Nonsmooth Cryptanalysis, with an Application to the Stream Cipher MICKEY

Elmar Tischhauser

Communicated by ???

Abstract. A new approach to the cryptanalysis of symmetric algorithms based on nonsmooth optimisation is presented. We develop this technique as a novel way of dealing with nonlinearity over $\mathbb{F}_2$ by modeling the equations corresponding to the algorithm as a continuous optimisation problem that avoids terms of higher degree. The resulting problems are not continuously differentiable, but can be approached with techniques from nonsmooth analysis. Applied to the stream cipher MICKEY, which is part of the eSTREAM final portfolio, this method can solve instances corresponding to the full cipher, although with time complexity greater than brute force. Finally, we compare this approach to classical pseudo-Boolean programming.

Keywords. Nonsmooth optimisation, cryptanalysis, stream ciphers, MICKEY, pseudo-Boolean optimisation.

2010 Mathematics Subject Classification. 94A60, 90C09, 90C56, 90C90.

1 Introduction

Symmetric encryption algorithms are among the most important building blocks for cryptographic protocols and applications. In order to meet the efficiency requirements emerging from these applications, they are commonly constructed using Boolean operations, integer arithmetic or arithmetic in finite fields of characteristic two with small extension degree which can in turn be efficiently described in terms of binary arithmetic.

In contrast to public-key cryptography, where the security of a scheme is usually based on a (tight) reduction or equivalence to a well-known “hard” computational problem, the security of symmetric primitives is based on withstanding continuous cryptanalytic evaluation. Therefore, the evolution of cryptanalytic techniques has

Elmar Tischhauser is a research assistant of the F.W.O., Fund for Scientific Research – Flanders. The work in this paper was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.
great impact on commonly accepted construction criteria for symmetric primitives. A prominent example of this evolution are differential and linear cryptanalysis and their refinements, which were successfully applied to many symmetric primitives, while as well motivating research into construction strategies providing resistance against these attacks.

1.1 “Solving” cryptographic algorithms

Generally, the operation of a symmetric primitive can be expressed as a system of equations over \( \mathbb{F}_2 \) or some field extension thereof. If an attacker is able to efficiently solve these equations (either deterministically or with non-negligible success probability), he can break the corresponding cryptographic algorithm: In the case of block or stream ciphers, this enables a known-plaintext attack by solving for the variables representing the key; for hash functions, obtaining a characterisation of the set of solutions will enable the attacker to construct collisions or even (second) preimages. It is worth noting that even the ability to solve a particular subsystem or simplified variant of the system of equations – for example with some variables assumed to have particularly useful constant values – might lead to significant cryptanalytic progress, since then all plaintexts, keys or to-be-hashed messages exhibiting this very structure will be susceptible to attack.

With nonlinearity being a fundamental requirement for a cryptographic primitive, the resulting systems of equations will typically be highly nonlinear and hence be difficult to solve directly. Solving even quadratic equations over finite fields is known to be an NP-complete problem, and approaches based on Gröbner bases [14] or linearisation techniques [16] have had limited success so far, especially when applied to full algorithms.

On the other hand, there exists a rich theory and supporting experimental evidence for solving real-valued systems of equations or optimising continuous objective functions [21, 28]. It therefore seems promising to apply the well-established algorithms from numerical optimisation to the cryptanalysis of symmetric primitives. This approach has been introduced by [32], considering polynomial models of the Boolean equations. Alternative approaches such as Mixed Integer Programming or Simulated Annealing have been followed in [11] and [10].

1.2 Formulation as continuous optimisation problems

Since numerical optimisation algorithms generally operate on (vector spaces over) the reals, a natural approach to applying them to systems of Boolean equations is to represent these systems as equations over the reals. First of all, the two Boolean
values have to be mapped to two distinct real numbers, two natural mappings being

\[ \text{False} \mapsto 0, \text{True} \mapsto 1 \quad (\text{"Standard representation"}) \]

and

\[ \text{False} \mapsto 1, \text{True} \mapsto -1 \quad (\text{"Fourier representation"}), \]

see [32]. There also exist efficient conversion mappings between these representations. Having fixed the conversion method, any Boolean function can be expressed in terms of polynomials with real coefficients. Choosing different representations for elements and Boolean operations can result in significantly different systems of equations exhibiting different numerical behaviour [32].

A clear advantage of polynomial representations of Boolean functions is the immediate applicability of well-established equation solving techniques. However, a drawback of the polynomial approach is that addition in \( \mathbb{F}_2 \) is necessarily converted into a nonlinear operation: for instance, using Fourier representation, \( a + b \) over \( \mathbb{F}_2 \) becomes \( a \times b \) over the reals. The approach proposed here seeks to avoid higher degrees by means of a nonsmooth model of Boolean equations.

Note that instead of solving a system of equations \( F(x) = 0 \) with \( F : \mathbb{R}^m \to \mathbb{R}^n \), one can alternatively solve the optimisation problem

\[
\min_{x \in \mathbb{R}^m} \| F(x) \|, \tag{1.1}
\]

where \( \| \cdot \| \) is any norm, since for all \( y \), \( \| y \| \geq 0 \) and \( \| y \| = 0 \) if and only if \( y = 0 \). Therefore, both numerical algorithms for solving equations and minimisation algorithms from nonlinear optimisation can be used. Since we are interested in solutions that can be mapped back to Boolean values, additional box constraints of the form \( x_i \in [0, 1] \) can be added.

Furthermore, unless the cipher has equivalent keys (which would be a weakness on its own), the systems of equations are expected to have precisely one solution over \( \mathbb{F}_2 \) which is not the case for optimisation problems in general.

1.3 Scope and organisation of the paper

This paper focuses on the development of a nonsmooth optimisation approach to cryptanalysis and applying it to solving equations representing the stream cipher MICKEY and small-scale variants. In nonsmooth optimisation, larger degrees of the equations can be avoided at the expense of losing the property of global differentiability. Nonsmooth optimisation algorithms generally require the resulting expressions to still be Lipschitz-continuous, which is the case for our model. To
the best of our knowledge, this is the first application of methods from nonsmooth optimisation in cryptanalysis.

Stream ciphers are a particularly attractive target for the numerical cryptanalysis approach since typically one bit of keystream is generated per state update, whereas block ciphers and hash functions commonly apply many iterations of a transformation to each plaintext or message block, which results in larger equation systems. Analyzing the stream cipher MICKEY is particularly interesting since it has a relatively small state size of 200 bits and is the only algorithm in the hardware-oriented eSTREAM portfolio without any published cryptanalytic results.

The paper is organised as follows. Section 2 briefly describes pseudo-Boolean optimisation, one of the earliest techniques using methods from continuous mathematics for discrete problems, and proves that one of its most important heuristics cannot generally approximate the optimum by a constant factor. Section 3 proposes a nonsmooth approach to solving systems of equations corresponding to cryptographic algorithms and gives an introduction to optimisation algorithms suitable for objective functions that are merely Lipschitz-continuous. In Section 4, we apply this approach to the stream cipher MICKEY and small-scale variants of it. Experimental results using different algorithms are presented and discussed for different attack scenarios. Finally, we discuss the effectiveness of the nonsmooth approach and compare it to the case of pseudo-Boolean optimisation. Section 5 concludes and outlines further directions of research in this area.

2 Pseudo-Boolean optimisation and the DDT heuristic

The idea of modeling discrete optimisation problems as continuous ones and subjecting the real-valued models to techniques inspired from nonlinear optimisation has previously been applied in the context of so-called pseudo-Boolean functions [23]. While this area is mostly aiming at efficient dedicated heuristics for NP-hard problems, the possibility of applying methods of convex analysis and nonlinear programming in general to problems which are otherwise discrete in nature is explicitly mentioned in [12].

Let $\mathbb{B} = \{0, 1\}$ and $\mathbb{U} = [0, 1]$. Functions of the type $f : \mathbb{B}^n \rightarrow \mathbb{R}$ are called pseudo-Boolean functions. They can be uniquely written as multi-linear polynomials, that is, as

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq \{1, \ldots, n\}} c_S \prod_{j \in S} x_j$$

(2.1)

with $c_S \in \mathbb{R}$. If one allows both the variables $x_i$ and their complements $\overline{x_i} \stackrel{\text{def}}{=} 1 - x_i$ to appear as literals, pseudo-Boolean functions can alternatively be written
as *posiforms*, i.e., as
\[
\phi(x_1, \ldots, x_n) = \sum_{T \subseteq \mathbb{L}} a_T \prod_{u \in T} u,
\] (2.2)

with \( \mathbb{L} = \{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\} \) and nonnegative coefficients, i.e. \( a_T \geq 0 \) if \( T \neq \emptyset \).

For \( T = \emptyset \), \( a_T \) can be negative, and by convention, \( \prod_{u \in \emptyset} u = 1 \).

The following property allows us to view the optimisation of a pseudo-Boolean function as a continuous nonlinear optimisation problem over the unit hypercube \( \mathbb{U}^n \):

**Fact 2.1** ([12]). For any pseudo-Boolean function \( f \),
\[
\min_{x \in \mathbb{B}^n} f(x) = \min_{a \in \mathbb{U}^n} f(a).
\]

By means of the derivative of a pseudo-Boolean function:
\[
\Delta_i(x) \overset{\text{def}}{=} \frac{\partial f}{\partial x_i}(x)
= f(x_1, \ldots, x_i = 1, \ldots, x_n) - f(x_1, \ldots, x_i = 0, \ldots, x_n),
\] (2.3)
a simple characterisation of local optima can be given:

**Fact 2.2** ([12]). Let \( f : \mathbb{B}^n \to \mathbb{R} \) be a pseudo-Boolean function. The vector \( x \) is a local minimum of \( f \) if and only if
\[
x_i = \begin{cases} 
1 & \text{if } \Delta_i(x) < 0, \\
0 & \text{if } \Delta_i(x) > 0
\end{cases}
\] (2.4)
for all \( i = 1, \ldots, n \).

This criterion can be used to obtain a characterisation of the \( i \)-th component of all local minima in terms of the other components of the vector. If this is successively done for all components in an elimination-like scheme, it results in a global minimum. If all successive derivatives of \( f \) and its transformations during the elimination process depend only on a bounded number \( k \) of variables, this algorithm can be shown to run in polynomial time of the input size and \( 2^k \) (see [17]). However, in general the execution time is exponential in the input size.

An alternative practical approach is to exploit the fact that the optimisation of a pseudo-Boolean function can be reduced in polynomial time to the optimisation of a quadratic pseudo-Boolean function [33] and design heuristics for the minimisation of this subtype of pseudo-Boolean functions.
Algorithm 1 The DDT heuristic for pseudo-Boolean optimisation

**Input:** Quadratic posiform $\phi(x) = \sum_{T \subseteq L} a_T \prod_{u \in T} u$.

**Output:** Approximation $\tilde{x}$ of a minimum of $\phi$.

1. $S \leftarrow \emptyset$, $\tilde{\phi} \leftarrow \phi$
2. *[Devour]* Find term $T$ with largest coefficient $a_T$ and set $S = S \cup \{T\}$.
3. *[Digest]* Draw all logical conclusions $C$ from the Boolean equation $\bigvee_{T \in S} T(x) = 0$. (2.5)
4. *[Tidy up]* Substitute the consequences $C$ into $\tilde{\phi}$ and simplify the result.
5. **if** $\tilde{\phi} \equiv \text{const}$ **then**
6. **return** a solution $\tilde{x}$ of equation (2.5).
7. **else**
8. go to step 2.
9. **end if**
One widely employed heuristic based on this fact is the DDT heuristic ("devour, digest, tidy up"), which is a greedy algorithm successively restricting the Boolean cube to smaller subcubes in which the terms with the highest remaining coefficient vanish [12]. It is described in Algorithm 1.

Due to the special form of the quadratic Boolean equation (2.5), the conclusions can be determined in polynomial time [12]. However, while reported to work well in practice, this heuristic does not provide strong guarantees about the quality of the solution it determines. In particular, we show that it does not always approximate the optimum by a constant factor.

**Definition 2.3.** Let $P$ a minimisation problem and denote by $P(i)$ the optimum of problem instance $i$. A heuristic $H$ for $P$ is called a constant-factor approximation algorithm for $P$ if there exists an $\varepsilon \geq 0$ such that $\frac{H(i)}{P(i)} \leq 1 + \varepsilon$ for all problem instances $i$.

**Theorem 2.4.** The DDT heuristic is not a constant-factor approximation algorithm for the pseudo-Boolean minimisation problem $PBO$.

**Proof.** We will prove the statement by explicitly constructing a family of problem instances for which the DDT heuristic fails to provide a constant-factor approximation. Specifically, for any dimension $n \geq 3$, define the following problem instance in the variables $x_1, \ldots, x_n$:

$$f(x_1, \ldots, x_n) = (a + 1)x_1 + \sum_{1<i\leq n} a x_i + \sum_{1<j\leq n} bx_1 x_j$$

(2.6)

for positive integers $a, b$ satisfying the conditions

$$a > b$$

(2.7)

and

$$a + 1 < (n - 1)b.$$  

(2.8)

Note that inequalities (2.7) and (2.8) are not contradictory, for example, $a = 4, b = 3$ is a valid choice for all $n \geq 3$.

The DDT heuristic, applied to this posiform, will take the following steps:

(i) Select literal $\overline{x_1}$, set $C := \{x_1 = 1\}$ and $\bar{\phi} := \sum_{1<i\leq n} a \overline{x_i} + \sum_{1<j\leq n} bx_j$.

(ii) This step is repeated $n - 2$ times for $k = 2, \ldots, n - 1$: Select literal $x_k$, set $C := \{x_1 = 1, \ldots, x_{k-1} = 1\}$ and $\bar{\phi} := \sum_{k<i\leq n} a \overline{x_i} + \sum_{k<j\leq n} bx_j + (k-1)b$.

(iii) Now, $\bar{\phi}$ is simplified to $ax_n + bx_n + (n - 2)b$. Select literal $x_n$, set $C := \{x_1 = \cdots = x_{n-1} = 1\}$ and $\bar{\phi} := (n - 1)b$, which is a constant expression in the $x_i$. 


(iv) Output the solution $x = (1, \ldots, 1)$ with $f(x) = (n - 1)b$.

However, $f$ does not have a minimum at $x = (1, \ldots, 1)$ for any $n \geq 3$. To see this, note that expression (2.6) can be characterised in terms of the Hamming weight $w_h(\cdot)$ of the vector $(x_2, \ldots, x_n)$: We have

$$f(x_1, \ldots, x_n) = \begin{cases} (a + 1) + (n - 1 - w_h(x_2, \ldots, x_n)) \cdot a & \text{if } x_1 = 0, \\ (n - 1 - w_h(x_2, \ldots, x_n)) \cdot a + w_h(x_2, \ldots, x_n) \cdot b & \text{if } x_1 = 1. \end{cases} \quad (2.9)$$

Suppose first that $x_1 = 0$. Abbreviating the weight of $(x_2, \ldots, x_n)$ by $w$, equation (2.9) becomes $g(w) = a + 1 + (n - 1 - w)a$ and since

$$\min_{(x_2, \ldots, x_n) \in \mathbb{B}^{n-1}} f(0, x_2, \ldots, x_n) = \min_{w \in \{0, \ldots, n-1\}} g(w),$$

the minimum of $f(x)|_{x_1=0}$ is given by

$$\arg \min_{w \in \{0, \ldots, n-1\}} g(w) = \arg \min_{w \in \{0, \ldots, n-1\}} -aw + na + 1 = n - 1,$$

and we have $g(n - 1) = f(0, 1, \ldots, 1) = a + 1$ for all $n$.

If, on the other hand, $x_1 = 1$, then equation (2.9) becomes $h(w) = (n - 1 - w)a + wb$ and the minimum of $f(x)|_{x_1=1}$ is given by

$$\arg \min_{w \in \{1, \ldots, n-1\}} h(w) = \arg \min_{w \in \{1, \ldots, n-1\}} (b - a)w + (n - 1)a = n - 1,$$

since $b - a < 0$ because of condition (2.7). We have $h(n - 1) = f(1, 1, \ldots, 1) = (n - 1)b$, which is strictly greater than $a + 1$ per assumption (2.8). Consequently, $f$ attains its minimum value $a + 1$ at $x = (0, 1, \ldots, 1)$. Furthermore, this minimum is unique since there is only one binary vector of length $n - 1$ with Hamming weight $n - 1$.

Comparing the output of the DDT heuristic with the actual minimum, we find that

$$\frac{\text{DDT}(f)}{\text{PBO}(f)} = \frac{b(n - 1)}{a + 1},$$

which is not a constant since it still depends on the problem size. This proves the claim. □
This result does of course not imply any restrictions regarding the practical usefulness of the DDT heuristic. Experimental results for the application of the DDT heuristic to pseudo-Boolean models of state and key recovery for the stream cipher MICKEY are given in Section 4.6.

3 Nonsmooth analysis and optimisation algorithms

Approaches to converting systems of equations corresponding to cryptanalytic problems such as recovering a previous internal state or the key to the reals and subjecting them to numerical optimisation have so far been focusing on polynomial representations [11, 32]. The success of this approach has been shown to significantly rely on both the overall dimension and the individual degree of the expressions not becoming prohibitively high [32]. While degree can be traded for dimensionality and vice versa, especially the models for full-scale algorithms turn out to be too big for most solvers.

3.1 A nonsmooth model of Boolean equations

An alternative approach to reducing the problems associated with the degree is to choose another real-valued representation that inherently avoids higher degrees. We propose the following representation providing this property:

\begin{align}
0, 1 \in \mathbb{F}_2 &\mapsto 0, 1 \in \mathbb{R} \\
\bar{a} &\mapsto 1 - a, \\
a \land b &\mapsto \min\{a, b\}, \\
a \lor b &\mapsto \max\{a, b\}, \\
a \oplus b &\mapsto \max\{a, b\} - \min\{a, b\} = |a - b|.
\end{align}

(3.1) (3.2) (3.3) (3.4) (3.5)

It is easy to see that for $a, b \in \mathbb{R}$, the roots of the Boolean expressions expressions correspond to the roots of their real-valued counterparts. However, while this correspondence also holds vice versa in the case of $\bar{\cdot}, \land$ and $\lor$ (e.g. $\min\{a, b\} = 0$ if and only if $a$ or $b$ are zero), $|a - b| = 0$ holds for any $a = b \in \mathbb{R}$, so specifically also for all values in the unit interval. This implies that while any solution in the Boolean domain is also a solution over the reals, there does not necessarily have to be a Boolean counterpart to all solutions of the real-valued model.

We refer to the maximum number of nested nonsmooth operations $\min, \max$ and $|\cdot|$ in an expression as its “nonsmooth degree”, or just “degree” if the context
is clear. For example, the nonsmooth degree of the expression \( x \) is zero, and 
\[
\max\{1 - x, |x - y|\}
\] has degree two.

### 3.2 Minimising nonsmooth functions: The Bundle method

Naturally, this representation is unsuitable for most numerical solvers, since they expect differentiable functions to calculate descent directions. However, nonsmooth analysis is a branch of calculus dealing with this type of functions. The crucial observation is that those functions and compositions of them are still locally Lipschitz-continuous:

**Definition 3.1 (Local Lipschitz continuity).** A function \( f : \mathbb{R}^n \to \mathbb{R}^m \) is called *locally Lipschitz-continuous* in a point \( x \) if there exist \( L > 0 \) and \( \delta > 0 \) such that 
\[
\|f(y) - f(x)\| \leq L\|y - x\| \quad \forall y \in B_\delta(x),
\]
where \( B_\delta(x) := \{ y \in \mathbb{R}^n : \|y - x\| < \delta \} \) denotes the open ball of radius \( \delta \) centered at \( x \).

Informally, this means that the slope of all secants of the function’s graph in some neighbourhood of \( x \) is bounded by the *Lipschitz constant* \( L \) (which may depend on \( x \)); \( f \) cannot change arbitrarily fast around \( x \). This property enables local line-search based techniques.

Convex functions are a particularly convenient subclass of locally Lipschitz-continuous functions:

**Definition 3.2 (Convex function).** A function \( f : X \subseteq \mathbb{R}^n \to \mathbb{R} \) is *convex* in \( X \) if and only if \( X \) is a convex set and for all \( x, y \in X \) and all \( \lambda \in [0, 1] \), 
\[
f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).
\]

Intuitively, convex functions have the property that any line segment connecting two points on its graph lies on or above the graph. It follows from the definition that a local minimum of a convex functions is automatically also a global minimum.

For the sake of a clearer exposition, we will first deal with the convex case, but the developed concepts readily carry over to the Lipschitzian case. The only major differences concern guarantees regarding global optimisation and, sometimes, the rate of convergence. An outline of how to adapt the algorithms to the nonconvex case can be found in Section 3.3.

A second prerequisite for a nonsmooth optimisation algorithm is a characterisation of descent directions.
**Definition 3.3** (Direction of descent). Let \( f : \mathbb{R}^n \rightarrow \mathbb{R} \). A direction \( s \in \mathbb{R}^n \backslash \{0\} \) is called a direction of descent (or descent direction) of \( f \) at point \( x \) if the directional derivative of \( f \) in \( x \) exists and is negative:

\[
f'(x, s) := \lim_{\lambda \to 0^+} \frac{f(x + \lambda s) - f(x)}{\lambda} < 0.
\]

A steepest descent direction of \( f \) at point \( x \) is a direction \( s \) such that

\[
f'(x, \frac{s}{\|s\|}) = \min_{\|t\|=1} f'(x, t).
\]

If \( f \) is differentiable in \( x \), then

\[
s := -\lambda \cdot \nabla f(x)
\]
gives a direction of steepest descent. In nonsmooth analysis, the role of the gradient is replaced by the notion of subgradients:

**Definition 3.4** (Subgradient, subdifferential). Let \( X \subseteq \mathbb{R}^n \) and \( f : X \rightarrow \mathbb{R} \) convex. The vector \( g \in \mathbb{R}^n \) is called a subgradient of \( f \) in \( x \) if

\[
f(y) - f(x) \geq g^T(y - x) \quad \forall y \in X.
\]

The subdifferential \( \partial f(x) \) is defined as the set of all subgradients of \( f \) at point \( x \).

However, subgradients cannot directly be used to calculate descent directions. In contrast to the smooth case, \(-g\) for \( g \in \partial f(x) \) is not necessarily a direction of descent.

An intuitive way of using the subdifferential for optimisation is the following: Consider the convex minimisation problem \( \min_{x \in X} f(x) \) with \( X \subseteq \mathbb{R}^n \) a compact set. Given pairs \((x^k, g^k)\) with \( x^k \in X \) and \( g^k \in \partial f(x^k) \), observe that the linear mapping

\[
l_k(x) := f(x^k) + g^k^T(x - x^k)
\]

is a minorant of \( f \) for each \( k \). Consequently, an underestimation of \( f \) is given by the piecewise linear model

\[
F_k^{\text{cp}}(x) := \max_{0 \leq j \leq k} l_j(x),
\]

called the cutting plane model of \( f \). By determining the next point of iteration \( x^{k+1} \) as the minimum of \( F_k^{\text{cp}}(x) \) subject to \( x \in X \), this model is incrementally refined by inclusion of another linear minorant \( l_{k+1} \) given by the subgradient \( g^{k+1} \).
at $x^{k+1}$. The gap between $f$ and $l_k$ is called the linearisation error and denoted $\alpha_k$. By the extreme value and Heine-Borel theorems [35], the minimisation of $F^\text{cp}_k(x)$ is well-defined. This idea is illustrated in Figure 1.

There are two main issues with this method. First, it only applies to compact sets $X$ which means that it cannot be used for unconstrained minimisation. Second, it can require a very large number of iterations which also causes an intolerable growth of the size of the cutting plane model. An example of this is provided in [24], where a minimisation problem over the unit ball in only 25 variables with nonsmooth degree 2 is shown to require at least $2^{59}$ iterations of the cutting plane method (and, accordingly, a model of $2^{59}$ planes).

The first issue can be resolved by adding a quadratic regularisation term and determining the next point of iteration by minimising $F^\text{cp}_k(x) + \frac{1}{\gamma_k} \|x - x^k\|^2$ instead, where $\gamma_k > 0$ is a parameter that can be adaptively chosen during the iteration. The second issue can be addressed by removing old and potentially irrelevant parts of the cutting plane model. This leads to the so-called bundle idea [30].

The idea of the Bundle method can be interpreted in two alternative ways: First, as a refinement of the cutting plane method which dynamically adapts the model to the progress of the iteration: If the improvement suggested by the next step is too
small, a “null step” is made to improve the model by sampling more subgradient information before proceeding; otherwise, a “serious step” is performed in which old and potentially irrelevant parts of the cutting plane model are removed. This ensures that the size of the model does not increase arbitrarily.

Its second – dual – interpretation, views the set of all convex combinations of the subgradients in the bundle as an approximation of the so-called \( \varepsilon \)-subdifferential:

**Definition 3.5** (\( \varepsilon \)-subgradient, \( \varepsilon \)-subdifferential). Let \( f : \mathbb{R}^n \to \mathbb{R} \) convex and \( \varepsilon \geq 0 \). A vector \( g \in \mathbb{R}^n \) is called \( \varepsilon \)-subgradient of \( f \) in \( x \) if

\[
f(y) - f(x) \geq g^T(y - x) - \varepsilon \quad \forall y \in \mathbb{R}^n.
\]

The \( \varepsilon \)-subdifferential \( \partial_\varepsilon f(x) \) is the set of all subgradients of \( f \) in \( x \).

The advantage of the \( \varepsilon \)-subdifferential in a point \( x_0 \) is that it contains information about subdifferentials in the neighbourhood of \( x_0 \) in the sense that for any \( \varepsilon > 0 \), there exists a \( \delta > 0 \) such that

\[
\bigcup_{y \in B_\delta(x_0)} \partial f(y) \subset \partial_\varepsilon f(x_0).
\]

Furthermore, it is easy to see that stationary points with respect to the \( \varepsilon \)-subdifferential are necessarily \( \varepsilon \)-optimal:

**Fact 3.6.** For a convex function \( f : \mathbb{R}^n \to \mathbb{R} \) and \( \varepsilon \geq 0 \), the following statements are equivalent:

1. \( \bar{x} \) is \( \varepsilon \)-optimal, i.e. \( \forall x. f(\bar{x}) \leq f(x) + \varepsilon \).
2. \( 0 \in \partial_\varepsilon f(\bar{x}) \).

Let \( P_X : x \mapsto \arg\min_{y \in X} \|y - x\| \) denote the orthogonal projection onto a convex closed set \( X \subseteq \mathbb{R}^n \). It can be shown that the negative of the orthogonal projection of the origin onto \( \partial_\varepsilon f(x) \) is a direction of steepest descent for all non-\( \varepsilon \)-optimal points [24]. The main problem with exploiting this, however, is that \( \partial_\varepsilon f(x) \) as a whole (in contrast to finding a single subgradient) and therefore also \(-P_{\partial_\varepsilon f(x)}(0)\) are difficult to determine. The Bundle method therefore approximates \( \partial_\varepsilon f(x) \) by the convex hull of a bundle of subgradients to determine directions of descent, see Figure 2.

A simplified version of the Bundle method for the convex real-valued case is described in Algorithm 2. The parameter \( \gamma \) combines the Bundle method with ideas from trust-region algorithms [34] by specifying a trust region radius for the cutting plane subproblem (3.8). The version presented here does put relatively
little effort into fine-tuning the trust region. In [34], for instance, an extensive inner iteration is performed to assess the trust region. The parameter $\eta \in (0, 1)$ specifies how big the improvement of the objective function compared to the improvement predicted by the cutting plane model has to be in order to carry out a serious step. Finally, $\varepsilon$ specifies an acceptable (additive) defect for the approximation of the minimum.

We briefly outline the operation of Algorithm 2. Throughout the description, vectors are indicated with Latin, and real numbers with Greek characters. The algorithm keeps a bundle of points $y_j$ and corresponding subgradients $g_j$ for $j \in J_k$. In each step, first the trust region radius $\gamma_k$ is assessed and a new step vector $s$ is determined by solving subproblem (3.8). This subproblem is a simple reformulation of the nonsmooth problem of minimising (3.7): Instead of minimising a piecewise linear max-type function, we find a simultaneous minimum upper bound for all the linear functions. Due to the regularisation term $\|s\|^2$, this is a quadratic optimisation problem with linear inequality constraints, which can be efficiently solved, for instance with the active set strategy [28].

Then, the stop criterion is evaluated: If the previous step and the sum of the linearisation errors with respect to the cutting planes with active Lagrange multipliers ($\lambda_j^k > 0$) was small, we have reached an $\varepsilon$-optimal point and stop. If this is not the case, the algorithm compares the actual improvement of the objective function to the prediction by the cutting plane model. If the improvement is significant enough, the iteration moves forward and prunes the bundle by only keeping those points $y_j$ and subgradients $g_j$ with nonzero Lagrange multipliers (i.e. those cutting planes that support $f$ from below). Otherwise, more subgradient information is sampled around the current point of iteration.
Algorithm 2: The Bundle method for nonsmooth optimisation (convex case).

**Input:** \( f : \mathbb{R}^n \to \mathbb{R} \) convex.

**Output:** Approximation \( \hat{x} \) of a minimum of \( f \).

1: Choose starting point \( x^0 \) and parameters \( \gamma^+ \geq \gamma^- > 0, \eta \in (0, 1) \) and \( \varepsilon \geq 0 \).
2: Determine \( g^0 \in \partial f(x^0) \) and set \( y^0 \leftarrow x^0, \alpha_0 \leftarrow 0, J_0 \leftarrow \{0\} \).
3: for \( k = 0, 1, \ldots \) do
4: Choose \( \gamma_k \in [\gamma^-, \gamma^+] \) and determine a KKT tuple \((s^k, \xi^k, \lambda^k)\) of the problem

\[
\begin{align*}
\min_{s, \xi} \quad & \xi + \frac{1}{2\gamma_k} \|s\|^2 \\
\text{s.t.} \quad & g^j^T s - \alpha^k_j \leq \xi \quad (\forall j \in J_k)
\end{align*}
\] (3.8)
5: Set \( v^k \leftarrow -\frac{1}{\gamma_k} s^k, \varepsilon_k \leftarrow \sum_{j \in J_k} \lambda^k_j \alpha^k_j \).
6: if \( \|v^k\| \leq \varepsilon \) and \( \varepsilon_k \leq \varepsilon \) then
7: return \( \hat{x} \leftarrow x^k \)
8: end if
9: if \( f(x^k + s^k) - f(x^k) \leq \eta \xi^k \) then \( \triangleright \) serious step
10: Set
\[
\begin{align*}
y^{k+1} & \leftarrow x^k + s^k \\
x^{k+1} & \leftarrow x^k + s^k \\
J_{k+1} & \leftarrow \{ j \in J_k \mid \lambda^k_j > 0 \} \cup \{ k + 1 \}
\end{align*}
\]
11: else \( \triangleright \) null step
12: Set
\[
\begin{align*}
y^{k+1} & \leftarrow x^k + s^k \\
x^{k+1} & \leftarrow x^k \\
J_{k+1} & \leftarrow \{ j \in J_k \mid \lambda^k_j > 0 \} \cup \{ j \mid y^j = x^k \} \cup \{ k + 1 \}
\end{align*}
\]
13: end if
14: Calculate
\[
\begin{align*}
f^{k+1} & \leftarrow f(y^{k+1}) \\
g^{k+1} & \in \partial f(y^{k+1}) \\
\alpha_j^{k+1} & \leftarrow f(x^k) - l_j(x^k) \quad \forall j \in J_{k+1}.
\end{align*}
\]
15: end for
For convex objective functions, this method converges globally. If \( f \) is not convex, a few modifications to the algorithm are necessary, and additional assumptions (i.e. requiring \( f \) to be semi-smooth) are required for a global convergence result [34]. This is detailed in the following section.

3.3 Dealing with nonconvex objective functions

The main advantage in convex (and dually, concave) optimisation is that local optima are global optima by definition. If the objective function is merely locally Lipschitz-continuous, a more general notion of subdifferential is required. While there exist many proposals for subdifferentials of locally Lipschitz-continuous functions, the Clarke subdifferential [15] is most widely used:

**Definition 3.7** (Clarke subdifferential). Let \( f : X \subset \mathbb{R}^n \to \mathbb{R} \) be locally Lipschitz-continuous and denote by \( X_d \subset X \) the set of points where \( f \) is differentiable. The **Clarke subdifferential** of \( f \) at point \( x \in X \) is the set

\[
\partial \operatorname{cl} f (x) := \operatorname{conv} \{ v \in \mathbb{R}^n \mid \exists (x_k) \text{ with } x_k \in X_d \text{ and } x_k \to x, \nabla f(x_k) \to v \text{ for } k \to \infty \}.
\]

For instance, the Clarke subdifferential of the absolute value function \( f(x) = |x| \) at \( x = 0 \) is \( \partial \operatorname{cl} f (0) = \operatorname{conv} \{-1, 1\} = [-1, 1] \). Another commonly used notion of subdifferential is the Demyanov-Rubinov quasidifferential [19].

Unlike the convex case, \( 0 \in \partial \operatorname{cl} f (\bar{x}) \) is only a necessary, not a sufficient condition for \( \bar{x} \) to be a local or global minimizer of \( f \). The same holds for maximizers. Consequently, the algorithms can only search for stationary points that are not guaranteed to be even locally optimal.

When applying the bundle idea to nonconvex nonsmooth optimisation, one major issue arises: The cutting plane model (3.7) is not any more an underestimate for \( f \), the linearisation error \( \alpha^j_k \) can become very small or even negative, which is especially problematic for very distant trial points \( y^j \) because the corresponding subgradients do not improve the model. A possible solution to this problem is to replace the linearisation error \( \alpha^j_k \) by the so-called subgradient locality measure [34]:

\[
\beta^k_j := \max \{ a^k_j, \mu \| x^k - y^j \|^2 \} \quad (3.9)
\]

with \( \mu \geq 0 \) parametrising the distance measure. Alternatively, the cutting plane model can be split in an under- and an overestimation of \( f \). The next point in the bundle is then selected in a way that both under- and overestimates predict a significant improvement. This has recently been proposed by [20].
Due to Rademacher’s famous theorem [9], a locally Lipschitz-continuous function $f$ is differentiable almost everywhere. This implies that the set of points where $f$ is not smooth has (Lebesgue) measure zero. If, additionally, $\partial^{cl} f(x)$ is a point-based approximation of $f$, that is, if

$$\sup_{v \in \partial^{cl} f(x+s)} \|f(x+s) - f(x) - v^T s\| = o(\|s\|) \quad \text{for } s \to 0,$$

$f$ is called *semi-smooth* and the rate of convergence is comparable to the convex case [9, 34]. It is worth noting that this only applies to the rate of convergence towards a stationary point which does not necessarily have to be a global minimum.

The expressions used in our nonsmooth model of Boolean equations given by equations (3.1)–(3.5) and arbitrary compositions of them are semi-smooth.

4 Applying techniques from nonsmooth optimisation to MICKEY

4.1 The stream cipher MICKEY

The stream cipher MICKEY is a hardware-oriented stream cipher designed by Babbage and Dodd with a security level of 80 bits [2]. It was submitted to the eSTREAM stream cipher competition and selected as one of three hardware-oriented algorithms in the final portfolio. While its first version could be attacked by a time-memory tradeoff with online complexity smaller than exhaustive key search (albeit more expensive precomputation, see [25]), no cryptanalysis results have been published for the current version 2.0.

MICKEY consists of two registers $R$ and $S$ of 100 bits each, $R$ being a linear feedback shift register with maximum period, and $S$ being a nonlinear feedback shift register. A cell $s^{(t)}_i$ of $S$ at time $t$ is updated by the rule

$$s^{(t+1)}_i = s^{(t)}_{i-1} \oplus (f^{(t)} \land F^c_s) \oplus \left( (C^0_i \oplus s^{(t)}_i) \land (C^1_i \oplus s^{(t)}_{i+1}) \right),$$

where $f^{(t)}$ is the feedback bit, $(F^{0,1})_i$ are constant binary sequences, and the complementation of $s_i$ and $s_{i+1}$ depends on the value of the constant binary sequences $(C^0)_i$ and $(C^1)_i$, respectively.

The clocking of both registers is done in a way that depends on bits from both registers. More specifically, if the control bit $c_R := s_{34} \oplus r_{67}$ is equal to one, the linear register $R$ is clocked $2^{50} - 157$ times ahead by XOR-ing each bit back into the current stage while shifting. Otherwise, $R$ is clocked normally. Analogously, the control bit $c_S := s_{67} \oplus r_{33}$ determines whether the sequence $F^0$ or $F^1$ is used in the update of $S$.  

During the setup phase, $R$ and $S$ are initialised with all zeros, afterwards the IV and the key are clocked in bit by bit, followed by 100 steps of preclocking with the input bit set to zero. The input procedure consists of XOR-ing the input bit to the feedback bits of both registers. Additionally, the cell $s_{50}$ of the $S$ register is XOR-ed into the feedback of the $R$ register. After the setup phase, keystream bits are generated by first outputting the XOR of the leftmost bits $s_0$ and $r_0$ of both registers and then clocking both registers (without the mixing of $s_{50}$ into the feedback of $R$).

4.2 A scalable variant

For cryptanalysis purposes, it is often useful to work with reduced or weakened versions of the algorithm in consideration. This can also provide a useful indication about the security margin of the full algorithm. Besides the obvious way of weakening MICKEY by reducing the number of steps during the preclocking, we have defined a scalable variant of MICKEY called Mini-MICKEY where both the key length $k$ and the register size $n$ can be varied independently. The structure of Mini-MICKEY has been chosen to resemble that of MICKEY as closely as possible.

The size of each register of Mini-MICKEY can be varied according to $4 \leq n \leq 100$; for the key length we require $4 \leq k \leq \min\{80, 2n\}$ bits, but $k$ can be arbitrary in principle. Mini-MICKEY’s $S$ register is a truncated version of MICKEY’s $S$ register, that is, the sequences $(C^0)_i, (C^1)_i$ and $(F^0)_i, (F^1)_i$ are simply restricted to their first $n$ elements. The LFSR $R$ is defined by the first primitive polynomial of degree $n$ in Table D in chapter 10 of [27], with one exception: if $n = 100$, MICKEY’s original feedback polynomial is used. The control bits for the irregular mutual clocking are defined by

$$c_R := s_{\lfloor n/3 \rfloor} \oplus r_{\lfloor 2n/3 \rfloor}$$  \hspace{1cm} \text{(4.1)}

$$c_S := s_{\lceil 2n/3 \rceil} \oplus r_{\lceil n/3 \rceil}.$$  \hspace{1cm} \text{(4.2)}

As in MICKEY, keystream bits are generated by the output function $s_0 \oplus r_0$. The mixing during the setup phase is done by XOR-ing $s_{\lceil n/2 \rceil}$ into the feedback bit of $R$. After the loading of the IV and the key, a preclocking of $n$ steps is applied. A general overview of the structure of Mini-MICKEY is given in Figure 3.

It should be noted that this does not exactly yield the full MICKEY when parameters $k = 80$ and $n = 100$ are selected. The differences lie in the selection of indices for the control bits: In (4.1) and (4.2), $s_{\lfloor n/3 \rfloor}, s_{\lceil 2n/3 \rceil}$ and $r_{\lceil 2n/3 \rceil}$ would have to be used instead of their floor function variants to match the indices used in the full MICKEY. However, the use of the ceiling function would imply that in the
important small cases $n = 4, 5$, the control bits would depend on the first and the feedback bit, in contrast to MICKEY’s design choice to determine the control bits as the XOR of “inner” register cells. On the other hand, we verified that selecting bits 33 and 66 instead of 34 and 67 in this variant with $n = 100$ does not cause Mini-MICKEY to behave numerically different than the original MICKEY.

### 4.3 Equations representing Mini-MICKEY

The state update in both of MICKEY’s registers is usually dense in the sense that in each clocking step, potentially all bits of both registers can require an update. In this case, the LFSR performs the “jumping” operation described in Section 4.1. Due to the mutual clocking, this occurs depending on bits from both the linear and the nonlinear register. Consequently, about $2nc$ new equations are generated for $c$ clockings.

As an illustration, using the standard nonsmooth representation according to equations (3.1)–(3.5), the following equations are generated for Mini-MICKEY with $n = k = 4$:

**First clock:**

\[
\begin{align*}
    r_{0,1,3}^{(1)} &= k_0 \\
    r_2 &= 0 \\
    s_{0,1,2,3}^{(1)} &= k_0
\end{align*}
\]
\(i\)-th clock (\(i > 1\)):

\[
\begin{align*}
  c_R^{(i)} &= |s_1^{(i)} - r_2^{(i)}| \\
  c_S^{(i)} &= |s_2^{(i)} - r_1^{(i)}| \\
  r_0^{(i+1)} &= |r_3^{(i)} - k_i - s_2^{(i)}| - \min \left\{ c_R^{(i)}, r_0^{(i)} \right\} \\
  r_1^{(i+1)} &= |r_0^{(i)} - r_3^{(i)} - k_i - s_2^{(i)}| - \min \left\{ c_R^{(i)}, r_1^{(i)} \right\} \\
  r_2^{(i+1)} &= r_1^{(i)} - \min \left\{ c_R^{(i)}, r_2^{(i)} \right\} \\
  r_3^{(i+1)} &= |r_2^{(i)} - r_3^{(i)} - k_i - s_2^{(i)}| - \min \left\{ c_R^{(i)}, r_3^{(i)} \right\} \\
  s_0^{(i+1)} &= \min \left\{ |s_3^{(i)} - k_i|, \max \left\{ \min \left\{ c_S^{(i)}, F_1^{(i)} \right\}, \min \left\{ 1 - c_S^{(i)}, F_0^{(i)} \right\} \right\} \right\} \\
  s_1^{(i+1)} &= |s_0^{(i)} - \min \left\{ |s_1^{(i)} - C_1^{(i)}|, |s_2^{(i)} - C_1^{(i)}| \right\} | - \min \left\{ |s_3^{(i)} - k_i|, \max \left\{ \min \left\{ c_S^{(i)}, F_1^{(i)} \right\}, \min \left\{ 1 - c_S^{(i)}, F_1^{(i)} \right\} \right\} \right\} \\
  s_2^{(i+1)} &= |s_1^{(i)} - \min \left\{ |s_2^{(i)} - C_2^{(i)}|, |s_3^{(i)} - C_2^{(i)}| \right\} | - \min \left\{ |s_3^{(i)} - k_i|, \max \left\{ \min \left\{ c_S^{(i)}, F_2^{(i)} \right\}, \min \left\{ 1 - c_S^{(i)}, F_2^{(i)} \right\} \right\} \right\} \\
  s_3^{(i+1)} &= |s_2^{(i)} - \min \left\{ |s_3^{(i)} - k_i|, \max \left\{ \min \left\{ c_S^{(i)}, F_3^{(i)} \right\}, \min \left\{ 1 - c_S^{(i)}, F_3^{(i)} \right\} \right\} \right\} |
\end{align*}
\]

Here, the parts of the register update depending on the control bits (“if \(c_s\) then \(A\), else \(B\)”) have to be expressed by \((c_s \land A) \lor (\neg c_s \land B)\). For equations not covering the key setup phase, the mixing of \(s_2\) into \(R\) and the XOR-ing of the key bits is omitted. Otherwise, the equations are structurally equivalent.

Due to the local updating behaviour of feedback shift registers, the nonsmooth degree of the individual equations ranges between 1 and 5. In particular, this local updating behaviour implies that the degree does not depend on the register length \(n\). By substitution, the number of new equations per clocking can be reduced, thereby trading lower dimensionality for higher degree.

**Properties of the Mini-MICKEY equations**

By direct calculation, one can verify that the MICKEY equations are not convex. However, as instantiations of the model described by equations (3.1)–(3.5), they are semi-smooth (see also Section 3.3). Consequently, it is to be expected
that while the optimisation algorithms will quite efficiently converge to stationary points, they need not necessarily be global extrema.

Since most efficient algorithms for estimation of Lipschitz constants only work in the univariate case, we only estimate an upper bound for the Lipschitz constant of the nonsmooth expressions: Each individual expression in the model is Lipschitz-continuous with constant $L = 1$, so for an expression with nonsmooth degree $d$, the Lipschitz constant is approximated by $L' = 2d$.

4.4 Attack scenarios

We considered the following attack scenarios, with the main focus being on the first and third scenario. All scenarios are in a known-plaintext setting, where a sequence of keystream bits can be determined by XOR-ing the known plaintext to the ciphertext.

Scenario 1: Recovering a previous state

In this scenario, given a sequence of keystream bits, the task is to recover the internal state of the stream cipher before the generation of this keystream. This should be infeasible for a secure stream cipher, since knowledge of the full internal state at some point allows the attacker to decrypt all future ciphertexts that are encrypted with keystream derived from this internal state.

From the attacker’s point of view, state recovery has the advantage that the loading of the IV and the key can be neglected, which leads to simpler equations. The challenge is that this scenario implies higher dimension of the equation systems since we need to recover all $2^n$ state bits which is typically larger than the length of the key. At least $2^n$ bits of keystream are required to carry out this attack.

Scenario 2: Recovering the key

Probably the most obvious target, this scenario encompasses the reconstruction of the $k$-bit key that was clocked into the cipher during the setup phase from the knowledge of a sequence of keystream bits.

Since for most sensible variants of Mini-MICKEY, $k < 2n$, this has the advantage of a smaller problem dimension, but the disadvantage of higher equation complexity due to the loading of the IV and the key and the preclocking process. At least $k$ bits of keystream are required for this attack.
Scenario 3: Guess and determine the key

This scenario is a special case of the second where a guess for part of the key (say $k_0$ out of the $k$ bits) is used to simplify the system of equations representing the key loading, preclocking, and the generation of the first $k - k_0$ keystream bits. Since there are only $k - k_0$ unknowns left, the keystream bits are expressed in terms of minimised Boolean functions of the unknown key bits. If this number is sufficiently small, Boolean minimisation algorithms such as the Quine-McCluskey [29] or the Espresso method [13] can be applied to simplify the Boolean functions.

In this scenario, very few very complex equations have to be solved. Additionally, the minimisation process is specific to and therefore has to be repeated for each key guess. Note also that in this scenario, the overall time complexity can be estimated even if it would be impractical to carry out the attack in practice: If $t_0$ denotes the time complexity of the minimisation and solving process for one key guess, the total attack complexity is given by $2^{k_0} \cdot t_0$. The attack is faster than a generic attack if $t_0$ is less than the complexity of $k - k_0$ trial decryptions.

4.5 Experimental results

We have subjected the nonsmooth systems of equations corresponding to various versions of Mini-MICKEY and the full MICKEY in the three attack scenarios to different solving algorithms using various approaches to model or adapt the Boolean system of equations. The results are summarised in Tables 1 to 3. The experiments were carried out on a machine with two Intel Xeon dual-core 2.8 GHz processors and 2 GB of memory. Only one core was used, and memory usage was negligible.

Algorithms

The nonsmooth optimisation algorithms used in the experiments were a bundle method (DFBM), the cutting angle method (ECAM), and a dynamical-systems based method (DSO). All algorithms only require the objective function to be Lipschitz, so they are suitable for our nonconvex models of Boolean equations.

DFBM (“Derivative-Free Bundle Method”) is a variant of the Bundle algorithm developed by Bagirov and Ugon [6] which uses the method of [5] to approximate the subdifferential. This method yields a particularly efficient procedure for calculating subgradients, so that the running time of the algorithm is commonly dominated by solving the quadratic subproblem (3.8) instead [6]. Like all Bundle methods, it converges to local minima. In our experiments, it was combined with random start as a globalisation strategy.
The ECAM (“Extended Cutting Angle Method”) intends to improve on DFBM for global optimisation by using the Lipschitz constant of the objective function to estimate its smallest conceivable minimum. Then, a sawtooth-like underestimation of the objective function is constructed [7]. While this method is theoretically guaranteed to find an $\varepsilon$-optimal approximation of the global minimum, it has the disadvantage of typically requiring a large number of function evaluations, and being very sensitive to increases in the dimension of the problem.

The third algorithm, DSO (“Dynamical Systems Based Optimisation”), is a heuristic obtaining descent directions by balancing samples of the objective function according to a physical model of forces [31].

Those algorithms and combinations of them are implemented in the GANSO library for general nonsmooth optimisation developed at the University of Ballarat [4, 8].

### Choice of representation

In the case of polynomial models, the choice of representation can have decisive influence on the success of an experiment [32]. By contrast, this has little effect in the nonsmooth case, where the different nonsmooth representations corresponding to standard and Fourier representation and their respective duals all caused very comparable behaviour of the algorithms. Specifically, they all exhibit the property that there is no one-to-one correspondence of solutions to $a \oplus b = 0$ between Boolean values and the unit interval, frequently leading the solvers to suggest non-Boolean values as “improvements” over non-matching Boolean points of iteration.

We therefore focus on the standard representation given by equations (3.1)–(3.5).

### Choice of norm

While the choice of representation had little effect on the behaviour of the minimisation algorithms, the choice of norm has a much greater impact. For all of the algorithms in the GANSO library, the maximum norm turned out to be the best choice.

### Equation preprocessing

The success of the solvers can depend significantly on how an optimisation problem is formulated. We investigated the effect of some equation preprocessing techniques to adapt the system of equations to the solver.
Trading equation complexity for dimensionality. The straightforward method of generating equations representing the stream cipher clocking results in far more individual equations than variables we are concerned with in one particular attack scenario. In case of the nonsmooth optimisation algorithms we applied, it usually turned out to be preferable combining many equations by substitution to reduce the dimensionality of the problem. Since only the expression nesting level (but not the degree) is affected, the resulting increase in equation complexity is usually tolerable.

Experimentally, the optimal balance between dimensionality and equation complexity was reached by organizing the individual subexpressions according to a balanced tree structure such that a maximum substitution level of about five subequations is not exceeded.

Ensuring Booleanness of the solutions without affecting convergence. To mitigate the issue with non-Boolean solutions suggested by the solvers, two approaches were considered:

The first involves adding a penalisation $P_\alpha(x)$ to the objective function, which then becomes $\|f(x)\| + P_\alpha(x)$, introducing a penalty increasing with the distance to the corners of the hypercube $[0,1]^n$ specified by the box constraints. In particular, the following terms were considered, with $\alpha$ being a parameter to fine-tune the weight of the penalty term:

- a triangle-like function: $x \mapsto \alpha \cdot \left(\frac{1}{2} - |x - \frac{1}{2}|\right)$, \hfill (4.3)
- a quadratic penalty term: $x \mapsto \alpha \cdot \left(x - x^2\right)$, \hfill (4.4)
- an exponential bell curve: $x \mapsto \alpha \cdot \left(e^{-8 \cdot (2x-1)^2}\right)$. \hfill (4.5)

As with the choice of norms, the maximum, sum, or average of all component deviations from $\{0,1\}$ can be used in $P_\alpha(x)$. These penalty terms are illustrated in Figure 4.

All of the above can also be combined with multiplication by $x$ or square rooted, the latter resulting in a steeper penalisation. In most cases, a damped exponential or a plain triangle penalizer were yielding the best results. This approach has the obvious disadvantage of causing convergence issues if the penalty terms modify the notion of descent too much. Its main merit is that it does not increase the dimension of the problem.

The second approach is to introduce a second set of equations of the form $x^2 = x$, directly forcing the variables to be Boolean. This has the advantage of not modifying the objective function, so that convergence is not affected by the penalisation terms, but increases the dimension. Furthermore, those equations
introduce additional local optima which might hinder convergence to a global optimum. For small instances with a dimension of less than 20, this approach turned out to be superior to the penalty terms, which in turn start to yield better results for greater dimensions.

4.6 The DDT heuristic and MICKEY

In addition to the nonsmooth algorithms, we applied the DDT heuristic to the systems of equations of Mini-MICKEY. The implementation used was a simplified variant of DDT as implemented by [22]. The experimental data from Table 4 suggests that this approach is inferior to the nonsmooth algorithms. However, extensions to this implementation of DDT are expected to improve the results.

4.7 Discussion

Results

In the state recovery scenario, due the comparatively high dimension of this problem, only very small instances ($n = 6$, resulting in a 72-dimensional instance) could be fully solved. For higher values of $n$, all solvers failed. In most of those cases, non-Boolean coordinates were output which could not be mapped back to a Boolean solution by rounding or similar procedures. A closer inspection of the final points of iteration revealed that all of the solvers got stuck in local optima. It is worth noting that longer keys do not have any significant influence in this scenario since we attempt to recover the state after the key has been clocked in.

Concerning key recovery, the results were more promising. The maximum problem size that the nonsmooth algorithms could fully solve were $n = 20$ and a
26-bit key. In those cases, additional equations of the form $x^2 = x$ were required to ensure Booleanness, resulting in a total dimension of 52.

In both scenarios, the results are given for the maximum key length given a particular state size for which the problems could still be solved. In particular, this means that all instances involving the same state but smaller key sizes can be successfully solved.

In the case of combining key recovery with Boolean minimisation according to a partial key guess, both the instances involving a key guess of 72 and 64 bits for the full MICKEY could be solved. The DSO solver proved particularly suitable for these problem instances. The increase in nonsmooth degree per equation turned out to be comparatively small: 8 for the 72-bit guess and 11 for the 64-bit guess. This is a result of the good Boolean minimisation performed by the QMC and Espresso algorithms. Additionally, the success rate was improved at the cost of running time by preprocessing the equations such that the maximum degree of 5 per equation was not exceeded.

It should be noted that the success of the solver in those cases does not yield a valid attack on the full MICKEY. The whole process of generating and minimising the equations has to be repeated for each key guess, which is clearly inferior to brute force. It is an open issue to extend this to more useful values of the number of guessed key bits.

Notes on the nonsmooth optimisation algorithms

The experimental results provided several insights about the applicability of the different nonsmooth optimisation algorithms to this type of systems of equations.

Generally, it could be observed that the dynamic systems based method (DSO) was particularly efficient for instances of small dimension. This algorithm was far less sensitive to the complexity of the individual expressions, while increasing the dimension proved problematic. By contrast, the DFBM and ECAM algorithms could still successfully solve some instances with higher dimension. This is due to the fact that the equation systems in the state and (full) key recovery scenarios had higher dimension but relatively small nonsmooth degree, which particularly also bounded the Lipschitz constant to a small value.

The DDT heuristic for pseudo-Boolean programming turned out to be an algorithm with reliable running time polynomial in the problem dimension, however at the expense of guarantees about the quality of the solution. The Bundle and ECAM methods provided rather unreliable running times, but the quality of the solution improved with the number of iterations. Especially the ECAM method, however, can take a prohibitive number of steps before reaching an $\varepsilon$-optimal solution, which in the context of cryptanalysis is only then an advantage if the $\varepsilon$-
optimal solution can be mapped back to the sought Boolean solution.

5 Conclusion

Summarising, the concept of nonsmooth models of Boolean equations is a generic approach that can be used in the cryptanalysis of any algorithm based on arithmetic over $\mathbb{F}_2$, including most stream ciphers, block ciphers and hash functions. The approach aims at avoiding real-valued expressions of higher degree by using nondifferentiable but Lipschitz-continuous functions. The theory of and available numerical software for nonsmooth optimisation can then be used to attempt solving the corresponding systems of equations.

Applied to the stream cipher MICKEY from the eSTREAM final portfolio, this approach can recover the 80-bit key of the full algorithm in $2^{64}$ iterations if 16 bits of keystream are available. However, while this attack requires very little known plaintext, the total time complexity is inferior to trying all $2^{80}$ keys. Small-scale variants of MICKEY can be solved in practical time without guessing key bits.

With the exception of the guess-and-determine scenario, one main issue with this approach compared to statistical attacks is the lack of predictability beyond practical attacks: An attack can only be shown to work if it can actually be carried out in practice. Extrapolating results from small-scale experiments to the full algorithms is an unreliable measure since the number of required iterations for a global optimum could grow exponentially (ECAM), or the algorithms could converge to local extrema without ever reaching the solution corresponding to the key or the previous state (DSO, DFBM).

However, these techniques could be applied to the various equation-solving subproblems occurring in typical cryptanalytic attacks where ad hoc or guess-and-determine techniques are currently used, for instance within collision attacks on SHA-1 or SHA-2 [26]. Those subproblems typically have lower dimensionality than full ciphers or hash functions, and improving the time required to solve them would directly improve the full attack complexities. For instance, [1] demonstrates several methods of how equation-solving techniques over finite fields (using Gröbner bases and SAT solvers) can be used to enhance differential attacks on block ciphers.

Acknowledgments. I would like to thank Vincent Rijmen for numerous valuable exchanges of ideas and his advice and encouragement.
References


Nonsmooth Cryptanalysis


Received ???.

Author information

Elmar Tischhauser, Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC and IBBT, Kasteelpark Arenberg 10, 3001 Heverlee, Belgium.
E-mail: elmar.tischhauser@esat.kuleuven.be
<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>Solver</th>
<th>Preprocessing</th>
<th>Execution time</th>
<th>Function eval.</th>
<th>Solvable</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>DFBM –</td>
<td>triangle penalty</td>
<td>$\ll 1$ s</td>
<td>$2^9$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td>triangle penalty</td>
<td>$\ll 1$ s</td>
<td>$2^{10}$</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
<td>triangle penalty</td>
<td>$\ll 1$ s</td>
<td>$2^{11.5}$</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td>triangle penalty</td>
<td>15 s</td>
<td>$2^{13}$</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td></td>
<td>triangle penalty</td>
<td>15 s</td>
<td>$2^{13.2}$</td>
<td>yes</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td>triangle penalty</td>
<td>28 s</td>
<td>$2^{16}$</td>
<td>for specific initial values</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td>triangle penalty</td>
<td>149 s</td>
<td>$2^{17.5}$</td>
<td>for specific initial values</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
<td>triangle penalty</td>
<td>3 min</td>
<td>$2^{19}$</td>
<td>} no (no improvement for more iterations)</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
<td>exp. penalty</td>
<td>3 min</td>
<td>$2^{19}$</td>
<td>} yes</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td></td>
<td>triangle penalty</td>
<td>52 s</td>
<td>$2^{19}$</td>
<td>yes</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td>triangle penalty</td>
<td>7 min</td>
<td>$2^{17.6}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td>triangle penalty</td>
<td>12 min</td>
<td>$2^{19.5}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>ECAM –</td>
<td>–</td>
<td>2.2 s</td>
<td>$2^{10.5}$</td>
<td>unreliably (local optima)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td>triangle penalty</td>
<td>2.3 s</td>
<td>$2^{10.7}$</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
<td>triangle penalty</td>
<td>2.3 s</td>
<td>$2^{11}$</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td>triangle penalty</td>
<td>52 s</td>
<td>$2^{14.4}$</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td></td>
<td>triangle penalty</td>
<td>52 s</td>
<td>$2^{15}$</td>
<td>yes</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td>triangle penalty</td>
<td>7 min</td>
<td>$2^{17.6}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td>triangle penalty</td>
<td>12 min</td>
<td>$2^{19.5}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>DSO –</td>
<td>–</td>
<td>$\ll 1$ s</td>
<td>$2^{8}$</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
<td>–</td>
<td>$\ll 1$ s</td>
<td>$2^{9}$</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td>–</td>
<td>$\ll 1$ s</td>
<td>$2^{12.1}$</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td></td>
<td>–</td>
<td>$\ll 1$ s</td>
<td>$2^{12.7}$</td>
<td>yes</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td>–</td>
<td>17 s</td>
<td>$2^{15.4}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td>triangle penalty</td>
<td>17 s</td>
<td>$2^{16.2}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td>triangle penalty</td>
<td>17 s</td>
<td>$2^{17}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ balancing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. State recovery for $n$ bit registers and $k$ bit key. Dimension is $2^n$. 
<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>Solver</th>
<th>Preprocessing</th>
<th>Execution time</th>
<th>Function eval.</th>
<th>Solvable</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>DFBM</td>
<td>triangle penalty</td>
<td>$\ll 1$ s</td>
<td>$2^5$</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
<td>triangle penalty</td>
<td>$\ll 1$ s</td>
<td>$2^{8.2}$</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td>triangle penalty</td>
<td>6 s</td>
<td>$2^{8.4}$</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td></td>
<td>triangle penalty</td>
<td>9 min</td>
<td>$2^{13.1}$</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td></td>
<td>triangle penalty</td>
<td>16 min</td>
<td>$2^{21.3}$</td>
<td>yes</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td></td>
<td>triangle penalty</td>
<td>20 min</td>
<td>$2^{21.6}$</td>
<td>yes</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>exp. penalty</td>
<td>20 min</td>
<td>$2^{22.1}$</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>26</td>
<td>exp. penalty</td>
<td>43 min</td>
<td>$2^{28.4}$</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ $x^2 = x$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>27</td>
<td>exp. penalty</td>
<td>43 min</td>
<td>$2^{29}$</td>
<td>for specific initial values</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ $x^2 = x$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>ECAM</td>
<td>exp. penalty</td>
<td>2 s</td>
<td>$2^{5.7}$</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td>exp. penalty</td>
<td>15 s</td>
<td>$2^{10.8}$</td>
<td>yes</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td></td>
<td>exp. penalty</td>
<td>94 min</td>
<td>$2^{23.1}$</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ balancing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>22</td>
<td>exp. penalty</td>
<td>111 min</td>
<td>$2^{25.2}$</td>
<td>no (local optima)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ balancing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>DSO</td>
<td>–</td>
<td>$\ll 1$ s</td>
<td>$2^{4.1}$</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>–</td>
<td>$\ll 1$ s</td>
<td>$2^{8.4}$</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>–</td>
<td>14 min</td>
<td>$2^{21.9}$</td>
<td>no (not Boolean)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Key recovery for $n$ bit registers and $k$ bit key. Dimension is $k$. 
<table>
<thead>
<tr>
<th>$k_0$</th>
<th>Solver</th>
<th>Preprocessing</th>
<th>Execution time</th>
<th>Function eval.</th>
<th>Solvable</th>
</tr>
</thead>
<tbody>
<tr>
<td>76</td>
<td>DFBM</td>
<td>exp. penalty</td>
<td>26 s</td>
<td>$2^{5.4}$</td>
<td>yes</td>
</tr>
<tr>
<td>72</td>
<td></td>
<td>exp. penalty</td>
<td>13 min</td>
<td>$2^{9.8}$</td>
<td>yes</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td>exp. penalty</td>
<td>24 min</td>
<td>$2^{13.1}$</td>
<td>yes</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>exp. penalty</td>
<td>59 min</td>
<td>$2^{18}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>76</td>
<td>ECAM</td>
<td>exp. penalty</td>
<td>33 s</td>
<td>$2^{5.2}$</td>
<td>yes</td>
</tr>
<tr>
<td>72</td>
<td></td>
<td>exp. penalty</td>
<td>1 min</td>
<td>$2^{9.5}$</td>
<td>yes</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td>exp. penalty</td>
<td>7 min</td>
<td>$2^{12.8}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>76</td>
<td>DSO</td>
<td>–</td>
<td>17 s</td>
<td>$2^5$</td>
<td>yes</td>
</tr>
<tr>
<td>72</td>
<td></td>
<td>–</td>
<td>54 s</td>
<td>$2^{9.2}$</td>
<td>yes</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td>–</td>
<td>2 min</td>
<td>$2^{12.2}$</td>
<td>yes</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>–</td>
<td>4 min</td>
<td>$2^{17.3}$</td>
<td>yes</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>–</td>
<td>11 min</td>
<td>$2^{22.5}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>triangle penalty</td>
<td>11 min</td>
<td>$2^{23}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>exp. penalty</td>
<td>11 min</td>
<td>$2^{23.1}$</td>
<td>no (not Boolean)</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>exp. penalty</td>
<td>11 min</td>
<td>$2^{25.5}$</td>
<td>no (not Boolean)</td>
</tr>
</tbody>
</table>

Table 3. Guess $k_0$ out of 80 key bits and determine the remaining key bits for the full MICKEY ($n = 100$ bit registers and $k = 80$ bit key). Dimension is $80 - k_0$. Balancing was used in all experiments.
<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>Preprocessing</th>
<th>Execution time</th>
<th>Solvable</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>triangle penalty</td>
<td>$\ll 1$ s</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>Preprocessing</th>
<th>Execution time</th>
<th>Solvable</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8</td>
<td>triangle penalty</td>
<td>$\ll 1$ s</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>Preprocessing</th>
<th>Execution time</th>
<th>Solvable</th>
</tr>
</thead>
<tbody>
<tr>
<td>76</td>
<td></td>
<td>exp. penalty</td>
<td>26 s</td>
<td>yes</td>
</tr>
<tr>
<td>72</td>
<td></td>
<td>exp. penalty</td>
<td>3 min</td>
<td>yes</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td>exp. penalty</td>
<td>4 min</td>
<td>no (local optimum)</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>exp. penalty</td>
<td>9 min</td>
<td>no (local optimum)</td>
</tr>
</tbody>
</table>

Table 4. Experimental results applying the DDT heuristic to the three attack scenarios.