Записки научных
семинаров ПОМИ
Том 399, 2012 г.

E. A. Hirsch, O. Melanich, S. I. Nikolenko

FEEBLY SECURE CRYPTOGRAPHIC PRIMITIVES

Abstract. In 1992, A. Hiltgen [9] provided 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 the 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 schemes, while public-key schemes are constructed using trapdoor functions. We continue Hiltgen's work by providing examples of *feebly secure trapdoor functions* where the adversary is guaranteed to spend more time than honest participants (also by a constant factor). We give both a (simpler) linear and a (better) non-linear construction.

§1. Introduction

Modern cryptography has virtually no provably secure constructions. Starting from the first Diffie–Hellman key agreement protocol [4] and the first public-key cryptosystem RSA [20], not a single public-key cryptographic protocol has been proven secure (naturally, there exist secure secret key protocols, e.g., the one-time pad scheme [22,24]). An unconditional proof of security would be hard to find indeed, since it would necessarily imply that $\mathrm{P} \neq \mathrm{NP}$.

There are complete cryptographic constructions, both one-way functions [15] and public-key cryptosystems [7] (see also [6]). However, the conditional results they provide are not related to widely believed assumptions of computational complexity (like $\mathrm{P} \neq \mathrm{NP}$). Moreover, the asymptotic nature of these completeness results does not permit us to say anything about

Key words and phrases: Feeble security, circuit complexity, trapdoor functions, provable security.

This article is a joint journal version of papers [12, 17]. The work has been supported by the Russian Fund for Basic Research, grants no. 11-01-00760-a and 11-01-12135-ofi-m-2011, the Russian Presidential Grant Programme for Leading Scientific Schools, and the Russian Presidential Grant Programme for Young Ph.D.'s grant no. MK-6628.2012.1.

how hard is it to break a given cryptographic protocol for keys of a specific length, which is exactly what one needs in practice.

At present, there is no hope to prove the security either in this "hard" sense or in the sense of classical cryptographic definitions [5]. But if we are unable to prove a superpolynomial gap between the complexities of honest parties and adversaries, perhaps we can prove at least *some* gap? In 1992, Alain Hiltgen [9] managed to present a function that is almost *twice* $(2 - o(1)$ times) 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 [14, 21]: 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. We have already noted that little more could be expected for this model at present. For example, the best known lower bound for the general circuit complexity of a specific Boolean function is $3n - o(n)$ (see [2, 25]).

In this work, we construct another feebly secure cryptographic primitive: namely, we present constructions of feebly secure trapdoor functions. 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, which has been known from the 1970s and which has been used in proving virtually all known bounds in general circuit complexity [2, 18, 23]. New methods would be of great interest; alas, there has been little progress in general circuit complexity since Blum's result of 1984 [2]. Over the latest years, efforts in circuit complexity have been relocated mostly towards results related to circuits with bounded depth and/or restricted set of functions computed in a node (see, e.g., [1, 8, 13, 19]). However, in our work we allow the most general $\mathbb{B}_{2,1}$ basis and do not restrict the depth of an adversary; a restricted adversary makes little sense in the cryptographic setting.

We do not introduce new techniques; thus, our main results are the constructions. We give both a (simpler) linear and a (better) non-linear constructions. Both our constructions consist of two parts, two functions combined into one by direct sum. For the first of these functions, the adversary's task (following a cryptographic tradition, we will call him Charlie)

is harder than the task of a sender encoding the message (Bob), and for the second function the adversary is worse off than the decoding receiver (Alice). We will show that if one part of the message is encoded in the first way, and another in the second, then Charlie's task will be harder than the task of both Alice and Bob. Exactly speaking, the inversion complexity for an adversary will be at least $\frac{7}{5}$ (respectively, $\frac{25}{22}$) times higher than the complexities of key generation, function evaluation, and inversion with trapdoor in our nonlinear (resp., linear) construction.

The paper is organized as follows. In Section 2, we give basic definitions of the notions used in this chapter. In Section 3, we establish certain combinatorial properties of the matrices we are working with (candidates for hard functions). In Section 4, we present the basic gate elimination method that will be used in further sections to prove lower bounds, and apply it to linear feebly one-way functions; in Section 5, we show a nonlinear feebly one-way construction. In Section 6 we present the constructions of a linear and a nonlinear feebly secure trapdoor function (family of circuits). In Sections 7 and 8 we prove their security. Section 9 concludes the paper. This work is a joint journal version of [12, 17].

§2. Definitions

Boolean circuits (see, e.g., [25]) are one of the few computational models that allow for proving *specific* rather than asymptotic lower bounds on the complexity. In this model, a function's complexity is defined as the minimal size of a circuit computing this function. Circuits consist of *inputs* and *gates*, and gates can implement various Boolean functions.

We 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 with two elements.

Definition 1. *Let Ω be a set of Boolean functions $f : \mathbb{B}^m \to \mathbb{B}$ (m may differ for different f). Then an* Ω-circuit *is a directed acyclic labeled graph with vertices of two kinds:*

- *vertices of indegree* 0 *(vertices that no edges enter) labeled by one of the variables $x_1, \ldots, x_n$, and*
- *vertices labeled by a function $f \in \Omega$ with indegree equal to the arity of f.*

Vertices of the first kind are called inputs *or input variables; vertices of the second kind,* gates. *The* size *of a circuit is the number of gates in it.*

Each gate of an Ω-circuit computes some Boolean function. The circuit complexity of a function $f : \mathbb{B}^n \to \mathbb{B}^m$ in the basis Ω is denoted by $C_\Omega(f)$ and is defined as the minimal size of an Ω-circuit that computes f (that has m *output* gates which compute the result of applying function f to input bits). We use circuits with arbitrary binary gates (this is known as *general circuit complexity*), i.e., every gate of a circuit is labeled by one of 16 Boolean functions from $\mathbb{B}_{2,1}$. To avoid negation gates, we allow also to define an output as the negation of the value obtained in a gate; to avoid identity gates, we also allow to define an output as the value of a variable or the negation of it. It is easy to see that constant gates 0 and 1 can also be eliminated, i.e., we can assume that all gates in a circuit compute Boolean functions that depend non-trivially on both inputs. In what follows, we denote by $C(f)$ the circuit complexity of f in this model.

We want the size of circuits breaking our constructions to be larger than the size of circuits that implement honest parties. For one-way functions, that means comparing computing to inverting. Following Hiltgen [9–11], for every injective function of n variables $f_n \in \mathbb{B}_{n,m}$ we can define its *measure of one-wayness* as

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

Hiltgen's work was to find sequences of functions $f = \{f_n\}_{n=1}^{\infty}$ with a large asymptotic constant $\lim_{l\to\infty} \inf_{n>l} M_F(f_n)$, which Hiltgen calls f's *order of one-wayness*. (He considers the case $m = n$; it remains an open problem to see if considering $m > n$ and thus injective and not bijective functions may help.) We will discuss his results in more detail in Section 4.

In this context, we have to give a more detailed definition of a trapdoor function than the regular cryptographic definition [5]: since we are interested in constants here, we must pay attention to all the details. The next definition does not say anything about the complexity of evaluation and the hardness of inversion, but merely sets up the dimensions.

Definition 2. *For given functions* $\mathrm{pi}, \mathrm{ti}, m, c : \mathbb{N} \to \mathbb{N}$, *a* feebly trapdoor candidate *is a sequence of triples of circuits*

$$\mathcal{C} = \{(\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}, \text{ where:}$$

- $\{\mathrm{Key}_n\}_{n=1}^{\infty}$ *is a family of sampling circuits* $\mathrm{Key}_n : \mathbb{B}^n \to \mathbb{B}^{\mathrm{pi}(n)} \times \mathbb{B}^{\mathrm{ti}(n)}$,

- $\{\mathrm{Eval}_n\}_{n=1}^{\infty}$ *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*
- $\{\mathrm{Inv}_n\}_{n=1}^{\infty}$ *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 security parameter n, *every seed* $s \in \mathbb{B}^n$, *and every input* $\mathrm{m} \in \mathbb{B}^{m(n)}$,

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

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 speaking, 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. In our constructions, $m(n) = c(n)$ and $\mathrm{pi}(n) = \mathrm{ti}(n)$.

To find how secure a function is, one needs to know the size of the minimal circuit that could invert the function without knowing the trapdoor information. In addition to the worst-case complexity $C(f)$, we introduce a stronger notion that we will use in this case.

Definition 3. *We denote by* $C_\alpha(f)$ *the minimal size of a circuit that correctly computes a function* $f \in \mathbb{B}_{n,m}$ *on more than* α *fraction of its inputs (of length* n*). Obviously,* $C_\alpha(f) \leqslant C(f)$ *for all* f *and* $0 \leqslant \alpha < 1$.

Definition 4. *A circuit* N breaks *a feebly trapdoor candidate*

$$\mathcal{C} = \{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$$

on seed length n *with probability* α *if, for uniformly chosen seeds* $s \in \mathbb{B}^n$ *and inputs* $m \in \mathbb{B}^{m(n)}$,

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

A size s circuit that breaks a feebly trapdoor candidate

$$\mathcal{C} = \{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$$

on seed length n in the sense of Definition 4 represents a counterexample for the statement $C_\alpha(Inv_n) > s$.

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.

Definition 5. *We say that a feebly trapdoor candidate*

$$\mathcal{C} = \{(\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$$

has order of security k with probability α *if*

$$\lim_{l\to\infty} \inf_{n>l} \min\left\{\frac{C_\alpha(f_{\mathrm{pi}(n)+c(n)})}{C(\mathrm{Key}_n)}, \frac{C_\alpha(f_{\mathrm{pi}(n)+c(n)})}{C(\mathrm{Eval}_n)}, \frac{C_\alpha(f_{\mathrm{pi}(n)+c(n)})}{C(\mathrm{Inv}_n)}\right\} \geqslant k,$$

where the function $f_{\mathrm{pi}(n)+c(n)} \in \mathbb{B}_{\mathrm{pi}(n)+c(n),m(n)}$ *maps*

$$\big(\mathrm{Key}_{n,1}(s), \mathrm{Eval}_n(\mathrm{Key}_{n,1}(s), m)\big) \mapsto m.$$

We say that a feebly trapdoor candidate has order of security k *if it has order of security* k *with probability* $\alpha = \frac{3}{4}$.

Example 1. We begin with simple examples. If there is no secret key at all, that is, $\mathrm{ti}(n) = 0$, then each feebly trapdoor candidate

$$\{(\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$$

has order of security at most 1, since the sequence of circuits $\{\mathrm{Inv}_n\}_{n=1}^{\infty}$ successfully inverts it. If $\{(\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$ implements a trapdoor function in the usual cryptographic sense, then $k = \infty$. Moreover, $k = \infty$ even if every adversary requires a superlinear number of gates. Our definitions are not designed to distinguish between these (presumably very different) cases.

Remark 2. 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 3. 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.

Over the rest of this paper, we develop the constructions of feebly trapdoor functions (that is, feebly trapdoor candidates with a nontrivial order of security). We present two constructions based on two different feebly secure one-way functions, linear and nonlinear.

§3. Matrices of hard functions

Our linear constructions are based on a linear function $f : \mathbb{B}^n \to \mathbb{B}^n$ shown by A. Hiltgen to be feebly one-way of order $\frac{3}{2}$ [9, 10]. We restrict ourselves to the case when $n \equiv 0 \pmod 4$ for reasons that will presently

become clear. Note that Definition 5 carries through this restriction: for $n \not\equiv 0 \pmod 4$ one can simply consider a circuit with input size equal to the lowest multiple of 4 greater than n.

In what follows, all computations are done over $\mathbb{F}_2$. We introduce standard matrix notation:

- $\boldsymbol{e}_k$ denotes the unit $k \times k$ matrix;
- $\boldsymbol{0}_k$, the zero $k \times k$ matrix;
- $\boldsymbol{e}_{ij}$, the matrix whose only nonzero element is at position (i, j);
- $\boldsymbol{e}_{i*}$, the matrix where the i^{th} row consists of 1's, and all other elements are zero;
- $\boldsymbol{e}_{*j}$, the matrix where the j^{th} column consists of 1's, and all other elements are zero;
- $\boldsymbol{1}_k$, the $k \times k$ matrix filled with 1's;
- $\boldsymbol{u}_k$, the upper triangular $k \times k$ matrix ($u_{ij} = 1 \Leftrightarrow i < j$);
- $\boldsymbol{l}_k$, the lower triangular $k \times k$ matrix ($l_{ij} = 1 \Leftrightarrow i > j$);
- $\boldsymbol{m}_\pi$, the permutation matrix for π ($m_{ij} = 1 \Leftrightarrow j = \pi(i)$).

By $\boldsymbol{e}$, $\boldsymbol{0}$, $\boldsymbol{1}$, $\boldsymbol{u}$, and $\boldsymbol{l}$ without subscripts we denote the correspondent matrices of dimension $\frac{n}{2} \times \frac{n}{2}$. We also set $\sigma = \sigma_n$ to be the cyclic permutation $1 \mapsto 2 \mapsto 3 \ldots \mapsto n$.

In this notation the matrix of the Hiltgen's function f is

$$A = \boldsymbol{e}_n + \boldsymbol{m}_\sigma + \boldsymbol{e}_{n,\frac{n}{2}+1}.$$

Lemma 6. *Let $n = 4k$ for some $k \in \mathbb{N}$. Then*

$$\begin{aligned} A^{-1} &= \begin{pmatrix} \boldsymbol{l} & 1 \\ 1 & \boldsymbol{e} + \boldsymbol{u} \end{pmatrix}, \\ A^{-2} &= \boldsymbol{1}_n \boldsymbol{u}_n + \boldsymbol{u}_n \boldsymbol{1}_n + \begin{pmatrix} \boldsymbol{e} + \boldsymbol{u}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{l}^2 \end{pmatrix}. \end{aligned}$$

Proof. Easy calculations. □

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

$$\mathfrak{A} = \begin{pmatrix} A^{-2} & A^{-1} \end{pmatrix}.$$

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

Lemma 7. *Let $n = 4k$ for some $k \in \mathbb{N}$. Then:*

(1) *All columns of $\mathfrak{A}$ (and, hence, A^{-1}) are different.*

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

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

Proof. Let us first interpret the results of Lemma 6. Each row of A contains two ones (on the diagonal and to the right) except for the last row that has three ones, in positions $(n,1)$, $(n,\frac{n}{2}+1)$, and (n,n). Each row of A^{-1} has at least $\frac{n}{2}$ non-zero elements (ones), and the $(\frac{n}{2}+1)^{\text{th}}$ row does not contain a single zero.

The A^{-2} matrix also has lots of ones: $(\mathbf{1}_n\boldsymbol{u}_n+\boldsymbol{u}_n\mathbf{1}_n)$ is an $n\times n$ matrix filled with zeroes and ones chequered, since

$$(\mathbf{1}_n\boldsymbol{u}_n)_{ij}=1 \quad\Leftrightarrow\quad j\equiv 1 \pmod 2,$$
$$(\boldsymbol{u}_n\mathbf{1}_n)_{ij}=1 \quad\Leftrightarrow\quad i\equiv 0 \pmod 2.$$

Moreover,

$$(\boldsymbol{e}+\boldsymbol{u}^2)_{ij}=1 \quad\Leftrightarrow\quad j>i \text{ and } i+j\equiv 0 \pmod 2,$$
$$(\boldsymbol{l}^2)_{ij}=1 \quad\Leftrightarrow\quad i>j \text{ and } i+j\equiv 0 \pmod 2,$$

and thus A^{-2} has two triangular blocks filled with ones: for $1\leqslant i\leqslant j\leqslant\frac{n}{2}$ and for $\frac{n}{2}+1<j<i\leqslant n$. Thus, each row of A^{-2} contains at least $\frac{n}{2}$ ones; moreover, its triangular blocks consisting of ones coincide with the triangular blocks of A^{-1} filled with zeroes, and the rest is covered with zeroes and ones chequered.

We now proceed to proving Lemma 7. The first claim is obvious.

The i^{th} row of A^{-1} contains $\frac{n}{2}+i$ nonzero entries for $i\leqslant\frac{n}{2}$ and $\frac{n}{2}+n-i$ nonzero entries for $i\geqslant\frac{n}{2}$. 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 $i\leqslant\frac{n}{2}$ and at least $\frac{n}{2}+\frac{1}{2}(i-\frac{n}{2}-1)$ nonzero entries for $i\geqslant\frac{n}{2}$. 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 $i\leqslant\frac{n}{2}$ and at least

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

nonzero entries for $i\geqslant\frac{n}{2}$, which proves the second claim.

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 removed 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 (and, to be honest, the only) technique for proving general circuit lower bounds. Hiltgen proved all his bounds with the following very simple argument due to Lamagna and Savage.

Proposition 8 ([14, 21]; [9, 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* $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in \mathbb{B}$ *such that*

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

Then $C(f) \geqslant n - 1$.

(2) *Let* $f = (f^{(1)}, \ldots, f^{(m)}) : \mathbb{B}^n \to \mathbb{B}^m$*, where* $f^{(k)}$ *is the* k^{th} *component of* f*. If the* m *component functions* $f^{(i)}$ *are pairwise different, neither of them equals the negation of another, and each of them satisfies* $C(f^{(i)}) \geqslant c \geqslant 1$ *then* $C(f) \geqslant c + m - 1$.

Proof. (1) Consider the minimal circuit of size s computing f. Since f depends (here and in what follows we say "depends" meaning "depends non-idly") on all n of its variables, each input gate must have at least one outgoing edge. Since the circuit is minimal, each of the other gates, except possibly the output, also must have at least one outgoing edge. Therefore, the circuit has at least $s+n-1$ edges. On the other hand, a circuit with s binary gates cannot have more than $2s$ edges. Therefore, $2s \geqslant s + n - 1$.

(2) Consider a circuit computing f. Note that it has at least $c - 1$ gates that do not compute any function of circuit complexity c or more (they are the first $c-1$ gates in some topological order). Also for each component function $f^{(i)}$ there must be a separate output gate different from those $c - 1$ first gates.

□

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 $A^{-1}\vec{y}$ is $\frac{n}{2}$) and thus produced lower bounds on the complexity of inverting the function (e.g. the complexity of computing $A^{-1}\vec{y}$ is at least $\frac{n}{2} + n - 2 = \frac{3n}{2} - 2$).

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.

Hiltgen proves such result for the square matrix A^{-1}; the error probability is $1/2$. However, in our constructions we also use rectangular 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" general circuit complexity [25].

The basic idea of this method is to use the following inductive argument. Consider a function f and a circuit of minimal size C that computes it. Now substitute some value c for some variable x thus obtaining a circuit for the function $f|_{x=c}$. The original circuit C can now 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 even 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.

First, we prove a prerequisite to the master lemma.

Lemma 9. *Let* $t \geqslant 0$. *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, there are no zero rows in* X, *and after removing any* t *columns of* X, *the matrix still has at least one row containing at least two nonzero entries. Then* $C(\chi) \geqslant t + 1$ *and, moreover, no circuit with less than* $t + 1$ *gates can compute* χ *on more than* $\frac{1}{2}$ *of the inputs.*

Proof. We argue by induction on t. For $t = 0$ the statement is obvious: a circuit with no gates cannot compute $x \oplus y$ on more than $\frac{1}{2}$ of the inputs.

Consider a circuit implementing χ on more than $\frac{1}{2}$ of the inputs and fix 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 more than $\frac{1}{2}$ of the inputs). We have to prove that we can eliminate at least t gates from this circuit, and there still will be at least one gate left.

It cannot be the case that the circuit contains no gates as two different linear functions h and χ cannot coincide on more than $\frac{1}{2}$ of the 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. To eliminate a gate, we simply substitute a value to x; substituting a value for one variable is equivalent to removing a column from the matrix, and it reduces t by at most 1.

To invoke the induction hypothesis, it remains to note that if h coincides with χ on more than $\frac{1}{2}$ of the inputs, then either $h|_{x=0}$ or $h|_{x=1}$ coincides with the corresponding restriction of χ on more than $\frac{1}{2}$ of the remaining inputs. Thus, if h did compute χ on more than $\frac{1}{2}$ of the inputs, substituting this value of x into h would yield a function of $n-1$ inputs that contradicted the induction hypothesis. □

The following is a "master" lemma that we will apply to our matrices.

Lemma 10. *Let $t, u \geqslant 1$. 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, and after removing any t columns of X, the matrix still has at least one row containing at least two nonzero entries. Then $C(\chi) \geqslant u + t$ and, moreover, $C_{3/4}(\chi) \geqslant u + t$.*

Proof. This time, we argue by induction on u. For $u = 1$ the induction base follows from Lemma 9.

Consider a circuit implementing χ on more than $\frac{3}{4}$ of the inputs and fix 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 more than $\frac{3}{4}$ of the 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. Neither of its input variables can be marked as an output, because while $u \geqslant 2$ each row still has at least two variables, and two different linear functions cannot coincide on more than $1/2$ of the inputs.

Let us show that one of the input variables x, y of g enters some other gate. Assume that this is not the case. Then h is a function of neither x nor y but only $g(x, y)$; we show that this cannot be the case for a function computing χ on more than $\frac{3}{4}$ of the inputs. Note that χ depends on x and y separately; in particular, for one of these variables, say x, there exists an output χ_i that depends only on x: $\chi_i = x \oplus \bigoplus_{z \in Z} z$, where $y \notin Z$. we remind that in an optimal circuit every gate is binary and depends non-idly on both inputs. In particular, there exist values a and b such that $g(0, a) = g(1, b)$. Thus, for every assignment of the remaining variables $h_i \neq \chi_i$ either on input strings with $(x = 0, y = a)$ or on input strings with $(x = 1, y = b)$, which makes it wrong on at least $\frac{1}{4}$ of all inputs.

Thus one of the input variables of g, say x, enters some other gate. By setting x to any constant we can eliminate at least 2 gates. To invoke the induction hypothesis, it remains to note that if h coincides with χ on more than $\frac{3}{4}$ of the inputs, then either $h|_{x=0}$ or $h|_{x=1}$ coincides with the corresponding restriction of χ on more than $\frac{3}{4}$ of the remaining inputs.

Thus, we have proven that as long as all rows of X contain more than one non-zero element, there exists a variable such that substituting a value for this variable eliminates at least 2 gates (we call these variables "good"). Substituting a value into a variable is equivalent to removing a column from X. Thus, one substitution reduces u and t by at most 1, and we apply the induction hypothesis.

Applying this induction for $u - 1$ steps gives us at least $2(u - 1)$ gates in total.

After all "good" variables have been eliminated, we begin a new induction, this time on t. It is simple: as long as there is at least one gate, we can remove the topmost gate, one gate per step. Note that we still need to choose the "correct" value for eliminated variables to stay in the half where h coincides with χ on more than $\frac{3}{4}$ of the inputs. 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 [25, Section 10.2]. We show it for our particular case.

Lemma 11. *Assume that a linear function ζ is determined by a block diagonal matrix*

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

and the matrices X_j satisfy the requirements of Lemma 10 with $u_j(n)$ and $t_j(n)$, respectively (the matrices may be non-square and of different dimensions). Then $C(\zeta) \geqslant \sum_{j=1}^{m}(u_j(n)+t_j(n))$ and, moreover, $C_{3/4}(\zeta) \geqslant \sum_{j=1}^{m}(u_j(n)+t_j(n))$.

Proof. We proceed similarly to Lemma 10. 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 $\vec{x}^{(1)}$ and the other one is from $\vec{x}^{(2)}$. Both cases from the proof of Lemma 10 go through smoothly in this situation: in the first case we substitute a value for the good variable, and the second case is impossible for the same reasons.

Thus, eliminating all columns from X_i leads to eliminating at least

$$2(u_i - 1) + (t_i - u_i + 2) = t_i + u_i$$

gates, and we obtain the overall bound of

$$C_{3/4}(\zeta) \geqslant \sum_{j=1}^{m}(u_j + t_j).$$

□

We now formulate the direct consequences of these lemmas and note upper bounds for our specific matrices.

Lemma 12. *Let* $n, n' \equiv 0 \pmod 4$,

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

where A_*^{-1} *denotes a matrix with the same structure as* A^{-1}*, but with dimension* n' *instead of* n*. Then* $C_{3/4}(\alpha) \geqslant \frac{3n}{2} - 2$*,* $C_{3/4}(\alpha_2) \geqslant \frac{13n}{4} - 5$*,* $C_{3/4}(\alpha_*) \geqslant \frac{3n'}{2} + \frac{13n}{4} - 7$*.*

Proof. Follows from Lemmas 10 and 11, by substituting the respective bounds $u(n)$ and $t(n)$ from Lemma 7 (in particular, $t(n) = n - 2$ for the matrix A^{-1} and $t(n) = 2n - 5$ for $\mathfrak{A}$). □

Lemma 13.

(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* $\begin{pmatrix} A^{-1} & A \end{pmatrix}$*.*
(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* $\begin{pmatrix} A^{-1} & A^{-1} \end{pmatrix}$*.*

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, $i = n, n-1, \ldots, \frac{n}{2}$, compute all outputs y_i, $i = n, n-1, \ldots, \frac{n}{2}$ 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 nonlinear feebly secure one-way function

Over the previous two sections, we have discussed *linear* feebly secure one-way functions. However, a *nonlinear* approach can yield better constants.

Our nonlinear feebly trapdoor constructions are based on a feebly one-way function resulting from uniting Hiltgen's linear feebly one-way function with the first computationally asymmetric function of four variables [16]. Consider the following function f_n:

$$\begin{aligned} y_1 &= (x_1 \oplus x_2)x_n \oplus x_{n-1}, && (1)\\ y_2 &= (x_1 \oplus x_2)x_n \oplus x_2, && (2)\\ y_3 &= x_1 \oplus x_3,\\ y_4 &= x_3 \oplus x_4,\\ &\dots\\ y_{n-1} &= x_{n-2} \oplus x_{n-1},\\ y_n &= x_n. \end{aligned}$$

In order to get f_n^{-1}, we sum up all rows except the last one:

$$y_1 \oplus \ldots \oplus y_{n-1} = x_1 \oplus x_2.$$

Further, substituting y_n instead of x_n to (1–2), we find x_2 and x_{n-1}. Then x_{n-2} can be found using y_{n-1} and x_{n-1}, etc. Thus the inverse function looks like

$$\begin{aligned} x_n &= y_n,\\ x_2 &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_2,\\ x_{n-1} &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1,\\ x_{n-2} &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1 \oplus y_{n-1},\\ x_{n-3} &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1 \oplus y_{n-1} \oplus y_{n-2},\\ &\dots\\ x_3 &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1 \oplus y_{n-1} \oplus \ldots \oplus y_4,\\ x_1 &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1 \oplus y_{n-1} \oplus \ldots \oplus y_3. \end{aligned}$$

Lemma 14. *The family of functions* $\{f_n\}_{n=1}^{\infty}$ *is feebly one-way of order* 2.

Proof. It is easy to see that f_n can be computed in $n+1$ gates. Each component function of f_n^{-1}, except for the last one, depends non-trivially of all n variables, and all component functions are different. Therefore, to compute f_n^{-1} we need at least $(n-1)+(n-2) = 2n-3$ gates (since f_n is invertible, Proposition 8 is applicable to f_n and f_n^{-1}). Therefore,

$$M_F(f_n) \geqslant \frac{2n-3}{n+1}.$$

On the other hand, f_n cannot be computed faster than in $n-1$ gates because all component functions f_n are different, and only one of them is trivial (depends on only one variable). At the same time, f_n^{-1} can be computed in $2n-2$ gates: one computes $(y_1 \oplus \ldots \oplus y_{n-1})y_n$ in $n-1$ gates and spends one gate to compute each component function except the last one. We get

$$\frac{2n-3}{n+1} \leqslant M_F(f_n) \leqslant \frac{2n-2}{n-1},$$

which is exactly what we need. □

The following theorem (Theorem 16) states that any circuit with less than precisely the necessary number of gates fails to invert our function on more than $\frac{3}{4}$ of its inputs. We will need it in proving the order of security of our trapdoor construction. First we prove a preparatory lemma.

Lemma 15. *In a circuit implementing* $f_n^{-1}|_{y_{i_1}=a_1,\ldots,y_{i_l}=a_l}$ *where* $l \leqslant n-3$, $n \notin \{i_1,\ldots,i_l\}$ *and* $\forall k \in [1..l]$ $a_{i_k} \in \{0,1,y_n,y_n \oplus 1\}$ *on more than* $\frac{3}{4}$ *of the inputs, each variable (except, possibly,* y_n*) cannot be marked as an output and enters at least one gate.*

Proof. First, note that for $i \neq n$

$$\begin{aligned} x_i|_{y_k=y_n} &= (y_1 \oplus \ldots \oplus y_{k-1} \oplus y_n \oplus y_{k+1} \oplus \ldots \oplus y_{n-1})y_n \oplus \ldots \\ &= (y_1 \oplus \ldots \oplus y_{k-1} \oplus 1 \oplus y_{k+1} \oplus \ldots \oplus y_{n-1})y_n \oplus \ldots = x_i|_{y_k=1}, \end{aligned}$$

if the linear part of x_i doesn't include y_k. Similarly, when one substitutes $y_k = y_n$, while the linear part of x_i depends on y_k, or $y_k = y_n \oplus 1$ the expression for x_i looks like one has substituted the value from $\{0,1\}$ instead of y_k.

(1) Proof by contradiction. We denote the actual function this circuit implements by h. Let $h_k = y_j$ be an output ($j \neq n$). As $l \leqslant n-3$, this output does not depend on at least one variable except for

y_n and y_j. Denote it by y_i. The following possibilities exhaust all possible cases.

(a) $k = n \Rightarrow x_k = y_n$.
Consider two different values of the variable y_n (0 and 1) and fix the values of all remaining variables. For one of the values of y_n, h computes a wrong value because

$$h_n|_{y_n=0} = h_n|_{y_n=1}$$

(as h_n does not depend on y_n) and

$$x_n|_{y_n=0} \neq x_n|_{y_n=1}.$$

Thus h differs from f_n^{-1} at least on $\frac{1}{2}$ of the inputs. Contradiction.

(b) $x_k = (y_i \oplus \ldots)y_n \oplus \bigoplus_{z \in Z} z$, » $y_i \notin Z$.
Consider two different values of the variable y_i (0 and 1). For each fixed value set of the remaining variables, where $y_n = 1$, for one of the values of y_i h computes a wrong value because

$$h_k(y_n = 1, y_i = 0, \ldots) = h_k(y_n = 1, y_i = 1, \ldots),$$

$$x_k(y_n = 1, y_i = 0, \ldots) \neq x_k(y_n = 1, y_i = 1, \ldots),$$

because $x_k|_{y_n=1} = y_i \oplus \ldots$.
Thus, we see that h differs from f_n^{-1} at least on $\frac{1}{4}$ of the inputs. Contradiction.

(c) $x_k = (y_i \oplus \ldots)y_n \oplus y_i \oplus \ldots$.
Similarly we have

$$h_k(y_n = 0, y_i = 0, \ldots) = h_k(y_n = 0, y_i = 1, \ldots),$$

$$x_k(y_n = 0, y_i = 0, \ldots) \neq x_k(y_n = 0, y_i = 1, \ldots),$$

because $x_k|_{y_n=0} = y_i \oplus \ldots$, and come to a contradiction the same way.

(2) Let y_j neither enter any gate nor be an output, i.e. h doesn't depend on y_j at all. By the structure of f_n^{-1} there exists such k that the output $x_k = (y_j \oplus \ldots)y_n \oplus y_j \oplus \ldots$ (for $j \neq 2$ one can take $k = 1$, for $j = 2$ one can take $k = 2$). Then

$$h_k(y_n = 0, y_j = 0, \ldots) = h_k(y_n = 0, y_j = 1, \ldots),$$

$$x_k(y_n = 0, y_j = 0, \ldots) \neq x_k(y_n = 0, y_j = 1, \ldots).$$

Thus h differs from f_n^{-1} at least on $\frac{1}{4}$ of the inputs. Contradiction.

□

Theorem 16. $C_{3/4}(f_n^{-1}) \geqslant 2n-4$.

Proof. Consider an optimal circuit implementing f_n^{-1} on more than $\frac{3}{4}$ of the inputs. We will substitute, step by step, the values from $\{0, 1, y_n, y_n \oplus 1\}$ instead of some variables (except for y_n) to eliminate at least two gates ("good" variables). The following lemma states that while we have at least three variables left and one of them is y_n, we can make "good" substitutions.

Lemma 17. *In a circuit computing $f_n^{-1}|_{y_{i_1}=a_1,\ldots,y_{i_l}=a_l}$ on more than $\frac{3}{4}$ of the inputs, where $l \leqslant n-3$, $n \notin \{i_1, \ldots, i_l\}$, and $\forall k \in [1..l]$ $a_{i_k} \in \{0, 1, y_n, y_n \oplus 1\}$, one can substitute a value from $\{0, 1, y_n, y_n \oplus 1\}$ for some variable other than y_n in such a way that at least two gates are eliminated, and the resulting circuit computes the corresponding restriction of f_n^{-1} on more than $\frac{3}{4}$ of the remaining inputs.*

Proof. We write f_n^{-1} instead of $f_n^{-1}|_{y_{i_1}=a_1,\ldots,y_{i_l}=a_l}$ for brevity. We denote the actual function this circuit implements by h. Let g be a gate whose incoming edges come from the inputs of the circuit (denote them y_i and y_j). Such a gate exists by Lemma 15. The following possibilities exhaust all possible cases.

(1) One of the input variables of g, say y_i, enters some other gate, and $i \neq n$. In this case, by setting y_i to any constant we can eliminate at least 2 gates. Note that if h coincides with f_n^{-1} on more then $\frac{3}{4}$ of the inputs, then either $h|_{y_i=0}$ or $h|_{y_i=1}$ coincides with the corresponding restriction of f_n^{-1} on more than $\frac{3}{4}$ of the remaining inputs.

(2) Neither y_i nor y_j is equal to y_n or enters any other gate. In this case h is a function of neither y_i nor y_j but only $g(y_i, y_j)$ (as neither y_i nor y_j is an output by Lemma 15). We show that it is impossible. By the structure of f_n^{-1} there exists an output $x_k = (y_i \oplus y_j \oplus \ldots)y_n \oplus y_i \oplus \bigoplus_{z \in Z} z$, where $y_j \notin Z$. There are two possible cases.

(a) h_k depends on $g(y_i, y_j)$.

(i) g is linear. Then

$$h_k(y_i = 0, y_j = 1, y_n = 0, \ldots) = h_k(y_i = 1, y_j = 0, y_n = 0, \ldots),$$

$$x_k(y_i = 0, y_j = 1, y_n = 0, \ldots) \neq x_k(y_i = 1, y_j = 0, y_n = 0, \ldots)$$

as well as

$$h_k(y_i = 0, y_j = 0, y_n = 0, \ldots) = h_k(y_i = 1, y_j = 1, y_n = 0, \ldots),$$

$x_k(y_i = 0, y_j = 0, y_n = 0, \ldots) \neq x_k(y_i = 1, y_j = 1, y_n = 0, \ldots)$,

therefore h differs from f_n^{-1} at least on $\frac{1}{4}$ of the inputs. Contradiction.

(ii) g is non-linear. Then there exist such a and b that

$h_k(y_i = a, y_j = 0, y_n = 1, \ldots) = h_k(y_i = a, y_j = 1, y_n = 1, \ldots)$,

$x_k(y_i = a, y_j = 0, y_n = 1, \ldots) \neq x_k(y_i = a, y_j = 1, y_n = 1, \ldots)$

as well as

$h_k(y_i = 0, y_j = b, y_n = 0, \ldots) = h_k(y_i = 1, y_j = b, y_n = 0, \ldots)$,

$x_k(y_i = 0, y_j = b, y_n = 0, \ldots) \neq x_k(y_i = 1, y_j = b, y_n = 0, \ldots)$,

therefore h differs from f_n^{-1} at least on $\frac{1}{4}$ of the inputs. Contradiction.

(b) h_k depends on neither y_i nor y_j. Then

$h_k(y_i = 0, y_n = 0, \ldots) = h_k(y_i = 1, y_n = 0, \ldots)$,

$x_k(y_i = 0, y_n = 0, \ldots) \neq x_k(y_i = 1, y_n = 0, \ldots)$,

therefore h differs from f_n^{-1} at least on $\frac{1}{4}$ of the inputs. Contradiction.

(3) Without loss of generality, $j = n$, y_i does not enter any other gate and g is *non-linear*. We show that it is impossible. Indeed, otherwise one can substitute to y_n some value such that neither of the outputs depends on y_i. By the structure of f_n^{-1}, for each value of y_n one can find an output $x_k = y_i \oplus \ldots$. Then

$h_k(y_i = 0, y_n = a, \ldots) = h_k(y_i = 1, y_n = a, \ldots)$,

$x_k(y_i = 0, y_n = a, \ldots) \neq x_k(y_i = 1, y_n = a, \ldots)$,

therefore h differs from f_n^{-1} at least on $\frac{1}{4}$ of the inputs. Contradiction.

(4) Without loss of generality, $j = n$, y_i does not enter any other gate and g is *linear*. Let g be an output, say h_k. Then either there exists such y_l that for some value of y_n, $x_k = y_l \oplus \ldots$, or $k = n$; both of this cases similarly lead us to a contradiction. Therefore, g has subsequent gates. We make one of the substitutions $y_i = y_n$ or $y_i = y_n \oplus 1$, so that h coincides with the corresponding restriction of f_n^{-1} on more than $\frac{3}{4}$ of the remaining inputs. Thus we eliminate both g and its subsequent gates, i.e. at least 2 gates.

□

By inductive application of Lemma 17 we can substitute $n-2$ "good" variables and eliminate at least $2n-4$ gates. □

Remark 4. It is easy to see that f_n^{-1} equals its linear part on the fraction $\frac{3}{4}$ of the inputs (namely, for $y_n = 0$ and for $\bigoplus_{i=1}^{n-1} y_i = 0$) and can be computed using $n-3$ gates. Therefore the constant $\frac{3}{4}$ in Theorem 16 cannot be improved.

§6. Three constructions

We are almost ready to present the constructions of our feebly trapdoor functions (recall Definition 2). In this section, we consider three different constructions, none of which works by itself; however, we will merge the first and the third (linear case) and the second and the third (nonlinear case) into one in the subsequent section, and the resulting mixture will be feebly secure.

In our first two constructions, inversion with trapdoor is faster than inversion without trapdoor, but, unfortunately, evaluating the function is exactly as hard as inversion without trapdoor in the second construction and even harder in the first. In terms of Definition 2, we now present feebly trapdoor candidates 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 $(\mathrm{pi}, \mathrm{ti})$, where ti is the random seed itself and $\mathrm{pi} = A(\mathrm{ti})$ (thus, the sampler can be implemented using $n+1$ gates).

In the first, linear construction, the output c for an input m is computed as follows:

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

An upper bound on evaluation circuit complexity immediately follows from Lemma 13; one can evaluate this function with a circuit of size $\frac{7n}{2}$. Inversion with trapdoor goes as follows:

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

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 $\mathrm{ti} \oplus c$ using n gates, then one applies A^{-1} using another $\frac{3n}{2} - 1$ gates (see Lemma 13).

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

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

Let us denote the function that the adversary has to compute by Adv_n. Its two arguments are the public information and the codeword. By Lemma 12, 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 $\frac{1}{4}$ of the inputs.

The second (nonlinear) construction is a bit simpler than the linear one. For f as in Section 5 we have:

$$\begin{aligned} \mathrm{Key}_n(s) &= (f_n(s), s), \\ \mathrm{Eval}_n(pi, m) &= f_n^{-1}(pi) \oplus m, \\ \mathrm{Inv}_n(ti, c) &= f_n^{-1}(pi) \oplus c = ti \oplus c, \\ \mathrm{Adv}_n(pi, c) &= f_n^{-1}(pi) \oplus c. \end{aligned}$$

In the first two constructions, evaluation is not easier than inversion without trapdoor. In order to fix this problem, we use also a different construction, a candidate trapdoor function with $c(n) = m(n) = n$ and $\mathrm{pi}(n) = \mathrm{ti}(n) = 0$. Our third construction is just the feebly one-way function itself. Thus, the public and trapdoor information are not used at all. In the linear case, the evaluation and inversion functions are as follows:

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

Similarly, in the nonlinear case we have, for f as in Section 5,

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

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 12 and Theorem 16, can be performed only by circuits of $\frac{3n}{2} - 2$ and $2n - 4$ gates each for the linear and nonlinear versions, respectively. Thus, in this construction evaluation is easy and inversion is hard, both for an honest participant of the protocol and for an adversary.

§7. A linear feebly secure trapdoor function

In the previous section, we have constructed three candidate trapdoor functions. In one of them, evaluation was easy while inversion was hard; in the other two, inversion with trapdoor was easy, and evaluation and inversion without trapdoor were both hard. We now combine the first and third of these functions into a linear construction where it is 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 (nontrivial) construction, while the second part m_2, of length αn, will be subject to the second construction (the one-way function itself). We will choose α later to maximize the relative hardness for an adversary.

We first consider the linear construction. In this construction, each participant has a block-diagonal matrix:

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

where A_* denotes the matrix with the same structure as A, but with dimension αn instead of n. Thus, in terms of Definition 2, 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 13 yields upper bounds for evaluation and inversion with trapdoor, and Lemma 12 yields a lower bound for the adversary:

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

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

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

$$\frac{13}{4}+\frac{3\alpha}{2}>\frac{7}{2}+\alpha \quad \text{and} \quad \frac{13}{4}+\frac{3\alpha}{2}>\frac{5}{2}+\frac{3\alpha}{2}.$$

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 5); since sampling is always strictly faster than evaluation and inversion with trapdoor, we are maximizing

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

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 18. *There exists a linear feebly trapdoor function with seed length* $\mathrm{pi}(n) = \mathrm{ti}(n) = n$*, input and output length* $c(n) = m(n) = 3n$*, and order of security* $\frac{25}{22}$.

Example 2. Let us give the minimal example for which this construction yields a nontrivial security guarantee. As we have already mentioned, all linear constructions in this chapter are based on a linear function $f : \mathbb{B}^n \to \mathbb{B}^n$ developed by A. Hiltgen [9]. The smallest n for which f is strictly harder to invert than to compute is 7, but since we need n to be a multiple of four, we consider $n = 8$ in this example. For $n = 8$, $C(f) = 9$, $C(f^{-1}) = 11$, $C(f^{-2}) = 11$. Lemma 13 yields precise estimates for the complexity of key generation, function evaluation, and its inversion with trapdoor:

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

On the other hand, Lemma 12 implies that

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

§8. A nonlinear feebly secure trapdoor function

In the nonlinear case, we unite the second and the third trapdoor candidates from Section 6 and get the following construction:

$$\begin{aligned}
\mathrm{Key}_n(s) &= (f_n(s), s),\\
\mathrm{Eval}_n(pi, m_1, m_2) &= (f_n^{-1}(pi) \oplus m_1, f_{\alpha n}(m_2)),\\
\mathrm{Inv}_n(ti, c_1, c_2) &= (f_n^{-1}(pi) \oplus c_1, f_{\alpha n}^{-1}(c_2)) = (ti \oplus c_1, f_{\alpha n}^{-1}(c_2)),\\
\mathrm{Adv}_n(pi, c_1, c_2) &= (f_n^{-1}(pi) \oplus c_1, f_{\alpha n}^{-1}(c_2)),
\end{aligned}$$

Upper bounds are easy to obtain. For the lower bound on Adv_n, however, we have to work a bit harder.

We introduce the following notation for Adv_n:

$$\begin{aligned}
(y_1, \ldots, y_n) &= pi,\\
(s_1, \ldots, s_n) &= c_1,\\
Y_n &= \{y_n, y_{n+\alpha n}\},\\
X_1 &= \{y_1, \ldots, y_n, s_1, \ldots, s_n\},\\
(y_{n+1}, \ldots, y_{n+\alpha n}) &= c_2,\\
S &= \{s_1, \ldots, s_n\},\\
X_2 &= \{y_{n+1}, \ldots, y_{n+\alpha n}\}.
\end{aligned}$$

The lower bound on the complexity of Adv_n is proven similarly to the proof for f_n^{-1} (see Theorem 16 and Lemma 15). At each step, we consider a gate g with two incoming input variables and perform a substitution that eliminates one (when substituting a variable from $S \cup Y_n$) or two (for other substitutions) gates in such a way that the circuit still coincides with the corresponding restriction of Adv_n on more than $\frac{3}{4}$ of the inputs. The difference is that we now have two blocks X_1 and X_2 of different size, and the first block computes $f_n^{-1} \oplus c_1$ rather than f_n^{-1}. It turns out that all cases when variables from the same block enter the gate g (except for variables from S) still go through smoothly when we have two blocks. If a variable from S enters g, we substitute a suitable constant instead and eliminate one gate. Other cases (when variables from different blocks enter g) are almost similar to the cases from Lemma 17. There are, however, two subtleties here.

(1) When a block X_r has only one variable x_i not from $Y_n \cup S$ left, we substitute some value instead of it. On one hand, it is useless

already since it may be an output and may not enter any gate; on the other hand, its presence significantly complicates our case-by-case analysis. If there are still variables from $X_r \cap S$ left, then, when we substitute a constant instead of x_i, they all may become outputs. Therefore, in this case we substitute instead of x_i any variable from S (or its negation).

(2) If the inputs of g come from different blocks, and one of them is $y_j \in Y_n$, then the gate g is necessarily linear and, substituting y_j (or its negation) instead of the second variable, we can eliminate two gates. If both inputs of g come from Y_n, we substitute one of them with the other (or its negation).

These arguments are formalized in Lemma 19 and Theorem 20. Lemma 19 states that while we still have at least one block that contains at least two variables not from $Y_n \cup S$ or at least one variable from S, each variable of this block (except, possibly, a variable from Y_n) cannot be an output and enters at least one gate.

Lemma 19. *Consider a restriction* $\mathrm{Adv}_n|_\sigma$ *or* $\mathrm{Adv}_n|_{\sigma \cup \{y_{i_n} = c\}}$*, where:*

(1) $\sigma = \{y_{i_1} = a_1, \ldots, y_{i_l} = a_l\} \cup \{y_{j_1} = a_{l+1}, \ldots, y_{j_m} = a_{l+m}\} \cup \{s_{p_1} = d_1, \ldots, s_{p_u} = d_u\}$;
(2) $l \leqslant n - 3$;
(3) $y_{i_1}, \ldots y_{i_l}, y_{i_n}$ *are in the same block, say* X_r;
(4) $\{y_{i_1}, \ldots y_{i_l}\} \cap Y_n = \emptyset$, $y_{i_n} \in Y_n$ (*at least two variables that do not belong to* $Y_n \cup S$ *remain in* X_r);
(5) $y_{j_1}, \ldots y_{j_m} \in X_t$, *where* $t = 3 - r$;
(6) $\forall k \in [1..l + m]$ $a_k \in \{0,1\} \cup Y_n \cup \{y \oplus 1 \mid y \in Y_n\}$ (*besides, we can substitute for the last variable from* $(X_t \setminus S) \setminus Y_n$ *a value from* $X_t \cap S \cup \{s \oplus 1 \mid s \in X_t \cap S\}$),
(7) $c \in (Y_n \setminus \{y_{i_n}\}) \cup \{y \oplus 1 \mid y \in Y_n \setminus y_{i_n}\}$;
(8) $u \geqslant 0$, $\forall k \in [1..u]$ $d_k \in \{0,1\}$.

Consider a circuit that implements one of these restrictions on more than $\frac{3}{4}$ *of the inputs. Then the following statements hold for this circuit.*

(1) *Each variable of* $X_r \setminus S$ *(except, possibly,* y_{i_n}*) cannot be marked as an output and enters at least one gate.*
(2) *If* $l \leqslant n - 2$ *then each variable from* $S \cap X_r$ *cannot be marked as an output and enters at least one gate.*
(3) *If* $l = n - 2$ *and the value from* $X_r \cap S \cup \{s \oplus 1 \mid s \in X_r \cap S\}$ *is substituted for the last variable of* $(X_r \setminus S) \setminus Y_n$ *then each variable*

from $S \cap X_r$ cannot be marked as an output and enters at least one gate.

Proof. We denote the actual function this circuit implements by h.

(1) The proof of Lemma 15 that dealt with a single block goes through in this case, too. Note that substituting a variable from another block X_t for a variable from X_r does not invalidate the proof of Lemma 15 since it is equivalent to simply renaming a variable in the block X_r. A substitution of the same variable for two variables y_{i_n} and y_{i_k} is equivalent to substituting y_{i_n} for y_{i_k}; similar for more than two variables. A substitution of the same variable (or its negation) for some variables from $X_r \setminus Y_n$ is equivalent to substituting constants and possibly renaming one of the variables. Its easy to see that all cases from the proof of Lemma 15 go through smoothly in this situation. If $S \subset X_r$ it changes nothing. It is also clear that y_{i_k} cannot be an output h_s in the block X_t since x_s does not depend on y_{i_k}.

(2) Let s_k be an output h_l. If x_l does not depend on s_k then h evidently differs from Adv_n at least on $\frac{1}{2}$ of the inputs. Otherwise $l = k$ and $\exists y_s : x_l = (y_s \oplus \ldots)y_p \oplus s_k \oplus \ldots$. Then, for one of the values of y_p, $x_l = y_s \oplus s_k \oplus \ldots$, and h differs from Adv_n at least on $\frac{1}{4}$ of the inputs, which is a contradiction. Suppose that s_k neither enters any gate nor is an output. Then h does not depend on s_k at all. However, there exists an output $x_k = s_k \oplus \ldots$, so h differs from Adv_n at least on $\frac{1}{2}$ of the inputs, and this is a contradiction again.

(3) Let s_k be an output h_l. If x_l does not depend on s_k then h evidently differs from Adv_n at least on $\frac{1}{2}$ of the inputs. Otherwise there are three possible cases.

(a) $x_l = (s_k \oplus \ldots)y_p \oplus s_k \oplus \ldots$. Then

$$h_l(y_p = 1, s_k = 0, \ldots) \neq h_l(y_p = 1, s_k = 1, \ldots),$$

$$x_l(y_p = 1, s_k = 0, \ldots) = x_l(y_p = 1, s_k = 1, \ldots),$$

therefore, h differs from Adv_n at least on $\frac{1}{4}$ of the inputs, and we have arrived to a contradiction.

(b) $x_l = (s_k \oplus \ldots)y_p \oplus \bigoplus_{z \in Z} z \oplus a, Z \subseteq X_r \setminus \{s_k\}, a \in \{0, 1\}$. Then

$$h_l(y_p = 0, s_k = 0, \ldots) \neq h_l(y_p = 0, s_k = 1, \ldots),$$

$$x_l(y_p = 0, s_k = 0, \ldots) = x_l(y_p = 0, s_k = 1, \ldots),$$

therefore, h differs from Adv_n at least on $\frac{1}{4}$ of the inputs, which is a contradiction.

(c) $x_l = (s_l \oplus \ldots)y_p \oplus s_k \oplus \ldots$. This case is similar to the previous one.

Suppose that s_k neither enters any gate nor is an output. Then there exists an output x_l, which is equal to either $s_k \oplus \ldots$ or $(s_k \oplus \ldots)y_p \oplus s_k \oplus \ldots$. Therefore, h differs from Adv_n at least on either $\frac{1}{2}$ or $\frac{1}{4}$ of the inputs, respectively, and this is a contradiction.

□

Theorem 20. $C_{3/4}(\mathrm{Adv}_n) \geqslant 3n + 2\alpha n - 8$.

Proof. We proceed similarly to Theorem 16. When there is only one variable $y_i \notin Y_n \cup S$ left in some block X_r, we do the following.

(1) If there are no variables from S left in X_r, we substitute to y_i such a value that h coincides with the corresponding restriction of Adv_n on more than $\frac{3}{4}$ of the remaining inputs.

(2) If $\exists s \in S \cap X_r$ then we substitute to y_i either s or $s \oplus 1$ so that h coincides with the corresponding restriction of Adv_n on more than $\frac{3}{4}$ of the remaining inputs.

Thus, in what follows we assume that in each block either there are no variables left that do not belong to $Y_n \cup S$ or there are at least two such variables.

Lemma 21. *Consider a restriction of* Adv_n *such that:*

(1) *the values from* $\{0,1\} \cup Y_n \cup \{y \oplus 1 \mid y \in Y_n\}$ *are substituted instead of some variables not from* $Y_n \cup S$*;*

(2) *the values from* $Y_n \cup \{y \oplus 1 \mid y \in Y_n\}$ *are substituted instead of some variables from* Y_n*;*

(3) *the values from* $\{0,1\}$ *are substituted instead of some variables from* S*;*

(4) *at least in one block, either there are at least two variables left* $\notin Y_n \cup S$ *or there is at least one variable from* S*.*

Consider a circuit that implements this restriction on more than $\frac{3}{4}$ *of the inputs. Then one can either*

(1) *substitute a value from* $\{0,1\} \cup Y_n \cup \{y \oplus 1 \mid y \in Y_n\}$ *into some variable not in* $Y_n \cup S$ *so that at least two gates are eliminated, or*

(2) *substitute a value from* $Y_n \setminus z \cup \{y \oplus 1 \mid y \in Y_n \setminus z\}$ *instead of* $z \in Y_n$ *or substitute a constant value to some variable* $s \in S$ *in such a way that at least one gate is eliminated.*

All these substitutions can be made in such a way that the resulting circuit computes the corresponding restriction of Adv_n *on more than* $\frac{3}{4}$ *of the remaining inputs.*

Proof. Let g be a gate whose incoming edges come from the inputs of the circuit. Such a gate exists by Lemma 19. Note that all cases from the proof of Lemma 17 go through smoothly if the inputs are from the same block. If one of the inputs of g is $s \in S$ then we substitute a constant value to s in such a way that h coincides with the corresponding restriction of Adv_n on more than $\frac{3}{4}$ of the remaining inputs. It remains to consider the cases when one of the inputs is in $X_1 \setminus S$ and the other is in X_2 (we denote them by y_i and y_j).

(1) One of the inputs, say y_i, enters some other gate and $i \neq n$. Then y_i is evidently a "good" variable.

(2) Neither y_i nor y_j enter any other gate, and $i \neq n$ and $j \neq n + \alpha n$. We show that it is impossible. By Lemma 19, neither y_i nor y_j is an output (since both blocks contain at least two variables not in $Y_n \cup S$). In this case h is a function of neither y_i nor y_j but only $g(y_i, y_j)$. There exists an output

$$x_k = (y_i \oplus \bigoplus_{v \in V} v) y_n \oplus y_i \oplus \bigoplus_{z \in Z} z$$

such that $y_j \notin Z \cup V$. There are two possible cases.

(a) g is linear. The proof is similar to the one in the case 2 of the Lemma 17.

(b) g is nonlinear. There exists an output

$$x_l = (y_j \oplus \bigoplus_{u \in U} u) y_{n+\alpha n} \oplus \bigoplus_{w \in W} w,$$

where $y_i \notin U \cup W$, $y_j \notin W$ (for brevity, in expressions for x_k and x_l we use y_n and $y_{n+\alpha n}$, respectively. However, we do take into account that in fact y_n can be substituted instead of $y_{n+\alpha n}$, and vice versa.) If h_k or h_l depends on neither y_i nor y_j then we come to a contradiction similar to case 2 of Lemma 17. Otherwise there exist such a and b that

$$h_k(y_i = 0, y_j = a, y_n = 0, \ldots) = h_k(y_i = 1, y_j = a, y_n = 0, \ldots),$$

$$x_k(y_i = 0, y_j = a, y_n = 0, \ldots) \neq x_k(y_i = 1, y_j = a, y_n = 0, \ldots)$$

as well as

$$h_l(y_i = b, y_j = 0, y_{n+\alpha n} = 1, \ldots) = h_l(y_i = b, y_j = 1, y_{n+\alpha n} = 1, \ldots),$$

$$x_l(y_i = b, y_j = 0, y_{n+\alpha n} = 1, \ldots) \neq x_l(y_i = b, y_j = 0, y_{n+\alpha n} = 1, \ldots).$$

Therefore, h differs from Adv_n on at least $\frac{1}{4}$ of the inputs, which is a contradiction.

(3) Without loss of generality, $i \neq n$ and y_i does not enter any other gates, $j = n+\alpha n$, and g is *nonlinear*. We show that it is impossible. Indeed, otherwise there exist such a that

$$h_{n+\alpha n}(y_i = a, y_{n+\alpha n} = 0) = h_{n+\alpha n}(y_i = a, y_{n+\alpha n} = 1),$$

$$x_{n+\alpha n}(y_i = a, y_{n+\alpha n} = 0) \neq x_{n+\alpha n}(y_i = a, y_{n+\alpha n} = 1).$$

Therefore, h differs from Adv_n on at least $\frac{1}{4}$ of the inputs, which is a contradiction.

(4) Without loss of generality, $i \neq n$ and y_i does not enter any other gates, $j = n + \alpha n$ and g is *linear*. First we show that g cannot be an output. Let g be an output, say h_k. The following possibilities exhaust all possible cases.

(a) $k < n$. By the data $\exists y_l \in X_1 : l \neq i$ and $l \neq n$. There exists such a that $x_k|_{y_n=a} = y_l \oplus \ldots$. Then

$$h_k(y_l = 0, y_n = a, \ldots) = h_k(y_l = 1, y_n = a, \ldots),$$

$$x_k(y_l = 0, y_n = a, \ldots) \neq x_k(y_l = 1, y_n = a, \ldots),$$

therefore h differs from Adv_n at least on $\frac{1}{4}$ of the inputs. Contradiction.

(b) $k \geqslant n$. Then $x_k|_{y_i=0} = x_k|_{y_i=1}$ and $h_k|_{y_i=0} \neq h_k|_{y_i=1}$. Therefore h differs from Adv_n at least on $\frac{1}{2}$ of the inputs. Contradiction.

So g has subsequent gates. We substitute to y_i either y_j or $y_j \oplus 1$ so that h coincides with the corresponding restriction of Adv_n on more than $\frac{3}{4}$ of the remaining inputs. Thus we eliminate both g and its subsequent gates, i.e., at least two gates.

(5) $i = n$ and $j = n + \alpha n$. We substitute to y_i either y_j or $y_j \oplus 1$ so that h coincides with the corresponding restriction of Adv_n on more than $\frac{3}{4}$ of the remaining inputs.

□

Thus, on each step we substitute either a "good" variable, a variable from Y_n, or a variable from S. While there exists a block with at least 2 variables not in $Y_n \cup S$ or with at least one variable from S, restrictions of the original circuit obtained by substitutions described in Lemmas 17 and 21 satisfy the assumptions of Lemma 21. Thus, in block X_1 we have $n-2$ "good" variables and n variables such that substituting each of them eliminates at least one gate, and in block X_2 we have $\alpha n - 2$ "good" variables. Therefore, we get a lower bound $2(n-2) + n + 2(\alpha n - 2) = 3n + 2\alpha n - 8$. □

Lemma 22. *The following upper and lower bounds hold for the components of our nonlinear trapdoor construction:*

$$\begin{aligned} C(\mathrm{Key}_n) &\leqslant n+1, \\ C(\mathrm{Eval}_n) &\leqslant 2n-2+n+\alpha n+1 = 3n+\alpha n-1, \\ C(\mathrm{Inv}_n) &\leqslant n+2\alpha n-2, \\ C_{3/4}(\mathrm{Adv}_n) &\geqslant 3n+2\alpha n-8. \end{aligned}$$

Proof. All upper bounds follow from the fact that one can compute f_n by $n+1$ gates, f_n^{-1} by $2n-2$ gates (see Theorem 14), and "$\oplus c_1$" by n gates. The lower bound follows from Theorem 20. □

To maximize the order of security of this trapdoor function (Definition 5), we have to find α that maximizes

$$\begin{aligned} \lim_{l\to\infty} \inf_{n>l} \min\left\{ \frac{C_{3/4}(Adv_n)}{C(Key_n)}, \frac{C_{3/4}(Adv_n)}{C(Eval_n)}, \frac{C_{3/4}(Adv_n)}{C(Inv_n)} \right\} \\ = \min\left\{ \frac{3+2\alpha}{1}, \frac{3+2\alpha}{3+\alpha}, \frac{3+2\alpha}{1+2\alpha} \right\} \\ = \min\left\{ \frac{3+2\alpha}{3+\alpha}, \frac{3+2\alpha}{1+2\alpha} \right\}. \end{aligned}$$

It is easy to see that this expression is maximized for $\alpha = 2$, and the optimal value of the order of security is $\frac{7}{5}$. We summarize this in the following theorem.

Theorem 23. *There exists a nonlinear feebly trapdoor function with seed length* $\mathrm{pi}(n) = \mathrm{ti}(n) = n$, *input and output length* $c(n) = m(n) = 3n$, *and order of security* $\frac{7}{5}$.

§9. Final remarks

After we published our papers [12,17], we were made aware of an earlier result by A. Hiltgen who, in his PhD thesis [10], constructed a nonlinear feebly one-way function of order 2. This function can also be used as a base for our nonlinear trapdoor construction, yielding the same order of security $\frac{7}{5}$ by similar arguments.

In [3, 12, 17], a superpolynomial bound on the success probability of adversary was claimed for adversaries with slightly weaker resources than ones for which the constant probability bound was proven. We have recently found a flaw in the proofs. Thus, these claims (in particular, the results of [12, Section 3.3]) remain open questions both for feebly one-way and feebly trapdoor functions.

Another direction for further research is to develop the notions of other feebly secure primitives. Also orders of security can certainly be improved; however, further progress here will remain very limited until new methods in general circuit complexity emerge.

Acknowledgements. The authors are grateful to Grigory Yaroslavtsev for a useful remark and to Yury Lifshits for references to A. Hiltgen's papers.

References

1. E. Allender, *Circuit Complexity before the dawn of the new millennium.* — In: Proceedings of the 16th Conference on Foundations of Software Technology and Theoretical Computer Science (1996), 1–18.
2. N. Blum, *A boolean function requiring* $3n$ *network size.* — Theoretical Computer Science **28** (1984), 337–345.
3. A. Davydow, S. I. Nikolenko, *Gate elimination for linear runctions and new feebly secure constructions.* — In: Proceedings of the 6th Computer science symposium in Russia, Lecture Notes in Computer Science **6651** (2001), 148–161.
4. W. Diffie, M. Hellman, *New directions in cryptography.* — IEEE Transactions on Information Theory **IT-22** (1976), 644–654.
5. O. Goldreich, *Foundations of Cryptography. Basic Tools.* Cambridge University Press (2001).
6. D. Grigoriev, E. A. Hirsch, and K. Pervyshev, *A complete public-key cryptosystem.* — Groups, Complexity, and Cryptology **1** (2009), 1–12.
7. D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen, *On robust combiners for oblivious transfers and other primitives.* — In: *Proceedings of EuroCrypt* 05, *Lecture Notes in Computer Science* **3494** (2005), 96–113.

8. J. Hastad, *Computational Limitations for Small Depth Circuits.* — MIT Press, Cambridge, MA (1987).
9. A. P. Hiltgen, *Constructions of Feebly-One-Way Families of Permutations.* — In: *Proc. of AsiaCrypt'92* (1992), pp. 422–434.
10. A. P. Hiltgen, *Cryptographically relevant contributions to combinatorial complexity theory.* — In: *ETH Series in Information Processing*, J. L. Massey, Ed., **3**. Konstanz: Hartung–Gorre (1994).
11. A. P. Hiltgen, *Towards a better understanding of one-wayness: facing linear permutations.* — In: *Proceedings of EuroCrypt'98*, Lecture Notes in Computer Science **1233** (1998), 319–333.
12. E. A. Hirsch, S. I. Nikolenko, *A feebly secure trapdoor function.* — In: Proceedings of the 4th Computer Science Symposium in Russia, Lecture Notes in Computer Science **5675** (2009), 129–142.
13. K. Iwama O. Lachish, H. Morizumi, and R. Raz, *An explicit lower bound of* $5n - o(n)$ *for boolean circuits.* — In: Proceedings of the 33rd *Annual ACM Symposium on Theory of Computing* (2001), pp. 399–408.
14. E. A. Lamagna, J. E. Savage, *On the logical complexity of symmetric switching functions in monotone and complete bases.* — Tech. rep., Brown University, Rhode Island, July 1973.
15. L. A. Levin, *One-way functions and pseudorandom generators.* — Combinatorica **7**, No. 4 (1987), 357–363.
16. J. Massey, *The difficulty with difficulty.* — A guide to the transparencies from the EUROCRYPT'96 IACR distinguished lecture. 1996.
17. O. Melanich, *Nonlinear feebly secure cryptographic primitives.* PDMI preprint 12/2009.
18. W. J. Paul, *A* $2.5n$ *lower bound on the combinational complexity of boolean functions.* — SIAM Journal of Computing **6** (1977), 427–443.
19. A. A. Razborov, *Bounded arithmetic and lower bounds.* — In: Feasible Mathematics II, P. Clote and J. Remmel, Eds., Progress in Computer Science and Applied Logic **13**, Birkhäuser (1995), 344–386.
20. R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems.* — Communications of the ACM **21**, No. 2 (1978), 120–126.
21. J. E. Savage, *The Complexity of Computing.* Wiley, New York (1976).
22. C. E. Shannon, *Communication theory of secrecy systems.* — Bell System Technical Journal **28**, No. 4 (1949), 656–717.

23. L. Stockmeyer, *On the combinational complexity of certain symmetric Boolean functions.* — Mathematical Systems Theory **10** (1977), 323–326.
24. G. S. Vernam, *Cipher printing telegraph systems for secret wire and radio telegraphic communications.* — Journal of the IEEE **55** (1926), 109–115.
25. I. Wegener, *The Complexity of Boolean Functions.* B. G. Teubner, and John Wiley & Sons (1987).

St.Petersburg Department
of the Steklov Mathematical Institute,
Fontanka 27,
St.Petersburg 191023,
Russia

E-mail: hirschpdmi.ras.ru
tushkanchik239mail.ru
sergeylogic.pdmi.ras.ru

Поступило 15 января 2012 г.