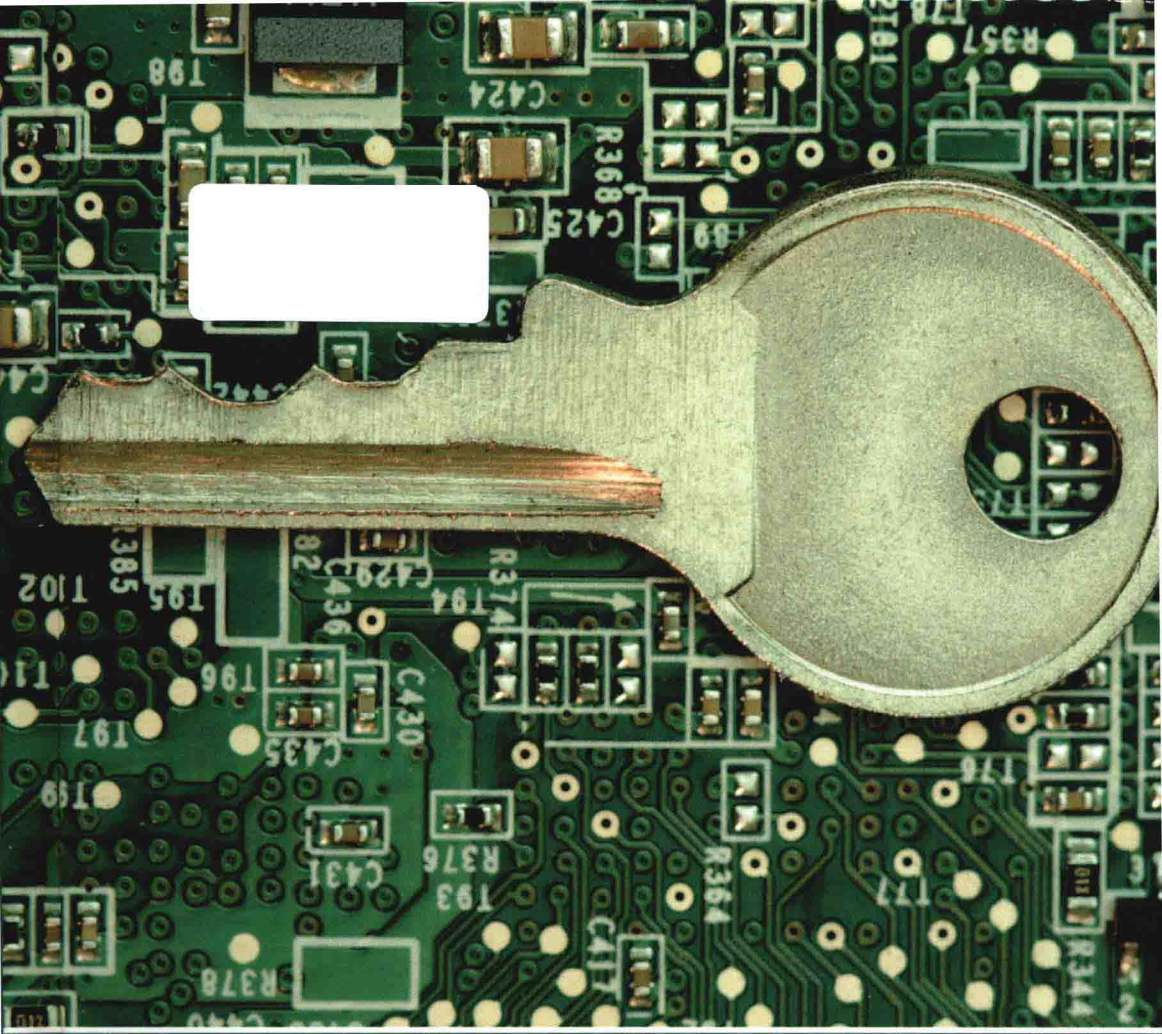

Computing Security & Cryptography Handbook

Stephen Mason

Computing Security & Cryptography Handbook

Edited by **Stephen Mason**

LANRYE
INTERNATIONAL

New Jersey

Published by Clanrye International,
55 Van Reypen Street,
Jersey City, NJ 07306, USA
www.clanryeinternational.com

Computing Security & Cryptography Handbook
Edited by Stephen Mason

International Standard Book Number: 978-1-63240-113-7 (Hardback)

Printed in China.

Computing Security & Cryptography Handbook

Preface

This book presents some critical security challenges that are faced in today's computing world. It discusses, in detail, the defense mechanisms against these kinds of attacks with the help of classical and modern approaches of cryptography and other defense mechanisms. The book not only deals with theoretical and fundamental aspects of cryptography, but also talks in length about different applications of cryptographic protocols and techniques in designing computing and network security solutions. This book will prove itself to be beneficial for researchers, engineers, graduate and doctoral students who are working in the same and related field.

The information contained in this book is the result of intensive hard work done by researchers in this field. All due efforts have been made to make this book serve as a complete guiding source for students and researchers. The topics in this book have been comprehensively explained to help readers understand the growing trends in the field.

I would like to thank the entire group of writers who made sincere efforts in this book and my family who supported me in my efforts of working on this book. I take this opportunity to thank all those who have been a guiding force throughout my life.

Editor

Contents

Part 1

Cryptography and Security in Computing

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.*

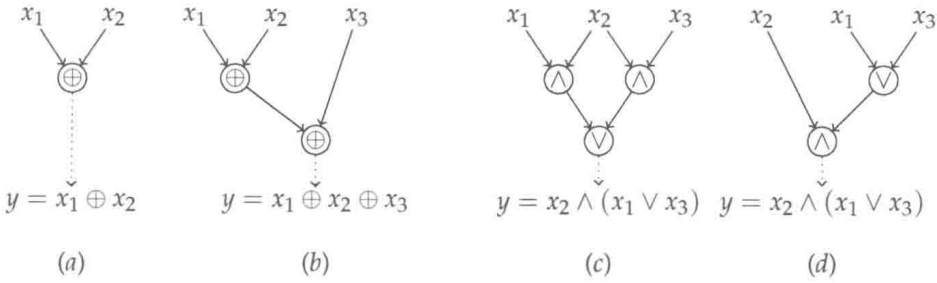

Fig. 1. Simple circuits: (*a*) $y = x_1 \oplus x_2$; (*b*) $y = x_1 \oplus x_2 \oplus x_3$; (*c*) a suboptimal circuit for $y = x_2 \wedge (x_1 \vee x_3)$; (*d*) an optimal one.

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:

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

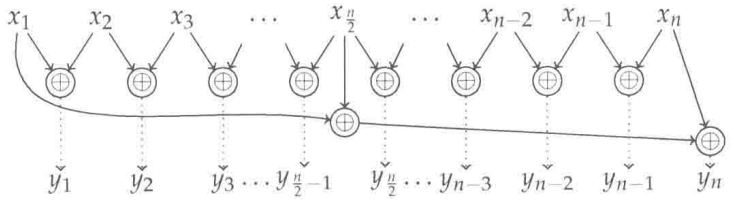

Fig. 2. Hiltgen's feebly one-way function of order $\frac{3}{2}$: a circuit for f.

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

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

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

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

that is,

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

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* pi, ti, $m, c : \mathbb{N} \to \mathbb{N}$, *a* feebly trapdoor candidate *is a sequence of triples of circuits*

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

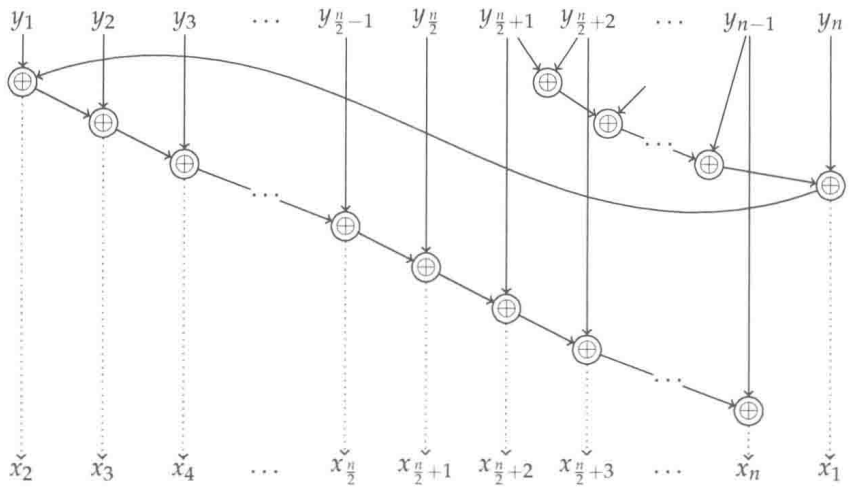

Fig. 3. Hiltgen's feebly one-way function of order $\frac{3}{2}$: a circuit for f^{-1}.

- $\{\mathrm{Seed}_n\}_{n=1}^{\infty}$ *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,* $C_\alpha(f) \le C(f)$ *for all* f *and* $0 \le \alpha \le 1$.

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* $m \in \mathcal{B}^{m(n)}$,

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