J. Yaghob (Ed.): ITAT 2015 pp. 38–42
Charles University in Prague, Prague, 2015

About Security of the RAK DEK

Richard Ostertág

Department of Computer Science, Comenius University,
Mlynská dolina, 842 48 Bratislava, Slovakia
ostertag@dcs.fmph.uniba.sk

Abstract: The RAK DEK operating unit is a standalone access control system. This unit, and its more advanced versions, are widely used in Slovakia to protect entrance doors to block of flats. In this paper we have studied security of RAK DEK with respect to timing attack. We have tried two attack vectors. This system shows to be invulnerable to our first attack, but we have succeeded with the other attack vector. Now we are in state of finishing functional exploit using identified vulnerability and investigation of its applicability to the more advanced version of this family of access control systems.

1 Introduction and Basic Description of the RAK DEK

The RYS is a Slovak company that develops and sales access control and door communication systems. This company develops its own line of access control systems based on iButton (a.k.a. touch or digital electronic key – DEK) and the RAK DEK operating-memory units.

These systems were designed for the apartment buildings and became very popular. They are also used to provide access control in commercial or industrial settings (e.g. hotels, offices, stores, schools, server housing) [1].

We choose to discuss this system because of its popularity in Slovakia. We have already described cloning of DEK and generally applicable brute-force attack in [2]. In this paper we have exploited specific properties of RAK DEK, so our conclusions apply only to this specific system. However, described timing attack may be applicable even to the other systems using 1-Wire protocol and serial number iButtons, but actual applicability has to be individually investigated.

1.1 Operating-Memory Unit

The operating-memory unit, e.g. RAK-DEK (see figure 1) is the brain of RYS access control system.

This unit is connected through its RELE output with door's electromagnet and through 4-pin connector on back-side with an iButton touch probe. This unit is capable to store serial numbers for hundreds of iButtons. If a user touches the touch probe with a DEK, the iButton serial number is transferred from the DEK to the operating-memory unit. If the transferred number is stored in the unit, the unit temporarily deactivates the electromagnets (using the RELE output) and the user is allowed to enter.

Figure 1: The RAK-DEK operating-memory unit

We are interested in the communication between the DEK and the operating-memory unit. As the DEK is just a standard DS1990R serial number iButton® from Maxim Integrated Products, Inc., this communication uses standardized 1-Wire protocol.

1.2 Serial Number iButton

The DS1990R is a rugged button-shaped data carrier, which serves as an electronic registration number. It is produced in two basic sizes (F3 and F5) as is schematically depicted on figure 2.

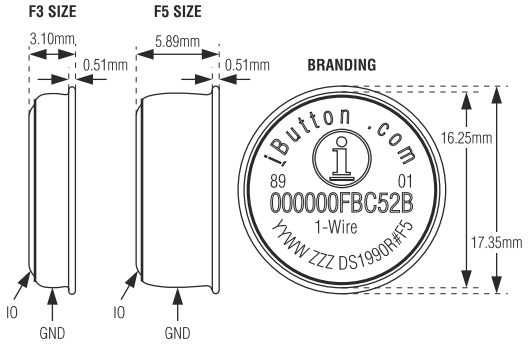

Figure 2: Schema of DS1990R serial number iButton

For the DEK an iButton of F5 size is used, together with a plastic holder for it (see figure 3). This holder can be put on a key chain and can be in different colors (but black is usually used).

Figure 3: Picture of DS1990R-F5 serial number iButton

Every DS1990R is factory lasered with a guaranteed unique 64-bit registration number that allows for the absolute traceability. This 64 bit registration (or serial) number has internal structure as depicted in figure 4.

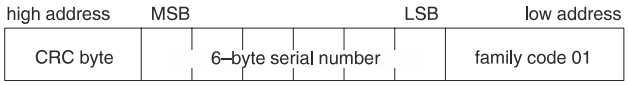

Figure 4: Data structure of a DS1990R serial number

It contains: six-byte device-unique serial number, one-byte family code and one-byte CRC verification. Every DS1990R have family code fixed to $(01)_{16}$. There are also another iButton devices with different family codes. E.g. $(10)_{16}$ is a temperature iButton, but they are not usually used in this kind of systems. Therefore every DEK can be considered as a 48 bits long factory set unique number (analogous to unique MAC addresses of network cards).

1.3 Communication Protocol between RAK-DEK and iButton

All iButton devices utilizes the 1-Wire protocol, which transfers data serially, half-duplex, through a single data lead (1-wire) and a ground return (GND).

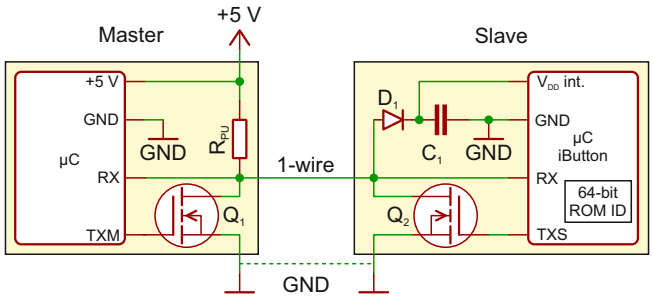

Figure 5: Simplified schema of an iButton and a master

Figure 5 depicts simplified implementation of the 1-wire communication using two micro-controllers with two unidirectional ports. The slave (in this case iButton) has no power source and is powered from an operating-memory unit using the parasite power system on data lead. This system consists of diode D_1 and capacitor C_1 and provides power to iButton during low voltage states of 1-wire bus.

The master uses input port RX to sense value on 1-wire bus. The slave uses its RX input port the same way. In the idle state 1-wire bus is pulled up to 5 V by resistor R_{PU}. In this state all RX ports read logical one. Standard defines that voltage should be at least 2.2 V to be interpreted as logical one.

If any device wants to set 1-wire bus to logical zero, it uses its output port (TXM or TXS) to activate its internal MOSFET switch (Q_1 or Q_2) to connect the data lead to the ground. As a result of this action, 1-wire voltage falls down to near 0 V. Standard defines that voltage should be at most 0.8 V to be interpreted as logical zero.

If device wants to set 1-wire bus to logical one, it just deactivates its internal MOSFET switch. If more devices set 1-wire bus state at the same time, then resulting state is logical AND of all states. In other words: if at least one device is setting 1-wire bus to logical zero, then resulting state is logical zero.

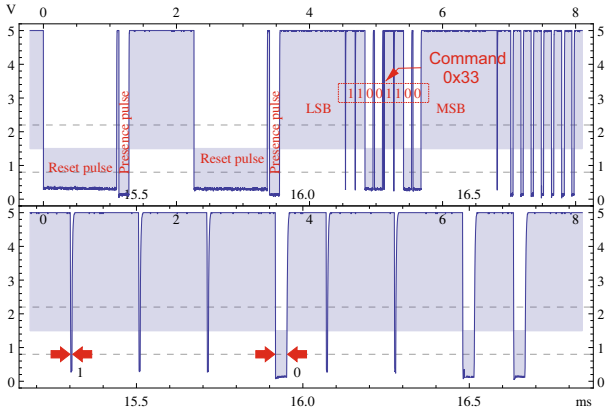

Figure 6: Example of real 1-wire communication

Communication always starts by the reset pulse issued by the master. The reset pulse is just long enough (in this case 1.1 ms) logical zero state of 1-wire bus (see figure 6). After this reset pulse all slave devices are reseted to well-known initial state. All slave devices respond to the reset pulse by the presence pulse, in this case with length of 0.149 ms. If no presence pulse is detected by the master, then no iButton is connected to the master. In this situation RAK-DEK waits for 100 ms and then tries again with another reset pulse. After successful detection of iButton, RAK-DEK makes a new, unnecessary, reset pulse for unknown reasons (again followed by the presence pulse).

After presence pulse, the master will send a command. RAK-DEK always sends the command 0x33, i.e. the “read

ROM" command. This command is transferred from the master to the slave by serial transfer within defined time slots. Any time slot is initiated by the master (in this case RAK-DEK) and starts by falling edge on the data lead. After 0.025 ms (after this falling edge), the iButton read state of the 1-wire bus. If it is at least 2.2 V, the master sends bit 1, otherwise bit 0. Bits are always sent from the least significant bit to more significant bits.

After receiving the "read ROM" command, the iButton is ready to send its 64-bit serial number stored in its ROM. Again, transfer is done in time slots initiated by the master from LSB to MSB. So, the slave is waiting for the falling edge. After 0.004 ms (after this falling edge) RAK-DEK turns off the switch Q_1 and the pull up resistor will raise the data lead to 5 V. So if iButton wants to send bit 1, it has just to wait. If iButton wants to send bit 0, then in this 0.004 ms interval iButton activates its switch Q_2 for 0.032 ms. In either case RAK-DEK reads state of 1-wire bus about 0.02 ms from the beginning of time slot. And again, if it is at least 2.2 V, then master receives bit 1, otherwise bit 0. In figure 6 we can see first 8 bits of serial number after command 0x33. In the case of DEK it is always 0x01 (family code). Lower half of figure 6 zooms to the last but one byte of serial number (in case of this specific key it is $(00110111)_2 = (37)_{16}$.

Communication ends when RAK-DEK receives whole 64-bit serial number. If received number is on internal list of authorized DEKs, then RAK-DEK releases electromagnet holding the doors. At this point RAK-DEK sends the reset pulse and the whole communication starts again. For more implementation details of the protocol see [3].

2 Hardware

To be able to interact with RAK-DEK we need to implement an iButton emulator. We decided to use an Arduino compatible hardware platform developed at Slovak University of Technology – Acrob [4], depicted on figure 7.

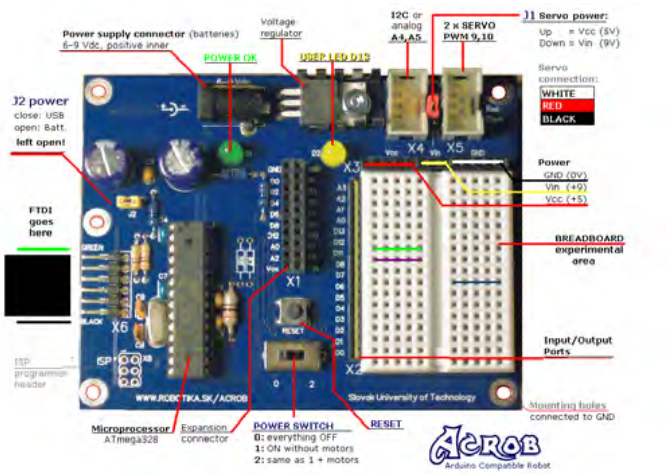

Figure 7: Acrob – an educational robotic platform

This hardware platform uses the Atmel ATmega328P microcontroller running on 16 MHz, which we programmed in C++ like language, using standard Arduino IDE [5].

In contrast to our previous paper [2], where we have simulated operating-memory unit by Acrob, now we have to buy a real RAK-DEK operating-memory unit, because timing attacks are very sensitive to implementation details. We still use one Acrob device for emulation of iButtons.

The 1-Wire protocol uses only one data line. We implement this line by connecting together digital pin 12 of Acrob, with the center pad of touch probe (this is equivalent to connecting directly with pin 2 of the RAK-DEK). This probe is connected to the RAK-DEK using 4-pin connector on the back-side of PCB. To establish a ground return we connect Acrob GND pin with outside ring of touch probe (this is equivalent to connecting directly with pin 1 on the RAK-DEK).

The touch probe gives us one more information channel – the LED. RAK-DEK is blinking with this LED to make it easier to locate the touch probe at night. Also the LED lights up for some time when iButton touches the probe.

To be able to analyze even this source of information we decided to use a photoresistor facing to the LED in the touch probe. We used a photoresistor module with an opamp used as a comparator and a potentiometer for setting a threshold. When light intensity is over the threshold, then DO pin of the module is on logical 0 level (near 0 V), otherwise it is on logical 1 level (near 3.3 V because we have used 3.3 V as V_{cc} for the module). We have connected DO pin on the photoresistor module to pin 8 on Acrob.

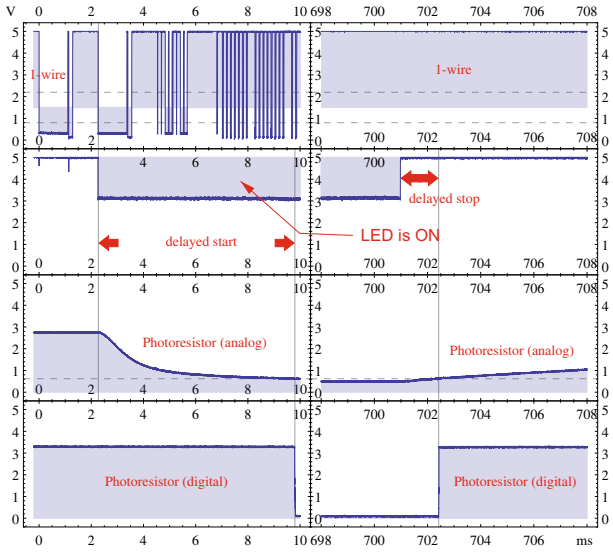

Figure 8: Calibration of the photoresistor module

Photoresistors are slow and that is why we can see a delayed start and a delayed stop in figure 8. We have rotated the potentiometer to set the threshold around 620 mV. By this calibration we obtained a small stop delay at cost of longer start delay and hight sensitivity to ambient light. In

this case it was not problem. We know, that LED starts to lit at the start of second reset pulse and the ambient light was shielded. In fact, the length of the stop delay is not important, we only need it to be constant. If smaller delays are needed then phototransistor can be used.

3 The Brute Force Attack

If we omit the predictable parts of serial numbers (i.e. family code and CRC), we have to find six bytes. Our empirical observations suggest that serial numbers are allocated in sequence. All keys we have seen so far had zeros in two most significant bytes of these six bytes. Therefore for a brute force attack it would be sufficient to try all 2^{32} serial numbers of the form mentioned above.

In our experiments we have observed that RAK-DEK is issuing the reset pulse every 100 ms when waiting for DEK. But if DEK is found, then next rest pulse does not come immediately, but always after 700 ms from the first. This does not leave any space for timing attack and substantially increases time for the bruteforce attack that we have estimated in [2]. If we assume 700 ms as an upper bound to try one serial number, we will need $700\,\text{ms} \times 2^{4\times 8}/60/60/24/365.5 \approx 95$ years for a successful brute force attack in the worst case.

4 The Timing Attack

As a last resort we have tried to analyze time that elapses from the moment we send 64-bit serial number to the moment LED goes off. Ours idea was to store one key, e.g. 0x0000000000000000 into RAK-DEK unit and then emulate two keys, e.g. 0xFF00000000000000 and 0x00000000000000FF, and measure time needed for the LED to go off in both cases. Through this experiment we have realized that RAK-DAK is firstly validating CRC and family code. It is not possible to do tests with an unrealistic DEK. Therefore we choose one valid DEK and make modifications only to its 6 inner bytes in such way to not modify resulting CRC. Then we tried to send four different keys to RAK-DEK with different positions of the first discrepancy from stored key. Resulting times are depicted in figure 9.

From this figure we can see, that RAK-DAK is clearly comparing DEK bytes form LSB to MSB, because time is increasing as position of first discrepancy goes to more significant bytes. Also we can see a nice linear relationship between the position and the time. Using a linear regression we estimated it to be:

$$f(p) = (1.33\,\text{ms})p + 307.96\,\text{ms}$$

Based on this liner regression we can say that test of one byte from electronic key takes approximately 1.3 ms. To verify correctness of this hypothesis we loaded some random DEKs into RAK-DEK. Then we tried to identify

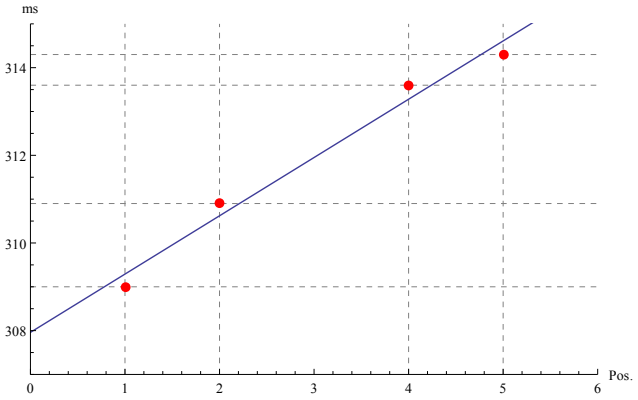

Figure 9: Position of first discrepancy vs. LED lit time. Positions are numbered from right (LSB) to left (MSB).

value on position 1 (position 0 always has value of 0x01). But our implementation did not work. Finally, we found that RAK-DAK is comparing key bytes from LSB to MSB, but firstly it checks if CRCs are equal. This is probably an optimization to speed up comparison of long byte sequences in case we have their CRCs already precomputed.

Using this information, we can do much better then brute force attack. We still need to search through the key space, but we can do it byte by byte now. Starting from CRC (at position 8) and then going from position 1 to 5, calculating value at position 6 in such way not to change resulting CRC. If we see that system response delayed by 1.3 ms we know, that we hit correct value for actual position and we can advance to next position, until correct DEK is found. Using this technique and our experience of position 5 and 6 to be zero on all known DEKs we can estimate time of successful attack, in worst case, as:

$$700\,\text{ms} \times 4 \times 2^8/60 \approx 12 \text{ minutes}.$$

5 Conclusion

We have investigated possibilities of timing attacks on RAK-DEK. We identified timing attack vulnerability exploiting LED on the touch probe. We are now in state of finishing a functional exploit using identified vulnerability and investigation of its applicability to more advanced version of this family of access control systems. This attack requires only access to an Arduino compatible device and a photoresistor (cost around 30.00 €). The time needed for this attack is less than 12 minutes.

On the other hand, this attack can easily be mitigated by disconnecting LED in the touch probe from RAK-DEK. Better solution would be to modify firmware of RAK-DEK to turn off LED with next reset pulse (which is already fixed to 700 ms after beginning of communication).

This work was supported by VEGA grant 1/0259/13.

References

[1] RYS: Access control and door entry systems. (http://www.rys.sk/html_eng/english.htm) [Online; accessed 8-July-2015].

[2] Ostertág, R.: About security of digital electronic keys. In: ITAT 2013: Information Technologies – Applications and Theory, North Charleston: CreateSpace Independent Publishing Platform (2013) 122–124 ISBN: 978-1490952000.

[3] Maxim Integrated Products, Inc.: Book of iButton standards (application note 937). http://www.maximintegrated.com/en/app-notes/index.mvp/id/937 (2002) [Online; accessed 8-July-2015].

[4] Balogh, R.: Acrob - an educational robotic platform. AT&P Journal Plus **10** (2010) 6–9 ISSN 1336-5010. http://ap.urpi.fei.stuba.sk/balogh/pdf/10ATPplusAcrob.pdf [Online; accessed 8-July-2015].

[5] Arduino: Arduino software. (http://www.arduino.cc/en/main/software) [Online; accessed 8-July-2015].