1

Provably Secure Cryptographic Constructions

Sergey I. Nikolenko
Steklov Mathematical Institute, St. Petersburg
Russia

1. Introduction

1.1 Cryptography: treading uncertain paths

Modern cryptography has virtually no provably secure constructions. Starting from the first Diffie–Hellman key agreement protocol (Diffie & Hellman, 1976) and the first public key cryptosystem RSA (Rivest et al., 1978), not a single public key cryptographic protocol has been proven secure. Note, however, that there exist secure secret key protocols, e.g., the one-time pad scheme (Shannon, 1949; Vernam, 1926); they can even achieve information–theoretic security, but only if the secret key carries at least as much information as the message.

An *unconditional* proof of security for a public key protocol would be indeed hard to find, since it would necessarily imply that P $\neq$ NP. Consider, for instance, a one-way function, i.e., a function such that it is easy to compute but hard to invert. One-way functions are basic cryptographic primitives; if there are no one-way functions, there is no public key cryptography. The usual cryptographic definition requires that a one-way function can be computed in polynomial time. Therefore, if we are given a preimage $y \in f^{-1}(x)$, we can, by definition, verify in polynomial time that $f(y) = x$, so the inversion problem is actually in NP. This means that in order to prove that a function is one-way, we have to prove that P$\neq$NP, a rather daring feat to accomplish. A similar argument can be made for cryptosystems and other cryptographic primitives; for example, the definition of a trapdoor function (Goldreich, 2001) explicitly requires an inversion witness to exist.

But the situation is worse: there are also no *conditional* proofs that might establish a connection between natural structural assumptions (like P$\neq$NP or BPP$\neq$NP) and cryptographic security. Recent developments in lattice-based cryptosystems relate cryptographic security to worst-case complexity, but they deal with problems unlikely to be NP-complete (Ajtai & Dwork, 1997; Dwork, 1997; Regev, 2005; 2006).

An excellent summary of the state of our knowledge regarding these matters was given by Impagliazzo (1995); although this paper is now more than 15 years old, we have not advanced much in these basic questions. Impagliazzo describes five possible worlds – we live in exactly one of them but do not know which one. He shows, in particular, that it may happen that NP problems are hard even on average, but cryptography does not exist (*Pessiland*) or that one-way functions exist but not public key cryptosystems (*Minicrypt*). [1]

[1] To learn the current state of affairs, we recommend to watch Impagliazzo's lecture at the 2009 workshop "Complexity and Cryptography: Status of Impagliazzo's Worlds"; video is available on the web.

Another angle that might yield an approach to cryptography relates to *complete* cryptographic primitives. In regular complexity theory, much can be learned about complexity classes by studying their complete representatives; for instance, one can study any of the numerous well-defined combinatorial NP-complete problems, and any insight such as a fast algorithm for solving any of them is likely to be easily transferrable to all other problems from the class NP. In cryptography, however, the situation is worse. There exist known complete cryptographic constructions, both one-way functions (Kojevnikov & Nikolenko, 2008; 2009; Levin, 1986) and public key cryptosystems (Grigoriev et al., 2009; Harnik et al., 2005). However, they are still mostly useless in that they are not really combinatorial (their hardness relies on enumerating Turing machines) and they do not let us relate cryptographic security to key assumptions of classical complexity theory. In short, it seems that modern cryptography still has a very long way to go to *provably* secure constructions.

1.2 Asymptotics and hard bounds

Moreover, the asymptotic nature of cryptographic definitions (and definitions of complexity theory in general) does not let us say anything about how hard it is to break a given cryptographic protocol for keys of a certain fixed length. And this is exactly what cryptography means in practice. For real life, it makes little sense to say that something is asymptotically hard. Such a result may (and does) provide some intuition towards the fact that an adversary will not be able to solve the problem, but no real guarantees are made: why is RSA secure for 2048-bit numbers? Why cannot someone come up with a device that breaks into all credit cards that use the same protocol with keys of the same length? There are no theoretical obstacles here. In essence, asymptotic complexity is not something one really wants to get out of cryptographic constructions. Ultimately, I do not care whether my credit card's protocol can or cannot be broken in the limit; I would be very happy if breaking my specific issue of credit cards required constant time, but this constant was larger than the size of the known Universe.

The proper computational model to prove this kind of properties is *general circuit complexity* (see Section 2). This is the only computational model that can deal with specific bounds for specific key lengths; for instance, different implementations of Turing machines may differ by as much as a quadratic factor. Basic results in classical circuit complexity came in the 1980s and earlier, many of them provided by Soviet mathematicians (Blum, 1984; Khrapchenko, 1971; Lupanov, 1965; Markov, 1964; Nechiporuk, 1966; Paul, 1977; Razborov, 1985; 1990; Sholomov, 1969; Stockmeyer, 1977; 1987; Subbotovskaya, 1961; 1963; Yablonskii, 1957). Over the last two decades, 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 (Ajtai, 1983; Cai, 1989; Furst et al., 1984; Håstad, 1987; Immerman, 1987; Razborov, 1987; 1995; Smolensky, 1987; Yao, 1985; 1990). However, we need classical results for cryptographic purposes because the bounds we want to prove in cryptography should hold in the most general $\mathbb{B}_{2,1}$ basis. It would be a very bold move to advertise a credit card as "secure against adversaries who cannot use circuits of depth more than 3".

1.3 Feebly secure cryptographic primitives

We cannot, at present, hope to prove security either in the "hard" sense of circuit complexity or in the sense of classical cryptographic definitions (Goldreich, 2001; 2004; Goldwasser & Bellare, 2001). However, if we are unable to prove a superpolynomial gap between the

complexities of honest parties and adversaries, maybe we can prove at least *some* gap? Alain Hiltgen (1992) managed to present a function that is *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 (Lamagna & Savage, 1973; Savage, 1976): 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 (see Section 3.2). 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 general circuit complexity of a specific Boolean function is $3n - o(n)$ (Blum, 1984) even though a simple counting argument proves that there exist plenty of Boolean functions with circuit complexity $\geq \frac{1}{n}2^n$ (Wegener, 1987).

In this chapter, we briefly recount feebly one-way functions but primarily deal with another feebly secure cryptographic primitive: namely, we present constructions of *feebly 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. To do so, we use the *gate elimination* technique which dates back to the 1970s and which has been used in proving virtually every single known bound in general circuit complexity (Blum, 1984; Paul, 1977; Stockmeyer, 1977). New methods would be of great interest; alas, there has been little progress in general circuit complexity since Blum's result of $3n - o(n)$. A much simpler proof has been recently presented by Demenkov & Kulikov (2011), but no improvement has been found yet.

We begin with linear constructions; in the linear case, we can actually nail gate elimination down to several well-defined techniques that we present in Section 3.3. These techniques let us present linear feebly trapdoor functions; the linear part of this chapter is based mostly on (Davydow & Nikolenko, 2011; Hirsch & Nikolenko, 2008; 2009). For the nonlinear case, we make use of a specific nonlinear feebly one-way function presented in (Hirsch et al., 2011; Melanich, 2009).

2. Basic definitions

2.1 Boolean circuits

Boolean circuits (see, e.g., (Wegener, 1987)) represent 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 *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.*

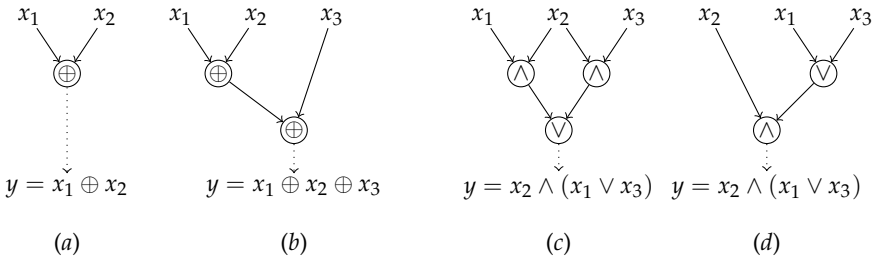

Fig. 1. Simple circuits: (*a*) $y = x_1 \oplus x_2$; (*b*) $y = x_1 \oplus x_2 \oplus x_3$; (*c*) a suboptimal circuit for $y = x_2 \wedge (x_1 \vee x_3)$; (*d*) an optimal one.

We usually speak of *outputs* of a circuit and draw them on pictures, but in theory, every gate of an Ω-circuit computes some Boolean function and can be considered as an output of the circuit. 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 gates which compute the result of applying function f to input bits).

In order to get rid of unary gates, we will assume that a gate computes both its corresponding function and its negation (the same applies to the inputs, too). Our model of computation is given by Boolean circuits with arbitrary binary gates (this is known as *general circuit complexity*); in other words, each gate of a circuit is labeled by one of 16 Boolean functions from $\mathbb{B}_{2,1}$. Several simple examples of such circuits are shown on Fig. 1.

In what follows, we denote by $C(f)$ the circuit complexity of f in the $\mathbb{B}_{2,1}$ basis that consists of all binary Boolean functions. We assume that each gate in this circuit depends of both inputs, i.e., there are no gates marked by constants and unary functions Id and $\neg$. This can be done without loss of generality because such gates are easy to exclude from a nontrivial circuit without any increase in its size.

2.2 Feebly secure one-way functions

We want the size of circuits breaking our family of trapdoor functions to be larger than the size of circuits that perform encoding. Following Hiltgen (1992; 1994; 1998), for every injective function of n variables $f_n \in \mathbb{B}_{n,m}$ we can define its *measure of one-wayness* as

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

The problem now becomes to find sequences of functions $f = \{f_n\}_{n=1}^{\infty}$ with a large asymptotic constant $\liminf_{n\to\infty} M_F(f_n)$, which Hiltgen calls f's *order of one-wayness*.

Hiltgen (1992; 1994; 1998) presented several constructions of feebly secure one-way functions. To give a flavour of his results, we recall a sample one-way function. Consider a function $f : \mathbb{B}^n \to \mathbb{B}^n$ given by the following matrix:

$$f(x_1, \ldots, x_n) = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & 0 & \cdots & 1 & 1 \\ 1 & 0 & 0 & \cdots & 1 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}, \tag{2}$$

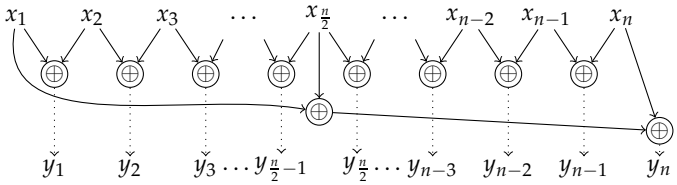

Fig. 2. Hiltgen's feebly one-way function of order $\frac{3}{2}$: a circuit for f.

that is (we assume for simplicity that n is even),

$$f_j(x_1,\dots,x_n) = \begin{cases} x_j \oplus x_{j+1}, & j = 1,\dots,n-1, \\ x_1 \oplus x_{\frac{n}{2}} \oplus x_n, & j = n. \end{cases} \tag{3}$$

Straighforward computations show that f is invertible, and its inverse is given by

$$f^{-1}(y_1,\dots,y_n) = \begin{pmatrix} 0\,0\,\dots\,0\,1\,1\,\dots\,1\,1 \\ 1\,0\,\dots\,0\,1\,1\,\dots\,1\,1 \\ 1\,1\,\dots\,0\,1\,1\,\dots\,1\,1 \\ \vdots\;\vdots\quad\vdots\;\vdots\;\vdots\quad\vdots\;\vdots \\ 1\,1\,\dots\,0\,1\,1\,\dots\,1\,1 \\ 1\,1\,\dots\,1\,1\,1\,\dots\,1\,1 \\ 1\,1\,\dots\,1\,0\,1\,\dots\,1\,1 \\ \vdots\;\vdots\quad\vdots\;\vdots\;\vdots\quad\vdots\;\vdots \\ 1\,1\,\dots\,1\,0\,0\,\dots\,0\,1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{\lfloor \frac{n}{2} \rfloor} \\ y_{\frac{n}{2}+1} \\ y_{\frac{n}{2}+2} \\ \vdots \\ y_n \end{pmatrix}, \tag{4}$$

that is,

$$f_j^{-1}(y_1,\dots,y_n) = \begin{cases} \left(y_1 \oplus \dots \oplus y_{j-1}\right) \oplus \left(y_{\frac{n}{2}+1} \oplus \dots \oplus y_n\right), & j = 1,\dots,\frac{n}{2}, \\ \left(y_1 \oplus \dots \oplus y_{\frac{n}{2}}\right) \oplus \left(y_{j-1} \oplus \dots \oplus y_n\right), & j = \frac{n}{2}+1,\dots,n. \end{cases} \tag{5}$$

It remains to invoke Proposition 6 (see below) to show that f^{-1} requires at least $\lfloor \frac{3n}{2} \rfloor - 1$ gates to compute, while f can be obviously computed in $n+1$ gates. Fig. 2 shows a circuit that computes f in $n+1$ gates; Fig. 3, one of the optimal circuits for f^{-1}. Therefore, f is a feebly one-way function with order of security $\frac{3}{2}$. For this particular function, inversion becomes strictly harder than evaluation at $n = 7$ (eight gates to compute, nine to invert).

2.3 Feebly trapdoor candidates

In the context of feebly secure primitives, we have to give a more detailed definition of a trapdoor function than the regular cryptographic definition (Goldreich, 2001): since we are interested in constants here, we must pay attention to all the details. The following definition does not say anything about the complexity and 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*

$$\mathcal{C} = \{(\mathrm{Seed}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}, \text{ where:} \tag{6}$$

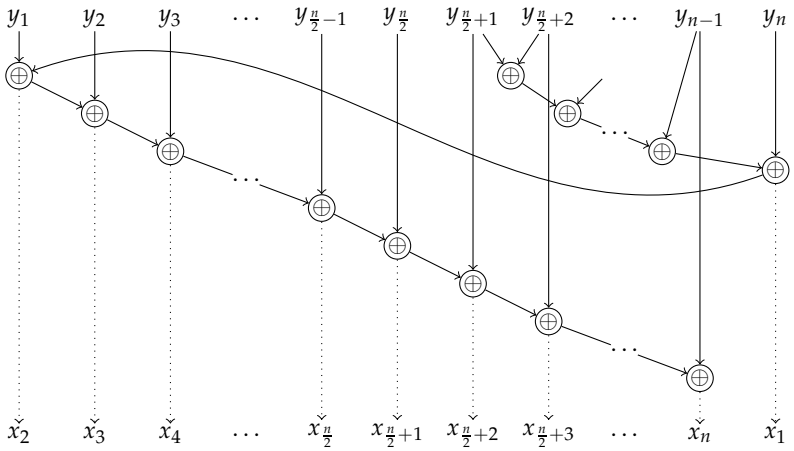

Fig. 3. Hiltgen's feebly one-way function of order $\frac{3}{2}$: a circuit for f^{-1}.

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

$$\mathrm{Inv}_n(\mathrm{Seed}_{n,2}(s), \mathrm{Eval}_n(\mathrm{Seed}_{n,1}(s), \mathrm{m})) = \mathrm{m}, \tag{7}$$

where $\mathrm{Seed}_{n,1}(s)$ *and* $\mathrm{Seed}_{n,2}(s)$ *are the first* $\mathrm{pi}(n)$ *bits ("public information") and the last* $\mathrm{ti}(n)$ *bits ("trapdoor information") of* $\mathrm{Seed}_n(s)$, *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. We call these functions "candidates" because Definition 2 does not imply any security, it merely sets up the dimensions and provides correct inversion. 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* $f \in \mathcal{B}_{n,m}$ *on more than* α *fraction of its inputs (of length* n*). Obviously,* $C_\alpha(f) \leq C(f)$ *for all* f *and* $0 \leq \alpha \leq 1$.

Definition 4. *A circuit* N breaks *a feebly trapdoor candidate* $\mathcal{C} = \{\mathrm{Seed}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ *on seed length* n *with probability* α *if, for uniformly chosen seeds* $s \in \mathcal{B}^n$ *and inputs* $m \in \mathcal{B}^{m(n)}$,

$$\Pr_{(s,m)\in U}\Big[N(\mathrm{Seed}_{n,1}(s), \mathrm{Eval}_n(\mathrm{Seed}_{n,1}(s), m)) = m\Big] > \alpha. \tag{8}$$

A size s circuit that breaks a feebly trapdoor candidate $\mathcal{C} = \{\mathrm{Seed}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ on seed length n in the sense of Definition 4 provides a counterexample for the statement $C_\alpha(Inv_n) > s$.

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. For a trapdoor function to be secure, circuits that break the function should be larger than the circuits computing it. In fact, in our results we can require that every such adversary fails with probability at least $\frac{1}{4}$.

Definition 5. *We say that a feebly trapdoor candidate* $\mathcal{C} = \{(\mathrm{Seed}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$ *has* order of security k with probability α *if*

$$\lim_{n\to\infty} \inf \min \left\{ \frac{C_\alpha(f_{\mathrm{pi}(n)+c(n)})}{C(\mathrm{Seed}_n)}, \frac{C_\alpha(f_{\mathrm{pi}(n)+c(n)})}{C(\mathrm{Eval}_n)}, \frac{C_\alpha(f_{\mathrm{pi}(n)+c(n)})}{C(\mathrm{Inv}_n)} \right\} \geq k, \tag{9}$$

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

$$(\mathrm{Seed}_{n,1}(s), \mathrm{Eval}_n(\mathrm{Seed}_{n,1}(s), m)) \mapsto m. \tag{10}$$

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

Let us first give a few simple examples. If there is no secret key at all, that is, $\mathrm{pi}(n) = 0$, then each feebly trapdoor candidate $\{(\mathrm{Seed}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$ has order of security 1, since the sequence of circuits $\{\mathrm{Inv}_n\}_{n=1}^{\infty}$ successfully inverts it. If $\{(\mathrm{Seed}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$ implement a trapdoor function in the usual cryptographic sense then $k = \infty$. Moreover, $k = \infty$ even if the bounds on the size of adversary are merely superlinear, e.g., if every adversary requires $\Omega(n \log n)$ gates. Our definitions are not designed to distinguish between these (very different) cases, because, unfortunately, any nonlinear lower bound on general circuit complexity of a specific function appears very far away from the current state of knowledge.

One could also 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.

In closing, 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.

3. Gate elimination techniques

3.1 Classical gate elimination

In this section, we first briefly cover classical gate elimination and then introduce a few new ideas related to gate elimination that have recently been presented by Davydow & Nikolenko (2011). Gate elimination is the primary (and, to be honest, virtually the only) technique we have to prove lower bounds in general circuit complexity; so far, it has been used for every single lower bound (Blum, 1984; Paul, 1977; Stockmeyer, 1977; Wegener, 1987). The basic idea of this method lies in 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 $f\mid_{x=c}$. The original circuit C can now be simplified, because the gates that had this variable as inputs become either unary (recall that negation can be embedded into subsequent gates) or constant (in this case we can even proceed to eliminating subsequent gates). After figuring out how many gates one can eliminate on every step, one proceeds by induction as long as it is possible to find a suitable variable that eliminates enough gates. Evidently, the number of eliminated gates is a lower bound on the complexity of f.

Usually, the important case here is when a gate is nonlinear, such as an AND or an OR gate. In that case, it is always possible to choose a value for an input of such a gate so that this gate becomes a constant and, therefore, its immediate descendants can also be eliminated. However, for linear functions this kind of reasoning also works, and in Section 3.3 we distill it to two relatively simple ideas.

To give the reader a flavour of classical gate elimination, we briefly recall the proof of the $2n-3$ lower bound for the functions of the form $f^{(n)}_{3,c} : \mathbb{B}^n \to \mathbb{B}$ defined by

$$f^{(n)}_{3,c}(x_1,\ldots,x_n) = ((x_1+\ldots+x_n+c) \mod 3) \mod 2)\,. \tag{11}$$

This proof can be found in many sources, including (Wegener, 1987). Note that every function $f^{(n)}_{3,c}$ has the following property: for every pair of variables x_j and x_k, $f^{(n)}_{3,c}$ has at least three different restrictions out of four possible assignments of values to x_j and x_k; this is easy to see since different assignments of x_j and x_k give three different values of $x_j + x_k$, resulting in functions with three different constants: $f^{(n-2)}_{3,0}$, $f^{(n-2)}_{3,1}$, and $f^{(n-2)}_{3,2}$. Now consider the topmost gate in some topological order on the optimal circuit computing $f^{(n)}_{3,c}$. Since it is topmost, there are two variables, say x_j and x_k, that come to this gate as inputs. At least one of these variables enters at least one other gate because otherwise, $f^{(n)}_{3,c}$ would depend only on $x_j \oplus x_k$ and not on x_j and x_k separately, giving rise to only two possible subfunctions among four restrictions. Therefore, there exists a variable that enters at least two gates; therefore, by setting this variable to a constant we eliminate at least two gates from the circuit. It remains to note that setting a variable to a constant transforms $f^{(n)}_{3,c}$ into $f^{(n-1)}_{3,c'}$, and we can invoke the induction hypothesis.

3.2 Gate elimination for feebly secure one-way functions

The following very simple argument is due to Lamagna and Savage; this argument actually suffices for all Hiltgen's linear examples.

Proposition 6 ((Lamagna & Savage, 1973; Savage, 1976); (Hiltgen, 1992, 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*

$$f(a_1,\ldots,a_{i-1},0,a_{i+1},\ldots,a_n) \neq f(a_1,\ldots,a_{i-1},1,a_{i+1},\ldots,a_n). \tag{12}$$

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

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 and each of them satisfies $C(f^{(i)}) \geq c \geq 1$ then $C(f) \geq c+m-1$.*

Proof. 1. Consider the minimal circuit of size s computing f. Since f depends (here and in what follows we say "depends" meaning "depends nontrivially") 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, $2s \geq s + n - 1$.

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). However, to compute any component function $f^{(i)}$ we have to add at least one more gate, and we have to add at least one gate for each component, since every new gate adds only one new function. Thus, we get the necessary bound of $c + m - 1$ 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 $\frac{n}{2} + n - 2 = \frac{3n}{2} - 2$).

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

Lemma 7. *Consider a function $f = \bigoplus_{i=1}^{n} x_i$. For any g that depends on only $m < n$ of these variables,*

$$\Pr_{x_1,\ldots,x_n}\left[f(x_1,\ldots,x_n) = g(x_{i_1},\ldots,x_{i_m})\right] = \frac{1}{2}. \tag{13}$$

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. □

This argument suffices for Hiltgen's feebly one-wayness result for the square matrix A^{-1}: first we apply the first part of Proposition 6 and see that every output has complexity at least $\frac{n}{2} - 1$, and then the second part of Proposition 6 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 7, gives the necessary $\frac{1}{2}$ error rate.

3.3 Gate elimination for linear functions

In this section, we deal with gate elimination for *linear* functions. We do not know how to prove that one cannot, in general, produce a smaller circuit for a linear function with nonlinear gates, but it is evident that we cannot assume any gates to be nonlinear in this setting. Thus, gate elimination distills to two very simple ideas. Idea 1 is trivial and has been noted many times before, while Idea 2 will allow us to devise feebly secure constructions in Section 4.

Since we are dealing with linear functions, we will, for convenience, state our results in terms of matrices over $\mathbb{F}_2$; the circuit complexity of a matrix $C_\alpha(A)$ is the circuit complexity of the corresponding linear function. By A_{-i} we denote the matrix A without its i^{th} column; note that if A corresponds to f then A_{-i} corresponds to $f\mid_{x_i=0}$. If a matrix A has a zero column A_i, it means that the corresponding function does not depend on the input x_i; in what follows, we will always assume that functions depend nontrivially on all their inputs and thus the matrices do not have zero columns; we call such matrices *nontrivial*. Note that if A is a submatrix of B then $C_\alpha(A) \leq C_\alpha(B)$ for all $\alpha \in [0,1]$.

Idea 1. Suppose that for n steps, there is at least one gate to eliminate. Then $C(f) \geq n$.

Theorem 8. *Fix a real number $\alpha \in [0,1]$. Suppose that $\mathcal{P} = \{P_n\}_{n=1}^{\infty}$ is a series of predicates defined on matrices over $\mathbb{F}_2$ with the following properties:*

- *if $P_1(A)$ holds then $C_\alpha(A) \geq 1$;*
- *if $P_n(A)$ holds then $P_m(A)$ holds for every $1 \leq m \leq n$;*
- *if $P_n(A)$ holds then, for every index i, $P_{n-1}(A_{-i})$ holds.*

Then, for every matrix A with $\geq n+1$ columns, if $P_n(A)$ holds then $C_\alpha(A) \geq n$.

Proof. The proof goes by straightforward induction on the index of P_i; the first property of $\mathcal{P}$ provides the base, and other properties takes care of the induction step. For the induction step, consider the first gate of an optimal circuit C implementing A. By the monotonicity property of $\mathcal{P}$ and the induction base, the circuit is nontrivial, so there is a first gate. Consider a variable x_i entering that gate. Note that if C computes f on fraction α of its inputs then for some c, $C\mid_{x_i=c}$ computes $f\mid_{x_i=c}$ on fraction α of its inputs. If we substitute this value into this variable, we get a circuit $C\mid_{x_i=c}$ that has at most $(\text{size}(C)-1)$ gates and implements A_{-i} on at least α fraction of inputs. □

Note that the first statement of Proposition 6 is a special case of Theorem 8 for $P_n(A) =$ "A has a row with $n+1$ ones". We also derive another corollary.

Corollary 9. *If A is a matrix of rank n, and each column of A has at least two ones, then $C(A) \geq n-2$.*

Proof. Take $P_n(A) =$"$\text{rank}(A) \geq n+2$ and each column of A has at least 2 ones". □

Idea 2. Suppose that for n steps, there exists an input in the circuit with two outgoing edges, and, moreover, in m of these cases both of these edges go to a gate (rather than a gate and an output). Then $C(f) \geq n+m$.

Theorem 10. *We call a nonzero entry* unique *if it is the only nonzero entry in its row. Fix a real number $\alpha \in [0,1]$. Suppose that $\mathcal{P} = \{P_n\}_{n=1}^{\infty}$ is a series of predicates defined on matrices over $\mathbb{F}_2$ with the following properties:*

- *if $P_1(A)$ holds then $C(A) \geq 1$;*
- *if $P_n(A)$ holds then $P_m(A)$ holds for every $1 \leq m \leq n$;*
- *if $P_n(A)$ holds then, for every index i, if the i^{th} column has no unique entries then $P_{n-2}(A_{-i})$ holds, otherwise $P_{n-1}(A_{-i})$ holds.*

Then, for every matrix A with $\geq n+1$ different columns, if $P_n(A)$ holds for some n then $C(A) \geq n$ and, moreover, $C_{\frac{3}{4}}(A) \geq n$.

Proof. We argue by induction on n; for $n = 1$ the statement is obvious.

Consider the first gate g in the optimal circuit implementing A. Since g is first, its incoming edges come from the inputs of the circuit; we denote them by x_i and x_j. There are three possible cases.

1. One of the input variables of g, say x_i, goes directly to an output y_k. Then by setting x_i to a constant we can eliminate one gate. however, in this case y_k corresponds to a row with only one nonzero element, so ith colum has a unique element, so $P_{n-1}(A_{-i})$ hold. Therefore, we invoke the induction hypothesis as $C(A_{-i}) \geq n-1$ and get the necessary bound.

2. One of the input variables of g, say x_i, goes to another gate. Then by setting x_i to a constant we can eliminate two gates, and by properties of P_n $P_{n-2}(A_{-i})$ holds, so we invoke the induction hypothesis as $C(A_{-i}) \geq n-2$.

3. Neither x_i nor x_j enters any other gate or output. In this case, A is a function of neither x_i nor x_j but only $g(x_i, x_j)$; we show that this cannot be the case for a function computing A on more than $\frac{3}{4}$ of the inputs. A itself depends on x_i and x_j separately because all of its columns are different; in particular, for one of these variables, say x_i, there exists an output y_k that depends only on x_i: $y_k = x_i \oplus \bigoplus_{x \in X} x$, where $x_j \notin X$. On the other hand, since every gate in an optimal circuit nontrivially depends on both inputs, there exist values a and b such that $g(0,a) = g(1,b)$. Thus, for every assignment of the remaining variables, either on input strings with $(x_i = 0, x_j = a)$ or on input strings with $(x_i = 1, x_j = b)$ the circuit makes a mistake, which makes it wrong on at least $\frac{1}{4}$ of all inputs. □

Note that Theorem 10 directly generalizes and strengthens Theorem 8.

Corollary 11. *Fix a real number $\alpha \in [0,1]$. Suppose that $\mathcal{R} = \{R_n\}_{n=1}^{\infty}$ and $\mathcal{Q} = \{Q_m\}_{m=1}^{\infty}$ are two series of predicates defined on matrices over $\mathbb{F}_2$ with the following properties:*

- *if $R_1(A)$ holds then $C(A) \geq 1$;*
- *if $R_n(A)$ holds then $R_k(A)$ holds for every $1 \leq k \leq n$;*
- *if $R_n(A)$ holds then, for every i, $R_{n-1}(A_{-i})$ holds;*
- *if $Q_1(A)$ holds then $C(A) \geq 1$;*
- *if $Q_m(A)$ holds then $Q_k(A)$ holds for every $1 \leq k \leq n$;*
- *if $Q_m(A)$ holds then, for every i, $Q_{m-1}(A_{-i})$ holds;*
- *if $Q_m(A)$ holds and A_{-i} has more zero rows than A (i.e., removing the i^{th} column has removed the last nonzero element from at least one row) then $Q_m(A_{-i})$ holds.*

Then, for every matrix A with $\geq n+1$ columns all of which are different, if $R_n(A)$ and $Q_m(A)$ hold for some $n \geq m$ then $C(A) \geq n+m$ and, moreover, $C_{\frac{3}{4}}(A) \geq n+m$.

Proof. Immediately follows from Theorem 10 for $P_n(A) = \exists k R_k(A) \wedge Q_{n-k}(A)$. □

Theorem 10 and Corollary 11 generalize several results that have been proven independently. For example, here is the "master lemma" of the original paper on feebly trapdoor functions.

Corollary 12 ((Hirsch & Nikolenko, 2009, Lemma 5))**.** *Let $t, u \geq 1$. Assume that χ is a linear function with matrix A over $\mathbb{F}_2$. Assume also that all columns of A are different, every row of A has at least u nonzero entries, and after removing any t columns of A, the matrix still has at least one row containing at least two nonzero entries. Then $C(\chi) \geq u + t$ and, moreover, $C_{3/4}(\chi) \geq u + t$.*

Proof. Take $P_n(A)$ ="After removing any n columns of A, it still has at least one nonzero row", $Q_0(A)$ ="true", and $Q_m(A)$ ="Every row of A has at least $m+1$ ones" for $m > 0$. Then $P_{t+1}(A)$ and $Q_{u-1}(A)$ hold, and $\mathcal{P}$ and $\mathcal{Q}$ satisfy the conditions of Corollary 11, which gives the desired bound. Note that in this case, Q_m for $m > 0$ cannot hold for a matrix where a row has only a single one, so in the gate elimination proof, for the first $u - 1$ steps two gates will be eliminated, and then for $t - u + 2$ steps, one gate will be eliminated. □

We also derive another, even stronger corollary that will be important for new feebly secure constructions.

Corollary 13. *Let $t \geq u \geq 2$. Assume that A is a $u \times t$ matrix with different columns, and each column of A has at least two nonzero elements (ones). Then $C(A) \geq 2t - u$ and, moreover, $C_{\frac{3}{4}}(A) \geq 2t - u$.*

Proof. Take $P_n(A)$ ="twice the number of nonzero columns in A less the number of nonzero rows in A is at least n". Then $P_{2t-u}(A)$ holds, and $\mathcal{P}_n$ satisfy the conditions of Theorem 10. □

Naturally, we could prove Corollaries 9 and 13 directly. We have chosen the path of generalization for two reasons: one, to make Theorem 14 more precise and more general, and two, to show the limits of gate elimination for linear functions. As we have already mentioned, for linear functions we cannot count on nonlinear gates that could eliminate their descendants. In Theorems 8 and 10, we have considered two basic cases: when there is only one edge outgoing from a variable and when there are two edges (going either to two gates or to a gate and an output). It appears that we can hardly expect anything more from classical gate elimination in the linear case.

3.4 Extension to block diagonal matrices

We finish this section with an extension of these results to block diagonal matrices. In general, we cannot prove that the direct sum of several functions has circuit complexity equal to the sum of the circuit complexities of these functions; counterexamples are known as "mass production" (Wegener, 1987). However, for linear functions and gate elimination in the flavours of Theorems 8 and 10, we can. The following theorem generalizes Lemma 6 of (Hirsch & Nikolenko, 2009).

Theorem 14. *Suppose that a linear function χ is given by a block diagonal matrix*

$$\begin{pmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & A_k \end{pmatrix}, \tag{14}$$

every A_j satisfies the conditions of Theorem 10 with predicates $\mathcal{P}^j = \{P_n^j\}_{n=1}^{\infty}$, and $P_{n_j}^j(A_j)$ hold for every j. Then $C(\chi) \geq \sum_{j=1}^{k} n_j$.

Proof. We invoke Theorem 10 with the predicate composed of original predicates:

$$P_n = \bigvee_{i_1+\ldots+i_k=n} P_{i_1}^1 \wedge P_{i_2}^2 \wedge \ldots \wedge P_{i_k}^k. \tag{15}$$

It is now straightforward to check that $\mathcal{P} = \{P_n\}_{n=1}^{\infty}$ satisfies the conditions of Theorem 10 (since every deleted column affects only one block), and the block diagonal matrix satisfies $P_{n_1+\ldots+n_k}$. □

4. Feebly secure trapdoor functions

4.1 Idea of the construction

Over this section, we will present two constructions of feebly secure trapdoor functions, a linear construction and a nonlinear one. Both of them have the same rather peculiar structure. It turns out that when we directly construct a feebly secure candidate trapdoor function such that an adversary has to spend more time inverting it than honest participants, we will not be able to make encoding (i.e., function evaluation) faster than inversion. In fact, evaluation will take *more* time than even an adversary requires to invert our candidates.

To achieve a feebly secure trapdoor function, we will add another block as a direct sum to that candidate. This block will represent a feebly secure one-way function, one of the constructions presented by Hiltgen (1992; 1994; 1998). In this construction, honest inversion and break are exactly the same since there is no secret key at all; nevertheless, both of them are harder than evaluating the function. Thus, in the resulting block diagonal construction break remains harder than honest inversion but they both gain in complexity over function evaluation. This idea was first presented by Hirsch & Nikolenko (2009) and has been used since in every feebly secure trapdoor function.

4.2 Linear feebly secure trapdoor functions

This section is based on (Davydow & Nikolenko, 2011). Let us first introduce some notation. By U_n we denote an upper triangular matrix of size $n \times n$ which is inverse to a bidiagonal matrix:

$$U_n = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \qquad U_n^{-1} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix};$$

note that U_n^2 is an upper triangular matrix with zeros and ones chequered above the main diagonal. We will often use matrices composed of smaller matrices as blocks; for instance, $(\,U_n\ U_n\,)$ is a matrix of size $n \times 2n$ composed of two upper triangular blocks.

Lemma 15. *1.* $C_{\frac{3}{4}}(U_n) = n - 1$.

2. $C_{\frac{3}{4}}(U_n^2) = n - 2$.

3. $C_{\frac{3}{4}}(U_n^{-1}) = n - 1$.

4. $C_{\frac{3}{4}}((\,U_n\ U_n\,)) = 2n-1$.
5. $3n-6 \le C_{\frac{3}{4}}((\,U_n^2\ U_n\,)) \le C((\,U_n^2\ U_n\,)) \le 3n-3$.
6. $3n-4 \le C_{\frac{3}{4}}((\,U_n\ U_n^{-1}\,)) \le C((\,U_n\ U_n^{-1}\,)) \le 3n-2$.

Proof. Lower bounds in items 1–3 are obvious: the matrices have no identical rows, and not a single input except one (two for item 2) is linked directly to an output. The lower bound in item 4 follows by simple counting: the first row of the matrix contains $2n$ nonzero elements, so at least $2n-1$ gates are needed to compute it. The lower bound from item 5 (respectively, 6) follows from Corollary 13: the matrix $(\,U_n^2\ U_n\,)$ (respectively, $(\,U_n\ U_n^{-1}\,)$) satisfies the assumptions of Corollary 13 for all except three (respectively, two) columns, and we can use Corollary 13 for $t = 2n-3$ (respectively, $t = 2n-2$) and $u = n$.

To prove upper bounds, we give direct constructions. To compute the matrix from item 1, note that each row differs from the previous one in only one position, so we can compute the outputs as $\mathrm{out}_i = \mathrm{out}_{i+1} \oplus \mathrm{in}_i$. Moreover, $\mathrm{out}_n = \mathrm{in}_n$, so we do not need more gates to compute it. The same idea works for item 2, but in this case, out_n and out_{n-1} are computed immediately, and $\mathrm{out}_i = \mathrm{out}_{i-2} \oplus \mathrm{in}_i$. To compute the matrix from item 3, we compute each row directly. To compute item 4, we note that $(\,U_n\ U_n\,) \cdot \left(\begin{smallmatrix} a \\ b \end{smallmatrix}\right) = U_n \cdot a \oplus U_n \cdot b = U_n \cdot (a \oplus b)$. Thus, we can use n gates to compute $a \oplus b$ and then get the result with $n-1$ more gates. To compute 5 and 6 note that $(\,A\ B\,) \cdot \left(\begin{smallmatrix} a \\ b \end{smallmatrix}\right) = A \cdot a \oplus B \cdot b$. Thus, we have divided the computation in two parts that can be done independently with previously shown circuits, and then we can use n gates to XOR the results of these subcircuits. □

We use the general idea outlined in Section 4.1. In the first construction, we assume that the lengths of the public key pi, secret key ti, message m, and ciphertext c are the same and equal n. Let $ti = U_n \cdot pi$, $c = (\,U_n^{-1}\ U_n\,) \cdot \left(\begin{smallmatrix} m \\ pi \end{smallmatrix}\right)$. In this case, an adversary will have to compute the matrix $(\,U_n\ U_n\,) \cdot \left(\begin{smallmatrix} c \\ ti \end{smallmatrix}\right) = (\,U_n\ U_n^2\,) \cdot \left(\begin{smallmatrix} c \\ pi \end{smallmatrix}\right)$. Thus, breaking this trapdoor function is harder than honest inversion, but the evaluation complexity is approximately equal to the complexity of the break, so we cannot yet call this function a feebly secure trapdoor function.

To augment this construction, consider a weakly one-way linear function A and use it in the following protocol (by I_n we denote the unit matrix of size n):

$$\begin{aligned}
\mathrm{Seed}_n &= \begin{pmatrix} U_n & 0 \\ 0 & I_n \end{pmatrix} \cdot (\,s\ s\,) = \begin{pmatrix} t_i \\ p_i \end{pmatrix}, \\
\mathrm{Eval}_n &= \begin{pmatrix} U_n^{-1} & U_n & 0 \\ 0 & 0 & A \end{pmatrix} \cdot \begin{pmatrix} m_1 \\ pi \\ m_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \\
\mathrm{Inv}_n &= \begin{pmatrix} U_n & U_n & 0 \\ 0 & 0 & A^{-1} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ ti \\ c_2 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}.
\end{aligned}$$

An adversary is now supposed to compute

$$\mathrm{Adv}_n = \begin{pmatrix} U_n & U_n^2 & 0 \\ 0 & 0 & A^{-1} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ pi \\ c_2 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}.$$

As a feebly one-way function A we take one of Hiltgen's functions with order of security $2-\epsilon$ that have been constructed for every $\epsilon > 0$ Hiltgen (1992); we take the matrix of this function to have order λn, where λ will be chosen below. For such a matrix, $C_{\frac{3}{4}}(A) = \lambda n + o(n)$, and

$C_{\frac{3}{4}}(A^{-1}) = (2-\epsilon)\lambda n + o(n)$. Now Lemma 15 and Theorem 14 yield the following complexity bounds:

$$\begin{aligned} C_{\frac{3}{4}}(\mathrm{Seed}_n) &= n-1, \\ C_{\frac{3}{4}}(\mathrm{Eval}_n) &= 3n + \lambda n + o(n) &&= (3+\lambda)n + o(n), \\ C_{\frac{3}{4}}(\mathrm{Inv}_n) &= 2n + (2-\epsilon)\lambda n + o(n) &&= (2+(2-\epsilon)\lambda)n + o(n), \\ C_{\frac{3}{4}}(\mathrm{Adv}_n) &= 3n + (2-\epsilon)\lambda n + o(n) &&= (3+(2-\epsilon)\lambda)n + o(n). \end{aligned}$$

The order of security for this protocol is

$$\lim_{n\to\infty}\left(\min\left(\frac{C_{3/4}(\mathrm{Adv}_n)}{C(\mathrm{Eval}_n)}, \frac{C_{3/4}(\mathrm{Adv}_n)}{C(\mathrm{Inv}_n)}, \frac{C_{3/4}(\mathrm{Adv}_n)}{C(\mathrm{Seed}_n)}\right)\right) = \\ = \min\left(\frac{3+(2-\epsilon)\lambda}{3+\lambda}, \frac{3+(2-\epsilon)\lambda}{2+(2-\epsilon)\lambda}\right).$$

This expression reaches maximum for $\lambda = \frac{1}{1-\epsilon}$, and this maximum equals $\frac{5-4\epsilon}{4-\epsilon}$, which tends to $\frac{5}{4}$ as $\epsilon \to 0$. Thus, we have proven the following theorem.

Theorem 16. *For every $\epsilon > 0$, there exists a linear feebly secure trapdoor function with seed length $\mathrm{pi}(n) = \mathrm{ti}(n) = n$, input and output length $c(n) = m(n) = 2n$, and order of security $\frac{5}{4} - \epsilon$.*

4.3 Nonlinear feebly secure trapdoor functions

Over the previous two sections, we have discussed *linear* feebly secure one-way functions. However, a *nonlinear* approach can yield better constants. This section is based on (Hirsch et al., 2011; Melanich, 2009).

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 (Massey, 1996). Consider a sequence of functions $\{f_n\}_{n=1}^{\infty}$ given by the following relations (we denote $y_j = f_j(x_1, \ldots, x_n)$):

$$\begin{aligned} y_1 &= (x_1 \oplus x_2)x_n \oplus x_{n-1}, \\ y_2 &= (x_1 \oplus x_2)x_n \oplus x_2, \\ y_3 &= x_1 \oplus x_3, \\ y_4 &= x_3 \oplus x_4, \\ &\ldots \\ y_{n-1} &= x_{n-2} \oplus x_{n-1}, \\ y_n &= x_n. \end{aligned} \tag{16}$$

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

$$y_1 \oplus \ldots \oplus y_{n-1} = x_1 \oplus x_2. \tag{17}$$

Further, substituting y_n instead of x_n, we find x_2 and x_{n-1}. The other x_k can be expressed via x_{n-1} in turn, so the inverse function is given by

$$\begin{aligned} x_n &= y_n, \\ x_2 &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_2, \\ x_{n-1} &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1, \\ x_{n-2} &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1 \oplus y_{n-1}, \\ x_{n-3} &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1 \oplus y_{n-1} \oplus y_{n-2}, \\ &\ldots \\ x_3 &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1 \oplus y_{n-1} \oplus \ldots \oplus y_4, \\ x_1 &= (y_1 \oplus \ldots \oplus y_{n-1})y_n \oplus y_1 \oplus y_{n-1} \oplus \ldots \oplus y_3. \end{aligned} \tag{18}$$

Lemma 17. *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 6 is applicable to f_n and f_n^{-1}). Therefore,

$$M_F(f_n) \geq \frac{2n-3}{n+1}. \tag{19}$$

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

$$\frac{2n-3}{n+1} \leq M_F(f_n) \leq \frac{2n-2}{n-1}, \tag{20}$$

which is exactly what we need. □

For the proof of the following theorem, we refer to (Hirsch et al., 2011; Melanich, 2009).

Theorem 18. $C_{3/4}(f_n^{-1}) \geq 2n-4$.

We can now apply the same direct sum idea to this nonlinear feebly one-way function. The direct sum consists of two blocks. First, for f as above, we have:

$$\begin{aligned} \mathrm{Key}_n(s) &= (f_n(s), s), \\ \mathrm{Eval}_n(pi, m) &= f_n^{-1}(pi) \oplus m, \\ \mathrm{Inv}_n(ti, c) &= f_n^{-1}(pi) \oplus c = ti \oplus c, \\ \mathrm{Adv}_n(pi, c) &= f_n^{-1}(pi) \oplus c. \end{aligned} \tag{21}$$

In this construction, evaluation is no easier than inversion without trapdoor.

For the second block we have

$$\begin{aligned} \mathrm{Eval}_n(m) &= f(m), \\ \mathrm{Inv}_n(c) &= f^{-1}(c), \\ \mathrm{Adv}_n(c) &= f^{-1}(c). \end{aligned} \tag{22}$$

Again, as above, it 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 Theorem 18, can be performed only by circuits with at least $2n-4$ gates. Thus, in this construction evaluation is easy and inversion is hard, both for an honest participant of the protocol and for an adversary.

We can now unite these two trapdoor candidates and get the following construction:

$$\begin{aligned} \mathrm{Key}_n(s) &= (f_n(s), s), \\ \mathrm{Eval}_n(pi, m_1, m_2) &= (f_n^{-1}(pi) \oplus m_1, f_{\alpha n}(m_2)), \\ \mathrm{Inv}_n(ti, c_1, c_2) &= (f_n^{-1}(pi) \oplus c_1, f_{\alpha n}^{-1}(c_2)) = (ti \oplus c_1, f_{\alpha n}^{-1}(c_2)), \\ \mathrm{Adv}_n(pi, c_1, c_2) &= (f_n^{-1}(pi) \oplus c_1, f_{\alpha n}^{-1}(c_2)), \end{aligned} \tag{23}$$

The proofs of lower bounds on these constructions are rather involved; we refer to (Hirsch et al., 2011; Melanich, 2009) for detailed proofs and simply give the results here.

Lemma 19. *The following upper and lower bounds hold for the components of our nonlinear trapdoor construction:*

$$\begin{aligned} C(\mathrm{Key}_n) &\le n+1, \\ C(\mathrm{Eval}_n) &\le 2n-2+n+\alpha n+1 = 3n+\alpha n-1, \\ C(\mathrm{Inv}_n) &\le n+2\alpha n-2, \\ C_{3/4}(\mathrm{Adv}_n) &\ge 3n+2\alpha n-8. \end{aligned} \tag{24}$$

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

$$\begin{aligned} \liminf_{i\to\infty} \min\left\{ \frac{C_{3/4}(Adv_n)}{C(Key_n)}, \frac{C_{3/4}(Adv_n)}{C(Eval_n)}, \frac{C_{3/4}(Adv_n)}{C(Inv_n)} \right\} = \\ = \min\left\{ \frac{3+2\alpha}{1}, \frac{3+2\alpha}{3+\alpha}, \frac{3+2\alpha}{1+2\alpha} \right\} = \min\left\{ \frac{3+2\alpha}{3+\alpha}, \frac{3+2\alpha}{1+2\alpha} \right\}. \end{aligned} \tag{25}$$

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 20. *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* $\frac{7}{5}$.

5. Conclusion

In this chapter, we have discussed recent developments in the field of feebly secure cryptographic primitives. While these primitives can hardly be put to any practical use at present, they are still important from the theoretical point of view. As sad as it sounds, this is actually the frontier of provable, mathematically sound results on security; we do not know how to prove anything stronger.

Further work in this direction is twofold. One can further develop the notions of feebly secure primitives. Constants in the orders of security can probably be improved; perhaps, other primitives (key agreement protocols, zero knowledge proofs etc.) can find their feebly secure counterparts. This work can widen the scope of feebly secure methods, but the real breakthrough can only come from one place.

It becomes clear that cryptographic needs call for further advances in general circuit complexity. General circuit complexity has not had a breakthrough since the 1980s; nonconstructive lower bounds are easy to prove by counting, but constructive lower bounds remain elusive. The best bound we know is Blum's lower bound of $3n - o(n)$ proven in 1984. At present, we do not know how to rise to this challenge; none of the known methods seem to work, so a general breakthrough is required for nonlinear lower bounds on circuit complexity. The importance of such a breakthrough can hardly be overstated; in this chapter, we have seen only one possible use of circuit lower bounds.

6. Acknowledgements

This work has been partially 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, grant no. NSh-3229.2012.1, and the Russian Presidential Grant Programme for Young Ph.D.Ss, grant no. MK-6628.2012.1.

7. References

Ajtai, M. (1983). σ_1^1-formulae on finite structures, *Annals of Pure and Applied Logic* 24: 1–48.

Ajtai, M. & Dwork, C. (1997). A public-key cryptosystem with worst-case/average-case equivalence, *Proceedings of the* 29^{th} *Annual ACM Symposium on Theory of Computing*, pp. 284–293.

Blum, N. (1984). A boolean function requiring 3*n* network size, *Theoretical Computer Science* 28: 337–345.

Cai, J. (1989). With probability 1, a random oracle separates PSPACE from the polynomial-time hierarchy, *Journal of Computer and System Sciences* 38: 68–85.

Davydow, A. & Nikolenko, S. I. (2011). Gate elimination for linear functions and new feebly secure constructions, *Proceedings of the* 6^{th} *Computer Science Symposium in Russia, Lecture Notes in Computer Science*, Vol. 6651, pp. 148–161.

Demenkov, E. & Kulikov, A. (2011). An elementary proof of a 3n-o(n) lower bound on the circuit complexity of affine dispersers, *Proceedings of the* 36^{th} *International Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science*, Vol. 6907, pp. 256–265.

Diffie, W. & Hellman, M. (1976). New directions in cryptography, *IEEE Transactions on Information Theory* IT-22: 644–654.

Dwork, C. (1997). Positive applications of lattices to cryptography, *Proceedings of the* 22^{nd} *International Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science*, Vol. 1295, pp. 44–51.

Furst, M., Saxe, J. & Sipser, M. (1984). Parity, circuits, and the polynomial-time hierarchy, *Mathematical Systems Theory* 17: 13–27.

Goldreich, O. (2001). *Foundations of Cryptography. Basic Tools*, Cambridge University Press.

Goldreich, O. (2004). *Foundations of Cryptography II. Basic Applications*, Cambridge University Press.

Goldwasser, S. & Bellare, M. (2001). *Lecture Notes on Cryptography*, Summer course on cryptography at MIT.

Grigoriev, D., Hirsch, E. A. & Pervyshev, K. (2009). A complete public-key cryptosystem, *Groups, Complexity, and Cryptology* 1: 1–12.

Harnik, D., Kilian, J., Naor, M., Reingold, O. & Rosen, A. (2005). On robust combiners for oblivious transfers and other primitives, *Proceedings of EuroCrypt âĂŹ05, Lecture Notes in Computer Science*, Vol. 3494, pp. 96–113.

Håstad, J. (1987). *Computational Limitations for Small Depth Circuits*, MIT Press, Cambridge, MA.

Hiltgen, A. P. (1992). Constructions of feebly-one-way families of permutations, *Proc. of AsiaCrypt '92*, pp. 422–434.

Hiltgen, A. P. (1994). Cryptographically relevant contributions to combinatorial complexity theory, *in* J. L. Massey (ed.), *ETH Series in Information Processing*, Vol. 3, Konstanz: Hartung-Gorre.

Hiltgen, A. P. (1998). Towards a better understanding of one-wayness: Facing linear permutations, *Proceedings of EuroCrypt '98, Lecture Notes in Computer Science*, Vol. 1233, pp. 319–333.

Hirsch, E. A., Melanich, O. & Nikolenko, S. I. (2011). Feebly secure cryptographic primitives.

Hirsch, E. A. & Nikolenko, S. I. (2008). A feebly secure trapdoor function, PDMI preprint 16/2008.

Hirsch, E. A. & Nikolenko, S. I. (2009). A feebly secure trapdoor function, *Proceedings of the* 4^{th} *Computer Science Symposium in Russia, Lecture Notes in Computer Science*, Vol. 5675, pp. 129–142.

Immerman, M. (1987). Languages which capture complexity classes, *SIAM Journal of Computing* 4: 760–778.

Impagliazzo, R. (1995). A personal view of average-case complexity, *Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, IEEE Computer Society, Washington, DC, USA, p. 134.

Khrapchenko, V. M. (1971). Complexity of the realization of a linear function in the class of π-circuits, *Mat. Zametki* 9(1): 36–40.

Kojevnikov, A. A. & Nikolenko, S. I. (2008). New combinatorial complete one-way functions, *Proceedings of the* 25^{th} *Symposium on Theoretical Aspects of Computer Science*, Bordeaux, France, pp. 457–466.

Kojevnikov, A. A. & Nikolenko, S. I. (2009). On complete one-way functions, *Problems of Information Transmission* 45(2): 108–189.

Lamagna, E. A. & Savage, J. E. (1973). On the logical complexity of symmetric switching functions in monotone and complete bases, *Technical report*, Brown University, Rhode Island.

Levin, L. A. (1986). Average case complete problems, *SIAM Journal of Computing* 15(1): 285–286.

Lupanov, O. B. (1965). On a certain approach to the synthesis of control systems – the principle of local coding, *Problemy Kibernet.* 14: 31–110.

Markov, A. A. (1964). Minimal relay-diode bipoles for monotonic symmetric functions, *Problems of Cybernetics* 8: 205–212.

Massey, J. (1996). The difficulty with difficulty: A guide to the transparencies from the EUROCRYPT'96 IACR distinguished lecture.

Melanich, O. (2009). Nonlinear feebly secure cryptographic primitives, PDMI preprint 12/2009.

Nechiporuk, E. I. (1966). A Boolean function, *Soviet Mathematics. Doklady* 7: 999–1000.

Paul, W. J. (1977). A $2.5n$ lower bound on the combinational complexity of boolean functions, *SIAM Journal of Computing* 6: 427–443.

Razborov, A. A. (1985). Lower bounds on monotone complexity of the logical permanent, *Mat. Zametki* 37(6): 887–900.

Razborov, A. A. (1987). Lower bounds on the size of bounded depth circuits over a complete basis with logical addition, *Mat. Zametki* 41(4): 598–608.

Razborov, A. A. (1990). Lower bounds of the complexity of symmetric boolean functions of contact-rectifier circuit, *Mat. Zametki* 48(6): 79–90.

Razborov, A. A. (1995). Bounded arithmetic and lower bounds, *in* P. Clote & J. Remmel (eds), *Feasible Mathematics II*, Vol. 13 of *Progress in Computer Science and Applied Logic*, Birkhäuser, pp. 344–386.

Regev, O. (2005). On lattices, learning with errors, random linear codes, and cryptography, *Proceedings of the* 37^{th} *Annual ACM Symposium on Theory of Computing*, pp. 84–93.

Regev, O. (2006). Lattice-based cryptography, *Proceedings of the* 26^{th} *Annual International Cryptology Conference (CRYPTO'06), Lecture Notes in Computer Science*, Vol. 4117, pp. 131–141.

Rivest, R. L., Shamir, A. & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM* 21(2): 120–126.

Savage, J. E. (1976). *The Complexity of Computing*, Wiley, New York.

Shannon, C. E. (1949). Communication theory of secrecy systems, *Bell System Technical Journal* 28(4): 656–717.

Sholomov, L. A. (1969). On the realization of incompletely-defined boolean functions by circuits of functional elements, *Trans: System Theory Research* 21: 211–223.

Smolensky, R. (1987). Algebraic methods in the theory of lower bounds for boolean circuit complexity, *Proceedings of the* 19^{th} *Annual ACM Symposium on Theory of Computing*, pp. 77–82.

Stockmeyer, L. (1977). On the combinational complexity of certain symmetric boolean functions, *Mathematical Systems Theory* 10: 323–326.

Stockmeyer, L. (1987). Classifying the computational complexity of problems, *Journal of Symbolic Logic* 52: 1–43.

Subbotovskaya, B. A. (1961). Realizations of linear functions by formulas using $\vee$, &, $\neg$, *Soviet Mathematics. Doklady* 2: 110–112.

Subbotovskaya, B. A. (1963). On comparison of bases in the case of realization of functions of algebra of logic by formulas, *Soviet Mathematics. Doklady* 149(4): 784–787.

Vernam, G. S. (1926). Cipher printing telegraph systems for secret wire and radio telegraphic communications, *Journal of the IEEE* 55: 109–115.

Wegener, I. (1987). *The Complexity of Boolean Functions*, B. G. Teubner, and John Wiley & Sons.

Yablonskii, S. V. (1957). On the classes of functions of logic algebra with simple circuit realizations, *Soviet Math. Uspekhi* 12(6): 189–196.

Yao, A. C.-C. (1985). Separating the polynomial-time hierarchy by oracles, *Proceedings of the* 26^{th} *Annual IEEE Symposium on the Foundations of Computer Science*, pp. 1–10.

Yao, A. C.-C. (1990). On ACC and threshold circuits, *Proceedings of the* 31^{st} *Annual IEEE Symposium on the Foundations of Computer Science*, pp. 619–627.