Physical and cryptoanalytical attacks on digital locking systems

Ivo Kubjas

1 Introduction

In many installations, traditional locks have been replaced with new type of locks, which do not rely on mechanical security for guarding the doors.

Mechanical locks hold several weaknesses, which are due to bad mechanical engineering, tolerances within cylinders, small number of different combinations etc. As the entry cost is small and learning curve for obtaining basic skills is steep, then there are many hobbyists who are into lock picking.

For example, simple lock picking sets are available starting from 5 euros, while more advanced and well made sets are available for less than 100 euros. *De facto* guide for autodidacts is MIT Guide to Lock Picking [7], which has been available for over two decades.

More recent methods as bump keying [8] have made most locks practically insecure against a well-learned adversary. The method of key bumping uses specially carved keys, which on being bumped with rubber hammer forces the pins in the lock against deformations and thus opening the lock.

Access right management for traditional locks is based on physical possession of the key. If one needs to revoke the access right, then either the key should be returned or the lock reset. Returning the key does not always imply removing the access right as it may be copied. Even if copying the keys has been made hard (eg. through licensing policies), it merely imposes organisational not physical restrictions.

All in all, these problems have lead to the creation of a new type of locks, called digital locks. Their security is based on some cryptographic protocols or in some cases - obscurity. In theory, there are several nearly-perfect protocols which have received enough scrutiny so that the locks could be impenetrable. In this review we see, that for a certain product this is not the case.

In general, when digital locks are attacked, then attacks are due to using insecure primitives. One has also to consider that embedded hardware runs in very constrained environments - the design has to consider the scarce amount of randomness and tightly bounded computing and storage capabilities.

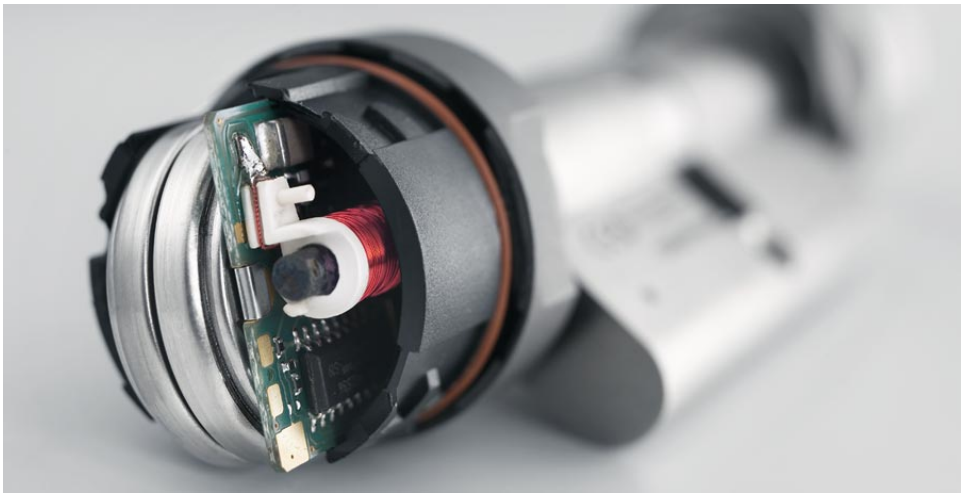


Figure 1: System 3060 cylinder[1]

This report covers combined attack methods against widely used SimonsVoss System 3060 described in [6], starting from physical aspects of the lock, analysing the cryptographic primitives and concluding with several attack vectors.

1.1 SimonsVoss System 3060

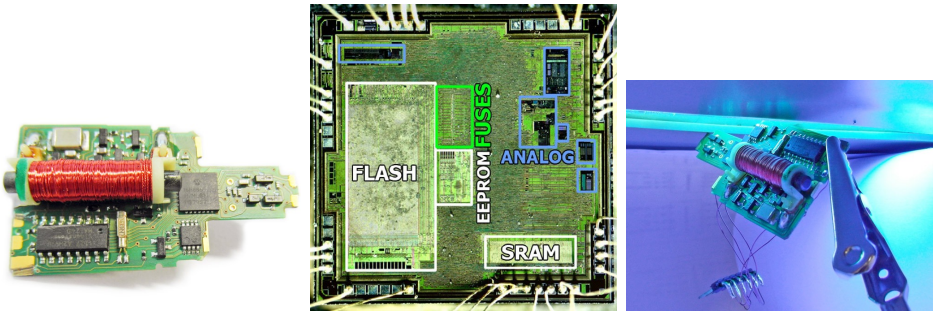
SimonsVoss Technologies GmbH has developed digital locking and access control system called System 3060. This product has been very popular throughout Europe, being installed in over a million doors and having over three million transponders. Its area of use is wide – from residential buildings to military installations.

The main parts of the system are the transponder and digital cylinder (figure 1). The cylinder is built such that it can be fitted into space meant for conventional locks and can thus be easily replaced. The cylinder has two knobs, one of which includes the electronics required for operating the mechanical part of the cylinder. The cylinder is battery operated but according to technical specification[3] should sustain 10 years or $150,000$ door openings.

The transponder is a battery operated remote control, which communicates with the cylinder upon button press. The transponder and cylinder establish mutual authentication and if access is granted, then the cylinder unlocks the mechanical part allowing the user to open the door.

The parts of the system can be integrated within an installation which can be managed by software. There can be $64,000$ cylinders and transponders within one installation. The cylinder costs about 320 euros and the

[1]Source `http://www.baulinks.de/webplugin/2012/i/1711-simonsvoss1.jpg`



(a) PCB of the cylinder (b) PIC16F886 (c) Exposing PCB to UV-C light

Figure 2: PCB of the cylinder and its details[2]

transponder costs about 30 euros.

The management software communicates with cylinders over wireless network, Ethernet or using a special programming transponder. Transponders can also be programmed to allow access to specified cylinders. If the communication is wireless, then it runs over 869 MHz and the communication between the transponder and cylinder is over 25 kHz.

Note however, that the management communication was not analysed in the article and it could also have weaknesses. In the article[6], the authors considered physical drawbacks of the cylinder, allowing reading out the keys of the system. They also considered cryptoanalytical attacks on the cipher used in the protocol.

2 Reverse engineering the cylinder

As the official specification of the system did not reveal the algorithms and methods used in System 3060, it was needed to reverse-engineer the cylinder in order to obtain necessary information.

The cylinder can be opened with a special opener tool. One of the knobs includes the electronics and it should be placed inside the area to be restricted. It was observed that in many cases electronics were located outside and thus are available for external manipulation.

Both the transponder and cylinder have a PCB containing similar elements for authentication. As the cylinder's PCB is granting access then components of its PCB were observed.

The PCB contains of three components: SimonsVoss specific Application Specific Integrated Circuit (ASIC), which is connected to Microchip PIC16F886 Microcontroller (μC). The data is stored on Electrically Erasable

[2]Source: [6]

Programmable Read-Only Memory, which is connected to the μC over a dedicated bus.

Since there was no information about the ASIC available, then it was needed to study it independently. As the ASIC is packaged into epoxy and is nearly impossible to reveal the die using mechanical methods, then White Fuming Nitric Acid can be used to reveal the die. The descriptions of the methods are available and an example of the method is described in [1].

The ASIC was photographed under an optical microscope. This revealed that the design was based on 2 μm gate array consisting of 2320 transistors. The number of transistors available implied that it was not possible to have any cryptographic algorithms implemented. Further analysis showed that the ASIC produced interrupts at timed interval for waking up the PIC and demodulated RF transmission. This implies that the actual processing is done within the PIC.

The PIC is a Microchip PIC16F886. According to its data sheet [2], its programmable memory has size of 8192 words and it also contains internal EEPROM for 256 bytes. It has also programmable and internal memory protection which is controlled by setting configuration bits. In order to read out the firmware of the PIC, the authors needed to unset the configuration bits.

Flash memories have a resemblance with a UV-EPROM memories which can be erased using UV light with a wavelength of 250 nm. In the PIC16F886 the cells which hold configuration bits are covered with metal shields (the EEPROM cells are not covered as the die is packaged in epoxy and in normal condition are not subject to any light) to thwart resetting the contents. But using UV light at an angle, then due to light bouncing between the shield and the plate (see figure 3), the configuration bits can be erased. This method was tested in [4] with PIC18F1230.

The authors of [6] decapsulated the microcontroller (μC) using White Fuming Nitric Acid and tried the UV lighting method with different angles and exposure times. The bits could be erased with a 20 minute exposure to UV light. If the internal EEPROM was covered with electrical isolation tape, then it was not affected by exposure. This lead to a state which allowed the authors to read out the programming memory and internal EEPROM.

IDA Pro was used to disassemble the binary dump to reverse-engineer the code. As the μC allows run time memory read and write, it was possible to read out the internal states of the program and then modify the execution.

3 Protocol

3.1 Key derivation

The transponder has two 128-bit keys $K_{T,\mathsf{ext}}$ and $K_{T,\mathsf{int}}$ where $K_{T,\mathsf{ext}}$ is stored on PCB's EEPROM and $K_{T,\mathsf{int}}$ is stored on μC' internal EEPROM.

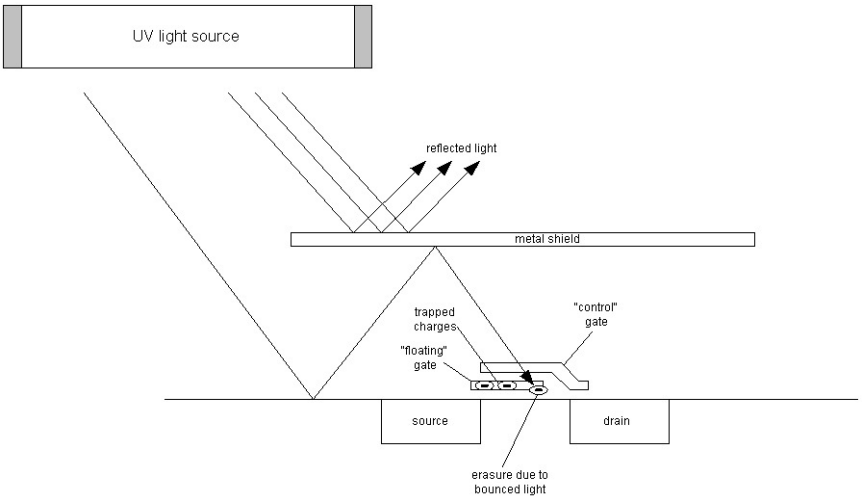


Figure 3: Clearing protected cells with UV light

The key used in authentication is acquired from $K_T = K_{T,\mathsf{ext}} \oplus K_{T,\mathsf{int}}$.

The cylinder has five 128-bit keys: four keys $K_{L,j,\mathsf{ext}}$, $0 \leq j \leq 3$ stored on PCB's EEPROM and one key $K_{L,\mathsf{int}}$ stored on μC's internal EEPROM. These keys can be used to derive four keys, denoted as system keys, using $K_{L,j} = K_{L,j,\mathsf{ext}} \oplus K_{L,\mathsf{int}}$. It was noted that the system keys are identical within one installation.

3.2 Challenge-response protocol

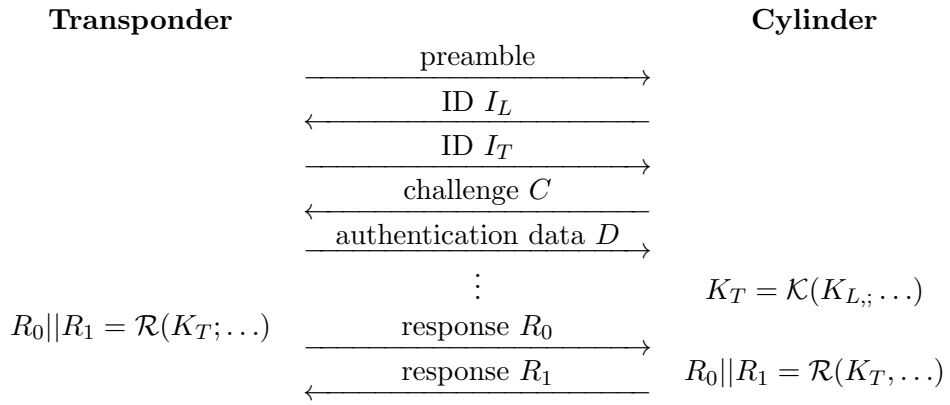


Figure 4: Challenge-response protocol between the transponder and the cylinder. Irrelevant messages omitted.

Each transponder and cylinder has identification tags I_T and I_L which have lengths 24 and 32 bit, respectively. The challenge-response protocol

consists of 11 messages, which are transmitted synchronously between the transponder and the cylinder. The protocol is started by the transponder on the button press. The challenge-response protocol is visualised on figure 4.

The intermediate message C is a random 88-bit challenge. The fixed authentication message D contains transponder specific information which is used in key derivation.

There are two cryptographic primitives available, namely the encryption function $\mathcal{D}$ and the obscurity function $\mathcal{O}$.

The keys are distributed such that the lock is able to compute transponders key K_T from its messages as

$$K_T = \mathcal{K}(K_{L,j}; P_0) = \mathcal{O}(P_0; \mathcal{D}(\mathcal{O}(K_{L,j}; P_0)_{64\ldots127}; \mathcal{O}(K_{L,j}; P_0)_{0\ldots63})||0\ldots0)$$

where

$$P_0 = (I_{T,0}, I_{T,1}, I_{T,2}\&0\text{xC7}, D_1, D_2\&0\text{x3F}, 0, \ldots, 0)_{2^7}{}^3. \tag{1}$$

Here $I_{T,i}$ and D_j denote i-th and j-th byte of I_T and D respectively.

The key used is chosen from the two most significant bits of the third byte of I_T. These two bits define j and key $K_{L,j}$ is chosen.

Then, the transponder's key is used to generate a 64-bit tag R which both parties compare to gain mutual authentication. R is computed as

$$R = \mathcal{R}(K_T; P_1, P_2) = \mathcal{D}(\mathcal{O}(\mathcal{O}(K_T; P_1); P_2)_{64\ldots127}; \mathcal{O}(\mathcal{O}(K_T; P_1); P_2)_{0\ldots63}).$$

To compare the values, R is split into 32-bit halves R_0 and R_1, where R_0 is sent by the transponder to the cylinder which compares it to its value. If it equals then it sends R_1 to transponder who compares it to its respective half.

The inputs for the response function $\mathcal{R}$ is calculated as

$$P_1 = (c_0, c_1, \ldots, c_{10}, D_6, D_7, D_8, D_9, 0)_{2^7} \tag{2}$$

and

$$P_2 = (I_{L,2}, I_{T,2}, I_{T,3}, D_3, D_4, D_5, 0, \ldots, 0)_{2^7}. \tag{3}$$

[3]This notation shows the total length of the array in bits. In the paper the size was 2^8, but the functions take input of length 2^7 bits.

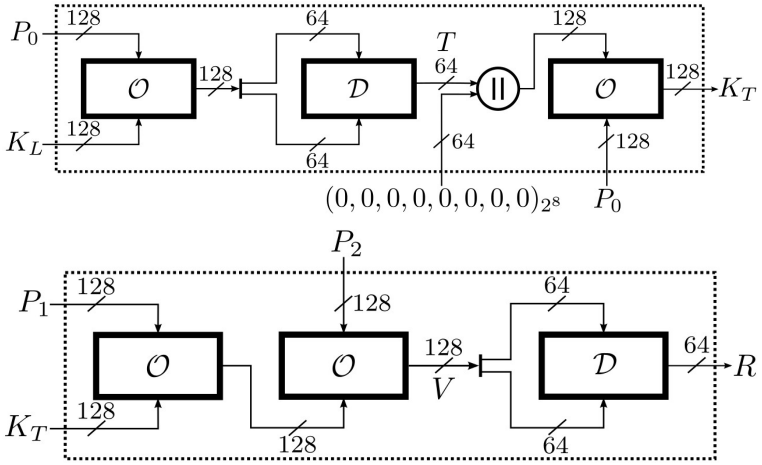


Figure 5: The chaining of obfuscation function and encryption function in key derivation function and response computation function.

3.3 Primitives $\mathcal{O}$ and $\mathcal{D}$

Two primitives are used in key derivation and response calculation. First of the primitives is a encryption function $\mathcal{D}$. Closer inspection revealed that it is a modified DES. The modifications which separated $\mathcal{D}$ from DES were not revealed in [6].

The obscurity function $\mathcal{O}$ seemed to be proprietary development. It works on two 128-bit inputs x and y, where x is the plaintext and y is the key. The output of the function is 128-bit. The algorithm as a pseudocode is referenced in algorithm 1.

Algorithm 1 Obscurity function

INPUT: Plaintext $x = x_0, \ldots, x_{15}$, key $y = y_0, \ldots, y_{15}$

for $r \in \{0, \ldots, 7\}$ **do**

$$z_0 = \begin{cases} RC_0 & \text{if } r = 0 \\ (x_{15} + 2(y_{15} + z_{15}) & \text{if } r \in \{1, \ldots, 7\} \end{cases}$$

for $i \in \{0, \ldots, 15\}$ **do**

$$z_i = \begin{cases} (x_{i-1} + 2(y_{i-1} + z_{i-1})) & \text{if } i \in \{1, 2, \ldots, 15\} \setminus \{4, 8, 13\}, \\ (x_{i-1} + 2(y_{i-1} + z_{i-1}) + RC_{r+1}) & \text{if } i = 8, \\ ((y_{i-1} + z_{i-1}) + x_{i-1} >> 1) & \text{if } i \in \{4, 13\}. \end{cases}$$

$$x_i = x_i + y_i + z_i \mod 2^8$$

end for

end for

return $x_0, \ldots, x_{15}$

4 Cryptanalysis

In [5], Klimov and Shamir gave a new construction for functions whose invertibility can be easily checked. These functions are called T-functions.

Definition 1. A function f from $\{0,1\}^{m\times n}$ to $\{0,1\}^{l\times n}$ is called a T-function if the k-th column of the output $[\Phi(x)]_{\star,k}$ depends only on the first k columns of the input: $[x]_{\star,1},\ldots,[x]_{\star,k}$.

The check of the existence of inverse mapping uses parametric invertible functions and triangular Feistel networks.

Definition 2. Parametric functions are functions $g(x_1,\ldots,x_a;\alpha_1,\ldots,\alpha_b)$ whose arguments are split by a semicolon into inputs (x_i-s) and parameters (α_j-s).

A parametric invertible function is a parametric function whose input-output relationship is invertible for any fixed value of the parameters:

$$\forall\alpha\forall x,y : g(x,\alpha) = g(y;\alpha) \leftrightarrow x = y.$$

Definition 3. A triangular Feistel network(TFN) is any mapping over bit matrices described by:

$$(p_{\star,0},\ldots,p_{\star,n-1}) \rightarrow (g_0(p_{\star,0}, g_1(p_{\star,1};p_{\star,0}),\ldots,g_{n-1}(p_{\star,n-1};p_{\star,0},\ldots,p_{\star,n-1}),$$

in which g_0 is an invertible function and function g_i for $i>0$ are parametric invertible functions.

To check if the inverse exists for a function, it must be checked if it is a TFN and then if its parametric mappings are parametric invertible functions. The technique for performing the testing used enumerating all input values, and so for a bitwise matrix, m should be relatively small.

In [6], the authors made the following important observations about the protocol:

1. Part of the internal DES key used in response computation function $\mathcal{R}$ is used as a part of the next challenge. Explicitly, equation

   $$(C_2^{(t+1)}, C_3^{(t+1)}, C_4^{(t+1)}, C_5^{(t+1)}, C_6^{(t+1)}) = (V_8^{(t)}, V_9^{(t)}, V_{10}^{(t)}, V_{11}^{(t)}, V_{12}^{(t)})$$

   holds. Here t denotes the run number of the response-challenge protocol between the transponder and the cylinder.

2. After 8 rounds the least significant bits of each x cell in obfuscation function $\mathcal{O}$ depends only on 32 bits of the key. More specifically, LSBs of the y cells occur in non-linear combinations and bits next to the LSBs occur only in linear combinations.

   This observation was generalised for any bit b in the 16 output bytes if arbitrary number of instances of $\mathcal{O}$ were chained (as in $\mathcal{R}$).

From these observations, it is possible to map the output of $\mathcal{D}$ in the key derivation to the leaked input bits of $\mathcal{D}$ in the response computation. This allows to discard $\mathcal{D}$ from the analysis and replace it with some function of leaked key bits.

The reduction is nontrivial. We denote by $F : \mathbb{F}_2^{128} \times \mathbb{F}_2^{128} \times \mathbb{F}_2^{128} \times \mathbb{F}_2^{64} \to \mathbb{F}_2^{40}$ the mapping

$$F(P_0, P_1, P_2, T) \to \mathcal{O}(\mathcal{O}(\mathcal{O}(P_0; T||0 \dots 0); P_1); P_2)_{64 \dots 103} = V_L, \qquad (4)$$

where T is the output of the DES function in the key derivation phase.

From 1, 2 and 3 we see that these values are known as they do not depend on keys. V_L is the permutation of leaked bits.

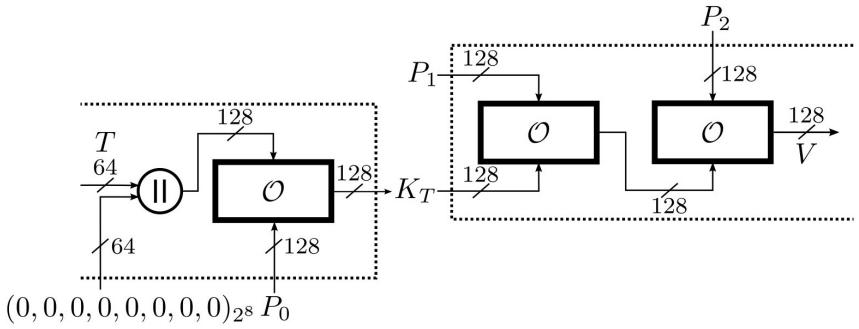


Figure 6: Combination of a part of the key derivation function and part of the response computation function which is being inverted.

Splitting 4 in eight 5-bit chunks the equation can be written as

$$F(P_0, P_1, P_2, T) = \begin{pmatrix} F_7(P_0, P_1, P_2, T) \\ \vdots \\ F_0(P_0, P_1, P_2, T) \end{pmatrix}.$$

Now, it was observed that not all F_i were dependent on all 64 bits of T. As an example, F_0 depends only on 30 bits of T where 7 of this bits are linearly dependent. If we denote by $T_{S_i^{(0)}}$ the bits of T on which F_i non-linearly depends on and by $T_{S_i^{(1)}}$ the bits of T on which F_i linearly depends on, then each F_i can be expressed as

$$F_i(P_0, P_1, P_2, T) = F_i(P_0, P_1, P_2, T_{S_i^{(0)}}) + L_i(T_{S_i^{(1)}}),$$

where L_i is linear.

As each $S_i = (S_{i^{(0)}} \cup S_{i^{(1)}}) \in S_{i+1}$, one can set up a recursive attack on F. Firstly, one guesses the bits of $T_{S_i^{(0)}}$ and sets up the linear equation system for the bits of $T_{S_i^{(1)}}$ and solves it. Solved values are used to find the bits of T_{S_i}. As $T_{S_6} = T_{S_7} = T$, then the candidate solutions can be tested against F_6 and F_7 to exclude false-positives.

5 Attacks

In conclusion, SimonsVoss System 3060 is susceptible to several attacks:

1. **Cloning of the transponder**: Using the methods introduced in section 2, one can read out the key from the transponder. Using specially built hardware it is possible to gain authorisation from the cylinders.

   If full-erase is performed on PIC, then the configuration bits are erased and it is possible to reprogram the transponder. This allows to copy existing transponders.

2. **Reading out the cylinder keys**: Similarly, the cylinder keys can be read out from the cylinder. We saw in section 3 that the transponder keys are derived from the cylinder keys. If the transponder ID is known, then using cylinder keys, it is possible to generate the key used by this transponder.

   Transponder ID can be obtained in several ways. Direct approach is to let the transponder send it by initiating the challenge-response protocol by pressing the button and recording the ID sent by the transponder.

   The lock also contains list of valid transponder IDs and it is possible to choose a convenient ID. Furthermore, frequently low IDs are used for emergency transponders (for rescue and police). One can guess the ID and generate a matching key.

   After generating the transponder key, it is possible to program an erased transponder with the corresponding key to be used to enter the room.

   There are several methods on how to obtain cylinder part of the installation. Firstly, one could use a publicly accessible door (front entrance or other frequently opened doors). Secondly, in some observed installations the PCBs were facing towards the exterior environment and could be obtained by simply opening the casing.

3. **Cryptanalytical attack**: Using the method described in section 4, it is possible to derive a transponder key if the ID of a valid transponder is known.

   To employ this attack, a cylinder has to be found where the transponder with corresponding ID has access to. The challenge-response protocol is initiated, challenge C is captured and authentication data D is sent. The protocol is then halted.

   This procedure is repeated several times until enough challenges have been obtained.

# pairs	avg. running time	avg. # key candidates
2	3.36 min	21.34
3	11.5 sec	1
4	1.2 sec	1
5	0.65 sec	1

Figure 7: Running time and the number of key candidates corresponding to the number of challenges captured

It is needed only to capture several challenges (see figure 7).

4. **Brute force**: If P_0 is known, then the key derivation function $\mathcal{K}$ has only 64 bits of entropy, which corresponds to the output of $\mathcal{D}$. Capturing the communication between a valid transponder and a cylinder, one can find try all values K'_T for K_T and compare $\mathcal{R}(K_T; I_L, I_T, C, D)$ to R and output K_T on success.

5.1 Mitigation

The findings were reported to SimonsVoss who is developing a patch to the firmware to circumvent the cryptoanalytical attack. The patch can be deployed over-the-air using remote management hardware.

To mitigate the physical attacks, SimonsVoss is looking to replace the hardware used in the transponders and cylinders which are developed for security applications.

As part of the mitigation, specific details of different attack scenarios were not disclosed in the article.

6 Conclusion

Although there are flaws in the initial SimonsVoss keyless locking system, their use of DES would imply that SimonsVoss has put enough effort into making the system secure. But for a successful attack it is only required to have a small flaw in the design or in the implementation.

Not less important is readiness to admit having drawbacks in the system and cooperating with researchers to fix the problems. There have been cases where the manufacturers have denied having problems or even have pursued for a warrant to forbid publishing relevant articles.

References

[1] Decapping ICs. `http://www.youtube.com/watch?v=mT1FStxAVz4`.

[2] Microchip PIC16F886 data sheet. http://ww1.microchip.com/downloads/en/DeviceDoc/41291G.pdf.

[3] SimonsVoss Smart Handle 3062 technical specification. http://www.simons-voss.com/Euro-Profile.1192.0.html?&L=1.

[4] Andrew Huang. Hacking the PIC 18F1230. http://www.bunniestudios.com/blog/?page_id=40, 2005.

[5] Alexander Klimov and Adi Shamir. A new class of invertible mappings. In BurtonS. Kaliski, etinK. Ko, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 470–483. Springer Berlin Heidelberg, 2003.

[6] Daehyun Strobel, Benedikt Driessen, Timo Kasper, Gregor Leander, David Oswald, Falk Schellenberg, and Christof Paar. Fuming acid and cryptanalysis: Handy tools for overcoming a digital locking and access control system. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 147–164. Springer Berlin Heidelberg, 2013.

[7] Ted the Tool. MIT Guide to lock picking. 1991.

[8] Barry Wels and Rop Gonggrijp. Bumping locks. 2005.