491	143694565
0	1818885056	1779033703	3144134277	1013904242	153703957	1359893119	2600822924	528734635	2640680499	3445623507
1	1791336710	1818885056	1779033703	3144134277	3654584741	153703957	1359893119	2600822924	3605623061	3998223117
2	3308878882	1791336710	1818885056	1779033703	2454790042	3654584741	153703957	1359893119	2245081789	571668185
3	2816749974	3308878882	1791336710	1818885056	4024115492	2454790042	3654584741	153703957	797419400	2641337057
4	3438756457	2816749974	3308878882	1791336710	2616304456	4024115492	2454790042	3654584741	1933833756	2205122613
5	4138956369	3438756457	2816749974	3308878882	3725170466	2616304456	4024115492	2454790042	1280097091	3924927014
6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Figure 2. SHA-256 Sheet

Implementation of the equation (1) with only Excel built-in functions:

$\mathrm{Ch}(E, F, G) = (E \wedge F) \oplus (\neg E \wedge G)$ is reformed as follow,

$$\mathrm{BITXOR}(\mathrm{BITAND}(E, F), \mathrm{BITAND}(\mathrm{BITXOR}(E, 0xffffffff), G)) \qquad (5)$$

Implementation of the equation (3) with only Excel built-in function:

$\sum 0(A) = (A >>> 2) \oplus (A >>> 13) \oplus (A >>> 22)$ is reformed as follow,

$$\mathrm{temp}_1 = BITOR(BITRSHIFT(A, 2), BITLSHIFT(BITAND(A, (2^2) - 1), 30))$$

$$\mathrm{temp}_2 = BITOR(BITRSHIFT(A, 13), BITLSHIFT(BITAND(A, (2^{13}) - 1), 19))$$

$$\mathrm{temp}_3 = BITOR(BITRSHIFT(A, 22), BITLSHIFT(BITAND(A, (2^{22}) - 1), 10))$$

And then

$$\mathrm{BITXOR}(\mathrm{BITXOR}(\mathrm{temp}_1, \mathrm{temp}_2), \mathrm{temp}_3) \qquad (6)$$

The equations (2) and (4) can be implemented in a similar manner.

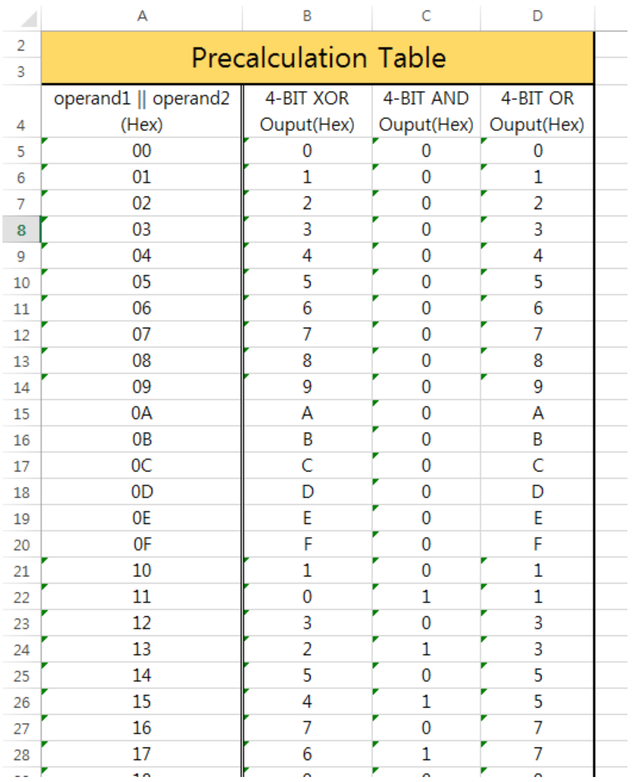
Components of SHA-256 for each round, equations (1), (2), (3) and (4), can be implemented in the MS Excel built-in function form. In other hands, if you have the MS Excel version more previous than 2013, you must implement function BITXOR(), BITOR(), BITAND() by using pre-calculation table and function BITRSHIFT() by division as follows.

2.2.1 Implementations of Bit-wise AND, XOR and OR Functions: This bit-wise operations are not supported by MS Excel 2010 or earlier versions. Therefore, pre-calculations are necessary to implement bit-wise operations as Figure 3. For Example, bit-wise XOR of 0x1 and 0x4 is implemented with a table for 4 bit data as follow [4].

$$0x1 \oplus 0x4 \rightarrow \mathrm{VLOOKUP}(0x1\ \&\ 0x4, \mathrm{TABLE\_RANGE},\ 1,2, \mathrm{TRUE}) \qquad (7)$$

The function VLOOKUP() performs seek a value in the table. The first parameter of this function means the value we want to find, the second indicates where the table is, the third is the index of the column for the first parameter, and the last parameter is the index of the column for return value. The above example, as a result, returns 0x5 at B25 cell by the equation (7). The ‘&’ is in the first parameter means the concatenation.

Note that if a larger table (bigger than Figure 3) is prepared for calculating bit-wise operations on more bits at a time, the execution time and the code size in a cell can be reduced. However, the size of the score file becomes significantly larger.



	A	B	C	D
2-3	Precalculation Table			
4	operand1 || operand2 (Hex)	4-BIT XOR Ouput(Hex)	4-BIT AND Ouput(Hex)	4-BIT OR Ouput(Hex)
5	00	0	0	0
6	01	1	0	1
7	02	2	0	2
8	03	3	0	3
9	04	4	0	4
10	05	5	0	5
11	06	6	0	6
12	07	7	0	7
13	08	8	0	8
14	09	9	0	9
15	0A	A	0	A
16	0B	B	0	B
17	0C	C	0	C
18	0D	D	0	D
19	0E	E	0	E
20	0F	F	0	F
21	10	1	0	1
22	11	0	1	1
23	12	3	0	3
24	13	2	1	3
25	14	5	0	5
26	15	4	1	5
27	16	7	0	7
28	17	6	1	7

Figure 3. Pre-Calculation Table for 4 Bit XOR Operations

2.2.2 Implementations of BITRSHIFT () and BITLSHIFT() Functions: These can be simply implemented by just dividing or multiplying by a proper number. For example, when 0x12345678 is shifted rightward by 4 positions, just we divide it by 16. As an application, the rotation operation in equation (3) and (4) can be implemented by combining these functions.

The worksheet in Figure 2 can be thought as a hash function and copied into any Excel files which need to perform the hash function SHA-256. In the next section, we are going to make use of it for user authentication and data encryptions based on the user's password. Of course, one can create a macro function with Visual Basic for MS Office. That is an easier way to build a hash function since it is very similar to any other programming languages. However, when a user download an Excel file, anti-virus software gives a warning message against user-defined macros or Visual Basic modules. Hence, users are not willing to open such a file with warning of possibly infected by computer virus. Our implementations do not use any macros at all and use only built-in functions. Consequently, ours can be passed the scanning by anti-virus programs and safely used without any harm to the system.

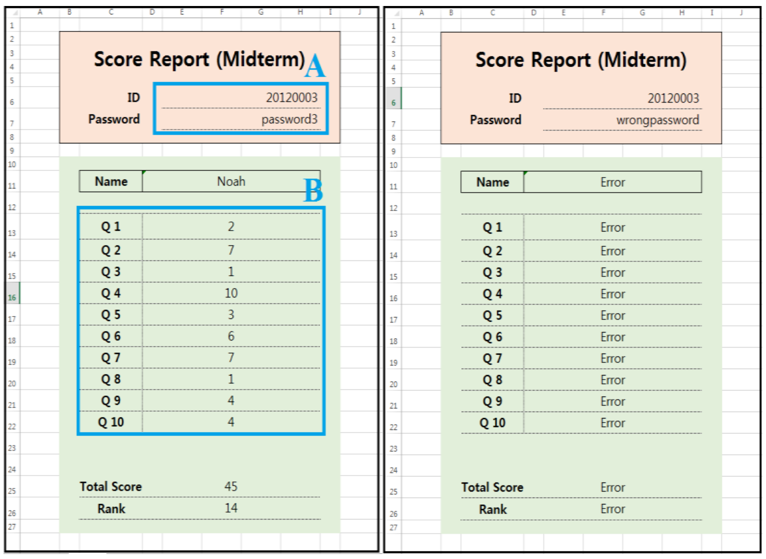
3. A Score Reporting Example

Suppose that an instructor finishes the grading in order to post the score report after an exam. Before the final scores are entered into the central management system, students are eager to see their scores determined by the initial grading as soon as possible. However, there is a privacy concern that the personal score could be exposed to the public or at least opened to the student in the same class. We propose a solution for this concern, here. Firstly, each student selects a password which is shared only between him/her and the instructor. This can be done by writing down a password together with ID and name on the test paper. Based on the hash value of the password, the scores are encrypted and filed into the worksheet. Once an Excel file with the score and information for authentication is created, the instructor posts the file on the web site. Then, students can decrypt their scores in a privacy preserving way.

3.1 Configuration of Example

How can we make such an Excel file? Let us take a close look at the score report within an Excel file. The file consists of three worksheets; cover sheet, data sheet, and SHA-256 sheet. Only one among them need to be seen to the students even though exposing all worksheets does not reduce the security strength, which does not give any clues to attackers.

3.1.1 The Cover Sheet: The cover sheet works as an interface to students. Thus the design of this worksheet is flexible and fit into the specific purpose. Basically, a student opens this file and fills out the cells corresponding to his/her ID number and password in the box A of Figure 4. The password is referred by other worksheets and used to decrypt the score which is presented in the box B. Otherwise, he/she encounters error messages as in the right of Figure 4.



Score Report (Midterm)

ID	20120003
Password	password3

Name	Noah
Q 1	2
Q 2	7
Q 3	1
Q 4	10
Q 5	3
Q 6	6
Q 7	7
Q 8	1
Q 9	4
Q 10	4
Total Score	45
Rank	14

Score Report (Midterm)

ID	20120003
Password	wrongpassword

Name	Error
Q 1	Error
Q 2	Error
Q 3	Error
Q 4	Error
Q 5	Error
Q 6	Error
Q 7	Error
Q 8	Error
Q 9	Error
Q 10	Error
Total Score	Error
Rank	Error

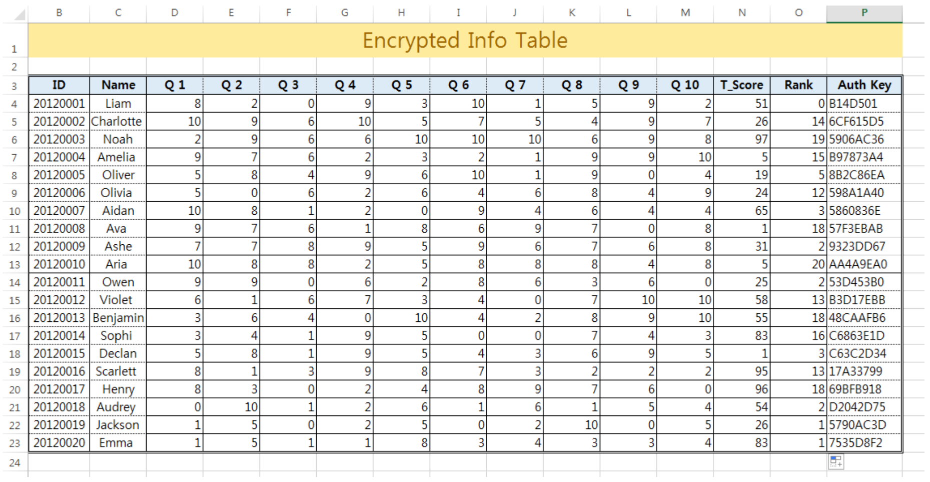
Figure 4. Cover Sheet

3.1.2 The Data Sheet: This sheet contains encrypted scores and authentication parameters. The table in Figure 5 is filled out as follows:

- From the students' password (selected at the exam, not entered in the cover sheet), calculate hash values with the help of SHA-256 sheet. The input to SHA-256 is the concatenation of the password and the nonce, where the nonce is not hidden but a unique value for this file such as the date of the exam.

  SHA-256(*password*||*nonce*)
- The first 4 bytes of hash value are stored in the last column 'Auth Key' as depicted in Figure 5. It will be used for authentication, when a password is entered in the cover sheet, to check the verification of ID is passed.
  - The rest bytes of the hash value are considered as a key stream. That is, scores are encrypted using this key stream. The encryption scheme will be explained in section 3.2.
  - As a result, each row holds the data for one person including ID, name, scores, and the

final column for authentication.



Encrypted Info Table

ID	Name	Q 1	Q 2	Q 3	Q 4	Q 5	Q 6	Q 7	Q 8	Q 9	Q 10	T_Score	Rank	Auth Key
20120001	Liam	8	2	0	9	3	10	1	5	9	2	51	0	B14D501
20120002	Charlotte	10	9	6	10	5	7	5	4	9	7	26	14	6CF615D5
20120003	Noah	2	9	6	6	10	10	10	6	9	8	97	19	5906AC36
20120004	Amelia	9	7	6	2	3	2	1	9	9	10	5	15	B97873A4
20120005	Oliver	5	8	4	9	6	10	1	9	0	4	19	5	8B2C86EA
20120006	Olivia	5	0	6	2	6	4	6	8	4	9	24	12	598A1A40
20120007	Aidan	10	8	1	2	0	9	4	6	4	4	65	3	5860836E
20120008	Ava	9	7	6	1	8	6	9	7	0	8	1	18	57F3EBAB
20120009	Ashe	7	7	8	9	5	9	6	7	6	8	31	2	9323DD67
20120010	Aria	10	8	8	2	5	8	8	8	4	8	5	20	AA4A9EA0
20120011	Owen	9	9	0	6	2	8	6	3	6	0	25	2	53D453B0
20120012	Violet	6	1	6	7	3	4	0	7	10	10	58	13	B3D17EBB
20120013	Benjamin	3	6	4	0	10	4	2	8	9	10	55	18	48CAAFB6
20120014	Sophi	3	4	1	9	5	0	0	7	4	3	83	16	C6863E1D
20120015	Declan	5	8	1	9	5	4	3	6	9	5	1	3	C63C2D34
20120016	Scarlett	8	1	3	9	8	7	3	2	2	2	95	13	17A33799
20120017	Henry	8	3	0	2	4	8	9	7	6	0	96	18	69BFB918
20120018	Audrey	0	10	1	2	6	1	6	1	5	4	54	2	D2042D75
20120019	Jackson	1	5	0	2	5	0	2	10	0	5	26	1	5790AC3D
20120020	Emma	1	5	1	1	8	3	4	3	3	4	83	1	7535D8F2

Figure 5. DATA Sheet

3.1.3 The SHA256 Sheet: The hash function SHA-256 is implemented in this sheet as described in Section 2. The input cell for SHA-256 is linked with the password cell in the cover sheet. That is, if one writes ID and password in the cover sheet, then the data are picked up by the SHA-256 sheet. Then the output cells are referred as a key stream in order to decrypt the score in the data sheet.

The decryption process is as follows:

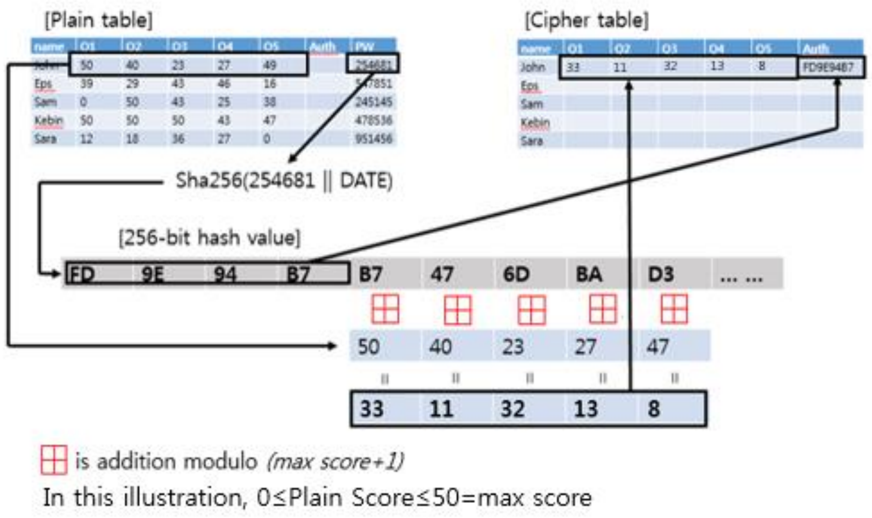
- Each student enters his/her ID and password in the cover sheet.
- The Excel program automatically picks up ID number and password from the cover sheet and calculates hash value of the password with the predetermined nonce using SHA-256 sheet. The input to SHA-256 is the concatenation of the password and the nonce,
- The first 4 bytes of the hash value are compared with the 'Auth Key' column depicted in Figure 5. If these are equal, then the verification of ID is passed.
- The rest bytes of the hash value are considered as a key stream. That is, scores are decrypted using this key stream. The decryption scheme will be explained in section 3.2.
- Decrypted scores are copied to the cover sheet.

3.2 Encryption and Decryption

According to reference [11], the strength of symmetric-key encryption algorithm based on a one-way hash function depends on the security of its underlying hash function. Because the cryptography hash function, the security of SHA-256 is guaranteed as a cryptographic standard suggested by NIST, SHA-256 based encryption scheme is trustworthy.

Encryption and decryption schemes are similarly to one-time pad or stream cipher. First, in order to convert plain scores into the encrypted form, we make key stream by using the hash function, SHA-256 with pre-shared password and the nonce. And then, we get encrypted scores by repeated byte-wise additions of two arrays, plain scores and key stream modulo (max score+1) which is the least upper bound of the score. When decrypting, we should make key stream like as encryption process. Subtract modulo (max score+1) key stream from encrypted scores. Then we can recover original scores. For example, there is a set of a student's scores, {50, 40, 23, 27, 47} where the maximum of each score is bounded by 50 points. If the hashed 256-bit value with his/her password and the nonce starts with 0xFD9E94B7 B7476DBA D317F9C6, then the five scores are successively encrypted as 50+0xB7 = 33 modulo 51, 40+0x47 = 11 modulo 51, 23+0x6D=32 modulo 51, 27+0xBA=13 modulo 51, and 47+0xD3=8 modulo 51. Unused bytes of hash value are just discarded. This example is described in Figure 6.

If an instructor want to report scores of several subjects with a single Excel file, the number of questions can be more than 28. Because the hash function we use in encryption has 32-byte output and the first 4 bytes of hash value are reserved (They are used in the authentication process), it is impossible to encrypt more than 28 questions with a hash value. We suggest to use hash chain [10] in order to generate large key stream of arbitrary length.


[Plain table]
[Cipher table]
Sha256(254681 || DATE)
[256-bit hash value]
FD 9E 94 B7 B7 47 6D BA D3 ... ...
50 40 23 27 47
33 11 32 13 8
is addition modulo (max score+1)
In this illustration, 0≤Plain Score≤50=max score

[Plain table]

name	Q1	Q2	Q3	Q4	Q5	Auth	PW
John	50	40	23	27	49		254681
Eps	39	29	43	46	16		5[illegible]7851
Sam	0	50	43	25	38		245145
Kebin	50	50	50	43	47		478536
Sara	12	18	36	27	0		951456

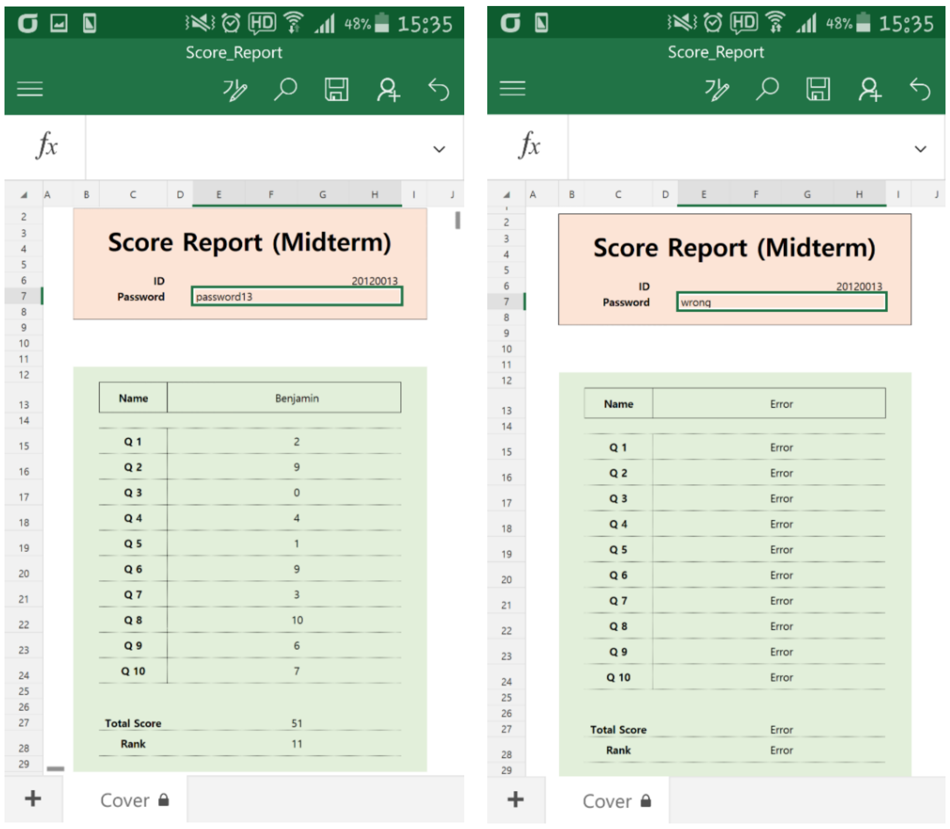
[Cipher table]

name	Q1	Q2	Q3	Q4	Q5	Auth
John	33	11	32	13	8	FD9E94B7
Eps						
Sam						
Kebin						
Sara						

Figure 6. The Illustration of the Encryption Process

3.3 Running in Mobile Devices

As for MS Excel versions, there are some differences in ways of implementation. Because bit-wise operation functions are built in MS Excel 2013, a product implemented in version 2013 has smaller size than those in earlier version. A file for Excel 2010 is even bigger 4 times than 2013 version file. When a student open a file in order to find out his/her scores in PC, he/she may not feel any differences, because it executes only one decryption process in the file. However, if students open a file in a mobile device such as smart phone, they may recognize slight difference of version-wise loading time. The more they use low-performance devices, the more feel the difference of execution time. There are several viewers supporting MS Excel 2013, which everyone can download freely from the app store for IOS or from the android market. Therefore we recommend to make the file in 2013 version for quick retrieval.



Score Report (Midterm)

ID	20120013
Password	password13

Name	Benjamin
Q 1	2
Q 2	9
Q 3	0
Q 4	4
Q 5	1
Q 6	9
Q 7	3
Q 8	10
Q 9	6
Q 10	7
Total Score	51
Rank	11

Score Report (Midterm)

ID	20120013
Password	wrong

Name	Error
Q 1	Error
Q 2	Error
Q 3	Error
Q 4	Error
Q 5	Error
Q 6	Error
Q 7	Error
Q 8	Error
Q 9	Error
Q 10	Error
Total Score	Error
Rank	Error

Figure 7. Cover Sheet Opened in a Mobile Device

4. Security Analysis

4.1. Authentication

If an attacker pretends to be a right user in order to see the scores, he have to find 4 bytes of hash value by guessing the user's password. It is infeasible to find an input which corresponds to a given output of SHA-256. However, with the probability 2-32, the attacker's guess can pass our authentication because only 4 bytes are used. This is a non-negligible possibility to success in our authentication process without knowing the correct password. The attacker with wrong password does not have the correct score in the decrypted form. In addition, he does not know whether his guessing is correct or not, either. Consequently, authentication with partial value is sufficient. Nevertheless, we can increase the number of bytes for authentication as we wish in order to increase the security strength for authentication scheme as Figure 8. Increasing the security level is not always desirable. The smaller authenticate key, the easier an attacker find an input collision in order to pass the authentication. But most of them are false positive which causes wrong decryptions. Therefore, with a small authentication key, the attacker hardly confirms that he has the correct password.

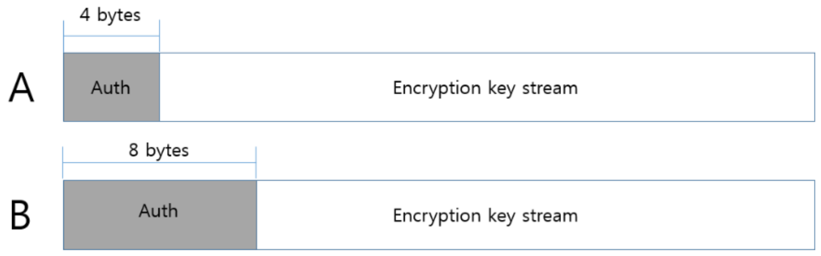


Figure 8. Usage of the Hash Value Depending on the Length of Authentication Bytes.

4.2. Encryption

The encrypted score in the table can be correctly decrypted only when the hash value is fully obtained. As for an attacker, there is no way to check his guessing is correct or not. He only knows that he passes the authentication possibly with a wrong password. One-time pad encryption with a key stream generated by the hash value is unconditionally secure unless the key stream is not reused. Therefore, the nonce is indispensable to the input of hash function. If one uses the same password repeatedly for several exams, the hash values are the same for all exams without nonce. Then from the encrypted data, some information can be leaked such as the difference of the score between exams.

In general, XOR operation is used in a symmetric encryption based on hash function. But our encryption scheme has modulo addition instead of XOR. The reason of this design is to preserve data format between plaintext and ciphertext. If the encryption is designed by using XOR, an attacker has non-negligible advantage such that his guessing for the password is correct or not. An attacker chooses a guessing password and then makes key stream with that one. We wish that he doesn't know guessing password is right or not, although he fortunately pass an authentication. But, if he gets plain scores which are more than maximum score, he can be sure that guessing password is wrong. Thus, we need format-preserving encryption scheme as [2], [3]. Hence we designed an encryption scheme by using modulo addition. As a result, any ciphertext and plaintext have values in the range from 0 to maximum score.

5. Conclusion

We have proposed a template for score reports which can be posted on the web in order for students to check their scores in a privacy preserving way. Based on ours, an instructor can efficiently and securely post students' scores with a single Excel file. Also, students can simply and rapidly find out their exam scores without installing any cumbersome programs in their PC or mobile device. The score report has three worksheets: cover sheet for user interface, data sheet for encrypted scores and key for authentication, and SHA-256 sheet. We implement a cryptographic hash function SHA-256 in a worksheet only using built-in function of Excel 2013. Merely copying SHA-256 sheet, an instructor can design the score report as he/she wants. In addition, our implementation of SHA-256 can be used for educational purpose in order to teach the structure of hash function.

Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2014M3C4A7030648)

References

[1] NIST, "Secure Hash Standard", Federal Information Processing Standards Publication 180-4, **(2015).**

[2] NIST, "Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption", NIST Special Publication 800-38G Draft, July, **(2013).**

[3] M. Bellare, P. Rogaway, T. Sies, "The FFX Mode of Operation for Format-Preserving Encryption", Draft 1.1, NIST, available at http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec.pdf, **(2010).**

[4] O-S Chok, Jayantha & Susantha Herath, "Computer Security Learning Laboratory: Implementation of DES and AES Algorithms using Spreadsheets". International Journal of Effective Management, 1(2), **(2004)**.

[5] "Family Educational Rights and Privacy Act (FERPA)", available at http://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html

[6] NEIS, "National Education Information System", available at http://neis.go.kr

[7] FIPS-197: "Advanced Encryption Standard", **(2001)**, available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[8] FIPS-46-3: "Data Encryption Standard(DES)", **(1999)**, available at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[9] D. Yoo, Minyoung Bae, Yongjin Yeom, "A Practical Online Score Reporting with an Excel File in a Privacy Preserving Way", International Workshop on IT-based Mathematical Education, **(2015)**.

[10] T. Page, "The application of hash chains and hash structures to cryptography", 4 August **(2009)**.

[11] Mohammad Peyravian, Allen Roginsky, Nev Zunic, "Hash-Based Encryption System", Computers & Security Vol.18. No.4, pp.345-350, **(1999)**.

[12] "Description of Excel built-in function", available at https://support.office.com/en-nz/home

[13] Valerie Strauss, "How some school 'data walls' violate U.S. privacy lqs", The Washington Post, available at https://www.washingtonpost.com/news/answer-sheet/wp/2014/02/21/how-some-school-data-walls-violate-u-s-privacy-law/ **(2014)** February 21.

[14] H. Yan, "Sensitive-resisting Relation Social Network Privacy Protection Model", IJSIA Vol 9, No. 8. **(2015)**.

[15] M. Park, J. H. Eom and T.-M. Chung, "Implementation of Privacy-Enhanced SMS Provider on the Android Platform", IJSIA , vol 9, no. 5. **(2015)**.

Authors

Dongchang Yoo, He has been studying Mathematics in Kookmin University, Korea, from 2007 until now. He will begin attending graduate school, Financial Information Department of Kookmin University. His research interests are design and analysis of cryptographic algorithms and random number generator.

Minyoung Bae, She has been studying Mathematics in Kookmin University, Korea, from 2012 until now. She will begin attending graduate school, Financial Information Department of Kookmin University. Her interests are cryptography and implementation.

Yongjin Yeom, He received his B.S., M.S., and Ph.D. degrees in Mathematics from Seoul National University, Korea, in 1991, 1994, and 1999, respectively. He worked at National Security Research Institute from 2000 to 2011. Since 2012, he has been working at the department of Mathematics and the department of financial information security in Kookmin University, where he is currently an associate professor. His research interests are design and analysis of cryptographic algorithms, and parallel implementations.