PDMI PREPRINT 16/2008

A FEEBLY TRAPDOOR FUNCTION*

Edward A. Hirsch Sergey I. Nikolenko

St. Petersburg Department of the
Steklov Institute of Mathematics of the
Russian Academy of Sciences

http://logic.pdmi.ras.ru/{~hirsch,~sergey}

September 12, 2008

Abstract. In 1992, A. Hiltgen [Hil92] provided the first constructions of provably (slightly) secure cryptographic primitives, namely *feebly one-way functions*. These functions are provably harder to invert than to compute, but the complexity (viewed as circuit complexity over circuits with arbitrary binary gates) is amplified only by a constant factor (in Hiltgen's works, the factor approaches 2).
In traditional cryptography, one-way functions are the basic primitive of private-key and digital signature schemes, while public-key cryptosystems are constructed with trapdoor functions. We continue Hiltgen's work by providing an example of a *feebly trapdoor function* where the adversary is guaranteed to spend more time than the honest participants (also by a constant factor).
Keywords: feebly one-way functions, circuit complexity, lower bounds, trapdoor functions.

*This research was partially supported by the Russian Foundation for Basic Research (RFBR) grant 08-01-00640-a, the Dynasty Foundation, the Presidential grant of the leading scientific schools NSh 4392.2008.1, and the fundamental research program of the Russian Academy of Sciences.

1 Introduction

Numerous public-key cryptographic protocols have been devised over the years, and yet there exists *not a single proof* of their security: neither an unconditional proof (that would necessarily imply P $\neq$ NP) nor a proof based on standard (not problem-specific) structural complexity assumptions. While universal primitives for one-way functions [Lev87] and public-key cryptosystems are known [HKN$^+$05] (see also [GHP08]), they give no connection to the core assumptions of traditional structural complexity theory; more, the asymptotic nature of polynomial-time reductions leaves no hope for confidence that a particular scheme cannot be broken at a particular key length. In general, it appears like there is still a very long way to go before cryptography can claim any *provably* secure public-key construction.

If we are unable to prove a superpolynomial gap between the complexities of honest parties and adversaries, can we prove at least *some* gap? In 1992, Alain Hiltgen [Hil92] managed to present a function that is *twice* harder to invert than to compute. His example is a linear function over $GF(2)$ with a matrix that has few non-zero entries while the inverse matrix has many non-zero entries; the complexity gap follows by a simple argument of Lamagna and Savage [LS73,Sav76]: every bit of the output depends non-idly on many variables and all these bits correspond to different functions, hence a lower bound on the complexity of computing them all together. The model of computation here is the most general one: the number of gates in a Boolean circuit that uses arbitrary binary Boolean gates. Note that little more could be expected for this model at present, since known lower bounds here are linear in the number of inputs [Blu84,Weg87].

In this work, we construct another feebly secure cryptographic primitive: namely, a feebly trapdoor function. Of course, in order to obtain the result, we have to prove a lower bound on the circuit complexity of a certain function; we use the gate elimination technique from circuit complexity of the eighties.

Our construction consists of two functions merged into one. For the first function, the adversary's task ("Charlie") is harder than that of the encryption party ("Bob"); for the second function, the adversary's task is harder than that of the decryption party ("Alice"). We prove then that if a part of the input is given to the first function and the rest of the input is given to the second function, Charlie's task is more difficult than that of both Alice and Bob. More formally, the complexity of inverting (decryption) without the use of trapdoor information is at least $\frac{25}{22}$ times greater than the complexities of honest encryption, decryption, and key generation.

In Section 2, we give basic definitions. Section 3 does preparational work, establishing some combinatorial properties of the matrices representing hard function candidates. Section 4 reviews the basic method of gate elimination, which we use to prove lower bounds on complexity, and applies it to the feeble security setting. Finally, Sections 6 and 7 present the construction of a feebly secure trapdoor function, Section 8 proves better guarantees on the success probability of weaker attackers, and Section 9 lists possible directions for further research.

2 Definitions and preliminaries

Let us denote by $\mathbb{B}_{n,m}$ the set of all 2^{m2^n} functions $f : \mathbb{B}^n \to \mathbb{B}^m$, where $\mathbb{B} = \{0, 1\}$ is the field of two elements. Our computational model is Boolean circuits with arbitrary binary gates. A circuit is a directed acyclic graph, every node of which has either in-degree zero (these nodes are called *circuit inputs*, or *variables*) or two (these nodes are called *gates*). Every gate is labelled by a binary Boolean function (any of the 16 functions in $\mathbb{B}_{2,1}$). Some of the gates are marked as *outputs*

(for technical reasons, we may also define an output as the negation of the value obtained in a gate: this way we can avoid introducing an extra negation gate). A circuit with n inputs and m outputs naturally computes a Boolean function in $\mathbb{B}_{n,m}$. We denote the size of a circuit by Size. The *circuit complexity* (or simply *complexity*) of a function f, denoted by $C(f)$, is the smallest number of gates in a circuit computing f (such circuit is called an *optimal* circuit for f), that is, $C(f) = \min_{c:\forall x\, c(x)=f(x)} \mathrm{Size}(c)$. We may safely assume that every gate of this circuit depends on both inputs, i.e., there are no constant functions and no unary functions Id and NOT, because such gates can be easily eliminated from the circuit without increasing the number of the gates.

Following Hiltgen, for every injective $f_n \in \mathbb{B}_{n,m}$ we can define its *measure of one-wayness*

$$M_F(f_n) = \frac{C(f_n^{-1})}{C(f_n)}.$$

Hiltgen's work was to find sequences of functions $f = \{f_n\}_{n=1}^{\infty}$ with large asymptotic constant

$$\liminf_{n\to\infty} M_F(f_n),$$

which Hiltgen calls f's *order of one-wayness*. We will discuss his results in more detail in Section 4.

We now extend this definition in order to capture feebly secure trapdoor functions. Since we are interested in constants here, we must pay attention to all the details.

Definition 1 *A* feebly trapdoor candidate *is a triple* $\mathcal{C} = \{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ *where*

- Key_n *is a family of sampling circuits* $\mathrm{Key}_n : \mathbb{B}^n \to \mathbb{B}^{\mathrm{pi}(n)} \times \mathbb{B}^{\mathrm{ti}(n)}$,
- Eval_n *is a family of evaluation circuits* $\mathrm{Eval}_n : \mathbb{B}^{\mathrm{pi}(n)} \times \mathbb{B}^{m(n)} \to \mathbb{B}^{c(n)}$, *and*
- Inv_n *is a family of inversion circuits* $\mathrm{Inv}_n : \mathbb{B}^{\mathrm{ti}(n)} \times \mathbb{B}^{c(n)} \to \mathbb{B}^{m(n)}$

such that for every n, *every seed* $s \in \mathbb{B}^n$, *and every input* $m \in \mathbb{B}^{m(n)}$,

$$\mathrm{Inv}_n(\mathrm{Key}_{n,2}(s), \mathrm{Eval}_n(\mathrm{Key}_{n,1}(s), m)) = m,$$

where $\mathrm{Key}_{n,1}(s)$ *and* $\mathrm{Key}_{n,2}(s)$ *are the first* $\mathrm{pi}(n)$ *bits ("public information") and the last* $\mathrm{ti}(n)$ *bits ("trapdoor information") of* $\mathrm{Key}_n(s)$, *respectively.*

Informally, n is the security parameter (the length of the random seed), $m(n)$ is the length of the input to the function, $c(n)$ is the length of the function's output, and $\mathrm{pi}(n)$ and $\mathrm{ti}(n)$ are lengths of the public and trapdoor information, respectively. We call this function a "candidate", because this definition does not imply any security, it merely sets up the dimensions and states the correctness of inversion. In our constructions, $m(n) = c(n)$ and $\mathrm{pi}(n) = \mathrm{ti}(n)$.

To find how secure a function is, we introduce the notion of a break. Informally, an adversary should invert the function without knowing the trapdoor information. We introduce break as inversion with probability greater than 50%. (We later investigate security against adversaries having smaller success probabilities, too.)

We denote by $C_{1/2}(f)$ the minimal size of a circuit that correctly computes a function $f \in \mathbb{B}_{n,m}$ on more than $\frac{1}{2}$ of its inputs (of length n). Obviously, $C_{1/2}(f) \le C(f)$ for all f.

Definition 2 *A circuit* N breaks *a feebly trapdoor candidate* $\mathcal{C} = \{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ *on seed length* n *if for the uniform probability distribution* U *taken over seeds* $s \in \mathbb{B}^n$ *and inputs* $m \in \mathbb{B}^{m(n)}$

$$\Pr_{s,\,m}\left[N(\mathrm{Key}_{n,1}(s), \mathrm{Eval}_n(\mathrm{Key}_{n,1}(s), m)) = m\right] > \frac{1}{2}.$$

Remark 1. In fact, in what follows we prove a stronger result: we prove that no circuit (of a certain size) can break our candidate *for any random seed* s, that is, for every seed s, every adversary fails with probability at least $1/2$.

For a trapdoor function to be secure, circuits that break the function should be larger than the circuits participating in its computation.

Definition 1. *A feebly trapdoor candidate* $\mathcal{C} = \{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ *has* order of security k *if for every sequence of circuits* $\{N_n\}_{n=1}^{\infty}$ *that break* f *on every input length* n*,*

$$\liminf_{n\to\infty} \min\left\{\frac{\mathrm{Size}(N_n)}{C(\mathrm{Key}_n)}, \frac{\mathrm{Size}(N_n)}{C(\mathrm{Eval}_n)}, \frac{\mathrm{Size}(N_n)}{C(\mathrm{Inv}_n)}\right\} \geq k.$$

Remark 2. As noted in Remark 1, we actually prove a stronger thing, namely

$$\liminf_{n\to\infty} \left\{\frac{C_{1/2}(f_{\mathrm{pi}(n)+c(n)})}{C(\mathrm{Key}_n)}, \frac{C_{1/2}(f_{\mathrm{pi}(n)+c(n)})}{C(\mathrm{Eval}_n)}, \frac{C_{1/2}(f_{\mathrm{pi}(n)+c(n)})}{C(\mathrm{Inv}_n)}\right\} \geq k,$$

where $f_{\mathrm{pi}(n)+c(n)} \in \mathbb{B}_{\mathrm{pi}(n)+c(n),m(n)}$ maps $\big(\mathrm{Key}_{n,1}(s), \mathrm{Eval}_n(\mathrm{Key}_{n,1}(s), m)\big)$ to m.

Remark 3. One could consider key generation as a separate process and omit its complexity from the definition of the order of security. However, we prove our results for the definition stated above as it makes them stronger.

Remark 4. Let us note explicitly that we are talking about *one-time* security. An adversary can amortize his circuit complexity on inverting a feebly trapdoor candidate for the second time for the same seed, for example, by computing the trapdoor information and successfully reusing it. Thus, in our setting one has to pick a new seed for every input.

In the next sections, we develop our construction of a feebly trapdoor function (that is, a feebly trapdoor candidate with nontrivial order of security).

3 Hard matrices

All our constructions are based on a linear function $f : \mathbb{B}^n \to \mathbb{B}^n$ shown by A. Hiltgen [Hil92] to be feebly one-way of order $3/2$. We restrict ourselves to the case when $n \equiv 0 \mod 4$ for reasons that will presently become clear. Note that Definition 1 carries through this restriction: for $n \not\equiv 0 \mod 4$ one can simply consider circuit with input size equal to the lowest multiple of 4 greater than n.

In what follows we denote the matrix of Hiltgen's function f by A. Each row of A contains two 1's (the diagonal element and the element next to it), except for the last row that contains three

1's in positions 1, $\lceil \frac{n}{2} \rceil$, and n. Here are the matrices A and A^{-1}:

$$A\boldsymbol{x} = \begin{pmatrix} 1\,1\,0\ldots0\ldots0\,0\,0 \\ 0\,1\,1\ldots0\ldots0\,0\,0 \\ 0\,0\,1\ldots0\ldots0\,0\,0 \\ \vdots\,\vdots\,\vdots \quad \vdots \quad \vdots\,\vdots\,\vdots \\ 0\,0\,0\ldots0\ldots1\,1\,0 \\ 0\,0\,0\ldots0\ldots0\,1\,1 \\ 1\,0\,0\ldots1\ldots0\,0\,1 \end{pmatrix} \boldsymbol{x}, \quad A^{-1}\boldsymbol{x} = \left(\begin{array}{c|c} 0\,0\ldots0 & 1\,1\ldots1\,1 \\ 1\,0\ldots0 & 1\,1\ldots1\,1 \\ 1\,1\ldots0 & 1\,1\ldots1\,1 \\ \vdots\,\vdots \quad \vdots & \vdots\,\vdots \quad \vdots\,\vdots \\ 1\,1\ldots0 & 1\,1\ldots1\,1 \\ \hline 1\,1\ldots1 & 1\,1\ldots1\,1 \\ 1\,1\ldots1 & 0\,1\ldots1\,1 \\ 1\,1\ldots1 & 0\,0\ldots1\,1 \\ \vdots\,\vdots \quad \vdots & \vdots\,\vdots \quad \vdots\,\vdots \\ 1\,1\ldots1 & 0\,0\ldots0\,1 \end{array}\right) \boldsymbol{x}.$$

Note that there are at least $\lceil \frac{n}{2} \rceil$ nonzero entries in each row of A^{-1}, and the $\lceil \frac{n}{2} \rceil$'s row has no zeroes at all. We will also need the square A^{-2}, which looks like

$$A^{-2}\boldsymbol{x} = \left(\begin{array}{c|c} 1\,1\,1\,1\ldots1\,1\,1 & 1\,0\,1\ldots1\,0\,1 \\ 1\,1\,1\,1\ldots1\,1\,1 & 0\,1\,0\ldots0\,1\,0 \\ 0\,1\,1\,1\ldots1\,1\,1 & 1\,0\,1\ldots1\,0\,1 \\ 1\,0\,1\,1\ldots1\,1\,1 & 0\,1\,0\ldots0\,1\,0 \\ 0\,1\,0\,1\ldots1\,1\,1 & 1\,0\,1\ldots1\,0\,1 \\ \vdots\,\vdots\,\vdots\,\vdots\,\vdots \quad \vdots\,\vdots\,\vdots & \vdots\,\vdots\,\vdots\,\vdots \quad \vdots\,\vdots \\ 1\,0\,1\,0\ldots1\,1\,1 & 0\,1\,0\ldots0\,1\,0 \\ 0\,1\,0\,1\ldots0\,1\,1 & 1\,0\,1\ldots1\,0\,1 \\ \hline 1\,0\,1\,0\ldots1\,0\,1 & 0\,1\,0\ldots0\,1\,0 \\ 0\,1\,0\,1\ldots0\,1\,0 & 1\,0\,1\ldots1\,0\,1 \\ 1\,0\,1\,0\ldots1\,0\,1 & 1\,1\,0\ldots0\,1\,0 \\ 0\,1\,0\,1\ldots0\,1\,0 & 1\,1\,1\ldots1\,0\,1 \\ \vdots\,\vdots\,\vdots\,\vdots\,\vdots \quad \vdots\,\vdots\,\vdots & \vdots\,\vdots\,\vdots\,\vdots \quad \vdots\,\vdots \\ 0\,1\,0\,1\ldots1\,0\,1 & 1\,1\,1\ldots1\,0\,1 \\ 1\,0\,1\,0\ldots0\,1\,0 & 1\,1\,1\ldots1\,1\,0 \end{array}\right) \boldsymbol{x}$$

provided n is divisible by 4 (that is why we restrict ourselves to this case). Note that this matrix also has the property that each row has at least $\frac{n}{2}$ 1's in it; moreover, its triangular blocks of 1's coincide with triangular blocks of 0's in A^{-1}, and the remaining space is filled with 0's and 1's chequered.

We are also interested in the $n \times 2n$ matrix $\mathfrak{A}$ consisting of A^{-2} and A^{-1} stacked together:

$$\mathfrak{A} = \left(A^{-2}\ A^{-1}\right).$$

We need the following properties of A^{-1} and $\mathfrak{A}$.

Lemma 1 *Let $n \equiv 0 \pmod 4$. Then*

1. *All columns of $\mathfrak{A}$ (and, hence, A^{-1}) are different.*
2. *Each row of A^{-1} (resp., $\mathfrak{A}$) contains at least $\frac{n}{2}$ (resp., $\frac{5n}{4}$) nonzero entries.*

3. *After eliminating all but two (resp., all but five) columns of A^{-1} (resp., $\mathfrak{A}$) there remains at least one row with two nonzero entries.*

Proof. The first claim is obvious. The i^{th} row of A^{-1} contains $\frac{n}{2}+i$ nonzero entries for $i \le \frac{n}{2}$ and $\frac{n}{2}+n-i$ nonzero entries for $i \ge \frac{n}{2}$. Thus, the second claim holds for the matrix A^{-1}. At the same time, the i's row of A^{-2} contains at least $\frac{3n}{4}-\frac{i}{2}$ nonzero entries for $i \le \frac{n}{2}$ and at least $\frac{n}{2}+\frac{1}{2}(i-\frac{n}{2}-1)$ nonzero entries for $i \ge \frac{n}{2}$. Therefore, the i^{th} row of A^{-2} contains at least

$$\frac{n}{2}+i+\frac{3n}{4}-\frac{i}{2}=\frac{5n}{4}+\frac{i}{2}$$

nonzero entries for $i \le \frac{n}{2}$ and at least

$$\frac{n}{2}+n-i+\frac{n}{2}+\frac{1}{2}(i-\frac{n}{2}-1)=\frac{7n}{4}-\frac{1}{2}(i-1) \ge \frac{5n}{4}$$

nonzero entries for $i \ge \frac{n}{2}$.

Let us now prove the third claim. Since A^{-1} has a row that contains only nonzero entries, all but one columns of this matrix should be eliminated to leave just one nonzero entry. The same holds for the left part of the matrix A^{-2} (see its first row). The same holds for the right part of the matrix A^{-2} without the last column (see its last row). □

4 Gate elimination

In this section, we first briefly remind about Hiltgen's methods and then introduce gate elimination as the primary technique for proving our bounds. Hiltgen proved all his bounds with the following very simple argument due to Lamagna and Savage.

Proposition 1 ([LS73,Sav76]; [Hil92, Theorems 3 and 4])

1. *Suppose that $f:\mathbb{B}^n \to \mathbb{B}$ depends non-idly on each of its n variables, that is, for every i there exist values $a_1,\dots,a_{i-1},a_{i+1},\dots,a_n \in \mathbb{B}$ such that*

$$f(a_1,\dots,a_{i-1},0,a_{i+1},\dots,a_n) \ne f(a_1,\dots,a_{i-1},1,a_{i+1},\dots,a_n).$$

   Then $C(f) \ge n-1$.
2. *Let $f=(f^{(1)},\dots,f^{(m)}):\mathbb{B}^n \to \mathbb{B}^m$, where $f^{(k)}$ is the k^{th} component of f. If the m component functions $f^{(i)}$ are pairwise different and each of them satisfies $C(f^{(i)}) \ge c \ge 1$ then $C(f) \ge c+m-1$.*

Proof. 1. Consider a minimal circuit of size s implementing f. Since f depends (here and further we say "depends on" meaning "depends non-idly on") on all n variables, each of the input gates should have at least one outcoming edge. Since the circuit is minimal, every gate in it, except perhaps the output, should have at least one outcoming edge. Therefore, the circuit has at least $s+n-1$ edges in it; on the other hand, a circuit of s binary gates cannot have more than $2s$ edges. Thus, $2s \ge s+n-1$.

2. Consider a circuit for f. Any such circuit has at least $c-1$ gates such the circuit complexities of functions they compute are at most $c-1$ (to see that, take the first $c-1$ gates in any topological order). Therefore, to calculate any component function $f^{(i)}$ we would have to add at least one gate, and keep doing it for every other function because a single new gate would only allow to calculate one function more. That gives the necessary bound of $c+m-1$.

Hiltgen counted the minimal complexity of computing one bit of the input (e.g. since each row of A^{-1} has at least $\frac{n}{2}$ nonzero entries, the minimal complexity of each component of $A^{-1}\boldsymbol{y}$ is $\frac{n}{2}$) and thus produced lower bounds on the complexity of inverting the function (e.g. the complexity of computing $A^{-1}\boldsymbol{y}$ is $\frac{n}{2}+n-2=\frac{3n}{2}-2$).

Besides, in cryptography it is generally desirable to prove not only worst-case bounds, but also that an adversary is unable to invert the function on a substantial fraction of inputs. The first step towards this would be to prove the following fact.

Proposition 2 *For each of the matrices constructed here, any circuit using less than the minimal necessary number of gates inverts it on less than $\frac{1}{2}$ of the inputs.*

In Hiltgen's works, this fact followed from a very simple observation (which was not even explicitly stated).

Lemma 2 *Consider a function $f=\bigoplus_{i=1}^{n} x_i$. For any function g that depends on only m of these variables with $m<n$*

$$\Pr_{x_1,\ldots,x_n}\left[f(x_1,\ldots,x_n)=g(x_{i_1},\ldots,x_{i_m})\right]=\frac{1}{2}.$$

Proof. Since $m<n$, there exists an index $j\in 1..n$ such that g does not depend on x_j. This means that for every set of values of the other variables, whatever the value of g is, for one of the values of x_j f coincides with g, and on the other value f differs from g. This means that f differs from g on precisely $\frac{1}{2}$ of the inputs. □

The argument suffices for Hiltgen's feebly one-wayness result for the square matrix A^{-1}: first we apply the first part of Proposition 1 and see that every output has complexity at least $\frac{n}{2}-1$, and then the second part of Proposition 1 yields the necessary bound of $\frac{3n}{2}-1$. Moreover, if a circuit has less than the necessary number of gates, one of its outputs inevitably depends on less than the necessary number of input variables, which, by Lemma 2, gives the necessary $\frac{1}{2}$ error rate.

However, in this paper we also use non-square matrices, and it turns out that a similar simple argument does not provide sufficient bounds for our matrices. Therefore, we use a different way of proving lower bounds, namely *gate elimination* that has been previously used for every lower bound in "regular" circuit complexity [Weg87]. The basic idea of this method is to use the following inductive argument. Consider function f and substitute some value c for some variable x thus obtaining a circuit for the function $f|_{x=c}$. The circuit can be simplified, because the gates that had this variable as inputs become either unary (recollect that the negation can be embedded into subsequent gates) or constant (in this case we can proceed to eliminating subsequent gates). The important case here is when the gate is non-linear, such as an AND or an OR gate. In this case it is always possible to choose a value for an input of such gate so that this gate becomes a constant. One then proceeds by induction as long as it is possible to find a suitable variable that eliminates many enough gates. Evidently, the number of eliminated gates is a lower bound on the complexity of f.

In this paper, gate elimination is used in one of its simplest forms. Here is a "master" lemma that we will apply to our matrices.

Lemma 3 *Let $t,u\geq 1$. Assume that $\chi:\mathbb{B}^{v(n)}\to\mathbb{B}^n$ is a linear function with matrix X over $GF(2)$. Assume also that*

– *all columns of X are different;*
– *every row of X has at least u nonzero entries;*
– *after removing any t columns of X, the matrix still has at least one row containing at least two nonzero entries.*

Then $C(\chi) \geq u + t$ and, moreover, $C_{1/2}(\chi) \geq u + t$.

Proof. Consider a circuit implementing χ on more than $\frac{1}{2}$ of the inputs and choose a topological order on its nodes. We denote the actual function this circuit implements by h (it does not need to be linear, but does have to coincide with χ on most inputs).

Consider the topmost gate g in this order. Since g is topmost, its incoming edges come from the inputs of the circuit, denote them by x and y. By Lemma 2, neither of its input variables can be marked as an output, because even after the first $u - 1$ deleted columns each row has at least two variables.

The following possibilities exhaust all possible cases.

1. One of the input variables of g, say x, enters some other gate. In this case by setting x to any constant we can eliminate at least 2 gates.
2. g is an non-linear gate, and neither x nor y enters any other gate. In this case, if g has children, by setting x to a suitable constant we can eliminate both g and its children, at least 2 gates in total.
   If g does not have children, it means that g is an output. Consider this output h_i. If χ_i depends on some other variable z we reach the 50% error rate by flipping z. To see this, consider two different assignments for z, 0 and 1, and fix values of all other variables. For one of the assignments to z, h will have a wrong answer, because $h_i(\ldots, z = 0, x = a, y = b, \ldots) = h_i(\ldots, z = 1, x = a, y = b, \ldots)$ (h_i does not depend on z), but $\chi_i(\ldots, z = 0, x = a, y = b, \ldots) \neq \chi_i(\ldots, z = 0, x = a, y = b, \ldots)$, because $\chi_i = z \oplus \ldots$. And if $\chi_i = x$, $\chi_i = y$, $\chi_i = x \oplus y$ or any of these cases with negation, we can substitute g with a linear gate that will not change anything else in the circuit, but will compute χ_i precisely, with zero error. This will get us into the conditions of case 3.
3. g is a linear gate, and neither x nor y enters any other gate. In this case, h does not depend on x or y separately, but only on $x \oplus y$. For a circuit that calculates χ precisely, this would mean that X has two coinciding columns, which is not true.
   For a circuit that tries to approximate χ, suppose that g is a linear gate, and neither x nor y enters any other gate. In this case, h does not depend on x or y separately, but only on $x \oplus y$. Let us show that in this case h differs from χ on at least half of the inputs. First, note that since χ does depend on x and y separately, there exists an output χ_i that depends only on one of these variables (say, x). Thus, $\chi_i = x \oplus \bigoplus_{z \in Z} z$, where $y \notin Z$. There are two cases: either the output h_i in function h depends on $x \oplus y$, or it does not depend on either x or y.
   In the first case, consider two different assignments for y, 0 and 1, and fix values of all other variables. For one of the assignments to y, h will have a wrong answer, because

$$h_i(\ldots, y = 0, x = a, \ldots) \neq h_i(\ldots, y = 1, x = a, \ldots)$$

   (when we flip y, we flip $x \oplus y$), but

$$\chi_i(\ldots, y = 0, x = a, \ldots) = \chi_i(\ldots, y = 1, x = a, \ldots),$$

   because χ_i does not depend on y at all. In the second case, the same holds if we flip x: h will not change its output, while χ will.

Thus, as long as all rows of X contain more than one variable, we are able to eliminate at least two gates by substituting one variable, at least $2(u-1)$ gates in total. Note that substituting a value for a variable is equivalent to removing a column from the matrix.

What happens after all "good" variables have been eliminated? We continue[1] the same gate elimination process with one exception: now it is also possible (both in a precise calculation and in an approximation) that one of the input variables or the result of the gate is marked as an output, and thus we will eliminate only one gate per step. This will continue as long as there is at least one row with two nonzero entries (put another way, if at least one row still has two nonzero entries, the circuit must contain at least one gate, and thus the topmost gate indeed exists). Therefore, we will eliminate at least $2(u-1) + ((t+1) - (u-1)) = u + t$ gates in total. □

In what follows we will also use block-diagonal matrices. Intuition hints that joint computation of two functions that have different inputs should be as hard as computing them separately (thus, the lower bound should be the sum of respective lower bounds). However, for certain functions it is not the case, as seen in [Weg87, Section 10.2] We show it for our particular case. For clarity, we state the result for two blocks and then extend it to any number of blocks.

Lemma 4 *Assume that an $n \times v_1(n)$ matrix X_1 and a $w(n) \times v_2(n)$ matrix X_2 satisfy the requirements of Lemma 3 with $u_1(n)$ (resp., $u_2(n)$) and $t_1(n)$ (resp., $t_2(n)$). (Note that they may be non-square and of different dimensions.)*

Denote by $\zeta(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) = (X_1\boldsymbol{x}^{(1)}, X_2\boldsymbol{x}^{(2)})$ the function determined by the block-diagonal matrix $\begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix}$. Then

$$C(\zeta) \geq u_1(n) + t_1(n) + u_2(n) + t_2(n) - 1$$

and, moreover,

$$C_{1/2}(\zeta) \geq u_1(n) + t_1(n) + u_2(n) + t_2(n) - 1.$$

Proof. We proceed similarly to Lemma 3. Note that when we substitute a variable from $x^{(1)}$, it does not change anything in X_2, and vice versa. Thus we substitute "good" variables (those that eliminate two gates) as long as we have them and then substitute "bad" variables (eliminating one gate per step) when we do not have good ones *separately for each matrix*. If one of the matrices runs out of rows that contain at least two nonzero entries (it may happen after eliminating $u_i(n)-1$ "good" and then $t_i(n) - u_i(n) + 2$ other variables from it), we substitute the remaining variables corresponding to this matrix and forget about this part of the block-diagonal matrix.

It can happen, however, that one of the inputs (variables) in the topmost gate is from $x^{(1)}$ and the other one is from $x^{(2)}$. It does not change anything when both these variables are "good" or both are "bad". However, if exactly one of these variables is "good", the situation is different, and we have to re-check cases 1–3 in the proof of Lemma 3.

The first two cases go smoothly, and we happily substitute a value for a variable if it eliminates two gates. Case 3 still cannot happen in a circuit calculating ζ precisely, because we still cannot get identical columns, and in an approximation, the argument is identical to that of Lemma 3. □

[1] One could see the end of proof here without continuing the gate elimination process, but we will re-use this proof later in Lemma 4.

Lemma 5 *Assume that a linear function ζ is determined by a block diagonal matrix*

$$\zeta\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\ldots,\boldsymbol{x}^{(m)}\right)=\begin{pmatrix}X_1 & 0 & \ldots & 0\\ 0 & X_2 & \ldots & 0\\ \vdots & \vdots & & \vdots\\ 0 & 0 & \ldots & X_m\end{pmatrix}\begin{pmatrix}\boldsymbol{x}^{(1)}\\ \boldsymbol{x}^{(2)}\\ \vdots\\ \boldsymbol{x}^{(m)}\end{pmatrix},$$

and the matrices X_j satisfy the requirements of Lemma 3 with $u_j(n)$ and $t_j(n)$, respectively (the matrices may be non-square and of different dimensions). Then

$$C(\zeta)\geq\sum_{j=1}^{m}(u_j(n)+t_j(n))-m+1$$

and, moreover,

$$C_{1/2}(\zeta)\geq\sum_{j=1}^{m}(u_j(n)+t_j(n))-m+1.$$

Proof. Note that all gate elimination arguments deal with only two input variables. These input variables may come from not more than two different blocks of ζ's block diagonal matrix. Thus, the proof is identical to the proof of Lemma 4. □

We now formulate the direct consequences of these lemmas and combinatorial lemmas about our specific matrices.

Lemma 6 *Let $n,n'\equiv 0 \pmod 4$,*

$$\alpha(\boldsymbol{x})=A^{-1}\boldsymbol{x},\qquad \alpha_2(\boldsymbol{x})=\left(A^{-1}\ A^{-2}\right)\boldsymbol{x},\qquad \alpha_*(\boldsymbol{x})=\begin{pmatrix}A^{-1} & A^{-2} & 0\\ 0 & 0 & A_*^{-1}\end{pmatrix}\boldsymbol{x},$$

where A_*^{-1} denotes a matrix with the same structure as A^{-1}, but with dimension n' instead of n. Then

$$C(\alpha)\geq\frac{3n}{2}-2,\qquad C(\alpha_2)\geq\frac{13n}{4}-5,\qquad C(\alpha_*)\geq\frac{3n'}{2}+\frac{13n}{4}-7,$$

and no circuit with less than the specified number of gates can compute the corresponding function on more than $\frac{1}{2}$ of the inputs.

Proof. Follows from Lemmas 3 and 4 by substituting the respective bounds $u(n)$ and $t(n)$ from Lemma 1 (in particular, $t(n)=n-2$ for the matrix A^{-1} and $t(n)=2n-5$ for $\mathfrak{A}$). □

On the other hand, note the following upper bounds.

Lemma 7 *1. There exists a circuit of size $\frac{3n}{2}-1$ that implements the linear function $\phi:\mathbb{B}^n\to\mathbb{B}^n$ with matrix A^{-1}.*

2. There exists a circuit of size $\frac{7n}{2}$ that implements the linear function $\phi:\mathbb{B}^{2n}\to\mathbb{B}^n$ with matrix $\left(A^{-1}\ A\right)$.

3. There exists a circuit of size $\frac{5n}{2}-1$ that implements the linear function $\phi:\mathbb{B}^{2n}\to\mathbb{B}^n$ with matrix $\left(A^{-1}\ A^{-1}\right)$.

Proof. 1. First construct the sum $\bigoplus_{i=1}^{n/2} x_i$ ($\frac{n}{2}-1$ gates). Then, adding one by one each of the inputs x_i, $i = \frac{n}{2}..n$, compute all outputs y_i, $i = \frac{n}{2}..n$ and, by the way, the sum of all inputs $\bigoplus_{i=1}^{n} x_i$ (this takes another $\frac{n}{2}$ gates). Finally, the first $\frac{n}{2}$ outputs will be computed by "subtracting" the first $\frac{n}{2}$ inputs from the sum of all inputs one by one (another $\frac{n}{2}$ gates).

2. To implement the left part of this matrix, we need $\frac{3n}{2} - 1$ gates. Afterwards we add to each output the two bits from the right part of the matrix (three bits in case of the last row); we add $2n + 1$ gates in this way.

3. Note that in this case

$$\phi(a, b) = \begin{pmatrix} A^{-1} & A^{-1} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = A^{-1}(a \oplus b)$$

for any $a, b \in \mathbb{B}^n$. Thus, we first add $a \oplus b$ (n gates) and then implement A^{-1} ($\frac{3n}{2} - 1$ gates). □

5 A toy example

It is usually illustrative to show the constructions on a toy example before embarking onto the general proof. In this section, we will show an example of what we mean by a feebly secure trapdoor function.

As already mentioned before, all our constructions are based on a linear function $f : \mathbb{B}^n \to \mathbb{B}^n$ developed by A. Hiltgen [Hil92]. The smallest n for which f is actually harder to invert than to compute is 7, but since we need n divisible by 4 we take $n = 8$ (also, in terms of Section 7, we use $\alpha = 1$). Thus, we have

$$A = \begin{pmatrix} 1&1&0&0&0&0&0&0 \\ 0&1&1&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 0&0&0&1&1&0&0&0 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&1&1 \\ 1&0&0&1&0&0&0&1 \end{pmatrix}, \quad A^{-1} = \begin{pmatrix} 0&0&0&1&1&1&1&1 \\ 1&0&0&1&1&1&1&1 \\ 1&1&0&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&0&1&1&1&1 \\ 1&1&1&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&0&0&0&0&1 \end{pmatrix}, \quad A^{-2} = \begin{pmatrix} 1&1&1&1&0&1&0&1 \\ 1&1&1&0&1&0&1&0 \\ 0&1&1&1&0&1&0&1 \\ 1&0&1&0&1&0&1&0 \\ 0&1&0&1&0&1&0&1 \\ 1&0&1&1&1&0&1&0 \\ 0&1&0&1&1&1&0&1 \\ 1&0&1&1&1&1&1&0 \end{pmatrix}.$$

In this case, $C(f) = 9$, $C(f^{-1}) = 11$, $C(f^{-2}) = 11$.

Our feebly secure trapdoor function will consist of the following components.

$$\mathrm{Key}_8 = \left(\begin{array}{c} 10000000 \\ 01000000 \\ 00100000 \\ 00010000 \\ 00001000 \\ 00000100 \\ 00000010 \\ 00000001 \\ \hline 00011111 \\ 10011111 \\ 11011111 \\ 11111111 \\ 11101111 \\ 11100111 \\ 11100011 \\ 11100001 \end{array}\right), \quad \mathrm{Eval}_8 = \left(\begin{array}{c|c|c} 00011111 & 11000000 & 00000000 \\ 10011111 & 01100000 & 00000000 \\ 11011111 & 00110000 & 00000000 \\ 11111111 & 00011000 & 00000000 \\ 11101111 & 00001100 & 00000000 \\ 11100111 & 00000110 & 00000000 \\ 11100011 & 00000011 & 00000000 \\ 11100001 & 10010001 & 00000000 \\ \hline 00000000 & 00000000 & 11000000 \\ 00000000 & 00000000 & 01100000 \\ 00000000 & 00000000 & 00110000 \\ 00000000 & 00000000 & 00011000 \\ 00000000 & 00000000 & 00001100 \\ 00000000 & 00000000 & 00000110 \\ 00000000 & 00000000 & 00000011 \\ 00000000 & 00000000 & 10010001 \end{array}\right),$$

$$\mathrm{Inv}_8 = \left(\begin{array}{c|c|c} 00011111 & 00011111 & 00000000 \\ 10011111 & 10011111 & 00000000 \\ 11011111 & 11011111 & 00000000 \\ 11111111 & 11111111 & 00000000 \\ 11101111 & 11101111 & 00000000 \\ 11100111 & 11100111 & 00000000 \\ 11100011 & 11100011 & 00000000 \\ 11100001 & 11100001 & 00000000 \\ \hline 00000000 & 00000000 & 00011111 \\ 00000000 & 00000000 & 10011111 \\ 00000000 & 00000000 & 11011111 \\ 00000000 & 00000000 & 11111111 \\ 00000000 & 00000000 & 11101111 \\ 00000000 & 00000000 & 11100111 \\ 00000000 & 00000000 & 11100011 \\ 00000000 & 00000000 & 11100001 \end{array}\right).$$

It is routine to verify that, indeed,

$$\mathrm{Inv}_8 \begin{pmatrix} \mathrm{Key}_{8,2}(r) \\ \mathrm{Eval}_8(\mathrm{Key}_{8,1}(r), m) \end{pmatrix} = m$$

for any input $m \in \mathbb{B}^8$ and any bit string (seed) $r \in \mathbb{B}^8$, where $\mathrm{Key}_{8,1}(r)$ and $\mathrm{Key}_{8,2}(r)$ are the first and the second eight-bit halves of $\mathrm{Key}_8(r)$, respectively. Here are the complexities (by Lemma 7):

$$\begin{aligned} C(\mathrm{Key}_8) &= 11, \\ C(\mathrm{Eval}_8) &= 29, \\ C(\mathrm{Inv}_8) &= 30. \end{aligned}$$

An adversary would have to compute the following function:

$$\mathrm{Adv}_8 = \left(\begin{array}{ccc|ccc|ccc} 1&1&1&1&0&1&0&1 & 0&0&0&1&1&1&1&1 & 0&0&0&0&0&0&0&0 \\ 1&1&1&0&1&0&1&0 & 1&0&0&1&1&1&1&1 & 0&0&0&0&0&0&0&0 \\ 0&1&1&1&0&1&0&1 & 1&1&0&1&1&1&1&1 & 0&0&0&0&0&0&0&0 \\ 1&0&1&0&1&0&1&0 & 1&1&1&1&1&1&1&1 & 0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&1&0&1 & 1&1&1&0&1&1&1&1 & 0&0&0&0&0&0&0&0 \\ 1&0&1&1&1&0&1&0 & 1&1&1&0&0&1&1&1 & 0&0&0&0&0&0&0&0 \\ 0&1&0&1&1&1&0&1 & 1&1&1&0&0&0&1&1 & 0&0&0&0&0&0&0&0 \\ 1&0&1&1&1&1&1&0 & 1&1&1&0&0&0&0&1 & 0&0&0&0&0&0&0&0 \\ \hline 0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&0 & 0&0&0&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&0 & 1&0&0&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&0 & 1&1&0&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&0 & 1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&0 & 1&1&1&0&1&1&1&1 \\ 0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&0 & 1&1&1&0&0&1&1&1 \\ 0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&0 & 1&1&1&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0 & 0&0&0&0&0&0&0&0 & 1&1&1&0&0&0&0&1 \end{array}\right).$$

By Lemma 6,

$$C(\mathrm{Adv}_8) \geq 31.$$

Thus, we get a feebly secure trapdoor function where inversion without the trapdoor information takes at least 31 gates, while evaluation takes 29 gates and inversion with trapdoor takes 30 gates. This means that this trapdoor function candidate with, so to speak, order of security $\frac{31}{30}$ ("so to speak", because Definition 1 handles only limits with n approaching infinity). In what follows, we generalize this construction and prove a $\frac{25}{22}$ order of security.

6 Two constructions

We are almost ready to present the construction of our feebly trapdoor function (recall Definition 1). In this section, we consider two different constructions of a candidate feebly trapdoor function. None of these constructions work; however, we will merge them into one in the subsequent section, and the resulting construction will do the job.

In our first construction, the inversion with trapdoor is faster than inversion without trapdoor, but, unfortunately, evaluating the function is even harder. In terms of Definition 1, we now present a feebly trapdoor candidate with identical lengths of the seed, public information, trapdoor, input, and output $c(n) = m(n) = \mathrm{pi}(n) = \mathrm{ti}(n) = n$. Given a random seed, the sampler produces a pair of public and trapdoor information (pi, ti), where ti is the random seed itself and $pi = A^{-1}(ti)$ (thus, the sampler can be implemented using $\frac{3n}{2} - 1$ gates). In this construction, evaluation produces the output c for an input m as follows:

$$\mathrm{Eval}_n(pi, m) = A^{-1}(pi) \oplus A(m).$$

Here is the evaluation matrix:

$$\left(\begin{array}{cccc|cccc}
0\,0\ldots0\,1\,1\ldots1\,1 & 1\,1\,0\ldots0\,0\,0\ldots0\,0\,0 \\
1\,0\ldots0\,1\,1\ldots1\,1 & 0\,1\,1\ldots0\,0\,0\ldots0\,0\,0 \\
1\,1\ldots0\,1\,1\ldots1\,1 & 0\,0\,1\ldots0\,0\,0\ldots0\,0\,0 \\
\vdots\;\;\vdots\quad\vdots\;\vdots\;\vdots\quad\vdots\;\;\vdots & \vdots\;\vdots\;\vdots\quad\vdots\;\vdots\;\vdots\quad\vdots\;\vdots\;\vdots \\
1\,1\ldots1\,1\,1\ldots1\,1 & 0\,0\,0\ldots1\,1\,0\ldots0\,0\,0 \\
1\,1\ldots1\,0\,1\ldots1\,1 & 0\,0\,0\ldots0\,1\,1\ldots0\,0\,0 \\
1\,1\ldots1\,0\,0\ldots1\,1 & 0\,0\,0\ldots0\,0\,1\ldots0\,0\,0 \\
\vdots\;\;\vdots\quad\vdots\;\vdots\;\vdots\quad\vdots\;\;\vdots & \vdots\;\vdots\;\vdots\quad\vdots\;\vdots\;\vdots\quad\vdots\;\vdots\;\vdots \\
1\,1\ldots1\,0\,0\ldots1\,1 & 0\,0\,0\ldots0\,0\,0\ldots1\,1\,0 \\
1\,1\ldots1\,0\,0\ldots1\,1 & 0\,0\,0\ldots0\,0\,0\ldots0\,1\,1 \\
1\,1\ldots1\,0\,0\ldots0\,1 & 1\,0\,0\ldots1\,0\,0\ldots0\,0\,1
\end{array}\right).$$

An upper bound on evaluation circuit complexity follows from Lemma 7; one can evaluate this function with a circuit of size $\frac{7n}{2}$.

Inversion with trapdoor goes as follows:

$$\mathrm{Inv}_n(ti, c) = A^{-1}(A^{-1}(pi) \oplus c) = A^{-1}(ti \oplus c).$$

Due to the nice linearity (note that bounds proven in previous sections do not apply here, because the inversion matrix has a lot of identical columns), this circuit can be implemented in $\frac{5n}{2} - 1$ gates: first one computes $ti \oplus c$ using n gates, then one applies A^{-1} using another $\frac{3n}{2} - 1$ gates (see Lemma 7).

Finally, an adversary has to invert Bob's message the hard way:

$$m = A^{-1}(A^{-1}(pi) \oplus c) = \mathfrak{A}\begin{pmatrix} pi \\ c \end{pmatrix}.$$

By Lemma 6, the complexity of this function is at least $\frac{13n}{4} - 5$ gates, and any adversary with less than $\frac{13n}{4} - 5$ gates fails on at least 50% of the inputs.

In this construction evaluation is harder than inversion without trapdoor. In order to fix this problem, we use also a different construction, also a candidate trapdoor function now with $c(n) = m(n) = n$ and $\mathrm{pi}(n) = \mathrm{ti}(n) = 0$. Our second construction is just Hiltgen's feebly one-way function. Thus, the public and trapdoor information is not used at all, and the evaluation–inversion functions are as follows:

$$\begin{aligned} \mathrm{Eval}_n(m) &= A(m), \\ \mathrm{Inv}_n(c) &= A^{-1}(c), \\ \mathrm{Adv}_n(c) &= A^{-1}(c). \end{aligned}$$

This construction, of course, is not a trapdoor function at all because inversion is implemented with no regard for the trapdoor. For a message m of length $|m| = n$ the evaluation circuit has $n+1$ gates, while inversion, by Lemma 7, can be performed only by circuits of $\frac{3n}{2} - 1$ gates each. Thus, in this construction evaluation is easy, while inversion is hard, with or without trapdoor.

7 The feebly secure trapdoor function

In the previous section, we constructed two candidate trapdoor functions. In one of them evaluation was easy, while inversion was hard; in the other inversion with trapdoor was easy, and evaluation and inversion without trapdoor were both hard.

We now combine the two functions. The resulting one will make it easier for both inversion with trapdoor and evaluation than for an adversary.

We split the input into two parts; the first part m_1, of length n, will be subject to our first (less trivial) construction, while the second part m_2, of length αn, will be subject to the second construction. We will choose α later to maximize the relative hardness for an adversary.

Now each participant has a block-diagonal matrix:

$$\mathrm{Eval}_n(pi, m) = \begin{pmatrix} A^{-1} & A & 0 \\ 0 & 0 & A_* \end{pmatrix} \begin{pmatrix} pi \\ m_1 \\ m_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix};$$

$$\mathrm{Inv}_n(ti, c) = \begin{pmatrix} A^{-1} & A^{-1} & 0 \\ 0 & 0 & A_*^{-1} \end{pmatrix} \begin{pmatrix} ti \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix},$$

$$\mathrm{Adv}_n(pi, m) = \begin{pmatrix} A^{-2} & A^{-1} & 0 \\ 0 & 0 & A_*^{-1} \end{pmatrix} \begin{pmatrix} pi \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix},$$

where A_* denotes the matrix with the same structure as A, but with dimension αn instead of n. Thus, in terms of Definition 1, we get a feebly trapdoor candidate where inputs and outputs are longer than the seed and the public and trapdoor information: $\mathrm{pi}(n) = \mathrm{ti}(n) = n$, $c(n) = m(n) = (1 + \alpha)n$.

Lemma 7 yields upper bounds for evaluation and inversion with trapdoor:

$$C(\mathrm{Eval}_n) \le \frac{7n}{2} + \alpha n + 1, \qquad C(\mathrm{Inv}_n) \le \frac{5n}{2} + \frac{3\alpha n}{2} - 2.$$

Lemma 6 yields

$$C_{1/2}(\mathrm{Adv}_n) \ge \frac{13n}{4} + \frac{3\alpha n}{2} - 7.$$

Therefore, to get a feebly trapdoor function we simply need to choose α such that

$$\frac{13}{4} + \frac{3\alpha}{2} > \frac{7}{2} + \alpha, \qquad \frac{13}{4} + \frac{3\alpha}{2} > \frac{5}{2} + \frac{3\alpha}{2}.$$

The second inequality is trivial, and the first one yields $\alpha > \frac{1}{2}$.

We would like to maximize the order of security of this trapdoor function (Definition 1); since sampling is always strictly faster than evaluation and inversion with trapdoor, we are maximizing

$$\min\left\{ \lim_{n\to\infty} \frac{C_{1/2}(\mathrm{Adv}_n)}{C(\mathrm{Inv}_n)}, \lim_{n\to\infty} \frac{C_{1/2}(\mathrm{Adv}_n)}{C(\mathrm{Eval}_n)} \right\} = \min\left\{ \frac{\frac{13}{4} + \frac{3\alpha}{2}}{\frac{5}{2} + \frac{3\alpha}{2}}, \frac{\frac{13}{4} + \frac{3\alpha}{2}}{\frac{7}{2} + \alpha} \right\}.$$

This expression reaches maximum when $\alpha = 2$, and the order of security in this case reaches $\frac{25}{22}$. We summarize this in the following theorem.

Theorem 1 *There exists a feebly trapdoor function with the seed length* $\mathrm{pi}(n) = \mathrm{ti}(n) = n$*, the length of inputs and outputs* $c(n) = m(n) = 3n$*, and the order of security* $\frac{25}{22}$*.*

8 A trapdoor function with a security guarantee

In the previous sections, we saw a construction of a linear feebly trapdoor function that guarantees that any circuit with less than precisely the necessary number of gates fails to invert this function on more that $\frac{1}{2}$ of its inputs.

In this section, we use this construction to design a function with a superpolynomial bound on the probability of success (namely, $2^{-c\sqrt{n}+o(\sqrt{n})}$). Let us denote by h the function an adversary had to compute in the previous section, and by X its matrix.

Consider the linear function H defined by the block diagonal matrix:

$$H\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m)}\right) = \begin{pmatrix} X & 0 & \ldots & 0 \\ 0 & X & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & X \end{pmatrix} \begin{pmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(m)} \end{pmatrix}.$$

By Lemma 5, H has complexity at least $m(C(h)-1)$. The dimensions of X are $(1+\alpha)n \times (2+\alpha)n$; we denote $n' = (1+\alpha)n$.

Lemma 8 *Fix a polynomial p. If a circuit computes H on more that $\frac{1}{p(m)}$ fraction of its inputs, and for each block in H:*

- *all columns of X are different;*
- *every row of X has at least $u(n)$ nonzero entries;*
- *after removing any $t(n)$ columns of X, this matrix still has at least one row containing at least two nonzero entries,*

then the complexity of this circuit is not less than $(u(n)+t(n))(m-\log p(m))$.

Proof. First, recall that H consists of m separate blocks with disjoint sets of variables X_i; let us denote $h_i = H\,|_{X_i}$. Since X_i are disjoint, mistakes in computing h_i are independent: if a circuit C computes h_i on β_i fraction of its inputs and h_j on β_j fraction of its inputs, it cannot compute H on more that $\beta_i\beta_j$ fraction of its inputs. Thus, there are at most $\log p(m)$ blocks where C can afford to make more than $\frac{1}{2}$ mistakes. During this proof we will call them "terrible" blocks.

After this remark we proceed by the same gate elimination induction we had been using several times already. Consider the topmost gate and the two variables that enter it. In the proof of Lemma 4, we marked variables as "good" or "bad" depending on whether they fall into a block where all "good" variables have been eliminated. This time, we do the same thing, but mark all variables in terrible blocks as "bad" in advance.

As in the previous proofs, whenever the topmost gate has at least one "bad" variable as input, we set this variable, thus eliminating only one gate from the circuit. Whenever the topmost gate has two "good" variables as inputs, we should always be able to eliminate two gates from the circuit. Let us elaborate on that a little bit.

There are still the same basic three possibilities as in Lemma 3. However, this time we are not able to substitute the precise value that we wish to substitute. This time we have to stick with the half of the inputs where the circuit errs on less than half of the remaining inputs (since the initial circuit errs on less than $\frac{1}{2}$ of these inputs, such a subcurcuit should exist). However, we can unite the first and the third cases with no loss of generality: a formula may depend exclusively on $x \wedge y$ not more than on $x \oplus y$; both cases lead to the same $\frac{1}{2}$ error rate after the same arguments.

The rest of the proof is the same as Lemma 3. We proceed by induction, eliminating two gates whenever two "good" variables enter a topmost gate, and eliminating one "bad" variable whenever it enters a topmost gate. Thus, the overall complexity is at least twice the number of "good" variables plus the number of remaining "bad" variables. We have to discard "terrible" blocks completely — after all, their complexity may actually be equal to zero. Thus, we get a bound of $(t+u)(m-\log p(m))$. □

Note also that stacking the matrices up in a large block diagonal matrix does not change the parameters of a feebly trapdoor function. Thus, we have obtained the following theorem.

Theorem 2 *There exists a feebly trapdoor function candidate* $\mathcal{C} = \{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ *with the seed length* $\mathrm{pi}(n) = \mathrm{ti}(n) = n$*, the length of inputs and outputs* $c(n) = m(n) = 3n$ *with complexities* $C(\mathrm{Inv}_n) \le \frac{11n}{2} + O(1)$*,* $C(\mathrm{Eval}_n) \le \frac{11n}{2} + O(1)$*,* $C(\mathrm{Key}_n) = n+1$ *and the order of security* $\frac{25}{22}$*.*

Moreover, no adversary with $< \frac{25}{4}n - \frac{5}{2}\delta\sqrt{n}$ *gates is able to invert this feebly trapdoor function on more than* $2^{-\delta\sqrt{n}+o(\sqrt{n})}$ *fraction of the inputs for any constant* $\delta > 0$*.*

9 Conclusion and further work

In this work, we have presented a new construction of a feebly secure cryptographic primitive. It is the first known construction of a provably secure trapdoor function, although "security" should be understood in a very restricted sense of Definition 1. Here is the list of possible further research directions:

1. Devise a more natural construction. Both the second (keyless) construction and the block-diagonal merge are counter-intuitive.
2. Substantially improve the order of security. While a certain improvement to the constant $\frac{25}{22}$ seems to be straightforward (for example, instead of A', one can use another Hiltgen's matrix that has the order of security approaching 2 instead of $\frac{3}{2}$), it is clear that the gate elimination technique will not lift us above linear bounds, so optimizing the constant (staying under 2) does not seem worthwhile. This is a general obstacle in (general) circuit complexity, and we feel that a major breakthrough should happen before nonlinear lower bounds for the general model are proven.
3. Devise other feebly secure cryptographic primitives.

References

[Blu84] Norbert Blum. A boolean function requiring $3n$ network size. *Theoretical Computer Science*, 28:337–345, 1984.

[GHP08] Dima Grigoriev, Edward A. Hirsch, and Konstantin Pervyshev. A complete public-key cryptosystem. *Groups, Complexity, Cryptology*, 1(1), 2008.

[Hil92] Alain P. Hiltgen. Constructions of freebly-one-way families of permutations. In *Proc. of AsiaCrypt '92*, pages 422–434, 1992.

[HKN+05] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfers and other primitives. In *Proc. of EuroCrypt'05, LNCS*, volume 3494, pages 96–113, 2005.

[Lev87] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[LS73] E. A. Lamagna and J. E. Savage. On the logical complexity of symmetric switching functions in monotone and complete bases. Technical report, Brown University, Rhode Island, jul 1973.

[Sav76] John E. Savage. *The Complexity of Computing.* Wiley, New York, 1976.

[Weg87] Ingo Wegener. *The Complexity of Boolean Functions.* B. G. Teubner, and John Wiley & Sons, 1987.