Technical Sciences, 2022, 25, 117–137

UNIWERSYTET WARMIŃSKO-MAZURSKI W OLSZTYNIE

Yearbook peer-reviewed scientific journal
ISSN 1505-4675
e-ISSN 2083-4527
TECHNICAL SCIENCES
Homepage: https://czasopisma.uwm.edu.pl/index.php/ts/

DOI: https://doi.org/10.31648/ts.7634

METHODS OF IMPLEMENTATION AND OPERATION OF HARDWARE DRM PROTECTION ON THE EXAMPLE OF PRINTER CARTRIDGES

Wojciech Cybowski

ORCID: 0000-0002-6062-2157
Faculty of Mathematics and Computer Science
University of Warmia and Mazury in Olsztyn

Received 19 March 2022, accepted 01 August 2022, available online 30 August 2022.

K e y w o r d s: printer, cartridge, ink, chip, reset, protection, DRM, reverse engineering.

A b s t r a c t

Tens of millions of printers are sold each year, including copiers and multifunction devices. Each printer needs an ink or toner cartridge, depending on the type of printer. This is where the problems arise that anyone who owns such a device has encountered – the price and availability of cartridges. A set of original cartridges for the printer we bought can sometimes cost more than the printer itself. On the other hand, it is not always possible to buy cheaper replacement consumables. This is often due to the fact that the manufacturer of the printer has implemented a Digital Rights Management (DRM) protection in the chip connected to the cartridge. Bypassing such protection is not trivial, and when a way to do so is learned, it is not disclosed because of the commercial value of such knowledge. The goal of this paper is to determine if it is possible to find out for ourselves how selected DRM protections work by using reverse engineering, and to develop a way to reset these protections so that we are not tied to the price and availability of consumables on the market, as we can remanufacture our cartridges ourselves, which is especially important in the context of the Right to Repair.

Correspondence: Wojciech Cybowski, Katedra Metod Matematycznych Informatyki, Wydział Matematyki i Informatyki, Uniwersytet Warmińsko-Mazurski, ul. Słoneczna 54, 10-710 Olsztyn, e-mail: wojciech.cybowski@uwm.edu.pl, https://github.com/wcyb.

Introduction

Currently, the vast majority of people using a computer also have a printer, and probably each of us once needed to print something. Usually, however, we do not think about the most important element of the printer – an ink cartridge or a toner cartridge. As long as the printer is printing, this item is ignored by the user. The problem occurs when the ink runs out. Then users can buy the original cartridge or its replacement. It would seem that the problem ends with choosing one of the above options – but the problem lies in the business model used by printer manufacturers. Often, the substitutes give way to the quality of the original cartridges, offering a low price in return. However, the problem arises when the cartridges are electronically protected, which has become the norm nowadays.

The aforementioned business model of printer manufacturers, called the razor and blades model (DHEBAR 2016), consists of selling printers at a low price, with minimal profit or even without profit, in return the loss is recovered from the sale of original cartridges. In search of increasing profits, manufacturers not only use electronic security in their cartridges, but also physical modifications (ROBINSON 2013). In the photo below is an example of reducing the capacity of the ink by the manufacturers – the HP300 cartridge from 2002 (left) and 2010.

Fig. 1. HP300 Cartridge
Source: based on ROBINSON (2013).

The obstacle on the way to cheap printing is mainly the hardware DRM used in the cartridges. In the past (about 15 years ago) it was enough to add ink or add toner and printing could be continued. Currently, often it is not possible to add ink to the cartridges without physically modifying them, due to their construction. However, adding ink is only half the problem – the electronic security measures or DRM used in currently produced cartridges means that even after adding ink, the printer will still not print, reporting lack of ink.

The method of bypassing these safeguards depends on the cartridge itself. Currently, two types of cartridges are used – integrated with the print head

or separate from it, in the form of an ink (or a toner) tank. For new printers in which the first type is used, there are often no substitutes on the market. The only option are refurbished cartridges. This means that the used cartridges have been filled with ink and new stickers have been attached, but the security features are not always resetted, which means that the printer will see the cartridge as empty. This is because the DRM may be too complicated or new at the moment to be bypassed. The reason may also be different – a small market of replacement manufacturers. It should be mentioned that the production of replacement cartridges means physical enclosures, materials and integrated circuits, and not only the ink itself, which production is much easier, so more companies are involved in its production for various industries.

Due to the fact that the sale of cartridges is a big source of income for printer manufacturers, and because security features of cartridges are still evolving, a replacement manufacturers are faced with a difficult task, when they have to develop new substitutes that would be fully compatible with the originals. It also means the total absence of any publicly available technical documentation about the protections or workarounds for them.

For a regular user, this often means that cheap printing is not possible. However, where protections have been broken, there are two options. For cartridges without an integrated print head, one can usually buy new chips, resetters for used chips that reset implemented protection or buy a replacement cartridge. Those with an integrated print head are now a much bigger problem because the protection chip is built into the print head itself.

In the introduction, one can find a general overview of the problem that is caused by DRM protections, as well as a general description of the difficulties that one has to face in order to solve this problem. The next section contains a selection of related papers that can be consulted for a better understanding of how DRM protections of various kinds work and how they are used. It also includes papers that show other approaches to solving the problem by using invasive reverse engineering. In the following chapter, the legal aspect related to the issue described in this paper is briefly discussed, based on both European and American legislation. The subsequent chapter is an explanation and description of the tools and terms used in this work with which not all readers may be familiar. The next and most important chapter describes the whole process of analysing how DRM protections used in selected printer cartridges work, how to bypass them and how to make DRM resetters. In this chapter the reader will also find a general description of how to analyze generic DRM protection of printer cartridges, as well as a comparison of the types of DRM protections currently in use. The final chapter is a summary of the work done.

Related work

In this work, it was decided to use non-invasive reverse engineering methods, because this was the lowest costs approach and at the same time was possible without the use of specialized equipment. The only exception was the analysis of the first target, where a fairly simple method of silicon die extraction was used. However, this method did not allow this operation to be carried out in a precise manner, which caused the silicon dies to be slightly damaged.

Alternative solutions to one of the problems, which is extracting data from the chip for later analysis, have been described by authors of other papers (SAMYDE et al. 2002, COURBON et al. 2016), but these solutions are more costly, require specialized equipment, and are invasive, which may not always be desirable.

Because DRM systems vary widely depending on what they are intended to restrict access to, it is not easy to create a summary that describes them all. One can find papers (CECHNER et al. 2018, NICKOLOVA, NICKOLOV 2008) that describe in general terms common use cases of DRM systems for various tasks, sometimes also demonstrating with an easy example how to circumvent the sample protection. There are also papers (LEE et al. 2018, DABHOLKAR et al. 2021) that describe the operation of a given type of DRM by means of examples, which are directly related to reality.

Legal Aspect

This chapter briefly introduces the legal aspect related to the topic covered in this work. It is also important to note – all the solutions presented in this paper are author's work, so the code and other resources of printer manufacturers covered by any rights have not been used.

Within the European Union, the use of reverse engineering to create programs similar to those analyzed is permitted by Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs (EUR-lex, online), as well as by the judgment of the Court of Justice of the European Union in Case C-406/10 (2012), which permits, among other things, the actions described in this paper. In the U.S. the law is more complicated (SAPP 2006, LAZARIDIS 2009, ADELMANN 2010), but reverse engineering in the manner described in this paper appears to be allowed, but for one's own safety, while in the U.S. and before proceeding with the activities described in this paper, it is best to seek the advice of an appropriate lawyer.

Due to the above mentioned regulations, resetters as well as replacement chips for cartridges from different manufacturers can be easily bought in countries around the world.

At this point we should also mention very important initiatives taking place both in the U.S. and Europe, namely the Right to Repair (Right to Repair European Campaign, online, European Parliament Think Tank on the Right to Repair, online, Right to Repair U.S. Campaign, online, Assembly Bill A7006A, online, Senate Bill S4104, online). Currently, the manufacturer's obligations towards consumers, who would like to repair or modify purchased equipment on their own, are not regulated, which results in lack of availability of electronic diagrams, parts and software needed e.g. for servicing. At present, there are initiatives which may change this, so that in the future, the manufacturer of electronic equipment will be obliged to provide consumers with electronic diagrams or parts needed to repair their equipment.

Terms and tools used

Terms

Below are terms that may not be obvious to the reader, but were used in this work:

– PCB – (short for Printed Circuit Board) mechanically supports and electrically connects electronic components by means of conductive paths, leads and other elements etched from copper sheets laminated on a non-conductive base. Components (for example, capacitors, resistors or active components) are generally soldered on a circuit board. Advanced PCBs may contain components embedded in the substrate;

– Chip – a popular term for an integrated circuit, i.e. a circuit containing from several to hundreds of millions of basic electronic components, both active and passive;

– Microcontroller – a small computer in a single integrated circuit. The microcontroller contains one or more processor cores along with memory and programmable input/output peripherals. Program memory in the form of NOR flash or OTP ROM (one-time programmable read-only memory) is often placed on the chip, along with a small amount of RAM. Microcontrollers are designed for embedded applications, as opposed to microprocessors used in personal computers or for other general applications;

– Resetter – an electronic tool used to reset the security measures used in the ink chip placed on the cartridge;

– Firmware – (also called embedded software) in electronic systems and computers, firmware is a type of computer program that provides control, monitoring and manipulation of created products and systems. Typical examples of devices containing embedded software are consumer devices, computers and peripherals;

– ASIC – (short for Application-Specific Integrated Circuit) is an integrated circuit (IC) tailored to the specific application, custom-designed and not intended for general use;

– I^2C (Inter-Integrated Circuit) – multi-receiver, multi-transmitter computer bus, created by Philips Semiconductors (now NXP Semiconductors). It is usually used for communication of low speed integrated circuits with processors and microcontrollers in short distance communication between PCBs;

– SPI – serial communication interface used in short distance communication, mainly in embedded systems. The interface was developed by Motorola in the late 1980s and has become a de facto standard.

Tools

The following tools were used to perform the steps described in this work:

– KiCad – an open source electronic design automation package (EDA for short). Facilitates the design of electronic circuit diagrams and their conversion into PCB designs. KiCad was originally developed by Jean-Pierre Charras. It has an integrated environment for electronic schematics and PCB design. The package includes tools that allow one to create a set of materials (so-called BOM), graphics on PCBs, Gerber files, and 3D views of PCBs and its elements;

– Logic – software provided to operate equipment created by the American manufacturer of electrical measuring instruments, Saleae. They sell mixed- signal logic analyzers as well as logic analyzers working only with digital signals;

– Atmel Studio – integrated development platform (IDP) for creating and debugging applications for AVR and SAM microcontrollers. It provides a hassle-free and easy-to-use environment for writing, building and debugging applications created in C/C++ or assembler. It also connects to debuggers, programmers and development kits that support AVR and SAM devices;

– Arduino – open source electronic equipment as well as a company dealing with its development, which designs and manufactures development kits, using mainly AVR microcontrollers, used to build devices and interactive objects that, with the help of sensors, can sense and control objects in the physical world;

– Binwalk – fast, open source, easy-to-use tool for analysis, reverse engineering and extracting data from the firmware.

Targets analysis and solutions development

What are we dealing with

Before we dive into the analysis of specific examples, the DRM protections being analyzed will be presented in a more general overview. Namely, this chapter will introduce the general operation of these protections, their types, evaluate their effectiveness, and provide general information on reproducing the operation of the generic cartridge DRM protection.

How does it work and what is it used for

The DRM protections of cartridges were created by printer manufacturers in order to limit and hinder the use of replacement consumables as much as possible. For some, this problem may seem insignificant, but it can be better illustrated by an analogy – what if the only way to buy the necessary parts for our car was to visit the manufacturer's authorised service centre? The result would be often high prices, inadequate to the value of the parts, along with limited availability and production period of those parts for a given car model. After all, in this scenario, the manufacturer would have a complete monopoly. This is currently the case with printer cartridges, where the manufacturer guarantees the availability of consumables for a limited period of time, e.g. 5 years from the end of production of a given printer model. On top of that, cars, like printers, often last long after production ends, and if there were no spare parts available from independent manufacturers, consumers would be left to dispose of a working device. Hence, it can be seen that DRM protections are not needed for the printer to function. How generic DRM protection works can be summarised as follows: the DRM-carrying chip on the cartridge contains data defined by the manufacturer. This data can include, for example, the serial number, date of manufacture and so on. Only the manufacturer of the device knows how to read and write this information, its format and meaning. In order for the printer to start printing when a new cartridge is inserted, it must first read this information. If the information read matches what the printer software expects, then the user can use their device. Otherwise, the printer refuses to function. In addition, during or after printing, the data on the chip is updated by the printer with, for example, the number of printed pages, used printing material, etc., so that the chip ultimately has a dual purpose – as a DRM carrier and cartridge status information.

Types of DRM protections in cartridges

The Table 1 shows and compares the types of DRM protections that can be encountered in printer cartridges.

Table 1

Table comparing DRM systems found in printer cartridges

DRM type	Effectiveness	The most effective methods of circumventing
ASIC	very high	invasive reverse engineering printer firmware analysis
Hardware encrypted semiconductor memory	high	side channel attacks invasive reverse engineering printer firmware analysis
Radio-frequency identification (RFID) tag	high	non-invasive reverse engineering printer firmware analysis
Semiconductor memory	medium	non-invasive reverse engineering printer firmware analysis
Fuse	low	replacement with a new one

As can be seen from the table above, not all types of DRM protections have been addressed in practice in this work. This is because, for example, in the case of DRM with encryption, the only difference from the process outlined in this paper for DRM without encryption would be the additional step needed to decrypt the data before analyzing it. There are various ways to do this, both invasive and non-invasive, e.g., analyzing the printer firmware, or a side channel attack. This is of course not trivial, and it is possible that it is even an issue worth a separate paper. However, due to the lack of a printer using this type of DRM at the time of writing this paper, the complexity of the problem, and the fact that the goal of this paper was not to develop a way to break hardware-implemented cryptographic functions in secure semiconductor memories, the study of this type of DRM protection, which, as mentioned, formally differs little from those presented in this paper, was omitted.

It is also worth mentioning here something that cannot necessarily be called a security feature, but which is shown in the table. We are referring to the use of a fuse as a very simple carrier of binary information. This works in a simple way – a new cartridge is fitted with a fuse with a specific interrupt current. When a cartridge with such a fuse is inserted into the printer, the printer checks for continuity of the circuit of which the said fuse is a part. If the printer detects continuity of the circuit, the printer then blows the fuse, which marks such cartridge as used. At the same time, the printer resets the counter implemented internally in its firmware, and starts counting up again. This protection is very easy to bypass – it is enough to place a new fuse in the cartridge so that the

process can be repeated. This protection can also be used in the opposite way, that is, the fuse is blown only after the printer calculates that the cartridge has been used up. This type of protection can also be found in photosensitive drums in laser printers.

How to analyze DRM protection of cartridges

The exact process describing how the analysis was performed, the tools and software used, and how the data maps were created is described in the following sections, separately for each chip analyzed. Here, the general process will be presented.

First, visually examine the chip to be analyzed, noting how many contacts it has as well as what components and how they were mounted on it. For this step, as well as subsequent ones, you will need a basic knowledge of electronics. After such an examination, one should already have a guess as to the hardware layer, i.e. what communication protocols can be used (depends on the number of contacts), for what voltage was the chip designed and if it is built with standard components.

With this knowledge we can proceed to the next step, which is intercepting the communication of the chip with the printer. Since after the initial analysis we know which contacts are used to power the chip and which have a different purpose, we should connect a logic analyzer to the contacts with an unknown purpose. This will allow us to capture the data sent to and from the chip. We should capture the data during several important moments, such as turning the printer on and off and printing the same test document several times. One can create and print documents that fit on one and two sheets of paper so that it is easier to locate, for example, data containing a print counter.

Next, a preliminary analysis of the captured data should be performed. If one suspects that a particular protocol such as SPI or I^2C is being used for communication, then one can use a decoder for that protocol, which should be available in the software to control one's logic analyzer. This will allow to split the captured data into packets and get a general idea of the contents of those packets.

The next, and penultimate, step is to correlate the printer's operation with the captured data. Having the data from the previous step and knowing that it was captured after printing one page of the test document, we can compare that data with subsequent data obtained after repeating the same operation. In the same way, having captured data from different events such as printing a blank page, printing a completely blackened page, printing multiple pages, and so on, it is possible to determine what changed in the data and how. Based on this, the data map can be created.

The last step is to verify the correctness of our data map, which can be most easily achieved by changing on the chip, in the appropriate place of its memory, the value describing the level of usage of the printing material in the cartridge. If, after this change, the printer reports a different level of print material usage, it can be concluded that we managed to interpret at least part of the captured data correctly. The process of resetting the DRM protection itself can be automated by creating a resetter that, based on the data map, will on its own modify specific locations in the memory of the security chip in an appropriate manner.

As mentioned in the introduction, the topic raised in this work is quite extensive and complex, so it is not possible to discuss all security features in every possible variant, therefore three different printers have been selected that will be examples of how to analyze and circumvent hardware DRM used in cartridges. For each of analyzed printers, a different approach to the problem of analyzing the implemented DRM was used, which will show that there are many ways to achieve the same goal, but not all are just as simple.

DRM 1

The first target was an inkjet printer. It uses separate cartridges for each of the primary colors (CMYK). In this printer, as well as the other two, the protection is in the form of a chip integrated with the cartridge, which chip, however, can be separated from the cartridge itself. Below is a photo of this chip from the front (contacts side) and back (elements side).

As can be seen at first glance, the number of contacts on the chip does not seem to match any of the standard interconnection interfaces used in electronic devices. However, the closest standard that could fit is SPI, which uses four data

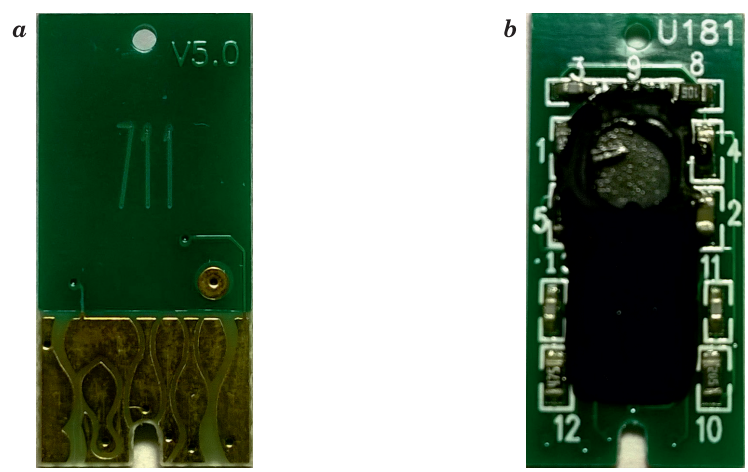

Fig. 2. Chip from the first analyzed cartridge: *a* – front, *b* – back

lines and two to supply power. Due to the fact that the form of communication with this chip does not seem to comply with any standard and due to the presence of additional, unusual in this type of device, components on the PCB, it is probably an ASIC. In the photo on the elements side, one can also see the coil which is a very unusual element in this case, because it does not seem necessary for the protection system. This suggests, along with the large area flooded with black epoxy resin, that we are dealing with not one but two integrated circuits. Pins used to transfer power could be easily identified in the printer's carriage connector using a multimeter and by measuring voltage. Pins on which the voltage value was constant at 3.3V could be considered as power supply pins.

However, the above information are not enough. It is still not known what pins are used to transfer data and what is their format. At this point, it was worth checking to see if any patents were associated with this chip, given its non-standard communication interface. As it turned out, the patent did exist, its designation is US 7,125,100 B2. It not only describes the different designs of the output contacts, but also their functionality. This information was very helpful in understanding the functions of individual pins, but as one can see in the previous photos, they did not fully described what was physically placed on the analyzed chip. This was caused by, among other things, by the very existence of this patent. In the analyzed replacement of the original chip, its manufacturer, in order not to violate the patent that described the specific appearance of the pins on the original chips, created they own pattern.

Knowing what to expect, a logic analyzer was connected to the printer's carriage connector in place of one of the chips. After collecting the data, analysis began. It turned out that the tested system seems to be an EEPROM memory with which one can communicate via a modified SPI interface. Thanks to this information, it was possible to start creating a prototype of the resetter using Arduino. However, it turned out to be more difficult than one might think. The problem was to identify the method of addressing the message for a given chip, because each had a different address. After a closer look at the collected data, an anomaly became noticeable, which suggested that the address was the first four bits, and the others were the answer from the chip. This anomaly was a slight cycle timing change during one clock cycle occurring just after these four bits. This moment is presented on Figure 3.

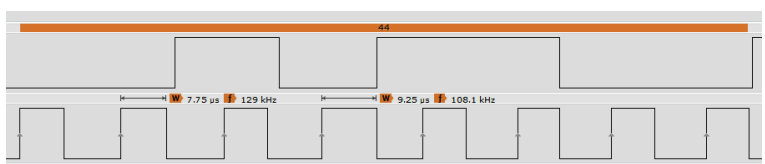

Fig. 3. Clock cycle anomaly – a longer pulse lasting 9.25 µs (right) and a normal pulse lasting 7.75 µs

Using the information obtained, a sketch was created for Arduino, which confirmed the previous theory. Full sketch is available on the author's Github. According to the created code, Arduino first sent half a byte of the chip's address, starting from the most significant bit, and then generated clock pulses to allow the chip to send a response which is the content of its memory. So the data transmitting line was also the receiving line, depending on the current operation. To save the data, one had to send four bits of the address, receive four bits from the chip, and then one could send the data that was to be stored in memory. The transmission was from the lowest address and this could not be changed.

However, the most important information was still missing – what the individual bytes stored in this memory mean. Therefore, their analysis began, printing several pages both in color and only with black text. The result of this analysis is the following data map:

Table 2

Data map for the first analyzed cartridge

Offset / Address	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0x00	CI[1]	ink dot counter			AU[2]	static data		
0x08	static data			0xB2	ink color[3]		static data	
0x10	manufacturer name					static data		
0x18	0x0							

[1] Chip identification: 0x3C – black, 0xFC – yellow, 0xBC – magenta, 0x7C – cyan
[2] Amount of ink used (0x00 – full, 0xFF – empty)
[3] 0xC3,0x65 – black, 0x83,0x66 – yellow, 0x43,0x67 – magenta, 0x03,0x68 – cyan

With the help of the information collected, it was possible to design a resetter that would reset data to the factory values, so that the printer would see the ink with such a chip as full. The same microcontroller model – ATmega48 was used to make it, as well as the other two. It has 4 kibibytes of flash memory, 512 bytes of SRAM memory and 256 bytes of EEPROM memory, which, however, was not needed in this application. The software controlling the microcontroller was written in C using Atmel Studio. Schematics of all the resetters, along with the code needed to make them work, are available on the author's Github.

With the schematic, one could design a printed circuit board based on it. This, as well as all documentation related to design of other resetters, was made using the KiCad package. The first resetter is shown below and pictures of the other two has been omitted, because they were very similar in the design and appearance.

Still, in case of this chip, this was not the end. The functions of the other pins that were not used for data transfer were still unknown. Not having a red fuming nitric acid, which could be used to dissolve the epoxy resin that integrated

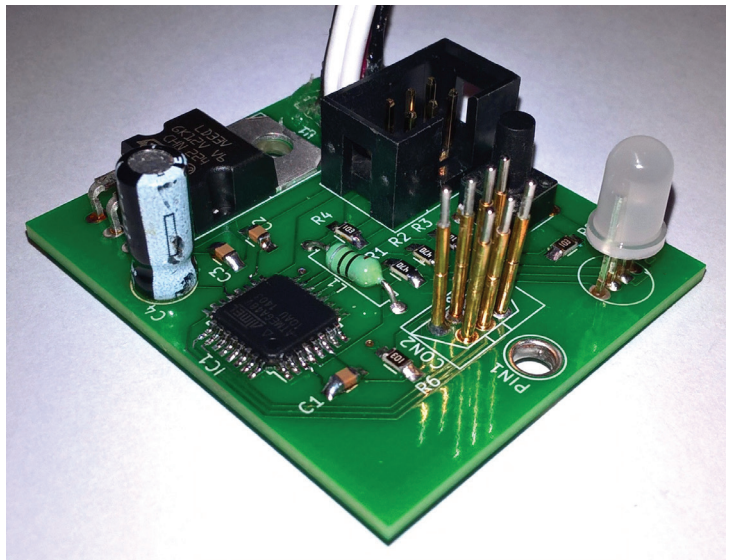

Fig. 4. Created resetter

circuits were flooded in, another approach (definitely safer) was used – using the properties of amorphous bodies, in this case the polymer containing epoxy groups, after heating to temperature T_g appropriate for this resin, called the transition temperature to the vitreous phase, the resin can be softened enough to be removed mechanically quite simply. The right temperature was chosen empirically because it was not known what type of resin was used in this case. After removing the resin, two integrated circuits could be seen – the first appeared to be part of a DC-DC converter. This can be deduced from the elements with which it was connected, including the previously mentioned coil, and from a fairly

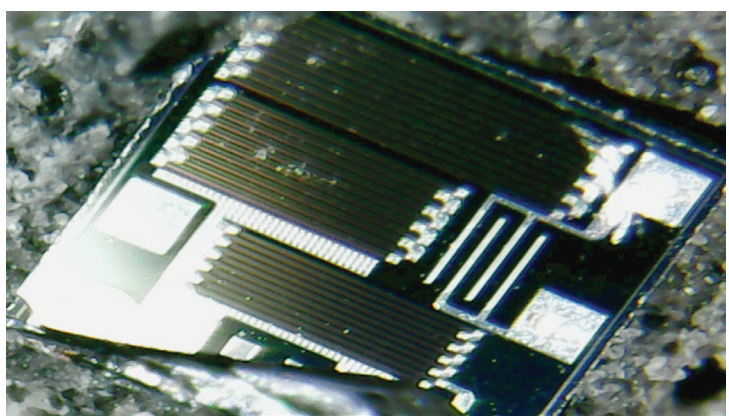

Fig. 5. First integrated circuit

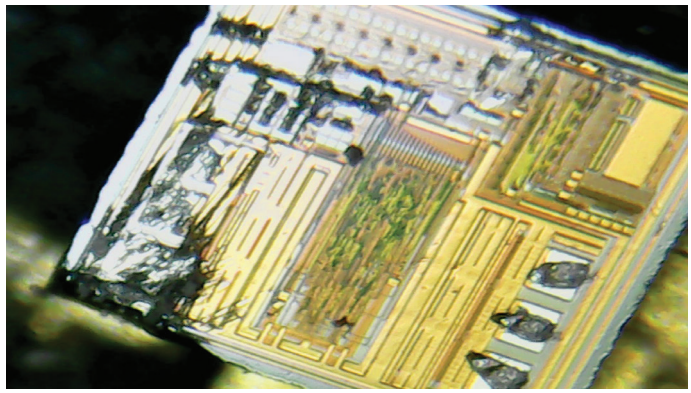

Fig. 6. Second integrated circuit

simple structure located on a silicon wafer. Most likely, it is part of the protection used in this type of chip, but as previous tests have shown, it is not needed to write and read data. The second integrated circuit was the one that contained all the information.

To sum up, in this method it was important to use the information available in the patent to understand how the communication interface is constructed, and then it was necessary to focus on analyzing the data being transmitted and the timing relationships of the signals being used to understand how the analyzed DRM works, which turned out to be an ASIC designed especially for this application.

DRM 2

The second target was a monochrome laser printer. On the toner container, there was only one chip due to the construction of said toner container, that was inseparably integrated with the waste toner container. In this case, the DRM analysis proved easier. As one can see in the pictures below, nothing was flooded with epoxy resin and the number of contacts on the chip indicates the use of the I^2C interface. After searching the Internet for the code found on the integrated circuit on the components side, it was found that it is a regular 24C02 EEPROM memory, i.e. with a capacity of two kilobits, with which one can communicate via the I^2C interface.

Considering the above, the method of writing and reading data was known in contrast to the first target. This made the work easier, but for the sake of confidence, another sketch was created for Arduino to confirm the assumptions and read the contents of this memory.

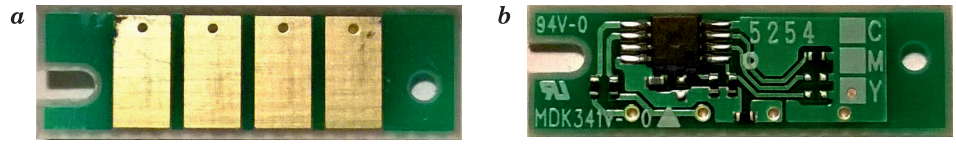

Fig. 7. Chip from the second analyzed cartridge: *a* – contacts side, *b* – components side

After confirming that it is a normal EEPROM memory, one could proceed to analyze the data contained in it. As it turned out, some data was written directly in the ASCII alphabet, but as for others it was not known what they meant. Due to the fact that all components were standard here, there was no patent that could facilitate the work with this chip. It was decided to check whether the manufacturer provides firmware updates for this printer model, but it turned out that there are no updates. Not being able to use the analysis method used for the previous target, and having no firmware available that analysis could help, the only thing left was to use ordinary data analysis similar to that used in the previous target.

A logic analyzer was again used for this, and test pages were printed to see what data is being changed and how it is changing. In this way, the purpose of the most important bytes in this memory could be determined, but the meaning of some data was still unknown. The English version of Wikipedia was helpful in the analysis, where in the article about laser printers (*Laser printing*, online) the exact operation was described, so that it was possible to associate changes of data with unknown purpose with the specific operation of the printer. However, there were still characteristic, uninitialized bytes between the data changed by the printer. As the analyzed chip came from the starter cartridge, it could be assumed that uninitialized data is related to this fact. For this reason, a new chip was bought, intended for ordinary cartridges (which did not differ from the starter, except for the amount of toner inside) and after reading the data from it, the purpose of uninitialized data in the chip from the starter cartridge was clarified. Thanks to that, there was enough information to create a data map which is presented below.

This knowledge allowed, in addition to resetting the data in the chip itself, to change the chip type from starter to normal. Now it was possible to design another resetter. Scheme of it did not differ much from the previous one, in this ase a built-in microcontroller module for communication via I^2C was used, in contrast to the previously used *bit-bang* method, i.e. manual change of states on the output pins.

To sum up, it was important here to learn exactly how this type of printer works, due to the lack of other possibilities to obtain information, to understand how the used DRM works, which in this case turned out to be a simple EEPROM memory where the data stored in it described the condition of the toner cartridge.

Table 3

Data map for the second analyzed cartridge

Address \ Offset	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0x00	M[1]	CV[2]	B[3]	R[4]	cartridge type[5]			0x0
0x08	IT[6]	0x0	EDP code					
0x10	cartridge serial number							
0x18	0x0				control number			
0x20	printer serial number							
0x28					PT[7]	OW[8]	0x0	
0x30	0x0							
0x38								
0x40	PP[9]				0x0			
0x48	0x0							
0x50								
0x58	control number				0x0			
0x60	0x0							
0x68	CC[10]		0x0					
0x70	amount of toner used				0x0		WT[11]	BF[12]
0x78	0x0							

[1] Model
[2] Cartridge version
[3] Brand
[4] Region of destination of the cartridge (e.g. Europe)
[5] 0x1,0x1,0x3 – starter, 0x3,0x1,0x1 – standard
[6] The initial amount of toner in percent
[7] The percentage of toner remaining
[8] Organic photo conductor drum wear
[9] Number of printed pages
[10] Number of cartridge cleaning cycles
[11] The percentage filling level of the waste toner container
[12] Bit field (cartridge status)

DRM 3

The third target was a gel printer. In this printer, like in the first one that was analyzed, a separate ink cartridge with its own chip was used for each color. However, this printer is significantly different in construction from most other similar inkjet printers because it has a separate, replaceable waste ink container, unlike a regular inkjet printer that has a special absorbing material built-in, which from the manufacturer's assumption is not user replaceable. Due to this fact, the container also had a chip.

On closer inspection, it turned out that the chips used in this printer model were the same as the ones used in the previously examined laser printer.

This made it easier to analyze the data because the tools created earlier could be used in this case, but it was still necessary to identify the meaning of the data stored in these chips. The analysis began with checking on the manufacturer's website whether it provides firmware updates. It turned out that, unlike the previous case, here updates were available. Access to the firmware enabled a different approach to analysis of data (and thus DRM) found in the chips used here, compared to previous printers. The file with the latest firmware was downloaded and its analysis began.

The analysis was carried out using the binwalk tool. Not knowing the structure or organization of data in the downloaded firmware, the functions analyzing the content of the file from the aforementioned program were used. It turned out that the firmware contained data packed into most likely two archives in the *gzip* format. Compression was also indicated by the high entropy of the data, a plot of which could be generated with the -E switch.

After using the binwalk program to cut out the first archive found and extract the data from it (one also could use standard Linux tools, i.e. *dd* and *gzip*, respectively), a single file was obtained which was then analyzed. It turned out that the obtained file contained further archives, as well as the program probably responsible for part of the upgrade process.

Using the same method as before, another two archives were extracted from the previously obtained file. After extracting the files from them (there was one file in each), as before, they were analyzed to find out what they contain. As it turned out, the first file contained the firmware update logic, but these information were not needed, so its content was not further analyzed.

The file containing the sought data was the second of the previously obtained. It contained all the printer control software and its network interface files, again packed into the *gzip* archive. Since the analysis of the compiled printer control code, and thus the process of communication with chips and the changes of relevant data, would be tedious and time-consuming, it was decided to check whether the software contained a debug interface, and thus messages that could contain useful information and thanks to this shorten the analysis time.

Using the standard Linux system tool, i.e. *strings*, which was searching for ASCII characters, it was possible to obtain information from the analyzed file, that were very helpful in developing a data map for the chips from this printer. The information obtained through this tool, however, were not enough to be able to immediately create a data map from them, because they did not refer to any specific data from currently used chips. It was known what data was saved in the chip, but it was not known where. That was a problem which was solved thanks to the quite often used functionality in modern electronic devices, i.e. the hardware debugging port. It is used by the manufacturer when creating equipment, but also in the factory and by service technicians during repairs.

Thanks to it, it was possible to send a command to the printer that requested information about currently used chips. This finally confirmed the correctness of previously obtained data, as well as helped in creating a data map. Using the information obtained, the data maps were created, respectively for the chip from the ink cartridge and the waste ink container (Tab. 4, 5).

Table 4

Data map for the third analyzed cartridge

Offset / Address	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0x00	M[1]	CV[2]	B[3]	R[4]	Type	IC[5]	0x0	0xFF
0x08	IIA[6]	0x0	0xFF				ISR[7]	0xFF
0x10	cartridge serial number[8]							
0x18	0xFF							
0x20	EDP code							
0x28	PL[9]	0x0						
0x30	0xFF							
0x38								
0x40	IED[10]	ink amount		printer work status (PWS)				
0x48	PWS	IIC[11]	HRC[12]	HCC[13]	AIC[14]	ISF[15]	SF[16]	0xFF
0x50	0xFF						CPC[17]	
0x58	BPC[18]		EPC[19]		TPC[20]			
0x60	amount of ink used							
0x68	0xFF							
0x70		cartridge status						
0x78	printer serial number							

[1] Model
[2] Cartridge version
[3] Brand
[4] Region of destination of the cartridge (e.g. Europe)
[5] Ink color – 1: black, 2: cyan, 3: magenta, 4: yellow
[6] The initial amount of ink in percent
[7] The number of refills of the ink delivery system
[8] Two digits are coded on each byte
[9] Percentage of ink left (100 means full, 255 means not initialized)
[10] Ink expiration date
[11] The amount of ink used during cartridge initialization
[12] The number of print head refresh cycles
[13] The number of print head cleaning cycles
[14] The number of air intrusions into the ink delivery system
[15] The number of fills of the ink delivery system
[16] Status flag
[17] The number of color pages printed
[18] The number of black and white pages printed
[19] The number of economy color pages printed
[20] The number of total pages printed

Table 5

Data map for the analyzed waste ink container

Address \ Offset	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0x00	M[1]	CV[2]	B[3]	CN[4]	0x0	AIU[5]		
0x08	AIU	serial number (SN)						
0x10	SN	TF[6]		TCF[7]	0x0			
0x18	printer serial number							
0x20	0x0							
0x28					TPC[8]			
0x30	0x0			WSF[9]	0x0			
0x38	CPC[10]							
0x40	0x0							
0x48								
0x50								
0x58	tank status		0x0				SF[11]	WSF
0x60	WSF		0x0			WSF	0x0	
0x68	0x0			AIS[12]	CPC			
0x70	0x0	BPC[13]				counter (C)		
0x78	C	0x0						
0x80	AIS	0x0						ISF[14]
0x88	0x0							
⋮	⋮							
0xFF	0x0							

[1] Model
[2] Cartridge version
[3] Brand
[4] Control number
[5] Amount of ink used
[6] Threshold value indicating a full tank
[7] Threshold value indicating a tank close to full
[8] The number of total pages printed
[9] Work status flag
[10] The number of color pages printed
[11] Status flag
[12] The number of air intrusions into the ink delivery system
[13] The number of black and white pages printed
[14] The number of fills of the ink delivery system

Having the above information, one could start creating the resetter. As mentioned in the introduction, the chips used in these cartridges turned out to be the same as the chip from the previous printer, so the changes in the resetter scheme consisted only in replacing the single-color LED with the RGB LED, so that operations could be signaled with a color suitable for the given cartridge.

To sum up, in this method, it was important to analyze the firmware and obtain data from it, which allowed to learn how the DRM works. As before, it turned out to be EEPROM memory in which individual bytes were information about the state of a given cartridge or waste ink container.

Conclusions

In the course of our work, we were able to confirm that it is possible to reconstruct the operation of DRM protections used in selected printer cartridges. It was also shown through examples how the hardware DRM systems used in printer cartridges work and how they can be circumvented. In addition, we developed resetters, which allowed to reset the security features implemented in selected cartridges without any specialist knowledge. Thanks to these actions, it is possible to reduce the cost of using printers, which is important both for an ordinary user with one device and for a company owning dozens of printers. There is no single, reliable method that can always guarantee success when it comes to figuring out how DRM systems work. It should also be noted that no metallurgical microscope or other specialized equipment was used in the analysis of the described examples, so the average reader should be able to apply the presented methods to other hardware DRM systems on their own. The documentation created and the designs of the resetters were decided to be made available as an open source project on the Github account of the author of this paper, so that those who will want to continue this research on the printers available to them will have a starting point and an idea of what to expect. The DRM systems presented in this paper were used in a total of 60 different models of printers, produced by two manufacturers.

References

ADELMANN T.C. 2010. *Are Your Bits Worn Out? The DMCA, Replacement Parts, and Forced Repeat Software Purchases*. http://www.jthtl.org/content/articles/V8I1/JTHTLv8i1_Adelmann.PDF.

Assembly Bill A7006A. https://www.nysenate.gov/legislation/bills/2021/A7006.

CECHNER B.T., MATHUR A., JAVAID A.Y. 2018. *Software Cracking: An Exploration of the Physical Realities of Software*. https://ieeexplore.ieee.org/abstract/document/8500125.

COURBON F., SKOROBOGATOV S., WOODS C. 2016. *Reverse engineering Flash EEPROM memories using Scanning Electron Microscopy*. https://www.cl.cam.ac.uk/~sps32/cardis2016_sem.pdf.

DABHOLKAR A., KAKARLA S., SAHA D. 2021. *Looney Tunes: Exposing the Lack of DRM Protection in Indian Music Streaming Services*. https://arxiv.org/abs/ 2103.16360.

DHEBAR A. 2016. *Razor-and-Blades pricing revisited*. https://www.sciencedirect.com/science/article/abs/pii/S0007681316000124.

Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs. https://eur-lex.europa.eu/ legal-content/EN/TXT/?uri=CELEX%3A32009L0024.

European Parliament Think Tank on the Right to Repair. https://www.europarl. europa.eu/thinktank/en/document/EPRS_BRI(2022)698869.

Judgment of the European Court of Justice in Case C-406/10. https://curia. europa.eu/juris/liste.jsf?language=en&jur=C,T,F&num=C-406/10&td=ALL.

Laser printing. https://en.wikipedia.org/wiki/Laser_printing.

LAZARIDIS N. 2009. *Anti-Circumvention Provisions Circumventing Copyright Law*. https://ssrn.com/abstract=1426193.

LEE S., KIM K., KANG J., KIM H. 2018. *Bypassing DRM protection in e-book applications on Android*. https://ieeexplore.ieee.org/abstract/document/8343109.

NICKOLOVA M., NICKOLOV E. 2008. *Hardware-based and Software-based Security in Digital Rights Management Solutions*. http://hdl.handle.net/10525/270.

Right to Repair European Campaign. https://repair.eu/.

Right to Repair U.S. Campaign. https://www.repair.org/.

ROBINSON D. 2013. *Printer ink cartridges: why you're paying more but getting a lot less*. https://www.theguardian.com/money/2013/feb/23/printer-ink-cartridges-paying-more-getting-less.

SAMYDE D., SKOROBOGATOV S., ANDERSON R., QUISQUATER J.J. 2002. *On a New Way to Read Data from Memory*. https://www.cl.cam.ac.uk/~rja14/Papers/SISW02.pdf.

SAPP H.A. 2006. *Garage Door Openers and Toner Cartridges: Why Congress Should Revisit the Anti-Circumvention Provisions of the DMCA*. https://digitalcommons.law.buffalo.edu/cgi/viewcontent.cgi?article=1030&context=buffaloipjournal.

Senate Bill S4104. https://www.nysenate.gov/legislation/bills/2021/S4104.