

# How microprobing can attack encrypted memory

Sergei Skorobogatov  
Computer Laboratory  
University of Cambridge  
Cambridge, United Kingdom  
e-mail: sps32@cam.ac.uk

**Abstract**—This paper exposes some weaknesses of encrypted embedded memory in secure chips. Smartcards and secure microcontrollers are designed to protect confidential internal information. For that they widely employ on-chip memory encryption. Usually both data and address buses are encrypted to prevent microprobing attacks. This paper shows how practical such attacks can be on real chips and whether memory encryption is as good as it is supposed to be. It was possible to extract the whole memory from a secure 8-bit microcontroller with as little as 8 probing needles. This paper questions the usual belief in that ion-doping-encoded and encrypted Mask ROM is ultimately secure. Implications for 16-bit and 32-bit microcontrollers are discussed as well. Some common weaknesses are exposed and possible countermeasures are discussed.

*Keywords*—microprobing attacks; glitching attacks; memory encryption; ROM; EEPROM; Flash memory

## I. INTRODUCTION

Modern semiconductor devices are expected to safeguard the secret information against extraction and modification. In the late 90s memory encryption was used primarily for PayTV smartcards. Today not only do most smartcards employ memory encryption but some microcontrollers use encryption for on-chip or embedded memory.

Microprobing attacks were known for a long time since their practicality was demonstrated on PayTV smartcards [1]. Second-hand equipment necessary for carrying out such attacks is widely available at affordable cost from second-hand resellers including Ebay. However, chip manufacturers have significantly improved chip designs to protect against such attacks. Most smartcards now have top layer sensing mesh which if disturbed with laser cutter or microprobing needle will trigger self-destruction of internal memory making the chip no longer functional. However, even that protection can be defeated with relatively low-cost approaches [2].

Secure chips usually do not employ bootloaders to prevent non-invasive attacks on Mask ROM [3]. Some secure chips have their internal bus encrypted to thwart invasive microprobing by making all captured data useless to the attacker. Defeating that kind of protection is far more challenging. First, because neither the algorithm nor the key location are known. Second, because the access to unencrypted information is buried deep inside the logic.

Even partial reverse engineering is unlikely to help because modern chips no longer employ block structures with nicely laid out bus lines. The logic design usually look as a sea of gates with bus lines scattered across the whole chip. Full reverse engineering of the logic will be too expensive and a time consuming process. On-chip encryption also has another benefit. In addition to address obfuscation in memory array it helps to protect the firmware in Mask ROM from being optically extracted. Without knowing both the algorithm and the key an attacker will be unable to make any use of the information.

The strength of the data bus and address bus encryption is demonstrated with the example on a particular chip. The results presented in this paper show that low-cost microprobing attacks can be successfully used to circumvent data bus and address bus encryption in embedded Mask ROM in a microcontroller with 8-bit CPU core. Challenges for 16-bit and 32-bit CPUs are discussed as well. The attack was carried out in three steps. The first step was aimed at trying to identify the CPU core and gather as much information as possible on it. The second step was aimed at finding the corresponding encrypted values for specific CPU instructions. The third step was used to defeat address bus encryption. As these attacks do not require expensive equipment they can pose a big problem to the hardware community. Without proper countermeasures in place security in some devices could be easily compromised.

Although Mask ROM extraction methods and microprobing attacks were described in [1], there are no publications on newer devices. The samples chosen for this paper are relatively old with 0.35 $\mu$ m fabrication process. The main reason for that is to demonstrate the attack process. Newer devices can be attacked in a similar way, however, more expensive equipment will be required. The main contribution of this paper is to demonstrate that a successful microprobing attack can be performed on embedded devices with encrypted memory.

This paper is organised as follows. Section 2 gives a brief introduction into embedded memory types and their strength against direct extraction from memory array. Section 3 demonstrates optical extraction of Mask ROM contents from smartcard and secure microcontroller. Section 4 shows microprobing attack on encrypted Mask ROM storage. Section 5 discusses possible implications for modern chips.

The impact of the research is discussed in the concluding section.

## II. BACKGROUND

Embedded systems are often based on microcontrollers – small integrated circuits with SRAM, ROM, EEPROM and Flash on a single silicon die. Some modern embedded systems are based on System-on-Chip (SoC) devices. Those have some additional processors and hardware engines to improve functionality and performance. Embedded firmware in microcontrollers and SoCs is stored in non-volatile memory. For that Mask ROM and Flash memory are usually used. Mask ROM is the cheapest storage but it does not allow reprogramming, while Flash memory can be reprogrammed a few hundred times. When more frequent changes are required, for example, for storing variables and system parameters, EEPROM is used. It usually allows up to a million reprogramming cycles and it can be accessed byte wise compared to block erase of Flash memory. Even if the main firmware is stored on a separate chip and loaded on power-up, there is still some non-volatile storage for bootloader. This allows the embedded CPU to boot from that trusted internal memory and then perform some security checks on the external storage.

Although some secure microcontrollers and smartcards encrypt the embedded memory to prevent microprobing attacks, such encryption is usually not cryptographically strong. Contrary to that, the external memory storage is usually encrypted with strong cryptographic algorithms like AES [4] or TDES [5]. There are several reasons why embedded storage is less protected. First, access to that memory is already a big challenge especially for modern deep submicron fabrication processes with more than ten metal layers. Second, strong encryption would require relatively large hardware support and will result in larger die size and, hence, higher production costs. Third, even if a very fast hardware crypto-processor is used, it will still require 8 bytes for TDES and 16 bytes for AES to be fetched from memory. This will introduce additional latency to the program execution. External memory is usually loaded into on-chip SRAM buffers in large blocks. This allows smoother code execution after initial delay.

Very often the internal memory bus encryption is implemented by enforcing XOR operations with some predefined blocks or with custom S-Boxes. In order to improve the security and prevent cryptanalysis such S-Boxes have additional address bus inputs. Ultimate security would involve designing a unique S-Box for each valid address of the on-chip memory. Not only the firmware or bootloader are encrypted but also volatile SRAM memory. This prevents microprobing attacks on data and sensitive variables.

Mask ROM can be implemented in different ways. Mask ROM usually has NOR or NAND structure according to the way transistors are connected inside the memory array [6, 7, 8]. There is an OR structure as well but the only difference between it and the NOR structure is that the transistors are connected to VCC instead of VSS. For each structure the information is encoded in different ways. The information is

placed into the ROM during chip fabrication and cannot be changed later.

In NOR ROM with active layer programming, the logic state is encoded by the presence or absence of a transistor. Information from this type of memory is easily extractable under an optical microscope. However for these technologies de-processing is required to expose the transistor layer covered by the top metal layers which obstruct observation. In NOR ROM with contact-layer programming, the information is encoded by the presence or absence of via plug from bit-line to the active area of a transistor. In old memory technologies, these plugs are visible under a microscope, but in modern memory technologies with planarised layers, de-processing is required to expose the plugs. In NOR ROM with programming using ion implants, the data is encoded by the threshold level of a transistor. This is achieved by creating transistors with different doping levels during fabrication. This type of memory provides a high level of protection against various kinds of attacks because the state of each transistor cannot be observed optically even after the de-processing procedure. This type of memory is very often used in smartcards to prevent code extraction from the memory. More secure chips use NAND ROM which offers more compact design, hence, more challenging to image them. In NAND ROM with metal layer programming, the information is encoded by short-circuiting the transistors. This type of memory has a very low level of protection against optical observation, as these metal fuses are clearly visible under a microscope. Still if the chip has more than two metal layers some de-processing is required to expose the encoding layer.

Unlike Mask ROM, which has only two stable logic states, EEPROM and Flash memory cells store analog values in the form of a charge on the floating gate of a MOS transistor. The floating-gate charge shifts the threshold voltage of the cell transistor and this is detected with a sense amplifier when the cell is read. The maximum charge the floating gate can accumulate varies from one technology to another and normally is between  $10^3$  and  $10^5$  electrons.

EEPROM memory has two transistors – one to select the cell and another with floating gate for charge storage. This allows the memory to be reprogrammed in bytes and it usually has endurance of over a million rewrite cycles. The main disadvantage is the size of memory cells which makes its use more expensive compared to later introduced Flash memory. Flash memory has a simpler structure, faster write and access time but unfortunately it cannot be reprogrammed in single bytes as it can be erased only in blocks, which is not convenient for small data updates. However, some chips employ two-transistor Flash cells designs similar to that of EEPROM to improve cell characteristics and data retention time.

Flash and EEPROM have many different layouts and each semiconductor manufacturer normally has its own memory design. Only NOR Flash structure is used as embedded memory. From the security point of view all floating-gate memories offer very good protection against invasive attacks, because the charge injected during programming is very small, and buried deeply inside the

memory cell, so it cannot be detected directly. De-processing does not reveal any information – only cell structure. The only practical invasive way of extracting the information is by microprobing the internal memory bus. This could be extremely difficult for modern submicron Flash memories which have multiple top metal layers over the data wires. There are some direct memory extraction techniques which allow direct charge detection with special scanning microscopes [9], but they require very expensive equipment and sophisticated sample preparation techniques. In addition, very long time is required to scan even a medium sized memory. As a result such approaches are not considered as a threat to modern smartcards and secure microcontrollers which widely use Flash and EEPROM memory for sensitive data storage.

The most important tool for invasive attacks is a microprobing station. It consists of five elements: a microscope, stage, device test socket, micromanipulators and probe tips. The microscope must have long working distance objectives – sufficient enough to accommodate six to eight probe tips between the sample and the objective lens. It should also have enough depth of focus to follow the probe tip movement.

For simple applications, a manually controlled probing station is enough and can be bought second-hand for less than \$5,000. Passive probe tips are very cheap (less than \$3 each) but active probes are quite expensive, however, they can easily be built from an operational amplifier and a passive tip soldered directly to its input [8].

Usually to extract the information such as memory contents or a secret key, microprobing is applied to the internal CPU data bus. It is difficult to observe the whole bus at a time in one go and various techniques can be used to overcome this. For instance, the same transaction or memory read operation can be repeated many times and then two to four probes are used to observe the signals which then are combined into a complete bus trace.

In silicon chips, the top-layer aluminum interconnect lines are covered by a passivation layer which needs to be removed before the probes can establish contact. The most convenient and easy-to-use passivation layer removing technique involves a laser cutting system. The system consists of the laser head mounted on the camera port of a microscope and the submicron-precision stage to move the sample. Such laser cutters can be bought second-hand for several thousand dollars.

For modern chips with more than three metal layers or smaller than  $0.35\mu\text{m}$  process more sophisticated tools such as Focused Ion Beam (FIB) workstation has to be used. Although such equipment is expensive and require a lot of maintenance, these systems are widely used by universities across the world in physics, material science and engineering. As a result they can be rented for under \$100 per hour rate.

### III. ROM EXTRACTION WITH OPTICAL MICROSCOPE

Two samples were chosen as a target for Mask ROM decryption experiments. One is an old smartcard used in banking industry for EMV chip-and-pin transactions from

2004 to 2008. Another is a custom secure microcontroller used in the car industry.

All the necessary samples were obtained on Ebay, however, only a very limited number of smartcard samples available.

No open access documentation is available for those two chips. However, the CPU type was possible to guess from the manufacturer logo and year of design marked on the chip. For the smartcard it was likely to be Hitachi H8/300 compatible CPU core, while for the microcontroller it was likely to be NEC 78K/0 compatible CPU core. Due to the lack of any programming specification for these chips it was impossible to gain access to the memory via any programming or debugging interfaces. That left the only option for direct memory extraction using optical microscopes.

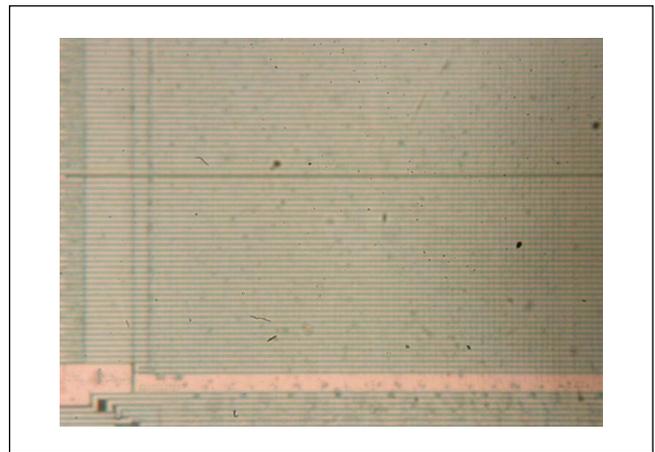


Figure 1. Front side image of the smartcard chip

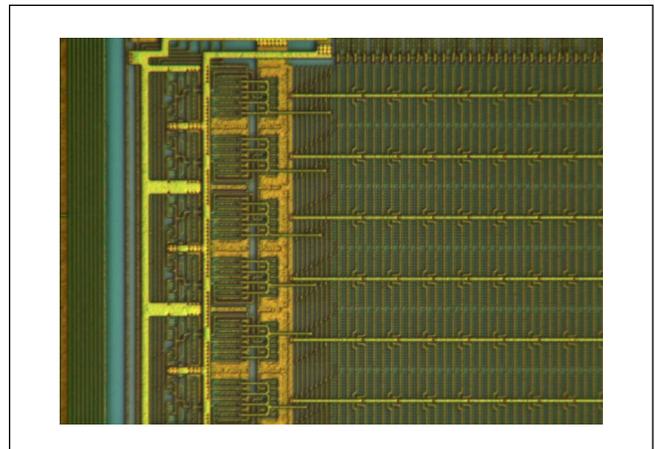


Figure 2. Front side image of the secure microcontroller chip

The first step in direct memory extraction was to figure out the memory type and structure for each chip. To achieve this both chips were decapsulated with fuming nitric acid using a standard procedure [1]. This allowed estimation of

the fabrication process as well as the type and size of embedded memory.

Both chips were found to be fabricated with 0.35 $\mu$ m CMOS process with three metal layers. The smartcard chip has 40kB of Mask ROM and 4kB of EEPROM, while microcontroller has 32kB of Mask ROM and 256 bytes of EEPROM.

The smartcard chip had some additional security features to make hardware attacks harder. The gaps between the metal layer lines were reduced to prevent microprobing attacks due to the higher chance of short circuiting them. Remaining space was filled with some dummy wires extended from existing wires and the routing was done with multiple jumping between different metal layers to obfuscate reverse engineering.

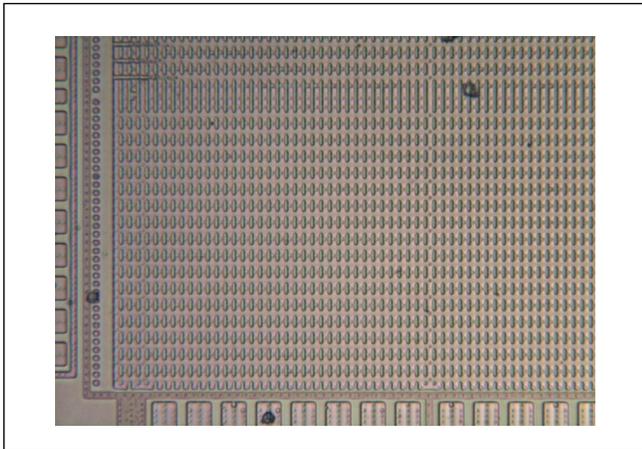


Figure 3. Mask ROM image of de-processed smartcard chip

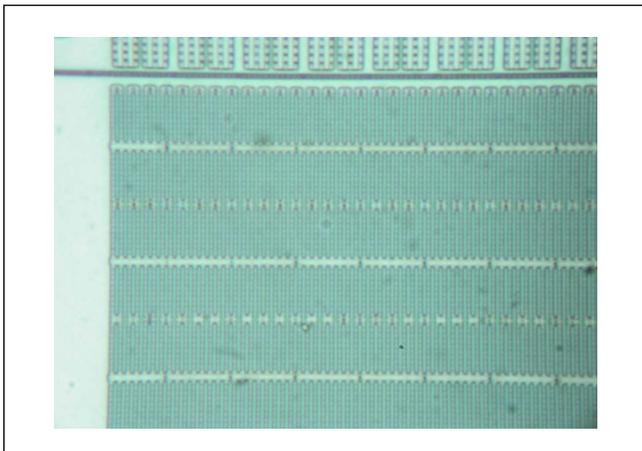


Figure 4. Mask ROM image of de-processed secure microcontroller chip

The Mask ROM areas of the smartcard photographed from the front side is presented in Figure 1, while the same area for the secure microcontroller is in Figure 2. In order to determine the structure of the Mask ROM both samples were de-processed layer by layer down to their transistor layer

with standard technique for top layers removing [9]. The result of this operation for both chips is presented in Figures 3 and 4.

No visible information was observed in any of the layers. The transistor layer reveal the structure of Mask ROMs for the tested chips. For the smartcard it was NOR ion-implanted doping encoded ROM and for the microcontroller it was NAND ion-implanted doping encoded ROM.

Although doping encoded Mask ROMs would deter many attackers there are still some ways of exposing the information using special chemical techniques [9]. In order to reveal the information the transistors have to be etched with doping dependent etchant. For these chips n-Si selective etchant was used. The result of the selective etching for both chips is presented in Figures 5 and 6.

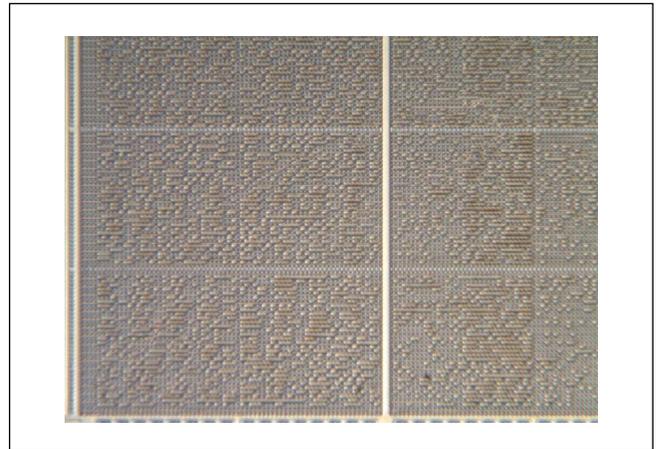


Figure 5. Mask ROM selective etching of the smartcard chip

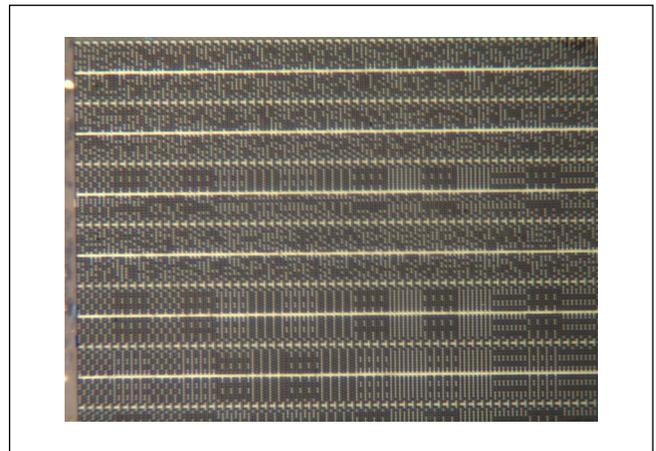


Figure 6. Mask ROM selective etching of the secure microcontroller chip

As it can be observed, both memory arrays have clearly visible patterns of repeated data. This is a strong indication for some encryption or obfuscation being used. The pattern comes from address lines being used as an input to block

encryption function. As predicted, no useful code was extracted from both chips.

There are two ways such encryption can be defeated. One is to reverse engineer the logic placed between Mask ROM and CPU. However, given the complex design of this chip with ASIC-like sea-of-gates this would require a lot of effort. Another approach is to inject data into the bus coming from the Mask ROM in order to build a correspondence table between the encrypted and plain text data.

#### IV. MICROPROBING EXPERIMENTS

In order to verify the idea of possible Mask ROM extraction using microprobing attacks the NEC secure microcontroller chip was used. The initial analysis of the wires coming to the Mask ROM array revealed that the data bus is likely to be multiplexed with eight lower bits of address bus. This fact simplified the attack, because the same probing needles can be used for observing the result of any injected faults into data bus.

Our microprobing station (Figure 7) allowed maximum eight probing needles to be used simultaneously. Therefore, some other means of synchronisation to the CPU cycles had to be used. The internal activity of the chip was observed with a simple power analysis by placing a small resistor into the power supply line and monitoring the voltage drop across it with an oscilloscope.

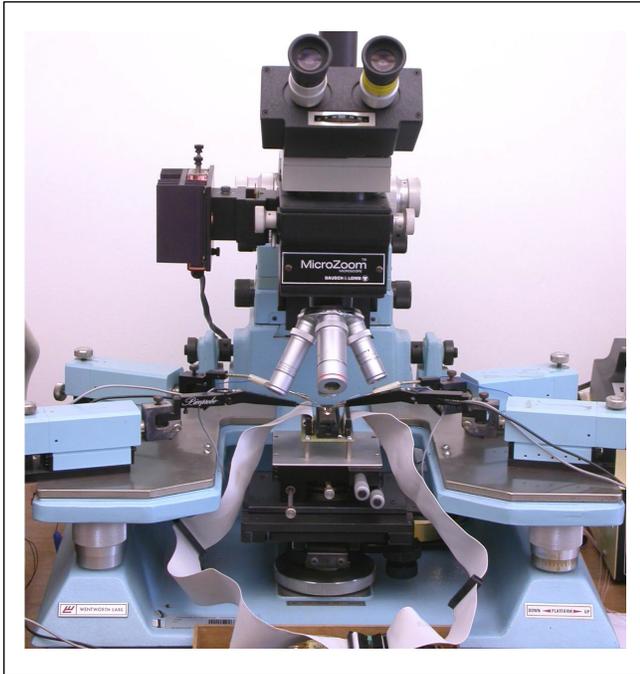


Figure 7. Microprobing setup with eight probing needles on the chip

Special test board was built to accommodate the decapsulated chip and run it. The Microchip PIC24HJ128 microcontroller [10] was used to drive the Reset and Clock inputs of the chip. The chip was constantly powered up from

external 3.3V power supply. The same PIC microcontroller was used to monitor the probed lines and to inject required signals into them. A simple PC program was written to send the commands to the PIC microcontroller via RS-232 interface and to receive the acquired data.

In order to establish a contact with internal wires on the chip surface the passivation layer needs to be removed above the metal wires. For that a laser cutter was used with 100× objective lens. For 0.35µm chip it was only practical to remove the passivation above top metal layer M3. Therefore, all data bus lines were traced to their presence in the top metal layer. Probing needles were placed on all eight data bus lines using micropositioners of the probing station (Figure 8). The view of the chip surface under the probing station microscope is presented in Figure 9 with marking of the bus bits.

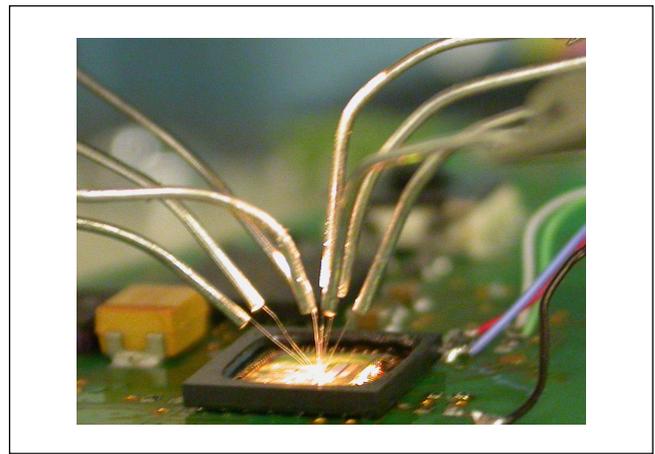


Figure 8. Microprobing setup with eight probing needles on the chip

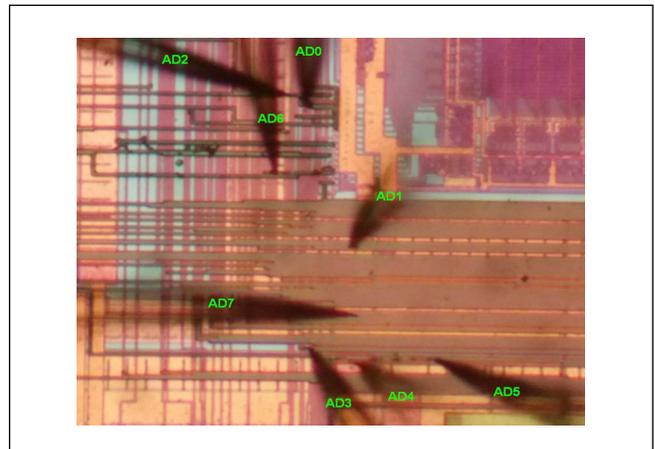


Figure 9. View on the chip surface under microscope

After powering up the microcontroller was run for a short period of time by controlling its Reset line. This allowed to acquire signals from the data bus lines and correlate them with power analysis. It was noticed that after reset the CPU

was running from the internal clock generator at approximately 150kHz. After approximately 1ms time it was switched to running from the external clock. Because the internal clock was not very stable it was necessary to adjust the timing of the injected signals by monitoring them with an oscilloscope. The power analysis observations are presented in Figure 10 for the time shortly after the Reset, and in Figure 11 after 1ms of the program run.

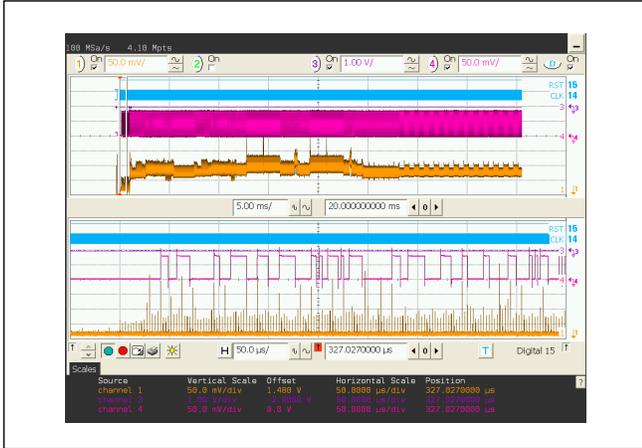


Figure 10. Power analysis combined with microprobing

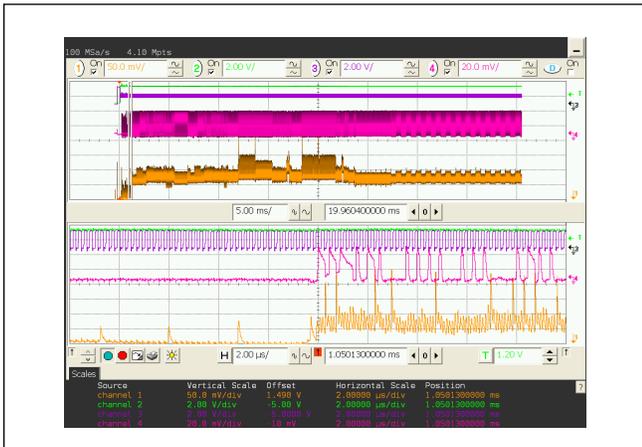


Figure 11. Power analysis combined with microprobing

The acquired information allowed not only to synchronise the signal injection into encrypted data bus, but to also separate address and data on the multiplexed bus. The first two bytes fetched by CPU correspond to the Reset vector. Therefore, any changes to the first byte will immediately result in the address set for fetching the first instruction. However, the address bus was also encrypted. In order to decrypt the address bus the CPU needs to be forced into executing non-branching instructions. This could be performed by supplying different data for the instruction and performing some statistical analysis. Because branch instructions represent only 10% of the whole instruction set,

they can be easily detected by observing non-constant low addresses for different input data.

It was found that the address bus is not actually encrypted but simply XORed with a constant. This allowed the true plain text data to be extracted for the first byte at address 0000h. By injecting data from 00h to FFh into data bus for the first byte the corresponding plain text data were extracted from subsequent address of the first CPU instruction. This allowed to build the S-Box of the decryption function which is presented in Appendix.

The next byte for address 0001h was found by searching the encrypted value that corresponds to plain text value of 00h. This in combination with the first byte decrypted value set to 00h resulted in forcing the CPU to subsequently execute the first instruction at address 0000h. Because the S-Box for that address was known it allowed setting the first byte to be a branch instruction followed by low address value in the next byte at address 0001h. That way it was possible to extract the S-Box decryption table for address 0001h and it is presented in Appendix. In the similar way by forcing the CPU to fetch branch instructions the S-Boxes for addresses 0002h and 0003h were extracted. This allowed execution of a simple code of **MOV A, addr16** instruction to fetch data from memory followed by **BR AX** instruction to put the data on the address bus [11]. As a result the whole memory space of the microcontroller was successfully extracted.

## V. DISCUSSIONS AND FUTURE WORK

The presented microprobing attack is the ultimate implementation of fault injection attack with full control over internal data bus lines. Although the theoretical strength of address bus encryption for 32kB ROM could be as high as  $2^{15!} \approx 9.09 \cdot 10^{133733}$ , in reality this can be cracked relatively easy. This is due to the fact that all CPU programs are executed sequentially. Therefore, a correspondence table for sequential addresses can be reconstructed in a relatively short time.

Data bus encryption offer very good protection unless an attacker can inject arbitrary data into the data bus. This way he can force the CPU to execute a branch instruction which will map the input of encrypted data into plain text data appearing on the internal address bus. Even if the ROM address bus is encrypted it can be defeated in the way discussed above. Theoretical strength of the data bus encryption for 8-bit bus is  $2^8! \approx 8.58 \cdot 10^{506}$  multiplied by the number of bytes. However, CPU execution flow could significantly reduce its strength.

From the extracted correspondence table between cipher text and plain text or S-Boxes for the first two bytes of the memory space one can clearly see their weakness (Tables I and II). Each table can be simplified to an XOR function. That is **XOR B6** for address 0000h and **XOR 3D** for 0001h. This significantly reduces the strength of the encryption. In fact the actual implementation is likely done with using a XOR function with a constant derived from address. This was likely achieved by using a fixed function of address lines fed into the XOR block on the data bus.

TABLE I. S-BOX TABLE FOR ADDRESS 0000

Plain text																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x	B6	B7	B4	B5	B2	B3	B0	B1	BE	BF	BC	BD	BA	BB	B8	B9
1x	A6	A7	A4	A5	A2	A3	A0	A1	AE	AF	AC	AD	AA	AB	A8	A9
2x	96	97	94	95	92	93	90	91	9E	9F	9C	9D	9A	9B	98	99
3x	86	87	84	85	82	83	80	81	8E	8F	8C	8D	8A	8B	88	89
4x	F6	F7	F4	F5	F2	F3	F0	F1	FE	FF	FC	FD	FA	FB	F8	F9
5x	E6	E7	E4	E5	E2	E3	E0	E1	EE	EF	EC	ED	EA	EB	E8	E9
6x	D6	D7	D4	D5	D2	D3	D0	D1	DE	DF	DC	DD	DA	DB	D8	D9
7x	C6	C7	C4	C5	C2	C3	C0	C1	CE	CF	CC	CD	CA	CB	C8	C9
8x	36	37	34	35	32	33	30	31	3E	3F	3C	3D	3A	3B	38	39
9x	26	27	24	25	22	23	20	21	2E	2F	2C	2D	2A	2B	28	29
Ax	16	17	14	15	12	13	10	11	1E	1F	1C	1D	1A	1B	18	19
Bx	06	07	04	05	02	03	00	01	0E	0F	0C	0D	0A	0B	08	09
Cx	76	77	74	75	72	73	70	71	7E	7F	7C	7D	7A	7B	78	79
Dx	66	67	64	65	62	63	60	61	6E	6F	6C	6D	6A	6B	68	69
Ex	56	57	54	55	52	53	50	51	5E	5F	5C	5D	5A	5B	58	59
Fx	46	47	44	45	42	43	40	41	4E	4F	4C	4D	4A	4B	48	49

TABLE II. S-BOX TABLE FOR ADDRESS 0001

Plain text																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x	3D	3C	3F	3E	39	38	3B	3A	35	34	37	36	31	30	33	32
1x	2D	2C	2F	2E	29	28	2B	2A	25	24	27	26	21	20	23	22
2x	1D	1C	1F	1E	19	18	1B	1A	15	14	17	16	11	10	13	12
3x	0D	0C	0F	0E	09	08	0B	0A	05	04	07	06	01	00	03	02
4x	7D	7C	7F	7E	79	78	7B	7A	75	74	77	76	71	70	73	72
5x	6D	6C	6F	6E	69	68	6B	6A	65	64	67	66	61	60	63	62
6x	5D	5C	5F	5E	59	58	5B	5A	55	54	57	56	51	50	53	52
7x	4D	4C	4F	4E	49	48	4B	4A	45	44	47	46	41	40	43	42
8x	BD	BC	BF	BE	B9	B8	BB	BA	B5	B4	B7	B6	B1	B0	B3	B2
9x	AD	AC	AF	AE	A9	A8	AB	AA	A5	A4	A7	A6	A1	A0	A3	A2
Ax	9D	9C	9F	9E	99	98	9B	9A	95	94	97	96	91	90	93	92
Bx	8D	8C	8F	8E	89	88	8B	8A	85	84	87	86	81	80	83	82
Cx	FD	FC	FF	FE	F9	F8	FB	FA	F5	F4	F7	F6	F1	F0	F3	F2
Dx	ED	EC	EF	EE	E9	E8	EB	EA	E5	E4	E7	E6	E1	E0	E3	E2
Ex	DD	DC	DF	DE	D9	D8	DB	DA	D5	D4	D7	D6	D1	D0	D3	D2
Fx	CD	CC	CF	CE	C9	C8	CB	CA	C5	C4	C7	C6	C1	C0	C3	C2

The results for data extraction were achieved on 8-bit CPU with multiplexed address and data bus. There will be

obvious implications for 16-bit or 32-bit CPUs with separate buses. However, on practice the address bus can be probed with a single probe one wire at a time. The necessity for the control of all 16 or 32 bits at a time will be dictated by the implementation of the cryptographic function. If like with our case of address dependent XOR functions, those CPUs will use independent 8-bit S-Boxes there will be no need for full control as each byte will influence different address lines.

On the countermeasures side it is obvious that a stronger cryptographic algorithm applied to wider data bus will be much harder to break. However, such algorithms require a lot of computational power and are not suitable for small microcontrollers and smartcards. Still there could be found a compromise between simplicity of S-Boxes and high latency of proper encryption algorithms.

The results presented in this paper were based on mid-range 8-bit and 16-bit microcontrollers fabricated with 0.35µm process. Devices with smaller topology will likely require more sophisticated tools, such as Focused Ion Beam (FIB) machines to establish the connections to the memory bus. This will inevitable increase the cost of the attack and deter many attackers. But still the vulnerability of a weak encryption cannot be ignored.

Further work could involve experimenting with 16-bit secure microcontrollers with encrypted on-chip memory. However, given their likely fabrication process of being 180nm or 130nm this will inevitably require the use of a FIB machine.

## VI. CONCLUSION

The research presented in this paper showed that embedded memory encryption on its own does not provide absolute protection against eavesdropping and direct fault injection into data bus. Whenever a CPU is involved in memory access the security of encrypted data is under threat. This is caused by the fact that many CPU instructions leak data on the address bus. Although many secure chips employ address bus encryption this is usually even easier to break than data bus encryption. By design most CPUs execute sequential code thus leaking all the necessary information needed to reconstruct unencrypted address values.

This paper presents an affordable and practical approach to program code extraction from the encrypted on-chip Mask ROM storage in a secure microcontroller. The CPU is a big threat in embedded and SoC devices. Not only it leaks a lot of side-channel information during the code execution [12], but it could also pass decrypted data to address bus or leak it through conditional actions.

Although it was performed on a chip fabricated with 0.35µm process it shows the way of defeating the data bus and address bus encryption. Because the attack allows full S-Box extraction there is no difference whether the encryption was cryptographically strong or not. The ability to run an arbitrary code on the CPU would in the end allow full control with the whole memory extraction. For 16-bit and 32-bit CPUs the attack will not only take longer to perform because of the larger number of possible combinations. It would also require full control of the data bus which is not

possible with low-cost manual probing stations like the one used for these experiments.

More robust testing and evaluation must be performed on semiconductor devices going into sensitive and secure applications with high risk factors like in banking, car, aviation and medical industries as well as critical infrastructure. Not only a strong encryption must be used, but also integrity check of the code being executed.

#### REFERENCES

- [1] O. Kömmerling, M.G. Kuhn: Design principles for tamper-resistant smartcard processors. USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA, May 1999
- [2] Christopher Tarnovsky: Security Failures In Secure Devices. Black Hat DC, February 21, 2008
- [3] Travis Goodspeed, Aurelien Francillon: Half-Blind Attacks: Mask ROM Bootloaders are Dangerous. In: Proceedings of the 3rd USENIX conference on Offensive technologies. USENIX Association, 2009
- [4] Advanced Encryption Standard. Federal Information Processing Standards Publication 197, November 2001
- [5] Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. Information Processing Standards Publication 800-67, January 2012
- [6] William D. Brown, Joe E. Brewer: Nonvolatile Semiconductor Memory Technology: A Comprehensive Guide to Understanding and Using NVSM Devices, IEEE Press, 1997.
- [7] Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic, Digital Integrated Circuits, Second Edition, Prentice-Hall, 2002
- [8] Sergei Skorobogatov, Semi-invasive attacks – A new approach to hardware security analysis, Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, April 2005.
- [9] Friedrich Beck, Integrated Circuits Failure Analysis: A Guide to Preparation Techniques, John Wiley & Sons, 1997
- [10] Microchip PIC24HJ32GP302/304, PIC24HJ64GPX02/X04 and PIC24HJ128GPX02/X04 16-bit Microcontrollers. Datasheet DS70293G, 2007-2012 Microchip Technology Inc.
- [11] NEC User's Manual 78K/0 Series Instructions. October 2001
- [12] P. Kocher, J. Jaffe, B. Jun: Differential Power Analysis. CRYPTO'99, LNCS, Vol. 1666, Springer-Verlag, 1999, pp 388–397M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.