ScreenCheck

BADGEMAKER 7

SC Mifare Keyfile Generator

· ID DESIGN · MANAGE · PRINT SYSTEM ·

www.screencheck.com

SCREEN CHECK

Contents

Contents 2
Introduction 3
Mifare Keys Explained 4
Installation 6
Using SC Mifare KeyFile builder 8
Change Password 8
Choose a Mifare type 9
Initialize Keys 9
Read Keyfile 12
Write KeyFile 13
Manually fill in the key file 13
Location of the keyfile 16

Introduction

The Mifare Keyfile generator is designed to produce mifare KeyFiles. These KeyFiles can be used by the **SCMifare.exe** and **ScMifareEnc.dll**, in version 4.2 or higher.

Earlier versions of ScreenCheck mifare applications worked with keys stored on a keycard, which had to be produced by ScreenCheck. To accommodate customer's request which was the need to create custom key's themselves, the Mifare Keyfile Generator application was created.

Due to keys being such a high security issue, the stored KeyFiles are encrypted. KeyFiles once stored on a card can only be read by the Mifare Keyfile generator program. A user must provide a password to gain access to the data. When a Keyfile is read this password must be supplied, to be able to decrypt the keys.

There are two types of mifare cards supported.

- **Mifare Classic 1K**
- **Mifare Classic 2K**

Mifare Keys Explained

A Mifare card is a storage block containing sectors, these sectors contain blocks.

The two supported types have the following layout.

Classic 1k:

- 16 sectors (*or tracks*), each sector contains 4 blocks,
- 3 user blocks (*block 0 to 2*) and one key block (*block 3*)
- Sector 0 block 0 cannot be used (*contains manufactory data*)
- Sector 0 block 1 and 2 can be used for the MAD.
- Each block contains 16 bytes.

Classic 4k:

- 40 sectors (*or tracks*).
- The first 32 sectors contain 4 blocks,
- 3 user blocks (*block 0 to 2*) and one key block (*block 3*)
- Sector 0 block 0 cannot be used (*contains manufactory data*)
- Sector 0 block 1 and 2 can be used for the MAD for sector 1 to 15.
- In case MAD is used, sector 16 block 0 to 2 contain the MAD
- information for sector 16 to 39.
- The last 8 sectors contain 16 blocks,
- 15 user blocks (*block 0 to 14*) and one key block (*block 15*).
- Block 0 key is for block 0 to 4
- Block 1 key is for block 5 to 9
- Block 2 key is for block 10 to 14
- Block 3 key is for block 15, which is the key block.
- Each block contains 16 bytes.

So for both **Classic 1k** and **Classic 4k** there is one key block for each sector. Each key block contains 16 bytes.

This is divided into two keys (**A/B**) of 6 bytes and a 4 byte access-code. The **A** and the **B** keys use 6 bytes (*typically expressed as 12 hexadecimal characters*). These keys are basically passwords each key can be used to protect a certain function.

Example

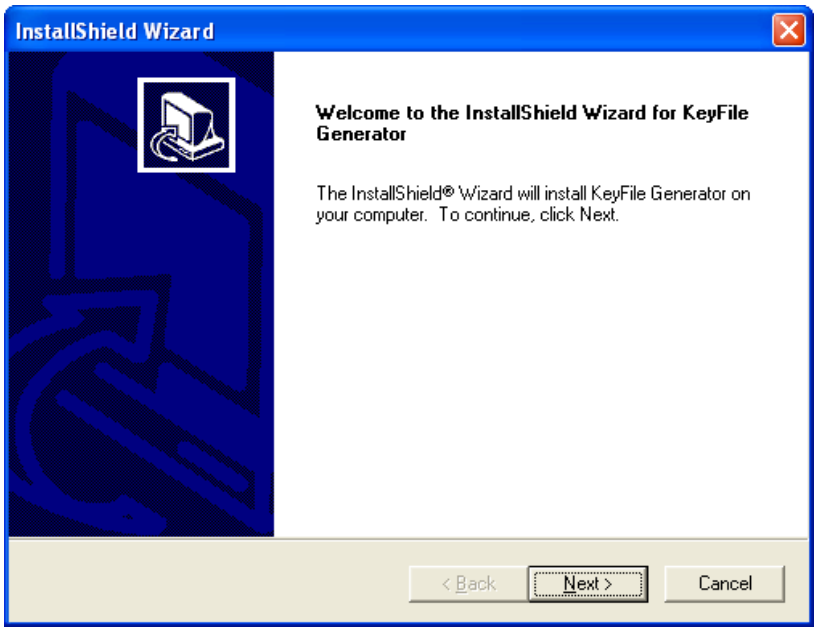
The **A** key could be required to read data in a sector, while the **B** key could be required to write data to a sector. Visa versa the **A** key could be required to deduct stored value from a sector, while the **B** key could be required to add stored value. To access data in a protected card sector, the reader must have a matching key configured.

Keys are used to protect data from being read or changed without authorization. Because each sector has its own separate key pair, a Mifare card can be used to store information encoded on the cards by separate vendors for separate applications, each vendor is prevented from modifying the other vendor's data accidentally or otherwise, simply by keeping keys secret. For this to work, the keys to the card's Mifare Applications Directory need to be known to all parties. Separate sets of

readers would be used to control each application and each reader would have only the appropriate keys available for its own application.

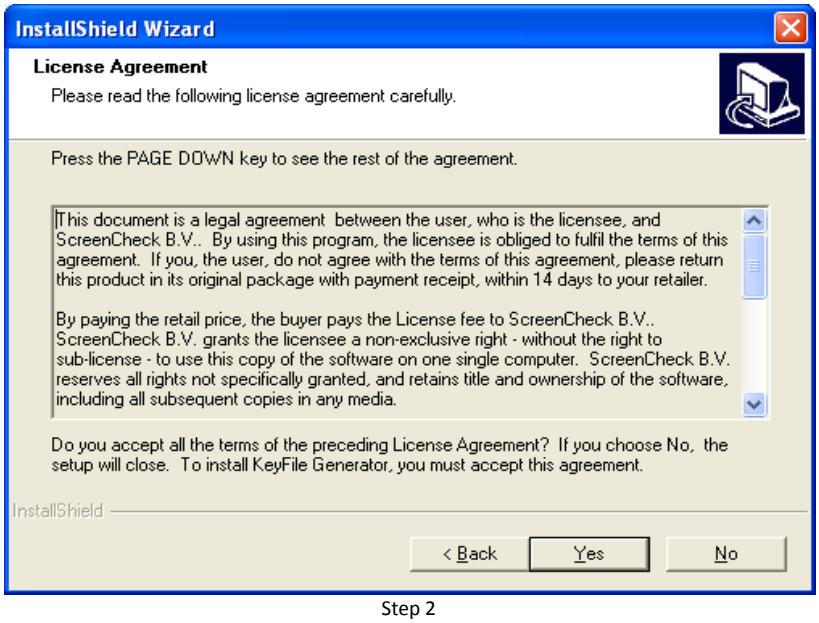
Installation

Select **SCKeyFileMifare.exe** to begin the installation. The following dialog is presented.



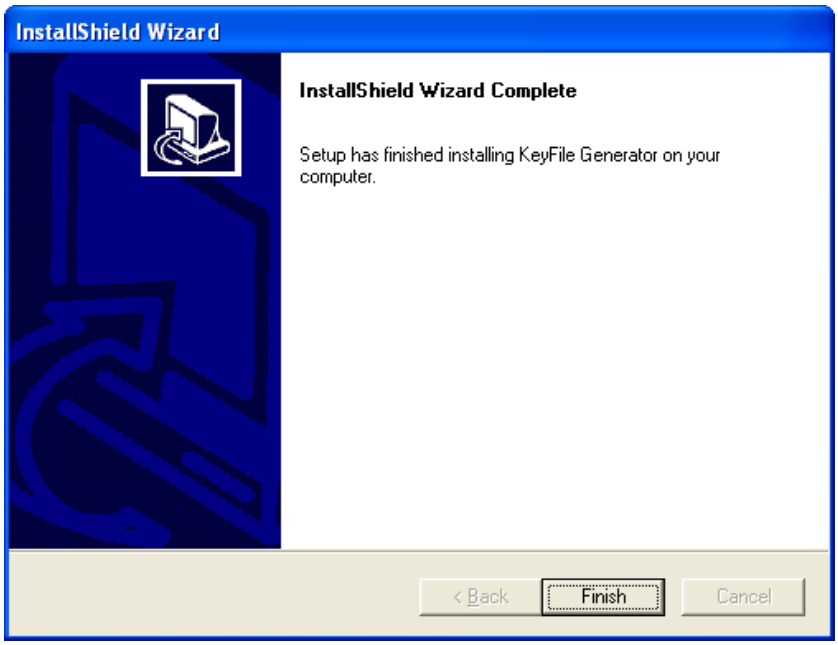
Step 1

Click **Next** to proceed. Please read the **License Agreement** and click **Yes**.



Step 2

Enter the required information – **User Name** and **Company Name**. Click **Next**.



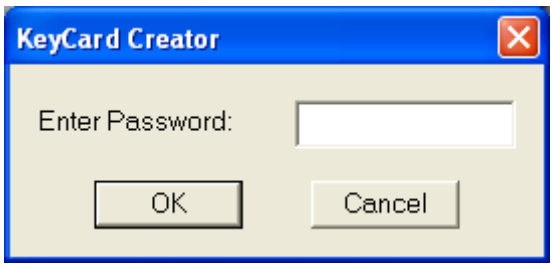
Step 3

Setup has completed installing the software; click **Finish** to end the installation.

Using SC Mifare KeyFile builder

To start the application click the **Start menu>All Programs>SCKeyFileGenerator> SCKeyFileGenerator.**

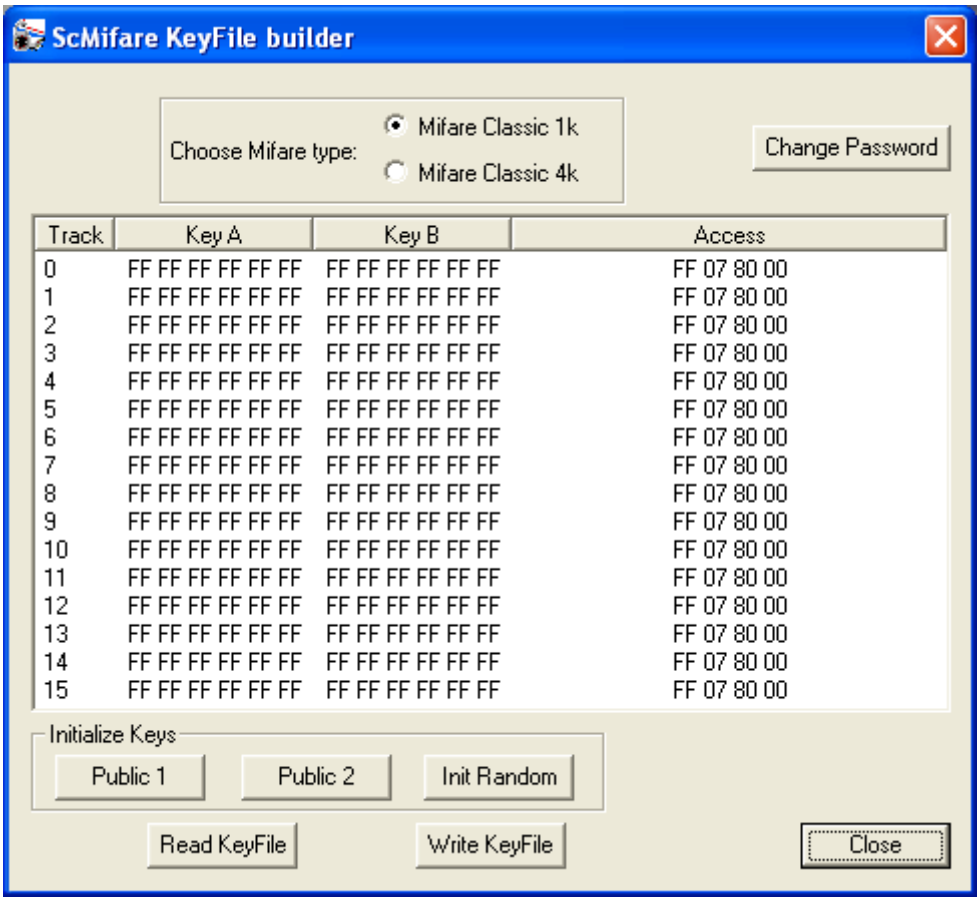
The operator must first enter a password to enter the software. The following log-in dialog is presented.



KeyCard Creator login

The default Password is: **1A1B1C1D1E1F**

If the login was successful, the following dialogue screen is presented.

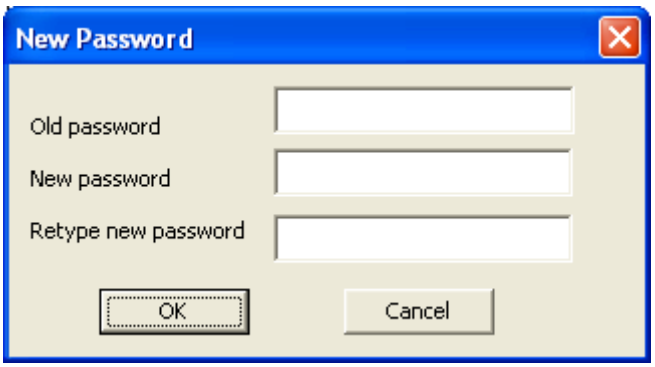


SCMifare KeyFile builder

Change Password

To change the default login password click **Change Password** button.

The following dialogue screen is displayed:

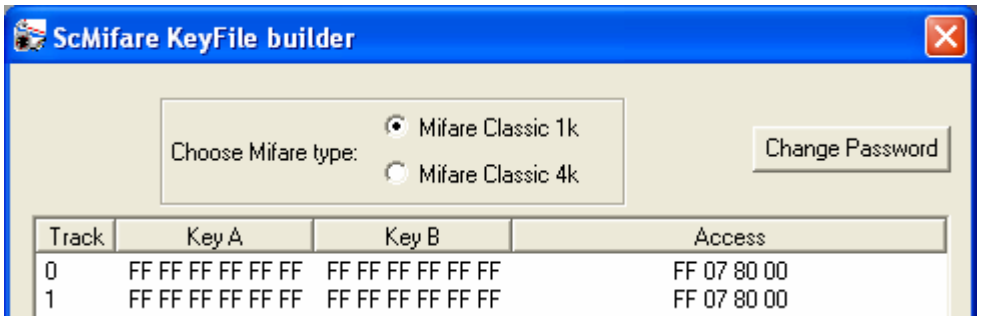


New Password dialog

You must enter first the old password and then enter a new password; you must confirm your new password before selecting **OK**.

Choose a Mifare type

On the top side of the dialogue screen you can select **Mifare Classic 1k** or **Mifare Classic 4k** (*Mifare Classic 1k is default*).



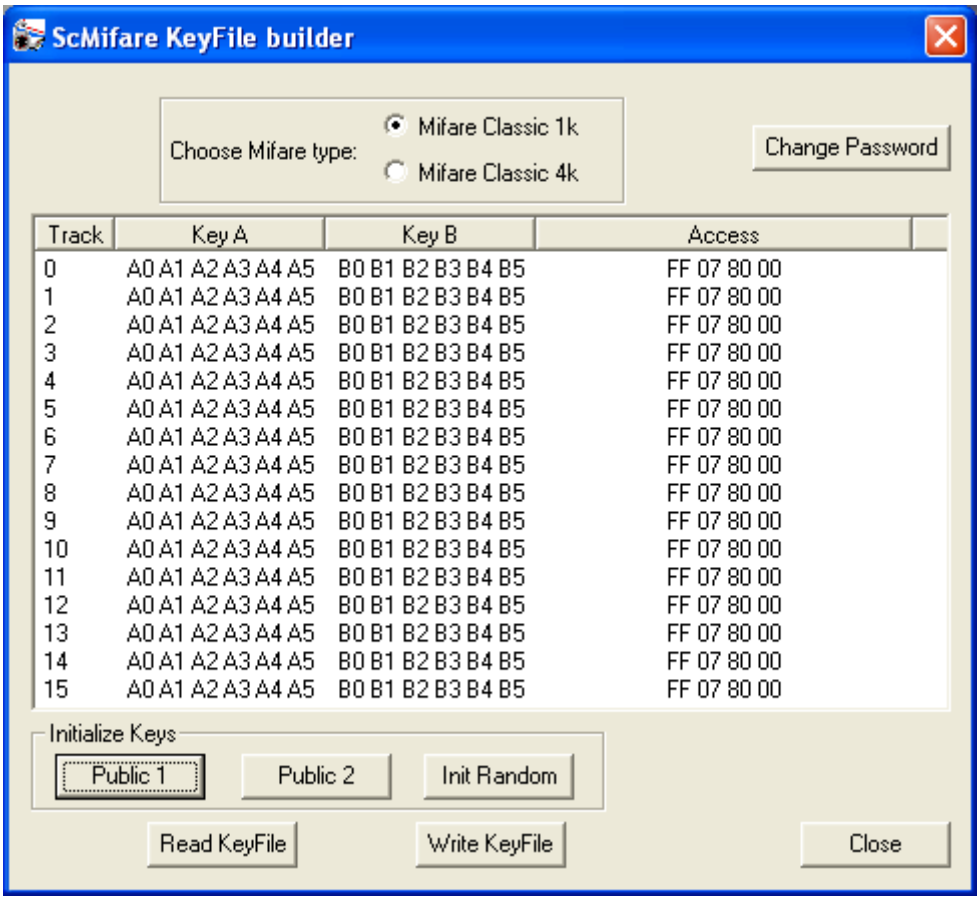
Also the default key displayed is **Public key 2** (**FFFFFFFFFFFF**).
The keys for a **4K Mifare** card are basically the same as for a **1K Mifare** card. There are more sectors, 40 instead of 16. But the layout of the keys is the same for all the sectors.

Be aware that sectors **32** to **39** contain 16 blocks instead of 4 blocks. So the access code divides the sector in three groups of 5 blocks plus one block to store the keys.

The examples in the next paragraphs will handle **Mifare 1K**.

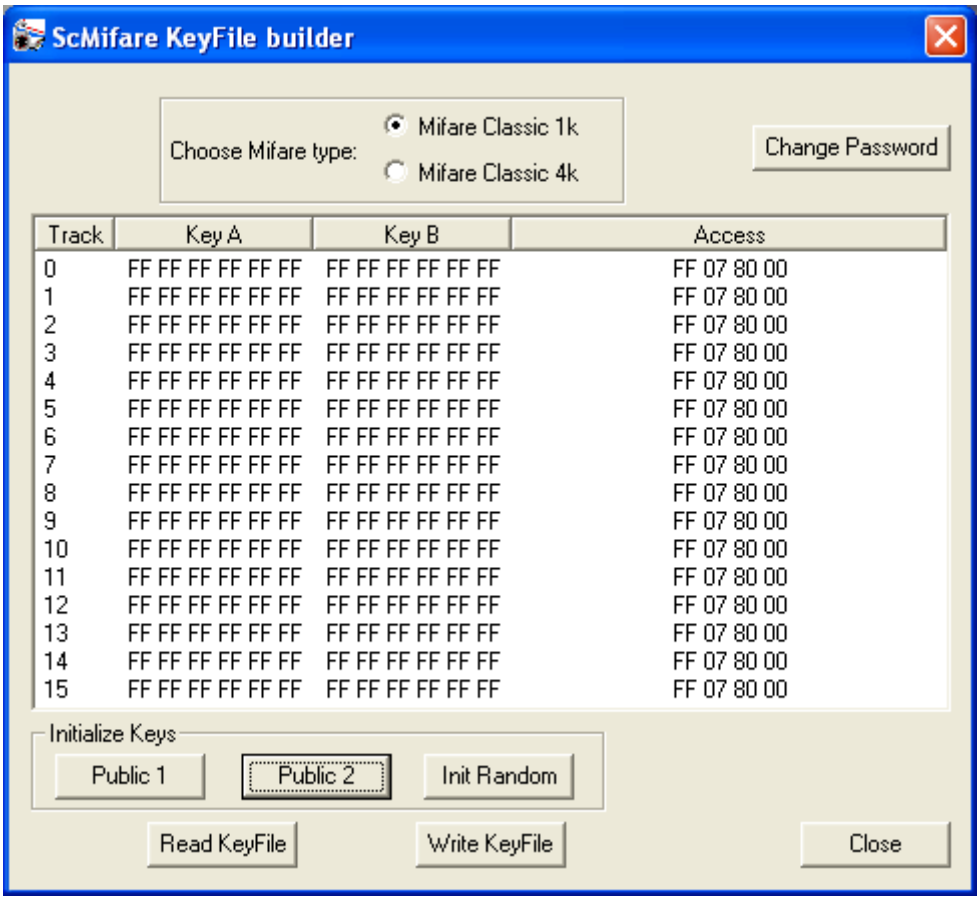
Initialize Keys

Public 1: This will create a **Public Key 1** key file. As shown below..



Public Key 1

Public 2: This will create a **Public Key 2** key file. As shown below.



Public Key 2

Init Random: This will create a random key file. You can specify an Access-Code for each block.

By default this will be configured as follows

Read:

- A/B

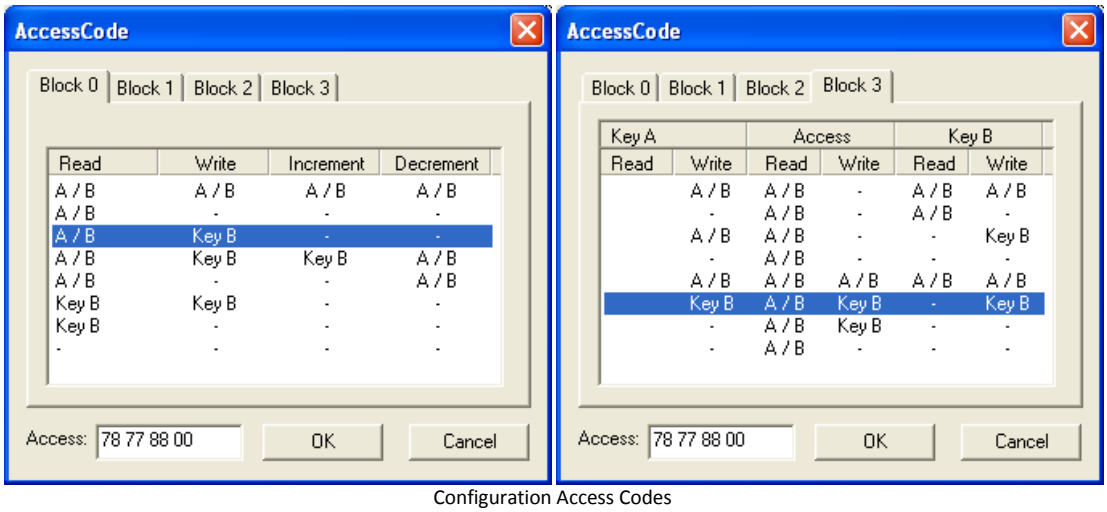
Write:

- B for block 0 till 2.

For block 3:

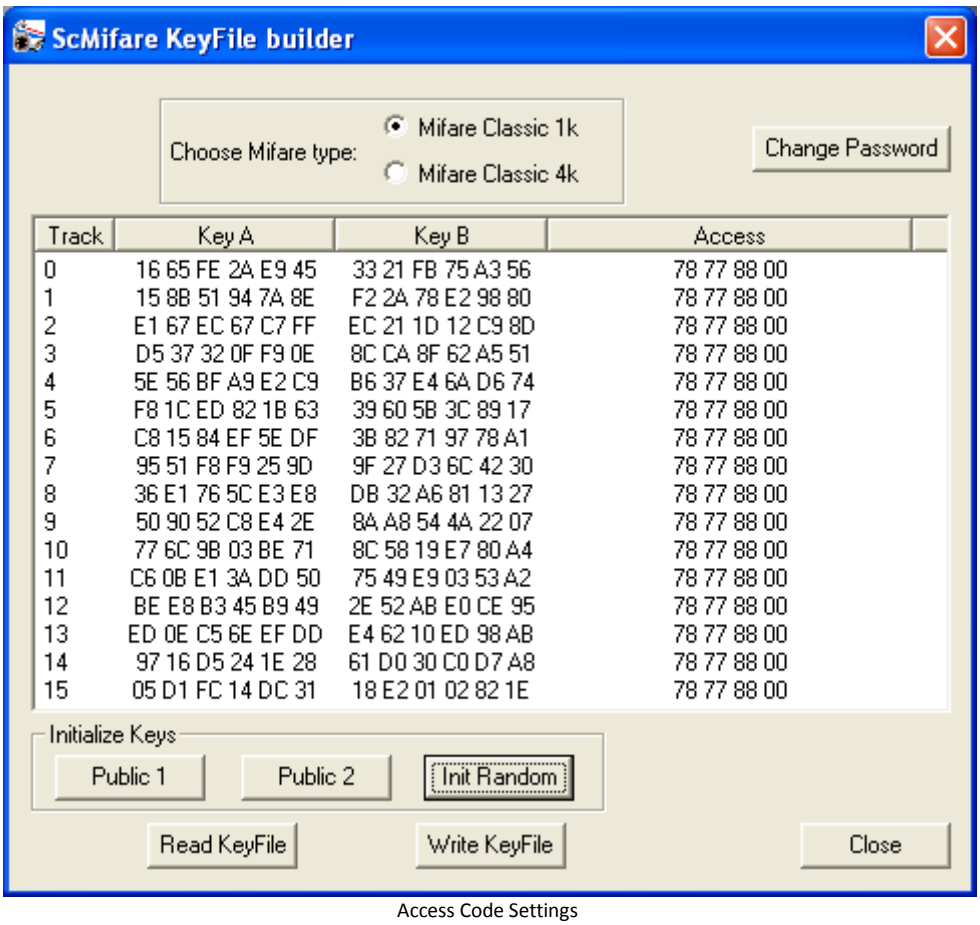
- Key A write with KeyB.
- Access code read with A/B, write with KeyB.
- KeyB write with KeyB.

The results are displayed in the following dialog.



Configuration Access Codes

The final result after accepting the default Access-Code settings, will look like as shown below.



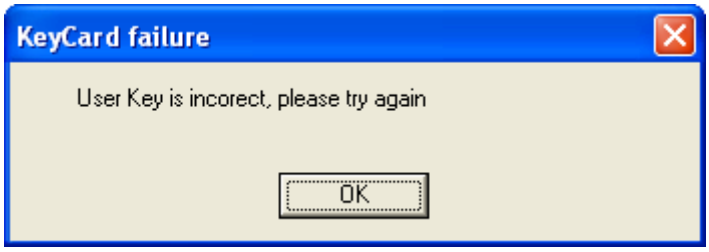
Access Code Settings

Read Keyfile

With this button, you can read the keys currently stored in the KeyFile. To demonstrate, first hit the Public 2 button, so the newly generated keys will be replaced by the public keys.

Click **Read KeyFile**, you are now requested to input the matching encryption key.

When an incorrect key is given, for example 12345678, you will receive the following error message:



Click **OK** and provide the correct key: “AG_7*1D1”.

The correct KeyFile will now be displayed.

Write KeyFile

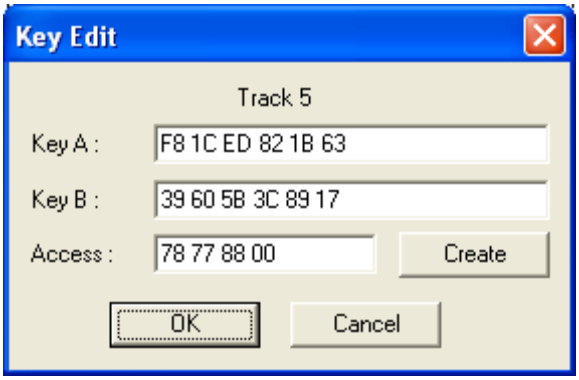
With this button**,** the new keys can be stored in a KeyFile. When this button is clicked, you are requested to enter an **8** character password.

➧ **PLEASE REMEMBER THIS PASSWORD. YOU WILL NEED THIS PASSWORD TO READ STORED KEYS.**

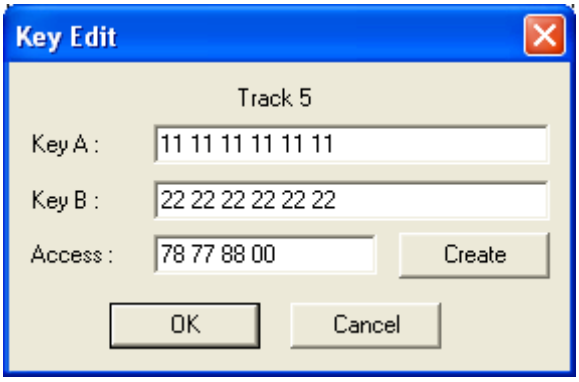
Encryption Key

Manually fill in the key file

In most cases you have specific keys for certain sectors, so you don’t want to use public keys or random keys. In this case double click the sector (*track*) you want to fill in, for example **Track 5**.



Now fill in the specific **A** and **B** key for this track as shown below. And set the Access code for this track (*sector*).

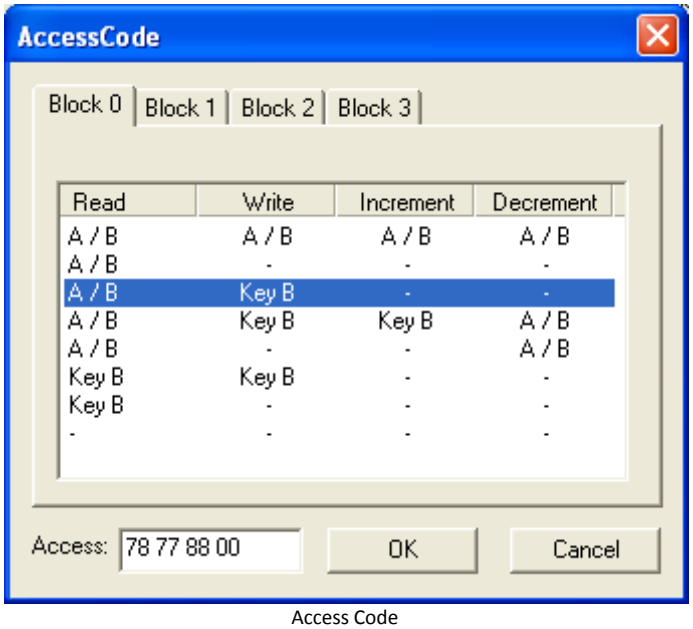


Key Edit

You can do this directly by typing the 4 bytes, but this means you have to know the meaning of these bytes as described in the Mifare specification. This is not easy, so to help you create the required Access-Code, you can select the **Create** button.

Now a tool to help you to create the Access-Code is shown. Here you can create the access conditions for the 4 blocks. Which the program will automatically convert to the correct 4 byte Access-Code.

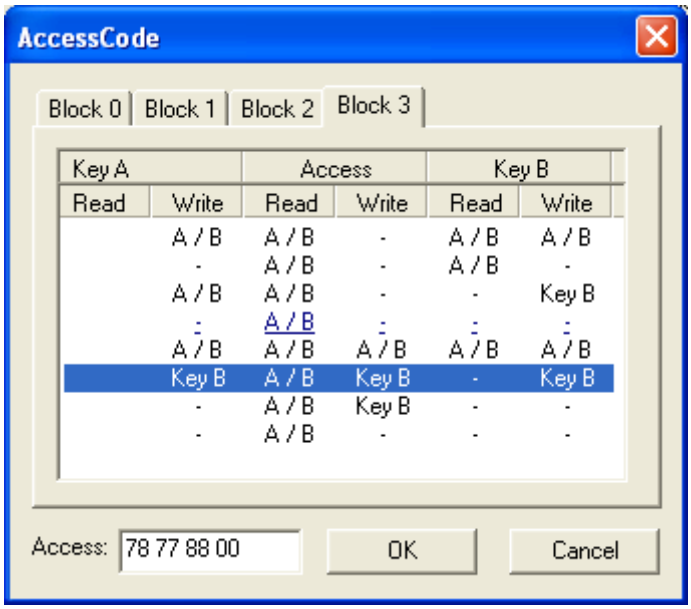
The options for the first three blocks are identical as shown below.



Access Code

Read	Write	Increment	Decrement	Explanation
A / B	A / B	A / B	A / B	Key A and B can be used for Read, Write, Increment and Decrement data
A / B	-	-	-	Key A and B can be used to read data, Write, Increment and decrement data is not possible
A / B	KeyB	-	-	Key A and B can be used to read, KeyB can be used to Write. Increment and Decrement are not possible
A / B	KeyB	KeyB	A / B	Key A and B can be used to read and Decrement KeyB can be used to Write and Increment
A / B	-	-	A / B	Key A and B can be used to read and Decrement Write and Increment are not possible
KeyB	KeyB	-	-	KeyB can be used to Read and Write Increment and Decrement are not possible
KeyB	-	-	-	KeyB can be used to Read Write, Increment and Decrement are not possible
-	-	-	-	Read, Write, Increment and Decrement are not possible

Block 3 which is the Key-Block is different from other blocks as shown below.



Access Code

KeyA		Access		KeyB		Explanation
read	write	read	write	read	write	
-	A/B	A/B	-	A/B	A/B	KeyA: Read not possible; Write with KeyA or B KeyB: Read and Write with KeyA or B Access: Read with KeyA or B; Write not possible
-	-	A/B	-	A/B	-	KeyA: Read and Write not possible KeyB: Read with KeyA or B; Write not possible Access: Read with KeyA or B; Write not possible
-	KeyB	A/B	-	-	KeyB	KeyA: Read not possiblel Write with KeyB KeyB: Read not possiblel Write with KeyB Access: Read with KeyA or B; Write not possible

-	-	A/B	-	-	-	KeyA: Read and Write not possible KeyB: Read and Write not possible Access: Read with KeyA or B; Write not possible
-	A/B	A/B	A/B	A/B	A/B	KeyA: Read not possible; Write with KeyA or B KeyB: Read and Write with KeyA or B Access: Read and Write with KeyA or B
-	KeyB	A/B	KeyB	-	KeyB	KeyA: Read not possiblel Write with KeyB KeyB: Read not possiblel Write with KeyB Access: Read with KeyA of B; Write with KeyB
-	-	A/B	KeyB	-	-	KeyA: Read and Write not possible KeyB: Read and Write not possible Access: Read with KeyA of B; Write with KeyB
-	-	A/B		-	-	KeyA: Read and Write not possible KeyB: Read and Write not possible Access: Read with KeyA or B; Write not possible

This way you can create keys and access codes for all sectors and store these in a new Mifare KeyFile.

Location of the keyfile

The KeyFile is stored in the mifare directory and is called **mifare.yek**. This file has to be there for the standalone application and/or the encoding *.dll to be able to access this file and work with the keys.