Reverse-Engineering a Cryptographic RFID Tag

Karsten Nohl and David Evans
University of Virginia
Department of Computer Science
{nohl,evans}@cs.virginia.edu

Starbug and Henryk Plötz
Chaos Computer Club
Berlin
starbug@ccc.de, henryk@ploetzli.ch

Abstract

The security of embedded devices often relies on the secrecy of proprietary cryptographic algorithms. These algorithms and their weaknesses are frequently disclosed through reverse-engineering software, but it is commonly thought to be too expensive to reconstruct designs from a hardware implementation alone. This paper challenges that belief by presenting an approach to reverse-engineering a cipher from a silicon implementation. Using this mostly automated approach, we reveal a cipher from an RFID tag that is not known to have a software or micro-code implementation. We reconstruct the cipher from the widely used Mifare Classic RFID tag by using a combination of image analysis of circuits and protocol analysis. Our analysis reveals that the security of the tag is even below the level that its 48-bit key length suggests due to a number of design flaws. Weak random numbers and a weakness in the authentication protocol allow for pre-computed rainbow tables to be used to find any key in a matter of seconds. Our approach of deducing functionality from circuit images is mostly automated, hence it is also feasible for large chips. The assumption that algorithms can be kept secret should therefore to be avoided for any type of silicon chip.

Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.
([A cipher] must not depend on secrecy, and it must not matter if it falls into enemy hands.)
August Kerckhoffs, *La Cryptographie Militaire*, January 1883 [13]

1 Introduction

It has long been recognized that security-through-obscurity does not work. However, vendors continue to believe that if an encryption algorithm is released only as a hardware implementation, then reverse-engineering the cipher from hardware alone is beyond the capabilities of likely adversaries with limited funding and time. The design of the cipher analyzed in this paper, for example, had not been disclosed for 14 years despite more than a billion shipped units. We demonstrate that the cost of reverse engineering a cipher from a silicon implementation is far lower than previously thought.

In some cases, details of an unknown cryptographic cipher may be found by analyzing the inputs and outputs of a black-box implementation. Notable examples include Bletchley Park's breaking the Lorenz cipher during World War II without ever acquiring a cipher machine [5] and the disclosure of the DST cipher used in cryptographic Radio Frequency Identification (RFID) tokens from Texas Instruments [4]. In both cases, researchers started with a rough understanding of the cipher's structure and were able to fill in the missing details through cryptanalysis of the cipher output for known keys and inputs. This black-box approach requires some prior understanding of the structure of a cipher and is only applicable to ciphers with statistical weaknesses. The output of a sound cipher should not be statistically biased and therefore should not leak information about its structure.

Other ciphers have been disclosed through disassembly of their software implementation. Such implementations can either be found in computer software or as microcode on an embedded micro-controller. Ciphers found through software disassembly include the A5/1 and A5/2 algorithms that secure GSM cell phone communication [1] and the Hitag2 and Keeloq algorithms used in car remote controls [3]. The cryptography on the RFID tags we analyzed is not known to be available in software or in a micro-code implementation; tags and reader chips implement the cipher entirely in hardware.

In this paper, we focus on revealing proprietary cryptography from its silicon implementation alone. Reverse-engineering silicon is possible even when very little is known about a cipher and no software implementation

exists. The idea of reverse-engineering hardware is not new. Hardware analysis is frequently applied in industry, government, and the military for spying, security assessments, and protection of intellectual property. Such reverse-engineering, however, is usually considered prohibitively expensive for typical attackers, because of the high prices charged by professionals offering this service. The key contribution of this work is demonstrating that reverse-engineering silicon is cheap and that it can be mostly automated. This is the first published work to describe the details of reverse-engineering a cryptographic function from its silicon implementation. We describe a mostly automated process that can be used to cheaply determine the functionality of previously unknown cipher implementations.

We demonstrate the feasibility of our approach by revealing the cipher implemented on the NXP Mifare Classic RFID tags, the world's most widely used cryptographic RFID tag [16]. Section 2 describes our reverse-engineering method and presents the cipher. Section 3 discusses several weaknesses in the cipher beyond its short key size. Weak random numbers combined with a protocol flaw allow for rainbow tables to be computed that reduce the attack time from weeks to minutes. Section 4 discusses some potential improvements and defenses. While we identify fixes that would increase the security of the Mifare cipher significantly, we conclude that good security may be hard to achieve within the desired resource constraints.

2 Mifare Crypto-1 Cipher

We analyzed the Mifare Classic RFID tag by NXP (formerly Philips). This tag has been on the market for over a decade with over a billion units sold. The Mifare Classic card is frequently found in access control systems and tickets for public transport. Large deployments include the Oyster card in London, and the SmartRider card in Australia. Before this work, the Netherlands were planning to deploy Mifare tags in OV-chipkaart, a nationwide ticketing system, but the system will likely be re-engineered after first news about a potential disclosure of the card's details surfaced [17]. The Mifare Classic chip currently sells for 0.5 Euro in small quantities, while tags with larger keys and established ciphers such as 3-DES are at least twice as expensive.

The cryptography found in the Mifare cards is a stream cipher with 48-bit symmetric keys. This key length has been considered insecure for some time (for example, the Electronic Frontier Foundation's DES cracking machine demonstrated back in 1998 that a moderately-funded attacker could brute force 56-bit DES [6]) and the practical security that Mifare cards have experienced in the past relies primarily on the belief that its cipher was secret. We find that the security of the Mifare Classic is even weaker than the short key length suggests due to flaws in its random number generation and the initialization protocol discussed in Section 3.

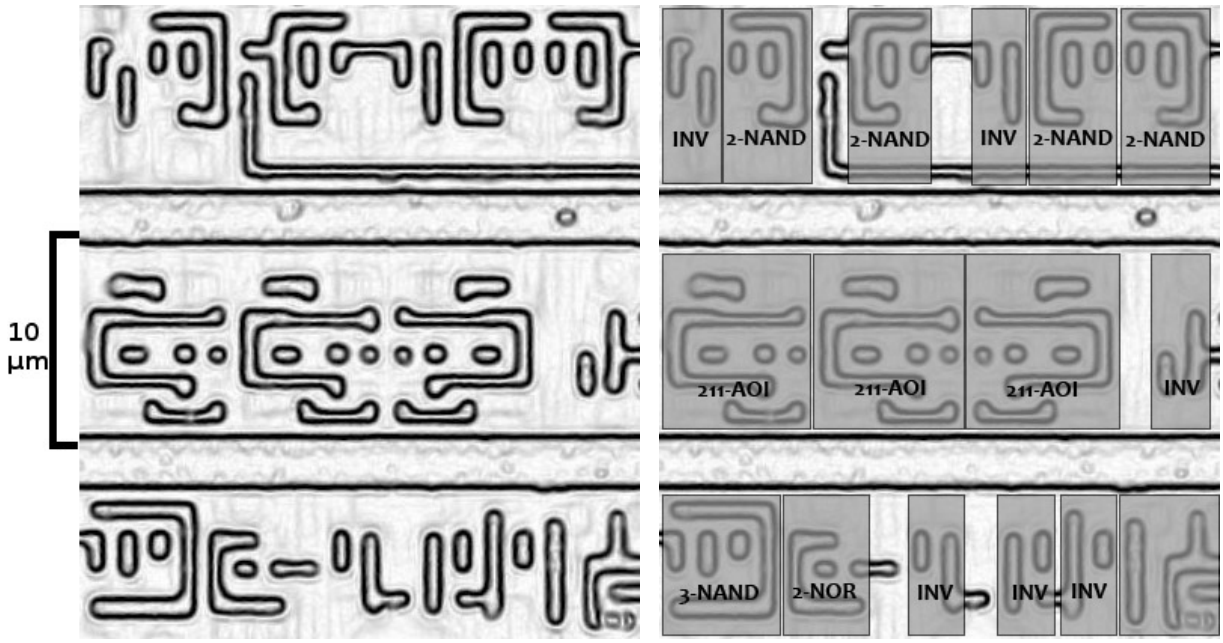
The data on the Mifare cards is divided into sectors, each of which holds two different keys that may have different access rights (e.g., read/write or read-only). This division allows for different applications to each store encrypted data on a tag—an option rarely used in practice. All secrets are set to default values at manufacturing time but changed before issuing the tags to users. Different tags in a system may share the same read key or have different keys. Sharing read keys minimizes the overhead of key-distribution to offline readers. We find, however, that the protocol level measures meant to prevent different users from impersonating each other are insufficient. Unique read and write keys should, therefore, be used for each tag and offline readers should be avoided as much as possible.

2.1 Hardware Analysis

The chip on the Mifare Classic tag is very small with a total area of roughly one square millimeter. About a quarter of the area is used for 1K of flash memory (a 4K version is also available); another quarter is occupied by the radio front-end and outside connectivity, leaving about half the chip area for digital logic including cryptography.

The cryptography functions make up about 400 2-NAND gate equivalents (GE), which is very small even compared to highly optimized implementations of standard cryptography. For example, the smallest known implementation of the AES block cipher (which was specifically designed for RFID tags) requires 3400 GEs [7]. The cryptography on the Mifare tags is also very fast and outputs 1 bit of key stream in every clock cycle. The AES circuit, by comparison, takes 1000 clock cycles for one 128-bit AES operation (10 milliseconds on a tag running at 106 kHz).

To reverse engineer the cryptography, we first had to get access to sample chips, which are usually embedded in credit card size plastic cards. We used acetone to dissolve the plastic card, leaving only the blank chips. Acetone is easier and safer to handle than alternatives such as fuming nitric acid, but still dissolves plastic cards in



(a) (b)

Figure 1: (a) Source image of layer 2 after edge detection; (b) after automated template detection.

about half an hour. Once we had isolated the silicon chips, we removed each successive layer through mechanical polishing, which we found easier to control than chemical etching. Simple polishing emulsion or sandpaper with very fine grading of $0.04\mu m$ suffices to take off micrometer-thick layers within minutes.

Although the polishing is mostly straightforward, the one obstacle to overcome is the chip tilting. Since the chip layers are very close together, even the smallest tilt leads to cuts through several layers. We addressed this problem in two ways. First, we embedded the millimeter-size chip in a block of plastic so it was easier to handle. Second, we accpeted that we could not completely avoid tilt using our simple equipment and adapted our image stitching tools to patch together chip layers from several sets of pictures, each imaging parts of several layers.

The chip contains a total of six layers, the lowest of which holds the transistors. We took pictures using a standard optical microscope at a magnification of 500x. From multiple sets of these images we were able to automatically generate images of each layer using techniques for image tiling that we borrowed from panorama photography. We achieved the best results using the open source tool `hugin` (http://hugin.sourceforge.net/) by setting the maximum variance in viewer angle to a very small value (e.g., 0.1°) and manually setting a few control points on each image.

The transistors are grouped in gates that each perform a logic function such as AND, XOR, or flip-flop as illustrated in Figure 1. Across the chip there are several thousand such logic gates, but only about 70 different types of gates. As a first step toward reconstructing the circuit, we built a library of these gates. We implemented template matching that given one instance of a logic gate finds all the other instances of the same gate across the chip. Our tools take as input an image of layer 2, which represents the logic level, and the position of instances of different logic gates in the image. The tools then use template matching to find all other instances of the gate across the image, including rotated and mirrored variants. Since larger gates sometimes contain smaller gates as building blocks, the matching is done in order of decreasing gate sizes.

Our template matching is based on *normalized cross-correlation* which is a well-known similarity test [14] and implemented using the MATLAB image processing library. Computing this metric is computationally more complex than standard cross-correlation, but the total running time of our template matching is still under ten minutes for the whole chip. Normalized cross-correlation is insensitive to the varying brightness across our different images and the template matching is able to find matches with high accuracy despite varying coloration and distortion of the structures that were caused by the polishing.

We then manually annotated each type of gate with its respective functionality. This step could be automated as well through converting the silicon-level depiction of each gate into a format suitable for a circuit simulation program. We decided against this approach because the overhead seemed excessive. For larger libraries that perhaps intentionally vary the library cells in an attempt to impede reverse-engineering, however, automation is certainly possible and has already been demonstrated in other projects [2].

Our template matching provides a map of the different logic gates across the chip. While it would certainly have been possible to reverse-engineer the whole RFID tag, we focused our attention on finding and reconstructing the cryptographic components. We knew that the stream cipher would have to include at least a 48-bit register and a number of XOR gates. We found these components in one of the corners of the chip along with a circuit that appeared to be a random number generator as it has an output, but no input.

Focusing our efforts on only these two parts of the chip, we reconstructed the connections between all the logic gates. This step involved considerable manual effort and was fairly error-prone. All the errors we made were found through a combination of redundant checking and statistical tests for some properties that we expected the cipher to have such as an even output distribution of blocks in the filter function. We have since implemented scripts to automate the detection of wires, which can speed the process and improve its accuracy. Using our manually found connections as ground truth we find that our automated scripts detect the metal connection and intra-layer vias correctly with reasonably high probability. In our current tests, our scripts detect over 95% of the metal connections correctly and the few errors they make were easily spotted manually by overlaying the source image and the detection result. These results are, however, preliminary, as many factors are not yet accounted for. To assess the potential for automation more thoroughly, we plan to test our tools on different chips, using different imaging systems, and having different users check the results.

In the process of reconstructing the circuit, we did not encounter any added obscurity or tamper-proofing. Because the cryptographic components are highly structured, they were particularly easy to reconstruct. Furthermore, we could test the validity of different building blocks by checking certain statistical properties. For example, the different parts of the filter function each have an even output distribution so that the output bits are not directly disclosing information about single state bits.

The map of logic gates and the connections between them provides us with almost enough information to discover the cryptographic algorithm. Because we did not reverse-engineer the control logic, we do not know the exact timing and inputs to the cipher. Instead of reconstructing more circuitry, we derived these missing pieces of information from protocol layer communication between the Mifare card and reader.

2.2 Protocol Analysis

From the discovered hardware circuit, we could not derive which inputs are shifted into the cipher in what order, partly because we did not reverse the control logic, but also because even with complete knowledge of the hardware we would not yet have known what data different memory cells contain. To add the missing details to the cipher under consideration, and to verify the results of the hardware analysis, we examined communication between the Mifare tags and a Mifare reader chip.

An NXP reader chip is included on the OpenPCD open source RFID reader, whose flexibility proved to be crucial for the success of our project. The OpenPCD includes an ARM micro-controller that controls the communication between the NXP chip and the Mifare card. This setup allows us to record the communication and provides full control over the timing of the protocol. Through timing control we can amplify some of the vulnerabilities we discovered as discussed in Section 3.

No details of the cipher have been published by the manufacturer or had otherwise been leaked to the public prior to this work. We guessed that the secret key and the tag ID were shifted into the shift register sequentially rather than being combined in a more complicated way. To test this hypothesis, we checked whether a reader could successfully authenticate against a tag using an altered key and an altered ID. Starting with single bit changes in ID and key and progressively extending our search to larger variations, we found a number of such combinations that indeed successfully authenticated the reader to the tag. From the pattern of these combinations we could derive not just the order of inputs, but also the structure of the linear feedback shift register, which we had independently found on the circuit level. Combining these insights into the authentication protocol with the results of our hardware analysis gave us the whole Crypto-1 stream cipher, shown in Figure 2.
The cipher is a single 48-bit linear feedback shift register (LFSR). From a fixed set of 20 state bits, the one bit of key stream is computed in every clock cycle. The shift register has 18 taps (shown as four downward arrows in

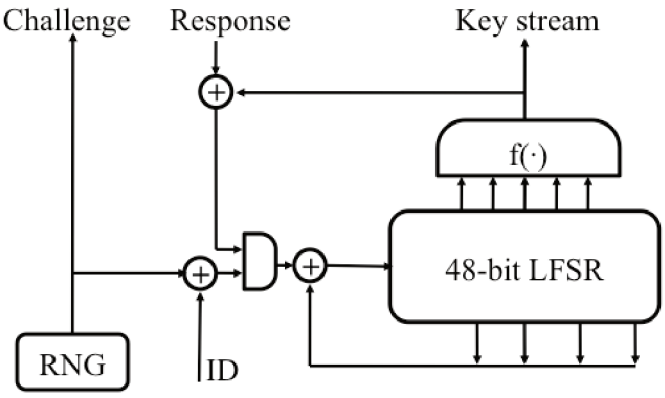


Figure 2: Crypto-1 stream cipher and initialization.

the figure) that are linearly combined to fill the first register bit on each shift. The update function does not contain any non-linearity, which by today's understanding of cipher design can be considered a serious weakness. The generating polynomial of the register is (with x^i being the ith bit of the shift register):

$$\begin{aligned} & x^{48}+x^{43}+x^{39}+x^{38}+x^{36}+x^{34}+x^{33}+x^{31}+x^{29} \\ + \quad & x^{24}+x^{23}+x^{21}+x^{19}+x^{13}+x^{9}+x^{7}+x^{6}+x^{5}+1. \end{aligned}$$

The polynomial is *primitive* in the sense that it is irreducible and generates all $\left(2^{48}-1\right)$ possible outputs in succession. To confirm this, we converted the Fibonacci LFSR into a Galois LFSR for which we can compute any number of steps in a few Galois field multiplications. We then found that the cipher state repeats after $\left(2^{48}-1\right)$ steps, but not after any of the possible factors for this number. The LSFR is hence of maximum-length.

The protocol between the Mifare chip and reader loosely follows the ISO 9798-2 specification, which describes an abstract challenge-response protocol for mutual authentication. The authentication protocol takes a shared secret key and a unique tag ID as its inputs. At the end of the authentication, the parties have established a session key for the stream cipher and both parties are convinced that the other party knows the secret key.

3 Cipher Vulnerabilities

The 48-bit key used in Mifare cards makes brute-force key searches feasible. Cheaper than brute-force attacks, however, are possible because of the cipher's weak cryptographic structure. While the vulnerability to brute-force attacks already makes the cipher weak, the cheaper attacks are relevant for many Mifare deployments such as fare collection where the value of breaking a particular key is relatively low. Weaknesses of the random number generator and the cryptographic protocol allow an attacker to pre-compute a codebook and perform key-lookups quickly and cheaply using rainbow tables.

3.1 Brute-Force Attack

In a brute-force attack an attacker records two challenge-response exchanges between the legitimate reader and a card and then tries all possible keys for whether they produce the same result.

To estimate the expected time for a brute-force attack, we implemented the cipher on FPGA devices by Pico Computing. Due to the simplicity of the cipher, 6 fully-pipelined instances can be squeezed into a single Xilinx Virtex-5 LX50 FPGA. Running the implementation on an array of 64 such FPGAs to try all 2^{48} keys takes under 50 minutes.

3.2 Random Number Generation

The random number generator (RNG) used on the Mifare Classic tags is highly insecure for cryptographic applications and further decreases the attack complexity by allowing an attacker to pre-compute a codebook.
The random numbers on Mifare Classic tags are generated using a linear feedback shift register with constant initial condition. Each random value, therefore, only depends on the number of clock cycles elapsed between the time the tag is powered up (and the register starts shifting) and the time the random number is extracted. The numbers are generated using a maximum length 16-bit LFSR of the form:

$$x^{16} + x^{14} + x^{13} + x^{11} + 1.$$

The register is clocked at 106 kHz and wraps around every 0.6 seconds after generating all 65,535 possible output values. Aside from the highly insufficient length of the random numbers, an attacker that controls the timing of the protocol controls the generated number. The weakness of the RNG is amplified by the fact that the generating LFSR is reset to a known state every time the tag starts operating. This reset is completely unnecessary, involves hardware overhead, and destroys the randomness that previous transactions and unpredictable noise left in the register.

We were able to control the number the Mifare random number circuit generated using the OpenPCD reader and custom-built firmware. In particular, we were able to generate the same "random" nonce in each query, thereby completely eliminating the tag randomness from the authentication process. Moreover, we found the same weakness in the 32-bit random numbers generated by the reader chip, which suggests that a similar hardware implementation is used in the chip and reader. Here, too, we were able to repeatedly generate the same number. While in our experiments this meant controlling the timing of the reader chip, a skilled attacker will likely be able to exploit this vulnerability even in realistic scenarios where no such control over the reader is given. The attacker can predict forthcoming numbers from the numbers already seen and precisely chose the time to start interacting with the reader in order to receive a certain challenge. The lack of true randomness on both reader and tag enable an attacker to eliminate any form of randomness from the authentication protocol. Depending on the number of precomputed codebooks, this process might take several hours and the attack might not be feasible against all reader chips.

3.3 Pre-Computing Keys

Several weaknesses of the Mifare card design add up to what amounts to a full codebook pre-computation. First, the key space is small enough for all possible keys to be included. Second, the random numbers are controllable. In addition, the secret key and the tag ID are combined in such a way that for each session key there exists exactly one key for each ID that would result in that session key. The key and the ID are shifted into the register sequentially, but no non-linearity is mixed in during this process. As explained in Section 2.2, for every delta of ID bits, there exists a delta of key bits that corrects for the difference and results in the same session key. Therefore, given a key that for some ID results in a session key, there exists a key for any ID that would result in the same session key. This bijective mapping allows for a codebook that was pre-computed for only a single ID to be used to find keys for all other IDs as well.

A codebook for all keys would occupy 1500 Terabytes, but can be stored more economically in rainbow tables. Rainbow tables store just enough information from a key space for finding any key with high probability, but require much less space than a table for all keys [9, 15]. Each "rainbow" in these tables is the repeated application of slight variants of a cryptographic operation. In our case, we start with a random key and generate the output of the authentication protocol for this key, then use this output as the next key for the authentication, generate its output, use that as the next key, and so on. We then only store the first and last value of each rainbow, but compute enough rainbows so that almost all keys appear in one of them. To find a key from such a rainbow table, a new rainbow is computed starting at a recorded output from the authentication protocol. If any one of the generated values in this series is also found in the stored end values of the rainbows, then the key used in the authentication protocol can be found from the corresponding start values of that matching rainbow. The time needed to find a key grows as the size of the tables shrinks.

Determining any card's secret key will be significantly cheaper than trying out all possible keys even for rainbow tables that only occupy a few Terabytes and can be almost as cheap as a database lookup. The fact that an attacker can use a pre-computed codebook to reveal the keys from many cards dramatically changes the economics of an attack in favor of the attacker. This means that even attacks on low-value cards like bus tickets might be profitable.

3.4 Threat Summary

To summarize the threat to systems that rely on Mifare encryption for security, we illustrate a possible attack. An attacker would first scan the ID from a valid card. This number is unprotected and always sent in the clear. Next, the attacker would pretend to be that card to a legitimate reader, record the reader message of the challenge-response protocol with controlled random nonces, and abort the transaction. Given only two of these messages, the key of the card can be found in the pre-computed rainbow tables in a matter of minutes and then used to read the data from the card. This gives the attacker all the information needed to clone the card.

4 Discussion

The illustrated attack is yet another example of security-by-obscurity failing. Weaknesses in the exposed cipher reveal the pitfalls of proprietary cipher design without peer-review. A few changes in the design would have made some of the discussed attacks infeasible and could have increased the key size within the same hardware constraints to make brute-force attacks less likely. Much better security, however, can only be achieved through better, more thoroughly analyzed ciphers.

4.1 Potential Fixes

The system is vulnerable against codebook attacks because of its weak random numbers and the linear combination of key and ID. Both can be fixed without adding extra hardware or slowing down the operation.

Better, yet still not cryptographically sound, random numbers can be generated by exploiting the fact that memory cells are initially in an undetermined state [10]. The same behavior can be caused in flip-flops like those that make up the state register of the stream cipher simply by not resetting the flip-flops at initialization time. The cipher state would start in a random state and then evolve using the cipher's feedback loop until a random number is needed. At this point, the register contains a mostly unpredictable number of the size of the state register.

Because this design generates random numbers within the same registers that are used for the cipher states, it eliminates the need for a separate additional PRNG circuit. The saved area could then be spent on increasing the size of the cipher state. In the area of the 48-bit Crypto-1 and its 16-bit RNG, a 64-bit stream cipher that also produces significantly better pseudo-random number could hence be implemented. This increases the size and quality of the random numbers and at the same time increases the key size beyond the point where brute-force attacks can be done cheaply.

To further improve the resistance against codebook attacks, the non-linear feedback should be combined with either key or ID when shifted into the register to break the bijective mapping between different key-ID pairs. This measure does not increase implementation costs, since we only integrate the output of the filter function which is already computed.

To improve the resistance of the cipher against statistical attacks, the update function must be made non-linear, either by feeding some intermediate result of the filter function into the linear register or by using a non-linear feedback shift register instead.

None of the possible fixes will make the cipher appropriate for high security applications, but they improve the resistance against the most concerning attacks and can be done without any additional implementation cost.

4.2 Possible Defenses

Possible ways to protect against the described attacks include using standard, peer-reviewed, established cryptography such as the 3-DES block cipher that is already found on some of the more expensive cards including some of the Mifare line of products. A cheaper alternative that can be implemented in about twice the size of Crypto-1 is the Tiny Encryption Algorithm (TEA) [12, 18]. This established low-cost block cipher has publicly been scrutinized for several years and is so far only known to be vulnerable to some expensive attacks [11]. While TEA is far more secure than Crypto-1, it is also much slower. A Mifare authentication takes little more than one millisecond, while a minimum-size implementation of TEA would take about ten times as long. This would still be fast enough for most applications where Mifare cards are currently used.

Other known ways to protect against card cloning include fraud detection algorithms that are widely used in monitoring credit card transactions. These algorithms detect unusual behavior and can prevent fraudulent transactions, but require storing and analyzing transaction data, which runs contrary to the desire for privacy in RFID applications. Fraud protection systems also require all readers to be constantly connected to a central server, which is not the case in some of the current and planned deployments of RFID tags where offline readers are used.

Tamper-proofing can be used to protect secret keys from attackers, but provides little help against hardware reverse-engineering because the structure of the circuits will always be preserved. The implementation, however, could be obfuscated to increase the complexity of the circuit detection. While we believe that obfuscations will not make our approach infeasible, we do not yet know to what degree obfuscations could increase the effort and cost required to reverse-engineer a circuit.

All low-cost cryptographic RFID tags are currently ill-suited for high security applications because they lack tamper-proofing and are vulnerable to relay attacks. In

these attacks, the communication between a legitimate reader and a valid card is relayed through a tunnel thereby giving the reader the false impression that the card is in its vicinity. No level of encryption can protect against relay attacks and new approaches such as distance bounding protocols are needed [8].

5 Conclusions

Reverse-engineering functionality from silicon implementations can be done cheaply, and can be automated to the point where even large chips are potential targets. This work demonstrates that the cost of finding the algorithm used in a hardware implementation is much lower than previously thought. Using template matching, algorithms can be recovered whose secrecy has so far provided a base for security claims. The security of embedded cryptography, therefore, must not rely on obscurity. Any algorithm given to users in form of hardware can be disclosed even when no software implementation exists and black-box analysis is infeasible. Once the details of a cryptographic cipher become public, its security must rely entirely on good cryptographic design and sufficiently long secret keys.

The cryptographic strength of any security system depends on its weakest link. Besides the cryptographic structure of the cipher, weaknesses can arise from protocol flaws, weak random numbers, or side channels. When random numbers are weak and the user identification is not properly mixed into the secret state, codebooks can be pre-computed that lead to attacks that are much more efficient than brute force. In the case of the Mifare Classic cards, the average attack cost shrinks from several hours to minutes. Their cryptographic protection is hence insufficient even for low-valued transactions.

The question remains open as to whether security can be achieved within the size of the Mifare Crypto-1 cipher. The area of less than 500 gates may be too small to even hold a sufficiently large state, regardless of the circuits needed for the complex operations required for strong ciphers.

Acknowledgments. This work was partially funded by the National Science Foundation through the CyberTrust program, award CNS 0627527. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect those of the National Science Foundation.

References

[1] Ross Anderson. *A5*. Post to *sci.crypt*, 17 June 1994.

[2] L. R. Avery, J. S. Crabbe, S. Al Sofi, H. Ahmed, J. R. A. Cleaver, D. J. Weaver. Reverse Engineering Complex Application-Specific Integrated Circuits (ASICs). In *Diminishing Manufacturing Sources and Material Shortages Conference*, 2002.

[3] Andrey Bogdanov. Attacks on the KeeLoq Block Cipher and Authentication Systems. In *RFIDSec*, 2007.

[4] Stephen C. Bono, Matthew Green, Adam Stubblefield, Ari Juels, Aviel D. Rubin, and Michael Szydlo. Security Analysis of a Cryptographically-Enabled RFID Device. In *USENIX Security Symposium*, 2005.

[5] Harvey G. Cragon. *From Fish to Colossus: How the German Lorenz Cipher was Broken at Bletchley Park*. Cragon Books, 2003.

[6] Electronic Frontier Foundation. Cracking DES. In *Secrets of Encryption Research, Wiretap Politics & Chip Design*, O'Reilly & Associates Inc., 1998.

[7] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In *Workshop on Cryptographic Hardware and Embedded Systems*, 2004.

[8] Gerhard P. Hancke and Markus G. Kuhn. An RFID Distance Bounding Protocol. In *SecureComm*, 2005.

[9] Martin E. Hellman. A Cryptanalytic Time-Memory Trade-Off. In *IEEE Transactions on Information Theory*, 1980.

[10] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Initial SRAM state as a Fingerprint and Source of True Random Numbers for RFID Tags. In *RFIDSec*, 2007.

[11] Seokhie Hong, Deukjo Hong, Youngdai Ko, Donghoon Chang, Wonil Lee, and Sangjin Lee. Differential Cryptanalysis of TEA and XTEA. In *International Conference on Information Security and Cryptology*, 2003.

[12] Pasin Israsena. Securing Ubiquitous and Low-cost RFID Using Tiny Encryption Algorithm. In *International Symposium on Wireless Pervasive Computing*, 2006.

[13] Auguste Kerckhoffs. La Cryptographie Militaire. In *Journal des Sciences Militaires*,

1883.

[14] J. P. Lewis. Fast Normalized Cross-Correlation. In *Vison Interface*, 1995.

[15] Philippe Oechslin. Making a Faster Cryptanalytic Time-Memory Trade-Off. In *Crypto*, 2003.

[16] NXP Semiconductors. *Philips Semiconductors leads contactless smart card market*, 2006.

[17] Andrew Tanenbaum. *News Summary of Broken Dutch Public Transit Card*. `www.cs.vu.nl/~ast/ov-chip-card`

[18] David J. Wheeler and Roger M. Needham. TEA, a Tiny Encryption Algorithm. In *Fast Software Encryption*, 1994.