Appears in *Proceedings: Financial Cryptography and Data Security – FC 2008, Cozumel, Mexico.*
Lecture Notes on Computer Science (LNCS), Springer Verlag.

Breaking Legacy Banking Standards with Special-Purpose Hardware

Tim Güneysu, Christof Paar

Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany

Abstract. In the field of eCommerce, online-banking is one of the major application requiring the usage of modern cryptography to protect the confidentiality and integrity of financial transactions between users and the banking system. In banking applications of some countries, the authorization of user transactions is performed with support of cryptographic One-Time-Password (OTP) tokens implementing ANSI X9.9-based challenge-response protocols.
The legacy ANSI X9.9 standard is a DES-based authentication method on which we will demonstrate an attack based on a special-purpose hardware cluster. In this work we show how to break such an OTP-token with little effort in terms of costs and time. With an investment of about US $ 10,000 we are able to perform an attack which computes the key of a DES-based OTP token in less than a week having only three challenge-response pairs. Our attack can even be scaled linearly according to the budget of the attacker resulting in even faster breaking times. With this work, we want to point out once more that the immediate migration from legacy products using the DES algorithm is absolutely mandatory for security critical applications.

Keywords: ANSI X9.9, Banking, Cryptanalysis, Special-Purpose Hardware.

1 Introduction

With the rise of the Internet during the last decades, this new communication medium has become increasingly relevant for financial transactions with respect to eCommerce and particularly, online-banking. At the same time, the Internet opens up new potential ways for digital criminals to attack banking applications what increases the demand for effective cryptography. Beside efficient data encryption schemes to protect the business transaction from being eavesdropped or altered by unauthorized parties, individuals need to authenticate themselves, e.g., for logging into an online-banking system of a financial institute. For this reason, several fundamentally different entity authentication techniques are in use worldwide. For instance, some banks make use of a combination of personal and transaction identification numbers (PIN/TAN), others employ single-use passwords which are generated by cryptographic tokens. In this contribution, we will focus on the latter, token-based One-Time-Passwords[1] (OTP) used for authenticating users in financial applications. Note that in this work the tokens of interests are tokens which respond deterministically to a given ANSI X9.9 challenge without any further source of entropy like time or user events being involved.

In order to establish common methods for cryptography, worldwide standards for computer authentication have been developed since the 1980s. Financial security was historically one of the main motivation for such standards. Based on the Data Encryption

[1] Our abbreviation is not to be confused with *one time pad* or *one-time programmable*, which is also sometimes denoted by OTP.

Standard (DES) cipher [12, 1], which was the most common cipher used for many years, several standards for authentication have been created, e.g., FIPS PUB 113 [11], ANSI X9.9 [2] and ISO 8730 [8]. In 1998, the Electronic Frontier Foundation (EFF) presented a hardware cracker called *Deep Crack* capable to break DES within 56 hours [7] due to its greatest weakness — the small key space. Deep Crack consisted of $1,536$ custom designed ASIC chips and was built for about US$ 250,000. As a response to cracking machines like this, it was agreed that new standards need be developed to replace the DES cipher, resulting in cipher schemes like Triple-DES [13] and the Advanced Encryption Standard (AES) [14] as a quick fix to the weak single-DES.

In 2004, the International Organization for Standardization (ISO) has withdrawn ISO 8730 for DES-based data authentication and replaced it with ISO 16609 [9] based on the AES. Similarly in 2004, the National Institute of Standards and Technology (NIST) has declared DES to be outdated and should be only used as a component of TDES [15]. Since 2005, DES is not recommended for use in any security-relevant application. Despite of this statement two years ago, it is known that there are still legacy DES-based systems which are also used in critical applications such as online-banking systems. We are aware of several banks in Europe, North and Central America[2] which are using OTP-tokens for banking systems to authorize financial transactions using a DES-based crypto-token according to ANSI X9.9. Although these DES-tokens partially support TDES operation, they still allow single DES operation as well as ANSI X9.9 authentication. We believe that such single DES-based systems are still in use in some banking applications due to compatibility reasons with legacy systems. One example of such crypto tokens which are still issued by banks for use in online-banking are the tokens by ActivIdentity [3], formerly ActivCard [20].

To emphasize the security weaknesses of legacy DES-based systems and, to hasten the replacement of unsecure crypto modules, we present the first hardware-based attack on single-DES tokens implementing a challenge-response protocol based on the common ANSI X9.9 standard. We would like to mention that we were able to actually break commercial on-line banking tokens in our lab. Again, we prefer not to name the manufacturer.

This contribution is structured as follows: in the next section, we briefly introduce relevant previous work related to DES. Next, we give a short introduction to the functionality of ANSI X9.9-based crypto tokens. Section 4 is dedicated to the development of attack scenarios against real-world protocols which are based on crypto tokens. The implementation of those are discussed in Section 6. Due to the fact that we can break ANSI X9.9-based tokens in about a week, we conclude our work with a strong recommendation for immediate replacement of obsolete crypto modules in security-sensitive environments.

[2] The names of respective institutions are known to the authors but will not be mentioned here. In fact, even large banks with about 2.5 million internet banking customers do not seem to have completely abandoned the use of single DES in their systems.

2 Previous Work

Although the DES has been reaffirmed for use in (US government) security systems several times until 1998, the worries about the inherent threat of its short key space was already raised in 1977 [6] when it was first proposed. A first detailed hardware design description for a brute force attacker was presented by Michael Wiener at the rump session of CRYPTO'93, a printed version of which is available in [21]. It was estimated that the machine could be built for less than a million US dollars. The proposed machine consists of $57,000$ DES chips that could recover a key every three and half hours. The estimates were updated in 1998 due to the advances in hardware for a million dollar machine to 35 minutes for each key recovery [22]. Ian Goldberg and David Wagner estimated the cost for building a DES brute force attacker using FPGAs to US$ 45,000 for a key recovery within a year [7]. In 1997, a detailed cost estimate for three different approaches for DES key search: distributed computing, FPGAs and custom ASIC designs, was compiled by Blaze et al. [4]. The first actual DES attack (presumingly outside government agencies) took place in 1998, and was based on the above mentioned *Deep Crack* [7]. To our knowledge, the latest step in the history of DES brute-force attacks took place in 2006, when the Cost Optimal Parallel Code Breaker (COPACOBANA) was built for less than US$ 10,000 [10]. COPACOBANA is capable of breaking DES in less than one week on average. We would like to stress that software-only attacks against DES still take more than 1,000 PC-years (worst case).

Most of these attacks assume that at least one complete plaintext-ciphertext pair is given. We will see that crypto tokens for bank applications typically do not provide such input, so that a smarter attack must be chosen to tackle this kind of systems. There are some theoretical contributions by Coppersmith et al. [5] as well as by Preneel and van Oorschot [16] considering the theoretical security of DES-based authentication methods (DES-MAC). But to our best knowledge an attack on an actually ANSI X9.9-based crypto system has not been proposed (or demonstrated) yet.

3 Basics of Token Based Data Authentication

We will now describe a OTP token-based data protocol according to FIPS 113 or ANSI X9.9 which is used for authentication in some real-world online-banking systems. Please note that in this work we assume that OTP tokens have a fixed, securely integrated static key inside and do not use additional entropy sources like time or events for computing the passwords. Indeed, there are tokens available which generate new passwords after a dedicated time interval (e.g., products like the RSA SecurID solution [18]) but those will not be the focus of this work. These type of tokens require additional assumptions concerning the unknown plaintext and thus are harder to attack. Thus, our contribution assumes fixed-key OTP tokens which can be used in combination with a challenge-response protocol. In such protocols, a decimal-digit challenge is manually entered into the token via an integrated keypad. The token in turn computes the corresponding response according to

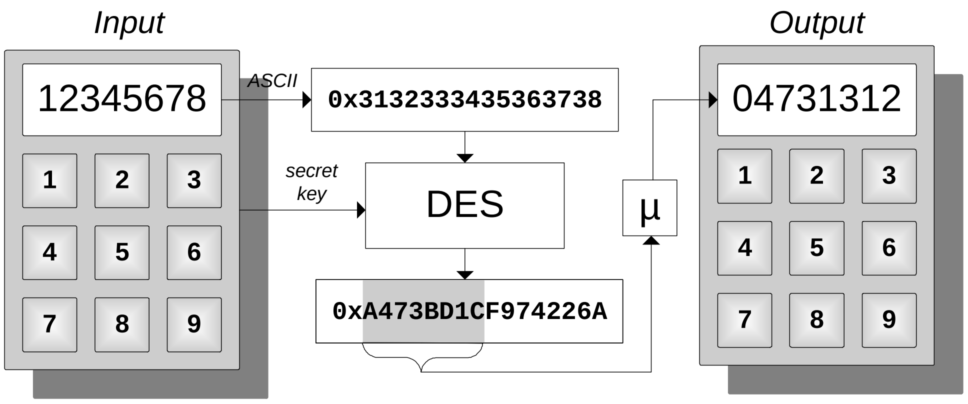

Fig. 1. Principle of the response generation on a ANSI X9.9-based crypto token.

the ANSI X9.9 standard. Tokens implementing this standardized authentication scheme (incorporating ANSI 3.92 DES encryption) have a often a fixed size LCD allowing for displaying 8 decimal digits for input and output.

After the user has typed in eight decimal digits as input (challenge), the value is converted to binary representation using standard ASCII code notation according to the ANSI X9.9 standard. For instance, the typed number “12345678” is converted into the 64-bit challenge value in hexadecimal representation

$$c = (\texttt{0x31}, \texttt{0x32}, \texttt{0x33}, \texttt{0x34}, \texttt{0x35}, \texttt{0x36}, \texttt{0x37}, \texttt{0x38}).$$

After recoding c is used as plaintext to the DES encryption function $r = e_k(c)$ with the static key k stored securely in the token. The output of the encryption function is the 64-bit ciphertext $r = (r_1, r_0)$ where each r_i denotes a 32 bit word to be transformed using a mapping μ to fit the 8-digit display of the token. The mapping μ takes the 8 hexadecimal digits of r_1 (32 bits) of the DES encryption as input, and converts each digit individually from hexadecimal (binary) notation to decimal representation. Let $H = \{\texttt{0}, \ldots, \texttt{9}, \texttt{A}, \ldots, \texttt{F}\}$ and $D = \{\texttt{0}, \ldots, \texttt{9}\}$ be the alphabets of hexadecimal and decimal digits, respectively. Then μ is defined as:

$$\mu : H \to D : \{\texttt{0}_H \mapsto \texttt{0}_D; \ldots; \texttt{9}_H \mapsto \texttt{9}_D; \texttt{A}_H \mapsto \texttt{0}_D; \ldots; \texttt{F}_H \mapsto \texttt{5}_D\}$$

Hence, the output after the mapping μ is an 8 decimal digit value which is displayed on the LCD of the token. Figure 1 shows how the response is generated on the token according to a given challenge. In several countries, this authentication method is used in banking applications whenever a customer needs to authenticate financial transactions. For this, each user of such an online-banking system possesses a personal token used to respond to challenges which are presented by the banking system on every security critical operation. In this context for example, a security critical operation can be the login to the banking system as well as the authorization of a money transfer. The central role in such a security-related application makes the secret token an interesting target for an attack.

4 Cryptanalysis of the ANSI X9.9-based Challenge-Response Authentication

With the knowledge of how an authenticator is computed in the challenge-response protocol, we will continue with identifying weaknesses to attack this authentication scheme. Firstly, ANSI X9.9 relies on the outdated DES algorithm for which we can build a low-cost special-purpose hardware machine to perform an exhaustive key search in under a week. Secondly, the output r of the DES encryption is only slightly modified. (Note that a more complex scrambling with additional dynamic input, like hash functions with salt, would make the attack considerably more complex.) The output r is only truncated to 32-bit and modified using the mapping μ to convert c_1 from hexadecimal to decimal notation. Due to the truncation to 32 bits, we need to acquire knowledge of at least two plaintext-ciphertext pairs when mounting an exhaustive key search returning only a single key candidate. The digit conversion in μ additionally reduces the information leaked by a single pair of plaintext-ciphertext which is addressed by Theorem 1.

Theorem 1. *Let $D = \{0, \dots, 9\}$ be the alphabet of decimal digits. With a single challenge-response pair (c, r) of an ANSI X9.9-based authentication scheme where $c, r \in D^8$, on average 26 bits of a DES key can be determined (24 bits in the worst case, 32 bits in the best case).*

Proof: Since only 32 bits of the output c for a given challenge c are exposed, this is a trivial upper bound for the information leakage for a single pair. Assuming the DES encryption function to be a pseudo-random function with appropriate statistical properties, the 32 most significant bits of c form 8 hexadecimal digits uniformly distributed over $H^8 = \{\mathtt{0}, \dots, \mathtt{9}, \mathtt{A}, \dots, \mathtt{8}\}^8$. The surjective mapping μ has the domain $F = \{\mathtt{0}, \dots, \mathtt{9}\}$ of which $T = \{\mathtt{0}, \dots, \mathtt{5}\}$ are double assigned. Hence, we know that $\Delta = F \backslash T = \{\mathtt{6}, \dots, \mathtt{9}\}$ are four fixed points which directly correspond to output digits of c yielding four bit of key information (I). The six remaining decimal digits $\Omega = F \cap T$ can have two potential origins allowing for a potential deviation of one bit (II). According to a uniform distribution of the 8 hexadecimal output digits, the probability that (I) is given for an arbitrary digit i of c is $Pr(i \in \Delta) = 1/4$. Thus, on average we can expect 2 out of 8 hexadecimal digits of c to be in Δ revealing four bits of the key whereas the remaining 6 digits introduce a possible variance of one unknown bit per digit. Averaged, this leads to knowledge of $R = 2 \cdot 4 + 6 \cdot 3 = 26$ bits of DES key material. Obviously, the best case with all 8 digits in Δ and worst case with no digits out of the set Δ provide 32 and 24 key bits, respectively.

According to Theorem 1, we can develop two distinguished attacks based on the knowledge of two and three known challenge-response pairs:

Corollary 1. *Given two known challenge-response pairs (c_i, r_i) for $i = \{0, 1\}$ of the ANSI X9.9 authentication scheme, an exhaustive key search using both pairs will reveal $2^4 = 16$ potential key candidates on average (256 candidates in the worst case, in the best case the actual key is returned).*

Proof: Assuming independence of two different encrypted blocks related to the same key in block ciphers, we can use accumulated results from Theorem 1 for key determination

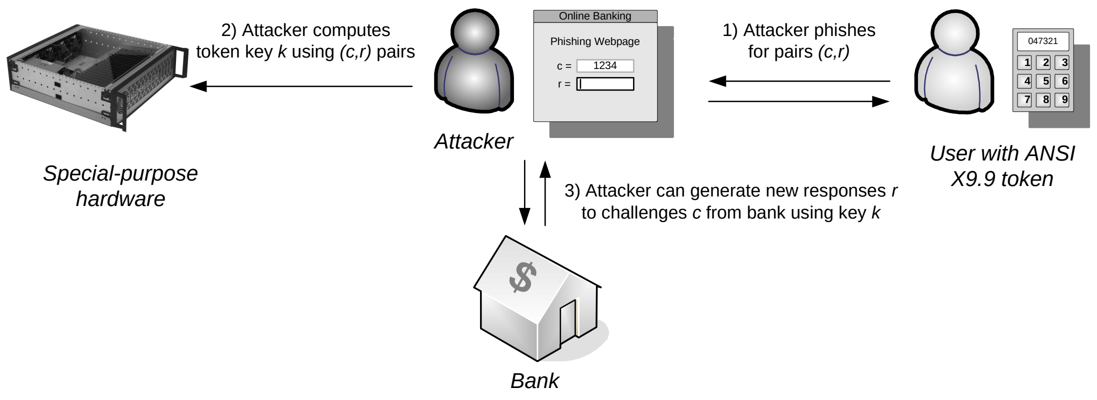
2) Attacker computes token key k using (c,r) pairs
Special-purpose hardware
Online Banking
Phishing Webpage
c = 1234
r =
Attacker
1) Attacker phishes for pairs (c,r)
047321
1 2 3
4 5 6
7 8 9
User with ANSI X9.9 token
3) Attacker can generate new responses r to challenges c from bank using key k
Bank

Fig. 2. Attack scenario for token-based banking applications using phishing techniques.

using multiple pairs (p_i, c_i). Hence, on average we can expect to determine 52 bits of the key where each c_i has 2 digits from the set Δ. Given a full DES key of 56 bit size results in 2^4 possible variations for key candidates. Having at least 4 digits from Δ for each c_i, this will lead to the best case resulting in a single key candidate. In the worst case and with no Δ digits in any c_i, we will end up with 48 bits of determined key material and $2^8 = 256$ possible remaining key candidates. As a consequence, the number of potential key candidates is directly dependent on how many digits of a c_i are fixed points and out of the set Δ.

Corollary 2. *Given three known challenge-response pairs of the ANSI X9.9 authentication scheme, an exhaustive key search based on this information will uniquely reveal the DES key.*

Proof: This case can be directly derived from Corollary 1. For this attack, $3 \cdot 24 = 72 > 56$ bits of key material can directly determined (even in the worst case) resulting in the correct key to be definitely identified.

5 Possible Attack Scenarios on Banking Systems

With these theoretical means at hand, we can begin to develop two attack variants for two and three plaintext-ciphertext pairs. Since we need only few pairs of information, an attack is feasible in a real-world scenario. For instance, if we consider a phishing attack on an online-banking system, we can easily imagine that two or three (faked) challenges are presented to the user, who is likely to respond with the appropriate values generated by his token. Alternatively spying techniques, e.g., based on malicious software like key-loggers or hidden cameras, can be used to observe the user while responding to a challenge. It is important to state that the freshness of these values do not play a role since we use the information only for computing the secret key and not for an unauthorized login attempt. Figure 2 shows a possible attack scenario on ANSI X9.9 tokens and associated banking applications based on phishing of challenge-response pairs (c, r). With at least two pairs of challenge-response data, we can perform an exhaustive key search on the DES key space implementing the specific features of ANSI X9.9 authentication. To cope with the DES key space of 2^{56} potential key candidates we will propose an implementation based on dedicated special-purpose hardware. In case that three pairs of challenge-responses pairs are given, we are definitely able to determine the key of the secret token using a single exhaustive key search. When only two pairs (c_i, r_i) are available to the attacker, then it is likely that several potential key candidates are returned from the key search (cf. Corollary 1). With 16 potential solutions on average, the attacker can attempt to guess the right solution by trial-and-error. Since most banking systems allow the user to enter up to three erroneous responds to a challenge in a row, two key candidates can be tried by the attacker at a time. Then, after a period of inactivity, the authorized user has probably logged into the banking application that resets the error counter and allows the attacker to start another trial session with further key candidates. On average, the attacker can expect to be successful after about four trial and error sessions, testing 8 out of the 16 keys from the candidate list. Hence, an attack on an ANSI X9.9-based token is very likely to be successful even with knowledge of only two given challenge-response pairs.

6 Implementing an Attack on ANSI X9.9-based Systems

In this section we will discuss the implementation of an hardware attack on the ANSI X9.9 scheme with given two or three challenge-response pairs (c_i, r_i). Our hardware architecture will be designed for use with two pairs since data paths and multiplexers in hardware are less complex with only two inputs. For each potential key of the DES key space, the corresponding authenticator is computed for the given challenges and compared against the known responses. Potential key candidates satisfying the comparison for both (c_i, r_i) are transferred to a host computer. If even a third challenge-response pair is available, this is compared on the host computer to reduce hardware complexity.
Next, we will introduce our special-hardware cluster platform used to perform an exhaustive key search out of 2^{56} possible key candidates in about a week.

6.1 FPGA-based Special-Purpose Hardware Cluster

Today low-cost Field Programmable Gate Arrays (FPGA) provide an interesting alternative to ASICs for the massive computational effort required for cryptanalytic applications and code breaking. Since FPGAs have evolved in gate complexity at reduced costs during the last years, their advantage of flexible configuration at runtime makes them competitive to ASIC devices which are designed for a single application only. Hence, a cluster system based on FPGAs can support a wide variety of applications providing a significant cost-performance advantage over PC-based machines.

6.2 Hardware Cluster Architecture

The COPACOBANA machine [10] is an existing FPGA cluster designed for parallel computations with only a minor demand on communication and volatile memory. It integrates

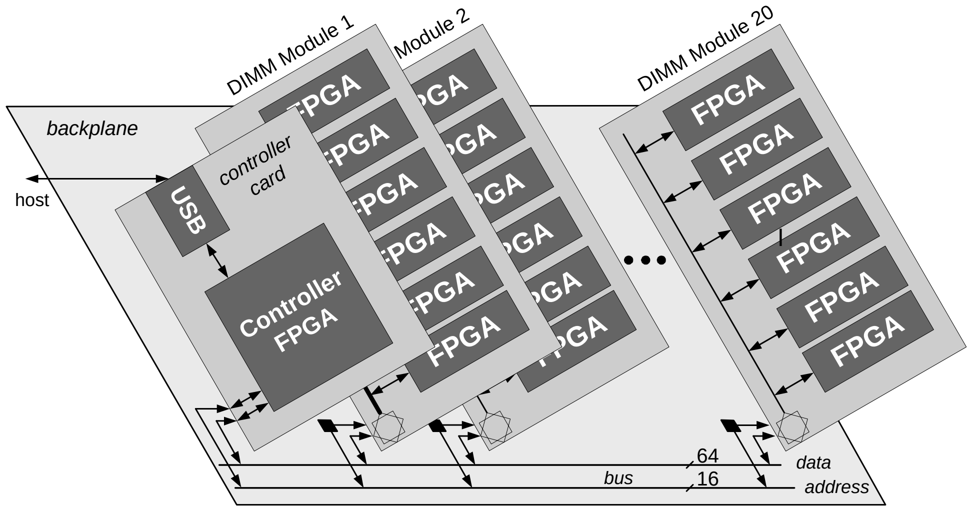

Fig. 3. The COPACOBANA cluster architecture provides 120 Spartan-3 FPGAs on a single backplane distributed over 20 DIMM-modules (each hosting 6 FPGA devices).

120 Xilinx Spartan-3 XC3S1000 FPGAs in a compact and modular design on a single backplane. Each FPGA consists of 1 million system gates providing 17280 equivalent logic cells which is sufficient for cost-optimized implementations of medium-scale circuits in reconfigurable hardware. The flexibility of our system platform has been achieved by 20 small and pluggable FPGA-modules (standard DIMM size) each populated with six FPGAs. The backplane hosts all FPGA-modules and a controller card which connects the COPACOBANA to a standard computer using a USB or Ethernet interface. All DIMM-modules are connected via the backplane by a 64-bit data bus and a 16-bit address bus. A detailed overview of the architecture is depicted in Figure 3. The COPACOBANA is an excellent platform for applications with very high computational requirements but low communication demands. A symmetric brute force attack is such a computational problem which can be hardly tackled with standard PCs. For example, an exhaustive key search on DES can be performed with the FPGAs clocked at 136 MHz, allowing for a total number of 544 million encryptions per second on *each* FPGA. Hence, the entire machine incorporating 120 FPGAs can compute about $2^{35.9}$ DES-encryptions per second resulting in an average key search duration of 6.4 days. A single standard PC performing 2 million DES encryptions per second, however, would take 585 years on average to complete this task [10]. Hence, a large number of PCs is required to achieve a equivalent computational power as provided by COPACOBANA. Due to the simple design of the COPACOBANA, its total material costs including manufacturing (but excluding initial design costs) are just about $ 10,000 which makes code breaking based on COPACOBANA affordable and realistic.

6.3 FPGA-based Attack Architecture

Originally, the DES was designed to be optimal for hardware-based implementations. Therefore, an FPGA implementation of the DES can be more than a 100 times faster than

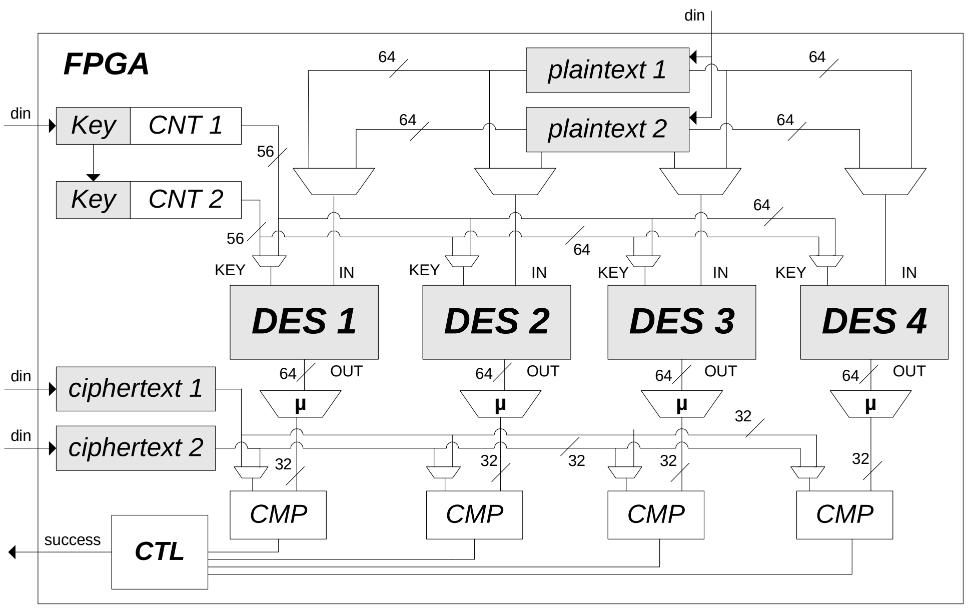

Fig. 4. Four ANSI X9.9 key search units based on fully pipelined DES cores in a Xilinx Spartan-3 FPGA.

an implementation on a conventional PC. This allows a hardware-based key search engine to be much faster and more cost efficient compared to an approach using a comparable software cluster.

The main goal of our hardware design is a key search of the token to be done in a highly parallelized fashion by partitioning the key space among the available FPGAs on the COPACOBANA. This requires hardly any interprocess communication, as each of the DES engines can search for the right key within its allocated key subspace.

Within the FPGAs, we used an optimized version of the highly pipelined DES implementation of the Université Catholique de Louvain's Crypto Group [17], which computes one encryption per clock per engine. Thus, it is possible to test one potential key per clock cycle. This scales linearly with the number of available parallel engines. For breaking ANSI X9.9-based authentication, we can fit four such DES engines inside a single FPGA, and therefore allow for sharing of control circuitry and the key space as shown in Figure 4. The FPGA architecture comprises two 64-bit *plaintext* registers for the challenges and two 32-bit *ciphertext* registers for storing the corresponding responses which have been acquired from the OTP-token. The key space to be searched is allocated to each chip as the most-significant 14-bits of the key which is stored in the *Key* register. The counter (*CNT 1*) is used to run through the least significant 40 bits of the key. The remaining two bits of the 56-bit key for each of the DES engines is hardwired and dedicated to each of them. Thus, for every such FPGA, a task is assigned to search through all the keys with the 16 most-significant bits fixed, in total 2^{40} different keys. The key space is partitioned by a connected host-PC so that each chip takes around 150 minutes (at 120 MHz) to test all ANSI X9.9 authenticators in its allocated key subspace. During a single check of an authenticator, the DES engines use the first challenge (*plaintext 1*) as a primary input to the encryption function. Then, the upper 32-bits of the generated ciphertext is mapped digit-per-digit by the function μ and compared to the value of the response stored in the register *ciphertext 1*. If any of the DES engines provides a positive match, the corresponding engine switches its input to the second challenge encrypting it with the same key. To match the pipelined design of the DES engine, we are using a shadow counter (*CNT 2*) implementing a delay tracking the key position at the beginning of the pipeline. In case that the derived authenticator from the second encryption compares successfully to the second response, the controller *CTL* reports the counter value to the host-PC as a potential key candidate. The host-PC keeps track of the key range that is assigned to each of the FPGAs and, hence, can match the right key from a given counter value. If no match is found until the counter overflows, the FPGA reports completion of the task and remains idle until a new key space is assigned. In case that a third challenge-response pair has been specified, the host-PC performs a verification operation of the reported key candidate in software. In case the verification was successful, the search is aborted and the key returned as result of the search.

7 Implementation Results

We have implemented the FPGA architecture shown in Figure 4 using the Xilinx ISE 9.1 development platform. After synthesis of the design incorporating four DES engines and the additional logic for the derivation of the ANSI X9.9 authenticator, the usage of 8,729 flip flops (FF) and 12,813 Look-Up-Tables (LUT) has been reported by the tools (56% FF and 83% LUT utilization of the Spartan-3 device, respectively). Furthermore, we included FPGA specific optimizations like pipelined comparators since n-bit comparators are likely to introduce a long signal propagation path reducing the maximum clock frequency significantly. By removing these potential bottlenecks, the design can be clocked at 120MHz after place-and-route resulting in a throughput of 480 million keys per FPGA per second. In total, a fully equipped COPACOBANA with 120 FPGAs can compute 57.6 billion ANSI X9.9 authenticators per second. Based on this, we can present time estimates for an attack provided that two challenge-response pairs are given. Recall that in this scenario we will be faced with several potential key candidates per run so that we have to search the entire key space of 2^{56} to build a list with all of them. Only then, we are able to identify the actual key in a separate step (cf. Section 5).

Similarly, we can present figures for an attack scenario where three challenge-response pairs are available. In this attack, we must test 2^{55} ANSI X9.9 authenticators on average to find the corresponding key what is half the time complexity of an attack having two known pairs of data. Note that all presented figures of Table 1 include material costs only (not taking energy and development costs into account).

For comparison with our hardware-based cluster, we have included estimations for a Intel Pentium 4 processor operating at 3GHz. This microprocessor allows for a throughput of about 2 million DES computations a second and thus we assume this as throughput estimate for generating ANSI X9.9 authenticators.

Table 1. Cost-performance figures for attacking the ANSI X9.9 scheme with two and three known challenge-response pairs (c_i, r_i).

Hardware System	**Cost**	**Attack using two pairs** (c_i, r_i)	**Attack using three pairs** (c_i, r_i)
1 Pentium4 @ 3GHz	\$ 50	1170 years	585 years
1 FPGA XC3S1000	\$ 50	4.8 years	2.4 years
1 COPACOBANA	\$ 10,000	14.5 days	7.2 days
100 COPACOBANAs	\$ 1 million	3.5 hours	104 min

Summarizing, an investment of \$ 1 million in COPACOBANA systems can break the ANSI X9.9-based security system of a bank account in less than 2 hours with only three given challenge-response pairs.

8 Conclusion

In this work we have presented a first hardware attack on the ANSI X9.9 authentication scheme as used in challenge-response protocols based on OTP-tokens in banking applications. Since legacy ANSI X9.9 employs the use of DES, it can be broken using a specialized brute-force attack based on special-purpose hardware. The key of OTP-token computing ANSI X9.9 responses to challenges issued by a banking application for user and transaction authentication can be broken for \$ 10,000 using at most three challenge-response pairs in about a week. Even worse, the performance of the attack can be scaled linearly according the the attacker's budget.
In fact, our work shows that not only the DES encryption scheme is completely broken but also its related standards like ANSI X9.9 as used in banking use-cases. We like to enforce all affected institutes to migrate their systems as fast as possible to abandon DES-based cryptography when not already done.

We would like to point out that software-only attacks against DES still take more than 1,000 PC-years (worst case), which is still a formidable hurdle even for determined attackers outside large organizations like government agencies. Thus, one lesson learned appears to be that it is not sufficient to only look at an attacker's available budget for computation, but also at the *type* of available computational platform.

Note: The attack architecture described in Section 6.3 has already been extensively tested using simulated challenge-response pairs. For an attack on existing devices, we acquired an off-the-shelf ANSI X9.9 OTP-token which is commercially available and currently used in real-world eCommerce systems. All upcoming results of this attack on an actual token will be presented on the project website of COPACOBANA available at [19].

References

1. Accredited Standards Committee X3. American National Standard X3.92: Data Encryption Algorithm (DEA), 1981.
2. Accredited Standards Committee X9. American National Standard X9.9: Financial Institution Message Authentication, 1994.
3. ActivIdentity. Token-based Identity Systems (OTP Tokens), 2007. `http://www.activeidentity.com`.
4. M. Blaze, W. Diffie, R. L. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener. Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security: A Report by an Ad Hoc Group of Cryptographers and Computer Scientists. Technical report, January 1996. `http://www.counterpane.com/keylength.html`.
5. Coppersmith, Knudsen, and Mitchell. Key recovery and forgery attacks on the macDES MAC algorithm. In *CRYPTO: Proceedings of Crypto*, volume 1880, page pp. 184. LNCS, 2000.
6. W. Diffie and M. E. Hellman. Exhaustive cryptanalysis of the NBS DES. *Computer*, 10(6):74–84, June 1977.
7. Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design*. O'Reilly & Associates Inc., July 1998.
8. International Organization for Standardization (ISO). ISO 8730/8731:1990 – Banking – Requirements for message authentication, 1990.
9. International Organization for Standardization (ISO). ISO 16609:2004 – Banking – Requirements for message authentication using symmetric techniques, 2004.
10. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler. Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker. In *Proc. of CHES 2006*, volume 4249, pages 101–118. Springer-Verlag, LNCS, 2006.
11. National Institute for Standards and Technology (NIST). FIPS PUB 113: Standard for computer data authentication, May 1985.
12. National Institute for Standards and Technology (NIST). FIPS PUB 46-2: Data Encryption Standard (DES), 1993.
13. National Institute for Standards and Technology (NIST). FIPS PUB 46-3: Data Encryption Standard (DES) and Triple DES (TDES), 1999.
14. National Institute for Standards and Technology (NIST). FIPS 197: Advanced Encryption Standard (AES), 2001.
15. National Institute of Standards and Technology. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, May 2004. `http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf`.
16. B. Preneel and P.C. Van Oorschot. Key recovery attack on ANSI X9.19 retail MAC. In *Electronics Letters*, volume 32 of *17*, pages 1568–1569. IEEE, Dept. of Electr. Eng., Katholieke Univ., Leuven, 1996.
17. G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat. Design Strategies and Modified Descriptions to Optimize Cipher FPGA Implementations: Fast and Compact Results for DES and Triple-DES. In *Field-Programmable Logic and Applications - FPL*, pages 181–193, 2003.
18. RSA - The Security Division of EMC^2. RSA SecurID, 2007. `http://www.rsa.com/`.
19. Sciengines GmbH. COPACOBANA - A Codebreaker for DES and other Ciphers. project and company website, 2008. `http://www.copacobana.org` and `http://www.sciengines.de`.
20. Verisign. Activcard tokens. Data Sheet. `http://www.verisign.com.au/guide/activcard/ActivCard_Tokens.pdf`.
21. M. J. Wiener. Efficient DES Key Search. In William R. Stallings, editor, *Practical Cryptography for Data Internetworks*, pages 31–79. IEEE Computer Society Press, 1996.
22. M. J. Wiener. Efficient DES Key Search: An Update. *CRYPTOBYTES*, 3(2):6–8, Autumn 1997.