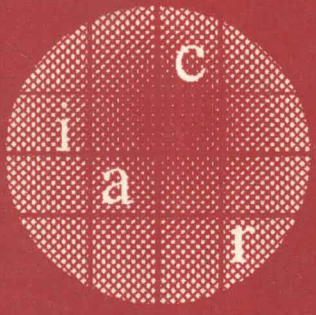
Bimal Roy
Willi Meier (Eds.)

LNCS 3017

Fast Software Encryption

11th International Workshop, FSE 2004
Delhi, India, February 2004
Revised Papers



Springer

Bimal Roy Willi Meier (Eds.)

Fast Software Encryption

11th International Workshop, FSE 2004
Delhi, India, February 5-7, 2004
Revised Papers

Springer

Volume Editors

Bimal Roy
Applied Statistics Unit, Indian Statistical Institute
203, B.T. Road, Calcutta 700 108, India
E-mail: bimal@isical.ac.in

Willi Meier
FH Aargau, 5210 Windisch, P.O. Box , Switzerland
E-mail: meierw@fh-aargau.ch

Library of Congress Control Number: 2004107501

CR Subject Classification (1998): E.3, F.2.1, E.4, G.2, G.4

ISSN 0302-9743
ISBN 3-540-22171-9 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN: 11011880 06/3142 5 4 3 2 1 0

Preface

Fast Software Encryption is a now eleven years old workshop on symmetric cryptography, including the design and analysis of block ciphers and stream ciphers as well as hash functions and message authentication codes. The FSE workshop was first held in Cambridge in 1993, followed by Leuven in 1994, Cambridge in 1996, Haifa in 1997, Paris in 1998, Rome in 1999, New York in 2000, Yokohama in 2001, Leuven in 2002, and Lund in 2003.

This Fast Software Encryption Workshop, FSE 2004, was held February 5–7, 2004 in Delhi, India. The workshop was sponsored by IACR (the International Association for Cryptologic Research) and organized in cooperation with the Indian Statistical Institute, Delhi, and the Cryptology Research Society of India (CRSI).

This year a total of 75 papers were submitted to FSE 2004. After a seven week reviewing process, 28 papers were accepted for presentation at the workshop. In addition, we were fortunate to have in the program two invited talks by Adi Shamir and David Wagner.

During the workshop a rump section was held. Seven presentations were made and all the presenters were given the option of submitting their presentations for possible inclusion in the proceedings. Only one paper from this session was submitted, which was refereed and accepted. This paper appears at the end of these proceedings.

We would like to thank the following people. First Springer-Verlag for publishing the proceedings in the Lecture Notes in Computer Science series. Next the submitting authors, the committee members, the external reviewers, the general co-chairs Subhamoy Maitra and R.L. Karandikar, and the local organizing committee, for their hard work. Bart Preneel for letting us use COSIC's Webreview software in the review process and Thomas Herlea for his support. We are indebted to Lund University, especially Thomas Johansson, Bijit Roy and Sugata Gangopadhyay for hosting the Webreview site. Additionally we would like to thank Partha Mukhopadhyay, Sourav Mukhopadhyay, Malapati Raja Sekhar, and Chandan Biswas for handling all the submissions and Madhusudan Karan for putting together the pre-proceedings. We would also like to thank the sponsors: Infosys Technology Ltd., Honeywell Corporation and Via Technology.

The organizing committee consisted of Sanjay Burman (CAIR, Bangalore), Ramendra S. Baoni (Bisecure Technologies Pvt. Ltd., Delhi), Hiranmoy Ghosh (Tata Infotech Ltd. Delhi), Abdul Sakib Mondal (Infosys Technologies Ltd., Bangalore), Arup Pal (ISI, Delhi), N.R. Pillai (SAG, Delhi), P.K.Saxena (SAG, Delhi), and Amitabha Sinha (ISI, Kolkata), who served as Treasurer. Thank you to them all.

February 2004

Bimal Roy and Willi Meier

Fast Software Encryption 2004

February 5–7, 2004, Delhi, India

Sponsored by the
International Association for Cryptologic Research

in cooperation with the
Indian Statistical Institute, Delhi
and
Cryptology Research Society of India

General Co-chairs
Subhamoy Maitra, Indian Statistical Institute, Kolkata
and
R.L. Karandikar, Indian Statistical Institute, Delhi

Program Co-chairs
Bimal Roy, Indian Statistical Institute, Kolkata
and
Willi Meier, Fachhochschule Aargau, Switzerland

Program Committee

Eli Biham	Technion Israel
Claude Carlet	INRIA, France
Don Coppersmith	IBM, USA
Cunsheng Ding	Hong Kong University of Science and Technology
Helena Handschuh	Gemplus, France
Thomas Johansson	Lund University, Sweden
Charanjit S. Jutla	IBM Research, USA
Lars R. Knudsen	Technical University of Denmark
Kaoru Kurosawa	Ibaraki University, Japan
Kaisa Nyberg	Nokia, Finland
C. Pandu Rangan	Indian Institute of Technology, Chennai
Dingyi Pei	Chinese Academy of Sciences
Bart Preneel	K.U. Leuven, Belgium
Matt Robshaw	Royal Holloway, University of London, U.K.
Serge Vaudenay	EPFL, Switzerland
C.E. Veni Madhavan	Indian Institute of Science, Bangalore
Xian-Mo Zhang	Macquarie University, Australia

External Reviewers

Gildas Avoine
Thierry Berger
Christophe Clavier
Orr Dunkelman
Marc Girault
Shoichi Hirose
Fredrik Jönsson
Ju-Sung Kang
Yong Li
Marine Minier
Enes Pasalic
Haavard Raddum
Xiaojian Tian
Akihiro Yamamura
Thomas Baignères
Qingjun Cai
Nicolas Courtois
Eric Filiol
Philippe Guillot
Tetsu Iwata
Jakob Johnsson
Tania Lange
Yi Lu
Jean Monnerat
Souradyuti Paul
Palash Sarkar
Xuesong Wang
Julien Brouchier
Elad Barkan
Carlos Cid
Christophe De Cannière
Henri Gilbert
Louis Guillou
Thomas Jakobsen
Pascal Junod
Joseph Lano
Miodrag Mihaljevic
Sean Murphy
Michael Quisquater
Takeshi Shimoyama
Guohua Xiong
Ju-Sung Kang

Table of Contents

New Cryptographic Primitives Based on Multiword T-Functions

Alexander Klimov and Adi Shamir

Computer Science department
The Weizmann Institute of Science, Rehovot 76100, Israel
{ask,shamir}@weizmann.ac.il

Abstract. A *T-function* is a mapping from n-bit words to n-bit words in which for each $0 \leq i < n$ bit i of the output can depend only on bits $0, 1, \ldots, i$ of the input. All the boolean operations and most of the numeric operations in modern processors are T-functions, and their compositions are also T-functions. In earlier papers we considered 'crazy' T-functions such as $f(x) = x + (x^2 \vee 5)$, proved that they are invertible mappings which contain all the 2^n possible states on a single cycle for any word size n, and proposed to use them as primitive building blocks in a new class of software-oriented cryptographic schemes. The main practical drawback of this approach is that most processors have either 32 or 64 bit words, and thus even a maximal length cycle (of size 2^{32} or 2^{64}) may be too short. In this paper we develop new ways to construct invertible T-functions on multiword states whose iteration is guaranteed to yield a single cycle of arbitrary length (say, 2^{256}). Such mappings can lead to stream ciphers whose software implementation on a standard Pentium 4 processor can encrypt more than 5 gigabits of data per second, which is an order of magnitude faster than previous designs such as RC4.

1 Introduction

There are two basic approaches to the design of secret key cryptographic schemes, which we can call 'tame' and 'wild'. In the tame approach we try to use only simple primitives (such as linear feedback shift registers) with well understood behaviour, and try to prove mathematical theorems about their cryptographic properties. Unfortunately, the clean mathematical structure of such schemes can also help the cryptanalyst in his attempt to find an attack which is faster than exhaustive search. In the wild approach we use crazy compositions of operations (which mix a variety of domains in a nonlinear and nonalgebraic way), hoping that neither the designer nor the attacker will be able to analyse the mathematical behaviour of the scheme. The first approach is typically preferred in textbooks and toy schemes, but real world designs often use the second approach.

In several papers published in the last few years [5, 6], we tried to bridge this gap by considering 'semi-wild' constructions which look like crazy combinations of boolean and arithmetic operations, but have many analyzable mathematical properties. In particular, we defined the class of T-functions which contains arbitrary compositions of plus, minus, times, or, and, xor operations on n-bit words,

B. Roy and W. Meier (Eds.): FSE 2004, LNCS 3017, pp. 1–15, 2004.

and showed that it is easy to analyse their invertibility and cycle structure for arbitrary word sizes. Such constructions can replace LFSRs and linear congruential mappings (which are vulnerable to correlation and algebraic attacks) in a new class of stream ciphers and pseudo random generators.

The paper is organized in the following way. In section 2 we recall the basic definitions from [5] for single word mappings, and consider several ways in which they can be extended to the multiword case. In section 3 we extend our bit-slice technique to analyse the invertibility of multiword T-functions. In section 4 we extend our technique from [6] to analyse the cycle structure of multiword T-functions. Finally, in section 5 we provide experimental data on the speed of several possible implementations of our functions on a PC.

2 Multiword T-Functions

Invertible mappings with a single cycle have many cryptographic applications. The main context in which we study them in this paper is pseudo random generation and stream ciphers. Modern microprocessors can directly operate on up to 64-bit words in a single clock cycle, and thus a univariate mapping can go through at most 2^{64} different states before entering a cycle. In some cryptographic applications this cycle length may be too short, and in addition the cryptanalyst can guess a 64 bit state in a feasible computation. A common way to increase the size of the state and extend the period of a generator is to run in parallel and combine the outputs of several generators with different periods. The overall period is determined by the least common multiple of their individual periods. This works well with LFSRs, whose periods $2^{n_1}-1, 2^{n_2}-1, \ldots$ can be relatively prime, and thus the overall period can be their product. However, our univariate mappings have periods of $2^{n_1}, 2^{n_2}, \ldots$ whose least common multiple is just $2^{\max(n_1,n_2,\ldots)}$.

A partial solution to this problem is to cyclically use a large number of different state update functions, starting from a secret state and a secret index. For example, we can use 64-bit words and $2^{16}-1$ different constants C_k to get a guaranteed cycle length of almost 2^{80} from the following simple generator:

Theorem 1. *Consider the sequence* $\{(x_i, k_i)\}$ *defined by iterating*

$$x_{i+1} = x_i + (x_i^2 \vee C_{k_i}) \bmod 2^n,$$
$$k_{i+1} = k_i + 1 \bmod m,$$

where each x *is an* n*-bit word and* C_k *is some* n*-bit constant for each* $k = 0, \ldots, m-1$*. Then the sequence of pairs* (x_i, k_i) *has a maximal period (of size* $m2^n$*) if and only if* m *is odd, and for all* k, $[C_k]_0 = 1$ *and* $\bigoplus_{k=0}^{m-1} [C_k]_2 = 1$.

A special case of this theorem for $m = 1$ is that the function $f(x) = x + (x^2 \vee C)$ is invertible with a single cycle if and only if both the least significant bit and the third least significant bit in C are 1, and the smallest such C is 5.

Unfortunately, the cyclic change of state update functions is inconvenient, and it cannot yield really large cycles (e.g., of 2^{256} possible states). We can try to solve the problem by using a single high precision variable x (say, with 256 bits), but the multiplication of such long variables can become prohibitively expensive. What we would like to do is to define the mapping by operating separately on the various input words, without trying to interpret the result as a natural mathematical operation on multi-precision words.

Let us first review the definitions from [5] in the case of univariate mappings. Let x be an n-bit word. We can view x as a vector of bits denoted by $([x]_{n-1}, \ldots, [x]_0)$, where the least significant bit has number 0. In this bit notation, the univariate function $f(x) = x + 1 \pmod{2^n}$ can be expressed in the following way:

$$\begin{array}{lll} [f(x)]_0 & = f_0([x]_0) & = [x]_0 \oplus 1 \\ [f(x)]_1 & = f_1([x]_1 ; [x]_0) & = [x]_1 \oplus \alpha_1([x]_0) \\ [f(x)]_2 & = f_2([x]_2 ; [x]_1 , [x]_0) & = [x]_2 \oplus \alpha_2([x]_1 , [x]_0) \\ & \vdots & \vdots \\ [f(x)]_{n-1} & = f_{n-1}([x]_{n-1} ; [x]_{n-2}, \ldots, [x]_0) & \\ & = [x]_{n-1} \oplus \alpha_{n-1}([x]_{n-2}, \ldots, [x]_0), & \end{array} \tag{1}$$

where each α_i denotes one of the carry bits. Note that for any bit position i, $[f(x)]_i$ depends only on $[x]_i, \ldots, [x]_0$ and does not depend on $[x]_{n-1}, \ldots, [x]_{i+1}$. We call any univariate function f which has this property a *T-function* (where 'T' is short for *triangular*). Note further that each carry bit α_i depends only on strictly earlier input bits $[x]_{i-1}, \ldots, [x]_0$ but not on $[x]_i$. This is a special type of a T-function, which we call a *parameter*. To provide some intuition from the theory of linear transformations on n-dimensional spaces, we can say that T-functions roughly correspond to lower triangular matrices, parameters roughly correspond to lower triangular matrices with zeroes on the diagonal, and a T-function can be roughly represented as a diagonal matrix plus a parameter.

Let us now define these notions for functions which map several input words into one output word. The natural extension of the notion of a T-function in this case is to allow bit i of the output to depend only on bits 0 to i of each one of the inputs. The observation which makes this notion interesting is that all the boolean operations and most of the arithmetic operations available on modern processors are T-functions. In particular, addition ('+'), subtraction ('binary $-$'), negation ('unary $-$'), multiplication ('$*$'), or ('$\vee$'), and ('$\wedge$'), exclusive or ('$\oplus$'), and complementation ('$\neg$') (where the boolean operations are performed on all the n bits in parallel and the arithmetic operations are performed modulo 2^n) are T-functions with one or two inputs. We call these eight functions *primitive operations*. Note that circular rotations and right shifts are not T-functions, but left shifts can be expressed as multiplication by a power of 2 and thus they are T-functions. Since the composition of T-functions is also a T-function, any 'crazy' function which contains arbitrarily many primitive operations is always a T-function.

In order to define multiword mappings f which can be iterated, we have to further extend the notion to functions with the same number m of input and output words. We can represent the multiword input as the following $n \times m$ bit matrix $B^{n \times m}$:

$$x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{pmatrix} = \begin{pmatrix} [x]_{0,n-1} & \cdots & [x]_{0,1} & [x]_{0,0} \\ [x]_{1,n-1} & \cdots & [x]_{1,1} & [x]_{1,0} \\ \vdots & \ddots & \vdots & \vdots \\ [x]_{m-1,n-1} & \cdots & [x]_{m-1,1} & [x]_{m-1,0} \end{pmatrix}. \tag{2}$$

We can now consider the columns of the bit matrix as parallel bit slices with no internal bit order, and say that a multiword mapping is a T-function if all the bits in column i of the matrix of output words can depend only on bits in columns 0 to i of the matrix of input words. In this interpretation it is still true that any composition of primitive operations is a multiword T-function, but some of the proven properties of univariate T-functions (e.g., that all the cycle lengths are powers of 2) are no longer true.

An alternative definition of multiword T-functions is to concatenate all the input words into one long word, to concatenate all the output words into one long word, and then to use the standard univariate definition of a T-function in order to limit which input bits can affect which output bits. If we denote the l input words by $x_u, x_v, \ldots$, then we define the single logical variable x by

$$x = (x_u, \ldots, x_w) = ([x]_{n(l-1)+(n-1)}, \ldots, [x]_{n(l-1)}, \ldots [x]_{n-1}, \ldots, [x]_0). \tag{3}$$

Note that in this interpretation $f(x) = (f_u, f_v) = (x_u + x_v, x_v)$ is a T-function, but the very similar $f(x) = (f_u, f_v) = (x_u, x_u + x_v)$ is not a T-function, and thus we cannot compose primitive operations in an arbitrary way. On the other hand, we can obtain many new types of T-functions in which low-order words can be manipulated by non-primitive operations (such as cyclic rotation) before we use them to compute higher order output words.

Our actual definition of multiword T-functions combines and generalizes these two possible interpretations. Let x be an $nl \times m$ bit matrix ($\mathbb{B}^{nl \times m}$):

$$\begin{pmatrix} [x]_{0,n(l-1)+(n-1)} & \cdots & [x]_{0,n(l-1)+1} & [x]_{0,n(l-1)} & \cdots & [x]_{0,n-1} & \cdots & [x]_{0,0} \\ [x]_{1,n(l-1)+(n-1)} & \cdots & [x]_{1,n(l-1)+1} & [x]_{1,n(l-1)} & \cdots & [x]_{1,n-1} & \cdots & [x]_{1,0} \\ \vdots & \ddots & \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ [x]_{m-1,n(l-1)+(n-1)} & \cdots & [x]_{m-1,n(l-1)+1} & [x]_{m-1,n(l-1)} & \cdots & [x]_{m-1,n-1} & \cdots & [x]_{m-1,0} \end{pmatrix}. \tag{4}$$

We consider it as an $m \times l$ matrix of n bits words:

$$x = \begin{pmatrix} x_{0,u} & \cdots & x_{0,w} \\ \vdots & \ddots & \vdots \\ x_{m-1,u} & \cdots & x_{m-1,w} \end{pmatrix}.$$

We concatenate the l words in each row into a single logical variable, and then consider the collection of the m long variables as the inputs to the T-function.

Finally, we allow the bits in column i of the output matrix to depend only on bits in columns $0, \ldots, i$ in the input matrix.

To demonstrate this notion, consider the following mapping over 4-tuples of words:

$$f(x) = \begin{pmatrix} x_{0,u} + x_{1,u}x_{0,v} & x_{0,v} + x_{1,v} \\ x_{0,u} - x_{1,u}(x_{1,v} \gg 1) & x_{0,v} \oplus x_{1,v} \end{pmatrix}.$$

This is a valid T-function under our general multiword definition even though it contains the non-primitive right shift operation $\gg 1$.

3 Bit Slice Analysis and Invertibility

The main tool we use in order to study the invertibility of T-functions is bit slice analysis. Its basic idea is to define the mapping from $[x]_i$ to $[f(x)]_i$ by abstracting out the complicated dependency on $[x]_{0\ldots i-1}$ via the notion of parameters. For example, the size of the explicit description of the mapping $[f(x)]_i = \phi([x]_0, \ldots, [x]_i)$ in the function $f(x) = x + (x^2 \vee 5)$ grows exponentially with i, but it can be written as $[f(x)]_i = [x]_i \oplus \alpha_i$, where α_i is some function of $[x]_0, \ldots, [x]_{i-1}$ (that is, a parameter). By using this parametric representation we can easily prove the invertibility of the mapping by induction on i, since if we already know bits 0 to $i-1$ of the input x and bit i of the output $f(x)$, we can (in principle) calculate the value of α_i and thus derive in a unique way bit i of the input. Intuitively, this is the same technique we use in order to solve a triangular system of linear equations, except that in our case the explicit description of α_i can be extremely complicated, and thus we do not use this technique as a real inversion algorithm for $f(x)$, but only in order to prove that this inverse is uniquely defined.

The main observation in [5] was that such an abstract parametric representation can be easily derived for any composition of primitive operations by the following recursive definition, in which i can be any bit position except zero:

$$\begin{array}{rl} [xy]_0 & = [x]_0 \wedge [y]_0 \\ \left[x \overset{+}{\oplus} y\right]_0 & = [x]_0 \oplus [y]_0 \\ \left[x \overset{\wedge}{\vee} y\right]_0 & = [x]_0 \overset{\wedge}{\vee} [y]_0 \\ [xy]_i & = [x]_i \, \alpha_{[y]_0} \oplus \alpha_{[x]_0} \, [y]_i \oplus \alpha_{xy} \\ \left[x \overset{+}{-} y\right]_i & = [x]_i \oplus [y]_i \oplus \alpha_{x \pm y} \\ \left[x \overset{\oplus}{\vee} y\right]_i & = [x]_i \overset{\oplus}{\vee} [y]_i \end{array} \tag{5}$$

To demonstrate this technique, consider our running example: $\left[x + (x^2 \vee 5)\right]_0 = [x]_0 \oplus \left[x^2 \vee 5\right]_0 = [x]_0 \oplus 1$ and, for $i > 0$, $\left[x + (x^2 \vee 5)\right]_i = [x]_i \oplus (\left[x^2\right]_i \vee [5]_i) \oplus \alpha_{x+(x^2 \vee 5)} = [x]_i \oplus (([x]_i \, \alpha_{[x]_0} \oplus \alpha_{[x]_0} \, [x]_i \oplus \alpha_{x^2}) \vee [5]_i) \oplus \alpha_{x+(x^2 \vee 5)} = [x]_i \oplus \alpha$.

This invertibility test can be easily generalized to the multivariate case (2). Let us show an example of such a construction. We start from an arbitrary non singular matrix which denotes a possible bit slice mapping, such as:

$$\begin{matrix} 1 & \alpha \\ 0 & 1 \end{matrix}$$

We can add to this linear mapping an affine part (where α, β, and γ are arbitrary parameters) and get the following bit slice structure:

$$\begin{pmatrix} [x_0]_i \\ [x_1]_i \end{pmatrix} \to \begin{pmatrix} [x_0]_i \oplus \alpha\,[x_1]_i \oplus \beta \\ [x_1]_i \oplus \gamma. \end{pmatrix} \tag{6}$$

It is easy to check that for $i > 0$ the i-th bit slice of the following mapping matches (6):

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \to \begin{pmatrix} x_0 + (x_0^2 \wedge x_1) \\ x_1 + x_0^2. \end{pmatrix}$$

Unfortunately, the least significant bit slice of this mapping is not invertible:

$$\begin{pmatrix} [x_0]_0 \\ [x_1]_0 \end{pmatrix} \to \begin{pmatrix} [x_0]_0 \oplus [x_0]_0\,[x_1]_0 \\ [x_1]_0 \oplus [x_0]_0\,. \end{pmatrix}$$

So we have to apply a little tweak to fix it:

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \to \begin{pmatrix} x_0 + ((x_0^2 \wedge x_1) \vee 1) \\ x_1 + x_0^2. \end{pmatrix}$$

The reader may get the impression that the bit slice mappings of invertible functions are always linear. From (5) it is easy to see that every expression which uses only $\oplus$, $+$, $-$ and $\times$ has linear i-th bit slice, but in general this is not true.

4 The Single Cycle Property

A T-function has the single cycle property if its repeated application to any initial state goes through all the possible states. Let us recall the basic results from [6] in the univariate case. Invertibility is a prerequisite of the single cycle property. If a T-function has a single cycle modulo 2^k then it has a single cycle modulo 2^{k-1}. If a T-function has a cycle of length l modulo 2^{k-1} then modulo 2^k it has either a cycle of length $2l$ or two cycles of length l. Taking into account the fact that modulo 2^1 a function has either one cycle of length two or two cycles of length one we can conclude that the size of any cycle of a T-function is always a power of 2.

In the univariate case a bit slice of an invertible T-function has the form $[f(x)]_i = [x]_i \oplus \alpha$. From (5) it follows that $f(x)$ has one of the following forms: $f_1(x) = x \oplus r_1(x)$, $f_2(x) = x + r_2(x)$ or $f_3(x) = xr_3(x)$, where the r_i are parameters (in the case of multiplication additionally we need $[r_3]_0 = 1$). It is easy to see that $[f_3(x)]_0 = [x]_0\,[r_3(x)]_0 = [x]_0$, that is it has two cycles modulo 2 and so it can not form a single cycle modulo 2^n. So, a single cycle function has either[1] the first or the second form. In order to analyse the cycle structure of these forms the following definitions of *even* and *odd* parameters

[1] Note that there is no *exclusive* or here since every function can be represented in both forms, for example $x + 1 = x \oplus (x \oplus (x + 1))$.

were introduced. Suppose that $r(x)$ is a parameter, that is $r(x) = r(x + 2^{n-1}) \pmod{2^n}$. So, $r(x) = r(x + 2^{n-1}) + 2^n b(x) \pmod{2^{n+1}}$. Consider

$$B[r,n] = 2^{-n} \sum_{i=0}^{2^{n-1}-1} (r(i + 2^{n-1}) - r(i)) \pmod 2 = \bigoplus_{i=0}^{2^{n-1}-1} b(i). \qquad (7)$$

The parameter is called *even* if $B[r,n]$ is always zero, and *odd* if $B[r,n]$ is always one[2].

Let us give several examples of even parameters:

- $r(x) = C$, where C is an arbitrary constant ($r(x) = r(x + 2^{n-1})$ and so, $b = 0$ and $B = 0$);
- $r(x) = 2x$ ($r(x + 2^{n-1}) = r(x) + 2^n \pmod{2^{n+1}}$, so $b(x) = 1$ and B is even as long as 2^{n-1} is even, that is for $n \geq 2$);
- $r(x) = x^2$ ($r(x + 2^{n-1}) = r(x) + 2^n x + 2^{2(n-1)}$, so $b(x) = [x]_0$ and B is even for $n \geq 3$);
- $r(x) = 4g(x)$, where $g(x)$ is an arbitrary T-function ($r(x + 2^{n-1}) - r(x) = 4(g(x + 2^{n-1}) - g(x)) = 0 \pmod{2^n}$);
- $r(x) = r'(x) \stackrel{+}{\oplus} r''(x)$, where r' and r'' are simultaneously even or odd parameters ($B = B' \oplus B''$).
- $r(x) = r'(x) \vee C$, where C is an arbitrary constant and $r'(x)$ is an even parameter (if $[C]_i = 0$ then $[r(x)]_i = [r'(x)]_i$, and if $[C]_i = 1$ then $[r(x)]_i = [C]_i$, so in both cases $[r(x)]_i$ is the same as for some even parameter.)

The following theorem was proved in [6]:

Theorem 2. *Let N_0 be such that $x \to x + r(x) \bmod 2^{N_0}$ defines a single cycle and for $n > N_0$ the function $r(x)$ is an* even *parameter. Then the mapping $x \to x + r(x) \bmod 2^n$ defines a single cycle for all n.*

We can use our running example of $f(x) = x + (x^2 \vee C)$ to demonstrate this theorem. If the binary form of C ends with $\ldots 1\overset{0}{1}1$, then $C = 5, 7 \pmod 8$, and $x^2 \vee C = C \pmod 8$ is an odd constant modulo 2^3 so $x + C$ has a single cycle modulo 2^3. In addition, x^2 is an even parameter for $n \geq 3$, and this is not affected by 'or'ing it with an arbitrary constant. In [6] it was shown that $x \to x + (x^2 \vee C)$ is the smallest nonlinear expression which defines a single cycle, in other words there is no nonlinear expression which defines a single cycle and consists of less than three operations.

Another important class of single cycle mappings is $f(x) = 1 + x + 4g(x)$ for an arbitrary T-function $g(x)$. It turns out that x86 microprocessors have an instruction which allows us to calculate $1 + x + 4y$ with a single instruction[3] and

[2] Note that in the general case B is a function of n, and thus the parameter can be neither even nor odd. We often relax these definitions by allowing exceptions for small n such as 1 or 2.

[3] The `lea` (load effective address) instruction makes it possible to calculate any expression of the form $C + R_1 + kR_2$, where C is a constant, R_1 and R_2 are registers and k is a power of two. Its original purpose was to simplify the calculation of addresses in arrays.

thus the single cycle mapping $f(x) = 1 + x + 4x^2$ can be calculated using only two instructions.

Odd parameters are less common and harder to construct. Their main application is in mappings of the form $x \oplus r(x)$:

Theorem 3. *Let N_0 be such that $x \to x \oplus r(x) \bmod 2^{N_0}$ defines a single cycle and for $n > N_0$ the function $r(x)$ is an* odd *parameter. Then the mapping $x \to x \oplus r(x) \bmod 2^n$ defines a single cycle for all n.*

Proof. Let us prove this by induction: suppose that the mapping defines a single cycle modulo 2^n and we are going to prove that it defines a single cycle modulo 2^{n+1}. Since there are only two possible cycle structures modulo 2^{n+1} (a single cycle of size 2^{n+1} or two cycles of size 2^n) we will prove that the second case is impossible, that is $\left[x^{(0)}\right]_n \neq \left[x^{(2^n)}\right]_n$ at least for one x. Recall that $[x]_n$ is the most significant bit of x modulo 2^{n+1}. Let $x^{(0)} = 0$, since r is a parameter it follows that $r(i) = r(i + 2^n) \pmod{2^{n+1}}$, and so $\left[x^{(2^n)}\right]_n = \left[r(x^{(0)}) \oplus \ldots \oplus r(x^{(2^n-1)})\right]_n = \bigoplus_{i=0}^{2^n-1} [r(i)]_n =$
$\bigoplus_{i=0}^{2^{n-1}-1}([r(i + 2^{n-1})]_n \oplus [r(i)]_n) = \bigoplus_{i=0}^{2^{n-1}-1}([r(i + 2^{n-1}) - r(i)]_n) = 1.$
Here we use the fact that $\left[r(i + 2^{n-1})\right]_n - [r(i)]_n = \left[r(i + 2^{n-1}) - r(i)\right]_n$.

Recently the related notions of measure preservation and ergodicity of compatible functions over p-adic numbers were independently studied by Anashin [1].[4] His motivation was mathematical rather than cryptographic, and he used different techniques. In order to study if a T-function is invertible (respectively, has a single cycle property) he tried to represent it as $f(x) = d + cx + pv(x)$ (respectively, $f(x) = c + rx + p(v(x + 1) - v(x))$), or to represent it as Mahler interpolation series, or to use the notion of uniform differentiability. The first characterization is the most general (that is $v(x)$ can be any T-function) and complete (he proved that every invertible (respectively, a single cycle function) can be represented in this form. For example, it follows that there exists $v_f(x)$, such that $f(x) = x + (x^2 \vee 5) = 1 + x + 2(v_f(x + 1) - v_f(x))$ and in order to prove that $f(x)$ defines a single cycle it is enough to find such a function $v_f(x)$. This example shows that this criterion is not that good in practice in checking properties of a given function but it allows us to construct arbitrary complex functions with needed properties. For the second approach any function f can be represented as a Mahler interpolation series $\sum_{i=0}^{\infty} a_i \left(\frac{x(x-1)\cdots(x-i+1)}{i!}\right)$. It turns out, for example, that a T-function is invertible if and only if $\|a_1\|_2 = 1$ and $\|a_i\|_2 \leq 2^{-\lfloor \log_2 i \rfloor - 1}$, for $i = 2, 3, \ldots$. This is used to prove theoretical results similar to the previous one, but once again for practical purposes it is usually hard to represent a given function as a Mahler series. The uniformly differentiable[5] func-

[4] This could be translated to our terminology as follows: *measure preservation* — invertibility, *ergodicity* — a single cycle property, *compatible* — T-function. To simplify reading we will continue to use our terminology, but an interested reader who will refer to his paper should keep in mind this "dictionary".

[5] It is exactly the same concept as the usual notion of uniform differentiability of real functions, but with respect to the p-adic distance.