The Theory of Dynamic Encryption, a New Approach to Cryptography

Bo Dömstedt[1], Jesper Jansson[2]

[1] Protego Information AB, Ideon, 223 70 Lund, Sweden,
bo.domstedt@protego.se
[2] Dept. of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden,
Jesper.Jansson@cs.lth.se

Abstract. In this paper we introduce a cipher where the mapping from the plaintext to the ciphertext consists of a computational process that will generate a new encryption system in response to every combination of input messages, initialization vectors, and cipher keys. The construction will have dynamic cryptanalytic properties, which we show is an obstacle that prevents the cryptanalyst from breaking the cipher. Ciphers of the introduced type may be adopted to implementation constraints and application specific issues, thereby substantially increasing the technical efficiency of implemented ciphers.

Keywords: Cipher construction, dynamic encryption, complexity theory.

1 Introduction

A message may be protected in transit or in storage by *encryption* [4,6] (Fig.1). The input message M is called *plaintext*. The *ciphertext* $C = f(K, M)$, an unintelligible from of the plaintext, is computed as a function of the plaintext and a finite secret *cipher key* K. The legitimate receiver may recover the plaintext from the ciphertext by applying an inverse transformation $M = f^{-1}(K, C)$. Both the sender and the receiver share a secret key K that must be distributed between the parties using secure means.

The scientist who construct ciphers is called a *cryptographer*. We assume that a *cryptanalyst* (or *enemy*) will be trying to find the secret cipher key K, or converting the ciphertext C back into intelligible form.

The distribution of message lengths, transmission frequencies, transmission errors, frequencies of repeated messages, context switching frequencies, etc., is likely to be different for every new application area. Ciphers may be used for many purposes, like link encryption, storage encryption, or authentication. It may be required that the cipher prevents a certain attack, or that it can easily be adopted to a certain plaintext format. Some ciphers will be implemented in software. Other ciphers will be implemented on Smart-Cards, FPGA:s, or on ASIC:s. Different implementation strategies will perform differently. A mismatch

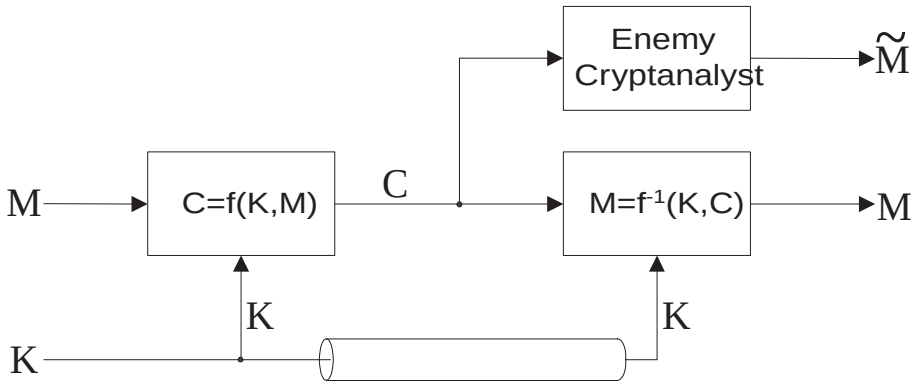

Fig. 1

between the selected cipher and the target technology and application area could decrease the technical efficiency obtained.

It is evident that no single cipher [9], no matter how flexible and efficient, will be optimal in meeting this challenge in respect to the above mentioned constraints. There will be a need for new cipher systems that meet the evolving demands of various application areas.

When a cryptographer is designing a new cipher, its security level may be difficult to establish. The security is an estimation of how difficult it would be to break the cipher without knowing the secret cipher key. Conventionally, it is assumed that the analysis made by the cryptographer and the cryptanalyst is based upon identical information – the cryptanalyst knows the system being used [1] (ch 11.2.2). A key point, that we show in this paper, is that this condition is necessary. A cryptanalytic break implies that the cryptanalyst has obtained a part of the secrets of the cipher corresponding to the degree of success. This opens the possibility to challenge this fundamental assumption by introducing a construction that will prevent the cryptanalyst from learning the details of the cipher being used.

A short description of classical methods for cipher construction is given. The proposed construction is introduced, including a short overview of major implementation problems. Finally, a short analysis is given of the properties of the proposed system.

2 Classical Cipher Construction Methods

The cryptanalyst may attack a cipher in a variety of ways [1, 5]. He may even discover new cryptanalytic methods that could possibly break a cipher efficiently and thoroughly [2]. The only objective of the cryptanalyst is to recover the plaintext. There is no restriction on what kinds of mathematics or methods that may be used.

An intuitive approach to the problem of constructing secure cipher functions is to evaluate proposed ciphers, and based upon the findings, incrementally improve these implementations. In practice, the cryptographer may only protect the cipher from cryptanalytic techniques known to him. Weak ciphers, that may be broken using existing methods, cannot be used for real applications [11] (ch 13.4). Stronger ciphers, that cannot be successfully cryptanalysed, will have unknown strength [11] (ch 12.3). It will not be clear, until a new cryptanalytic method is discovered, if these systems need to be improved.

3 Dynamic Encryption

Suppose that a cryptanalyst is trying to identify a function $C = F_1(M)$ by using a table of corresponding pairs of input and output $\{M_i, C_i\}$. For special functions $F()$, for example linear functions, this problem may be solved efficiently. That can not be the case for general functions. As an example, set $C = F_1(M) = |M|$ and compare $F_1(M)$ to $F_2(M)$, where $F_2(M) = \begin{cases} |M| & \text{if } M \neq M_0 \\ 0 & \text{if } M = M_0 \end{cases}$.

It is clear that no finite number of experiments $\{M_i, C_i\}$ will suffice for the cryptanalyst to separate the two functions $F_1()$ and $F_2()$ from each other, as the arbitrary number M_0 may have been set to any value. We have proven the following theorem:

Theorem 1. *There can not exist an algorithm that can identify a general computational process based upon the input/output relation.*

We conclude that if we use a cipher that includes a general computational process, and keep all construction parameters of that process secret, the cryptanalyst will face a problem which he will be unable to solve. We must however carefully observe the principle of A. Kerckhoffs that "No inconvenience should occur if the system falls into the hands of the enemy." [1] (pp 196). We see specifically that simply using an "optimal" encryption algorithm, that is kept secret, will not be a solution.

Assuming that Church-Turing's thesis is true, any general computational process may be implemented as a software for a universal Turing machine, or a general purpose computer [10, 13]. The specific universal machine that we will make use of here must have a few specific properties. It should be designed into accepting any binary string as valid input, i.e no input string shall be rejected as having wrong syntax. This requirement is equivalent to that the set of operations, of the universal machine, is devised such that an operation will be selected in response to any possible input information stream [3]. This modification is of no difficulty, and can be implemented without restricting the set of possible computations.

The input stream must further be kept secret, as knowledge of this would essentially be equivalent to knowing the key of the system. We propose selecting the combination of the secret cipher key K and the input plaintext M as the secret input stream. This choice will not pose any difficulties, as the universal

machine may use any binary string as input. We see that the secret input stream and the internal memory of the universal machine, may easily be protected during encryption or decryption, and can be erased afterwards.

The output from the universal machine would be in the form of a binary string [7] (ch 7.7), that we use as an internal key stream to an invertible mapping. We note that this is required as the universal machine will implement a one-way function, and an identical output can be obtained only by applying identical input. We have reached the configuration shown in Fig. 2, where feedback loops are shown from both the input plaintext and the output ciphertext, to the input of the universal machine.

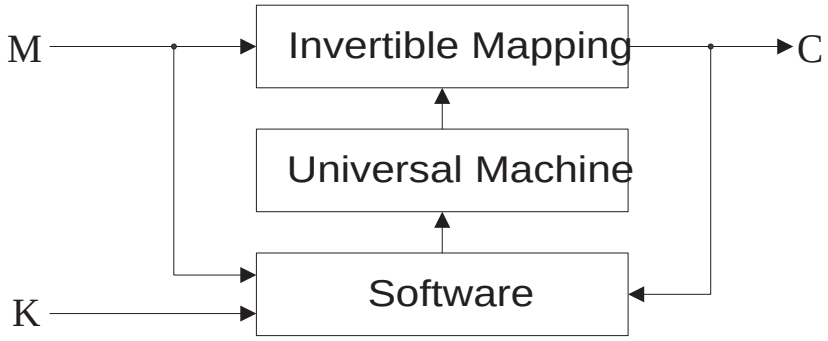

Fig. 2

In any actual implementation of Dynamic Encryption, the internal memory of the universal machine will be limited. This will collapse the set of computations that can be performed into a subset of the regular machines [7] (ch 2). This will not necessarily be an exploitable weakness as there may not be any way of deducing the number of internal states, or how they are connected, just by observing the output string corresponding to a single input string. Intuitively, a higher-level description of the computation must be found to facilitate cryptanalysis.

From an overall point of view we see that when designing an implementation of dynamic encryption, effort must be made to "approximate" the computational power of the unrestricted universal Turing machine by selecting design parameters accordingly. Particularly, we note that the size of the secret input stream should not be too small, and we conclude that the minimum size of the input message, or rather its entropy, must not be too low. The same argument applies to the secret cipher key K [3], that must be substantially longer than the "combination lock" type of cipher keys used in conventional ciphers [11] (ch 7.1, 7.5).

4 Analysis of the Proposed System

Any successful break of a cipher, equivalent to the complete recovery of the keys of the cipher, will reveal to the cryptanalyst all details of the cipher, except parts that cannot influence the ciphertext. This is obvious as the cryptanalyst may simulate the encryption of a message using the recovered key. If the cryptanalyst has not obtained a sufficiently detailed description of the cipher to facilitate cryptanalysis, the cryptanalyst must first investigate the structure of the system prior to searching for the secret plaintext. We conclude that if the cryptanalyst cannot obtain a sufficiently detailed description of the cipher, cryptanalysis will not be possible.

If we, as another example, assume that the cryptanalyst has been able to break a particular instance of the secret input stream $P_n = \{M_n, K_n\}$, the cryptanalysis method will apply to another stream only if the cryptanalysis method can be applied in general. If the break, on the other hand, exploits some specific weakness of the input stream P_n, the cryptanalysis method will only apply to this stream, and cannot work against any other combination of key and message. We conclude that, if specific instances of ciphertext are vulnerable, the cryptanalysis effort must be restarted from the beginning for every transmitted message.

We note that the general problem of investigating the properties of software has been extensively studied in the software industry [8]. We may compare an cryptanalyst, attempting cryptanalysis on Dynamic Encryption, and a software engineer, struggling with debugging a problematic software. The cryptanalyst would be investigating the properties of a universal machine that reads a string $\{M, K\}$ and outputs a string $\{C\}$. We assume that the string $\{M\}$ is known by the cryptanalyst, who attempts to find a key $\{K\}$ such that $\{C\} = \{C_0\}$. The software engineer will be investigating the behaviour of a general purpose computer executing a software $\{P\}$ with input $\{x\}$. The engineer observes the output $\{Y\}$, and tries to find an input $\{x\}$ that makes the computation behave in some specified way $\{Y\} = \{Y_0\}$. The comparison $\{C\} = \text{computation}(\{M, K\})$; $\{Y\} = \text{computation}(\{P, x\})$ shows that breaking the proposed cipher will be at least as hard as debugging software. Clearly, the cryptanalyst will not be allowed to inspect the software, single-step using a debugger, or inspect the internal state of the memory, tools that are essential for the success of the software engineer.

5 Concluding Remarks

That certain properties of the universal Turing machine leads to computationally difficult problems is well known [7, 10, 13], and it has previously been suggested that this may be one of the foundations of cipher security [14, 15]. We have found that the security of the proposed concept may be argued from both a theoretical and a practical point of view. The proposed system has in our opinion many interesting and favorable properties that have not been exploited before. In this paper an attempt has been made to further investigate this concept and create a platform for further research in this area.

References

1. Bauer, Friedrich L. "Decrypted Secrets - Methods and Maxims of Cryptology", Springer-Verlag Berlin Heidelberg 1997, ISBN 3-540-60418-9.
2. Biham, Eli and Shamir, Adi. "Differential Cryptanalysis of the Data Encryption Standard", Springer-Verlag, 1993, ISBN 0-387-97930-1, 3-540-97930-1.
3. Dömstedt, Bo and Stenfeldt, Mats. "Processing method and apparatus for converting information from a first format into a second format", Patent Applications PCT/SE99/01740; EP 98118910.3, Lateca Computer Inc N.V.
4. Fåk, Viiveke (ed.); Ekhall, Stig-Arne; Cristoffersson, Per; Widman, Kjell-Ove; et al. "Crypto Users' Handbook", North-Holland 1988, ISBN 0-444-70484-1.
5. Hellman, M.; Merkle, R.; Schroeppel, R.; Washington, L.; Diffie, W.; Pohlig S.; Schweitzer, P. "Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard", Technical Report SEL-76-042, Sept 9, 1976, Information Systems Laboratory, Department of Electrical Engineering, Stanford University.
6. Herlestam, Tore. "Kryptering - för säkerhets skull", Tidningen Elteknik med aktuell elektronik, pages 24-26 in #14, 1979.
7. Hopcroft, John E. and Ullman, Jeffrey D. "Introduction to Automata Theory, Languages and Computation", Addison-Wesley Publishing Company, 1979, ISBN 0-201-02988-X.
8. Kaner, Cem; Falk, Jack; Nguyen, Hung Quoc. "Testing Computer Software", Second Edition, Thomson Computer Press, 1993, ISBN 1-85032-847-1.
9. National Institute of Standards and Technology. "Advanced Encryption Standard (AES) Development Effort", http://csrc.nist.gov/encryption/aes/
10. Papadimitriou, Christos H. "Computational Complexity", Addison-Wesley Publishing Company, 1994, ISBN 0-201-53082-1.
11. Schneier, Bruce. "Applied Cryptography", Second edition, John Wiley & Sons, 1996, ISBN 0-471-11709-9.
12. Shannon, Claude Elmwood. "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol 28, 1949, pp 656-715.
13. Turing, Alan Mathison. "On computable numbers, with an application to the Entscheidungsproblem", Proc. Lond. Math. Soc. 42, 230-265 1936; received May 25,1936, Appendix added August 28; A correction, ibid., 43 pp 544-546, 1937.
14. Wolfram, Stephen. "Computer Software in Science and Mathematics", Scientific American, Vol 251 pp 188-203, Sept 1984.
15. Wolfram, Stephen. "Origins of Randomness in Physical Systems", Physical Review Letters, Vol 55, pp 449-452, 29 July 1985.