JURNAL INFOTEL
Informatics - Telecommunication - Electronics
Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

Smart card security mechanism with dynamic key

Noprianto[1]*, Vivi Nur Wijayaningrum[2]
[1,2] Department of Information Technology, Politeknik Negeri Malang
[1,2] 09 Soekarno Hatta Street, Malang, East Java, Indonesia
*Corresponding email: noprianto@polinema.ac.id

Received 28 March 2021, Revised 11 June 2021, Accepted 18 August 2021

Abstract — As a currently popular technology, the use of smart cards continues to increase in various fields and the rapid development of technology. Therefore, data security stored on a smart card needs to focus on avoiding misuse of data by unauthorized parties. However, it is not enough for the security mechanism to be carried out only during the communication process of sending data. Then, the mechanism for securing data on the smart card also needs to be done. In this study, a data security technique using dynamic keys is proposed by changing the key and access conditions on the smart card according to predetermined rules. Dynamic keys are a new mechanism proposed to authenticate smart cards using a different key on each card. This technique ensures that the keys used to access each smart card are different so that the risk of data duplication and modification threats can be minimized. In addition, this mechanism is a low-cost security privacy protection. The test results show that the data security technique using dynamic keys ensures that read and write access to the smart card can only be done if the keys match the rules.

Keywords – data, dynamic key, security, smart card, technology

I. INTRODUCTION

Currently, information technology is developing rapidly in line with the human need for information. There are many information technology-based media used to support daily activities, one of which is electronic payment instruments as a substitute for cash. A smart card as an electronic media is not only used for cash transactions (e-money) but can also be used as an identity card (e-identity). A smart card can also become a multi-functional card that can replace various cards such as ID cards, library cards, driving licenses, and others [1]. In addition, smart cards can perform various operations for transaction needs because of their microprocessor [2].

With the various benefits of smart cards to meet everyday human needs, smart cards are popular in various fields, such as banking. [3], academics [4], healthcare [5][6], and transportation [7]. Therefore, data security aspects on smart cards need to be considered, especially related to transaction activities using smart cards [8]. Then, it is because irresponsible parties can duplicate the data stored on a smart card or modify it [9]. Securing data on the smart card needs to be done to avoid harmful activities to the smart card owner.

Other researchers have conducted several studies discussing smart card security. For example, Kundarap et al. used cryptography to solve security problems in contactless smart cards. Security mechanisms are implemented during the communication process to prevent criminal acts such as eavesdropping, denial of service, covert transactions, and man-in-the-middle attacks. In addition, the data is not written directly to write data to the smart card, but the data is encrypted first. It is also done when data is read from the smart card. The data is decrypted first, and then the data can be displayed [10].

In another study, Akram et al. used pseudorandom number generators for cryptographic mechanisms [11]. Similar studies have also been proposed by Malina et al. regarding authentication using a zero-knowledge protocol. In this case, the card owner must prove their knowledge of the private key stored on the smart card. Next, the private key is mapped to the public user ID [12]. Most of the studies that have been conducted by several previous researchers only focus on security in communication channel protocols. Securing data stored on the smart card is also necessary.

In this study, a data security technique using dynamic keys is proposed by changing the key and access conditions on the smart card based on the UID of each smart card. Unique Identifier (UID) is an alphanumeric on the smart card and acts as the identity of each card with a size of four bytes or seven bytes, according to the type of card. Because it is the identity created by the factory when producing the card, the UID value on the card is permanent. The proposed low-cost security technique is carried out in the smart card section. Thus, the keys used by a smart card will be different from other smart cards. It aims to reduce the risk of security threats in data stored on the smart card, such as illegal modification or duplication of data.

II. RESEARCH METHODS

In this study, a card is used to determine the success or the read and write failure process before and after conditions the changing key. The process of reading data from the card depends on the type of card used. The card used in this study is Mifare 1K. Reading or writing data from the card begins with detecting the contactless tag, authenticating it. Finally, the reading or writing process on the card can be carried out at the address of a particular sector or block. The stages used in accessing the smart card are shown in Fig. 1 for writing data to the smart card and Fig. 2 for reading data from the smart card.

In Fig. 1, it is shown that to write data into a smart card. The process begins by giving a command or request from the reader to the smart card to detect the UID. This process also aims to determine the type of tag or card used, for example, Mifare 1K cards, Mifare 4K, ISO 14443-A, ISO 14443-B, or other types. In addition, this step also checks whether the card is a new card or an old card. A new card is usually used to replace an old card damaged, or its data contents cannot be read. Furthermore, if the card used is new, it is necessary to authenticate using the card's default key, key A or key B. The new card certainly has a different UID from the old card's UID because the UID of each card is different. New cards can be detected or defined when successfully authenticated using the default key, the key set when the factory manufactured the card. Generally, the default key value is FFFFFFFFFFFF or A0A1A2A3A4A5 in hexadecimal. After authenticating using the default key, the key needs to be changed based on the rule to be defined. Therefore, the UID will be obtained again. Authentication is done again using the new key that has been created. If the authentication process is successful, data writing to the smart card can be done based on certain blocks on the smart card. In this case, the old card does not require changing the key, and authentication can be used directly using the previously defined key. While on new cards, it is necessary to change the default key first to a more secure key.

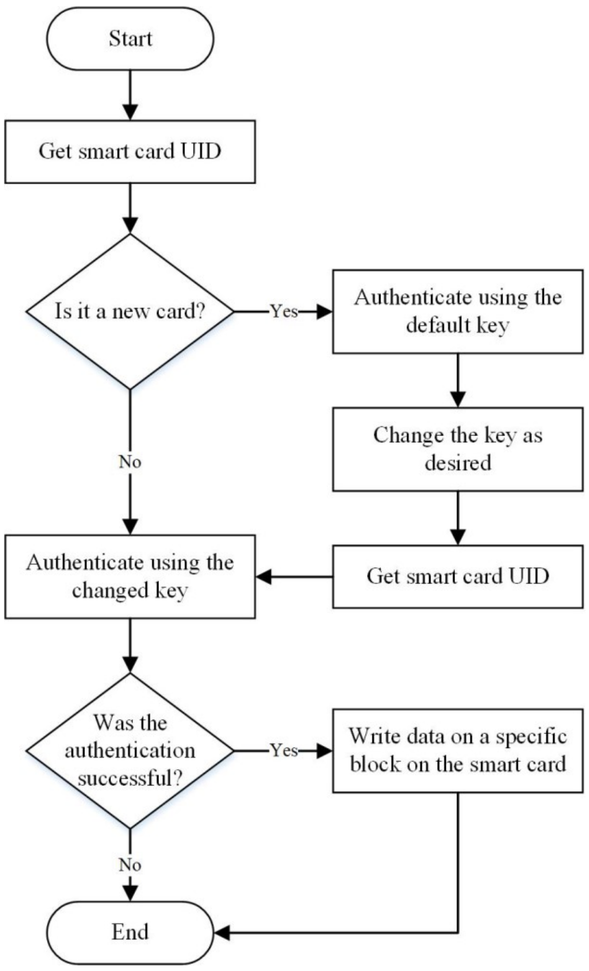

Start
Get smart card UID
Is it a new card?
Yes
Authenticate using the default key
Change the key as desired
No
Get smart card UID
Authenticate using the changed key
Was the authentication successful?
Yes
Write data on a specific block on the smart card
No
End

Fig.1. Flowchart of writing data to the smart card

The authentication process is the same as a login process on a system. This process is done by sending a read command which contains a six-byte key, the type of key used (key A or key B), and authentication for certain sectors via a smart card reader to the smart card. If the data block to be accessed is still in one sector, the authentication process is only done once, even though the block is changed. Authentication needs to be done again if the data is to be accessed in a different sector.

The process is similar to reading data from a smart card when writing data to the smart card. This process begins with getting the UID, then authenticating using a key to read data from the smart card. It is because a smart card has key A and key B with different roles. For example, key A is only used for writing data, while key B is for reading data. Furthermore, if the authentication process is successful, data reading from the smart card in a certain block can be performed. The flowchart of reading data from the smart card is shown in Fig. 2.

The process of reading or writing data on the smart card is determined by the access bit in the trailer sector. The access options consist of "never", "key A", "key B", or "key A|B" (key A or key B). Table 1 shows the access conditions for the trailer sector, and Table 2 for the data block [13].

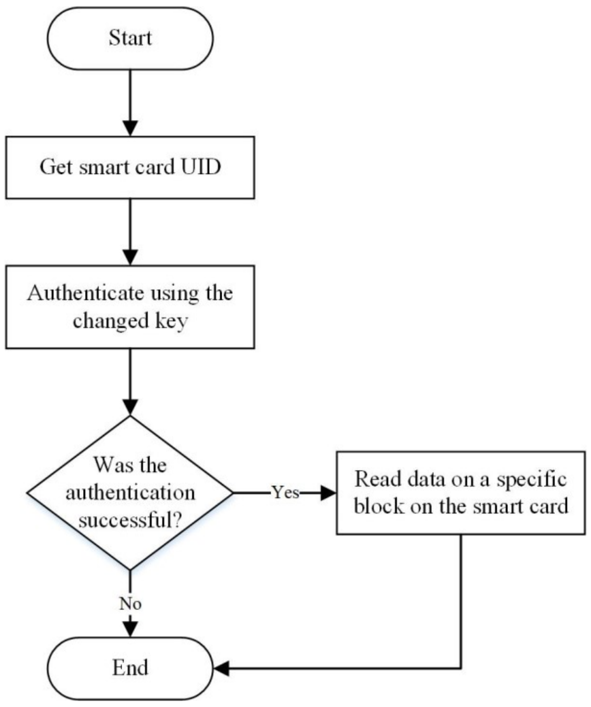

Start
Get smart card UID
Authenticate using the changed key
Was the authentication successful?
Yes
Read data on a specific block on the smart card
No
End

Fig.2. Flowchart of reading data from a smart card

Table 1. Access conditions for the sector trailer

Access bits			Access condition for					
			Key A		Access bits		Key B	
C1	C2	C3	R	W	R	W	R	W
0	0	0	N	A	A	N	A	A
0	1	0	N	N	A	N	A	N
1	0	0	N	B	A|B	N	N	B
1	1	0	N	N	A|B	N	N	N
0	0	1	N	A	A	A	A	A
0	1	1	N	B	A|B	N	N	B
1	0	1	N	N	A|B	N	N	N
1	1	1	N	N	A|B	N	N	N

Table 2. Access conditions for data blocks

Access bits			Access condition for			
C1	C2	C3	R	W	I	DTR
0	0	0	A|B	A|B	A|B	A|B
0	1	0	A|B	N	N	N
1	0	0	A|B	B	N	N
1	1	0	A|B	B	B	A|B
0	0	1	A|B	N	N	A|B
0	1	1	B	B	N	N
1	0	1	B	N	N	N
1	1	1	N	N	N	N

In Tables 1 and 2, R represents the Read operation, W represents the Write operation, I represents the Increment operation, DTR represents the Decrement, Transfer, and Restore operation. N is a symbol for "never" access, A is a symbol for "key A" access, B is a symbol for "key B" access, and A|B is a symbol for "key A|B" access.

The process of reading and writing data on the smart card can be summarized into the pseudocode shown in Algorithm 1.

Algorithm 1. Read and Write Data

```
Transmit polling command
Transmit authentication command sector n
IF success
    i = 0
    FOR i UNTIL 3
        Transmit read command block i
ELSE
    Authentication error
```

In Algorithm 1, transmit aims to send commands from the reader to the smart card using the APDU protocol. The Application Protocol Data Unit (APDU) is the communication unit between the reader and the smart card. First, the poll command is used to get the UID and detect the type of card used. Next, authentication commands are sent to specific sectors and keys. If the authentication is successful, then the block reading on the sector can be carried out. The block reading process is repeated three times because, in a card, three blocks can be filled with data in each sector, while the other block is used for the trailer sector. Another condition, if the authentication fails, then the read or writes operation cannot be performed because the key used may be wrong.

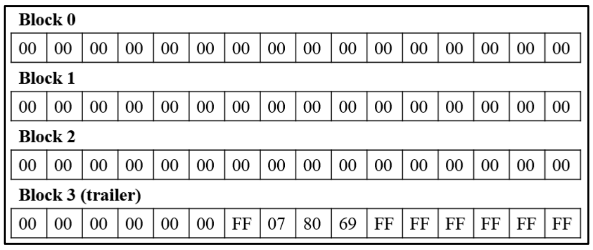
When reading or writing data is carried out, the system will respond in a series of hexadecimal numbers to declare the success or failure of the read or write process. Table 3 shows information about the error codes generated in response to the system.

Table 3. Error Codes

Error Codes (hex)	Meaning
0x00	No error
0x14	Mifare authentication error
0x01	Time out
0x90 0x00	Operation finished

III. RESULTS

In this study, the type of smart card used to store data is Mifare 1K. The data used is data from the parking application. Fig. 3 is an example of initial data on a card. In Fig. 3, it can be seen that a card consists of one sector, which contains four blocks (three data blocks and one trailer block). The data block can be used for writing or reading operations, while the trailer block can only store key A, access condition, and key B.



Block 0

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Block 1

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Block 2

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Block 3 (trailer)

00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF

Fig.3. Initial data

As shown in Fig. 3, the entire data block has a value of 00 which means that no data is stored on the card or is still empty. Whereas in the trailer block, the first six digits are key A with a value of 00 as the default value, then the next four digits are access conditions with a value of FF 07 80 69, and the last six digits are key B with the value FF as the default value.

A. *Data*

The details of the data to be written on a card are shown in Table 4.

Table 4. Data to be stored on the card

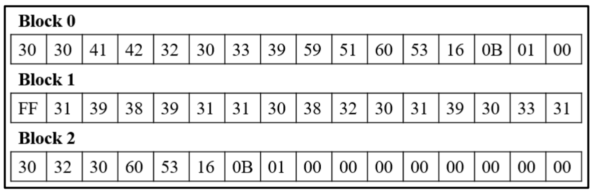
Data Attributes	Value
Vehicle license plate	AB2039YQ
Transaction date	2021-03-18 15:57:47
Login status	1
Gate code	255
Employee ID number	198911082019031020
Expired date	2021-03-18 15:57:47
Card status	1

Based on the data attributes that need to be stored on a card, data mapping is needed to determine the maximum byte length for each data attribute. It aims to maintain consistency when writing data to the card. Table 5 shows information regarding the maximum length of each data attribute with a total of 40 bytes.

Table 5. Size of Data Attributes

Data Attributes	Size (byte)
Vehicle license plate	10
Transaction date	4
Login status	1
Gate code	2
Employee ID number	18
Expired date	4
Card status	1
Total	**40**

The data values that have been mapped must first be converted into hexadecimal. Therefore, the data can be written onto the card. If converted into the hexadecimal form, the first data, AB2039YQ vehicle license plate, will become 41 42 32 30 33 39 59 51. Because the vehicle license plate only consists of eight digits, while the implementation of mapping on the card requires 10 digits for the vehicle number, it is necessary to do zero-padding, which is to add two digits to the left of the data (00AB2039YQ) so that the data in hexadecimal form becomes 30 30 41 42 32 30 33 39 59 51. Overall, the mapping data is shown in Fig. 4. Total data is 40 bytes, requiring three data blocks with the remaining eight bytes of unused data.



Block 0

30	30	41	42	32	30	33	39	59	51	60	53	16	0B	01	00

Block 1

FF	31	39	38	39	31	31	30	38	32	30	31	39	30	33	31

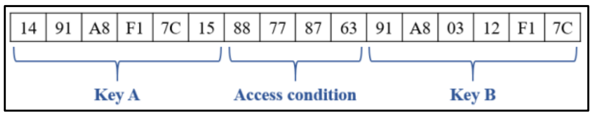
Block 2

30	32	30	60	53	16	0B	01	00	00	00	00	00	00	00	00

Fig.4. Data after being mapped

B. *Key A, Key B, and Access Condition*

In this study, the key A and B values are dynamic according to the smart card UID. The UID is a unique smart card identifier that is owned by each card. In addition to utilizing the UID smart card, the keys will be reversed and added with two bytes of static data to increase security. Furthermore, in the access condition, an access rule is created on the block to read data using key A or key B. While writing, data is set only to use key B. It aims to avoid unauthorized third-party access. For example, to modify data on the card. Fig. 5 shows the format of key A, key B, and access conditions in a block trailer.



$$\underbrace{14\ 91\ A8\ F1\ 7C\ 15}_{\text{Key A}}\ \underbrace{88\ 77\ 87\ 63}_{\text{Access condition}}\ \underbrace{91\ A8\ 03\ 12\ F1\ 7C}_{\text{Key B}}$$

Fig.5. Key A, key B, and access condition

In Fig. 5, the value in key A is 14 91 A8 F1 7C 15 with a length of six bytes, consisting of four UID smart cards and two bytes of data that can be defined as desired. In this case, the current year, 2021, is used. The UID of the smart card needs to be reversed, and the two bytes of static data also need to be converted to hexadecimal form. Fig. 6 illustrates obtaining the value of key A based on the UID and the current year as static data.

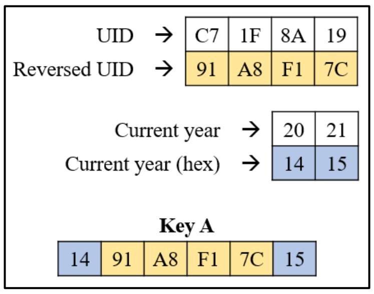


Fig.6. Illustration of forming key A

Furthermore, the value of key B is also obtained in the same way as key A. However, what distinguishes between key A and key B is the pattern used. Key B also uses four bytes of the smart card UID and two bytes of static data. Therefore, to form the key B, the four bytes of the UID need to be reversed and then divided into two parts, then two bytes are placed on the left and two bytes on the right, while the remaining bytes for the third and fourth bytes are filled with other static data. In this case, the current month and date are used, 03 18. Fig. 7 shows an illustration to obtain the value of key B based on the UID and the current month and date as static data.

UID →	C7	1F	8A	19
Reversed UID →	91	A8	F1	7C

Current month & date →	03	18
Current month & date (hex) →	03	12

Key B

91	A8	03	12	F1	7C

Fig.7. Illustration of forming key B

In this study, the rule for reading and writing access used is reading data using key A or key B and writing data using only key B. Thus, based on the access condition rules in Table 1 and Table 2. Then, the access condition values for data blocks and sectors trailer are shown in Fig. 8.

110	→	block 1
110	→	block 2
110	→	block 3
001	→	sector trailer

Fig.8. Access condition

In Fig. 8, access bit 110 in the data block states that the keys A|B are used for reading operations, while key B is used for write operations. Furthermore, 001 in the trailer sector means that the key A|B is used to write operations on byte key A and byte key B. Each access bit in Fig. 8 must be converted into a one-byte format as shown in Fig. 9 to Fig. 11.

Based on the invert results in Fig. 9 to Fig. 11, the access condition value for data block 0 to data block 2 is 88 77 87. Furthermore, in the fourth byte (block 3), the value used is as desired. In this case, the value used is 63 (hex), so the access condition value is 88 77 87 63, as shown in Fig. 5.

C. *Writing Data to the Smart Card*

After the data and key authentication format have been determined, all the data is converted into hexadecimal to be written on a card. It is because 40 bytes of data only takes up one sector. The writing process can be done in any sector between 0 and 15 because the Mifare 1K card has a capacity of 1024 bytes or 1 kilobyte. However, it would be better if the data was written in other than sector 0 because sector 0 is generally used to store the UID and manufacturer number. Fig. 12 is the result of writing the data in Table 3 onto the smart card.

Data block 0	
1	Negation of 1st bit in the sector trailer
0	Negation of 1st bit in the data block 2
0	Negation of 1st bit in the data block 1
0	Negation of 1st bit in the data block 0
1	Negation of 0th bit in the sector trailer
0	Negation of 0th bit in the data block 2
0	Negation of 0th bit in the data block 1
0	Negation of 0th bit in the data block 0
10001000 → 88 (hex)	

Fig.9.Access bit for data block 0

Data block 1	
0	The 0th bit of the sector trailer
1	The 0th bit of the data block 2
1	The 0th bit of the data block 1
1	The 0th bit of the data block 0
0	Negation of 2nd bit in the sector trailer
1	Negation of 2nd bit in the data block 2
1	Negation of 2nd bit in the data block 1
1	Negation of 2nd bit in the data block 0
01110111 → 77 (hex)	

Fig.10.Access bit for data block 1

Data block 2	
1	The 2nd bit of the sector trailer
0	The 2nd bit of the data block 2
0	The 2nd bit of the data block 1
0	The 2nd bit of the data block 0
0	The 1st bit of the sector trailer
1	The 1st bit of the data block 2
1	The 1st bit of the data block 1
1	The 1st bit of the data block 0
10000111 → 87 (hex)	

Fig.11. Access bit for data block 2

Block 0															
30	30	41	42	32	30	33	39	59	51	60	53	16	0B	01	00
Block 1															
FF	31	39	38	39	31	31	30	38	32	30	31	39	30	33	31
Block 2															
30	32	30	60	53	16	0B	01	00	00	00	00	00	00	00	00
Block 3 (trailer)															
14	91	A8	F1	7C	15	88	77	87	63	91	A8	03	12	F1	7C

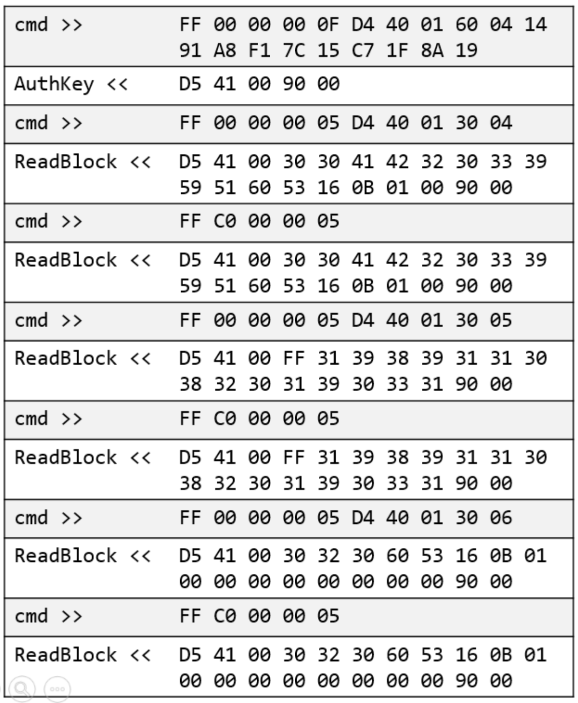
Fig.12. Results of writing data to the smart card

IV. DISCUSSION

Several test scenarios were carried out related to the use of key A or key B to perform data reading and writing operations on the smart card. It is according to predetermined access conditions (88 77 87 63), and the arranged key A|B. Tests were carried out using an ACR122U reader and a card with UID = C7 1F 8A 19. The communication process from the reader to the smart card begins with the command to get the UID, authentication according to the key used, and continues with the operation to be carried out, reading or writing data.

A. Reading Data Using the Key A

The first test scenario is to read data using key A. The test results are shown in Fig. 13.



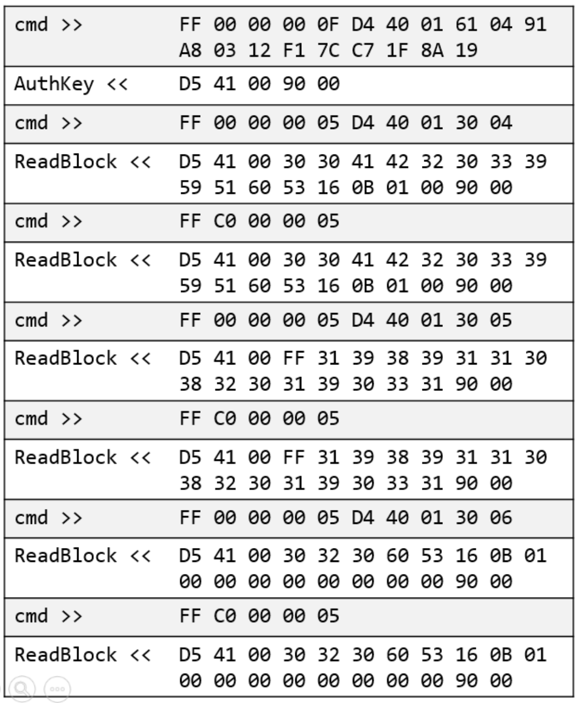
```
cmd >>          FF 00 00 00 0F D4 40 01 60 04 14
                91 A8 F1 7C 15 C7 1F 8A 19
AuthKey <<      D5 41 00 90 00
cmd >>          FF 00 00 00 05 D4 40 01 30 04
ReadBlock <<    D5 41 00 30 30 41 42 32 30 33 39
                59 51 60 53 16 0B 01 00 90 00
cmd >>          FF C0 00 00 05
ReadBlock <<    D5 41 00 30 30 41 42 32 30 33 39
                59 51 60 53 16 0B 01 00 90 00
cmd >>          FF 00 00 00 05 D4 40 01 30 05
ReadBlock <<    D5 41 00 FF 31 39 38 39 31 31 30
                38 32 30 31 39 30 33 31 90 00
cmd >>          FF C0 00 00 05
ReadBlock <<    D5 41 00 FF 31 39 38 39 31 31 30
                38 32 30 31 39 30 33 31 90 00
cmd >>          FF 00 00 00 05 D4 40 01 30 06
ReadBlock <<    D5 41 00 30 32 30 60 53 16 0B 01
                00 00 00 00 00 00 00 00 90 00
cmd >>          FF C0 00 00 05
ReadBlock <<    D5 41 00 30 32 30 60 53 16 0B 01
                00 00 00 00 00 00 00 00 90 00
```

Fig.13. Test results of reading data using the key A

Fig. 13 shows that reading data using key A is expected because all data blocks have been read successfully. It can be seen when the reader gives a command to the smart card, the response obtained is in the form of a code in the last two bytes (90 00), and the error code on the third byte is 00, which means there is no error.

B. Reading Data Using the Key B

The second test scenario reads data using key B. The test results are shown in Fig. 14.



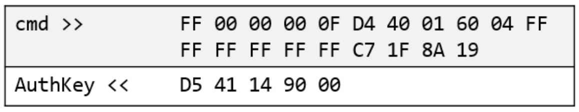
```
cmd >>          FF 00 00 00 0F D4 40 01 61 04 91
                A8 03 12 F1 7C C7 1F 8A 19
AuthKey <<      D5 41 00 90 00
cmd >>          FF 00 00 00 05 D4 40 01 30 04
ReadBlock <<    D5 41 00 30 30 41 42 32 30 33 39
                59 51 60 53 16 0B 01 00 90 00
cmd >>          FF C0 00 00 05
ReadBlock <<    D5 41 00 30 30 41 42 32 30 33 39
                59 51 60 53 16 0B 01 00 90 00
cmd >>          FF 00 00 00 05 D4 40 01 30 05
ReadBlock <<    D5 41 00 FF 31 39 38 39 31 31 30
                38 32 30 31 39 30 33 31 90 00
cmd >>          FF C0 00 00 05
ReadBlock <<    D5 41 00 FF 31 39 38 39 31 31 30
                38 32 30 31 39 30 33 31 90 00
cmd >>          FF 00 00 00 05 D4 40 01 30 06
ReadBlock <<    D5 41 00 30 32 30 60 53 16 0B 01
                00 00 00 00 00 00 00 00 90 00
cmd >>          FF C0 00 00 05
ReadBlock <<    D5 41 00 30 32 30 60 53 16 0B 01
                00 00 00 00 00 00 00 00 90 00
```

Fig.14. Test results of reading data using the key B

Fig. 14 shows that reading data using key B is expected because all data blocks have been read successfully. It can be seen when the reader gives a command to the smart card, the response obtained is in the form of a code in the last two bytes (90 00), and the error code on the third byte is 00, which means there is no error.

C. Reading Data Using the Default Key

The third test scenario is to read data using the default key. The test results are shown in Fig. 15.



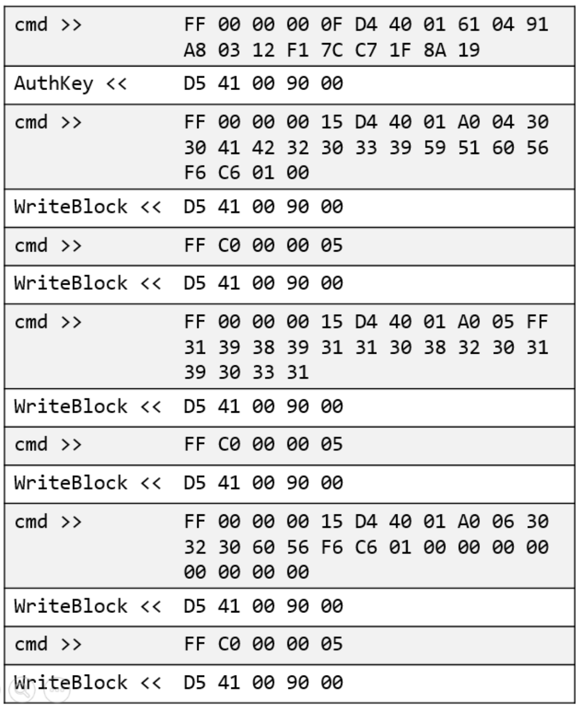
```
cmd >>          FF 00 00 00 0F D4 40 01 60 04 FF
                FF FF FF FF FF C7 1F 8A 19
AuthKey <<      D5 41 14 90 00
```

Fig.15. Test results of reading data using the default key

Fig. 15 shows that reading data using the default key cannot be done because key A and key B have been changed according to the mechanism proposed in this study. In this test scenario, the third byte generates an error code of 14 which means an error occurred. It shows that the test results are as expected.

D. Writing Data Using the Key B

The fourth test scenario is to write data using key B. The test results are shown in Fig. 16.



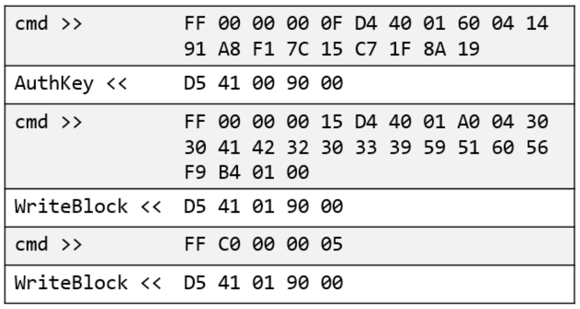
```
cmd >>          FF 00 00 00 0F D4 40 01 61 04 91
                A8 03 12 F1 7C C7 1F 8A 19
AuthKey <<      D5 41 00 90 00
cmd >>          FF 00 00 00 15 D4 40 01 A0 04 30
                30 41 42 32 30 33 39 59 51 60 56
                F6 C6 01 00
WriteBlock <<   D5 41 00 90 00
cmd >>          FF C0 00 00 05
WriteBlock <<   D5 41 00 90 00
cmd >>          FF 00 00 00 15 D4 40 01 A0 05 FF
                31 39 38 39 31 31 30 38 32 30 31
                39 30 33 31
WriteBlock <<   D5 41 00 90 00
cmd >>          FF C0 00 00 05
WriteBlock <<   D5 41 00 90 00
cmd >>          FF 00 00 00 15 D4 40 01 A0 06 30
                32 30 60 56 F6 C6 01 00 00 00 00
                00 00 00 00
WriteBlock <<   D5 41 00 90 00
cmd >>          FF C0 00 00 05
WriteBlock <<   D5 41 00 90 00
```

Fig.16. Test results of writing data using the key B

Fig. 16 shows that writing data using key B was successfully performed. An error code indicates this in the third byte of 00, which means there is no error. It shows that the test results are as expected.

E. Writing Data Using the Key A

The last test scenario is to write data using key A. The test results are shown in Fig. 17.



```
cmd >>          FF 00 00 00 0F D4 40 01 60 04 14
                91 A8 F1 7C 15 C7 1F 8A 19
AuthKey <<      D5 41 00 90 00
cmd >>          FF 00 00 00 15 D4 40 01 A0 04 30
                30 41 42 32 30 33 39 59 51 60 56
                F9 B4 01 00
WriteBlock <<   D5 41 01 90 00
cmd >>          FF C0 00 00 05
WriteBlock <<   D5 41 01 90 00
```

Fig.17. Test results of writing data using the key A

Fig. 17 shows that writing data using key A cannot be done even though during the authentication process, there was no error. However, the error code response on the third byte has a value of 01 which means an error occurred. It shows that the test results are as expected.

The mechanism for securing data using dynamic keys and access conditions is still possible for others to know, for example, by performing authentication with all possible combinations of the six-byte key and trying to analyze the pattern of keys that have been successfully identified. Therefore, in further research, it is necessary to develop data security techniques using encryption algorithms such as DES [14] and 3DES [15] or hardware such as the Secure Access Module (SAM) [12] to increase the security of data stored on smart cards.

V. CONCLUSION

Based on the test results of data security techniques on the smart card using the A | B key and access conditions, it can be concluded that this data security technique can secure data stored on the smart card by providing limitations on the operation of writing and reading data in sectors or data blocks. The dynamic key generation depends on the UID of each smart card, so a key that can be used to access a smart card cannot be used to access another smart card as long as the pattern of the key is unknown. This condition is, of course, very important to avoid duplicating data on the smart card, modifying data already stored on the smart card, or rewriting data on the smart card.

REFERENCES

[1] P. K. Singh, N. Kumar, and B. K. Gupta, "Smart Card ID: An Evolving and Viable Technology," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 3, pp. 115–124, 2018.

[2] H. Taherdoost, S. Sahibuddin, and N. Jalaliyoon, "Smart Card Security; Technology and Adoption," *Int. J. Secur.*, vol. 5, no. 2, pp. 74–84, 2011.

[3] D. S. K. Putra, M. A. Sadikin, and S. Windarta, "S-Mbank: Secure mobile banking authentication scheme using signcryption, pair based text authentication, and contactless smart card," in *2017 15th international conference on quality in research (QiR): international symposium on electrical and computer engineering*, 2017, pp. 230–234.

[4] A. Y. Ananta *et al.*, "Smart monitoring system for teaching and learning process at the university," in *IOP Conference Series: Materials Science and Engineering*, 2020, vol. 732, no. 1.

[5] A. K. Das, A. K. Sutrala, V. Odelu, and A. Goswami, "A Secure Smartcard-Based Anonymous User Authentication Scheme for Healthcare Applications Using Wireless Medical Sensor Networks," *Wirel. Pers. Commun.*, vol. 94, no. 3, pp. 1899–1933, 2017.

[6] W. Yang *et al.*, "Securing mobile healthcare data: A smart card based cancelable Finger-Vein Bio-Cryptosystem," *IEEE Access*, vol. 6, pp. 36939–36947, 2018.

[7] M. Arnone, T. Delmastro, G. Giacosa, M. Paoletti, and P. Villata, "The Potential of E-ticketing for Public Transport Planning: The Piedmont Region Case Study," *Transp. Res. Procedia*, vol. 18, pp. 3–10, 2016.

[8] I. M. Insan, P. Sukarno, and R. Yasirandi, "Multi-Factor Authentication Using a Smart Card and Fingerprint (Case Study: Parking Gate)," *Indones. J. Comput.*, vol. 4, no. 2, pp. 55–66, 2019.

[9] K. Markantonakis, M. Tunstall, G. Hancke, I. Askoxylakis, and K. Mayes, "Attacking smart card systems: Theory and practice," *Inf. Secur. Tech. Rep.*, vol. 14, no. 2, pp. 46–56, 2009.

[10] A. Kundarap, A. Chhajlani, R. Singla, M. Sawant, M. Dere, and P. Mahalle, "Security for Contactless Smart Cards Using Cryptography," in *International Conference on Network Security and Applications*, 2010, pp. 558–566.

[11] R. N. Akram, K. Markantonakis, and K. Mayes,

"Pseudorandom number generation in smart cards: An implementation, performance and randomness analysis," in *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, 2012.

[12] L. Malina, V. Benes, J. Hajny, and P. Dzurenda, "Efficient and secure access control system based on programmable smart cards," in *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, 2017, pp. 32–36.

[13] NXP, "MIFARE Classic EV1 1K - Mainstream contactless smart card IC for fast and easy solution development." pp. 1–36, 2018.

[14] A. M. Sison, B. T. Tanguilig, B. D. Gerardo, and Y. C. Byun, "Implementation of Improved DES Algorithm in Securing Smart Card Data," in *Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity*, Berlin, Heidelberg: Springer, 2012, pp. 252–263.

[15] Ratnadewi, R. P. Adhie, Y. Hutama, A. Saleh Ahmar, and M. I. Setiawan, "Implementation Cryptography Data Encryption Standard (DES) and Triple Data Encryption Standard (3DES) Method in Communication System Based Near Field Communication (NFC)," in *Journal of Physics: Conference Series*, 2018, vol. 954, no. 1.