Electronic Colloquium on Computational Complexity, Report No. 82 (2011)

Secure Computation with Information Leaking to an Adversary

Miklós Ajtai
IBM Research, Almaden Research Center

May 19, 2011

Abstract

Assume that Alice is running a program P on a RAM, and an adversary Bob would like to get some information about the input or output of the program. At each time, during the execution of P, Bob is able to see the addresses of the memory cells involved in the instruction which is executed and the name of the instruction. In addition to this, at certain times, Bob can even see the contents of all of the memory cells involved in the instruction. We will call a time when this happens a compromised time. Bob can choose the compromised times in an adaptive way, that is, immediately before the instruction at time t is executed, Bob, using all of the information at his disposal, can decide whether time t will be compromised or not. The only restriction on his choice is, that among m consecutive instructions there can be at most εm whose time is compromised, where $\varepsilon > 0$ is a small constant. We show that if $m = c\lfloor \log n \rfloor$, where $c > 0$ is a large constant, then for each program P, using n memory cells and time $T = O(\texttt{poly}(n))$, Alice can construct a functionally equivalent program P', such that the probability that Bob gets any nontrivial information about the input of P is negligible, and the time and space requirements of P' grows, compared to P, only by a factor of $\texttt{poly}(\log n)$. We assume that the program P' gets its input in an encoded form, namely each input bit b is encoded by a random $0,1$-sequence of length m whose parity is b. The output bits must be encoded by P' in a similar way.

As part of the proof of the result described above we also construct for all positive integers m, and for all boolean circuits C of size n a functionally equivalent circuit C' of size $O(n\texttt{poly}(m))$ with the following properties. Assume that an adversary can observe each bit going through the wires of the circuit C' independently with a probability of ε, where $\varepsilon > 0$ is a small constant, and each input/output bit of C is encoded by m input/output bits of C' the same way as described above for RAMs. Then, such an adversary, while observing C', can get any information about the input/output of the circuit C only with a probability of ne^{-cm}, where $c > 0$ is a constant.

1 Introduction

1.1 The history of the problem.

The problem of secure computation in the presence of an adversary who has some partial information about the computation was studied for the special case of oblivious computing. In this case the adversary knows the memory access pattern. E.g., for a RAM this means that the adversary knows at each time which memory cells were accessed by the machine at that

ISSN 1433-8092

time. The problem of doing computation relatively efficiently so that such an adversary does not gain any essentially new information was solved first for Turing machines by Pippenger and Fischer [13]. The question for RAMS was formulated by Goldreich in [6], and solutions gradually making the oblivious simulation more efficient and the computational model more realistic were given by Goldreich [6], Ostrovsky [11], [12], Goldreich and Ostrovsky [7], and Ajtai [3], [2]. It is proved in [3] that a program P running on a RAM can be obliviously simulated, with a negligible probability of failure, on a RAM with a random number generator so that the loss of efficiency in terms of time and memory requirements is only a factor of $\log n$, where n is the memory requirement of P. (Independently, another solution was given by Damgård, Meldgaard, and Nielsen for the same problem in [4], where the failure probability is 0.) The present result can be considered as an extension of the result in [3] in the sense that we show that the theorem remains true for a much stronger adversary. From a technical point of view, however, the proof of the present result is completely different from the proof of the results of [3] or the other proofs concerning oblivious simulation.

The general question of computation performed on a device D, in the presence of an adversary who may get some partial information about the inner working of D, got much attention in the last decade. The main motivating force was that various side channel attacks, when the adversary through physical measurements may get some information about what is happening in the device, were realized in practice. Ishai, Sahai, and Wagner considered the case in [9], when the device D is a circuit C and the adversary may find out the values of the bits flowing through certain wires of C. We will call these the compromised wires. Depending on how the compromised wires are selected by the adversary: adaptively, at random etc., and what can be the total number of compromised wires, there are many possibilities. In [9] rigorous foundations of formulating these type of questions are given. Two types of circuits are considered stateless and stateful circuits. The boolean circuits in the usual sense, where we are interested in hiding the input and output form the adversary and after each evaluation of the circuit no information remains in its wires or gates are the stateless circuits. In this paper we are interested only in stateless circuits so we will refer to them simply as circuits.

The notion of a private circuit is introduced in [9] in the following sense. A circuit is given, and an adversary can access the information in certain wires of the circuit. There may be some restrictions on the choices of the compromised wires. The circuit is private (with respect to an adversary) if the adversary cannot get any information about the input and the output of the circuit. One of the results of [9] is about the case when the adversary chooses adaptively the compromised wires but there is a limit t on the total number of such wires. It is proved in [9], using a technique from the theory of multiparty computations, that for each circuit C of size n, there exists an efficiently constructed and functionally equivalent circuit C' so that C' is private against an adversary, who can choose adaptively t compromised wires, and the size of C' is at most $O(nt^2)$. Each input bit and output bit b of C is encoded by a $0,1$-sequence of length $t+1$ whose parity is b. Each gate of C is simulated in C' by a gadget of size $O(t^2)$. The privacy of the circuit is perfect in the sense that the distribution of the bits seen by the adversary does not depend on the input and output of the circuit. The result remains valid if we allow t compromised wires form each gadget.

In the present paper we prove a theorem which can be formulated in the framework of [9]. Assume that we have and adversary of the following type. The compromised wires are chosen independently, each with probability p, and the adversary knows the bits flowing through the compromised wires. Such an adversary will be called a p-random adversary. In this paper we show that if $p = \varepsilon$, where $\varepsilon > 0$ is a small constant, then for each boolean circuit C of size n, we can construct a functionally equivalent circuit C', so that $|C| = O((\log n)^4 n)$, each input/output bit of C is encoded by the parity of a $0,1$-sequence of length $m = O(\log n)$, and C is private against an ε-random adversary in a statistical sense, that is the probability that any information about the input/output of C will reach the adversary is negligible. In [9] the general case of randomly chosen compromised wires is reduced to the to the worst-case problem using the fact that with high probability the number of compromised wires can be only a constant times larger than its expected value. The results in [9] imply that a private circuit against a p-random adversary can be constructed if $p = O((\log n)^{-1})$. Our result improves the probability allowed for the choice of the compromised wires from $p = O((\log n)^{-1})$ to a small constant. The construction in [9] however is more efficient if we consider the size of the circuit C'. The paper [9] contains another construction as well for a private circuit against p-random adversaries.

For constructions of private circuits several reasonable additional assumptions can be made, for example Goldwasser and Rothblum in [8] consider circuits with simple leak-proof hardware component. Faust *et al.* in [5] consider adversaries who does not get individual bits flowing in the device, but some function of them. A general model for handling sidechannel attacks is given by Micali and Reyzin in [10]. They formulate the principle that "computation and only computation leaks information". That is, an inactive part of the memory is not in danger. In our results about computations on RAMs, we follow this principle; only the contents and addresses of those memory cells which are involved in an instruction executed at time t, may leak to the adversary at time t.

1.2 Informal description of the results

We prove two related theorems: a theorem about RAMs, resilient against certain type of attacks and another theorem about constructing private circuits. The circuit theorem is used as a step in the proof about RAMs.

Leak-proof simulation on RAMs. Assume that an adversary tries to get some information about the input/output of a program P running on a RAM with n memory cells each containing $O(\log n)$ bits. At each time, during the execution of P, the adversary is able to see the addresses of the memory cells involved in the instruction which is executed, and the name of this instruction as well. In addition to this, at certain times, the adversary can even see the contents of all of the memory cells involved in the instruction. (This includes the instruction pointer and all of the registers which are used by the machine to determine and store the result of the computation at that time.) We will call a time when this happens a compromised time. The adversary can choose the compromised times in an adaptive way, that is, immediately before the instruction at time t is executed, the adversary, using all of the information at his disposal, can decide whether time t will be compromised or not. The only restriction on his choice is, that among m consecutive instructions there can be at most εm whose time is compromised, where $\varepsilon > 0$ is a small constant. We will call such an adversary an (ε, m)-moderate adversary. We show that if $m = c\lfloor \log n \rfloor$, where $c > 0$ is a large constant, then for each program P, using n memory cells and time $T = O(\texttt{poly}(n))$, there exists an explicitly constructed and functionally equivalent program P', such that the probability that an (ε, m)-moderate adversary gets any nontrivial information about the input of P is negligible (polynomially small in n), and the time and space requirements of P' grows, compared to P, only by a factor of $\texttt{poly}(\log n)$. We assume that the program P' gets its input in an encoded form, namely each input bit b is encoded by a random $0,1$-sequence of length m whose parity is b. The output bits must be encoded by P' in a similar way. (Trivial information is the upper bound n on the memory requirement of the program P, the upper bound T on the time requirement of the program P and the number and timing of the input and output instructions executed by P. We do not try to hide these values from the adversary.)

Private circuits with leaking randomly chosen wires. As part of the proof of the result about RAMs described above, we also construct, for each positive integer m, and for each boolean circuit C of size n, a functionally equivalent circuit C' of size $O(n\texttt{poly}(m))$ with the following property. The circuit C' gets each input bit b of C encoded as a random $0,1$ sequence of length m whose sum modulo 2 is b, and C' provides its output with the same encoding. Moreover, an adversary who can observe each bit going through the wires of the circuit C' independently with a probability of $\varepsilon > 0$, where ε is a small constant, can get any information about the input/output of the circuit C only with a probability of at most ne^{-cm}, where $c > 0$ is a constant. That is, if ne^{-cm} is negligible, then C' is a private stateless circuit, with respect to the adversary described above, in the sense defined in [9],

2 The main results

2.1 Private circuits

We are considering finite probabilistic boolean circuits with fan-in at most two. They may be probabilistic in the sense that in addition to their usual input nodes, that we will call deterministic input nodes, they may have probabilistic input nodes as well which get their $0,1$ values at random independently and with uniform distribution. (Another essentially equivalent way to create random bits is to allow probabilistic gates, see [9].) We always assume that the set of all deterministic input nodes has a fixed ordering and make the same assumption about the output nodes as well. Assume that the probabilistic circuit C has k deterministic input nodes and l output nodes. Then C computes a random function, that is, for each $x \in \{0,1\}^k$, by evaluating C we determine the value of a random variable $\xi_{x,C}$ which takes its values in $\{0,1\}^l$. The randomness of $\xi_{x,C}$ comes from the probabilistic inputs of C.

Suppose that m, k, l are positive integers. We will say that the triplet $C = \langle C_0, \langle I_0, ..., I_{k-1}\rangle, \langle T_0, ..., T_{l-1}\rangle\rangle$ is a block circuit with k input blocks, l output blocks, and block-size m, if (a) C_0 is a probabilistic boolean circuit with mk deterministic input nodes and ml output nodes, (b) $I_0, ..., I_{k-1}$ is a partition of the set of all deterministic input nodes of C_0 into blocks of consecutive input nodes, each of size m, and (c) $T_0, ..., T_{l-1}$ is a partition of the set of all output nodes of C_0 into blocks of consecutive output nodes, each of size m. The size of C is defined as the size of C_0 and it will be denoted by $|C|$.

For each $a = \langle a_0, ..., a_{k-1}\rangle \in \{0,1\}^k$ we define a random variable $\eta_{a,C}$ in the following way. (The random variable η will have the same role for block circuits as the random variable ξ has for circuits. The difference between them is that preprocessing and postprocessing is incorporated only in η.) First we select km random $0,1$ bits $a_{i,j}$, $i=0,1,...,k-1$, $j=0,1,...,m-1$ independently, with uniform distribution on $\{0,1\}$ and with the condition that $\sum_{j=0}^{m-1} a_{i,j}=a_i$ (in F_2, the field with two elements) for $i=0,1,...,m-1$. We call this step the preprocessing.

Then we place the bits $a_{i,0},...,a_{i,m-1}$ at the input nodes in I_i in this order, for $i=0,1,...,k-1$. After that the probabilistic boolean circuit C_0 is evaluated, and we get ml output bits $b_{i,j}$, $i=0,1,...,m-1$, $j=0,1,...,l-1$, where the bits $b_{i,j}$, $j=0,1,...,m-1$ appear on the output nodes in T_i in this order, for $i=0,1,...,l-1$. The bits $b_0,...,b_{l-1}$ are defined by $b_i=\sum_{j=0}^{m-1} b_{i,j}$, (in F_2). The computation of these sums is called the postprocessing. The value of the random variable $\eta_{a,C}$ is the vector $b=\langle b_0,...,b_{l-1}\rangle$. $a=\langle a_0,...,a_{k-1}\rangle$ will be called the original input of C and $b=\langle b_0,...,b_{l-1}\rangle$ will be called the original output of C (referring to the motivation that the role of the block circuit C will be to simulate a boolean circuit with input a and output b.)

Assume that $C=\langle C_0,\langle I_0,...,I_{k-1}\rangle,\langle T_0,...,T_{l-1}\rangle\rangle$ is a block circuit with block size m and $\varepsilon>0$. We define an adversary who is observing the computation done by C, that is, the computation of a value of the random variable $\eta_{a,C}$ for some a, and tries to get some information about the pair $\langle a,b\rangle$. We will call the adversary an ε-random adversary if the information reaching him is determined in the following way.

Each wire of the circuit C_0 is declared compromised with a probability of ε so that all of these decisions are mutually independent and also independent from the probabilistic inputs of the circuit C_0 and the randomizations during the preprocessing. The adversary gets all of the bits that are going through compromised wires during the evaluation of C_0. More precisely the adversary gets the set W of all pairs $\langle w,b_w\rangle$ where w is the name of a compromised wire and b_w is the bit going through the wire w. (We assume that each wire has a unique name.) For each fixed $a\in\{0,1\}^k$, and $b\in\{0,1\}^l$, the conditional distribution of W determined by the computation of the value of the random variable $\eta_{a,C}$ and the selections of the compromised wires, with the condition that $\eta_{a,C}=b$, will be denoted by $\Phi_{a,b}^{(\varepsilon,C)}$. We will say that the block circuit C is (ε,p)-secure if, with a probability of at least $1-p$, the ε-random adversary does not gain any information about pair $\langle a,b\rangle$ where a,b are the original input and output. That is, C is (ε,p)-secure if for each $a\in\{0,1\}^k$, there exists an event A_a with respect to all of the randomizations, such that $\texttt{prob}(A_a)\geq 1-p$, and the conditional distribution of $\Phi_{a,b}^{(\varepsilon,C)}$ with the condition A_a, does not depend on the values of a and b. (We may think that $\neg A_a$ is a small probability event describing a situation when we do not state anything about the knowledge of the adversary.)

Assume now that $\mathcal{C}$ is a probabilistic boolean circuit with k deterministic inputs and l outputs, and C is a block circuit with k input blocks, l output blocks, and blocksize m. We will say that the circuit $\mathcal{C}$ and the block circuit C are functionally equivalent if for each $a\in\{0,1\}^k$ the random variables $\xi_{a,\mathcal{C}}$ and $\eta_{a,C}$ have identical distributions.

Theorem 1 *There exist $\varepsilon>0$, $c>0, c_1>0, c_2>0$, such that if m,k,l are positive integers, $m\geq 2$, and $\mathcal{C}$ is a probabilistic boolean circuit with k deterministic input nodes and l output nodes, then there exists a blockcircuit C with k input blocks, l output blocks and with block size m, such that, (i) C and $\mathcal{C}$ are functionally equivalent, (ii) $|C|\leq c_1m^4|\mathcal{C}|$, and (iii) C is (ε,p)-secure, where $p=e^{-cm}|\mathcal{C}|$.*

Moreover, C can be constructed from $\mathcal{C}$ in time $m^{c_2}|\mathcal{C}|$.

Remark. 1. It is possible to show, with minor modifications of the proof, that for each fixed $\varepsilon_1 > 0$, the upper bound $|C| \leq c_1 m^4 |\mathcal{C}|$ can be replaced by $|C| \leq c_1 m^{3+\varepsilon_1} |\mathcal{C}|$.

2. In the proof we assume that the gates of the circuits, $\mathcal{C}$ and C are performing the operations $x + y, x \times y$ and $x + 1$. We prove a generalization of the theorem for an arbitrary finite field F, where the gates are performing these operations in the field F and we also have a gate which performs the operation $-x$. In this case the inputs of the circuit $\mathcal{C}$ are not bits but elements of the field F. Each input elements a_i is encoded by a sequence $a_{i,0}$ from the elements of F which is chosen at random and with the condition $\sum_{j=0}^{i} a_{i,j} = a_j$. In a similar way, the output of C is an encoding of the output of $\mathcal{C}$.

2.2 Random access machines

We use a von Neumann type random access machine, where data and program are not distinguished. For the definition for such a machine see e.g., in [1] the random access stored program (RASP) machines, in the modified form where the contents of the memory cells are not arbitrary integers, but integers in the interval $[0, 2^q - 1]$, where q is the number of bits in a single memory cell.

In order to present the main result concisely, without too many definitions, we restrict our attention to one specific RAM described in [1], with the mentioned modifications. For each positive integer $q \geq 10$, $\mathcal{M}_q$ is a machine with 2^q memory cells each containing a sequence of $0, 1$ bits of length q. These cells will be called $\texttt{cell}(0), \texttt{cell}(1), ..., \texttt{cell}(2^q - 1)$. We will consider the sequences contained in the cells as the binary representations of natural numbers from the interval $[0, 2^q - 1]$, so if we say that a cell contains the natural number i, we mean it contains its binary representation. (We may restrict the number of memory cells if needed, by simply saying that a particular program is using only the first n cells for some $n < 2^q$. Therefore our requirement that the machine has $2^q - 1$ memory cells, is not a real restriction). The state of the machine at each time is a function which assigns to each memory cell its content.

$\mathcal{M}_q$ has γ_0 instructions, where γ_0 is a constant (does not depend on q). The names or encodings of these instructions are integers in $[0, 2^q - 1]$.

$\texttt{cell}(0)$ is called the accumulator of the machine and $\texttt{cell}(1)$ the instruction pointer. (The choice of these particular cells for the mentioned roles have no significance.)

The machine $\mathcal{M}_q$ has six types of instructions. (a) arithmetic operations, (b) an instruction to generate a random number, (c) instructions moving data between the memory cells, (d) control transfer instructions, which determine which instruction will be executed next, (e) input/output instructions, and (f) the halt instruction to terminate the execution of the program.

The machine $\mathcal{M}_q$ is working in cycles, each cycle counts as one time unit. In each cycle it does the following. It checks the content of the instruction pointer. Its content is interpreted as an address, of a memory cell, say, number i. Then the machine executes the instruction whose name is in cell i. An instruction may have parameters (for the sake of simplicity we assume that each instruction has at most one parameter). A parameter typically is the address of a memory cell. The content of cell $i + 1$ is considered as the parameter of the instruction in cell number i. The machine executes the instruction with the indicated parameter and then, if it is not a control transfer instruction, it increases the content of the instruction pointer by 2. If it is a control transfer instruction then the instruction defines the new content of the instruction pointer. We will say that *a memory cell or register is involved in an instruction* if its content is used by the machine to execute the instruction, or the result of the instruction is placed in it. We will use this concept by considering an adversary who wants to get some information about what the machine is doing, and at each instruction knows which memory cells/registers are involved in the instructions, and for some of the instructions he may even know their contents.

(a) The arithmetic instructions are $+, \times, -, \lfloor x/y \rfloor$, the constants $0, 1$, and $2^q - 1$. In case of the arithmetic operations of the form $f(x, y)$, at the time when the machine reads the instruction, which is in $\texttt{cell}(i)$, x must be in the accumulator, and y must be in the memory cell whose address is the parameter of the instruction, that is, y is in $\texttt{cell}(a)$ where a is the content of $\texttt{cell}(i+1)$. The result appears in the accumulator. In the case of the constants, the result of the instruction, that is, the constant, appears in the accumulator (and the value of the parameter is irrelevant).

(b) an instruction to generate a random number. A random integer from the interval $[0, 2^q - 1]$ appears in the accumulator, the value of the parameter is irrelevant.

(c) instructions moving data between the memory cells. Read instruction: if the value of the parameter is a, then the instruction puts the content of $\texttt{cell}(a)$ into the accumulator. Write instruction: if the value of the parameter is a, then the instruction puts the content of the accumulator into $\texttt{cell}(a)$.

(d) control transfer instructions, GOTO X instruction. If the value of the parameter is X then the content of the instruction pointer is changed into a. "IF $X = 0$ THEN GOTO Y" instruction. If the content of the accumulator is 0 then the content of the instruction pointer is changed into Y, where Y is the value of the parameter, otherwise the value of the instruction pointer is increased by 2. "IF $X > 0$ THEN GOTO Y" If the content of the accumulator is greater than 0, then the content of the instruction pointer is changed into Y, where Y is the value of the parameter, otherwise the value of the instruction pointer is increased by 2.

(e) input/output instructions. INPUT instruction. The input is written in the accumulator. OUTPUT instruction. The value of the accumulator is given as output. In both cases the value of the parameter is irrelevant.

(f) HALT instruction. Terminates the execution of the program.

The first few memory cells sometimes will be also called registers. (Intuitively this corresponds to the CPU of a computer.) A state of the machine $\mathcal{M}_q$, as we have indicated earlier, is a function which assigns to each of it cells a possible content.

A state of the machine $\mathcal{M}_q$, as we have indicated earlier, is a function which assigns to each of it cells a possible content. A history of the machine $\mathcal{M}_q$ is a function defined on an initial segment I of the natural numbers which assigns a state $S(t)$ to each $t \in I$ with the following property. Suppose that $t, t+1 \in I$ and in $S(t)$ the content of the instruction pointer is a. If a is the name of an instruction different of the input instruction and the random number generator instruction, then we get $S(t+1)$ from $S(t)$, by executing the instruction a with the values contained in the memory cells of $\mathcal{M}_q$ defined by $S(t)$. If a is the input or random number generator instruction then $S(t+1)$ must be a state that is obtained from $S(t)$ by executing instruction a with the values of the memory cells described in $S(t)$, and with a suitably chosen value of the input or the generated random number.

Before the machine starts to work, a program P_0 of constant length is placed in the memory. We will call this the starting program. We may think that this is a small program, whose role is to write in the memory a larger program P and data for P. When we will consider an adversary who wants to get some information about what the machine is doing, we will assume that P_0 is known to the adversary. Because of this, the exact way as P_0 gets into the machine is irrelevant. We usually will assume that the starting program knows what is the number of memory cells that the program can use and what is the total amount of time that can be used by the program. We assume that these two parameters are placed in the memory of the machine right after the starting program. Since we suppose that these parameters are known to the adversary, it is irrelevant what is the mechanism of placing these parameters into the machine.

We assume that a program P can run on the machine in the following way. Program P_0, the starting program, is already in the machine and asks for inputs. The first part of the input received by P_0 is the program P, (when we say program P, some data for the program may be included in it). P_0 writes the program P into the memory and then transfers the control to program P. While P is running it may ask for input and may also provide output. In this situation we will say that the program P_0 runs the program P.

We will call the machine $\mathcal{M}_q$ described above the basic RAM.

In the definition of adversaries we will also assume that there is a limit n on the number of memory cells used by the machine and a limit T on the time, so that the program must stop before time T. These will be always known to the adversary at the time when the machine starts working. n and T will be also known by the starting program of $\mathcal{M}_q$.

The benign adversary of $\mathcal{M}_q$. We define an adversary, called benign adversary, for $\mathcal{M}_q$ who gets only information that is revealed by the actions connecting $\mathcal{M}_q$ to its environment, that is, information that could not be hidden anyway efficiently. Other adversaries on various type of machines will be compared to this benign adversary. Our goal will be to transform the machine $\mathcal{M}_q$ and its starting program in a way that an (ε, m)-moderate adversary with the new machine/program does not get more information about the input than the benign adversary with the same input with the original machine/program.

Definition Assume that $\mathcal{M}_q$ is the basic RAM, an adversary $\mathcal{B}_q$ is observing it, and gets the following information. (i) the upper bound n on the number of memory cells used, and the upper bound T on the total time used by the machine, (ii) the time of all input instructions, (iii) the time of all output instructions, (iv) the time of the $HALT$ instruction. We also assume that $\mathcal{B}_q$ gets the values n, T before the machine starts to work and gets each other piece of information at the time when the corresponding instruction is executed. $\mathcal{B}_q$ will be called the benign adversary of $\mathcal{M}_q$.

RAM with parity encoded i/o. In the following we describe a computational model which is the extension of the basic RAM $\mathcal{M}_q$, with preprocessing of the input and postprocessing of the output. Every input bit and output bit b will be replaced by a random $0,1$-sequence whose sum is congruent to b modulo 2. (As we pointed out earlier without these steps an (ε, m)-moderate adversary could get, with a nonnegligible probability, some nontrivial information about the input.)

Definition. 1. Assume that $q \geq 10$ and m are positive integers. We define a machine $\mathcal{P}_{q,m}$ in the following way. The machine gets its input as a finite sequence of integers $a_0, a_1, ..., a_{l-1}$ form the interval $[0, 2^q - 1]$. It will be important later that the possible inputs of $\mathcal{M}_q$ and $\mathcal{P}_q$ are the same. First the input $a_0, ..., a_{l-1}$ is transformed into a $0,1$ sequence $b_0, b_1, ..., b_{ql-1}$ by replacing each integer with its binary representation containing exactly q bits. Then each bit b_i is replaced by a $0,1$-sequence $B_i = \langle b_{i,0}, ..., b_{i,m-1}\rangle$ of length m so that $\sum_{j=0}^{m-1} b_{i,j} \equiv b_i \pmod 2$ and the various sequences B_i, $i = 0, 1, ..., ql-1$ are chosen independently and with uniform distribution from the set of sequences satisfying the described condition.

Let $s = \langle s_0, s_1, ..., s_{mql-1}\rangle$ be the $0,1$-sequence which is the concatenation of the sequences $B_0, B_1, ...$ in this order. The sequence s is given to the machine $\mathcal{M}_q$ as an input sequence. Here we consider each s_i as a integer in $[0, 2^{q-1}]$, therefore each input instruction the machine $\mathcal{M}_q$ puts a single s_i, into $\texttt{cell}(0)$. Assume that the output of the machine $\mathcal{M}_q$ at the input s is the sequence $u_0, u_1, ..., u_{k-1}$, let $v_i = \sum_{j=im}^{(i+1)m-1} u_i$, for $i = 0, 1, ..., r-1$, where $r = \lfloor k/q \rfloor - 1$ and let w_i be the least nonnegative residue of v_i modulo 2. The output of the machine $\mathcal{P}_{q,m}$ is the sequence $w_0, ..., w_{r-1}$.

It is irrelevant how the machine $\mathcal{P}_{q,m}$ is executing the described preprocessing and postprocessing steps since we will consider adversaries who can get only information about the various states of the machine $\mathcal{M}_q$ used in $\mathcal{P}_{q,m}$ but not about the preprocessing and postprocessing steps.

2. If μ, ν are probability measures on the same σ-algebra $\mathcal{A}$, then the distance of μ and ν is $\sup\{|\mu(B) - \nu(B)| + |\mu(D) - \nu(D)|\}$ taken for all $B, D \in \mathcal{A}$ with $B \cap D = \emptyset$.

3. We will denote the machine $\mathcal{M}_q$ by $\mathcal{M}_q[P, n, T]$, if it is working with the starting program P, with only the first n memory cells, and with an upper bound T on the time used. In a similar way the machine $\mathcal{P}_{q,m}$ will be denoted by $\mathcal{P}_{q,m}[Q, n', T']$ if it is using the machine $\mathcal{M}_q$ with the starting program Q, with the first n' memory cells only and with an upper bound T' on the time used by $\mathcal{M}_q$. We will say that the machines $\mathcal{M}_q[P, n, T]$ and $\mathcal{P}_{q,m}[Q, n', T']$ are functionally equivalent if for the same input sequence the distribution of their output sequences are identical. The two machines are functionally equivalent with an error of δ if the distance of the two mentioned distributions is at most δ. (These distributions are determined by the random steps of $\mathcal{M}_q$ in both machines and by the random preprocessing of the input in $\mathcal{P}_{q,m}$.)

The (ε, m)-moderate adversary for $\mathcal{P}_{q,m}$. We define a set of adversaries for $\mathcal{P}_{q,m}$ that we will call the (ε, m)-moderate adversaries. Such an adversary will get some information about the working of the machine $\mathcal{P}_{q,m}$ at each time t, where we count the time t in the machine $\mathcal{M}_q$ which works in $\mathcal{P}_{q,m}$. The information received by the adversary depends on the state of the machine $\mathcal{M}_q$ at time t and a decision of the adversary. Namely the adversary $\mathbf{B} = \mathbf{B}_{q,m,\varepsilon}$ may declare time t as a compromised time or a noncompromised time. This is done by the adversary, right before the instructions at time t is executed, using all the knowledge available for him at that time. We describe below the information received by $\mathbf{B}$ at compromised or not compromised times. In the decision whether t will be a compromised time $\mathbf{B}$ can use probabilistic steps. There is no computational restriction on the decision process of $\mathbf{B}$. In other words $\mathbf{B}$ has a strategy of the following form. For each possible time t and for each finite $0,1$ sequence s (representing the information that $\mathbf{B}$ received before time t) a random variable $\xi_{t,s}$ with $0,1$ values is fixed. Assume that the information that $\mathbf{B}$ got before time t is represented by a $0,1$-sequence s_t. Then at time t adversary $\mathbf{B}$ takes a random value of the random variables ξ_{t,s_t} and declares the time t compromised if this value is 1. Otherwise t is declared a noncompromised time.

The strategy $\mathcal{S}$ described by the random variables $\xi_{t,s}$ will be called an (ε, m)-moderate strategy if each time interval I of length m contains, with probability 1, at most εm compromised times. An (ε, m)-moderate adversary is an adversary who is using an (ε, m)-moderate strategy for the selection of compromised times.

Assume first that t is a noncompromised time. Then $\mathbf{B}$ gets the following information.
(a) the information that the benign adversary would get till time t, and the values of m and ε.
(b) the addresses of all of the memory cells involved in the instruction executed at time t together with their roles in the instruction (but *not* their contents) and the name of the instruction.

Suppose now that t is a compromised time. Then $\mathbf{B}$ gets the following information.
(c) all of the information that $\mathbf{B}$ would get if t would be a noncompromised time
(d) the contents of all of the memory cells involved in the instruction executed at time t (with the indication which contents belongs to which memory cells).

Measuring the strength of an adversary. We will measure the strength of an adversary (for $\mathcal{M}_q$ or $\mathcal{P}_{q,m}$) by its ability to guess correctly what was the input of the machine, provided the input was selected at random and with uniform distribution from a set $\{a, b\}$ of two elements, where the set $\{a, b\}$ is known to the adversary.

Assume that the information received by an adversary $\mathcal{V}$ till the machine M has stopped is represented by a finite $0, 1$-sequence s. If a, b are possible inputs for the machine, then $\mathcal{V}$, using the information s, may try to guess whether a or b was the input. We will say that $\mathcal{V}$ can distinguish a form b with a bias greater than ι if there exists a function S (depending on a and b), so that if h is a random element of the set $\{a, b\}$ with uniform distribution, and at the input h the adversary $\mathcal{V}$ gets the information s while the machine is working, then $\texttt{prob}(S(s) = h) > \frac{1}{2} + \iota$, where the probability is taken for the randomization of h, the randomization of the machine, and the randomization of the strategy of the adversary together. $\texttt{bias}_{\mathcal{V}}(a, b)$ will denote the largest realnumber δ so that for all $\iota < \delta$ the adversary $\mathcal{V}$ can distinguish a from b with a bias greater than ι.

Definition. Assume that P, Q are starting programs for $\mathcal{M}_q$ and q, m, n, T, n', T' are positive integers, $\varepsilon > 0$, $\delta > 0$. We will say that $\mathcal{P}_{q,m}[Q, n', T']$ with an (ε, m)-moderate adversary δ-simulates $\mathcal{M}_q[P, n, T]$ with the benign adversary if the following conditions are satisfied:

(a) $\mathcal{M}_q[P, n', T']$ and $\mathcal{P}_{q,m}[Q, n, T]$ are functionally equivalent with an error of at most δ,

(b) for all possible inputs a, b we have $\texttt{bias}_{\mathbf{B}_{q,m,\varepsilon}}(a, b) \le \texttt{bias}_{\mathcal{B}_q}(a, b) + \delta$, where $\mathbf{B}_{q,m,\varepsilon}$ is the (ε, m)-moderate adversary defined for $\mathcal{P}_{q,m}[Q, n', T']$ and $\mathcal{B}_q$ is the benign adversary defined for $\mathcal{M}_q[P, n, T]$.

Theorem 2 *There exist $c_1, c_2, c_3, c_4 > 0$, so that for each starting program P, there exists a starting program Q, such that for all sufficiently small $\varepsilon > 0$, and for all positive integers m, q, n, T with $q \ge 10$, $\log n \le m \le n \le T \le 2^n$, we have that $\mathcal{P}_{q,m}[Q, nm^{c_1}, Tm^{c_2}]$, with an (ε, m)-moderate adversary, δ-simulates $\mathcal{M}_q[P, n, T]$, with the benign adversary, where $\delta = c_3 T \max(n^{-\log n}, e^{-c_4 m})$.*

Remark 1. Note that the starting programs P, Q do not depend on q. This is possible since we consider a program as a sequence of integers in the range $[0, 2^q - 1]$ and we suppose that the names of the instructions for $\mathcal{M}_q$ are fixed independently of q.

2. If $m = O(\log n)$ then the increase in the time requirement of $\mathcal{P}_{q,m}$ with respect to $\mathcal{M}_q$ is $O((\log n)^\gamma)$, where $\gamma \ge 1$ is a constant. $\gamma = \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4$ comes from four sources. γ_1 from the theorem about oblivious simulation, γ_2 from the circuit simulation of Theorem 1, γ_3 from the

simulation of the original q-bit arithmetic operations of $\mathcal{M}_q$ by bitwise boolean operations, and γ_4 from the remaining part of the proof. As the proofs of this paper are given, we have $\gamma_2 = 4$, but with minor modifications of the proof we can get $\gamma_2 = 3 + \varepsilon_1$ for each fixed constant $\varepsilon_1 > 0$. We get $\gamma_3 = 2$ if we perform the arithmetic operations in the obvious way. Finally $\gamma_4 = 1$, and it comes from the transformation of the program in a way that the (ε, m)-moderate adversary will have essentially the same knowledge as an ε-random adversary, as will be described later. (As a consequence if we are concerned about only ε-random adversaries then $\gamma_4 = 0$.) The same upper bound holds for the increase of the space requirement of $\mathcal{P}_{q,m}$ compared to $\mathcal{M}_q$. There is however some possibility for improvement here, since while simulating the circuit C of Theorem 1 with a RAM, we may repeatedly reuse the space which is needed for the evaluation of a subcircuit of C, which plays the role of a single gate of $\mathcal{C}$.

3 Sketch of the proofs.

The proof of Theorem 2 is using Theorem 1. Apart from the earlier results about oblivious simulation, the proof of Theorem 1 is the most important part of the proof of Theorem 2.

3.1 Sketch of the proof of Theorem 1

We may assume that the fan-in of each gate of the circuit $\mathcal{C}$ is at most two, since replacing an arbitrary circuit with a functionally equivalent circuit with fan-in two will increase the size of the circuit only by a constant factor. We may also assume that the adversary chooses compromised gates and not compromised wires. The information that the adversary gets will be the sequence of bits that arrives as input to the compromised gate. (That is, the adversary knows the bits and their sources.) Each gate is declared compromised with a probability of ε and the decisions made for the various gates are mutually independent. It is easy to see that for each ε-random adversary B_1 who gets information from the compromised wires there is a 2ε-random adversary B_2 who gets information from compromised gates as described above and B_2 always knows everything that is known by B_1. Therefore it is sufficient to prove the theorem for the case when the adversary selects compromised gates.

We will construct the block circuit C in the following way. We assume that in the circuit $\mathcal{C}$ there are only gates which correspond to the algebraic operations in the field F_2, that is, we have gates for addition multiplication and adding 1 to the input. For each of the gate types X, where X can be $+$, $\times$ or $+1$, we will construct a block circuit $C^{(X)}$ with block size m (and with probabilistic input nodes as well) which performs the algebraic operation associated with the gate type X on the values encoded by its input blocks and provides the result in encoded form on its only output block. For example if X is the product gate then $C^{(X)}$ must be a circuit with two input blocks I_0, I_1 and one output block T_0 each of them of size m, so that if $a_0, ..., a_{m-1}$ are the input values on block I_0, $b_0, ..., b_{m-1}$ are the input values on block I_1, and $v_0, ..., v_{m-1}$ are the output values on block T_0, then they satisfy the equation $\sum_{i=0}^{m-1} v_i = (\sum_{i=0}^{m-1} a_i)(\sum_{j=0}^{m-1} b_j)$. This must hold independently of the values of the probabilistic inputs. That is, the block circuit $C^{(X)}$ must be functionally equivalent to the gate X. This construction has the same framework as the corresponding circuit construction in [9], but the choices of the circuits $C^{(X)}$ will be different.

In addition to the functional equivalence with gate X we also want the circuit $C^{(X)}$ to be (ε, p')-secure with $p' = e^{-c_0 m}$, for some constant $c_0 > c_1$. We may hope that if we succeed in constructing such circuits $C^{(X)}$ for each of the three field operations $+$, $\times$ and $+1$ then we may construct the required (ε, p)-secure circuit C' in the following way. We replace each gate Y of type X in $\mathcal{C}$, by a copy Y' of the circuit $C^{(X)}$, so that if the outputs of the gates Y_0, Y_1 are the inputs of Y then the output blocks of the circuits Y_0', Y_1' are the input blocks of the circuit Y'. Unfortunately such a C is not necessarily (ε, p)-secure. E.g., if X is the $+1$ gate, then $C^{(+1)}$ can be the circuit which adds 1 to the first input bit and leaves all of other input bits unchanged. It is easy to see that $C^{(+1)}$ defined this way is (ε, p)-secure for, say, $p = e^{-\frac{1}{2}m}$, $\varepsilon = \frac{1}{4}$ and m is sufficiently large. Let $\mathcal{C}$ be a circuit which consists of a long sequence of $+1$ gates. Suppose that we construct a block circuit C by replacing each gate of $\mathcal{C}$ with the blockcircuit $C^{(+1)}$ defined above. Clearly C is not (ε, p)-secure, since if the sequence of gates in $\mathcal{C}$ is long enough, the adversary, with high probability, will know each input bit.

Therefore we will require a stronger property P from the circuits $C^{(X)}$, where X is a gate type. This stronger property will be inherited from the building blocks $C^{(X)}$ to the circuit C, constructed from them. To formulate this property first we define a probability distribution, that will depend on a real parameter $\theta \in (0, 1)$, and will be called θ-cylindrical distribution. It will be defined on the set of all deterministic inputs of the circuit, that is, if $I_0, ..., I_{k-1}$ are the input blocks the set of all F_2-valued functions defined on $\mathcal{I} = \bigcup_{i=0}^{k-1} I_i$. This set will be denoted by $\texttt{func}(\mathcal{I}, F_2)$. In the definition of a θ-cylindrical distribution we will use the following notation. If $a \in \texttt{func}(\mathcal{I}, F_2)$ then $\mathbf{h}(a, i) = \sum_{x \in I_i} a(x)$ for $i = 0, 1, ..., k-1$.

A probability distribution H will be called θ-cylindrical if there exists a function f defined on $\mathcal{I}$ and with values in F_2 such that

(i) H is concentrated on the set of extensions of the function f, that is, $H(a) > 0$ implies that a is an extension of f,

(ii) $|\texttt{domain}(f) \cap I_i| \leq \theta m = \theta |I_i|$ for $i = 0, 1, ..., k-1$,

(iii) For all $a, b \in \texttt{func}(\mathcal{I}, F_2)$, if $H(a) > 0$ and $H(b) > 0$, then for all $i = 0, 1, ..., k-1$, $\mathbf{h}(a, i) = \mathbf{h}(b, i)$

The function f will be called the handle of the distribution H and the set $\texttt{domain}(f)$ will be called the base set of the distribution H.

Property (iii) and (ii) together implies that for all extensions a of f onto $\mathcal{I}$ the function $\mathbf{h}(a, i)$, defined for $i = 0, 1, ..., k-1$, uniquely determines $H(a)$. (Recall that for a fixed input a the function $\mathbf{h}(a, i)$ is the original input.)

We define the θ-cylindricity of distribution on the set of all possible outputs in a similar way, by using the output block sequence $T_0, ..., T_{l-1}$ everywhere instead of the input block sequence $I_0, ..., I_{k-1}$. The function $\mathbf{h}$ is also defined from the output block sequence in this case.

Our goal is to construct the circuits $C^{(X)}$ in a way that if the input arrives with a θ-cylindrical distribution H, for a suitably chosen $\theta \in (0, 1)$ then the distribution of the information that reaches the ε-random adversary $\mathcal{Y}$ is uniquely determined by the handle f of the distribution H. This implies that $\mathcal{Y}$, even if he gets the handle f as extra information, will not learn anything about the probabilities $H(a)$ in addition to the fact that they form a θ-cylindrical distribution with handle f. We also want to construct the circuits $C^{(X)}$ in a way that their outputs, from the point of view of the adversary, have θ-cylindrical distributions. This will make possible an inductive argument when we prove that the circuit C has the same nice properties as the building blocks $C^{(X)}$.

The requirement about the θ-cylindrical distribution of the output cannot be satisfied for an ε-random adversary, because usually the information that such and adversary gains about the output cannot be described in such a nice concise way as a θ-cylindrical distribution. (For example, the situation where three bits $u, v, u+v$ are in the output but the adversary knows only $u+v$, may cause a problem.) Because of that, we will consider another "stronger" adversary, who knows more than the ε-random adversary, but the theorem still holds for him, and his knowledge about the output is θ-cylindrical for a suitably chosen $\theta \in (0,1)$. Such an adversary in this sketch will be called an ε-superior adversary. We assume that after the random set of gates A has been selected, whose elements are chosen independently and with probability ε, the ε-superior adversary may select another set of compromised gates $\mathcal{S}(A)$, where $\mathcal{S}$ is a fixed function which defines the ε-superior adversary. That is, for the ε-superior adversary, defined by the function $\mathcal{S}$, the set of compromised gates is $A \cup \mathcal{S}(A)$. We will consider ε-superior adversaries when the input arrives with a θ-cylinidrical distribution H. In this case we assume that the handle of the distribution H (and so it base set too) is known to the ε-superior adversary.

The property that we will require of the circuits $C^{(X)}$ will depend on three parameters, θ, ε, and p and will be called (θ, ε, p)-cylindricity. The meaning of θ, ε, p, roughly speaking, will be that if, the input arrives with a θ-cylindrical distribution then with a probability of at least $1-p$ the output also has θ-cylindrical distribution from the point of view of a suitably chosen ε-superior adversary. The actual requirement is stronger than that, we require that even if the complete deterministic input, all of the km bits, are known to this adversary $\mathcal{Y}$, then still the distribution of the output, form the point of view of $\mathcal{Y}$, is θ-cylindrical. This is possible since the circuit has probabilistic inputs as well which supply the randomness for the θ-cylindrical distribution. We will also require that with a probability of at least $1-p$ the same ε-superior adversary does not gain any information about the original input. (Recall that if $I_0, ..., I_{k-1}$ are the input blocks of a circuit C and $a_{i,0}, ..., a_{i,m-1}$ are the input bits on block I_i, then the sequence $\langle \sum_{j=0}^{m-1} a_{i,j} \mid i = 0, 1, ..., k-1 \rangle$ is the original input.)

In the definition below if $\mathcal{Y}$ denotes an adversary then $\mathcal{Y}^+$ will denote the adversary, who gets all of the information that is available for $\mathcal{Y}$ and in addition to that all of the km input bits of the circuit arriving at the input nodes in the blocks $I_0, ..., I_{k-1}$. First we define what is a (θ, ε, p)-cylindrical adversary, and then what is a (θ, ε, p)-cylindrical circuit.

Definition. (*This definition is given only for the "sketch of the proof". The final definition, provided during the proof, is slightly different from but equivalent to the present one.*) Assume that $\theta > 0, \varepsilon > 0, p \in [0,1]$ and $\mathcal{Y}$ is an ε-superior adversary defined by the function $\mathcal{S}$ for a probabilistic block circuit C with block size m and input block sequence $I = \langle I_0, ..., I_{k-1} \rangle$. The adversary $\mathcal{Y}$ is called a (θ, ε, p)-cylindrical adversary, if for all function $f \in \mathtt{func}_\theta(\bigcup_{j=0}^{k-1} I_j, F_2)$ we have that with a probability of at least $1-p$, with respect to the randomization of the ε-random set A used by $\mathcal{Y}$, the following two conditions are satisfied:

(1) *For each fixed deterministic input of the circuit C the corresponding output, from the point of view of adversary $\mathcal{Y}^+$, has a θ-cylindrical distribution, with base set*

$$(A \cup \mathcal{S}(A)) \cap \bigcup_{i=0}^{l-1} T_i$$

(2) *Assume that H_0 and H_1 are θ-cylindrical input distributions for the block circuit C so that their base sets and their handles are identical. Then the distributions of the knowledge of the ε-superior adversary $\mathcal{Y}$, is the same for H_0 and H_1. (Since the set A has been already fixed, the randomness in these distributions are provided by the randomizations in H_i and the random inputs of the circuit.)*

The block circuit is called (ε, θ, p)-cylindrical if there exists a function $\mathcal{S}$ so that the corresponding ε-superior adversary is (ε, θ, p)-cylindrical. □

Using Bayes' theorem we can show that condition (1) implies the following: if the input is arriving with a θ-cylindrical distribution H, then the distribution of the output from the point of view of $\mathcal{Y}$ is θ-cylindrical with base set $(A \cup \mathcal{S}(A)) \cap \bigcup_{i=0}^{l-1} T_i$

When we define the circuits $C^{(X)}$ so that they are (θ, ε, p)-cylindrical, the parameters θ, ε, and $p = e^{-cm}$ will be selected such that $0 < \varepsilon \ll c \ll \theta \ll 1$, where $a \ll b$ means that a is sufficiently small with respect to b. Assume now that a circuit $\mathcal{C}$ is given and we replace each gate X in it with a block circuit of type $C^{(X)}$. We will show that the resulting block circuit C is $(\theta, \varepsilon, p')$-cylindrical where $p' = |\mathcal{C}|p$ (see Lemma 3 and Lemma 4). The proof of this fact is an induction on the number of the gates. It will follow from condition (2) of the definition of a cylindricity that C will be (ε, p')-secure and this will complete the proof of Theorem 1. Therefore we reduced the problem of proving the theorem to the problem of constructing (θ, ε, p)-cylindrical block circuits $C^{(X)}$ for each of the gate type X.

Before we construct the block circuits $C^{(X)}$ for the gates $+, \times$ and $+1$, we define a new gate, the identity gate whose input is same as its output. The corresponding block circuit with block size m, will be denoted by $\mathbf{C}_m^{(=)}$. We want to construct $\mathbf{C}_m^{(=)}$ so that it has one input block, one output block, and if the deterministic input is $b_0, ..., b_{m-1}$, the output is $a_0, ..., a_{m-1}$, then $\sum_{i=0}^{m-1} b_i = \sum_0^{m-1} a_i$, and for all $0 < \varepsilon \ll c \ll \theta \ll 1$, and for all sufficiently large m the circuit $\mathbf{C}_m^{(=)}$ is $(\theta, \varepsilon, e^{-cm})$-cylindrical. We will also call the circuit $\mathbf{C}_m^{(=)}$ the copying circuit. Its construction is based on a constant degree expander.

3.1.1 The copying circuit $\mathbf{C}_m^{(=)}$

Recall that the task of the copying circuit $\mathbf{C}_m^{(=)}$ is to produce from an encoded input sequence of length m an encoded output sequence of also length m so that the sum of the encoded input sequence, that is, the original input, is the same as the sum of the encoded output sequence. The copying circuit only copies the (original, one bit) input. Apart from that, the copying circuit also refreshes the randomness of the encoded input. That is, if the ε-random adversary knows a little bit more about the input of the copying circuit than what is our goal, then the design of the copying circuit will make sure that his knowledge about the output of the circuit falls back to an acceptable level.

The deterministic input nodes of the copying circuit C will be denoted by $\wp_0, ..., \wp_{s-1}$ the output nodes will be denoted by $\mathcal{T}_0, ..., \mathcal{T}_{s-1}$. Assume further that G is a d-regular expander on the set of vertices $V = \{0, 1, ..., m-1\}$, where d is a constant and m is sufficiently large. We also assume that the expansion factor of the graph G is $1 + \alpha$, where $\alpha > 0$ is a constant, that is, for each $X \subseteq V$ with $|X| \le \frac{1}{2}m$ we have that the set of all points x such that either $x \in X$ or x has a neighbor in X, has at least $(1 + \alpha)|X|$ elements. The set of edges in G will be denoted by E. Given the graph G with these properties we define a block circuit $C = \mathbf{C}^{(=)}_{m,G}$ with one input block and one output block, that we will call the copying circuit associated with the graph G. The copying circuit C has m deterministic input nodes, m output nodes, and for each edge $e \in E$, it has a probabilistic input node $\mathcal{G}_e$.

Assume that δ_r is the deterministic input on $\wp_r$ for $r = 0, 1, ..., m-1$ and $\gamma(e))$, $e \in E$ is the probabilistic input on $\mathcal{G}_e$. Using the operations $+$ in F_2 we define the output ν_i, $i = 0, 1, ..., m-1$ on the output node $\mathcal{T}_i$ by

$$\nu_i = \delta_i + \sum\{\gamma((i,j)) \mid (i,j) \in E\}$$

that is, the ith output bit is the sum of the ith input bit and the sum of the random values $\gamma((i,j))$ attached to the edges of G incident to the vertex i. We construct the circuit $\mathbf{C}_m = \mathbf{C}^{(=)}_{m,G}$, using only $+$ gates, which compute the described value of ν, given δ and γ as deterministic and probabilistic inputs. The sum of the deterministic inputs will be the sum of the outputs. This is a consequence of the fact that if we add the outputs then each term $\gamma(e)$ will occur exactly twice and so in F_2 they cancel each other. (Later when we consider the problem over an arbitrary finite field F, then in the definition of the output the term $\gamma(i,j)$ will have a sign depending on whether $i < j$ or $i > j$, so the cancellation will be guaranteed even if $F \neq F_2$.)

For the proof of the (θ, ε, p)-cilindricity of the circuit C we will need the fact that if in the expander graph G we have a set $A \subseteq V$ with less than $\varepsilon' m$ elements, where $\varepsilon' > 0$ is a small constant then the complement of A contains a set $D \subseteq V$ which is connected, and it has at least $m - d(1 + \frac{d}{\alpha})|A| \geq \frac{1}{2}m$ elements (see Lemma 10). We will use this statement for the case when A contains all of the points $j \in V$ so that the ε-random adversary has seen a partial result in the computation of the output which depends on either δ_i or ν_i or $\gamma((i,j))$ for some $j \in V$. This definition of A makes sure that the probabilistic inputs $\gamma((i,j))$, $i, j \in D$ can be changed arbitrarily, without changing the knowledge of the adversary. Since D is connected, using such changes in the probabilistic input we can change the output on the set $H = \{\mathcal{T}_i \mid i \in D\}$ in an arbitrary way, with the only constraint that the sum of the output values of H remains the same. This observation will be used to show, that with high probability, C has θ-cylindrical distribution from the point of view of a suitably chosen ε-superior adversary, if the set of inputs is fixed. The fact that this adversary does not have any information about the original input, provided that the input arrives with θ-cyclical distribution, will simply follow the from the fact that he has no information at all about the input values in D, and $|D| \geq (1-\theta)m$.

3.2 Block circuits for addition and multiplication

We define the block circuit $C^{(+)}$ which corresponds to the gate $+$ as follows. It has two input blocks and one output blocks. First it adds the input vectors component by component and then applies a copying circuit $\mathbf{C}^{(=)}_{m,G}$ to the result. We define the block circuit $C^{(+1)}$ in a similar way, it adds one to the first input bits and leaves the other $m-1$ input bits unchanged. After that it applies a copying circuit to the result. In both cases the proof of (θ, ε, p)-cylindricity is essentially the same as for the copying circuit.

The most problematic part of the proof is the construction of the block circuit $C^{(\times)}$ that we will call the multiplication circuit. In this case direct computation would reveal the input for an ε-random adversary. The construction of $C^{(\times)}$ is based on the following idea. Suppose we want to compute the product ab, $a, b \in F_2$. We may split a at random into a sum $a = a_0 + a_1$. (In the block circuit $C^{(\times)}$ this calculation and randomization will be done with sequences of length m encoding a, a_0, a_1.) We have $ab = a_0b + a_1b$. Now we split b at random into two parts but do this independently at the two occurrences of b. We get $ab = a_0b_0 + a_1b_1 + a_1b_2 + a_1b_3$ where $b = b_0 + b_1 = b_2 + b_3$. At the next step we split a_0 and a_1 into two parts independently at their various occurrences, and get for ab a sum with eight terms, each is a product, where the first factor was derived from a the second from b. We continue this, alternately splitting the first or second factors from each term at random independently and with uniform distribution. Assuming that $m = 2^d$, after $3d$ steps we will have a sum with m^3 terms, each one is a product. As we told earlier the factors of these products are encoded in $C^{(\times)}$ as the sum of sequences of length m. Assume that uv is such a product where u is encoded by the sequence $\bar{u}$ and v is encoded by the sequence $\bar{v}$. $C^{(\times)}$ computes u from $\bar{u}$ by adding the bits of $\bar{u}$. It is important that this computation, namely the $m-1$ additions, is arranged in the natural way on the nodes of a binary tree of depth d. Then $C^{(\times)}$ computes v from $\bar{v}$ in a similar way. In the possession of u and v, $C^{(\times)}$ computes the product $w = uv$ and then encodes the result w by a random $0,1$-sequence $\bar{w}$ of length m. The encoding of w by $\bar{w}$ is calculated on the nodes of a binary tree with m leaves, starting at the root, getting the encoding on the leaves, and randomly splitting each bit on a node into a random sum given on its two children. Finally after this has been done with all of the m^3 products of type uv, the block circuit $C^{(\times)}$ adds the m^3 vectors $\bar{w}$ by repeatedly using the circuit $C^{(+)}$. (The order of these additions is irrelevant, we will assume that they are performed in a sequential way). The sum of the m^3 vectors $\bar{w}$ is the output of the block-circuit $C^{(\times)}$.

In the described circuit $C^{(\times)}$ we reduced the computation of a single product to additions and the computation of m^3 different products. At first it may seem, that computing m^3 product, without leaking information to the adversary, is more difficult than the original task with a single product. However, this is not the case, since we do not mind if information leaks to the adversary about $o(\varepsilon m^3)$ random products out of the total m^3. We will show that such a leak, with high probability, does not give any information about the original product ab to the adversary. Therefore by parallelizing the computation of the product we made a certain amount of leaks acceptable.

We want to prove that the block circuit $C^{(\times)}$ is (θ, ε, p)-cylindrical, where $0 < \varepsilon \ll c \ll \theta \ll 1$ and $p = e^{-cm}$. We have to show that there exists an ε-superior adversary $\mathcal{Y}$ for the circuit $C^{(X)}$ which satisfies conditions (1) and (2) from the definition of cylindricity. For the proof of condition (1) it will be sufficient to know that the last addition circuit $C^{(+)}$, which gives the output of $C^{(\times)}$, ends with a copying circuit $C_{\texttt{end}}$. Since we know already that the copying circuit is (θ, ε, p)-cylindrical there exists an ε-superior adversary $\mathcal{Y}_c$ for this copying circuit. If we choose an ε-superior adversary for $C^{(\times)}$ such that it knows the same about the circuit $C_{\texttt{end}}$ as $\mathcal{Y}_c$ then it is easy to see that it will satisfy condition (b). (For the proof of this fact we may use Bayes' theorem with respect to all the possible evaluations of the input nodes of $C_{\texttt{end}}$.)

For the proof of condition (2) we consider an ε-superior adversary $\mathcal{Y}$, who gets the following information:

(i) Assume that A is the random subset of the set of nodes of $C^{(\times)}$ whose elements are selected independently and with a probability of ε, and $\texttt{full}(A)$ is the set of all nodes of $C^{(\times)}$ which contains a gate which sends an input to a gate in A. $\mathcal{Y}$ gets the output $g(a)$ of each gate

$a \in \texttt{full}(A)$ which is computed during the evaluation of the circuit $C^{(\times)}$. (Here we used the output of the gates instead of their inputs. This is the reason that instead of A we had to use the set $\texttt{full}(A)$.)

(ii) we assume that the input of $C^{(\times)}$ arrives with a θ-cylindrical distribution H which has base set Y and handle f. $\mathcal{Y}$ gets the function f, (that is, $\texttt{domain}(f)$ and for each $x \in \texttt{domain}(x)$ the value $f(x)$).

(iii) $\mathcal{Y}$ gets all of the information that was received by $\mathcal{Y}_c$ about $C_{\texttt{end}}$ (for the sake of simplicity we will ignore this information in this sketch, since it does not have any essential role in the remaining part of the proof).

We will show that the distribution of the knowledge of $\mathcal{Y}$ does not depend on the original input. Assume that $(\delta_0, \delta_1) \in F_2 \times F_2$, $(\delta_0', \delta_1') \in F_2 \times F_2$ are possible choices for the original input, that is, the circuit $C^{(\times)}$ computes either the product $\delta_0\delta_1$ or the product $\delta_0'\delta_1'$. We show that the distribution of the knowledge of $\mathcal{Y}$ is the same in both cases.

Suppose that δ_0, δ_1, f, g are given. We are interested in the conditional probability of the event that the output of each gate $a \in \texttt{full}(A)$ is $g(a)$, with the condition that the input arrives with a θ-cylindric distribution H with handle f and the original input is (δ, δ_0). (As we said we ignore now the knowledge of $\mathcal{Y}$ given in (iii), but in the final proof we will extend the set A to cover some part of $C_{\texttt{end}}$.) The conditional distribution corresponding to this conditional probability can be expressed by generating all of the input bits, both probabilistic and deterministic of $C^{(\times)}$ with uniform distribution, and with the condition that they are compatible with f, with the original input (δ_0, δ_1), and with the outputs $g(a)$ for all $a \in \texttt{full}(A)$.

Since we considered here uniform distribution for the input bits, the probability of the described conditional distribution is proportional of the number of inputs satisfying the conditions. For given g, f, δ_0, δ_1, we will denote this number by $N(g, f, \delta_0, \delta_1)$. We have to show that $N(g, f, \delta_0, \delta_1) = N(g, f, \delta_0', \delta_1')$. We want to show that $N(g, f, \delta_0, \delta_1)$ is the number of solutions of a system of polynomial equations over F_2.

With each input node (deterministic or probabilistic) of $C^{(\times)}$ we associate an unknown z_h. Then for each node a of $C^{(\times)}$ we define a polynomial p_a in the indeterminates z_h by recursion on a. If a is an input node then $p_a = z_a$. If a is a $+$ gate which gets its input from the gates u, v then $p_a = p_u + p_v$. In a similar way if u is a $\times$ gate then $p_a = p_u p_v$ and if a is a $+1$ gate getting its input from u then $p_a = p_u + 1$.

We consider now the system consisting of the following equations over F_2, where I_0, I_1 denote the input blocks of the circuit $C^{(\times)}$.

$$\begin{array}{ll} \sum_{x \in I_j} p_x = \delta_j & \text{for } j = 0, 1, \\ p_u = f(u) & \text{for } u \in \texttt{domain}(f), \\ p_b = g(b) & \text{for } b \in \texttt{full}(A). \end{array}$$

This system will be denoted by $\Gamma(g, f, \delta_0, \delta_1)$. Clearly the number of solutions of $\Gamma(g, f, \delta_0, \delta_1)$ is $N(g, f, \delta_0, \delta_1)$. Therefore we have to show that the number of solutions of $\Gamma = \Gamma(g, f, \delta_0, \delta_1)$, $\Gamma' = \Gamma(g, f, \delta_0', \delta_1)'$ are the same.

As a first step we will show that one of Γ and Γ' has a solution then the other one also must have at least one solution. If Γ and Γ' would be linear systems this would be sufficient to show that their number of solutions are equal, since Γ and Γ' differ only in their inhomogeneous parts. (In a linear system, if the number of solutions is not zero then it is equal to the number of solutions of the corresponding homogeneous system.) Unfortunately Γ and Γ' are not linear, since if b is a product gate then the polynomial p_b is nonlinear. However, the mentioned result above linear systems, can be extended to certain nonlinear systems using a diagonalization argument, as described in Lemma 7. We will show that Γ, Γ' are systems of this type so Lemma 7 implies that if both have at least one solutions then the numbers of their solutions are equal. (We do not sketch here this part of the proof.) Therefore our task is to show that if Γ has at least one solution then Γ' also have at least one solution.

For the proof of this fact we have to return to the definition of $C^{(\times)}$. First we give a more formal description of the process used in the multiplication circuit.

3.3 The tree structure of the product circuit

We describe here, as part of the "sketch of the proof" some elementary definitions abut trees and two lemmas in their final forms since they are simple and at the same time very important for our proofs about the multiplication circuit.

Definition. 1. A binary tree T of depth d will mean the following structure. A graph on the set T with $2^{d+1}-1$ elements, that are arranged in d levels, $L_0, ..., L_d$, with $|L_i| = 2^i$. Each point x of level i, that is L_i, is connected by an edge to exactly two points of L_{i+1} which are called the successors of x for $i = 0, 1, ..., d-1$. If $x, y \in L_i$, $x \neq y$ then they have no common successors. The graph has no other edges than the ones described above. The only element of L_0 is the root of the tree, it will be denoted by t_0. For each $x \in L_0 \cup ... \cup L_{d-1}$ one of the two successors of x is designated as the left successor and denoted by $\mathbf{l}x$, and the other one is the right successor denoted by $\mathbf{r}x$. For all $t \in L_0 \cup ... \cup L_{d-2}$, the four successors of the successors of t will be denoted by $r_0(t) = \mathbf{ll}t$, $r_1(t) = \mathbf{rl}t$, $r_2(t) = \mathbf{lr}t$, $r_3(t) = \mathbf{rr}t$. T_0 denotes the union of all of the even levels $L_0, L_2, ...$ and T_1 denotes the union of all of the odd levels. Since we will use only binary trees in this paper, a tree of depth d will always mean a binary tree of depth d.

2. Assume that T is a binary tree of depth d with levels $L_0, ..., L_d$. Each element of L_d is called a leaf of the tree. If a is a leaf of the tree then the set of vertices in the shortest path leading from a to the root t_0 is called a branch and will be denoted by $\texttt{branch}(a)$. Clearly $|\texttt{branch}(a)| = d+1$.

We define a partial ordering $\leq_T$ on the set of vertices of T. It is defined by: $a <_T b$ iff the tree has a branch containing both a and b, and b is closer to the root. Obviously t_0 is the largest element of T with respect to $\leq_T$. $\square$

The next lemma describes part of the computation performed by the multiplication circuit.

Lemma 1 *Assume that λ is a function defined on the tree T with depth d so that the values of λ are in a field F, and*

(3) *for each $i = 0, ..., d-2$ and for each $t \in L_i$ we have that $\lambda(t) = \lambda(r_0(t)) + \lambda(r_1(t)) = \lambda(r_2(t)) + \lambda(r_3(t))$*

Then for all $j = 0, 1, ..., d-1$ we have $\lambda(t_0)(\lambda(\mathbf{l}t_0) + \lambda(\mathbf{r}t_0)) = \sum_{t \in L_j} \lambda(t)(\lambda(\mathbf{l}t) + \lambda(\mathbf{r}t))$

The proof of the lemma is an induction on j. The multiplication circuit recursively defines a function λ which is defined on the nodes of a tree of depth $3d$. If we want to compute the product ab then $\lambda(t_0) = a$, $\lambda(\mathbf{l}t_0) = b_0$, $\lambda(\mathbf{r}t_1) = b_1$, where b was split at random into the sum $b = b_0 + b_1$ with uniform distribution. Recursively we define the elements $\lambda(r_i(t))$ for $i = 0, 1, 2, 3$ in the following way. We split the element $\lambda(t)$ into two random sums $\lambda(t) = u_0 + u_1 = u_2 + u_3$ independently and with uniform distribution. Then we define $\lambda(r_{i(t)})$ by $\lambda(r_{i(t)}) = u_i$ for $i = 0, 1, 2, 3$. Lemma 1 implies that $ab = \sum_{t \in L_{3d-1}} \lambda(t)(\lambda(\mathbf{l}t) + \lambda(\mathbf{r}t))$.

The multiplication circuit of course does not work with the values $\lambda(t)$ directly but each $\lambda(t)$ is encoded as the sum of the elements of a sequence $\Lambda(t)$ of length m. The actual values of $\lambda(t)$ are computed by the circuit for only $t \in L_{3d} \cup L_{3d-1}$.

The recursive definition of $\lambda(t)$ is translated into a recursive definition of the function $\Lambda(t)$ namely to get $\Lambda(r_i(t))$ the circuit has to cut the vector $\Lambda(t)$ independently and with uniform distribution into two random sums $\Lambda(t) = U_0 + U_1 = U_2 + U_3$. Then we define $\Lambda(r_{i(t)})$ by $\Lambda(r_{i(t)}) = U_i$ for $i = 0, 1, 2, 3$. This random cutting can be easily done by defining U_0, U_2 as independent random vectors given by the probabilistic inputs of $C^{(\times)}$ and then $U_1 = \Lambda(t) - U_0$, $U_3 = \Lambda(t) - U_2$. After the random splitting is done a copying circuit is applied to all four terms to refresh their randomness.

With these process we can guarantee that the adversary will not get any nontrivial information until levels L_{3d-1} and L_{3d} are reached. Here however it may happen that for each fixed $t \in L_{3d} \cup L_{3d-1}$ with a probability of ε', where $0 < \varepsilon \ll \varepsilon' \ll 1$ the adversary learns what is $\lambda(t)$, where $0 < \varepsilon \ll \varepsilon' \ll 1$. The following lemma guarantees that even if the adversary knows $\lambda(t)$ on such a random set, this will not able him to rule out certain values for a and b, that is, for $\lambda(t_0), \lambda(\mathbf{l}t_0)$ and $\lambda(\mathbf{r}t_0)$. We need a few definitions for the statement of the lemma.

Definition. Assume that T is a tree of depth d. We will use the following notation: $\mathcal{L}_i = \bigcup_{s=0}^{i} L_i$, $\mathcal{L}_{2j}^{(0)} = \bigcup_{s=0}^{j} L_{2s}$, and $\mathcal{L}_{2k+1}^{(1)} = \bigcup_{s=0}^{k} L_{2s+1}$, provided that $i, 2j, 2k+1 \le d$. $\square$

Definition. Assume that λ is a function so that $\texttt{domain}(\lambda) \supseteq \mathcal{L}_{2d}^{(0)} = L_0 \cup L_2 \cup ... L_{2d}$, where T is a tree of depth at least $2d$, and the values of λ are in a field F, and

(4) *for each* $i = 0, ..., d-1$, *and for each* $t \in L_{2i}$ *we have* $\lambda(t) = \lambda(r_0(t)) + \lambda(r_1(t)) = \lambda(r_2(t)) + \lambda(r_3(t))$

Then, we will say that λ is a well-balanced function on $\mathcal{L}_{2d}^{(0)}$. Assume now that $\texttt{range}(\lambda) \subseteq F$, $\texttt{domain}(\lambda) \supseteq \mathcal{L}_{2d+1}^{(1)} = L_1 \cup L_3 \cup ... \cup L_{2d+1}$, the depth of T is at least $2d+1$, and

(5) *for each* $i = 0, ..., d-1$, *and for each* $t \in L_{2i+1}$, *we have* $\lambda(t) = \lambda(r_0(t)) + \lambda(r_1(t)) = \lambda(r_2(t)) + \lambda(r_3(t))$

Then we will say that λ is a well-balanced function on $\mathcal{L}_{2d+1}^{(1)}$. If λ is defined on $\mathcal{L}_d$ then we will say that it is well-balanced on $\mathcal{L}_d$, if it is well-balanced on both $\mathcal{L}_{2d'}^{(0)}$ and $\mathcal{L}_{2d''+1}^{(1)}$, where $d' = \lfloor \frac{d}{2} \rfloor$ and $d'' = \lfloor \frac{d-1}{2} \rfloor$. $\square$

We will use the next lemma for our proof about the multiplication circuit with d:=$3d$. The lemma essentially says that if a well-balanced function λ is known at a small random subset of leaves of the tree T then the values $\lambda(t_0), \lambda(\mathbf{l}t_0)$, $\lambda(\mathbf{r}t_0)$ still can be arbitrary. To make the lemma more applicable for the proof about the multiplication circuit, instead of using the leaves we select the random set at the previous level and then extend it to the leaves.

Lemma 2 *Assume that,*

(6) *$\varepsilon > 0$, d is a positive integer, and T is a tree of depth d,*

(7) *A' is a random subset of L_{d-1} so that all of the events $x \in A'$, $x \in L_{d-1}$, are mutually independent, and $\mathtt{prob}(x \in A') \le \varepsilon$ for all $x \in L_{d-1}$, and*

(8) *$A = A' \cup \{x \in L_d \mid \exists y \in A',\ x \le_T y\}$.*

Then the probability of the following event is at least $1 - 3(4\varepsilon)^{2^{\bar{d}-1}}$, where $\bar{d} = \lfloor \frac{d}{2} \rfloor$:
For all functions λ' defined on A with values in the field F, if λ' has a well-balanced extension to $\mathcal{L}_d = T$, then for all $\delta_0, \delta_1, \delta_2 \in F$, λ' has a well-balanced extension λ to T, so that $\lambda(t_0) = \delta_0$, $\lambda(\mathbf{l}t_0) = \delta_1$, $\lambda(\mathbf{r}t_0) = \delta_2$ where t_0 is the root of the tree T.

Using this lemma we will start constructing a solution of Γ' from a solution of Γ in the following way. From the solution of Γ we get a well-balanced function λ so that $\lambda(t_0) = \delta_0$ and $\lambda(\mathbf{l}t_0) + \lambda(\mathbf{r}t_0) = \delta_1$. We now want to construct another well-balanced function λ' so that $\lambda'(t_0) = \delta'_0$, $\lambda'(\mathbf{l}t_0) + \lambda'(\mathbf{r}t_0) = \delta'_1$ and it is compatible to the knowledge of the adversary $\mathcal{Y}$, with the assumption that the function λ' was computed by the multiplication circuit. As we have said the adversary $\mathcal{Y}$ knows very little about the value of $\lambda(t)$ for $t \in L_i$ with $i < 3d - 1$. $\mathcal{Y}$ may know the values of λ on a random subset of the last two levels but on this random subset we define λ' by $\lambda' = \lambda$. According to Lemma 2 such a λ' with high probability can be extended into a well-balanced function λ' with the required property $\lambda'(t_0) = \delta'_0$, $\lambda'(\mathbf{l}t_0) + \lambda'(\mathbf{r}t_0) = \delta'_1$.

We will construct a solution of Γ' using the function λ'. First we have to construct a function Λ' defined on T, whose values are m dimensional vectors over F_2, with the property that the sum of the components of $\Lambda'(t)$ is $\lambda'(t)$ for all $t \in T$. Moreover we have to do this in a way that the defined vectors $\Lambda'(t)$ does not contradict to the knowledge of the adversary.

We will define $\Lambda'(t)$ for $t \in L_i$ by recursion on i. ($\Lambda'(t_0)$ and $\Lambda(\mathbf{l}t_0)$ is the input of the circuit in the two input blocks $\Lambda'(\mathbf{r}t_0) = 0$.) In this recursive definition the only problematic part is the bottom level of the tree. Before that, when we split the vectors $\Lambda(t)$ into random sums in two different ways, the basic properties of the copying circuit will ensure that the recursive step can be accomplished and at the same time the unknowns z_h, for the probabilistic inputs h of C occurring in this part of the circuit, can be evaluated in a way which is compatible to the definition of $\Lambda(t)$ and the equations in Γ'

At the two bottom levels the definition of $\Lambda(t)$ is more problematic. This is the part where the multiplication circuit computes $\lambda(t)$ from $\Lambda(t)$. This computation is performed on a binary tree $\mathbf{T}_t$ of depth d. The problem can be that even if the node where $\lambda(t)$ is computed is not compromised, there can be so many compromised nodes in the tree $\mathbf{T}_t$ that the adversary will know what is $\lambda(t)$. To avoid this difficulty we include all such nodes t in the set A of Lemma 2. Because of this inclusion we have to show that the probability, that a node t in the bottom two levels will get into the set A, is small. The following lemma will be used for the proof of this fact.

Lemma 17 *For all $\varepsilon' > 0$ and for all sufficiently small $\varepsilon > 0$ the following holds. Assume that T is a binary tree of depth d and H is a subset of the leaves of the tree with at least 2^{d-1}*

elements. Suppose further that A is a random subset of T so that each element of T will belong to A with a probability of at most ε, and the events $t \in A$ are independent for all $t \in T$. Then the probability that there exists a branch of T disjoint from A and containing an element of H is at least $1 - \varepsilon'$.

Actually we will use a slightly stronger but more complicated version of the lemma that we will formulate as a Corollary. We use this lemma/corollary for both for the tree $\mathbf{T}_t$ and for the tree that we use when we replace the product $uv \in F_2$ by a sequence of length m. The remaining part of the multiplication circuit when we add the m^3 products is easier, here as in the upper part of the tree T the basic properties of the addition circuits are sufficient for the recursive definition of the solution of Γ'.

3.4 Sketch of the proof of Theorem 2.

For the sake of simplicity we assume that $m = O(\log n)$. As a first step in the proof of Theorem 2, we show that it is enough to prove it in the special case, when we replace the (ε, m)-moderate adversary with another ε-random adversary. We will call an adversary for the machine $\mathcal{P}_{q,m}$ an ε-random adversary, if the compromised times are selected by the adversary at random in a way that for each fixed t the probability that t is compromised is at most ε and the events "t is compromised" for various values of t are mutually independent. An ε-random adversary gets the same information as the (ε, m)-moderate adversary would get with the same choices of compromised times.

Assume that the theorem holds for $c\varepsilon$-random adversaries, for some constant $c > 1$, that we will fix later. Let Q' be the starting program of $\mathcal{P}_{m,q}$ such that $\mathcal{P}_{m,q}[Q']$ is secure against $c\varepsilon$-random adversaries. We want to define the starting program Q so that $\mathcal{M}_{m,q}[Q]$ is secure against (ε, m)-moderate adversaries. We define Q in a way that $\mathcal{M}_q[Q]$ simulates the machine $\mathcal{M}_q[Q']$, and so $\mathcal{P}_{q,m}[Q]$ simulates the machine $\mathcal{P}_{q,m}[Q']$. The simulation will simply delay each step of $\mathcal{M}_q[Q']$ by a random amount of time. More precisely before Q executes instruction X in this simulation, Q generates a random integer $g \in [0, m-1]$ with uniform distribution, and if $g = 0$, then Q executes X in the simulation of $\mathcal{M}_q[Q']$ by using no more than c time units for some constant $c > 0$. Otherwise, instead of executing X, it generates another random integer g etc., till it gets a $g = 0$ and instruction X is executed. (The constant c may be greater than 1 since the simulation of a single instruction in a controlled way requires several instructions.) In an interval of length m, starting at t_0, the (ε, m)-moderate adversary can have at most εm compromised times. If these compromised times are fixed in advance then clearly the described method as Q' chooses the time interval J of length c when instruction X is executed, guarantees that the probability that there is a compromised time in J is at most $c\varepsilon$. In general the compromised times are not fixed in advance. Our conclusion however remains true since the adversary does not get any new information till time t_1, when instruction X is executed. Therefore the compromised times chosen by him in the interval $[t_0, t_1]$ can be considered as fixed in advance.

For the proof of Theorem 2 with an ε-random adversary we use the results of [3] and [2] about oblivious simulation. It is proved there that every RAM M_1 can be obliviously simulated by another RAM M_2 of the following structure. The memory cells of the simulating RAM are partitioned into two sets. The first set contains `cell`(0), `cell`(1),..., `cell`(c) (including the instruction pointer and the accumulator) where c is a constant. This will be called the CPU. The set of remaining cells is called the memory. A program is running in the CPU, and the memory is used only for storing and retrieving individual bits, that is, the content of each memory cell $\texttt{cell}(k+1), \texttt{cell}(k+2), ...$ is always 0 or 1. M_1 and M_2 are functionally equivalent, the time and space requirements of M_2 is increased, compared to M_1 only by a $\texttt{poly}(\log(n))$ factor, and an adversary who knows at each time which instruction is executed in M_2 and what are the addresses of the memory cells involved in the instruction, with high probability does not gain any nontrivial information about the input of the machine. It is an easy consequence of this fact that the simulation of M_1 can also be done by a different type of machine that we will call a composite machine, which have the additional nice property that the adversary may also see the contents of each memory cells in the CPU (but not in the memory) at each time. We only give here the definition of the machine, the proof is based on the idea that each state of the CPU of M_2 can be represented in the memory and the change from such a represented state to the next one can be performed by a sequence of boolean operations on the individual bits of this representation. Moreover the sequence of these boolean operation is given once and for all, it does not depend on the input of the machine. In the next section we give the complete (and final) definition of the composite machine, and in the following section using this definition we continue the sketch of the proof of Theorem 2.

3.5 The composite machine

The composite machine, that will be denoted by $\mathcal{C}_{q,c,k}$, will depend on three integer parameters q, c and k. We will assume that $c > 0$ is a fixed constant, $q > 10$, and k is sufficiently large. The machine $\mathcal{C}_{q,c,k}$ has two parts. The first part will be denoted by $\bar{\mathcal{M}}_{q,c}$. This will work as a RAM $\mathcal{M}_{q,c}$, that is, a RAM with word length q and with c memory cells, but with a special interpretation on its input and output instructions. Namely the output and input instructions will be used only for communication between $\bar{\mathcal{M}}_{q,c}$ and the second part of the composite machine which will be denoted by $\mathcal{A}_k$. $\mathcal{A}_k$ has k memory cells each containing 1 bit. We may think of $\mathcal{C}_{q,c,k}$ as a *RAM* $\mathcal{M}_{q,c}$ with only c memory cells, but with a large external memory $\mathcal{A}_k$ containing k bits. $\mathcal{M}_{q,c}$ through its output instruction is able to perform boolean operations on the bits stored in $\mathcal{A}_k$, and also to move individual bits inside $\mathcal{A}_k$. $\mathcal{M}_{q,c}$ with its input instruction is able to read individual bits stored in $\mathcal{A}_k$.

Definition. We define a machine $\mathcal{A}_k$ with k memory cells $x_0, \ldots, x_{k-1}$ each containing a single bit of information, 0 or 1. The machine will be called a semi-RAM because it will have instructions, but will not have a mechanism to decide which is the next instruction to be executed. Such a decision will come always from an outside source. The machine $\mathcal{A}_k$ has the instructions listed below. After the name of the instruction we describe its effect. (The last instruction does not change the content of any of the memory cells, yet it will have a useful role when we will simulate $\mathcal{A}_n$ with another machine.)

INPUT. The next bit of information from an input buffer, is written into the memory cell x_0.

OUTPUT. The content of memory cell x_0 is given as output.

WRITE i. Assume that A is the content of cell x_0. The content of cell x_i is changed into A.

READ i. Assume that A is the content of cell x_i. The content of cell x_0 is changed into A.

NEGATION. The content of cell x_0 is replaced by its boolean negation.

AND. Assume that A_i is the content of cell x_i for $i = 0, 1$. The content of cell x_0 is replaced by $A_0 \wedge A_1$.

EXCLUSIVE OR. Assume that A_i is the content of cell x_i for $i = 0, 1$. The content of cell x_0 is replaced by $A_0 + A_1$ where $+$ denotes the boolean operation "exclusive or" (or equivalently addition in the field F_2).

RANDOM. A random $0, 1$ bit with uniform distribution is written into cell x_0.

REFRESH. There is no change in the contents of the memory cells. □

Remark. 1. The machine $\mathcal{A}_k$ does not have a way to decide which instruction will be executed. We will define another machine, machine $\bar{\mathcal{M}}_{q,c}$ below which will make this decision.

2. The INPUT and OUTPUT instructions of $\mathcal{A}_k$ communicate with the outside world and *not* with the machine $\bar{\mathcal{M}}_{q,c}$.

3. The operation EXCLUSIVE OR can be easily expressed by the boolean AND and NEGATION. The only reason why it is among the instructions is that later we will simulate $\mathcal{A}_k$ on an $\mathcal{M}_q$ type machine where each bit of $\mathcal{A}_k$ will be represented by a $0, 1$-sequence of length m. The simulation of the EXCLUSIVE OR on $\mathcal{M}_q$ will be much more efficient than the simulation of even a single AND instruction. □

Definition. We define a machine $\mathcal{C} = \mathcal{C}_{q,c,k} = \langle \mathcal{A}_k, \bar{\mathcal{M}}_{q,c} \rangle$ which consists of two parts: a semi-RAM $\mathcal{A}_k$ and a RAM $\bar{\mathcal{M}}_{q,c}$ which is a machine of the type $\mathcal{M}_q$ with exactly c memory cells, but its input and output instruction work in a different way than in $\mathcal{M}_q$. The machine $\mathcal{C}$ will be called a composite-RAM. $\mathcal{C}$ works in the following way. We assume that each of the instruction of $\mathcal{A}_k$ has a name represented by a natural number which can be the content of a memory cell of $\bar{\mathcal{M}}_{q,c}$, and so can be an output of $\bar{\mathcal{M}}_{q,c}$. When the machine starts there is a deterministic program P_0 already in the memory of $\bar{\mathcal{M}}_{q,c}$, which immediately starts to work. Suppose that $\bar{\mathcal{M}}_{q,c}$ gives an output which is the name of an $\mathcal{A}_k$ instruction, possibly together with a parameter. Then $\mathcal{A}_k$ immediately executes this instruction, and after that $\bar{\mathcal{M}}_{q,c}$ continues its work. When $\bar{\mathcal{M}}_{q,c}$ executes an input instruction then the content of the memory cell x_0 of $\mathcal{A}_k$ is written immediately into `cell`(0) of $\bar{\mathcal{M}}_{q,c}$ and $\mathcal{A}_k$ executes a REFRESH instruction. The composite machine $\mathcal{C}_{q,c,k} = \langle \bar{\mathcal{M}}_{q,c}, \mathcal{A}_k \rangle$ works this way till $\bar{\mathcal{M}}_{q,c}$ executes HALT instruction when the machine $\mathcal{C}$ stops. □

Remark. 1. The memories of $\bar{\mathcal{M}}_{q,c}$ and $\mathcal{A}_k$ are completely separate. There is no other information transfer between the two machine, than the ones explicitly described in the definition of $\mathcal{C}$. E.g., the outputs of $\mathcal{A}_k$ will not necessarily reach $\bar{\mathcal{M}}_{q,c}$, we may think of them as outputs to the outside word. Of course if $\bar{\mathcal{M}}_{q,c}$ needs this information it can get it using its own input instruction which copies the content of x_0 into `cell`(0).

2. When $\bar{\mathcal{M}}_{q,c}$ gives an output which is a name of a $\mathcal{A}_k$ instruction then, as we said in the definition, the corresponding instruction is executed in $\mathcal{A}_k$, however this name (or its parameter) is not considered as an input for $\mathcal{A}_k$.

3. There is no instruction of $\bar{\mathcal{M}}_{q,c}$ which writes a $0, 1$-bit into a memory cell of $\mathcal{A}_k$. This however can be accomplished using other instructions. For example, the instruction sequence, "WRITE 1, NEGATION, AND" puts a 0 into x_0.□

3.6 The sketch of the proof of Theorem 2 continued.

The results of [3] and [2] about oblivious simulation has the following easy consequence that we formulate in terms of composite machines. For the sketch of this proof we give only an informal statement.

Lemma A. (Informal statement.) *For each RAM M_1, with n memory cells each containing $q = O(\log n)$ bits, there exists a composite machine $\mathcal{C} = \mathcal{C}_{q,c,k}$ with $k = O(n\texttt{poly}(\log n))$ which is functionally equivalent to M_1 (assuming here that M_1 gets its input and provide its output bit by bit) such that the following holds. Suppose that $\mathcal{X}$ is an adversary who knows at each time the contents of all of the memory cells in $\bar{\mathcal{M}}_{q,c}$ and (as a consequence) which instruction is executed in the machines $\bar{\mathcal{M}}_{q,c}, \mathcal{A}_k$, and which are the addresses of the memory cells involved in these instructions, (but $\mathcal{X}$ does not get directly any information about the contents of the memory cells of $\mathcal{A}_k$). Then $\mathcal{X}$, with high probability, will not get any nontrivial information about the input of the machine M_1.*

The next step in the proof of Theorem 2 is that using Theorem 1 we simulate the composite machine $\mathcal{C}_{c,q,k}$ on a RAM $\mathcal{M}' = \mathcal{P}_{q,m}$, with n' memory cells, where $n' = O(k\texttt{poly}(\log n))$, which gets the input bits in a parity encoded form with encoding sequences of length $m = O(\log n)$. In this simulation we represent each bit b contained in a memory cell x_i of $\mathcal{A}_k$ by a $0,1$ sequence of length m whose parity is b, and whose elements are stored in separate memory cells. $\mathcal{M}'$ preforms each instruction of $\bar{\mathcal{M}}_{q,c}$, with the exception of output and input instructions in their original forms which will not cause any problem since the adversary will know the contents of the memory cells in $\bar{\mathcal{M}}_{q,c}$ anyhow. At an output instruction of $\bar{\mathcal{M}}_{q,c}$, the machine $\mathcal{M}'$ performs the corresponding $\mathcal{A}_k$ instruction using the m-bit representations of the contents of the memory cells of $\mathcal{A}_k$. In the case of the instructions NEGATION, AND and "EXCLUSIVE OR", this step is done by $\mathcal{M}'$ using the block circuits, simulated by the RAM, whose existence is guaranteed by Theorem 1. For instructions WRITE, READ, and REFRESH the circuit $\mathbf{C}_m^{(=)}$ is used. The random instruction is simply the bit by bit randomization of a $0,1$ sequence of length m. The INPUT instruction of $\mathcal{A}_k$ is simulated by $\mathcal{M}'$ by making a $0,1$-sequence of length m from the next m input bits of $\mathcal{M}'$. The output instruction of $\mathcal{A}_k$ is simulated by $\mathcal{M}'$ by giving the bits of the sequence, representing the content of cell x_0, as an output.

Finally when $\bar{\mathcal{M}}_{q,c}$ executes an INPUT instruction then $\mathcal{M}'$ adds the bits of the sequence representing the content of x_0 and writes it into $\texttt{cell}(0)$ of $\bar{\mathcal{M}}_{q,c}$ and then simulates the REFRESH instruction the same way as described above.

The definition of $\mathcal{M}'$ implies that it is functionally equivalent to M_1. As we have indicated already, we apply Theorem 1 to the circuits simulated by $\mathcal{M}'$. Using this and Lemma **A**, we can prove that an ε-random adversary of $\mathcal{M}'$, with high probability does not get any nontrivial information about the input of M_1.

4 Circuits over a finite field

In this section we assume that F is a finite field.

In the following sections we will define several concepts related to block circuits. At the end of the paper we list the frequently used notations, with some hints (but not precise definitions) to their meanings, and the pagenumber of the complete definition.

4.1 Block circuits over a finite field

We will consider circuits in this section whose gates are performing addition and multiplication over a finite field F. For $|F| = 2$ we get the boolean circuits which will be the most important special case.

Definition. $\mathtt{func}(A, B)$ will denote the set of all functions defined on the set A with values in the set B

Definition. 1. Suppose that F is a finite field. A sequence from the elements of F will be called an F-sequence.

2. Assume that F is a finite field. An F-circuit C is a pair $\langle G, f\rangle$ with the following properties: G is an acyclic directed graph on the finite set of vertices V without loops and multiple edges, whose edges are labelled by the elements of the set $\{1, 2\}$, so that the indegree of each vertex is zero, one, or two. The directed graph G will be denoted by $\mathtt{graph}(C)$. The set V will be also denoted by $\mathtt{set}(C)$ and the number of elements of V by $\mathtt{size}(C)$, that is, $\mathtt{size}(C) = |\mathtt{set}(C)|$. The elements of V will be also called the nodes of C.

The source nodes of the graph G are called the input nodes of C, and the sink nodes of G are called the output nodes of C. The set of input nodes will be denoted by $\mathtt{inset}(C)$ the set of output nodes by $\mathtt{outset}(C)$.

f is a labeling of the set $\mathtt{set}(C)\backslash\mathtt{inset}(C)$ with the elements of the set of symbols $\{+1, \times(-1), +, \times, \}$, so that the indegree of a $v \in \mathtt{set}(C)\backslash\mathtt{inset}(C)$ is 1 iff $f(v)$ is the symbol "$+1$" or the symbol "$\times(-1)$". We will say that the node v with the label $f(v)$ is an "$f(v)$ gate". Assume that $v_0, \ldots, v_{k-1}$ are the input nodes of C and $w_0, \ldots, w_{l-1}$ are the output nodes of C. Suppose that $\delta = \langle \delta_0, \ldots, \delta_{k-1}\rangle$ is a sequence from the elements of F. We define the output of C at the input δ in following way:

If $v \in \mathtt{set}(C)$, let $\mathtt{depth}(v)$ be the maximal length of the directed paths in the graph G ending in v. We assign a $0, 1$ value $\chi(v)$ to each $v \in \mathtt{set}(C)$ by recursion on $\mathtt{depth}(v)$. Suppose $\mathtt{depth}(v) = 0$, then v is an input node, and so $v = v_i$ for some $i = 0, 1, \ldots, k-1$. By definition $\chi(v) = \delta_i$. Assume that $\mathtt{depth}(v) > 0$. If $f(v) =$ "$+1$", then $\chi(v) = \chi(u) + 1$, where u is the unique node of G such that there is an edge pointing from u to v. If $f(v) =$ "$\times(-1)$", then $\chi(v) = -\chi(u)$, where u is the unique node of G such that there is an edge pointing from u to v. If $f(v) =$ "$+$" then $\chi(v) = \chi(u_1) + \chi(u_2)$, where u_1, u_2 is the unique pair of nodes so that for $i = 1, 2$ an edge of color i is pointing from u_i to v. In a similar way if $f(v) =$ "$\times$" then $\chi(v) = \chi(u_1) \times \chi(u_2)$. The output of the circuit C at δ is the sequence $\langle \chi(w_0), \ldots, \chi(w_{l-1})\rangle$. The function χ will be called the evaluation function of the circuit C at input δ. If we want to indicate the dependency of χ on C and δ we will write $\chi_\delta^{(C)}$. Sometimes the input δ will not be given as a sequence, but as a function, defined on the set of input nodes, with values in F.

We define a partial ordering $\leq_C$ on $\mathtt{set}(C)$, by "$a \leq_C b$ iff there exists a directed path in $\mathtt{graph}(C)$ (possible of length 0), leading from a to b". We will say that $\leq_C$ is the partial ordering induced by the circuit C.

3. A *probabilistic F-circuit* C is a pair $\langle C', R\rangle$, where C' is an F-circuit and R is a subset of the set of inputs nodes of C'. The nodes in R will be called the probabilistic input nodes and we will use the notation $R = \mathtt{rand}(C)$. The remaining input nodes will be called deterministic input nodes and the set formed by them will be denoted by $\mathtt{detin}(C)$. (Therefore $\mathtt{inset}(C) = \mathtt{detin}(C) \cup \mathtt{rand}(C)$). When the circuit computes a function $f(x_0, \ldots, x_{k-1})$, $x_i \in F$, it gets the elements $x_0, \ldots, x_{k-1}$ through the deterministic input nodes, so in a functional sense, these are the only nodes which transmit the "input". The input given through the deterministic input nodes will be called the deterministic input. The probabilistic input nodes will provide random values which are used for making probabilistic computations with the circuit. We get the output of the probabilistic circuit $C = \langle C', R\rangle$ at a given deterministic input in the following way. For each $w \in \mathtt{rand}(C)$, let ν_w be a random variable which takes its values with uniform distribution in F. We randomize ν_w independently for all $w \in \mathtt{rand}(C)$. Now we consider the F-circuit C' so that its input on $\mathtt{detin}(C)$ is the same as the input of C and its input on the input node w is ν_w for all $w \in \mathtt{rand}(C)$. The output of C' with this input will be the output of C. Therefore the output of a probabilistic circuit C at a given input is a random variable. $\square$

Definition. Assume that m, n are positive integers and C is a probabilistic F-circuit, so that $\mathtt{detin}(C)$ is partitioned into sets $I_0, \ldots, I_{k-1}$ of size m, and the set of the output nodes of C is partitioned into sets $T_0, \ldots, T_{l-1}$ of size n. Then the circuit C, together with the partitions described above, will be called a probabilistic block F-circuit of type $((m, n), k, l)$. If $m = n$ then we will also say that C is of type (m, k, l). We will always assume that a linear ordering of each input block and output block is fixed. If we say that the input on block I_j is $a_0, ..., a_{m-1}$, then we assume that a_0 is assigned the first element of I_j according this ordering, a_1 to the second element etc. $\square$

Definition. Suppose that $W = \langle W_0, ..., W_{s-1}\rangle$ is a finite sequence of pairwise disjoint sets each containing exactly m elements and $a \in \mathtt{func}(\bar{W}, F)$ where $\bar{W} = \bigcup_{i=0}^{s-1} W_i$. $\mathbf{h}_W(a)$ will denote the sequence of length s, whose ith element is $\sum_{x \in W_i} a(x)$. If $W = \langle I_0, ..., I_{k-1}\rangle$ is the sequence of input blocks of the block F-circuit C, and a is the deterministic input of C, then the sequence $\mathbf{h}_W(a)$ is called the block input of the circuit C at input a. Assume now that g is the restriction of the evaluation function $\chi_a^{(C)}$ to the set $\bigcup_{i=0}^{l-1} T_i$, where $W = \langle T_0, ..., T_{l-1}\rangle$ is the sequence of output blocks of the block F-circuit C. Then the sequence $\mathbf{h}_W(g)$, is called the block output of the circuit C. $\square$

4.2 The theorem about private F-circuits

Definition. 1. Assume that the probabilistic F-circuit C has k deterministic input nodes and l output nodes, where F is a finite field. Then C computes a random function, that is, for each $x \in F^k$, by evaluating C we determine the value of a random variable $\xi_{x,C} = \xi_{x,c}^{(F)}$ which takes its values in F^l. The randomness of $\xi_{x,C}$ comes from the probabilistic inputs of C.

2. Assume now that C is a probabilistic block F-circuit with input blocks $I_0, ..., I_{k-1}$ output blocks $T_0, ..., T_{l-1}$ and with blocksize m.

For each $a = \langle a_0, ..., a_{k-1}\rangle \in F^k$ we define a random variable $\eta_{a,C}$ in the following way. First we select km random elements of F, $a_{i,j}$, $i = 0, 1, ..., k-1$, $j = 0, 1, ..., m-1$ independently, with uniform distribution on F and with the condition that $\sum_{j=0}^{m-1} a_{i,j} = a_i$ for $i = 0, 1, ..., m-1$. We call this step the preprocessing.

Then we place the elements $a_{i,0}, ..., a_{i,m-1}$ at the input nodes in I_i in this order, for $i = 0, 1, ..., k-1$. After that the probabilistic F-circuit C is evaluated, and we get ml output elements $b_{i,j}$, $i = 0, 1, ..., m-1$, $j = 0, 1, ..., l-1$, where the elements $b_{i,j}$, $j = 0, 1, ..., m-1$ appear on the output nodes in T_i in this order, for $i = 0, 1, ..., l-1$. The elements $b_0, ..., b_{l-1}$ are defined by $b_i = \sum_{j=0}^{m-1} b_{i,j}$. The computation of these sums is called the postprocessing. The value of the random variable $\eta_{a,C}$ is the vector $b = \langle b_0, ..., b_{l-1}\rangle$. $a = \langle a_0, ..., a_{k-1}\rangle$ will be called the original input of C and $b = \langle b_0, ..., b_{l-1}\rangle$ will be called the original output of C (referring to the motivation that the role of the block circuit C will be to simulate an F-circuit with input a and output b.) according to our earlier definition, we may also refer to the original input as the block input, and refer to the original output as the block output.

3. Assume that C is an F-circuit and $X \subseteq \mathtt{set}(C)$. $\mathtt{full}(X)$ will denote the set of all nodes a of C so that either $a \in X$ or there is and edge pointing from a to and element of x. That is, $\mathtt{full}(X)$ contains the gates in X and all of the gates that produce their inputs.

4. Assume that $\langle C, \langle I_0, ..., I_{k-1}\rangle, \langle T_0, ..., T_{l-1}\rangle\rangle$ is a block circuit with block size m and $\varepsilon > 0$. We define an adversary who is observing the computation done by C, that is, the computation of a value of the random variable $\eta_{a,C}$ for some a, and tries to get some information about the pair $\langle a, b\rangle$. We will call the adversary an ε-random adversary if the information reaching him is determined in the following way.

Each element of $\mathtt{set}(C)$, is declared compromised with a probability of ε so that all of these decisions are mutually independent and also independent from the probabilistic inputs of the circuit C and the randomizations during the preprocessing. Assume that X is the set of compromised elements. The adversary gets all of the values on the elements of $\mathtt{full}(X)$ that we get during the evaluation of the circuit C. More precisely the adversary gets the set W of all pairs $\langle w, \chi^C(w)\rangle$, where $w \in X$. For each fixed $a \in F^k$, and $b \in F^l$, the conditional distribution of W determined by the computation of the value of the random variable $\eta_{a,C}$ and the selections of the compromised gates, with the condition that $\eta_{a,C} = b$, will be denoted by $\Phi_{a,b}^{(\varepsilon,C)}$. We will say that the block circuit C is (ε, p)-secure if, with a probability of at least $1-p$, the ε-random adversary does not gain any information about pair $\langle a, b\rangle$ where a, b are the original input and output. That is, C is (ε, p)-secure if for each $a \in F^k$, there exists an event A_a with respect to all of the randomizations, such that $\mathtt{prob}(A_a) \geq 1-p$, and the conditional distribution of $\Phi_{a,b}^{(\varepsilon,C)}$ with the condition A_a, does not depend on the values of a and b. (We may think that $\neg A_a$ is a small probability event describing a situation when we do not state anything about the knowledge of the adversary.)

Assume now that $\mathcal{C}$ is a probabilistic F-circuit with k deterministic inputs and l outputs, which computes a value of the random variable $\xi_{a,\mathcal{C}}$ at input a, and C is a block F-circuit with k input blocks, l output blocks, and blocksize m which, together with preprocessing and postprocessing computes a value of the random variable $\eta_{a,\mathcal{C}}$ at original input a. We will say that the F-circuit $\mathcal{C}$ and the block F-circuit C are functionally equivalent if for each $a \in F^k$ the random variables $\xi_{a,\mathcal{C}}$ and $\eta_{a,C}$ have identical distributions. $\square$

The following theorem is a generalization of Theorem 1 in the sense that it speaks about Fcircuits and block F-circuits over an arbitrary finite field F while Theorem 1 is only about the boolean, tha is, F_2 case. Note that the choices of the constants ε, c, c_1, c_2 do not depend on the field F,

Theorem 3 *There exist $\varepsilon > 0$, $c > 0, c_1 > 0, c_2 > 0$, such that if m, k, l are positive integers, $m \geq 2$, F is a finite field and $\mathcal{C}$ is a probabilistic F-circuit with k deterministic input nodes and l output nodes, then there exists a block F-circuit C with k input blocks, l output blocks and with block size m, such that, (i) C and $\mathcal{C}$ are functionally equivalent, (ii) $|C| \leq c_1 m^4 |\mathcal{C}|$, and (iii) C is (ε, p)-secure, where $p = e^{-cm}|\mathcal{C}|$.*

Moreover, C can be constructed from $\mathcal{C}$ in time $m^{c_2}|\mathcal{C}|$.

Remark. 1. It is possible to show, with minor modifications of the proof, that for each fixed $\varepsilon_1 > 0$, the upper bound $|C| \leq c_1 m^4 |\mathcal{C}|$ can be replaced by $|C| \leq c_1 m^{3+\varepsilon_1} |\mathcal{C}|$.

4.3 (θ, ε, p)-cylindrical circuits

Definition. 1. Assume that $W = \langle W_0, ..., W_{s-1} \rangle$ is a finite sequence of pairwise disjoint finite sets and F is a finite field. The set of all sequences $f = \langle f_0, f_1, ..., f_{s-1} \rangle$ such that $f_i = \texttt{func}(W_i, F)$ will be denoted by $\texttt{seq}(W, F)$. For such a sequence f and for all $i = 0, 1, ..., s-1$ we define an element $\mathbf{h}_W(f, i) \in F$ by $\mathbf{h}_W(f, i) = \sum_{x \in W_i} f_i(x)$.

If $W = \langle W_0, ..., W_{s-1} \rangle$ is the input/output block sequence of an F-circuit C, we will consider the deterministic input g, which is in $\texttt{func}(W, F)$, also as the sequence $\langle g|_{W_0}, ..., g|_{W_0} \rangle$. If it does not cause any misunderstanding we will denote this sequence also by g.

2. Assume that $A \subseteq B$ are finite sets, F is a finite field, $u \in F$, and $f \in \texttt{func}(A, F)$. Then the set of all functions $g \in \texttt{func}(B, F)$ with $\sum_{x \in B} g(x) = u$ and $f \subseteq g$, will be denoted by $\texttt{extension}_F(f, B, u)$.

Definition. In the following if we say that H is a probability distribution on the finite set X, then we will assume that H is a measure defined on the set of all subsets of X, so that $H(X) = 1$. Sometimes will write $\texttt{prob}_H(Y)$ instead of $H(Y)$. $\square$

Definition. Let $W = \langle W_0, ..., W_{s-1} \rangle$ be a finite sequence of pairwise disjoint finite sets, each containing exactly m elements, and let $\xi = \langle \xi_0, ..., \xi_{s-1} \rangle$ be a random variable so that each value of ξ is an element of $\texttt{seq}(W, F)$ where F is a finite field. Suppose that $\theta \in (0, 1)$. We say that the random variable ξ is θ-cylindrical (or its distribution H is θ-cylindrical) with respect to W and F, if there exists an $Y \subseteq \bar{W} = \bigcup_{i=0}^{s-1} W_i$ and a function $f \in \texttt{func}(Y, F)$ with the following properties:

(a) $|\ Y \cap W_i \ | \leq \theta m$ for all $i = 0, 1, \ldots, s-1$,

(b) Assume that $a = \langle a_0, ..., a_{s-1} \rangle \in \texttt{seq}(W, F)$, and $\texttt{prob}(\xi = a) > 0$. Then for all $x \in Y$, and $i = 0, 1, ..., s-1$, we have $a_i(x) = f(x)$.

(c) Assume that $g \in \texttt{func}(\{0, 1, ..., s-1\}, F)$ such that $\texttt{prob}(\mathcal{A}_g) > 0$, where $\mathcal{A}_g$ is the event $\forall i \in [0, s-1], \sum_{x \in W_i} \xi_i = g(i)$. Then the random variables $\xi_0, ..., \xi_{s-1}$ with the condition $\mathcal{A}_g$ are mutually independent, and with this condition, for each $i = 0, 1, ..., s-1$, the random variable ξ_i has uniform distribution on $\texttt{extension}_F(f_i, W_i, g(i))$, where $f_i = f|_{W_i \cap Y}$.

A function f with the described properties will be called a handle of the random variable ξ (or its distribution H), and the corresponding set $Y = \texttt{domain}(f)$ will be called a base set of ξ (or of H). $\square$

Definition. 1. Assume that C is a probabilistic block F-circuit with the input blocks $I_0, ..., I_{k-1}$, and H is a probability distribution on $\mathtt{func}(\mathtt{detin}(C), F)$. We define a probability distribution $\mathtt{ext}(H)$ on $\mathtt{func}(\mathtt{inset}(C), F)$ which will be called the natural extension of H onto $\mathtt{func}(\mathtt{inset}(C), F)$. We get a random function g with distribution $\mathtt{ext}(H)$, if we first pick a random g' defined on $\mathtt{detin}(C)$ with distribution H and then extend it into a function with values in F and defined on $\mathtt{inset}(C)$, so that all of the possible extensions have the same probability. g' is the extended function.

2. If b is an input for C, that is, a function with values in F defined on $\mathtt{inset}(C)$, then $\mathtt{DET}(b)$ will denote the restriction of b to $\mathtt{detin}(C)$. $\square$

Definition. 1. Assume that C is a probabilistic block F-circuit, H is a probability distribution on $\mathtt{func}(\mathtt{detin}(C), F)$, f is a function defined on a set $X \subseteq \mathtt{set}(C)$ with values in F, and we pick a random input (both deterministic and probabilistic) b for C with distribution $\mathtt{ext}(H)$. If $\mathtt{prob}(\forall x \in X, \chi_b^{(C)}(x) = f(x)) > 0$, then we say that the function f is compatible with the distribution H.

2. Suppose that the function f defined on the set $X \subseteq \mathtt{set}(C)$ is compatible with the distribution H, and we pick a random input b with distribution $\mathtt{ext}(H)$. The conditional distribution of the output of C with the condition $\forall x \in X, \chi_b^{(C)}(x) = f(x)$ will be denoted by $\mathtt{cond}_{\mathtt{output}}(H, f)$. The distribution of $\mathtt{DET}(b)$ with the condition $\forall x \in X, \chi_a^{(C)}(x) = f(x)$ will be denoted by $\mathtt{cond}_{\mathtt{detin}}(H, f)$. $\square$

Remark. This last definition is motivated by the fact that if an adversary knows the function f and knows that $\chi_b^{(C)}(x) = f(x)$ for all $x \in X$ and that the distribution of the deterministic input was H, then from the point of view of the adversary, that is, conditioned by the adversary's knowledge, the distribution of the deterministic input will be $\mathtt{cond}_{\mathtt{detin}}(H, f)$, and the analogue statement holds for the output. $\square$

Proposition 1 *Suppose that m is a positive integer, $0 < \theta < \frac{1}{2}$, and $\xi = \langle \xi_0, ..., \xi_{s-1} \rangle$ is a θ-cylindrical random variable with respect to W and F, where F is a finite field and $W = \langle W_0, ..., W_{s-1} \rangle$ is a finite sequence of pairwise disjoint sets, each containing exactly m elements. Then H has a unique handle and a unique base set.*

Proof. The definition of a θ-cylindrical random variable ξ states that ξ has a handle and a base set. We have to prove only their uniqueness. Let $\bar{\xi}$ be the unique joint extensions of the functions $\xi_0, ..., \xi_{s-1}$ to the set $\bar{W} = \bigcup_{i=0}^{s-1} W_i$. First we note that if there is only a single base set Y then property (b) of the definition of an θ-cylindrical distribution implies that for all handles f, g of H, f and g are identical on Y and $\mathtt{domain}(f) = \mathtt{domain}(g)$, consequently $f = g$. Therefore it is sufficient to prove the uniqueness of the base set. Suppose that $h \in \mathtt{func}(Z, F)$ where Z is a subset of $\bar{W}$. In the proof below we will say that h is ξ-positive if it has an extension a onto $\bar{W}$ so that $\mathtt{prob}(\bar{\xi} = a) > 0$.

Assume that contrary to our statement, Y and Y' are base sets of ξ and $Y' \backslash Y \neq \emptyset$ and f is a handle of H with $\mathtt{domain}(f) = Y$. Clearly f is ξ-positive. $|Y \cap W_i| \leq \theta m < \frac{m}{2}$ for all $i = 0, 1, ..., m-1$, therefore property (c) of the definition of a θ-cylindrical distribution implies that all of the extensions $g \in \mathtt{func}(Y \cup Y', F)$ of f are also ξ-positive. (This is a consequence of the fact that $|T_i \cap Y \cup Y'| < m$ an so such a function g can be extended to a function

$g' \in \mathtt{func}(\bar{W}, F)$ so that $h_W(g', i)$ as a function of i can be arbitrary.) Since $|F| \geq 2$ this implies that there exist two distinct ξ-positive extensions g_0, g_1 of F onto $Y \cup Y'$. Since g_0, g_1 are identical to f on Y their restrictions g_0', g_1' to Y' must be distinct. We got that there are two distinct ξ-positive functions g_0', g_1' defined on the same base set Y'. This however contradicts property (b) of the definition of a θ-cylindrical distribution. *Q.E.D.*(Proposition 1)

Proposition 2 *Suppose that m is a positive integer, $0 < \theta < \frac{1}{2}$, η is a random variable taking its values in the finite set A, and for all $a \in A$, $\xi^{(a)}$ is a θ-cylindrical random variable to W and F, where F is a finite field and $W = \langle W_0, ..., W_{s-1} \rangle$ is a finite sequence of pairwise disjoint sets, each containing exactly m elements. Assume further that all of the random variables ξ_a, $a \in A$ has the same base set B and the same handle f. Let ξ be a random variable defined in the following way. To select a random value of ξ first we choose a random value "a" of η then a random value b of $\xi^{(a)}$. The element b is the random value of ξ. In other words $\xi = \xi_\eta$. Then ξ is a θ-cylindrical random variable with base set B and handle f.*

Proof. We have to show that the random variable ξ satisfies conditions (a), (b), (c) of the definition of a θ-cylindrical distribution. We will use the uniqueness of the base set and the handle, proved in Proposition 1, of the distribution ξ^a, $a \in A$. Condition (a) is clearly satisfied since it holds for each distribution ξ^a.

Condition (b). Let $\bar{\xi} = \xi_0 \cup ... \cup \xi_{s-1}$ and assume that $\mathtt{prob}(\xi = \langle a_0, ..., a_{s-1} \rangle) > 0$. Then, by the definition of ξ, there exists a $b \in A$ such that $\mathtt{prob}(\xi^{(b)} = \langle a_0, ..., a_{s-1} \rangle) > 0$. Since condition (b) holds for the random variable ξ^b, we have that for all $x \in B \cap W_i$, $a_i(x) = f(x)$ for $i = 0, 1, ..., s-1$.

Condition (c) is an immediate consequence of the fact that it holds separately for each fixed random variable $\xi^{(a)}$, $a \in A$ with the same base set and handle. *Q.E.D.*(Proposition 2)

Proposition 3 *Suppose that m is a positive integer, $0 < \theta < \frac{1}{2}$, $\xi = \langle \xi_0, ..., \xi_{s-1} \rangle$ is a θ-cylindrical random variable with respect to W and F, where F is a finite field and $W = \langle W_0, ..., W_{s-1} \rangle$ is a finite sequence of pairwise disjoint sets, each containing exactly m elements. Assume further that , π is a permutation of the set $0, 1, ..., s-1$ and $W^{(\pi} = \langle W_{\pi(0))}, W_{\pi(1)}, ..., W_{\pi(s-1)} \rangle$. Let $\xi^{(\pi)}$ be the random variable defined by $\xi^{(\pi)} = \langle \xi_{\pi(0)}, ..., \xi_{\pi(s-1)} \rangle$. Then $\xi^{(\pi)}$ is a θ cylindrical distribution with respect to $W(\pi)$ and F.*

Proof. Since the conditions of the definition of θ-cylindricity do not depend on the order of the elements of the sequence $\langle W_0, ..., W_{s-1} \rangle$, the statement of the proposition is an immediate consequence of that definition. *Q.E.D.*(Proposition 3)

Proposition 4 *Suppose that m is a positive integer, $0 < \theta < \frac{1}{2}$, $W = \langle W_0, ..., W_{s-1} \rangle$, where W_i, $i = 0, 1, ..., s-1$, are pairwise disjoint finite sets each with m elements, and $\xi = \langle \xi_0, ..., \xi_{s-1} \rangle$ is a random variable with θ-cylindrical distribution. Assume further that $i \in \{0, 1, ..., s-1\}$, $a_j \in \mathtt{func}(W_j, F)$ for all $j = 0, 1, ..., i-1$, and $\mathtt{prob}(\xi_0 = a_0 \wedge ... \wedge \xi_j = a_j) > 0$. Then the conditional distribution of the sequence $\langle \xi_i, \xi_{i+1}, ..., \xi_{s-1} \rangle$ with the condition $\xi_0 = a_0, ..., \xi_{i-1} = a_{i-1}$ is θ-cylindrical with respect to $\langle W_i, W_{i+1}, ..., W_{s-1} \rangle$ and F.*

Proof. We claim that if B is the base set and f is the handle of the of the random variable ξ and $W' = \bigcup_{j=i}^{s-1} W_i$ then $B \cap W'$ is the base set and $f|_{W'}$ is the handle of the mentioned conditional distribution of the random variable $\langle \xi_i, ..., \xi_i \rangle$. Conditions (a), (b), and (c) are immediate consequence of the corresponding conditions for the random variable ξ. *Q.E.D.*(Proposition 4)

We define now a set of adversaries for a block circuit C who will be stronger than an ε-random adversary, in the sense that they will have at least as much information as the ε-random adversary. The advantage of the newly defined adversaries will be that their knowledge of the output of the circuit can be described more easily than for an arbitrary ε-random adversary.

Definition. 1. The set of all subsets of the set X will be denoted by $\texttt{pow}(X)$.

2. Suppose that A is a finite set. We will say that a randomly chosen set X is an ε-random subset of A if for each $a \in A$ we have $\texttt{prob}(a \in X) = \varepsilon$, and the various events $a \in X$, for $a \in A$ are mutually independent.

3. Assume that, C is a probabilistic block F-circuit. A selection function of C is a function $\mathcal{S}$ in $\texttt{func}(\texttt{pow}(\texttt{set}(C)), \texttt{pow}(\texttt{set}(C)))$.

Definition. We assume that C is a probabilistic block F-circuit, and its deterministic input arrives with a θ-cylindrical distribution H defined on $\texttt{func}(\texttt{detin}(C), F)$ for some $\theta > 0$. We define an adversary against the circuit C. This adversary will be uniquely determined by the real $\varepsilon > 0$ and a selection function $\mathcal{S}$ and will be denoted by $\texttt{adv}_C(\varepsilon, \mathcal{S})$. The adversary will be called the ε-random adversary with selection function $\mathcal{S}$. The adversary $\mathcal{Y} = \texttt{adv}_C(\varepsilon, \mathcal{S})$ gets the following information while the circuit is evaluated at input a, which was selected with distribution H:

(9) *the base B_H and the handle f_H of distribution H,*

(10) *an ε-random subset set X of $\texttt{set}(C)$ is chosen, then $\mathcal{Y}$ gets the set $V = \{\langle w, \chi_a^{(C)}(w) \rangle \mid w \in \texttt{full}(X \cup \mathcal{S}(X))\}$. That is, $\mathcal{Y}$ gets the evaluation values at each node in $\texttt{full}(X \cup \mathcal{S}(X))$ at input a.*

This completes the description of the adversary $\mathcal{Y}$. The random set X in condition (10) will be called the ε-random subset of the adversary. The triplet $Z = \langle V, B_H, f_H \rangle$ is the knowledge acquired by the adversary $\mathcal{Y}$ during the evaluation of the F-circuit C.

For each adversary $\mathcal{Y} = \texttt{adv}_C(\varepsilon, \mathcal{S})$ we define another adversary $\mathcal{Y}^+ = \texttt{adv}^+_C(\varepsilon, \mathcal{S})$. $\mathcal{Y}^+$ gets all the information that reaches $\mathcal{Y}$, and $\mathcal{Y}^+$ also gets each element of the deterministic input, that is, an F-valued function a defined on $\texttt{detin}(C)$ so that for each $a \in \texttt{detin}(C)$, $a(x)$ is the input at the input node x. Therefore the knowledge of the adversary $\mathcal{Y}^+$ is the quadruplet $Z = \langle V, B_H, f_H, a \rangle$.

Assume that ξ is an arbitrary random variable. For a fixed distribution H of the input, the knowledge Z of the adversary $\mathcal{X}$, where $\mathcal{X} = \mathcal{Y}$ or $\mathcal{X} = \mathcal{Y}^+$, is a random variable. The distribution of ξ from the point of view of the adversary $\mathcal{X}$ with knowledge Z_0 is defined as the conditional distribution of ξ with the condition $Z = Z_0$. $\Box$

The definition of the adversary $\mathcal{Y}^+$ seems pointless, since $\mathcal{Y}^+$ knows the complete deterministic input so we may think that there is nothing to hide from $\mathcal{Y}^+$. This is not completely true since $\mathcal{Y}^+$ may not know all of the probabilistic inputs of C. The adversary $\mathcal{Y}^+$ will be used to speak about the randomness in the output of the circuit C which is coming from its random inputs.

For the next definitions recall, that if C is a block F-circuit with k input blocks $I_0, ..., I_{k-1}$ with blocksizes m, and on block I_j the input values are $a_{i,j}$, $j = 0, 1, ..., m-1$, then the original input or block input is defined as the sequence $\langle \sum_{j=0}^{m-1} a_{0,j}, \ldots \sum_{j=0}^{m-1} a_{k-1,j} \rangle$. Sometimes we will consider this sequence as a function g defined on the set $\{0, 1, ..., k-1\}$ such that $g(i) = \sum_{j=0}^{m-1} a_{i,j}$.

Definition. 1. Suppose that H_1, H_2 are θ-cylindrical distributions with respect to W, F, where $W = \langle W_0, ..., W_{s-1} \rangle$ is a finite sequence of finite sets, and F is a finite field. We will say that H_1 and H_2 are similar if they have the same base set and the same handle.

2. Assume that $\xi = \langle \xi_0, ..., \xi_{s-1} \rangle$ is a θ-cylindrical random variable with respect to $W = \langle W_0, ..., W_{s-1} \rangle$ and F. ξ is called pure, if there exists a $g \in \mathtt{func}(\{1, ..., s-1\})$ such that for all sequences $a = \langle a_0, ..., a_{s-1} \rangle$ with $a_i \in \mathtt{func}(W_i, F)$, we have that $\mathtt{prob}(\xi = a) > 0$ implies $\mathbf{h}_W(a, i) = g(i)$ for $i = 0, 1, ..., s-1$.

Remark. Assume that the sequence $W_0, ..., W_{i-1}$ is the sequence of input blocks of a block F-circuit C. Then ξ is pure iff there is only one possible block input (i.e., original input) such that the value of ξ has this block input with a positive probability.

Definition. 1. Assume that $\theta > 0$, $W = \langle W_0, ..., W_{s-1} \rangle$, is a sequence of finite sets with $|W_i| = m$, for $i = 0, 1, ..., s-1$. Then the set of all functions f such that $\mathtt{domain}(f) \subseteq \bigcup_{i=0}^{s-1} W_i$, $\mathtt{range}(f) \subseteq F$ and $|\mathtt{domain}(f) \cap W_i| \leq \theta m$ for $i = 0, 1, ..., s-1$ will be denoted by $\mathtt{func}_\theta(W, F)$.

2. Assume that $\mathcal{X}$ is an adversary for a block F-circuit C. If the deterministic input of the circuit C is arriving with distribution H then the distribution of the knowledge of the adversary $\mathcal{X}$ will be denoted $\mathcal{D}_{\mathcal{X},H}$. Let $\mathcal{Y} = \mathtt{adv}_C(\varepsilon, \mathcal{S})$ be an ε-random adversary, and let $X \subseteq \mathtt{set}(C)$. The conditional distribution of the knowledge of the adversary $\mathcal{Y}$ with the condition that the ε-random subset that $\mathcal{Y}$ is using is the set X, will be denoted by $\mathcal{D}_{\mathcal{Y},H,X}$, where the input is arriving with distribution H.

3. Assume that $\theta > 0, \varepsilon > 0, p \in [0,1]$ and $\mathcal{Y} = \mathtt{adv}_C(\varepsilon, \mathcal{S})$ is an ε-random adversary with selection function $\mathcal{S}$ for a probabilistic block F-circuit C with block size m and input block sequence $I = \langle I_0, ..., I_{k-1} \rangle$. The adversary $\mathcal{Y}$ is called a (θ, ε, p)-cylindrical adversary, if for all function $f \in \mathtt{func}_\theta(I, F)$ we have that with a probability of at least $1-p$, with respect to the randomization of the ε-random set X used by $\mathcal{Y}$, the following two conditions are satisfied:

(11) *for each fixed deterministic input of the circuit C the corresponding output, from the point of view of adversary $\mathcal{Y}^+ = \mathtt{adv}_C^+(\varepsilon, \mathcal{S})$, has a θ-cylindrical distribution, with base set $(X \cup \mathcal{S}(X)) \cap \bigcup_{i=0}^{l-1} T_i$, with respect to the sequence of all output blocks $\langle T_0, T_1, ..., T_{l-1} \rangle$ and F.*

(12) *Suppose that H_0, H_1 are similar and pure θ-cylindrical distributions for the deterministic input of C whose common handle is f. Then $\mathcal{D}_{\mathcal{Y},H_0,X} = \mathcal{D}_{\mathcal{Y},H_1,X}$.*

A set X satisfying conditions (11) and (12) will be called an acceptable set with respect to adversary $\mathcal{Y}$ and handle f. $\square$

Proposition 5 *Condition* (11), in the definition of a (θ, ε, p)-cylindrical adversary, implies that the following condition is also satisfied by such an adversary.

(13) *if the input is arriving with a θ-cylindrical distribution H then the distribution of the output, from the point of view of adversary $\mathcal{Y} = \mathtt{adv}_C(\varepsilon, \mathcal{S})$, is θ-cylindrical with base set $\mathcal{S}(X) \cap \bigcup_{i=0}^{l-1} T_i$.*

Moreover condition (12) of the definition is equivalent to the following condition, (where the assumption "H_i is pure" is omitted form condition (12)):

(14) *Suppose that H_0, H_1 are similar θ-cylindrical distributions for the deterministic input of C. Then $\mathcal{D}_{\mathcal{Y},H_0,X} = \mathcal{D}_{\mathcal{Y},H_1,X}$.*

Proof. The second statement of the proposition is an immediate consequence of the fact that an arbitrary θ-cylindrical distribution H is a mixture of pure θ-cylindrical distributions H_α, $\alpha \in A$ with the same base sets and handles.

For the proof of the first statement we use this fact as well. From the point of view of the adversary $\mathcal{Y}$ if the input arrived with distribution H_α then the output of circuit C has θ-cylindrical distribution G_α with base B and handle f. Since B is determined by the function $\mathcal{S}$ and f is determined by the values of the output of C on the set B, which is known to $\mathcal{Y}$, we get that B and f from the point of view of the adversary $\mathcal{Y}$ does not depend on α. Therefore for all $\alpha \in A$ the distributions of the output, from the point of view of $\mathcal{Y}$, and with the condition that the input came with distribution H_α, are θ-cylindrical with the same base set and handle. Consequently Proposition 2 implies that if the input comes with distribution the θ-cylindrical distribution H, then the distribution of the output is θ-cylindrical from the point of view of $\mathcal{Y}$. *Q.E.D.*(Proposition 5)

Definition. Assume that $\theta > 0, \varepsilon > 0, p \in [0, 1]$ and C is a probabilistic block F-circuit. We will say that C is a (θ, ε, p)-cylindrical circuit, if there exists a selection function $\mathcal{S}$, such that $\mathcal{Y} = \mathtt{adv}(\theta, \varepsilon, \mathcal{S})$ is (θ, ε, p)-cylindrical adversary for the circuit C. In this case we will also say that C is a cylindrical circuit with concentration θ, leak probability ε, and error probability p. $\Box$

The motivation for the definition of an (θ, ε, p)-cylindrical circuit is the following. Assume that an ε-random adversary is watching the circuit. We want to construct circuits which can serve as secure components of a larger circuit. That is, we want to define a secure circuit so that if a larger circuit is put together from several secure circuits then the adversary, with high probability, will not gain any information about the original input. We claim that "(θ, ε, p)-cylindrical circuits" can play the role of "secure circuits" if p is small.

Definition. Assume that C is a probabilistic block F-circuit. Let $H = \mathtt{inset}(C)$. If $h \in H$ then we associate with H an indeterminate z_h. For each node a of C we define a polynomial $p_a \in F[z_h \mid h \in H]$. We define p_a by recursion on the depth of a in the same way as we have defined the evaluation function χ of a circuit. Namely for each input node a, we have $p_a = z_a$. Assume that a is not an input node, the indegree of a is one, a is labelled with "+1", and there is an edge pointing from u to a. Then $p_a = p_u + 1$. In a similar way if a is labelled by $\times(-1)$, then $p_a = -p_u$. Suppose now that the indegree of a is 2 and u_0, u_1 are distinct nodes so that there is an edge pointing from u_i to a for $i = 0, 1$. Then according to whether the label of a is $+$, or $\times$, we have $p_a = p_{u_0} + p_{u_1}$, or $p_a = p_{u_0} \times p_{u_1}$.

The polynomials p_a, $a \in \mathtt{set}(C)$ will be called the evaluating polynomials of the F-circuit C.

Assume that $I_0, \ldots, I_{k-1}$ are the input blocks of the block F-circuit C and $T_0, \ldots, T_{l-1}$ are the output blocks of the block F-circuit C of type $((n,m),k,l)$. We define polynomials $P_0, \ldots, P_{k-1}$ and $Q_0, \ldots, Q_{l-1}$ by $P_j = \sum_{h \in I_j} p_h = \sum_{h \in I_j} z_h$ for $j = 0,1,\ldots,k-1$, and $Q_j = \sum_{a \in T_j} p_a$, $j = 0,1,\ldots,l-1$. $P_0, \ldots, P_{k-1}$ will be called the input block polynomials and $Q_0, \ldots, Q_{l-1}$ the output block polynomials. $\square$

Definition. Assume that f, g are elements of the polynomial ring $F[z_0, \ldots, z_{r-1}]$, where F is a field. We will say that f and g are functionally equivalent over F, and we will write $f \equiv_F g$, if for all $a_0, .., a_{r-1} \in F$, we have $f(a_0, \ldots, a_{r-1}) = g(a_0, \ldots, a_{r-1})$. Since a field F is fixed in the remaining part of the paper, "f is functionally equivalent to g" or $f \equiv g$ will mean that f is functionally equivalent to g over the field F that we have fixed. $\square$

Definition. Assume that $F_0(x_0, \ldots, x_{k-1}), \ldots, F_{l-1}(x_0, \ldots, x_{k-1})$ are polynomials over the field F. We say that the probabilistic block F-circuit C of type $((m,n),k,l)$ computes the polynomials $F_0, \ldots, F_{l-1}$, if the following holds. For all $i = 0,1,\ldots,l-1$, $F_i(P_0, \ldots, P_{k-1}) \equiv Q_i$, where $P_0, \ldots, P_{k-1}, Q_0, \ldots, Q_{l-1}$ are the block polynomials of C. $\square$

Definition. $a \ll b_1, ..., b_k$ will mean "a is sufficiently small with respect to $b_1, ..., b_k$. $\square$

Lemma 3 *Assume that the reals ε, c, θ are chosen so that $0 < \varepsilon \ll c \ll \theta \ll 1$, the integer m is sufficiently large, and F is a finite field. Then there exist probabilistic block F-circuits $C_i^{(m)}$, for $i = 0,1,2,3,4,5,6,7$ with the following properties:*

(15) *for $i = 0,1,2,3,4,5,6.7$, $C_i^{(m)}$ is of polynomial size in m, and $\mathtt{graph}(C_i)^{(m)}$, with its labeling, can be computed in time polynomial in m,*

(16) *for $i = 0,1,2,3,4,5,6,7$, the circuit $C_i^{(m)}$ is (θ, ε, p)-cylindrical, where $p = e^{-cm}$,*

(17) *$C_0^{(m)}$ is of type $((m,m),1,1)$ and computes the polynomial x_0*

(18) *$C_1^{(m)}$ is of type $((m,m),1,2)$ and computes the polynomials x_0, x_0*

(19) *$C_2^{(m)}$ is of type $((m,m),1,1)$ and computes the polynomial $1 + x_0$*

(20) *$C_3^{(m)}$ is of type $((m,m),1,1)$ and computes the polynomial $-x_0$*

(21) *$C_4^{(m)}$ is of type $((m,m,),2,2)$ and computes the polynomials x_0, x_0x_1*

(22) *$C_5^{(m)}$ is of type $((m,m,),2,1)$ and computes the polynomial x_0x_1*

(23) *$C_6^{(m)}$ is of type $((m,m,),2,2)$ and computes the polynomials $x_0, x_0 + x_1$*

(24) *$C_7^{(m)}$ is of type $((m,m,),2,1)$ and computes the polynomials $x_0 + x_1$*

(25) *for* $i = 0, 1, 2, 3$*, the outdegree of each node of* $\mathtt{graph}(C_i^{(m)})$ *remains below an absolute constant.*

We will prove this lemma in the following sections.

Remark. We use Lemma 3 for the proof of Theorem 3. Statements (23) and (25) are not needed for this proof we will use them in the proofs about leak resistant RAMs.

Lemma 4 *Lemma 3 implies Theorem 3.*

Proof. For the sake of simplicity we prove the lemma first in the special case when the F-circuit $\mathcal{C}$ has no probabilistic input. The proof of the general case can be obtained from this proof with minor modifications as we will indicate later. (This is only an assumption about the circuit $\mathcal{C}$, the block circuit C whose existence is stated in the theorem will be probabilistic.) Assume that the statement of Lemma 3 is true and an F-circuit $\mathcal{C}$ without probabilistic inputs is given with the properties described in the assumptions of Theorem 3. For the proof of this theorem we allow in $\mathcal{C}$ an identity gate also called an $=$ gate whose output is the same as its input. If we allow this in $\mathcal{C}$ then we may assume without the loss of generality that all of the $=$ gates in $\mathcal{C}$ have fan-out 1 or 2, all other gates in $\mathcal{C}$ have fan-outs exactly 1, and the input nodes also have fan-outs exactly 1. We may also assume without the loss of generality that the output of each deterministic input node is the input of an $=$ gate, that is, the deterministic inputs are simply copied before used. This last assumption will make possible to avoid exceptional cases in our inductive proof. The part of the circuit $\mathcal{C}$ consisting of $\mathtt{detin}(\mathcal{C})$ the set of deterministic input nodes of $\mathcal{C}$ and those nodes of $\mathcal{C}$ who get their input from $\mathtt{detin}(\mathcal{C})$ will be denoted by $\mathcal{C}_0$. Clearly $|\mathcal{C}_0| = 2|\mathtt{detin}(\mathcal{C})|$.

Our assumption about the F-circuit $\mathcal{C}$ implies that there exists a sequencence of F-circuits $\mathcal{C}_0, ..., \mathcal{C}_t$, such that for each $i = 0, 1, ..., t$, $\mathtt{set}(\mathcal{C}_i)$ is an upward closed subset of $\mathtt{set}(\mathcal{C})$ with respect to $\leq_{\mathcal{C}}$ (recall that the input nodes are the maximal elements of $\leq_{\mathcal{C}}$) and whose edges and labeling are inherited from $\mathcal{C}$, and the following conditions are satisfied:

(26) $\mathcal{C}_0$ *is the circuit containing the input nodes and the nodes immediately below them, as defined above.*

(27) $\mathtt{set}(\mathcal{C}_i) \subset \mathtt{set}(\mathcal{C}_{i+1})$ *for* $i = 0, 1, ..., t-1$*, and* $\mathcal{C}_t = \mathcal{C}$,

(28) *for all* $i = 0, 1, ..., t-1$*,* $1 \leq |\mathtt{set}(\mathcal{C}_{i+1}) \backslash \mathtt{set}(\mathcal{C}_i)| \leq 2$,

(29) *for all* $i = 0, 1, ..., t-1$*,* $|\mathtt{set}(\mathcal{C}_{i+1}) \backslash \mathtt{set}(\mathcal{C}_i)| = 2$*, implies that the two elements of* $\mathtt{set}(\mathcal{C}_{i+1}) \backslash \mathtt{set}(\mathcal{C}_i)$ *are equality gates which get their inputs from the same node of* $\mathcal{C}_i$,

(30) *for all* $i = 0, 1, ..., t-1$*,* $|\mathtt{set}(\mathcal{C}_{i+1}) \backslash \mathtt{set}(\mathcal{C}_i)| = 1$*, implies that if* $a \in \mathtt{set}(\mathcal{C}_{i+1}) \backslash \mathtt{set}(\mathcal{C}_i)$ *is an equality gate whose input comes from the node* $b \in \mathcal{C}_i$*, then the fan-out of* b *in* $\mathcal{C}$ *is* 1.

Such a sequence $\mathcal{C}_0, ..., \mathcal{C}_i, ..., \mathcal{C}_t$ can be easily constructed by recursion on i. The definition of $\mathcal{C}_0$ is given in condition (26). If $\mathcal{C}_0, ..., \mathcal{C}_i$ are already given with the required properties then let a be a maximal element in the partial orders set $\leq_{\mathcal{C}}$ which is not in $\mathcal{C}_i$. If a is not an equality gate then $\mathtt{set}(\mathcal{C}_{i+1}) = \mathtt{set}(\mathcal{C}_i) \cup \{a\}$. Assume that a is an $=$ gate and gets its input from the node $b \in \mathtt{set}(\mathcal{C}_i)$. If the fanout of b is one then $\mathtt{set}(\mathcal{C}_{i+1}) = \mathtt{set}(\mathcal{C}_i) \cup \{a\}$. If the fanout of b is 2 then let $a' \neq a$ be the other node of $\mathcal{C}$ which gets its input from b. We claim that $a' \notin \mathcal{C}_i$. Assume that contrary to our statement $a' \in \mathcal{C}_i$. Let j be the smallest nonnegative integer with $a' \in \mathcal{C}_j$. Since b is an equality gate with fan-out 2 and $a' < b$ we have that $j \geq 1$. We got $a \notin \mathcal{C}_j$ and $a' \in \mathcal{C}_j \backslash \mathcal{C}_{j-1}$ and a, a' are both equality gates getting their input from b. This howerver constradicts to condition (30) of the inductive assumption. Therefore $a' \notin \mathcal{C}_i$. We define $\mathcal{C}_{i+1}$ by $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{a, a'\}$.

We construct the probabilistic block F-circuit C in the following way. Assume that the input nodes of $\mathcal{C}$ are $a_0, ..., a_{r-1}$. We replace each input node a_i of $\mathcal{C}$ with an input block I_i consisting of m elements. Then we replace each gate of $\mathcal{C}$ by a copy of a block circuit whose existence is guaranteed by Lemma 3 according to the following rules: for $=$ gates with fan-out 1 we use $C_0^{(m)}$, for $=$ gates with fan-out 2 we use $C_1^{(m)}$, for $+1$ gates we use $C_2^{(m)}$, for $\times(-1)$ we use $C_3^{(m)}$, for $\times$ gates we use $C_5^{(m)}$, and for $+$ gates we use $C_7^{(m)}$. We assume that the copies of the circuits used for these replacements are pairwise disjoint. Assume that the gate x was replaced by the circuit $G(x)$. If x, y are nodes of $\mathcal{C}$ and the ith output of the y is the jth input of x then we identify the ith output block of y with the jth input block of x. We extend this rule for the case when y is an input node of $\mathcal{C}$. (Since the fan-out of each gate is at most the number of output blocks of the corresponding block circuit, we identify with each input/output element of a gate with at most one other element, that is, the identification does not extend because of the transitivity of equality.) This completes the definition of the circuit C. For each $i = 0, 1, ..., t$, C_i will denote the block F-circuit that we get from the F-circuit $\mathcal{C}_i$ with this construction.

By Lemma 3, each circuit that we have used in this construction is functionally equivalent to the gate replaced by it. Consequently the block F-circuit C_i is functionally equivalent to the F-circuit $\mathcal{C}_i$ for $i = 0, 1, ..., t$. We show by induction on i that

(31) *for all $i = 0, 1, ..., t$ the block F-circuit C_i is $(\theta, \varepsilon, p')$-cylindrical where $p' = |\mathcal{C}_i| e^{-cm}$.*

$i = 0$. recall that $\mathcal{C}_0$ is defined by condition (26). If the input nodes of the circuit C are $a_0, ..., a_{r-1}$ then C_0 consists of r pairwise disjoint blockcircuits $D_0, ..., D_{r-1}$, such that each of them is a copy of $C_0^{(m)}$. To show that C_0 is (θ, ε, p)-cylindrical with $p = |\mathcal{C}_0| e^{-cm}$, we have to show that there exists an (θ, ε, p)-cylindrical adversary $\mathcal{Y}$ for C_0. For the definition such an adversary $\mathcal{Y}$ we have to define its selection function $\mathcal{S}$. According to Lemma 3, each block circuit D_i is $(\theta, \varepsilon, e^{-cm})$-cylindrical so it has an $(\theta, \varepsilon, e^{-cm})$-cylindrical adversary $\mathcal{Y}_i$ whose selection function will be denoted by $\mathcal{S}_i$ for $i = 0, 1, ..., r-1$. The selection function $\mathcal{S}$ is defined in the following way: for all $X \subseteq \mathtt{set}(C_0)$, $\mathcal{S}(X) = \bigcup_{i=0}^{r-1} \mathcal{S}_i(X \cap \mathtt{set}(D_i))$.

We show that the adversary $\mathcal{Y}$ defined this way satisfies the conditions of $(\theta, \varepsilon, |\mathcal{C}_0| e^{-cm})$-cylindricity. First we choose an ε-random subset X of $\mathtt{set}(C_0)$. Lemma 3 implies that for each fixed $i = 0, 1, ..., r-1$ with a probability of at least $1 - e^{-cm}$ the set $X \cap \mathtt{set}(D_i)$ is an acceptable set with respect to adversary $\mathcal{Y}_i$, that is, it satisfies conditions (11), (12) of the definition of $(\theta, \varepsilon, e^{-cm})$-cylindricity for the circuit D_i. We claim that

(32) *if for all $i = 0, 1, ..., r-1$, the sets $X \cap D_i$ are acceptable with respect to $\mathcal{Y}_i$, then the set X is acceptable with respect to $\mathcal{Y}$.*

Since $X \cap \texttt{set}(D_i)$, will be acceptable for all $i = 0, 1, ..., r-1$, the probability that the assumption of condition (32) is true is at least. $1 - re^{-cm}$. Consequently condition (32) implies that C_0 has a $(\theta, \varepsilon, |\mathcal{C}_0|e^{-cm})$-cylindrical adversary.

For the proof of condition (32) assume that $X \cap D_i$ is acceptable for $i = 0, 1, ..., r-1$. We have to show that X is acceptable as well, that is, conditions (11) and (12) are satisfied by X and C_0.

Condition (11) for X and C_0. By the definition of $\mathcal{Y}^+$ we may assume that the input of circuit C_0 is fixed and consider the distribution of the output of C_0 conditioned with the knowledge of the adversary which includes the input and the evaluation values of the circuit on the set $\texttt{full}(X \cup \mathcal{S}(X))$.

We know that the output of the circuit D_i has θ-cylindrical distribution from the point of view of $\mathcal{Y}^+$. Let B_i be the base set and let f_i be the handle of this distribution. We claim that the output of C_0 has θ-cylindrical distribution from the point of view of $\mathcal{Y}^+$ with base set $\bigcup_{i=0}^{r-1} B_i$ and handle $\bigcup_{i=0}^{r-1} f_i$. We have to show that conditions (a), (b), and (c) of the definition of a θ-cylindrical distribution are satisfied for the distribution of the output of C_0. Conditions (a) and (b) follow immediately about the corresponding conditions for the outputs of D_i, $i = 0, 1, ..., r-1$. For the proof of condition (c) assume that $a = \langle a_1, ..., a_{r-1}\rangle$ is the output sequence of C_0 where $a_i \in \texttt{func}(T_i, F)$ and $T = \langle T_0, ..., T_{r-1}\rangle$ is the output block sequence of C_0. Assume that $g \in \texttt{func}(\{0, 1, ..., r-1\}, F)$ and we consider the distribution of a with the additional condition $\mathbf{h}_T(a, i) = g(i)$ for all $i = 0, 1, ..., r-1$. The θ-cylindricity of the circuits D_i implies that for each fixed i, with the condition $g(i) = h_T(a, i)$ and from the point of view of $\mathcal{Y}_i^+$ the distribution of a_i is uniform on the set S_i of all extensions of f_i onto T_i. Since the various circuits D_i are pairwise disjoint their probabilistic inputs are independent. Therefore $\mathcal{Y}^+$ and $\mathcal{Y}_i^+$ have the same relevant information about the distribution of a_i, and so the distribution of a_i is uniform on S_i from the point of view of $\mathcal{Y}^+$ and with the condition $g(j) = \mathbf{h}_T(a, j)$ for all $j = 0, 1, ..., r-1$. The independence of the probabilistic inputs of the various circuits D_i, $i = 0, 1, ..., r-1$ also implies that $a_0, ..., a_{r-1}$ are mutually independent from the point of view of $\mathcal{Y}$ and with the condition $g(j) = h_T(a, j)$ for all $j = 0, 1, ..., r-1$. This completes the proof of condition (c).

Condition (12) for C_0. Assume that H_0, H_1 are similar and pure θ-cylindrical distributions with respect to the input block sequence of C and F, and the input of C_0 is arriving with either distributions H_0 or distribution H_1. Condition (12) for the circuit D_i implies that the distribution Z_i, of the knowledge of the adversary $\mathcal{Y}_i$ of D_i is the same for H_0 and H_1. Since the various circuits D_i are pairwise disjoint their probabilistic inputs are independent so the joint distribution of Z_i, $i = 0, 1, ..., r-1$ is also the same for H_0 and H_1. Since $Z = \{Z_0, ..., Z_{r-1}\}$ is the knowledge of $\mathcal{Y}$, we get that the distribution of Z is the same for H_0 and H_1. This completes the proof of the fact the C_0 is $(\theta, \varepsilon, |\mathcal{C}_0|e^{-cm})$-cylindrical .

Assume now that $0 < i \leq t$, and C_{i-1} is a $(\theta, \varepsilon, |\mathcal{C}_{i-1}|e^{-cm})$-cylindrical distribution. We prove that C_i is a $(\theta, \varepsilon, |\mathcal{C}_i|e^{-cm})$-cylindrical circuit.

We consider the case when $\texttt{set}(\mathcal{C}_i)\backslash\texttt{set}(\mathcal{C}_{i-1})$ is a singleton containing a + gate γ which adds the values calculated at nodes α and β. The other possibilities for $\texttt{set}(\mathcal{C}_i)\backslash\texttt{set}(\mathcal{C}_{i-1})$ can be handled in a similar way. This assumption implies that we get the circuit C_i by "gluing" together the circuits C_{i-1} and a copy D of the circuit $C_7^{(m)}$ of Lemma 3. More precisely assume that the input blocks of D are I_0, I_1 and the output blocks of C_{i-1} corresponding to the nodes

α, β of $\mathcal{C}_{i-1}$ are T_0, T_1. We get C_i by identifying I_0 with T_0 and I_1 with T_1.

We proceed in a similar way as in the $i = 0$ case. To define a (θ, ε, p)-cylindrical adversary $\mathcal{Y}_i$ for C_i with $p = |C_i|e^{-cm}$ we have to define a selection function $\mathcal{S}_i$. Assume that $\mathcal{Y}_{i-1}$ is a $(\theta, \varepsilon, |\mathcal{C}_{i-1}|e^{-cm})$-cylindrical adversary for C_{i-1} with selection function $\mathcal{S}_{i-1}$ whose existence is guaranteed by the inductive assumption, and $\mathcal{Y}$ is a $(\theta, \varepsilon, e^{-cm})$-cylindrical adversary for D with selection function $\mathcal{S}$ whose existence is stated in Lemma 3. Let X be an arbitrary subset of $\mathtt{set}(C_i)$. Then we define $\mathcal{S}_i(X)$ by $\mathcal{S}_i(X) = \mathcal{S}_{i-1}(X \cap \mathtt{set}(C_{i-1})) \cup \mathcal{S}(X \cap \mathtt{set}(D))$.

Let X be an ε-random subset of $\mathtt{set}(C_i)$. With a probability of at least $1 - |\mathcal{C}_{i-1}|e^{-cm} - e^{-cm}$ the set $X \cap \mathtt{set}(C_{i-1})$ is acceptable with respect to the adversary $\mathcal{Y}_{i-1}$ of C_{i-1}, and the set $X \cap (\mathtt{set}(D))$ is acceptable with respect to the adversary $\mathcal{Y}$ of D. We claim that if both sets are acceptable, then the set X is acceptable with respect to the adversary $\mathcal{Y}_i$ of C_i. To prove this we have to show that the set X satisfies conditions (11) and (12).

Condition (11). Assume that the deterministic input of C_i is fixed. According to the inductive assumption condition (11) holds for the circuit C_{i-1} and therefore the distribution of the output $a = \langle a_0, ..., a_{s-1} \rangle$ of C_{i-1} on its output blocks $T_0, ..., T_{s-1}$ is θ-cylindrical from the point of view of $\mathcal{Y}_{i-1}$. Let $\bar{B} = \bigcup_{i=0}^{s-1} B_i$ be the base set and $\bar{f} = f_0 \cup ... \cup f_{s-1}$ be the handle of the distribution of a, where $B_i \subseteq T_i$ and $\mathtt{domain}(f_i) = B_i$. Condition (11) implies that the base set $\bar{B}$ is uniquely determined by the set X and $\bar{f}$ is uniquely determined by the knowledge of $\mathcal{Y}$.

Let T_s be the output block of D. Condition (11) for the the circuit D with the set $X := X \cap \mathtt{set}(D)$ implies that the distribution of the output on T_s from the point of view of $\mathcal{Y}^+$ is θ-cylindrical. Naturally this is a different distribution for each possible input of D which are given on T_0 and T_1. However the base set B is uniquely determined by X and the handle is uniquely determined by the knowledge of $\mathcal{Y}$, which includes the set X. Therefore Lemma 2 implies that from the point of view of $\mathcal{Y}_i$ the distribution of the output on T_s is θ-cylindrical.

We want to show that the distribution of the output of C_i on the outputblock sequence $T_2, ..., T_{s-1}, W_0$ is θ-cylindrical from the point of view of $\mathcal{Y}_i$, with base set $B \cup \bigcup_{i=2}^{s-1} B_i$ and with handle $f_2 \cup ... \cup f_{s-1} \cup f$. Conditions (a), (b) of the definition of θ-cylindricity immediately follow from the corresponding conditions for $\mathcal{Y}$.

Condition (c). We will prove condition (c) of the definition of a θ-cylindrical random variable in the following equivalent form.

(c') Assume that $g \in \mathtt{func}(\{0, 1, ..., s-1\}, F)$ such that $\mathtt{prob}(\mathcal{A}_g) > 0$ where $\mathcal{A}_g$ is the event $\forall i \in [0, s-1], \mathbf{h}_W(\xi, i) = g(i)$. Then for all $j = 0, 1, ..., s-1$, and for all $b_0, ..., b_{j-1} \in F$ if $\mathtt{prob}(\forall r \in [0, j-1], \xi_r = b_r \mid \mathcal{A}_g) > 0$, then with the condition $\mathcal{A}_g \wedge \forall r \in [0, j-1], \xi_r = b_r$ the random variables ξ_j has uniform distribution on $\mathtt{extension}_F(f_j, W_j, g(j))$, where $f_j = f|_{W_j \cap Y}$.

The equivalence of (c) and (c') is an immediate consequence of the definition of the independence of random variables.

Assume that the knowledge of adversary $\mathcal{Y}^+$, that is, the deterministic input of C_i and the evaluation values of C_i on the set X are fixed and we randomize the probabilistic inputs of C with these conditions. With respect to this randomization $\langle a_0, a_1, a_2, ..., a_{s-1} \rangle$, the output sequence of circuit C_{i-1} and a_s the output of D, are random variables. Therefore the output of C_i, is also a random variable. We will denote this random variable by $\xi = \langle \xi_0, ..., \xi_{s-2} \rangle$, where $\xi_0 = a_2, ..., \xi_{s-3} = a_{s-1}, \xi_{s-2} = a_s$. Let $g \in \mathtt{func}(\{0, 1, ..., s-2\}, F)$ and let $\mathcal{A}_g$ be the event $\forall i \in [0, s-2] \mathbf{h}_W(\xi, i) = g(i)$, where $W = \langle T_2, ..., T_{s-2}, T_s \rangle$. We have to show that condition (c') is satisfied with these choices of the parameters. For $j < s-2$ the statement of (c') follows from the fact that the circuit C_{i-1} is θ-cylindrical and so satisfies condition (c'). Assume now that $j = s-2$. We have to show that with condition $\mathcal{A}_g \wedge \forall i \in [2, s-1], a_i = b_{i-2}$ the random variable a_s has uniform distribution on the set $\texttt{extension}_F(f, B, g(s-2))$. We compute this distribution using Bayes' theorem. For each possible fixed value of a_0 and a_1 the choice of the set X implies that the output of D has θ-cylindrical distribution. Therefore condition $\mathcal{A}_g$ implies that for each fixed value of a_0 and a_1, the output of D, that is, a_s has uniform distribution on $\texttt{extension}(f, W_0, g(s-2))$. According to Bayes' theorem the distribution of a_s must be uniform on the same set if we do not fix a_0 and a_1. This completes the proof of condition (11).

Condition (12). It is sufficient to show that if H is a pure θ-cylindrical distribution with base set B_H and handle f_H, then we are able to generate the knowledge of the adversary $\mathcal{Y}$, as a random variable, with knowing only the pair $P = \langle B_H, f_H \rangle$. Since $X \cap \texttt{set}(C_{i-1})$ is acceptable with respect to $\mathcal{Y}_{i-1}$ we have that the distribution of the knowledge of $\mathcal{Y}_{i-1}$ depends only on P, that is, it can be generated as a random variable depending only on P. Assume now that Z_{i-1}, the knowledge of $\mathcal{Y}_{i-1}$, is fixed. The fact that $X \cap \texttt{set}(C_{i-1})$ is acceptable with respect to $\mathcal{Y}_{i-1}$ and Proposition 5 implies that if the input is arriving with distribution H, the output from the point of view of $\mathcal{Y}_{i-1}$ has θ-cylindrical distribution with base set $B_0 \cup B_1$ and handle $f_0 \cup f_1$. This implies however, using now the fact that $X \cap \texttt{set}(D)$ is acceptable with respect to $\mathcal{Y}$, that the distribution of the knowledge Z of $\mathcal{Y}$ depends only on B and F therefore we can generate Z, knowing only Z_{i-1}. Since Z and Z_{i-1} together uniquely determine Z_i the knowledge of $\mathcal{Y}_i$, we got that the distribution of Z_i depends only on B_H and f_H. This completes the proof of condition (12) and the proof of the $i > 0$ case in the inductive assumption.

If the boolean circuit $\mathcal{C}$ contains probabilistic inputs as well, then we treat them the same way as if they were deterministic inputs. That is, in the circuit C a new input block J_x of size m will correspond to each probabilistic input node x of $\mathcal{C}$. The input nodes of this block are probabilistic inputs of C, and their sum will correspond to the random input on the node x. In the construction of C we treat the blocks of type J_x the same way as if they were deterministic input blocks. *Q.E.D.*(Lemma 4)

4.4 Circuits and related linear systems over F

Definition. Suppose that $G = \{G_\gamma = 0 \mid \gamma \in \Gamma\}$ is a finite set of equations, where for each $\gamma \in \Gamma$, G_γ is a polynomial over the field F. Then we will say that G_γ is a polynomial system of equations over the field F, or shortly, G is an F-system. Suppose that the unknowns of the system G are x_h, $h \in H$. An evaluation ϑ of the variables x_h with values in F is a solution of the system G, if for each $\gamma \in \Gamma$, $G_\gamma(\vartheta(x_1), \ldots, \vartheta(x_i)) = 0$, where $x_1, \ldots, x_i$ are all of the unknowns occurring in the polynomial G_γ. In this case we will also say that the evaluation ϑ, satisfies the system G. $\Box$

Definition. If C is a probabilistic block circuit, then $\texttt{block}_{\texttt{in}}(C)$ will denote the set of its input blocks and $\texttt{block}_{\texttt{out}}(C)$ will denote the set of its output blocks.

2. Suppose that C is a probabilistic block F-circuit of type $((m, n), k, l)$, and $\delta \in \texttt{func}(\texttt{block}_{\texttt{in}}(C), F)$, and $\texttt{block}_{\texttt{in}}(C) = \{I_0, \ldots, I_{k-1}\}$. The system of equations $\{P_i = \delta(I_i) \mid i = 0, 1, \ldots, k-1\}$ will be denoted by $\mathcal{I}(\delta)$, where P_i, $i = 0, 1, \ldots, k-1$ are the input block polynomials of C. Assume now that $A \subseteq \texttt{set}(C)$, and $g \in \texttt{func}(A, F)$. The system of equations

$\{p_a = g(a) \mid a \in A\}$ will be denoted by $\mathcal{E}(g)$. $\Box$

Definition. Assume that Γ is a finite set and for each $\gamma \in \Gamma$, $G_\gamma(x_0, \ldots, x_{k-1}, y_0, \ldots, y_{l-1})$ is a polynomial over the field F. We say that the probabilistic block F-circuit C of type $((m,n),k,l)$ satisfies the equation $G(x_0, \ldots, x_{k-1}, y_0, \ldots, y_{l-1}) = 0$, if for all $\gamma \in \Gamma$, $G_\gamma(P_0, \ldots, P_{k-1}, Q_0, \ldots, Q_{l-1}) \equiv 0$, where $P_0, \ldots, P_{k-1}, Q_0, \ldots, Q_{l-1}$ are the block polynomials of C. $\Box$

For an F-circuit C, the polynomials p_b, $b \in C$, were created by putting indeterminates $z_h = p_h$ on the input nodes h of C, and then performing the field operations on the polynomials as we go down the nodes of the circuits from the inputs to the outputs. The polynomial that we get on node b is p_b. Assume now that a function $g \in \mathtt{func}(A, F)$ is given where $A \subseteq \mathtt{set}(C)$. We modify the definition of p_b in a way that if in the recursive process we would have to use the polynomial p_a for some $a \in A$ then we will use instead of p_a the constant polynomial $g(a)$. For each $b \in \mathtt{set}(C)$ we will define this way a polynomial $p_b^{(g)}$. These polynomials will be used to give an equivalent form of the system $\mathcal{E}(g)$ where the function g has a more explicit role.

Definition. Assume that C is a probabilistic block F-circuit, and $H = \mathtt{inset}(C)$. Suppose further that $A \subseteq \mathtt{set}(C)$, $g \in \mathtt{func}(A, F)$. If $h \in H$ then we associate with H an indeterminate z_h. For each node b of C we define a polynomial $p_b^{(g)} \in F[z_h \mid h \in H]$. (For $A = \emptyset$ this will be the polynomial p_b defined earlier). We define $p_b^{(g)}$ by recursion on the depth of b in a similar way as we have defined the polynomial p_b. For each input node b we have $p_b^{(g)} = z_b$. Assume that b is not an input node.

Intuitively the definition of $p_b^{(g)}$ is the following. We execute the field operation associated with gate b on polynomials defined in the following way. If an input of gate b in the circuit C is coming from an element $a \in A$ then the corresponding polynomial will be the constant polynomial $g(a)$. If the input is coming from an element $u \notin A$ then the corresponding polynomial will be $p_u^{(g)}$ which has been already defined. We execute the field operation of b on the polynomials defined this way and the resulting polynomial is $p_b^{(g)}$.

More precisely $p_b^{(g)}$ is defined in the following way. If the indegree of b is one, and b is labelled with "+1", and there is an edge pointing from u to a for some $u \in \mathtt{set}(C)$. Then we distinguish two cases. If $u \in A$ then $p_b^{(g)} = g(u) + 1$. If $u \notin A$, then $p_b^{(g)} = p_u^{(g)} + 1$. In a similar way if the label is $\times(-1)$ then, in the case $u \in A$ we have $p_b^{(g)} = -g(u)$, and for $u \notin A$ our definition is $p_b^{(g)} = -p_u^{(g)}$.

Suppose now that the indegree of a is 2 and u_0, u_1 are distinct nodes so that there is an edge pointing from u_i to a for $i = 0, 1$. First we define a function G on the set $\{u_0, u_1\}$ by $G(u_i) = g(u_i)$ if $u_i \in A$, and $G(u_i) = p_{u_i}^{(g)}$ otherwise, for $i = 0, 1$. Then, according to whether the label of b is $+$, or $\times$, we have $p_b^{(g)} = G(u_0) + G(u_1)$, or $p_b^{(g)} = G(u_0) \times G(u_1)$. $\Box$

Lemma 5 *Suppose that C is a probabilistic block F-circuit of type $((m,n),k,l)$, $A \subseteq \mathtt{set}(C)$, and $g \in \mathtt{func}(A, F)$. The system of equations $\{p_a^{(g)} = g(a) \mid a \in A\}$ is equivalent to the system $\mathcal{E}(g)$ in the sense that their sets of solutions are identical.*

Proof. The statement of the lemma is an immediate consequence of the recursive definitions of the polynomials p_a, $p_a^{(g)}$. *Q.E.D.*(Lemma 5)

Definition. Assume that C is a probabilistic block F-circuit of type $((m,n),k,l)$, and $A \subseteq \mathtt{set}(C)$, and the following holds:

(33) *for all $\delta, \delta' \in \mathtt{func}(\mathtt{block}_{\mathtt{in}}(C), F)$, and for all $g \in \mathtt{func}(A,F)$, the number of solutions of the system $\mathcal{I}(\delta) \cup \mathcal{E}(g)$ is identical to the number of solution of the system $\mathcal{I}(\delta') \cup \mathcal{E}(g)$.*

Then we will say that the block input of the circuit C is invisible from the set A. $\square$

Lemma 6 *Assume that C is a probabilistic block F-circuit of type $((m,n),k,l)$, and $A \subseteq B \subseteq \mathtt{set}(C)$. If the block input of the circuit C is invisible form the set B then it is also invisible from the set A.*

Proof. We will denote the number of solutions of a system of equations E by $\mathcal{N}(E)$. Assume that the block input of the circuit C is invisible from the set B. Let $g \in \mathtt{func}(A,F)$ and $\delta^{(0)}, \delta^{(1)} \in \mathtt{func}(\mathtt{block}_{\mathtt{in}}(C), F)$. We have to prove that $\mathcal{N}(\mathcal{I}(\delta^{(0)}) \cup \mathcal{E}(g)) = \mathcal{N}(\mathcal{I}(\delta^{(1)}) \cup \mathcal{E}(g))$. Let H be the set of all extensions h of g onto B with values in F. By 5 we have that for all $j \in \{0,1\}$, $\mathcal{N}(\mathcal{I}(\delta^{(j)}) \cup \mathcal{E}(g)) = \sum_{h \in H} \mathcal{N}(\mathcal{I}(\delta^{(j)}) \cup \mathcal{E}(h))$. The assumption that the block input of C is invisible from the set B implies that for each fixed $h \in H$ we have $\mathcal{N}(\mathcal{I}(\delta^{(0)}) \cup \mathcal{E}(h)) = \mathcal{N}(\mathcal{I}(\delta^{(1)}) \cup \mathcal{E}(h))$ which implies the required equality. *Q.E.D.*(Lemma 6)

Lemma 7 *Assume that $\theta > 0, \varepsilon > 0, p \in [0,1]$ and $\mathcal{Y} = \mathtt{adv}_C(\varepsilon, \mathcal{S})$ is an ε-random adversary with selection function $\mathcal{S}$ for a probabilistic block F-circuit C with input block sequence $I = \langle I_0, ..., I_{s-1} \rangle$, and $f \in \mathtt{func}_\theta(I,F)$. Then, for each fixed set $X \subseteq \mathtt{set}(C)$ the following condition implies condition* (12) from the definition of a (θ, ε, p)-cylindrical adversary:

(34) *the block input of the circuit C is invisible from the set* $\mathtt{domain}(f) \cup \mathtt{full}(\mathcal{S}(X))$

Remark. Condition (34) implies that condition (12) holds not only for the function f but also holds with f:=g, where g is an arbitrary F valued function with $\mathtt{domain}(g) = \mathtt{domain}(f)$. $\square$

Proof of Lemma 7. Assume that H_0, H_1 are similar and pure θ-cylindrical distributions for the deterministic input of C and f is their common handle. Let $Y = \mathtt{base}(H_0) = \mathtt{base}(H_1) = \mathtt{domain}(f)$. Our assumption is that the block input of of C is invisible from the set $Y \cup \mathtt{full}(\mathcal{S}(X))$. The definition of a pure θ-cylindrical distribution implies that the probability that the knowledge of $\mathcal{Y}$, that is, $\langle V, Y, f \rangle$, takes a certain value can be expressed from the number of solutions of a system of polynomial equations over F. This is true since *Q.E.D.*(Lemma 7)

We will frequently use the following well-known fact.

Proposition 6 *Assume that two inhomogeneous system of linear equations over a finite field are identical with the exception of their inhomogeneous parts. If both of them has at least one solution then their numbers of solutions are the same.*

Proof. The number of solutions of both systems is identical to the number of solutions of the corresponding homogeneous system. *Q.E.D.*(Proposition 6)

Definition. A linear term of a multivariate polynomial is a term of the polynomial which contains only a single indeterminate, and the degree of this indeterminate is 1. $\square$

Proposition 7 *Assume that E is a finite set, and for all $\alpha \in E$, q_α is a polynomial over the finite field F, so that its constant term is 0, and U is the set of indeterminates in the polynomials q_α, $\alpha \in E$. Assume further that there exist sets $E_0, E_1 \subseteq E$, $U_0, U_1 \subseteq U$ so that the following conditions are satisfied:*

(35) *$\{E_0, E_1\}$ is a partition of E, and $\{U_0, U_1\}$ is a partition of U,*

(36) *all of the polynomials in the set $\tilde{E}_0 = \{q_\alpha \mid \alpha \in E_0\}$ are linear polynomials and may contain indeterminates only from the set U_0*

(37) *the indeterminates of U_1 may occur only in the polynomials of the set $\tilde{E}_1 = \{q_\alpha \mid \alpha \in E_1\}$ and only in linear terms,*

(38) *for each $\varphi \in \texttt{func}(E, F)$, if the system of equations $q_\alpha = \varphi(\alpha)$, $\alpha \in E$ has at least one solution in F, then every solutions of the system $q_\alpha = \varphi(\alpha)$, $\alpha \in E_0$ can be extended into a solution of $q_\alpha = \varphi(\alpha)$, $\alpha \in E$ in F.*

Then, for all $\varphi, \psi \in \texttt{func}(E, F)$ if both systems $q_\alpha = \varphi(\alpha)$, $\alpha \in E$ and $q_\alpha = \psi(\alpha)$, $\alpha \in E$ have at least one solutions in F, then they have the same number of solutions in F.

Proof. Assume that the systems $q_\alpha = \varphi(\alpha)$, $\alpha \in E_0$ and $q_\alpha = \psi(\alpha)$, $\alpha \in E_0$ both have a solution in F. Since they are linear, Proposition 6 implies, that they have the same number of solutions in F. Therefore it is sufficient to show that if ϑ is an arbitrary evaluation of the unknowns in U_0 which is a solution of $q_\alpha = \varphi(\alpha)$, $\alpha \in E_0$, then the number of extensions of ϑ into a solution of the system $q_\alpha = \varphi(\alpha)$, $\alpha \in E_1$ does not depend on ϑ. This is however a consequence of the fact that, for a fixed ϑ, the system E_1 is linear in the variables in U_1. According to condition (38) this system always have a solution, therefore the linearity and Proposition 6 implies that the number of its solutions does not depend on ϑ. *Q.E.D.*(Proposition 7)

Definition. 1. Assume that F is a field, H is a finite set, for each $h \in H$, z_h is an indeterminate, $G \subseteq H$ and f is an element of the polynomial ring $F[z_h \mid h \in H]$. $\texttt{lin}_G(f)$ will denote the polynomial in $F[z_h \mid h \in H]$ consisting of those terms of f which are of the form γz_g for suitably chosen $\gamma \in F$ and $g \in G$. If $Y \subseteq F[z_h \mid h \in H]$ then $\overline{\texttt{lin}}_G(Y)$ will denote the vectorspace generated by the set $\{\texttt{lin}_G(f) \mid f \in Y\}$

2. Assume that $B \subseteq F[z_h \mid h \in H]$ and $G \subseteq H$. The G-dimension of the set B will be the dimension of $\overline{\texttt{lin}}_G(B)$ over the field F. This number will be denoted by $\dim_G(B)$. $\square$

Definition. 1. Assume that $\sigma \in \texttt{func}(\texttt{inset}(C), F)$, and

$$G = \{G_\gamma(x_0, \ldots, x_{k-1}, y_0, \ldots, y_{l-1}) = 0 \mid \gamma \in \Gamma\}$$

is a polynomial system of equations over F. We say that σ satisfies the system G, if after substituting $\sigma(h)$ for z_h for each $h \in \texttt{inset}(C)$, the value of all of the polynomials $G_\gamma(P_0, \ldots, P_{k-1}, Q_0, \ldots, Q_{l-1})$, $\gamma \in \Gamma$ is 0.

2. Assume that f, g are functions so that $\texttt{domain}(g) \subseteq \texttt{domain}(f)$. $f \natural g$ will denote a function defined on $\texttt{domain}(f)$, so that $f \natural g$ is identical to g on $\texttt{domain}(g)$ and identical to f on $\texttt{domain}(f) \backslash \texttt{domain}(g)$.

3. Suppose that q is a polynomial over the field F containing the indeterminates z_h, $h \in H$, and $f \in \texttt{func}(H, F)$. Then $q|_{z_h = f(h)}$ is the value that we get, if in the polynomial q we substitute $f(h)$ for the indeterminate z_h, for all of $h \in H$. $\square$

4.5 Percolative circuits

Definition. 1. Recall that the set of input/output blocks of a block F-circuit C is denoted by $\mathtt{block_{in}}(C)/\mathtt{block_{out}}(C)$. We will use the notation $\mathtt{block_{io}}(C) = \mathtt{block_{in}}(C) \cup \mathtt{block_{out}}(C)$ for the set of all input and output blocks together.

2. Assume that $\sigma \in \mathtt{func}(\mathtt{inset}(C), F)$. $\sigma^{(\mathtt{out})}$ will denote the function defined on $\mathtt{outset}(C)$ which describes the output of the circuit C at input σ. (Equivalently we may define $\sigma^{(\mathtt{out})}$ for each $a \in \mathtt{outset}(C)$, by $\sigma^{(\mathtt{out})}(a) = \chi^{(C)}_{\sigma}(a) = p_a|_{z_h=\sigma(h)}$.) The function defined on $\mathtt{detin}(C) \cup \mathtt{outset}(C)$ so that it is identical to σ on $\mathtt{detin}(C)$, and identical to $\sigma^{(\mathtt{out})}$ on $\mathtt{outset}(C)$ will be denoted by $\sigma^{(\mathtt{io})}$. $\bar{\sigma}^{(\mathtt{in})}$ will denote the function so that $\bar{\sigma}^{(\mathtt{in})}(J)$ is defined iff $J \in \mathtt{block_{in}}(C)$, and $\bar{\sigma}^{(\mathtt{in})}(J) = \sum_{b \in J} \sigma(b)$. $\bar{\sigma}^{(\mathtt{out})}$ will denote the function so that $\bar{\sigma}^{(\mathtt{out})}(J)$ is defined iff $J \in \mathtt{block_{out}}(C)$, and $\bar{\sigma}^{(\mathtt{out})}(J) = \sum_{b \in J} \sigma^{(\mathtt{out})}(b)$. $\bar{\sigma}^{(\mathtt{io})}$ is the common extension of the functions $\bar{\sigma}^{(\mathtt{in})}$, and $\bar{\sigma}^{(\mathtt{out})}$ to the set $\mathtt{block_{io}}(C)$.

3. Assume that ϑ is a function defined on the set $\mathtt{block_{io}}(C)$, $\mathtt{block_{in}}(C) = \{I_0, \ldots, I_{k-1}\}$, $\mathtt{block_{out}}(C) = \{T_0, \ldots, T_{l-1}\}$, and $G(x_0, \ldots, x_{k-1}, y_0, \ldots, y_{l-1})$ is a polynomial over F. We will say that ϑ satisfies the polynomial system of equations $G = \{G_\gamma = 0 \mid \gamma \in \Gamma\}$, with respect to the indicated orderings of the input and output blocks, if for all $\gamma \in \Gamma$ we have $G_\gamma(\vartheta(I_0), \ldots, \vartheta(I_{k-1}), \vartheta(T_0), \ldots, \vartheta(T_{l-1})) = 0$.

Suppose now that τ is a function defined on a set containing $\mathtt{detin}(C) \cup \mathtt{outset}(C)$. We define a function ϑ_τ on $\mathtt{block_{io}}(C)$ by $\vartheta_\tau(J) = \sum_{x \in J} \tau(x)$, for all $J \in \mathtt{block_{io}}(C)$. We will say that τ satisfies the system G, with respect to the indicated orderings of the input and output blocks, if this is true for the function ϑ_τ. In both cases we will always assume, that orderings of the input and output blocks are fixed for each circuit and we are considering only these fixed orderings. Therefore the expression "with respect to the indicated orderings" will be omitted. $\square$

Motivation. Assume that C is a block F-circuit and an adversary knows the values of the evaluation function $\chi^{(C)}(x)$ if $x \in A$, where $A \subseteq \mathtt{set}(C)$. Suppose further that the circuit satisfies a system of polynomial equations Γ. The following definition expresses a property of the set A, which essentially say that this knowledge of the adversary does not mean too much. Namely this property, called A, Γ-percolativity will say that there exists a large subset X of the input nodes and output nodes together, such that if on X we change the evaluation function in a way which is compatible to Γ, but otherwise arbitrary, then what we get is still compatible with the knowledge of the adversary in the sense that with the new changed input values the probability of the new output values is not 0 from the point of view of the adversary. We will show later that the adversary cannot distinguish the original inputs δ_0 and δ_1 of certain circuits, by showing that their probabilities, from the point of view of the adversary who knows the evaluation values on the set A, can be expressed by the number of solutions of systems E_0, E_1 of equations over F. Using the results about polynomial systems proved in the previous section, we only have to show that if one of the systems has a solution then the other system also has a solution. The notion of A, Γ-percolativity will be used for this purpose, it will help us to construct a solution of E_1 from a solution of E_0. The unknowns of these systems will be the inputs of the circuits both deterministic and probabilistic, corresponding to the two original inputs δ_0, δ_1.

Definition. Assume that m, n, k, l are positive integers, C is a probabilistic block F-circuit of type $((m,n),k,l)$, which satisfies the system of polynomial equations Γ over F. Suppose further that $A \subseteq \mathtt{set}(C)$, $I_0, \ldots, I_{k-1}$ are the input blocks and $T_0, \ldots, T_{l-1}$ are the output blocks of C. We will say that the circuit C is percolative with respect to A and Γ, or A, Γ-percolative, if there exists a set $X \subseteq \mathtt{detin}(C) \cup \mathtt{outset}(C)$, so that the following two conditions are satisfied.

(39) *For each* $i = 0, 1, \ldots, k-1$, $|X \cap I_i| > \frac{1}{2}|I_i|$, *and for all* $j = 0, 1, \ldots, l-1$, $|X \cap T_j| > \frac{1}{2}|T_j|$.

(40) *For all* $\sigma \in \mathtt{func}(\mathtt{inset}(C), F)$, *and for all* $\kappa \in \mathtt{func}(X, F)$, *if* $\sigma^{(\mathtt{io})} \natural \kappa$ *satisfies the system* Γ, *then there exists a* $\rho \in \mathtt{func}(\mathtt{inset}(C), F)$ *so that* $\rho^{(\mathtt{io})} = \sigma^{(\mathtt{io})} \natural \kappa$, *and for all* $a \in \mathtt{full}(A)$, *we have* $\chi_\sigma^{(C)}(a) = \chi_\rho^{(C)}(a)$.

$\square$

Remark. The assumption that $\sigma^{(\mathtt{io})} \natural \kappa$ satisfies Γ does not follow from the assumption that C satisfies Γ, since the values of $\sigma^{(\mathtt{io})} \natural \kappa$ on the input nodes and output nodes of C are not necessarily identical to the values of any evaluation function of C on these nodes. $\square$

4.6 The copying circuit

We will assume that the reals ε, c, θ are chosen so that $0 < \varepsilon \ll c \ll \theta \ll 1$, m is a sufficiently large integer and F is a finite field. (Recall that $a \ll b$ stands for a is sufficiently small with respect to b.) We define a probabilistic block F-circuit $\mathbf{C}_m^{(=)}$, of type $((m,m),1,1)$, which computes the polynomial x, so that the circuit is (θ, ε, p)-cylindrical, where $p = e^{-cm}$. This circuit will be called the copying circuit since its block input, the sum of the deterministic input values, is the same as its block output, the sum of the output values. We can say that it simply copies the block input. However the deterministic input as a sequence of length m, with high probability, will be different from the output sequence. Of course, the computation of the function x is trivial, if there are no other requirements, the output could be the same as the input. The (θ, ε, p)-cylindricity however, excludes this and other very simple solutions.

The circuit $\mathbf{C}_m^{(=)}$ will be needed in the construction of other (θ, ε, p)-cylindrical circuits of type $((m,m),k,l)$ whose existence is stated in Lemma 3.

Definition. 1. Assume that $G = \langle V, E \rangle$ is a graph and $A \subseteq V$. $N(A)$ will denote the set of all points $x \in V$ with the property that there exists an $a \in A$, such that either $a = x$ or x is a neighbor of a.

2. Assume that, d, m are positive integers, $\alpha > 0$ is a real, and $G = \langle V, E \rangle$ is a d-regular expander graph on the set of vertices $V = \{0, 1, \ldots, m-1\}$ with expansion factor $1 + \alpha$, that is, for each $A \subseteq V$ with $|A| \le \frac{m}{2}$ we have $N(A) \ge (1+\alpha)|A|$. We will call such an expander graph a $(d, 1+\alpha)$-expander on m vertices. If we say that a graph is a $(d, 1+\alpha)$-expander on m vertices, we will also assume that the vertex set is the set $\{0, 1, \ldots, m-1\}$ unless we explicitly state it otherwise.

The copying circuit that we define below will depend on the graph G so we will write $\mathbf{C}_{m,G}^{(=)}$ for $\mathbf{C}_m^{(=)}$ if we want to emphasize this dependence. $\mathbf{C}_{m,G}^{(=)}$ has m deterministic input nodes and $\frac{1}{2}dm$ probabilistic input nodes. We associate each probabilistic input node with an edge of G, so we will consider the probabilistic input as a function $\gamma \in \texttt{func}(E,F)$. $\mathbf{C}^{(=)}_{m,G}$ will be a type $((m,m),1,1)$ circuit, therefore it will have a single input block and a single output block. The deterministic input nodes will be denoted by $\wp_0,\ldots,\wp_{m-1}$ and the output nodes by $\mathcal{T}_0,\ldots,\mathcal{T}_{m-1}$. With each edge $e \in E$ of the graph G we associate a probabilistic input node $\mathcal{G}_e$ of $\mathbf{C}_m$. There are no other probabilistic input nodes. If we say that $\delta = \langle \delta_0,\ldots,\delta_{m-1}\rangle \in F^m$ is the deterministic input, the function $\gamma \in \texttt{func}(E,F)$ is the probabilistic input, and $\nu = \langle \nu_0,\nu_1,\ldots,\nu_{m-1}\rangle \in F^m$ is the output, then we mean that δ_i is given at the node $\wp_i$, for $i = 0,1,\ldots,m-1$, $\gamma(e)$ is given at the node $\mathcal{G}_e$, for all $e \in E$, and ν_i appears at the node $\mathcal{T}_i$ for $i = 0,1,\ldots,m-1$.

Assume that $\delta_0,\ldots,\delta_{m-1}$ is the deterministic input and $\gamma \in \texttt{func}(E,F)$ is the probabilistic input. Using the operations $+$ and multiplication by -1, both in F, we define the output $\nu_0,\nu_1,\ldots,\nu_{m-1}$ in the following way:

$$\nu_i = \delta_i + \sum\{\gamma((i,j)) \mid (i,j) \in E \wedge i < j\} + \sum\{-\gamma((i,j)) \mid (i,j) \in E \wedge i > j\}$$

that is, the output bit at $\mathcal{T}_i$ is the sum of the random values $\gamma((i,j))$ attached to the edges whose smaller incident vertex is i, minus the sum for the edges where i is the larger incident vertex.

We construct the circuit $C = \mathbf{C}^{(=)}_{m,G}$, using only $+$ and $(-1)\times$ gates, which computes the described value of ν, given δ and γ as deterministic and probabilistic inputs.

For each $e \in E$, $\mathbf{C}^{(=)}_{m,G}$ has a $(-1)\times$ gate w_e, which computes $-\gamma(e)$. For the the computation of the output bit ν_i, $\mathbf{C}^{(=)}_{m,G}$ uses d "+" gates $g_{0,i} >_C g_{1,i} >_C \ldots >_C g_{d-1,i} = \mathcal{T}_i$ which compute the sum in the definition of ν_i using δ_i, $\gamma(e)$ or $-\gamma(e)$ for the edges $e = (j,i) \in E$. This completes the definition of the circuit $\mathbf{C}^{(=)}_{m,G}$. $\square$

We will use later the following immediate consequence of the definition of $\mathbf{C}^{(=)}_{m,G}$.

Proposition 8 *Assume that $C = \mathbf{C}^{(=)}_{m,G}$ is a copying circuit, where $G = \langle V,E\rangle$ is a d-regular graph. Then $|\texttt{set}(\mathbf{C}^{(=)}_{m,G})| = m + 2md$, and for each $b \in \texttt{set}(C)$ the set $\{x \in \texttt{set}(C) \mid x \geq_C b\}$ has at most $3d+1$ elements and the set $\{x \in \texttt{set}(C) \mid x \geq_C b\} \cap \texttt{rand}(C)$ has at most d elements.*

Proof. There are m deterministic input nodes, $\frac{1}{2}md$ probabilistic input nodes $\mathcal{G}_e$ and for each of them a $(-1)\times$ gate w_e. All of the remaining gates, including the output nodes are the gates $g_{j,i}$, $j = 0,1,\ldots,d-1$, $i = 0,1,\ldots,m-1$. This gives $|\mathbf{C}^{(=)}_{m,G}| = m + \frac{1}{2}md + \frac{1}{2}md + md = m + 2md$.

For every $a \in \texttt{set}(C)$ there exists an output node $\mathcal{T}_i$, $i \in \{0,1,\ldots,m-1\}$ so that $a \geq_C \mathcal{T}_i$. Therefore it is sufficient to prove the lemma in the special case when $b = \mathcal{T}_i$ for some $i = 0,1,\ldots,m-1$. The construction of $\mathbf{C}^{(=)}_{m,G}$ implies that for all $x \geq_C \mathcal{T}_i$ there exists an $e = (i,j) \in E$ and a $k \in \{0,1,\ldots,d-1\}$ so that one of the following conditions are satisfied: (i) $x = \wp_i$, (ii) $x = g_{k,i}$, (iii) $x = \mathcal{G}_e$ (iv) $x = w_e$. Case (i) contributes one element, case (ii) contributes d elements (including $\mathcal{T}_i$), and case (iii) also contributes d elements, case (iv) contributes at most d elements. This implies that there exists at most $3d+1$ choices for the element $x \geq_C \mathcal{T}_i$. When we estimate the number of elements of $\{x \in \texttt{set}(C) \mid x \geq_C b\} \cap \texttt{rand}(C)$, then cases (i),(iii),(iv) must be excluded, so we get the upper bound d. *Q.E.D.*(Proposition 8)

Lemma 8 *For all positive integers m,d and for all d-regular graph G if $\mathbf{C}^{(=)}_{m,G}$ is a copying circuit then the following holds:*

(41) *if $\delta_0, \ldots, \delta_{m-1}$ is the deterministic input and $\nu_0, \ldots, \nu_{m-1}$ is the output, then $\sum_{i=0}^{m-1} \delta_i = \sum_{i=0}^{m-1} \nu_i$. Consequently $P_0(\delta_0, \ldots, \delta_{m-1}) = Q_0(\delta_0, \ldots, \delta_{m-1})$, where P_0 is the only input block polynomial, and Q_0 is the only output block polynomial of $\mathbf{C}_m^{(=)}$.*

Proof. Condition (41). The definition of $\mathbf{C}_{m,G}^{(=)}$ implies that for each fixed pair i, j with $i < j$, we have a $\gamma((i,j))$ term in the definition of ν_i and a $-\gamma((i,j))$ term in the definition of ν_j. $\gamma((i,j))$ does not occur in the definition of any ν_k with $k \notin \{i,j\}$. Therefore $\sum_{i=0}^{m-1} \nu_i = \sum_{i=0}^{m-1} \delta_i$. Using this identity the statement about the input block and output block polynomials are immediate consequences of their definitions. *Q.E.D.*(Lemma H0)

For the proof of the fact the copying graph is (θ, ε, p)-cylindrical for certain choices of the graph G $\mathbf{C}_{m,G}^{(=)}$ we will need the following Lemma about expander graphs.

Lemma 9 *Assume that n, d are positive integers and α is a positive realnumber. Suppose further that $G = \langle V, E\rangle$ is a d-regular graph with n vertices and for all $X \subseteq V$ with at most $\frac{n}{2}$ vertices we have that $|N(X)| \geq (1+\alpha)|X|$. Assume further that $B \subseteq V$ with $n - (1+\frac{d}{\alpha})|B| > 0$. Then $V \backslash B$ has a connected component containing more than $\frac{n}{2}$ points. Moreover, if C is the unique connected component of $V \backslash B$ with more than $\frac{n}{2}$ point, then $|C| \geq n - (1+\frac{d}{\alpha})|B|$.*

Proof. Let $C_1, \ldots, C_k$ be all of the connected components of $V \backslash B$ which contain at most $\frac{n}{2}$ elements. Since for each $i = 1, \ldots, k$, C_i is a maximal connected set in $V \backslash B$ we have $N(C_i) \backslash C_i \subseteq B$. According to the assumptions about G in the lemma, $|C_i| \leq \frac{n}{2}$ implies that $|N(C_i) \backslash C_i| \geq \alpha |C_i|$. Since the graph is d-regular, each point of B is in the set $N(C_i)$ for at most d different values of i. Consequently $\sum_{i=1}^{k} \alpha |C_i| \leq \sum_{k=1}^{n} |N(C_i) \backslash C_i| \leq d|B|$ and therefore $\sum_{i=1}^{k} |C_i| \leq \frac{d}{\alpha}|B|$. We get that if $C = V \backslash (B \cup C_1 \cup \ldots \cup C_k)$ then $|C| \geq n - (1+\frac{d}{\alpha})|B|$. According to the assumptions of the lemma $n - (1+\frac{d}{\alpha})|B| > 0$, and so $|C| \geq n - (1+\frac{d}{\alpha})|B| > 0$. Consequently the definitions of $C_1, \ldots, C_k$ imply that C is the unique connected component of $V \backslash B$ with more than $\frac{n}{2}$ points and with the claimed lower bound on its size. *Q.E.D.*(Lemma 9)

Lemma 10 *Assume that n, d are positive integers $\alpha > 0$, the graph G is a $(d, 1+\alpha)$ expander on the set of vertices $V = \{0, 1, ..., m-1\}$, and $C = \mathbf{C}_{m,G}^{(=)}$ is the copying block F-circuit defined from the graph G. Suppose further that $A \subseteq$ `set`(C), and $m > d(1+\frac{d}{\alpha})|A|$. Then there exists a set $D \subseteq \{0, 1, ..., m-1\}$, so the if P_D is the set of all probabilistic input nodes $\mathcal{G}_{i,j}$ of C with $i, j \in D$, then the following conditions are satisfied:*

(42) *for all $x \in A$ and $y \in P_D$, we have $x \not\leq_C y$*

(43) $|D| \geq \max\{\frac{m}{2}, m - d(1+\frac{d}{\alpha})|A|\}$

(44) *Assume that we fix all the deterministic inputs of C in an arbitrary way and also fix all of the probabilistic inputs of C as well in an arbitrary way, with the exceptions of the ones belonging to the probabilistic input nodes in P_D. Then there exists an $a \in F$ and a $\varphi \in$ `func`$(\{0, 1, ..., m-1\} \backslash D, F)$ such that if we randomize the probabilistic inputs at the nodes of P_D independently and with uniform distribution, then the following holds. The output of C, as a function defined on the set of output nodes `outset`(C), is uniformly distributed on the set of all functions ρ such that for all $i \in \{0, 1, ...m\} \backslash D$ implies $\rho(\mathcal{T}_i) = \varphi(i)$ and $\sum_{x \in \texttt{outset}(C)} \rho(x) = a$.*

(45) *The set D is a connected subset of the graph G.*

The following definition, using the set of whose existence is claimed in this lemma, will be used later in the construction of the multiplication circuit.

Definition. Assume that $C = \mathbf{C}^{(=)}_{m,G}$ is a copying circuit and $A \subseteq C$. We arbitrarily select and fix a set D satisfying the conditions of Lemma 10. The set the set of all output nodes $\mathcal{T}_i$ of C with $i \in D$, will be denoted by $\mathtt{exit}(A, C)$.

Corollary 4 *if a set $D \subseteq \{0, 1, ..., m-1\}$ satisfies conditions* (42) and (45) then it also satisfies condition (44), and for all positive integer j, with $j \leq |D|$, it has a subset D' with $|D'| = j$ which also satisfies conditions (44) and (45) with $D{:=}D'$.

Proof of Lemma 10. Assume that a set A is given with $A \subseteq \mathtt{set}(C)\backslash\mathtt{detin}(C)$, and $m > d(1 + \frac{d}{\alpha})|A|$. Let B be the set of all $i \in V = \{0, 1, ..., m-1\}$ so that there exists a $j \in V$ and a $b \in A$ with $\mathcal{G}_{(i,j)} \geq y$. According to Proposition 8, each fixed $y \in A$ has at most d upper bounds in X, therefore $|B| \leq d|A|$. Lemma 9 implies that $V\backslash B$ has a connected component D with at least $\max\{\frac{m}{2}, m - (1 + \frac{d}{\alpha})|B|\} \geq \max\{\frac{m}{2}, m - d(1 + \frac{d}{\alpha})|A|\}$ elements. We claim the set D satisfies all of the conditions of the Lemma. The construction of D already guarantees that conditions (42), (43), and (45) are satisfied by D. For the proof of condition (44) we need the following.

Definition. If V is a finite set and F is a finite field then $\mathtt{func}_0(V, F)$ will denote the set of functions $f \in \mathtt{func}(V, F)$ with $\sum_{x \in V} f(x) = 0$.

Proposition 9 *Assume that F is a finite field $G = \langle V, E\rangle$ is a connected graph, without loops and multiple edges, $V = 0, 1, ..., n-1$, and for all integers i,j with $0 \leq i < jn$, and $(i, j) \in E$, a function $\psi_{i,j} \in \mathtt{func}(V, F)$ is defined by $\psi(i) = 1, \psi(j) = -1$ and for all $k \in \{0, 1, ..., n-1\}\backslash\{i, j\}$, $\psi(k) = 0$. Let Ψ_G be the vectorspace generated by all of the functions ψ_e, $e \in E$. Then $\Psi_G = \mathtt{func}_0(V, F)$.*

Proof. Clearly $\Psi_G \subseteq \mathtt{func}_0(V, F)$. We prove $\Psi_G = \mathtt{func}_0(V, F)$ by induction on n. Let x be an arbitrary point of V, and let y be an element with a maximal distance from x. Clearly $W = V\backslash\{y\}$ is a connected subset of V, therefore by the inductive assumption $\Psi_{G'} = \mathtt{func}_0(W, F)$, where G' is the restriction of the graph G to the vertex set W. Since G is connected there exists a $z \in W$ with $(x, z) \in G$. $\psi_{(x,z)}$ and $\mathtt{func}_0(W, F)$ together clearly generate $\mathtt{func}(V, F)$. *Q.E.D.*(Proposition 9)

Proposition 10 *Assume that $a_0, ..., a_{k-1}$ generate V, where V is a vectorspace over a finite field F. If $x \in F$, then the number of sequences $\langle \alpha_0, ..., \alpha_{k-1}\rangle \in F$ with $\sum_{i=0}^{k-1} \alpha_i a_i = x$ does not depend on the choice of the element x.*

Proof. The set of all sequence with the given property is a coset of the subspace containing all sequences whose sum is 0. *Q.E.D.*(Proposition LL4)

Proof of Lemma 4 continued. Using Proposition 9 condition (44) is an immediate consequence of the definition of the circuit $\mathbf{C}^{(=)}_{m,G}$, and the fact that D is a connected set. Indeed changing the input values corresponding to the input nodes in P_D we may change the output in any way on the set $\{\mathcal{T}_i \mid i \in D\}$ provided that we do not change the sum of the output values. Clearly the output values outside this set remain unchanged. The uniformity of the distribution follows from Proposition 10. *Q.E.D.*(Lemma 10)

Proof of Corollary 4. The first statement of the Corollary holds since in our proof we have used the lower bound on $|D|$ only to show that (43) holds, but is was not needed for (44). The second statement is a consequence of the fact that for each connected graph G with n elements and for each $j \in [1, n]$, G has a connected subgraph with j elements. *Q.E.D.*(Corollary LL2)

Lemma 11 *Assume that $m \geq 2$, $C = \mathbf{C}^{(=)}_{m,G}$ is an $(d, 1 + \alpha)$ copying circuit, and $A \subseteq \mathtt{set}(C)$ with the property that $m > (3d + 1)|A|$. Then the block input of C is invisible from the set A.*

Proof. Suppose that $g \in \mathtt{func}(A, F)$ and for all $s \in F$, $\delta^{(s)}$ is the function which assigns s to the single input block of C. We have to show that the number of solution of the system $\mathcal{I}(\delta^{(s)}) \cup \mathcal{E}(g)$ does not depend on the choice of $s \in F$. The fact that we have not used multiplication gates in the definition of $\mathbf{C}^{(=)}_m$ implies that all of these systems are linear. Moreover only their inhomogeneous part depends on s. Therefore according to Proposition 6 it is sufficient to show that if one of these systems has at least one solution then each of them system has at least one solution.

Assume that ρ is an evaluation of the unknowns z_i, $i = 0, 1, \ldots, m - 1$, z_e, $e \in E$, which is a solution of the system $\mathcal{I}(\delta^{(s)}) \cup \mathcal{E}(g)$ for some $s \in F$. (Here for the sake of brevity we wrote z_i instead of $z_{\wp_i}$ and z_e instead of $z_{\mathcal{G}_e}$.) Let s' be another arbitrary element of F. We show that $\mathcal{I}(\delta^{(s')}) \cup \mathcal{E}(g)$ also has a solution.

The assumption $m > (2d + 1)|A|$ and Proposition 8 imply that there exists an integer $i_0 \in \{0, 1, \ldots, m - 1\}$ so that for all $a \in A$, $\wp_{i_0} \not\succeq_C a$. We define an evaluation σ of the unknowns by $\sigma(z_{i_0}) = \rho(z_{i_0}) + s' - s$, and $\sigma(z_h) = \rho(z_h)$ if $h \in E \cup (\{0, 1, \ldots, m - 1\}\backslash\{i_0\})$. Clearly $\sum_{j=0}^{m-1} \sigma(z_j) = s' - s + \sum_{j=0}^{m-1} \rho(z_j) = s'$, that is, σ is a solution of $\mathcal{I}(\delta^{s'})$. The choice of the element a, and the definition of the polynomial $p^{(g)}_a$ implies that, $p^{(g)}_a$ does not contain the unknown z_{i_0}, and consequently if we evaluate $p^{(g)}_a$ according to ρ and σ we get the same element of F. This implies that σ is also a solution of $\mathcal{E}(g)$. *Q.E.D.*(Lemma 11)

Lemma 12 *Assume that d is a positive integer, $\alpha > 0$, and the reals ε, c, θ are chosen so that $0 < \varepsilon \ll c \ll \theta \ll \alpha, d$, the integer m is a sufficiently large, F is a finite field, and $C = \mathbf{C}^{(=)}_{m,G}$ is the copying circuit, where G is an $(d, 1 + \alpha)$-expander on n vertices. Then the circuit C is (θ, ε, p)-cylindrical, where $p = 2^{-cm}$.*

Proof. We have to define a (θ, ε, p)-cylindrical adversary $\mathcal{Y}$ for the circuit C. For the definition of the selection function $\mathcal{S}$ of adversary $\mathcal{Y}$ let X be a subset of C. If $m > d(1 + \frac{d}{\alpha})|X|$ then let D be the subset of $\{0, 1, ..., m - 1\}$ whose existence is guaranteed by Lemma 10 with A:=X. Otherwise $D = \emptyset$. We define $\mathcal{S}(X)$ by $\mathcal{S}(X) = \{\mathcal{T}_i \mid i \notin D\}$. This, by the definition of D implies $(X \cap \mathtt{outset}(C)) \subseteq \mathcal{S}(X)$. We have to show that this definition satisfies conditions (11) and (12) from the definition of (θ, ε, p)-cylindrical adversary with $p = 2^{-cm}$.

First we give an upper bound on an ε-random subset X of $\mathtt{set}(C)$. Let $\beta > 0$ be such that $\theta << \beta << \alpha, d$. Chernoff's inequality implies that , with a probability of at least 2^{-cm},

$|X| \leq \beta\theta m$. We show that such a set X is acceptable with respect to adversary $\mathcal{Y}$. Since $|X| \leq \beta m$, and β is sufficiently small with respect to d and α, condition (43) of Lemma 10 implies that $|D| \geq 1 - \theta m$ and so $|\mathcal{S}(X)| \leq \theta m$.

Condition (11) is an immediate consequence of condition (44) of Lemma LL1. Condition (12) follows from Lemma 11. *Q.E.D.*(Lemma 12)

Lemma 13 *Assume that d is a positive integer, $\alpha > 0$, $\varepsilon > 0$ is sufficiently small, m is a sufficiently large positive integer, $C = \mathbf{C}^{(=)}_{m,G}$ is a $(d, 1+\alpha)$-copying circuit, $A \subseteq \mathtt{set}(C)$, $|A| \leq \varepsilon m$ and Γ is the system of equation consisting of the only equation $P_0 = Q_0$. Then the circuit $\mathbf{C}^{(=)}_{m,G}$ is A, Γ-percolative.*

Proof. Since $|A| \leq \varepsilon m$ and $\varepsilon > 0$ is sufficiently small with respect to d and α the set A satisfies the assumptions of Lemma 10. Let D be the set whose existence is stated in that lemma. We define the set X whose existence is required in the definition of percolativity by $X = \bigcup_{i \in D}\{\wp_i, \mathcal{T}_i\}$. We claim that conditions (39) and (40) are satisfied by the set X. Condition (39) is a consequence of the lower bound (43) of Lemma 10 and the assumption $|A| \leq \varepsilon m$. Condition (40) is a consequence of condition (44) of Lemma 10. Indeed, this implies that the values of the probabilistic inputs on the input nodes in P_D can be selected in a way that the output outside X does not change and the output on $X \cap \mathtt{outset}(C)$ can be arbitrary with the only restriction that the sum of all output values must be the same as the sum of all input values. *Q.E.D.*(Lemma 13)

5 Trees

In this section we will use the definitions about trees stated in Section 3.3. We will also prove Lemma 1 and Lemma 14 which were stated in that section. We need these results about trees before we continue the definition of the block F-circuits whose existence is claimed in Lemma 3. These two lemmas and Lemma 17 which will be formulated in this section (also about trees), will be important for proving the (θ, ε, p)-cylindricity of the multiplication circuit that we will define later. We start with the proof of the two lemmas that were stated in section 5.

Proof of Lemma 1. We prove the lemma by induction on j. For $j = 0$ the only element of L_j is t_0, therefore the two sides of the equality in the conclusion of the lemma are identical. Assume now that the lemma holds for $j-1$. Then we have $\lambda(t_0)(\lambda(\mathbf{l}t_0)+\lambda(\mathbf{r}t_0)) = \sum_{w \in L_{j-1}} \lambda(w)(\lambda(\mathbf{l}w)+\lambda(\mathbf{r}w)) = \sum_{w \in L_{j-1}}[\lambda(w)\lambda(\mathbf{l}w) + \lambda(w)\lambda(\mathbf{r}w)]$

By assumption (3) of the lemma for each $w \in L_{j-1}$ we may replace the first occurrence of $\lambda(w)$ in the last expression by $\lambda(r_0(w)) + \lambda(r_1(w))$ and its second occurrence by $\lambda(r_2(w)) + \lambda(r_3(w))$. We get

$\lambda(t_0)(\lambda(\mathbf{l}t_0)) + \lambda(\mathbf{r}t_0))) = \sum_{w \in L_{j-1}}[(\lambda(r_0(w)) + \lambda(r_1(w)))\lambda(\mathbf{l}w) + (\lambda(r_2(w)) + \lambda(r_3(w)))\lambda(\mathbf{r}w)]$

Each $t \in L_j$ is either of the form of $\mathbf{l}w$ or $\mathbf{r}w$ for some $w \in L_{j-1}$ and we get each t exactly once this way. Moreover if $t = \mathbf{l}w$, then $r_0(w) = \mathbf{l}t$, $r_1(w) = \mathbf{r}t$, if $t = \mathbf{r}w$, then $r_2(w) = \mathbf{l}t$, $r_3(w) = \mathbf{r}t$, therefore the last equality implies the conclusion of the lemma. *Q.E.D.*(Lemma 1)

In the proof Lemma 2 we will handle the even and odd levels of the tree separately. The following lemma is formulated about the even levels, but will be equally applicable to the odd ones.

Lemma 14 *Assume that, $\varepsilon > 0$, d is a positive integer, T is a tree of depth $2d$ and A' is a random subset of L_{2d-1} so that all of the events $x \in A'$, $x \in L_{2d-1}$, are mutually independent, and $\texttt{prob}(x \in A') \leq \varepsilon$ for all $x \in L_{2d}$. Let $A = \{a \in L_{2d} \mid \exists b \in A', a \leq_T b\}$.*

Then the probability of the following event is at least $1 - (4\varepsilon)^{2^{d-1}}$:

For all functions λ' defined on A with values in the field F, if λ' has a well-balanced extension to $\mathcal{L}^{(0)}_{2d} = \bigcup_{i=0}^{d} L_{2i}$, then for each $\delta \in F$, λ' has a well-balanced extension λ to $\mathcal{L}^{(0)}_{2d}$, so that $\lambda(t_0) = \delta$, where t_0 is the root of the tree.

Corollary 5 *Suppose that we define the set A of the lemma as a random subset of L_{2d} so that the events $x \in L_{2d}$ are mutually independent and $\texttt{prob}(x \in A) \leq \varepsilon$. Then the probability of the event described in the last paragraph of the lemma is at least $1 - (2\varepsilon)^{2^d}$*

Proof. In the proof of this lemma "well-balanced" will always mean "well-balanced on $\mathcal{L}^{(0)}_{2d}$". Let B be the event described in the lemma. We have to show that $\texttt{prob}(B) \geq 1 - (4\varepsilon)^{2^d}$. We will show that if B does not hold then we have the following:

There exists a subset D of $T_0 = \mathcal{L}^{(0)}_{2d}$ with the following properties:

(46) $t_0 \in D$

(47) *For each $i = 0, ..., d-1$, if $t \in L_{2i} \cap D$ then there exists exactly two elements $u \in L_{2i+2} \cap D$ so that $u \leq t$. Moreover, either $r_0(t) \in D \wedge r_1(t) \in D$, or $r_2(t) \in D \wedge r_3(t) \in D$.*

(48) $L_{2d} \cap D \subseteq A$

First we show that the probability that such a set D exists is at most $(4\varepsilon)^{2^d}$. Conditions (46) and (47) imply that for each $i = 0, 1, ..., d$ we have $|L_{2i} \cap D = 2^i|$.

According to condition (47) we have $|D \cap L_{2i}| = 2^i$, $i = 0, 1, ..., d-1$, and for each $t \in L_{2i}$, $i = 0, ..., d-1$ the two elements u of $L_{2i+2} \cap D$ with $u \leq t$ can be selected in 2 different ways. Consequently the number of sets D satisfying conditions (46) and (47) is $2^{2^d - 1}$.

For each fixed D satisfying conditions (46) and (47) the probability, for the randomization of A, that D satisfies condition (48) as well is at most $\varepsilon^{\frac{1}{2}|L_{2d-1} \cap D|} = \varepsilon^{2^{d-1}}$.

Therefore the probability that there exists a set D with properties (46), (47), and (48) is at most $2^{2^d - 1} \varepsilon^{2^{d-1}} \leq 4^{2^{d-1}} \varepsilon^{2^{d-1}} = (4\varepsilon)^{2^d}$.

Now we show that $\neg B$ implies the existence of a set D with properties (46), (47), and (48).

Assume that a function λ' defined on A is fixed so that it has a well-balanced extension to T_0.

For each $t \in T$, M_t will denote the set of all $x \in T_0$ with $x \leq t$. We say that a function μ, defined on M_t with values in F, is well-balanced on M_t if for all $i = 1, ..., d-1$ and $w \in M_t \cap L_{2i}$ we have $\mu(w) = \mu(r_0(w)) + \mu(r_1(w)) = \mu(r_2(w)) + \mu(r_3(w))$. According to this definition, $t \in L_{2d}$ implies that every F-valued function defined on $M_t = \{t\}$ is well balanced.

A $t \in T_0$, will be called a free element of T_0 if for each $\delta \in F$ there exists a well-balanced function μ on M_t so that μ and λ' are compatible (identical on the intersection of their domains) and $\mu(t) = \delta$.

We claim that

Proposition 11 *if $t \in L_{2i}$ for some $i = 1, ..., d-1$, and t is not free then either both $r_0(t)$ and $r_1(t)$ are not free or both $r_2(t)$ and $r_3(t)$ are not free.*

Proof. Let $\rho_i = r_i(t)$ for $i = 0, 1, 2, 3$. The four sets M_{ρ_i}, $i = 0, 1, 2, 3$ are pairwise disjoint. Assume that contrary to our claim there exist $k \in \{0, 1\}$ and $l \in \{2, 3\}$ so that ρ_k and ρ_l are free. We may assume that e.g., $k = 0$, $l = 2$. We will show that this implies that for each $\delta \in F$, λ' has a well-balanced extension κ to M_t so that $\kappa(t) = \delta$. Let μ be a well-balanced extension of λ' to T_0. κ will be identical to μ on M_{ρ_1} and M_{ρ_3}. Since ρ_0 is free there is a well balanced extension κ' of λ' to M_{ρ_0} so that $\kappa'(\rho_0) = \delta - \mu(\rho_1)$. In the same way we get a well-balanced extension κ'' of λ' onto M_{ρ_2} so that $\kappa''(\rho_2) = \delta - \mu(\rho_3)$. κ will be the unique function on M_t so that $\kappa(t) = \delta$, κ is identical to μ on $M_{\rho_1} \cup M_{\rho_3}$, and κ is an extension of both κ' and κ''. The definitions of κ' and κ'' imply that κ is well-balanced on M_t in contradiction to the assumption that t is not free. *Q.E.D.*(Proposition 11).

Finally if we have $\neg B$, then clearly the element t_0 is not free. Using Proposition 11 we can construct recursively a set D consisting of elements which are not free, so that D satisfies conditions (46), (47), (48). *Q.E.D.*(Lemma 14)

The proof of Corollary 5 is almost identical to the proof of the Lemma. We get the better bound on the probability, since $|D \cap L_{2d}| = 2|D \cap L_{2d-1}|$, and so our upper bound on probability that the random set A of the Corollary contains $D \cap L_{2d}$ is much smaller, than the upperbound on the probability that the random set A' of the Lemma, contains $D \cap L_{2d-1}$.

Proof of Lemma 2. The Lemma is an immediate consequence of Lemma 14. We consider only the case when d is even, say, d=2d', the $d = 2d' + 1$ case can be handled in a similar way. We apply Lemma 14 three times in the following situations. (a) The even levels of the tree T with d:=d'. We get a function λ_0 with $\lambda_0(t_0) = \delta_0$ (b) The even levels of the tree T', where T' consists of all of the nodes $x \in T$ with $x \leq_T \mathbf{l}t_0$. We get a function λ_1 with $\lambda_1(\mathbf{l}t_0) = \delta_0$ (c) The even levels of the tree T'', where T'' consists of all of the nodes $x \in T$ with $x \leq_T \mathbf{r}t_0$. We get a function λ_2 with $\lambda_0(\mathbf{r}t_0) = \delta_2$. The common extension of the functions λ_0, λ_1, and λ_2 will be λ. Since all of the functions λ_j, $j = 0, 1, 2$ were well-balanced on their domains λ is well-balanced too. *Q.E.D.*(Lemma 2)

Lemma 15 *There exist $\alpha > 0, \beta > 0$ so that for all sufficiently large integer n we have the following. Assume that $\xi_1, ..., \xi_n$ are independent random variables with $0, 1$-values, $a_1, ..., a_n$ are reals in the interval $[0, 1]$, $\texttt{prob}(\xi_i = 1) = p$, for $i = 1, ..., n$, $S = \sum_{i=1}^n a_i$, and $\sigma = \sum_{i=1}^n \xi_i a_i$. Then with a probability of at least $1 - n^{-\alpha}$ we have that $|p\frac{S}{n} - \frac{\sigma}{n}| < n^{-\beta}$.*

In the proof of Lemma 15 we will use the following lemma about the sum of independent random variables. For a proof see [14], Corollary A.7. in Appendix A.

Lemma 16 *Assume that $Z_1, ..., Z_n$ are independent random variables with $\texttt{prob}(Z_i = 1) = p$, $\texttt{prob}(Z_i = 0) = 1 - p$ for $i = 1, ..., n$, where $p \in [0, 1]$, and $Z = Z_1 + ... + Z_n$. Then, for all $a > 0$, $\texttt{prob}(|Z - pn| > a) < 2e^{-ea^2/n}$.*

Proof of Lemma 15. $T = p\frac{S}{n} - \frac{\sigma}{n} = \sum_{i=1}^n \frac{1}{n}(pa_i - \xi_i a_i)$. We partition the set $\{1, ..., n\}$ into pairwise disjoint classes $C_1, ..., C_d$. Suppose $|C_j| = \nu_j$. Then $T = \sum_{j=1}^d \frac{\nu_j}{n} \sum_{i \in C_j} \frac{1}{\nu_j}(pa_i - \xi_i a_i)$.

Since T is a convex linear combination of the sums $U_j = \sum_{i\in C_j} \frac{1}{\nu_j}(pa_i - \xi_i a_i)$, it would be sufficient to show that if the constants $\alpha > 0, \beta > 0$ are sufficiently small then for each fixed $j = 1, ..., d$, with a probability of at least $1 - d^{-1}n^{-\alpha}$, we have $|U_j| < n^{-\beta}$. Although this will not hold for the classes C_i that we will define later, but the following weaker condition will hold:

For all $j = 2, ..., d$, $|U_j| < \frac{1}{2}n^{-\beta}$ and $\frac{\nu_1}{n}U_1 < n^{-\frac{1}{6}}$. Clearly this is also sufficient for the conclusion of the lemma.

Let I_i be the interval $((1-n^{-\frac{1}{2}})^{i+1}, (1-n^{-\frac{1}{2}})^i]$, for $i = 0, 1, ..., q-1$, where q is the smallest positive integer with $(1-n^{-\frac{1}{2}})^q < \frac{1}{n}$, and let $I_q = [0, (1-n^{-\frac{1}{2}})^q)$. Clearly the intervals $I_0, ..., I_q$ cover the interval $[0,1]$, and $q \le 2n^{\frac{1}{2}}\log n$.

Let $X_j = \{i \in [0,q] \mid a_i \in I_j\}$. Let $B = \{j \in [0, q-1] \mid |X_j| < n^{\frac{1}{4}}\}$, $D = \bigcup_{j\in B} X_j$. Now we define the classes $C_1, ..., C_d$. C_1 will be the set $D \cup X_q$. The other classes will be the sets X_j, $j \in [0,q]\backslash B$ in an arbitrary order.

First we estimate $|U_1|$, by estimating the sums corresponding to D and X_d separately. We have $\sum_{i\in D} |\frac{1}{n}(pa_i - \xi_i a_i)| \le \frac{1}{n}qn^{\frac{1}{4}} \le \frac{1}{n}n^{\frac{1}{4}}2n^{\frac{1}{2}}\log n = 2n^{-\frac{1}{4}}\log n < n^{-\frac{1}{5}}$. Clearly $|\sum_{i\in X_q} \frac{1}{n}(pa_i - \xi_i a_i)| \le n\frac{1}{n}\frac{1}{n} \le \frac{1}{n}$. Therefore $|U_1| < n^{-\frac{1}{5}} + \frac{1}{n} < n^{-\frac{1}{6}}$ as claimed.

Assume now that $j \in [1,n]\backslash B \cup \{q\}$. We estimate $S_j = \sum_{i\in X_j} \frac{1}{\nu_j}(pa_i - \xi_i a_i)$, where $\nu_j = |X_j|$. The interval I_j is of the following form $I_j = ((1-n^{-\frac{1}{2}})b_j, b_j]$. The definition of X_j implies that for each $i \in X_j$ there is a real r_i with $0 \le r_j \le n^{-\frac{1}{2}}$ so that $a_i = b_j(1-r_i)$. Therefore $S_j = \frac{1}{\nu_j}\sum_{i\in X_j}[(p-\xi_i)b_j - (p-x_i)b_j r_i] = R_i + \frac{1}{\nu_j}b_j\sum_{i\in X_j}(p-\xi_i)$, where $|R_i| \le n^{-\frac{1}{2}}$. Lemma 16 implies that if $\gamma > 0$ is a small constant then $\mathtt{prob}(|\sum_{i\in X_j}(p-\xi_i)| \ge \nu_j^{\frac{1}{2}}n^{\gamma}) < 2e^{-\nu_j n^{2\gamma}/\nu_j} = 2e^{-n^{2\gamma}}$. Therefore if $U_k = S_j$ then with a probability of at least $1 - n^{-\alpha}$ we have $|U_k| < \nu_j^{-\frac{1}{2}}n^{\gamma} < n^{-\beta}$ as required. *Q.E.D.*(Lemma 15)

Lemma 17 *For all $\varepsilon' > 0$ and for all sufficiently small $\varepsilon > 0$ the following holds. Assume that T is a binary tree of depth d and H is a subset of the leaves of the tree with at least 2^{d-1} elements. Suppose further that A is a random subset of T so that each element of T will belong to A with a probability of at most ε, and the events $t \in A$ are independent for all $t \in T$. Then the probability that there exists a branch of T disjoint from A and containing an element of H is at least $1 - \varepsilon'$.*

Corollary 6 *The statement of the lemma remains true if we replace its conclusion with the following. "Then the probability that there exists a branch B of T disjoint from A, containing an element of H and satisfying condition* (49) *below is at least $1 - \varepsilon'$."*

(49) *For all $b \in B$, $a \in A$, the element "a" is not an immediate ancestor of b, and the element b is not an immediate ancestor of "a".*

Proof of Lemma 17. Let B be the set of all points of the tree which can be connected to the root with a path disjoint from A. L_i denotes the set of points of T whose distance from the root is i, $G_i = L_i \cap B$.

Sketch of the proof. First we prove the $|G_i)| \ge 2^{\frac{i}{2}}$ lower bound which holds with a probability of $1 - \varepsilon'$ for all $i = c_0, ..., d$ for a sufficiently small constant $\varepsilon' > 0$. We use Lemma 15 to show that if this lower bound holds for $|G_i|$ then it holds for $|G_{i+1}|$ as well. This will be a consequence of the fact that for each element of G_i the expected number of its successor in G_{i+1} is $(1-\varepsilon)2$. Therefore to get a lower bound on G_{i+1} from a lower bound on G_i, we have to add independent random variables with large expected values. With the help of Lemma 15 we will estimate the probability of the event that the sum is much smaller then expected.

Using the lower bound on $|G_i|$ we continue the proof in the following way. We will show the following by induction on i. We consider the natural partial ordering of the tree so that the root is the largest element. If $t \in T$ then M_t will denote the set $\{x \in T \mid x \le t\}$. For all $i = 0, 1, ..., d$ let $W_i = \bigcup_{t \in G_i} H \cap M_t$, and let $\tau_i = |W_i||G_i|^{-1}2^{i-d}$. In other words, τ_i is the density of H among those elements of L_d, which have an upper bound in G_i. $|H| \ge 2^{d-1}$ implies that for $i = 0$, this density is at least $\frac{1}{2}$, (provided the $G_0 \ne \emptyset$). With a similar application of Lemma 15 that we used to prove the lower bound on $|G_i|$, we will show that the difference $\tau_i - \tau_{i+1}$ is small with high probability. Using the lower bound on $|G_i|$ we will be able to show that this happens simultaneously for all $i = 0, 1, ..., d-1$ with high probability. Adding up all of the differences $\tau_i - \tau_{i+1}$ we get $\tau_d > \frac{1}{4}$ which implies the statement of the lemma. *End of sketch.*

Now we continue the proof of Lemma 17.

Claim 1 *There exists, $c_1 > 0$ so that if $\varepsilon > 0$ is sufficiently then for all positive integers d the following holds. Let K_i be the event $|G_i| \ge 2^{\frac{i}{2}}$ for all $i = 0, 1, ..., d$. Then $\mathtt{prob}(K_{i+1}|K_i) \ge 1 - 2^{-c_1 i}$ for all $i = 0, 1, ..., d-1$.*

Proof. First we note that for a fixed $c_1 > 0$ and fixed positive integer i the inequality $\mathtt{prob}(K_{i+1}|K_i) \ge 2^{-c_1 i}$ holds for all sufficiently small $\varepsilon > 0$. This is a consequence of the fact that $\mathtt{prob}(|G_i| = 2^i) \ge 1 - 2^i\varepsilon$. Therefore it is sufficient to show that that there exists an absolute constant c_0 so that the statement in Claim 1 holds for all $i \in [c_0, d-1]$.

Assume that the elements of A which are in levels $L_0, ..., L_i$ has been already randomized so that K_i holds and $G_i = \{t_1, ..., t_m\}$ For each $j = 1, ..., 2m$ we define a random variable ξ_j for the randomization of the elements of A in level L_{i+1}. Namely ξ_{2j} is the number of left successors of t_j in level L_{i+1} which are not in the set A, and ξ_{2j+1} is the number of right successors with the same property. (Left and right successors can be chosen arbitrarily.) The definition of A implies that $\xi_1, ..., \xi_{2m}$ are mutually independent and ξ_j takes the values $0, 1$ with probabilities $(1-\varepsilon)$, ε. Clearly $|G_{i+1}| = \sum_{j=1}^{2m} \xi_j$.

Applying Lemma 15 to the sequence $\xi_1, ..., \xi_n$, where $n = 2m$ we get that

(50) *with a probability of at least $1 - n^{-\alpha}$, $|G_{i+1} - (1-\varepsilon)2|G_i|| \le n^{-\beta}2|G_i|$, and so $|G_{i+1}| \ge (1 - \varepsilon - 2n^{-\beta})2|G_i|$, provided that $i > c_0$, where c_0 is an absolute constant.*

The assumption that K_i holds implies that $n = G_i > 2^{\frac{i}{2}}$ so we have that with a probability of at least $1 - 2^{1-\frac{\alpha i}{2}}$ we have $|G_{i+1}| \ge (1 - \varepsilon - 2^{1-\frac{\beta i}{2}})2^{\frac{i}{2}} \ge 2^{\frac{(i+1)}{2}}$. This completes the proof of Claim 1.

Claim 2 *For all $\varepsilon' > 0$ if $\varepsilon > 0$ is sufficiently small, then for all positive integers d, with a probability of at least $1 - \varepsilon'$, for the randomization of A, we have that for all $i = 0, 1, ..., d$, $|G_i| \ge 2^{\frac{i}{2}}$.*

Proof. The series $\sum_{i=0}^{\infty} 2^{-c_1 i}$ is convergent, where $c_1 > 0$ is the constant from Claim 1. Let i_0 be the smallest positive integer so that $\sum_{i=i_0}^{\infty} 2^{-c_1 i} < \varepsilon'/2$. Suppose now that $\varepsilon > 0$ is sufficiently small with respect to ε'. Then with a probability of at least $1 - \frac{\varepsilon'}{2}$ we have $|G_i| = 2^i \geq 2^{\frac{i}{2}}$ for all $i = 0, 1, ..., i_0$. According to Claim 1, with a probability of at least $1 - \frac{\varepsilon'}{2}$ we also have that for all $i = i_0, ..., d-1$, $K_i \to K_{i+1}$ and therefore $|G_i| \geq 2^{\frac{i}{2}} \to 2^{\frac{i+1}{2}}$. Therefore with a probability of at least $1 - \varepsilon'$ we have that $|G_i| \geq 2^{\frac{i}{2}}$ for all $i = 0, 1, ..., d$, which completes the proof of Claim 2

We assume that $\leq$ is the natural partial ordering of the binary tree T, where the root is the largest element and $a < b$ iff they are on the same branch of tree and b is closer to the root. If $t \in T$ then M_t will denote the set $\{x \in T \mid x \leq t\}$.

For all $i = 0, 1, ..., d$ let $W_i = \bigcup_{t \in G_i} H \cap M_t$, and let $\tau_i = |W_i||G_i|^{-1} 2^{i-d}$.

We will show that:

Claim 3 *there exists an $i_2 > 0$ so that if $\alpha' > 0$ and $\beta' > 0$ are sufficiently small then for all positive integer d the following holds. Assume that the tree T the sets, A, G_i, and the real τ_i are defined as above. Then, for all $i = i_2, ..., d-1$, if $|G_i| \geq 2$ then with a probability of at least $1 - |G_i|^{-\alpha'}$ we have $\tau_{i+1} > \tau_i - |G_i|^{-\beta'}$.*

Proof. Condition (50) (of the proof of Claim 1) and the facts $n = 2|G_i|$, $|G_i| \geq 2$ implies that if α, β are sufficiently small absolute constants then

(51) *with a probability of at least $1 - |G_i|^{-\alpha}$ we have that $|G_{i+1}| = 2(1-\varepsilon)(1+R_1)|G_i|$, where $|R_1| < |G_i|^{-\beta}$.*

By the definition of τ_i we have,

$$\frac{\tau_{i+1}}{\tau_i} = \frac{|W_{i+1}||G_{i+1}^{-1}|2^{i+1-d}}{|W_i||G_i^{-1}|2^{i-d}}$$

Therefore, using property (51) we get that with a probability of at least $1 - |G_i|^{-\alpha}$ we have that

$$\frac{\tau_{i+1}}{\tau_i} = \frac{|W_{i+1}|}{|W_i|}(1-\varepsilon)(1+R_1)$$

where $|R_1| < |G_i|^{-\beta}$

Let $G_i = \{t_1, ..., t_m\}$ and for all $j = 1, ..., m$ let $t_{j,0}$, $t_{j,1}$, be the two successors of t in the tree T, and let κ_A be the characteristic function of the set A defined on T. For each $j = 1, ..., m$, $\delta = 0, 1$ we define the random variable $\zeta_{j,\delta}$ by $\zeta_{j,\delta} = 1 - \kappa_A(t_{j,\delta})$. (We assume here that A has been randomized already on $L_0 \cup ... \cup L_i$.) Let $b_{i,\delta} = 2^{i+1-d}|H \cap M_{t_{j,\delta}}|$. We have $|W_i| = 2^{d-i-1} \sum_{j=1}^{m} \sum_{\delta=0}^{1} b_{j,\delta} \zeta_{j,\delta}$. We apply Lemma 15 with $n = 2m$ and $\xi_{2j+\delta} := \zeta_{j,\delta}$, $a_{2j+\delta} = b_{j,\delta}$ for $j = 1, ..., m$, $\delta = 0, 1$, $p := (1-\varepsilon)$.

We have $S = \sum_{j=1}^{m} \sum_{\delta=0}^{1} b_{j,\delta} = \sum_{j=1}^{m} \sum_{\delta=0}^{1} 2^{i+1-d}|H \cap M_{t_{j,\delta}}| = 2^{i+1-d}|W_i|$ and
$\sigma = \sum_{j=1}^{m} \sum_{\delta=0}^{1} \kappa_A(t_{j,\delta}) b_{j,\delta} = 2^{i+1-d} \sum_{t \in G_{i+1}} |H \cap M_t| = 2^{i+1-d}|W_{i+1}|$.

Therefore $n = |G_i|$ and Lemma 15 imply that $|(1-\varepsilon)2^{i+1-d}|W_i||G_i|^{-1} - 2^{i+1-d}|W_{i+1}||G_i|^{-1}| \leq |G_i|^{-\beta}$. Using property (51), we substitute the second occurrence of $|G_i|^{-1}$ by $2(1-\varepsilon)(1+R_1)|G_{i+1}|^{-1}$. Since $\tau_i = |W_i||G_i|^{-1}2^{i-d}$ we get that

$$|(1-\varepsilon)2\tau_i - (1-\varepsilon)2\tau_{i+1}| \leq |G_i|^{-\beta} + R_1(1-\varepsilon)2\tau_{i+1}$$

where $|R_1| \le |G_i|^{-\beta_0}$ for an absolute constant β_0. This implies that with a probability of at least $1 - |G_i|^{-\alpha}$ we have $|\tau_i - \tau_{i+1}| < |G_i|^{-\beta}$ which completes the proof of Claim 3

Claim 4 *for all $\varepsilon' > 0$ if $\varepsilon > 0$ then with a probability of at least $1 - \varepsilon'$ we have that $\tau_i > \frac{1}{4}$ for all $i = 1, ..., d$ and consequently T has a branch ending in a point of H and not containing any points from A.*

Proof. Let $k > \max\{i_2, 4\}$, where i_2 is the integer whose existence is stated in Claim 3. If $\varepsilon > 0$ is sufficiently small, then with a probability of at least $1 - \frac{\varepsilon'}{2}$ the set $L_0 \cup ... \cup L_k$ does not contain any points from A, therefore $|H| \ge 2^{d-1}$ implies that $\tau_k \ge \frac{1}{2}$. $k \ge 4$ and Claim 1 implies that $|G_i| > 2^{\frac{i}{2}} > 2$ for all $i > 2$. Therefore by Claim 3 we have that for each fixed $i \ge k$ with a probability of at least $1 - 2^{\frac{-\alpha' i}{2}}$ we have that $\tau_{i+1} > \tau_i - 2^{\frac{-\beta' i}{2}}$. Since the series $\sum_{i=0}^{\infty} 2^{\frac{-\alpha' i}{2}}$ and $\sum_{i=0}^{\infty} 2^{\frac{-\beta' i}{2}}$ converge, we have that if k is a sufficiently large constant, then with a probability of at least $1 - \frac{\varepsilon'}{2}$ we have $\tau_{i+1} > \tau_i - 2^{\frac{-\beta' i}{2}}$ for all $i \ge k$ and so $\tau_d \ge \tau_k - \frac{1}{4} \ge \frac{1}{4}$ which completes the proof of Claim 4. *Q.E.D.*(Lemma M6)

Proof of Corollary 6. We will write $x \succ y$ for "x is an immediate ancestor of y. First note that in condition (49) the statement $\neg(a \succ b)$ can be omitted since it follows from the earlier assumption that A and B are disjoint. Therefore we have to estimate the probability of the existence of a branch B with the following properties.

(52) *(i) B is disjoint from A, (ii) B contains an element from H, and (iii) for all $b \in B$, $a \in A$, we have $\neg(b \succ a)$.*

Assume that an $x \in T$ is fixed. Let $\mathcal{R}(x)$ be the event $x \in A \vee (\exists a \in A, x \succ A)$. Clearly, with respect to the randomization of A as defined in the lemma, we have $\texttt{prob}(\mathcal{R}(x)) \le 3\varepsilon$. The events $\mathcal{R}(x)$, $x \in T$ are not independent, so we cannot use Lemma 17 directly with $\varepsilon{:=}3\varepsilon$, and $A{:=}A' = \{x \in T \mid \mathcal{R}(x)\}$. This is however only a superficial difficulty, with a minor modification of the set A' we can make it suitable for the application of the lemma. We define the modified set A' in the following way.

We arrange all of the elements of T in a sequence $t_0, t_1, ..., t_s$ such that if $i < j$, $t_i \in L_i$ and $t_k \in L_j$ then $k < j$. We define $t_k \in A'$ by recursion on k. $t_0 \in A'$ iff $\mathcal{R}(t_0)$. Assume that $t_i \in A'$ has been defined for all $i = 0, 1, ..., k-1$.

If $\exists i \in \{0, 1, ..., k-1\}$, such that $t_i \in A' \wedge t_i >_T t_k$, then we put the element t_k in A' with a probability of 3ε. This randomization is independent of the randomization of A, and of all of the randomizations which were made earlier in this recursive definition. If there exists no $i \in \{0, 1, ..., k-1\}$ with $t_i \in A' \wedge t_i >_T t_k$, then $t_k \in A'$ if $\mathcal{R}(t_k)$.

Clearly the events $\mathcal{R}(t)$ are mutually independent and if a branch B does not contain any elements form A' then it meets the requirements of the corollary. Therefore Lemma 17 with $\varepsilon{:=}3\varepsilon$ and $A{:=}A'$ implies its corollary. *Q.E.D.*(Corollary 6)

6 Various block F-circuits

In this section we will construct several block F-circuits of type $((m, m), k, l)$. In each case we will have $k, l \in \{0, 1, 2\}$. We will show that some of these circuits are $(\theta, \varepsilon_1, p)$-cylindrical. In the constructions of these circuits we will use a copying circuit $\mathbf{C}^{(=)}_{m,G}$, where G is an expander graph. We will always assume about $\theta, \varepsilon_1, p, G$, and F that they satisfy the condition formulated in the following definition.

Definition. The following condition will be called the standard assumption.

(53) *G is a $(d, \alpha+1)$ expander graph, $0 < \varepsilon_1 \ll c_1 \ll \theta \ll \alpha, d$, the integer m is sufficiently large, F is a finite field, and $p = 2^{-c_1 m}$.* $\square$

In the remaining part of the paper we will use the standard assumption unless we explicitly state it otherwise.

Definition. Assume that C is a block F-circuit of type $((m,n),k,l)$ and $G = \langle V, E\rangle$, $V = \{0,1,\ldots,m-1\}$ is a $(d, 1+\alpha)$ expander graph. We define another block F-circuit $\mathtt{pre}_G(C)$ which will be called the pre-copied version of C with respect to G. $C' = \mathtt{pre}_G(C)$ will be a circuit also of type $((m,n),k,l)$. Assume that the deterministic input of C' consists of the k blocks $\delta^{(j)} = \langle \delta^{(j)}_0, \ldots, \delta^{(k-1)}_{m-1}\rangle$, for $j = 0,1,\ldots,k-1$. We take k disjoint copies $C_0,\ldots,C_{k-1}$ of the copying circuit $\mathbf{C}^{(=)}_{m,G}$ and copy the block $\delta^{(j)}$ with the circuit C_j and get the output $\mu^{(j)} = \langle \mu^{(j)}_0, \ldots, \mu^{(j)}_{m-1}\rangle$ for all $j = 0,1,\ldots,k-1$. The circuit C gets the blocks $\mu^{(j)}$ as deterministic input and computes the output, the blocks $\nu^{(s)} = \langle \nu^{(s)}_0, \ldots, \nu^{(s)}_{n-1}\rangle$ for $s = 0,\ldots,l-1$. The probabilistic inputs of $C, C_0, \ldots, C_{k-1}$ are mutually independent. The sequence of blocks $\nu^{(0)},\ldots,\nu^{(l-1)}$ is the output of the circuit $C' = \mathtt{pre}_G(C)$.

We define in a similar way the circuit $\mathtt{post}_G(C)$, which will be called the post-copied version of C with respect to G. Assume that C is a block F-circuit of type $((m,n),k,l)$ and $G = \langle V, E\rangle$, $V = \{0,1,\ldots,n-1\}$ is a $(d,1+\alpha)$ expander graph. The definition of $\bar{C} = \mathtt{post}_G(C)$ is the following. We take l disjoint copies $C_0,\ldots,C_{l-1}$ of the circuit $\mathbf{C}^{(=)}_{n,G}$. Assume that the input of $\bar{C}$ is the sequence of blocks $\delta^{(0)},\ldots,\delta^{(k-1)}$. We give this sequence of blocks to the circuit C as an input and then we copy the jth output block of C with the circuit C_j, for $j = 0,1,\ldots,l-1$. The output block $\nu^{(j)}$ of C_j will be the jth output block of $\bar{C}$. The probabilistic inputs of the circuits $\bar{C}, C_0, \ldots, C_{l-1}$ are mutually independent. $\square$

Lemma 18 *Assume that C is a blockcircuit of type $((m,m),k,l)$ and G is a $(d,1+\alpha)$-expander, $\theta, \varepsilon, p \in (0,1)$, $C' = \mathtt{post}_G(C)$, and C_0 is the copying circuit $\mathbf{C}^{(=)}_{m,G}$ which was added to the circuit C to get C'. Assume further that $\mathcal{Y} = \mathtt{adv}_{C'}(\varepsilon, \mathcal{S})$ is an adversary for the circuit C_0. We define a selection function $\mathcal{S}_0$ for the copying circuit C_0 by $\mathcal{S}_0(X) = \mathtt{set}(C_0) \cap \mathcal{S}(X)$ for all $X \subseteq \mathtt{set}(C_0)$. Let $\mathcal{Y}_0 = \mathtt{adv}_{C_0}(\varepsilon, \mathcal{S}_0)$ be the corresponding adversary for C_0. Then, for all $X \subseteq \mathtt{set}(C')$, if condition* (11) is satisfied by $C{:=}C_0$, $X{:=}X \cap C_0$, $\mathcal{S}{:=}\mathcal{S}_0$, $\mathcal{Y}{:=}\mathcal{Y}_0$ then it is also satisfied by $C{:=}C'$, X, $\mathcal{S}$ and $\mathcal{Y}$.

Proof. Let a be a fixed deterministic input of C'. We restrict the function $\chi^{(C')}_a$ to the set of deterministic input nodes of C_0. Let a_0 be the restricted function. Applying Bayes' theorem with the various possible choices fort the function a_0 we get that the distribution of C' with input a is also θ-cyclical with the same base set. *Q.E.D.*(Lemma 18)

Doubling circuit. We will define below a probabilistic block F-circuit $\mathbf{C}^{(=,2)}_{m,G}$ of type $((m,m),1,2)$, which computes the polynomials x, x, so that the circuit is $(\theta, \varepsilon_1, p)$-cylindrical with the choices of the parameters according to the standard assumptions described in condition (53).

Definition. Assume that m is a positive integer and $G = \langle V, E\rangle$ is a $(d, 1+\alpha)$-expander. We define a circuit $\mathbf{C}^{(=,2)}_{m,G}$ that we will call doubling a circuit. Let $\bar{C}$ be the following deterministic circuit of type $((m,m),1,2)$. Suppose that $\delta = \langle \delta_0, \ldots, \delta_1\rangle$ is the deterministic input of $\bar{C}$. Then we define the output values $\langle \nu^{(0)}_0, \ldots, \nu^{(0)}_{m-1}\rangle$, $\langle \nu^{(1)}_0, \ldots, \nu^{(1)}_{m-1}\rangle$ on the two output blocks of $\bar{C}$, by $\nu^{(j)}_i = \delta_i$ for $i = 0, 1, \ldots, m$. $\mathbf{C}^{(=,2)}_{m,G}$ is defined as $\mathtt{post}_G(\bar{C})$. $\Box$

The following lemma is an immediate consequence of the definition of $C = \mathbf{C}^{(=,2)}_{m,G}$.

Lemma 19 *The circuit $C = \mathbf{C}^{(=,2)}_{m,G}$ computes the polynomials x_0 and x_1. Equivalently we have $P^{(C)}_0 = Q^{(C)}_0 = Q^{(C)}_1$.*

Definition. *Constant* 0 *circuit and constant* 1 *circuits.* We define probabilistic block F-circuits $\mathbf{C}^{(0)}_{m,G}$ and $\mathbf{C}^{(1)}_{m,G}$, both of type $((m,m),0,1)$, where $G = \langle \{0,1,\ldots,m-1\}, E\rangle$ is a $(d, \alpha+1)$ expander graph. For all $i = 0, 1$, $\mathbf{C}^{(i)}_{m,G}$ computes the constant polynomial i without any deterministic input. The circuit $\mathbf{C}^{(i)}_{m,G}$ always gives a block output consisting of a single block of size m with $Q_0 = i$. $\mathbf{C}^{(i)}_{m,G}$ has no deterministic input nodes and for each $e \in G$ it has one probabilistic input node. We define $\mathbf{C}^{(i)}_{m,G}$ in the following way. The circuit first computes the sequence $i, 0, \ldots, 0$ of length m and then copies it with the copying circuit $\mathbf{C}^{(=)}_{m,G}$. The output of this copying circuit is also the output of the circuit for $i = 0, 1$. $\Box$

The following lemma is an immediate consequence of the definition of $\mathbf{C}^{(i)}_{m,G}$

Lemma 20 *For all $i = 0, 1$, the circuit $C = \mathbf{C}^{(i)}_{m,G}$ computes the constant polynomial i. Equivalently we have $Q^{(C)}_0 = i$.*

Definition. *Addition circuit $\mathbf{C}^{(+)}_{m,G}$, subtraction circuit $\mathbf{C}^{(-)}_{m,G}$, and the circuit $\mathbf{C}^{(+1)}_{m,G}$.* We define a probabilistic block F-circuit $\mathbf{C}^{(+)}_{m,G}$ of type $((m,m),2,1)$, where $G = \{0,1,\ldots,m-1\}$ is a $(d, \alpha+1)$-expander, which computes the polynomial $x_0 + x_1$. First we define a deterministic block F-circuit C' of type $((m,m),2,1)$. Assume that the inputs of C' in the two deterministic input blocks are $\delta_0, \ldots, \delta_{m-1}$ and $\gamma_0, \ldots, \gamma_{m-1}$. Then the output of C' is the sequence $\mu_0, \ldots, \mu_{m-1}$, where $\mu_0 = \delta_0 + \gamma_0, \mu_1 = \delta_1 + \gamma_1, \ldots, \mu_{m-1} = \delta_{m-1} + \gamma_{m-1}$. $\mathbf{C}^{(+)}_{m,G}$ is defined as $\mathbf{C}^{(+)}_{m,G} = \mathtt{post}_G(C')$. The subtraction circuit $\mathbf{C}^{(-)}_{m,G}$ is defined in a similar way, the only difference is that now $\mu_i = \delta_i - \gamma_i$, for $i = 0, 1, \ldots, m-1$. The circuit $\mathbf{C}^{(+1)}_{m,G}$ is a probabilistic block F-circuit of type $((m,m),1,1)$. We define first a circuit C' of the same type. If the deterministic input of C' is $\delta_0, \delta_1, \ldots, \delta_{m-1}$, then its output is $1 + \delta_0, \delta_1, \ldots, \delta_{m-1}$. Finally we have $\mathbf{C}^{(+1)}_{m,G} = \mathtt{post}_G(C')$. $\Box$

The following lemma is an immediate consequence of these definitions.

Lemma 21 *The circuit $C = \mathbf{C}^{(+)}_{m,G}$ computes the polynomial $x_0 + x_1$, the circuit $C = \mathbf{C}^{(-)}_{m,G}$ computes the polynomial $x_0 - x_1$, and the circuit $\mathbf{C}^{(+1)}_{m,G}$ computes the polynomial $x_0 + 1$. Equivalently, for $C = \mathbf{C}^{(+)}_{m,G}$ we have $P^{(C)}_0 + P^{(C)}_1 = Q^{(C)}_0$, for $C = \mathbf{C}^{(-)}_{m,G}$ we have $P^{(C)}_0 - P^{(C)}_1 = Q^{(C)}_0$, and for $C = \mathbf{C}^{(+1)}_{m,G}$ we have $P^{(C)}_0 + 1 = Q^{(C)}_0$.*

Definition. *Random generator circuit $\mathbf{C}^{(\mathtt{rand})}_m$.* We define a probabilistic block F-circuit $\mathbf{C}^{(\mathtt{rand})}$ of type $((m,m),0,1)$, so that it has no deterministic input and the sum of the elements of the output sequence, has uniform distribution on F. The circuit has m probabilistic input nodes and m output nodes. If the probabilistic input values are $\gamma_0, \ldots, \gamma_{m-1}$, then the output values are $\nu_0 = \gamma_0, \ldots, \nu_{m-1} = \gamma_{m-1}$. $\square$

The following lemma is an immediate consequence of this definition.

Lemma 22 *If $\nu_0, \ldots, \nu_{m-1}$ are the output values of the circuit $\mathbf{C}^{(\mathtt{rand})}_m$ then $\sum_{i=0}^{m-1} \nu_i$ has uniform distribution on F.*

Definition. Suppose that $\alpha = \langle \alpha_0, \ldots, \alpha_{m-1} \rangle$, $\alpha_i \in F$ for $i = 0, 1, \ldots, m-1$. We will use the notation $\mathbf{S}\alpha = \sum_{i=0}^{m-1} \alpha_i$.$\square$

Lemma 23 *Under the standard assumptions described in* (53) the following circuits are are $(\theta, \varepsilon_1, p)$-cylindrical: $\mathbf{C}^{(=,2)}_{m,G}$, $\mathbf{C}^{(0)}_{m,G}$, $\mathbf{C}^{(1)}_{m,G}$, $\mathbf{C}^{(+)}_{m,G}$, $\mathbf{C}^{(-)}_{m,G}$, $\mathbf{C}^{(+1)}_{m,G}$, $\mathbf{C}^{(\mathtt{rand})}_m$.

Proof. For the circuits $\mathbf{C}^{(=,2)}_{m,G}$, $\mathbf{C}^{(0)}_{m,G}$, $\mathbf{C}^{(1)}_{m,G}$, $\mathbf{C}^{(+)}_{m,G}$, $\mathbf{C}^{(-)}_{m,G}$ we can prove the statement of the lemma in the same way as we have done it for the copying circuit. In the case of $\mathbf{C}^{(=,2)}_{m,G}$ we have now two output blocks and in the cases of $\mathbf{C}^{(+)}_{m,G}$, $\mathbf{C}^{(-)}_{m,G}$ we have two input blocks, however this does not cause any problems. *Q.E.D.*(Lemma 23)

Definition. Let $C = \mathbf{C}^{(+)}_{m,G}$. The system of equations consisting of the only equation $P^{(C)}_0 + P^{(C)}_1 = Q^{(C)}_0$ will be called the canonical system of the circuit $\mathbf{C}^{(+)}_{m,G}$. $\square$

Lemma 24 *Assume that d is a positive integer, $\alpha > 0$, $\varepsilon > 0$ is sufficiently small, m is a sufficiently large positive integer, $C = \mathbf{C}^{(+)}_{m,G}$ is an addition circuit where G is a $(d, 1+\alpha)$-expander, $A \subseteq \mathtt{set}(C)$, $|A| \le \varepsilon m$ and Γ is the canonical system of $C = \mathbf{C}^{(+)}_{m,G}$. Then the circuit $\mathbf{C}^{(+)}_{m,G}$ is A, Γ-percolative.*

Proof. The proof is very similar to the proof of Lemma 13, which is an analogue statement about the circuit $\mathbf{C}^{(=)}_{m,G}$. *Q.E.D.*(Lemma 24)

In the following definition we define a probabilistic block F-circuit $\mathbf{C}^{(\mathtt{spl})}_{m,G}$ of type $((m,m),1,4)$ so that if the deterministic input $\delta \in \mathtt{sequence}_{m,F}$ is fixed, where $\mathtt{sequence}_{m,F}$ denotes the set of all sequences of length m, whose elements are in F, then the output consists of four sequences $\mu^{(0)}, \mu^{(1)}, \mu^{(2)}, \mu^{(3)} \in \mathtt{sequence}_{m,F}$ given on the four output blocks with the property that $\mathbf{S}\delta = \mathbf{S}\mu^{(0)} + \mathbf{S}\mu^{(1)} = \mathbf{S}\mu^{(2)} + \mathbf{S}\mu^{(3)}$. Moreover $\mathbf{S}\mu^{(i)}$, $i = 0, 1, 2, 3$ have uniform distribution on F, and the random variables $\mathbf{S}\mu^{(0)}$ and $\mathbf{S}\mu^{(2)}$ are independent. This circuit will be needed for the definition of an $(\theta, \varepsilon_1, p)$-cylindrical circuit whose block output is the product of its block inputs, that is, the circuit computes the product of two elements of F in the same sense as the circuit $\mathbf{C}_{m,G}$ products the sum of two elements. The factors a, b of the product ab will be repeatedly split into two random parts. This way we get $ab = (a_0+a_1)b = a_0(b_0+b_1)+a_1(b_2+b_3)$ where $a = a_0 + a_1$, $b = b_0 + b_1 = b_2 + b_3$. We will get the four values b_0, b_1, b_2, b_3 as the block outputs of a circuit $\mathbf{C}^{(\mathtt{spl})}_{m,G}$ whose block input is b.

Definition. *Splitting circuit* $\mathbf{C}^{(\mathtt{spl})}_{m,G}$. First we define a probabilistic block F-circuit $\mathbf{C}^{(\mathtt{wspl})}_{m}$ of type $((m, m), 1, 4)$ that we will call a weak splitting circuit. $C_w = \mathbf{C}^{(\mathtt{wspl})}_{m}$ has m deterministic input nodes $q_0, \dots, q_{m-1}$, $2m$ probabilistic input nodes $r_{0,s}, \dots, r_{m-1,s}$, $s = 0, 1$ and $4m$ output nodes $t_{i,j}$, $i = 0, 1, \dots, m-1$, $j \in \{0, 1, 2, 3\}$. The output nodes form four output blocks: $t_{0,j}, \dots, t_{m-1,j}$, $j = 0, 1, 2, 3$. Assume that the deterministic input of C_w is β_i at node q_i, and its probabilistic input is $\gamma_{i,s}$ at node $r_{i,s}$, for $i = 0, 1, \dots, m-1$ and $s = 0, 1$. Then the output of C_w at the output node $t_{i,2s+j}$ is $\gamma_{i,s} + j\beta_i$ for all $i = 0, 1, \dots, m-1$, and $s, j \in \{0, 1\}$. This completes the definition of $C_w = \mathbf{C}^{(\mathtt{wspl})}_{m}$. We define $\mathbf{C}^{(\mathtt{spl})}_{m,G}$ by $\mathbf{C}^{(\mathtt{spl})}_{m,G} = \mathtt{pre}_G(\mathbf{C}^{(\mathtt{wspl})}_{m})$. The four output block of $C = \mathbf{C}^{(\mathtt{spl})}_{m,G}$ will be denoted by $T_0^{(C)}, T_1^{(C)}, T_2^{(C)}, T_3^{(C)}$, where $T_{2s+j}^{(C)}$ is the output block defined by the output values $\gamma_{i,s} + j\beta_i$, $s, j \in \{0, 1\}$, $i \in \{0, 1, \dots, m-1\}$. The output block polynomial corresponding to the output block $T_k^{(C)}$ will be denoted by $Q_k^{(C)}$ for $k = 0, 1, 2, 3$. $\square$

The following Lemma describes the connections between the block input and the block outputs of the circuit $\mathbf{C}^{(\mathtt{spl})}_{m,G}$, and also says that the two ways, as the block input splits into two parts by the the circuit, are independent. The proof of the lemma is an immediate consequence of the definition.

Lemma 25 *If $C = \mathbf{C}^{\mathtt{spl}}_{m,G}$, $G = \langle V, E\rangle$, then $P_0^{(C)} = Q_0^{(C)} + Q_1^{(C)} = Q_2^{(C)} + Q_3^{(C)}$. Assume that $\wp_i$ are the deterministic input nodes of C, and $\mathcal{G}_e$, $e \in E$, $r_{i,s}$, $i \in \{0, 1, \dots, m-1\}$, $s \in \{0, 1\}$, are the probabilisitic input nodes of C. Suppose further that the deterministic input $\delta_0, \dots, \delta_{m-1}$ on the nodes $\wp_0, \dots, \wp_{m-1}$ is fixed. Then, for each $j = 0, 1, 2, 3$, $\bar{Q}_j^{(C)}$, has uniform distribution on F, where we get $\bar{Q}_j^{(C)}$ from the polynomial $Q_j^{(C)}$ by substituting δ_i for $z_{\wp_i}$, $i \in \{0, 1, \dots, m-1\}$, γ_e for $z_{\mathcal{G}_e}$, $e \in E$, and $\gamma_{i,s}$ for $z_{q_{i,s}}$, $i = 0, 1, \dots, m-1$, $s \in \{0, 1\}$, where the random variables γ_e, $e \in E$, $\gamma_{i,s}$, $i = 0, 1, \dots, m-1$, $s \in \{0, 1\}$ are mutually independent and each has uniform distribution on F. Moreover the random variable $\bar{Q}_0^{(C)}$ and $\bar{Q}_2^{(C)}$ are independent.*

Definition. Let $C = \mathbf{C}^{(\mathtt{spl})}_{m,G}$. The system of equation consisting of the equations $P_0^{(C)} = Q_0^{(C)} + Q_1^{(C)} = Q_2^{(C)} + Q_3^{(C)}$ will be called the canonical system of the circuit $\mathbf{C}^{(\mathtt{spl})}_{m,G}$. $\square$

Lemma 26 *Assume that d is a positive integer, $\alpha > 0$, $\varepsilon > 0$ is sufficiently small, m is a sufficiently large positive integer, $C = \mathbf{C}^{(\mathtt{spl})}_{m,G}$ is a splitting circuit where G is a $(d, 1+\alpha)$-expander, $A \subseteq \mathtt{set}(C)$, $|A| \le \varepsilon m$, $\kappa_0, \kappa_1 \in F$ and Γ is the system of equation consisting of the canonical system of $\mathbf{C}^{(\mathtt{spl})}_{m,G}$ and the equations $Q_0^{(C)} = \kappa_0$, $Q_2^{(C)} = \kappa_1$. Then the circuit $\mathbf{C}^{(\mathtt{spl})}_{m,G}$ is A, Γ-percolative.*

Proof. The proof is very similar to the proof of Lemma 13 *Q.E.D.*(Lemma 26)

The following deterministic block F-circuit is of type $((m,1),1,1)$, that is, unlike all of the other circuit considered so far, its only output "block" is of length 1. This circuit will be used only in the construction of the multiplication circuit.

Definition. *Total sum circuit* $\mathbf{C}^{(\mathbf{S})}_{m,G}$. We define a probabilistic F-circuit $\mathbf{C}^{(\mathbf{S})}_{m,G}$ of type $((m,1),1,1)$, where $G=\langle\{0,1,\ldots,m-1\},E\rangle$ is a $(d,1+\alpha)$ expander graph, and m is a power of 2. First we define a deterministic F-circuit $\mathbf{D}^{(\mathbf{S})}_{m}$ of type $((m,1),1,1)$ and then we define $\mathbf{C}^{(\mathbf{S})}_{m,G}$ by $\mathbf{C}^{(\mathbf{S})}_{m,G}=\mathtt{pre}_G(\mathbf{D}^{(\mathbf{S})}_{m})$.

If the input of $\mathbf{D}^{(\mathbf{S})}_{m}$ is the sequence $\delta_0,\ldots,\delta_{m-1}$, $\delta_i\in F$, then the output is $\sum_{i=0}^{m-1}\delta_i$. A circuit with this property can be constructed in many different ways. Our specific choice, which is important to guarantee that the circuit is percolative in some sense, is the following. Let T be a binary tree of depth $\log_2 m$ with $2m-1$ elements, that we consider now as a partially ordered set, where the root is the smallest element and the leaves are the maximal elements. The leaves are the input nodes. Each other node is a "+" gate which computes the sum of its two successors. The root is the output node. This completes the definition of $\mathbf{D}^{(\mathbf{S})}_{m}$ and as we have indicated earlier $\mathbf{C}^{(\mathbf{S})}_{m,G}=\mathtt{pre}_G(\mathbf{D}^{(\mathbf{S})}_{m})$. $\Box$

Lemma 27 *Assume that d,α, the parameters of the expander graph G are fixed. Then for all $\varepsilon>0$ there exists an $\varepsilon'>0$ so that if m is a sufficiently large power of two, $C=\mathbf{C}^{(\mathbf{S})}_{m,G}$ is a total sum circuit then the following holds. Assume that A is a random subset of $\mathtt{set}(C)$ so that for all $x\in\mathtt{set}(C)$, $\mathtt{prob}(x\in A)\le\varepsilon'$ and the events $x\in A$, for $x\in\mathtt{set}(C)$ are mutually independent, and Γ is the system consisting of the only equation $Q_0^{(C)}=P_0^{(C)}$. Then with a probability of at least $1-\varepsilon$ the circuit $\mathbf{C}^{(\mathbf{S})}_{m,G}$ is A,Γ-percolative.*

Proof. By the definition of the circuit $\mathbf{C}_{m,G}(\mathbf{S})$, it has two parts, a copying circuit $C_1=\mathbf{C}^{(=)}_{m,G}$ whose deterministic input is the deterministic input of C, and a circuit $C_2=\mathbf{D}^{(\mathbf{S})}_{m}$. C_2 does not have probabilistic input nodes. It has the structure of a binary tree whose leaves are its deterministic input nodes, they are identified with the output nodes of C_1. The other nodes are all + gates. The output node of C_2 is the root of the tree, and it is also the output node of C.

Γ_1 will denote the equation $P_0=Q_0$ with respect to circuit C_1. Let $A_1=A\cap\mathtt{set}(C_1)$. We assume that $\varepsilon'>0$ is sufficiently small with respect to ε. The expected size of the set A_1 is $\varepsilon'|\mathtt{set}(C_1)|\le\varepsilon'(1+2d)m\le\varepsilon'3md$. According to Chernoff's inequality, with a probability of at least $1-\frac{\varepsilon}{2}$, we have $|A_1|\le 6d\varepsilon'm$.

Therefore Lemma 13 implies that if $\varepsilon'>0$ is sufficiently small then the copying circuit C_1 is A_1,Γ_1-percolative. Let $\mathcal{P}_1$ be the event that C_1 is A_1,Γ_1-percolative, and if $\mathcal{P}_1$ holds let $X_1\subseteq\mathtt{detin}(C_1)\cup\mathtt{outset}(C_1)$ whose existence is guaranteed by the definition of percolativity.

Let $A_2=A\cap C_2$. Since $\mathcal{P}_1\ge 1-\frac{\varepsilon}{2}$, Lemma 17 implies that with a probability of at least $1-\varepsilon$ with respect to the randomization of A the following condition is satisfied:

(54) *$\mathcal{P}_1$ holds and there exists a path in the tree of C_2, whose every point is in $\mathtt{set}(C_2)\backslash A$, which connects a point x of $X_1\cap\mathtt{outset}(C_1)$ with the output node (that is, the root of the tree).*

If condition (54) is satisfied then x_A will denote a point $X_1 \cap \mathtt{outset}(C_1)$ with the described property.

We claim that if condition (54) is satisfied by the random choice of the set A then the circuit C is A, Γ-percolative. We define the set X whose existence is required by the definition on percolativity by $X = (X_1 \cap \mathtt{detin}(C)) \cup \mathtt{outset}(C_2)$. Assume that $\sigma \in \mathtt{func}(\mathtt{inset}(C), F)$, $\kappa \in \mathtt{func}(X, F)$ and $\sigma^{(\mathtt{io})} \natural \kappa$ satisfies Γ. We have to define the function $\rho \in \mathtt{func}(\mathtt{inset}(C), F)$ whose existence is also required by the definition of percolativity.

Let κ_1 be in $\mathtt{func}(X_1, F)$ defined in the following way. κ_1 restricted to $X_1 \cap \mathtt{detin}(C_1) = X \cap \mathtt{detin}(C)$ is identical to κ restricted to the same set. Now we define the restriction of κ_1 to $X_1 \cap \mathtt{outset}(C_1)$. On the set $X_1 \cap \mathtt{outset}(C_1) \backslash \{x_A\}$, κ_1 is identical to $\chi_\sigma^{(C)} = \chi_\sigma^{(C_1)}$, on x_A it is defined in the unique way which ensures that

$$\sum_{x \in \mathtt{detin}(C_1)} (\sigma \natural \kappa)(x) = \sum_{y \in \mathtt{outset}(C_1)} (\chi_\sigma^{(C_1)} \natural \kappa)(y)$$

Since C_1 is A_1, Γ_1 percolative, there exists a function $\rho_1 \in \mathtt{func}(\mathtt{inset}(C_1), F)$ such that $\rho_1^{(\mathtt{io})} = \sigma^{(\mathtt{io})} \natural \kappa_1$ and for all $a \in \mathtt{full}(A)_1$ we have $\chi_\sigma^{(C_1)}(a) = \chi_{\rho_1}^{(C_1)}(a)$.

We define the function ρ required by the A, Γ-percolativity of the circuit C by $\rho = \rho_1$. Since $\mathtt{inset}(C) = \mathtt{inset}(C_1)$ we have $\rho \in \mathtt{func}(\mathtt{inset}(C), F)$. We have to show that the function ρ meets the requirements of the definition of percolativity.

Assume that u is the unique output node of C. $\rho^{(\mathtt{io})}$ and $\sigma^{(io)} \natural \kappa$ are identical on $\mathtt{inset}(C) = \mathtt{inset}(C_1)$ because σ and ρ_1 are identical on the same set. Assume that u is the unique output node of C. Clearly $\rho(u) = \sum_{x \in \mathtt{detin}(C)} \rho(x) = \sum_{x \in \mathtt{detin}(C)} \rho_1(x) = \sum_{x \in \mathtt{outset}(C_1)} \rho_1(x)$. Since $\rho^{(\mathtt{io})}$ satisfies Γ, by the definition $\kappa_1(x_A)$ this last sum is $\sigma^{(\mathtt{io})}(u)$.

Finally we have to show that $\chi_\sigma^{(C)}(a) = \chi_\rho(a)$ for all $a \in \mathtt{full}(A)$. If $a \in A$ this follows from $\rho = \rho_1$ and form the corresponding condition for the percolativity of C_1. The fact that κ_1 differs only at x_A from χ_σ, implies that in $\mathtt{set}(C_2)$, the function $\chi_\rho^{(C)}$ will differ from $\chi_\sigma^{(C)}$ on only those points which are below x_A, that is, which are on the path leading from x_A to the root of the tree. However this path, by its definition, does not contain any points from $\mathtt{full}(A)$. (Here we used the fact that if a branch of the tree does not contain any point from A then it cannot contain any points from $\mathtt{full}(A)$.) *Q.E.D.*(Lemma 27)

Definition. *Reverse-sum circuit* $\mathbf{C}_{m,G}^{(\mathbf{S}^{-1})}$. We define a probabilistic F-circuit, the reverse-sum circuit $\mathbf{C}_{m,G}^{(\mathbf{S}^{-1})}$, of type $((1, m), 1, 1)$, where $G = \langle \{0, 1, \ldots, m-1\}, E \rangle$ is a $(d, 1+\alpha)$ expander graph, and m is a power of 2. First we define a deterministic F-circuit $\mathbf{D}_m^{(\mathbf{S}^{-1})}$ of type $((m, 1), 1, 1)$ and then we define $\mathbf{C}_{m,G}^{(\mathbf{S}^{-1})}$ by $\mathbf{C}_{m,G}^{(\mathbf{S}^{-1})} = \mathtt{post}_G(\mathbf{D}_m^{(\mathbf{S}^{-1})})$.

The definition of $C = \mathbf{D}_m^{(\mathbf{S}^{-1})}$ is the following. We want to define the circuit in a way that if the single input is δ and the output is $\nu_0, \ldots \nu_{m-1}$ then $\delta = \sum_{i=0}^{m-1} \nu_i$. Let T be a binary tree of depth $\log_2 m$ with $2m - 1$ elements, so that each element which is not a leaf, has a left successor and a right successor. We now consider the partial ordering on the tree where the root is the largest elements and the leaves are the minimal elements. The set of nodes of C will be the union of two sets, (a) the set of nodes of the tree T and (b) a set W of probabilistic input nodes so that $W \cap T = \emptyset$ and $|W| = m - 1$. The root of the tree is the deterministic input node, the leaves of the tree are the output nodes. We fix a one-to-one map g which maps the set of all non-leaf nodes of the tree onto W. The random input element of F at the input node $x \in W$ will be denoted by $\gamma(x)$.

The circuit computes a boolean value τ_t for each $t \in T$. The values of τ_t on the leaves t will be the output. We define τ_t for $t \in L_i$ by recursion on i, where L_i is the ith level of the tree. Assume that δ is the input of the circuit. If $i = 0$, $t \in L_0$, then t is the root of the tree and $\tau_t = \delta$. Assume that τ_r has been already defined for all $r \in L_{i-1}$ and let $t \in L_i$. Suppose that $s \in L_{i-1}$ so that t is a successor of s. If t is a left successor then $\tau_t = \gamma(g(s))$. If t is a right successor then $\tau_t = \tau_s - \gamma(g(s))$. Using the definition of τ_t we can prove by induction on i that $\sum_{t\in L_i} \tau_t = \delta$. Therefore if $\nu_0, \ldots, \nu_{m-1}$ are the values τ_t, $t \in L_{\log_2 m}$ is some order, then we have $\delta = \sum_{j=0}^{m-1} \nu_j$. This completes the definition of the circuit $\mathbf{C}_m^{(\mathbf{S}^{-1})}$ and as we have already indicated earlier $\mathbf{C}_{m,G}^{(\mathbf{S}^{-1})} = \mathtt{post}_G(\mathbf{D}_m^{(\mathbf{S}^{-1})})$ $\Box$

Lemma 28 *Assume that d, α, the parameters of the expander graph G are fixed. Then for all $\varepsilon > 0$ there exists an $\varepsilon' > 0$ so that if m is a sufficiently large power of two, $C = \mathbf{C}_m^{(\mathbf{S}^{-1})}$ is a reverse sum circuit then the following holds. Assume that A is a random subset of $\mathtt{set}(C)$ so that for all $x \in \mathtt{set}(C)$, $\mathtt{prob}(x \in A) \le \varepsilon'$ and the events $x \in A$, for $x \in \mathtt{set}(C)$ are mutually independent, and Γ is the system consisting of the only equation $Q_0^{(C)} = P_0^{(C)}$. Then with a probability of at least $1 - \varepsilon$ the circuit $\mathbf{C}_m^{(\mathbf{S}^{-1})}$ is A, Γ-percolative.*

Proof. The proof is similar to the proof of Lemma 27. By the definition of the circuit $\mathbf{C}_{m,G}(\mathbf{S}^{-1})$, it has two parts, a copying circuit $C_1 = \mathbf{C}_{m,G}^{(=)}$ whose output is the output of C, and a circuit $C_2 = \mathbf{D}_m^{(\mathbf{S}^{-1})}$. We will use the notation about $\mathbf{D}_m^{\mathbf{S}^{-1}}$ given in its definition. $\mathtt{set}(C_2)$ is the union of two disjoint set T and W. T is a binary tree with respect to $\le_{C_2}$ whose root is its only deterministic input node an it is also the only deterministic input node of C. W is the set of probabilistic input nodes, for each nonleaf $t \in T$, $g(t)$ is a probabilistic input node with the role described in the definition of $\mathbf{D}_m^{(\mathbf{S}^{-1})}$. The leaves of T are the output nodes of C_2, and they are identified with the deterministic input nodes of C_1.

We will need the following property of the circuit C_2. L will denote the set of all leaves of T.

(55) *Assume that $\beta \in \mathtt{func}(\mathtt{inset}(C), F)$, B is a branch of T and $v \in B \cap L$, $a, b \in F$, with the property that*

$$a = b + \sum\{\chi_\beta^{C_2}(x) \mid x \ne v \wedge x \in L\}$$

Then there exists a $\gamma \in \mathtt{func}(\mathtt{inset}(C), F)$ such that (i) $\gamma(g(t)) \ne \beta(g(t))$ implies $t \in B$, and (ii) $\gamma(t_0) = a \wedge \gamma(v) = b$.

We can prove this statement by recursively choosing the values of the elements $\gamma((g(t)))$ as we go down with the element t on the branch B starting at the root t_0. We have to choose each $\gamma(g(t))$, $t \in B$ in a way that if t is in $L_i \cap B$, where L_i is the ith level of T then

$$a = \chi_\gamma^{(C_2)}(t) + \sum\{\chi_\beta^{C_2}(x) \mid x \ne t \wedge x \in L_i\}$$

Q.E.D.(Lemma 28)

According to the definition of $\mathbf{D}_m^{\mathbf{S}^{-1}}$ such a choice of $\gamma(g(t))$ is always possible. Using the fact that C_2 satisfies (55) we can prove Lemma 28 in the same way as we have proved Lemma 27, but in this case because of the different ordering of the tree we need Corollary 6 of Lemma 17. *Q.E.D.*(Lemma 28)

We define now a block F-circuit C which performs multiplication, however in a way that the value of the product may leak to an ε-random adversary with a nonnegligible probability. Because of this, the circuit will be called the leaky product circuit. In spite of this defect, it will have a role in the construction of a block F-circuit which performs multiplication without such a leak. The circuit will have three input blocks and it will compute the polynomial $x_0(x_1+x_2)$. The only reason why we don't use a circuit with two input blocks and computing the polynomial x_0x_1 is, that the the construction of the leak-proof multiplication circuit is easier to formulate if we have the mentioned three input blocks.

The circuit $C = \mathbf{C}_{m,G}^{(\mathtt{leaky}\times)}$, to be defined below, will do the following. It has three input blocks. Assume that the block inputs, (that is, the sums of the deterministic input values) are a_0, a_1, a_2. First C computes a_0, a_1, a_2 using the subcircuits C_0, C_1, C_2. After that it computes $a_0(a_1 + a_2)$, and finally, using a reverse sum circuit C_3, it produces its single output block so that the block output, that is the sum of the output values is $a_0(a_1 + a_2)$.

Definition. We define a deterministic block F-circuit $C = \mathbf{C}_{m,G}^{(\mathtt{leaky}\times)}$ of type $((m,m),3,1)$. We will call this circuit the leaky product circuit. For the construction of C let C_0, C_1, C_2 be copies of the total sum circuit $\mathbf{C}_{m,G}^{(\mathbf{S})}$ and let C_3 be a copy of the reverse sum circuit $\mathbf{C}_{m,G}^{(\mathbf{S}^{-1})}$ so that the sets $\mathtt{set}(C_i)$, $i = 0,1,2,3$ are pairwise disjoint. $\mathtt{set}(C)$ will be $\{u\} \cup \bigcup_{i=0}^{3} \mathtt{set}(C_i)$ such that $u \notin \bigcup_{i=0}^{3} \mathtt{set}(C_i)$ and u is a "+" gate which adds the output of C_1 and C_2. All of the gates of C_i, $i = 0,1,2,3$ will perform the same operations as in their original role in C_i, with the exception of w the deterministic input node of C_3. The node w will be now a "$\times$" gate which computes the product of the values which are computed at the output node of C_0 and at the node u. The deterministic input nodes of C are the deterministic input nodes of C_0, C_1, and C_2. The probabilistic input nodes of C are the probabilistic input nodes C_3. The output nodes of C are the output nodes of C_3. Consequently if $\delta^{(0)} = \langle \delta_{0,0}, \ldots, \delta_{m-1,0} \rangle$, $\delta^{(1)} = \langle \delta_{0,1}, \ldots, \delta_{m-1,1} \rangle$, and $\delta^{(2)} = \langle \delta_{0,2}, \ldots, \delta_{m-1,2} \rangle$ is the deterministic input of C so that $\delta^{(j)}$ is given at the input nodes of C_j, for $j = 0,1,2$, and ν is the output of C, then $\nu = (\sum_{i=0}^{m-1} \delta_{i,0})[(\sum_{i=0}^{m-1} \delta_{i,1}) + (\sum_{i=0}^{m-1} \delta_{i,1})]$. The system of equations consisting of the only equation $Q_0^{(C)} = P_0^{(C)}(Q_1^{(C)} + Q_2^{(C)})$ will be called the canonical system of the circuit $C = \mathbf{C}_{m,G}^{(\mathtt{leaky}\times)}$ $\square$

Lemma 29 *For all $\varepsilon > 0$ there exists an $\varepsilon' > 0$ so that if m is a sufficiently large power of two, $C = \mathbf{C}_{m,G}^{(\mathtt{leaky}\times)}$ is a leaky product circuit then the following holds. Assume that A is a random subset of* $\mathtt{set}(C)$ *so that for all $x \in \mathtt{set}(C)$, $\mathtt{prob}(x \in A) \le \varepsilon'$ and the events $x \in A$, for $x \in \mathtt{set}(C)$ are mutually independent, and Γ is the canonical system of $\mathbf{C}_{m,G}^{(\mathtt{leaky}\times)}$. Then, with a probability of at least $1 - \varepsilon$, the circuit $\mathbf{C}_{m,G}^{(\mathtt{leaky}\times)}$ is A,Γ-percolative.*

Proof. The lemma is a consequence of Lemma 27 and Lemma 28. *Q.E.D.*(Lemma 29)

7 Circuit for Multiplication

In this section we assume that two absolute constants d and $\alpha > 0$ are given so that for each m we can construct in an efficient way an $(d, 1+\alpha)$-expander on the set of vertices $\{0, 1, \ldots, m-1\}$. We assume that such a method of construction and the corresponding graph $G_m = \langle \{0, 1, \ldots, m-1\}, E_m \rangle$ is fixed for all m. Therefore when we are speaking about the block F-circuits, $\mathbf{C}^{(=)}_{m,G}, \mathbf{C}^{(+)}_{m,G}, \mathbf{C}^{(\mathtt{spl})}_{m,G}$ etc., we will always assume that $G = G_m$ and because of this the subscript G may be omitted, that is, we may write $\mathbf{C}^{(=)}_{m}, \mathbf{C}^{(+)}_{m}, \mathbf{C}^{(\mathtt{spl})}_{m}$ etc.

We will define an $(\theta, \varepsilon_1, p)$-cylindrical block F-circuit $\mathbf{C}^{(\times)}_{m} = \mathbf{C}^{(\times)}_{m,G}$ which computes the function $x_0 x_1$, where θ, ε_1 and p are chosen according to the standard assumptions described in condition (53). For using this circuit in the proof of Theorem 7 we will need a slightly modified version of this circuit a probabilistic block F-circuit of type $((m, m,)2, 2)$, that is a circuit which has two input blocks and two output blocks each of them of size m, and computes the functions $x_0, x_0 x_1$, as required in Lemma 3. In other words the first block output is identical to the first block input, and the second block output is the product of the two block inputs. This circuit will be put together from a copying circuit of $\mathbf{C}^{(=)}_{m,G}$ and a product circuit $\mathbf{C}^{(\times)}_{m,G}$, that we will define later in this section. The result that we will need in the proof of Theorem 7 is the following lemma.

Lemma 30 *For all positive integer m there exists a probabilistic block F-circuit $\mathbf{C}^{(\times,)}_{m,G}$ of type $((m, m), 2, 2)$ which computes the functions $x_0, x_1 x_2$ so that under the standard assumptions described in* (53) the circuit $\mathbf{C}^{(\times,)}_{m,G}$ is $(\theta, \varepsilon_1, p)$-cylindrical. Moreover the directed graph defining the circuit is of polynomial size in m, and can be computed by a Turing machine, together with its labelings, in time polynomial in m.

As a first step we define a block F-circuit $\mathbf{C}^{(\times,+)}_{m} = \mathbf{C}^{(\times,+)}_{m,G}$ with three input blocks which computes the function $x_0(x_1 + x_2)$. We may get $\mathbf{C}^{(\times)}_{m}$ from $\mathbf{C}^{(\times,+)}_{m}$, by replacing the input nodes where x_2 appears by gates which produce always 0s independently of the input.

We define $\mathbf{C}^{(\times,+)}_{m}$ as the composition of different circuits of the types we have defined in the previous section, for example, circuits $\mathbf{C}^{(=)}_{m}, \mathbf{C}^{(+)}_{m}, \mathbf{C}^{(\mathtt{spl})}_{m}$ etc. Each of these circuits will occur in many different copies. To do this, first we define a general abstract process of composition block F-circuits, which are arranged on a partially ordered set, and then we will apply it for the construction of the circuit $\mathbf{C}^{(+,\times)}_{m}$.

7.1 Composition of F-circuits given on a partially ordered set

Definition. If P is a partially ordered set and $a, b \in P$, we will say that b covers a or a precedes b if $a < b$ and there exists no $c \in P$ with $a < c < b$. In this situation we will write $a \prec b$.$\square$

The following definition describes a construction of a block F-circuit from a set Q of given F-circuits. Some of the output blocks of a circuit $C \in F$ may be identified with input blocks of other circuits $C_0, C_1, \ldots \in F$, with the intuitive meaning that first C performs a computation and then gives its output in the form of blocks to the circuits $C_0, C_1, \ldots$ as input blocks. The set of circuits in Q has a partial ordering and each circuit $C \in Q$ may pass information to a circuit $D \in Q$ only if C covers Q.

Definition. 1. We say that the pair $Q = \langle \Phi, \leq_\Phi \rangle$, is a compatible F-circuit family, if Φ is a set of probabilistic block F-circuits, $\leq_\Phi$ is a partial ordering of the set Φ, and the following two conditions are satisfied:

(56) *For all $C, D \in \Phi$, if $\mathtt{set}(C) \cap \mathtt{set}(D) \neq \emptyset$, then either C covers D, or D covers C with respect to $\leq_\Phi$.*

(57) *For all $C, D \in \Phi$, D covers C implies that there exists a nonempty subset X of $\mathtt{block_{in}}(C)$ so that each element of X is also an element of $\mathtt{block_{out}}(D)$ and $\mathtt{set}(C) \cap \mathtt{set}(D) = \bigcup X$.*

2. Assume that $Q = \langle \Phi, \leq_\Phi \rangle$, is a compatible F-circuit family. We define a probabilistic block F-circuit $C^{(Q)}$, that we will call the composition of the family Q, in the following way. In the definition of an F-circuit C we used a directed graph denoted by $\mathtt{graph}(C)$. $\mathtt{graph}(C^{(Q)})$ will be the union of the graphs $\mathtt{graph}(C)$ for all $C \in \Phi$. The union is taken in the sense that the set of vertices of the union is the union of the sets of vertices of the graphs $\mathtt{graph}(C)$, $C \in \Phi$, and the set of edges of the union is the union of the sets of edges of the graphs $\mathtt{graph}(C)$, $C \in \Phi$. Note that the definition of a compatible family implies, that if the head of a directed edge of $\mathtt{graph}(C^{(Q)})$ is a vertex of $\mathtt{graph}(C)$, while its tail is a vertex of $\mathtt{graph}(D)$ for some $C, D \in \Phi$, then $C \leq D$. This implies that $\mathtt{graph}(C^{(Q)})$ is acyclic. The labeling of the nodes of $\mathtt{graph}(C^{(Q)})$ which defines the gates of the circuit will be the common extension of the labelings of the graphs $\mathtt{graph}(C)$, $C \in \Phi$. Since the input nodes have no labels in the circuits $C \in \Phi$, the definition of a compatible family implies, that such a common extension always exists.

We have to define the set of deterministic and probabilistic input nodes of $C^{(Q)}$. A node x of $\mathtt{graph}(C^{(Q)})$ is a deterministic input node of $C^{(Q)}$, if x has indegree 0 in $C^{(Q)}$, and there exists a $C \in \Phi$, so that x is a deterministic input node of Φ. All of the other nodes of $\mathtt{graph}(C^{(Q)})$ with indegree 0 will be probabilistic input nodes. The nodes of $\mathtt{graph}(C^{(Q)})$ with outdegree 0 will be the output nodes of the circuit $C^{(Q)}$.

This definition implies that if x is a deterministic input node of $C^{(Q)}$ then there is a unique $C \in \Phi$ so that x is a deterministic input node of C, and if x is in the input block I of C then all of the other elements of I are also deterministic input nodes of $C^{(Q)}$. Therefore we may define the input block of $C^{(Q)}$ in the following way: the input nodes x, y of $C^{(Q)}$ are in the same input block if there exists a $C \in \Phi$ so that x and y are in the same input block of C.

We define the output blocks of $C^{(Q)}$ in a similar way. The output nodes x, y of $C^{(Q)}$ are in the same output block if there exists a $C \in \Phi$ so that x and y are in the same output block of C.

This definition implies that for each input/output block Y of $C^{(Q)}$ there exists exactly one $C \in \Phi$ such that Y is an input/output block of C. □

Remark. 1. If an input is given for $C^{(Q)}$, we can evaluate $C^{(Q)}$, by recursively evaluating all of the circuits C, $C \in \Phi$, going downward on the partial ordering Φ, starting with a maximal element $C \in \Phi$, and always evaluating a circuit which is maximal among the remaining elements of Φ. The definition of a compatible family implies that all of the inputs of such a maximal circuit has been already computed.

2. We included the partial ordering $\leq_\Phi$ in the definition of a compatible family, only for the sake of clarity. If only the set Φ is known from a compatible family, the partial ordering can be easily defined from it. Another reason for the inclusion of the partial ordering in the definition is, that in our most important example for a compatible family, we will start the construction of the family by describing the partial ordering $\leq_Q$ as an abstract partially ordered set. $\square$

7.2 Percolativity and composition of F-circuits

The notion of percolativity will be very important in the proof of the $(\theta, \varepsilon_1, p)$-cylindricity of the circuit $\mathbf{C}_{m,G}^{(\times,+)}$ to be defined later, as a composition of a family of F-circuit. In this subsection we will consider for an arbitrary family of circuits $Q = \langle \Phi, \leq_\Phi \rangle$, what are the consequences of percolativity of some elements of Φ to the composition $C^{(Q)}$. Later we will apply these results to the circuit $\mathbf{C}_{m,G}^{(\times,+)}$.

Recall that an A, Γ-percolative circuit C, where C satisfies the system of equations Γ and $A \subseteq \texttt{set}(C)$, by definition, has the following property. Suppose that σ is an evaluation of both the deterministic and probabilistic inputs of C and $\sigma^{(\texttt{io})}$ is the restriction of the evaluation function χ_σ to the set consisting of the deterministic input nodes and the output nodes. We change the function $\sigma^{(\texttt{io})}$ on a relatively large set X so that the new function that we get, $\sigma^{(\texttt{io})} \natural \kappa$, also satisfies Γ. Then there exists an evaluation ρ of all of the inputs of C, both deterministic and probabilistic, so that the evaluation function χ_ρ restricted to the set of deterministic input nodes and output nodes is identical to $\sigma^{(\texttt{io})} \natural \kappa$, and on the elements of the set A, the functions χ_ρ and χ_σ are identical.

Motivation. For the construction of the product circuit we will need the following. Assume that a compatible family of circuits $Q = \langle \Phi, \leq_\Phi \rangle$ and a subset $A \subseteq C^{(Q)}$ is given. Together with each circuits $C \in \Phi$, a system of equation G_C is also given so that C satisfies G_C. Let σ be an evaluation of all of the input nodes of $C^{(Q)}$ with values in F. λ will be a function defined on the set of all input and output blocks of the circuits in Φ with values in F. (If a block is the output block of a $C_1 \in \Phi$ and the input block of a $C_2 \in \Phi$ then λ takes a single value on it.) For each $C \in \Phi$, λ_C will be the restriction of λ to the input/output blocks of C. We assume that λ_C satisfies the system G_C for all $C \in \Phi$.

The central question in the construction of the product circuit Φ will be the following. Does there exists an evaluation ρ of the input variables of $Q^{(C)}$ so that χ_ρ is compatible to λ in the sense that on each input/output block X of each $C \in \Phi$ the sum of the values of χ_ρ on X is $\lambda(X)$. The importance of the existence of such a ρ will be that it guarantees the existence of a solution for a system of equation that we will associate with $C^{(Q)}$. The existence of a solution, using the lemmas about systems over F, will guarantee that for certain systems, where the observation of the adversary (on the set A) defines the inhomogeneous part and the inputs are the unknowns, the number of solutions are independent of the inhomogeneous part. This can be translated into the language of probabilities and will give that from the point of view of the adversary the probabilities of the various values for the block inputs are the same.

We formulate below Lemma 31, which guarantees the existence of such an input ρ under certain conditions. We assume the existence of an evaluation σ of the input nodes of $C^{(Q)}$, so that for each $C \in \Phi$ with the property that C is not $A \cap \texttt{set}(C), G_C$-percolative, the evaluation function χ_σ restricted to $\texttt{set}(C)$ is compatible to λ restricted to the blocks of C. (That is the sum of χ_σ on each block of C is the value of λ on that block.) χ_ρ will be identical to χ_σ on the nodes of these circuits C, while for the remaining circuits $C \in \Phi$ we will construct χ_ρ by using the $A \cap \mathtt{set}(C), G_C$-percolativiy of C.

Definition. Let $Q = \langle \Phi, \leq_\Phi \rangle$ be a compatible family of circuits, and let $A \subseteq \mathtt{set}(C^{(Q)})$. Suppose that for all $C \in \Phi$, G_C is a polynomial system of equations over F so that the circuit C satisfies the system G_C. $\mathtt{perc}(\Phi, A, G_x)$ will denote the set of all $C \in \Phi$ so that C is percolative with respect to $A \cap \mathtt{set}(C)$, and G_C. (Here G_x denotes the function which assigns G_C to C for all $C \in \Phi$.) We will write $\mathtt{perc}(\Phi, A)$ for $\mathtt{perc}(\Phi, A, G_x)$ if the choice of the function G_x is clear from the context. $\square$

Definition. Assume the $Q = \langle \Phi, \leq_\Phi \rangle$ is a compatible family of F-circuits and $\sigma \in \mathtt{func}(\mathtt{inset}(C^{(Q)}), F)$. For all $C \in \Phi$ we define a function $\sigma_C \in \mathtt{func}(\mathtt{inset}(C), F)$, in the following way. We evaluate the circuit $C^{(Q)}$, with the input defined by σ. Assume that we get the evaluation function χ_σ defined on $\mathtt{set}(C^{(Q)})$. Suppose that $x \in \mathtt{inset}(C)$. We define $\sigma_C(x)$ by $\sigma_C(x) = \chi_\sigma(x)$. $\square$

Lemma 31 *Assume that $Q = \langle \Phi, \leq_\Phi \rangle$ is a compatible family of F-circuits, the systems G_C, the set A, and the functions $\sigma, \lambda, \lambda_C$ are given with the following properties:*

(58) *for all $C \in \Phi$, G_C is a polynomial system of equations over F so that the circuit C satisfies the system G_C,*

(59) $A \subseteq \mathtt{set}(C^{(Q)})$,

(60) $\sigma \in \mathtt{func}(\mathtt{inset}(C^{(Q)}), F)$,

(61) $\lambda \in \mathtt{func}(\bigcup\{\mathtt{block}_{\mathtt{io}}(C) \mid C \in \Phi\}, F)$, *and for each $C \in \Phi$, λ_C is the restriction of λ to* $\mathtt{block}_{\mathtt{io}}(C)$. *Moreover for each $C \in \Phi$. λ_C satisfies the system G_C,*

(62) *for all $C \notin \mathtt{perc}(\Phi, A, G_x)$, $\lambda_C = (\bar{\sigma}_C)^{(\mathtt{io})}$*

Then there exists a $\rho \in \mathtt{func}(\mathtt{inset}(C), F)$, so that for each $a \in A$ we have $\chi_\sigma^{(C^{(Q)})}(a) = \chi_\rho^{(C^{(Q)})}(a)$ (where $\chi_\alpha^{(C^{(Q)})}$ is the evaluation function of the circuit $C^{(Q)}$ at input α), and for each $C \in \Phi$ and $J \in \mathtt{block}_{\mathtt{in}}(C) \cup \mathtt{block}_{\mathtt{out}}(C)$, we have $\lambda(J) = \sum\{\chi_\rho^{(C^{(Q)})}(b) \mid b \in J\}$.

Proof. According to the definition of the function $\mathtt{perc}$, for each $C \in \mathtt{perc}(\Phi, A)$, C is A_C, G_C-percolative, where $A_C = A \cap \mathtt{set}(C)$, and therefore there exists a set $X_C \subseteq \mathtt{detin}(C) \cup \mathtt{outset}(C)$, so that conditions (39) and (40) of the definition of percolativity are satisfied with $A{:=}A_C$, $X{:=}X_C$.

Proposition 12 *Let W be the set of all $\langle C, J \rangle$ so that: $C \in \mathtt{perc}(\Phi, A)$, $J \in \mathtt{block}_{\mathtt{io}(C)}$, and $\lambda(J) \neq \bar{\sigma}_C^{(\mathtt{io})}(J)$*

Then there exists a binary function Z so that $Z(C, J)$ is defined iff $\langle C, J \rangle \in W$ with the property that for all $\langle C, J \rangle \in W$ we have:

(a) $\emptyset \neq Z(C, J) \subseteq X_C$, and

(b) if there exists a $D \in \Phi$ so that $\langle D, J \rangle \in W$, then $Z(C, J) = Z(D, J)$.

We define a set $Z(C,J)$ for all $\langle C,J\rangle \in W$. Assumption (62) of the lemma and the definition of the set W implies that for all $\langle C,J\rangle \in W$ we have that $C \in \mathtt{perc}(\Phi,A)$. Assume first that a pair $\langle C,J\rangle \in W$ is given so that $J \in \mathtt{block_{io}}(D)$ for some $D \in \Phi$, $D \neq C$. With the previous argument we get that $D \in \mathtt{perc}(\Phi,A)$ as well. $C, D \in \mathtt{perc}(\Phi,A)$ implies that X_C, X_D satisfy condition (39) from the definition of percolativeness, therefore we have $|J \cap X_C| > \frac{1}{2}|J|$ and $|J \cap X_D| > \frac{1}{2}|J|$, and consequently $J \cap X_C \cap X_D \neq \emptyset$. For such a pair $C, J \in W$ we define the set $Z(C,J)$ by $Z(C,J) = J \cap X_C \cap X_D$. Clearly condition (a) of the lemma is satisfied by $Z(J,C)$. The circuit D, with the described properties, is uniquely determined by C and J, and so for such a D we have $Z(C,J) = Z(D,J)$, that is, condition (b) is satisfied as well.

Assume now that $\langle C,J\rangle \in W$ and there exists no $D \in \Phi\backslash\{C\}$ so that $J \in \mathtt{block_{io}}(D)$. In this case let $Z(C,J) = X_C \cap J$. Since $|J \cap X_C| > \frac{1}{2}|J|$ condition (a) is satisfied. Condition (b) follows from the assumption that there exists no $D \neq C$ with $\langle D,J\rangle \in W$. *Q.E.D.*(Proposition 12)

Proposition 13 *For each $C \in \Phi$ let $\bar{X}_C = \bigcup\{Z(C,J) \mid \langle C,J\rangle \in W\}$, where the set W and the function Z are defined in Proposition 12. Then for all $C \in \Phi$, there exists a function $\kappa_C \in \mathtt{func}(X_C, F)$ with the following properties.*

(63) *For all $x \in X_C\backslash\bar{X}_C$, $\kappa_C(x) = \sigma_C(x)$*

(64) *for all $\langle C,J\rangle \in W$ we have*

$$\sum\{\sigma_C(x) \mid x \in J\backslash Z(C,J)\} + \sum\{\kappa_C(x) \mid x \in Z(C,J)\} = \lambda(J)$$

(65) *if $x \in \mathtt{domain}(\kappa_C) \cap \mathtt{domain}(\kappa_D)$ for some $C, D \in \Phi$ then $\kappa_C(x) = \kappa_D(x)$.*

Proof. We define κ_C for all $x \in X_C\backslash\bar{X}_C$ by $\kappa_C(x) = \sigma_C(x)$, so condition (63) is obviously satisfied. Condition (64) in itself can be easily satisfied, since $Z(C,J) \neq \emptyset$. The value of $\kappa_C(x)$ is already given outside $Z(C,J)$, and we define it inside $Z(C,J)$, so that the equality in condition (64) holds. To ensure that condition (65) is also satisfied, we take the pairs $\langle C,J\rangle \in W$ in some order, and if one of the pairs $\langle C,J\rangle$, $\langle D,J\rangle$, e.g., $\langle C,J\rangle$, comes first, then we define κ_C on J so that it satisfies condition (64), and then define κ_D, so that it is identical to κ_C on J. By proposition 12 we have $Z(C,J) = Z(D,J)$. σ_C and σ_D are identical on the set J, since they both take on J the same values as the function $\chi_\sigma^{(C^{(Q)})}$. Therefore κ_D satisfies condition (64) with the pair $\langle D,J\rangle$. This completes the definitions of the functions κ_C, $C \in \Phi$. *Q.E.D.*(Proposition 13)

Proof of Lemma 31 continued. For the definition of ρ first we define a $\rho_C \in (\mathtt{inset}(C), F)$ for each $C \in \Phi$. (This will be compatible to our earlier notation, since we will get a ρ such that ρ_C is the restriction of $\chi_\rho^{(C^{(Q)})}$ to the set $\mathtt{inset}(C)$.) We define ρ_C by recursion on C according to the partial ordering $\leq_\Phi$. Assume that $C \in \Phi$ and ρ_D has been already defined for all $D \in \Phi$ with $D > C$. If $C \notin \mathtt{perc}(\Phi,A)$ then $\rho_C = \sigma_C$. Assume that $C \in \mathtt{perc}(\Phi,A)$ and so C is $A \cap C$ percolative. Since κ_C is defined on the set X_C, the $A \cap C$-percolativeness of C implies that there exists a $\rho_C \in \mathtt{func}(\mathtt{inset}(C), F)$ so that condition (40) of the definition of percolativeness holds with $\sigma{:=}\sigma_C, \kappa{:=}\kappa_C, \Gamma{:=}G_C$, and $\rho{:=}\rho_C$. ρ will be the unique element of $\mathtt{func}(\mathtt{inset}(C^{(Q)}), F)$, such that for each $C \in \varphi$ and $x \in \mathtt{inset}(C)$, we have $\chi_\rho^{(C^{(Q)})}(x) = \rho_C(x)$*Q.E.D.*(Lemma 31).

7.3 The definition of $\mathbf{C}_{m,G}^{(+,\times)}$

We will define a compatible family $Q = \langle \Phi, \leq_\Phi \rangle$ of block F-circuits, and their composition the block F-circuit $C^{(Q)}$ will be the circuit $\mathbf{C}_{m,G}^{(\times,+)}$.

Definition. We define a compatible family of F-circuits $Q = \langle \Phi, \leq_\Phi \rangle$. First we define a partially ordered set $\langle P, \leq_P \rangle$ and a one-to-one map φ on P so that for each $a \in P$, $\varphi(a)$ is a probabilisitic block F-circuit. The set $\{\varphi(a) \mid a \in P\}$ will be Φ, and the partial ordering on Φ will be defined by $\varphi(a) \leq \varphi(b)$ iff $a \leq_P b$.

Remark. In the following definition of $\mathbf{C}_{m,G}^{(\times,+)}$, assuming that m is a power of two, we use a parameter $\bar{m} = m^3$. We make this choice of the value of $\bar{m}$ for the sake of simplicity. Selecting $\bar{m}$ in any other way so that $\bar{m} \geq \omega(m)m^2$ and $\bar{m}$ is a power of two, would be good, as well, provided that $\omega(m)$ is a function with $\lim_{m\to\infty} \omega(m) = \infty$. A smaller value for $\bar{m}$ would make the size of the corresponding circuit also smaller. □

The definition of $\langle P, \leq_P \rangle$. Assume that $m = 2^{d_0}$, $d_0 > 1.$, $\mathbf{d} = \log_2 \bar{m} = 3d_0$. Let $\bar{m} = m^3$, $\mathbf{d} = \log_2 \bar{m} = 3d_0$. and T is a binary tree of depth $\mathbf{d}$ with levels $L_0 = \{t_0\}, L_1, \ldots, L_{\mathbf{d}}$, where $|L_i| = 2^i$. We will use two different partial ordering on T, $\leq_T$ and $\leq'$. The partial ordering $\leq_T$, where the root is the largest element and the leaves are the minimal elements, has been already defined after the definition of a binary tree.

The partial ordering $\leq'$ is defined in the following way. Assume that $a, b \in T$, $a \in L_i, b \in L_j$. For the definition of $a \leq' b$ we consider two cases separately: (a) $i, j \in \{0, 1, \ldots, \mathbf{d} - 1\}$, and (b) $\{a, b\} \cap L_{\mathbf{d}} \neq \emptyset$. If (a) holds then $a \leq' b$ iff $a \leq_T b$ and $i \equiv j \pmod 2$. If (b) holds then $a \leq' b$ iff $a \leq_T b$. Note that, with respect to the partial ordering $\leq'$, the set T has three maximal elements, namely the elements of $L_0 \cup L_1$.

We will get the partially ordered set $\langle P, \leq_P \rangle$ by extending $\langle T, \leq' \rangle$ downward. Assume that $\mathbf{a}_0, \ldots, \mathbf{a}_{\bar{m}-1}$ are the minimal elements of $\leq'$, that is, the leaves of the tree T, arranged in an arbitrary order. (E.g., we may associate a $0, 1$, sequence of length $\mathbf{d}$ with each leaf using left and right successors, and order them as integers.) We get the set P by adding $\bar{m} - 1$ new elements to the set T, namely the elements $\mathbf{b}_1, \ldots, \mathbf{b}_{\bar{m}-1}$, that is, $P = T \cup \{\mathbf{b}_1, \ldots, \mathbf{b}_{\bar{m}-1}\}$. (Note that there is no $\mathbf{b}_0$.) The partial ordering $\leq_P$ is the unique partial ordering on the set P with the following properties:

(a) For all $x, y \in T$, we have $x \leq_P y$ iff $x \leq' y$.

(b) $\mathbf{b}_1 >_P \mathbf{b}_2 >_P \ldots >_P \mathbf{b}_{\bar{m}-1}$

(c) for all $x \in P$, and for all $i = 1, \ldots, \bar{m} - 1$, $\mathbf{b}_i \leq_P x$ iff

$$x \in \mathtt{branch}(\mathbf{a}_0) \cup \bigcup_{j=1}^{i} \left(\{\mathbf{b}_i\} \cup \mathtt{branch}(\mathbf{a}_j)\right)$$

where $\mathtt{branch}(u)$ is the unique branch of the tree T containing the leaf u.

(d) for all $x \in P$, and for all $i = 1, \ldots, \bar{m} - 1$, $\mathbf{b}_i \geq_P x$ iff $x = \mathbf{b}_j$ for some $j \in \{i, \ldots, \bar{m} - 1\}$. This completes the definition of the partially ordered set $\langle P, \leq_P \rangle$. □

Definition. Instead of the map φ, first we define a map $\bar{\varphi}$ on P, so that its values are probabilistic block F-circuits so that their sets of nodes are pairwise disjoint. Then, for all pairs $a, b \in P$ with $a \prec_P b$, we will identify an input block of $\bar{\varphi}(a)$ with an output block of $\bar{\varphi}(b)$. For each $x \in P$ the circuit, that we get after these modifications from the circuit $\bar{\varphi}(x)$, will be the circuit $\varphi(x)$.

Now we define the function $\bar{\varphi}$. The sets of nodes of the various circuits $\bar{\varphi}(a)$, $a \in P$ will be disjoint. We will not repeat this requirement at each time we define a new $\bar{\varphi}(x)$, we assume that its nodes are automatically chosen with this property.

If $x \in T \backslash L_{\mathbf{d}}$, then $\bar{\varphi}(x)$ is a splitting circuit $\mathbf{C}_m^{(\texttt{spl})}$. For each $i = 0, 1, \ldots, m-1$, $\bar{\varphi}(\mathbf{a}_i)$ is a leaky product circuit: $\mathbf{C}_m^{(\texttt{leaky}\times)}$. For each $i = 1, \ldots, \bar{m}-1$, $\bar{\varphi}(\mathbf{b}_i)$ is an addition circuit $\mathbf{C}_m^{(+)}$. This completes the definition of $\bar{\varphi}$. □

Definition. *Identifications of input blocks and output blocks in $\bar{\varphi}(a)$ and $\bar{\varphi}(b)$, for $a \prec b$, $a, b \in P$.* The only maximal elements of P are $u_0 = t_0, u_1 = \mathbf{l}t_0$, and $u_2 = \mathbf{r}t_0$, where t_0 is the root of the tree T. For each $i = 0, 1, 2$, there is no $x \in P$ with $u_i \prec x$, therefore the input block of $\bar{\varphi}(u_i)$ will not be identified with anything. (These input blocks will be the input blocks of $C^{(Q)}$ as well.)

Suppose now, that $b \in L_j$ for some $j \in \{0, 1, \ldots, d-3\}$. With such a choice of b, there are exactly four elements a of P so that $a \prec_P b$, namely $r_i(b) \in L_{j+2}$, $i = 0, 1, 2, 3$. (r_i was defined at the beginning of section 5 about trees.). Assume that the output blocks of the splitting circuit $\bar{\varphi}(b)$, are $T_i^{(\texttt{spl})}$, $i = 0, 1, 2, 3$. We identify the output block $T_i^{(\texttt{spl})}$ with the single input block of $\bar{\varphi}(r_i(b))$, for $i = 0, 1, 2, 3$. (The assumption $b \in L_j$, $j \le d-3$ implies that $\bar{\varphi}(r_i(b))$ is a splitting circuit.)

Assume now that $b \in L_{\mathbf{d}-2}$. In this case the output block $T_i^{(\texttt{spl})}$ of $\bar{\varphi}(b)$ is identified with the first input block of the leaky product circuit $\bar{\varphi}(r_i(b))$ for $i = 0, 1, 2, 3$. (Recall that a leaky product circuit has three input blocks.)

Suppose that $b \in L_{\mathbf{d}-1}$. Then the output blocks $T_i^{(\texttt{spl})}$ of $\bar{\varphi}(b)$, for $i = 0, 1$, are identified with the second and third input blocks of the leaky product circuit $\bar{\varphi}(\mathbf{l}b)$. The output blocks $T_i^{(\texttt{spl})}$ of $\bar{\varphi}(b)$, for $i = 2, 3$ are identified with the second and third input blocks of the leaky product circuit $\bar{\varphi}(\mathbf{r}b)$.

Assume that $i \in \{1, 2, \ldots, \bar{m}-1\}$. The single output block of the leaky product circuit $\bar{\varphi}(\mathbf{a}_i)$ is identified with the second input block of the addition circuit $\bar{\varphi}(\mathbf{b}_i)$. The single output block of $\bar{\varphi}(\mathbf{a}_0)$ is identified with the first input block of $\bar{\varphi}(\mathbf{b}_1)$.

Let $b = \mathbf{b}_i$, for some $i = 1, 2, \ldots, \bar{m}-2$. The only output block of the addition circuit $\bar{\varphi}(\mathbf{b}_i)$ is identified with the first output block of the circuit $\bar{\varphi}(\mathbf{b}_{i+1})$.

There is no $x \in P$ with $x \prec \mathbf{b}_{\bar{m}-1}$, therefore the single output block of $\bar{\varphi}(\mathbf{b}_{\bar{m}-1})$ is not identified with anything. (This will be the outputblock of $C^{(Q)}$ as well.)

For all $a \in P$, the circuit $\varphi(a)$ is defined as the modified form of the circuit $\bar{\varphi}(a)$ after these identifications. Let $\Phi = \{\varphi(a) | a \in P\}$, and for all $x, y \in \Phi$, we will have $x \le_\Phi y$ iff there exists $a, b \in P$ with $a \le_P b$ and $x = \varphi(a)$, $y = \varphi(b)$.

This completes the definition of the family $Q = \langle \Phi, \le_\Phi \rangle$. We define the probabilistic block F-circuit by $\mathbf{C}_{m,G}^{(\times,+)} = C^{(Q)}$, that is, $\mathbf{C}_{m,G}^{(\times,+)}$ is the composition of the family Q. □

Definition. By the definition above, the circuit $\varphi(\mathbf{b}_{\bar{m}-1})$ is an addition circuit $\mathbf{C}_{m,G}^{(+)}$, whose only output block is the only output block of $\mathbf{C}_{m,G}^{(\times,+)}$. According to the definition of the circuit $\mathbf{C}_{m,G}^{(+)}$ this circuit ends with a copying circuit of the type $\mathbf{C}_{m,G}^{(=)}$. This copying circuit will be denoted by $C_{\mathtt{end}}$. Suppose that Y is an arbitrary set. Then $\mathcal{X}(Y)$ will denote the set of all output nodes of $C_{\mathtt{end}}$ which do not belong to $\mathtt{exit}(Y \cap \mathtt{set}(C_{\mathtt{end}}), C_{\mathtt{end}})$. (Recall that $\mathtt{exit}(A, C)$ is a set of output nodes of a copying circuit C, which corresponds to the connected set in Lemma 10.) $\Box$

7.4 Basic properties of $\mathbf{C}_{m,G}^{(\times,+)}$

In the remaining part of this section the family Q will always denote the family of block circuits $\langle \Phi, \leq_\Phi \rangle$, defined in the previous subsection, whose composition is the F-circuit $\mathbf{C}_{m,G}^{(\times,+)}$.

Definition. For each circuit $C \in \Phi$ the set consisting of all of the deterministic input blocks and all of the output blocks of C will be denoted by Ψ_C. We define the set Ψ by $\Psi = \bigcup\{\Psi_C \mid C \in \Phi\}$. (Note that some of the elements of Ψ can be an input block of a $C \in \Phi$ and at the same time an output block of another $C' \in \Phi$.) The set of all deterministic input blocks of a $C \in \Phi$ will be denoted by $\Psi_{C,\mathtt{in}}$ and the set of all outputblocks of C by $\Psi_{C,\mathtt{out}}$. $\Box$

Later we will use the following observation, which is an immediate consequence of the definition above.

Proposition 14 *Assume that $J_i \in \Psi_{\varphi(x_i)}$, where $x_i \in P$ for $i = 0, 1$ and $x_0 \neq x_1$. Then, $J_0 = J_1$ implies that there exists an $i \in \{0, 1\}$, so that $J_i \in \Psi_{\varphi(x_i),\mathtt{out}}$, $J_{1-i} \in \Psi_{\varphi(x_{1-i}),\mathtt{in}}$, and x_i covers x_{1-i} in the ordering $\leq_P$.*

Definition. Assume that $x \in P$. The deterministic input blocks of the circuit $\varphi(x)$ will be denoted by $I_0^{(x)}, I_1^{(x)}, \ldots$ and its output blocks by $T_0^{(x)}, T_1^{(x)}, \ldots$ in the order that was specified in the definition of the corresponding circuit type, (splitting, leaky product, or addition). Suppose that $\sigma \in \mathtt{func}(\mathtt{inset}(C^{(Q)}), F)$. We define a function $\tilde{\sigma}$ on Ψ. If $S \in \Psi_C$ for some $C \in \Phi$ then $\tilde{\sigma}(S) = \overline{(\sigma_C)}_{\mathtt{io}}(S) = \sum_{x \in S} \chi_\sigma^{(C)}(x)$. Note that the circuit C is not uniquely determined by the block S, however the value of $\tilde{\sigma}(S)$ is independent of the choice of C, since $x \in S \in \Psi_C \cap \Psi_D$ implies $\chi_\sigma^{(C)}(x) = \chi_\sigma^{(D)}(x)$. $\Box$

First we show that the circuit $\mathbf{C}_m^{(\times,+)} = C^{(Q)}$ performs multiplication in the required sense. More precisely the following holds.

Lemma 32 *The circuit $C^{(Q)}$ computes the polynomial $x_0(x_1 + x_2)$. Equivalently the following holds. Assume that I_0, I_1, I_2 are the deterministic input blocks of $C^{(Q)}$, where I_0 is also the deterministic input block of $\varphi(t_0)$, I_1 is the deterministic input block of $\mathbf{l}t_0$, I_2 is the deterministic input block of $\mathbf{r}t_0$, and T_0 is the output block of $C^{(Q)}$, where T_0 is also the output block of $\varphi(\mathbf{b}_{\bar{m}-1})$. Then, for all $\sigma \in \mathtt{func}(\mathtt{inset}(C^{(Q)}), F)$, we have $\bar{\sigma}_{\mathtt{in}}(I_0)(\bar{\sigma}_{\mathtt{in}}(I_1) + \bar{\sigma}_{\mathtt{in}}(I_2)) = \bar{\sigma}_{\mathtt{out}}(T_0)$, or equivalently*

$$\sum_{x \in I_0} \chi_\sigma^{(C^{(Q)})} \left(\sum_{x \in I_1} \chi_\sigma^{(C^{(Q)})} + \sum_{x \in I_2} \chi_\sigma^{(C^{(Q)})} \right) = \sum_{x \in T_0} \chi_\sigma^{(C^{(Q)})}$$

Proof. Using the notation $\mathbf{s} = \bar{\sigma}_{\mathtt{in}}(I_0)(\bar{\sigma}_{\mathtt{in}}(I_1) + \bar{\sigma}_{\mathtt{in}}(I_2))$ we have to show that $\mathbf{s} = \bar{\sigma}_{\mathtt{out}}(T_0) = \tilde{\sigma}(T_0^{(\mathbf{b}_{\bar{m}-1})})$. This is a consequence of Lemma 1. Indeed, Lemma1 and the connections between the input and output vectors of a splitting circuit imply, that for all $j = 0, 1, \dots, \mathbf{d} - 2$

$$\mathbf{s} = \sum_{t \in L_j} \tilde{\sigma}(I^{(t)}) \left(\tilde{\sigma}(I^{(\mathbf{l}t)}) + \sigma(I^{(\mathbf{r}t)}) \right)$$

We consider this sum for $j = \mathbf{d} - 2$. For each $t \in L_{\mathbf{d}-2}$, $\varphi(t)$ is a splitting circuit with $\tilde{\sigma}(I^{(t)}) = \sum_{i=0}^{1} \tilde{\sigma}(T_i^{(t)}) = \sum_{i=2}^{3} \tilde{\sigma}(T_i^{(t)})$. Therefore the last expression for $\mathbf{s}$ and the way we identified the input blocks of the leaky product circuits $\varphi(a)$, $a \in L_{\mathbf{d}}$ with the output blocks of the splitting circuits $\varphi(x), \varphi(y)$, where $a <_T x <_T y$ and $x \in L_{\mathbf{d}-1}, y \in L_{\mathbf{d}-2}$, implies that

$$\mathbf{s} = \sum_{t \in L_{\mathbf{d}}} \tilde{\sigma}(I_0^{(t)}) \left(\tilde{\sigma}(I_1^{(t)}) + \tilde{\sigma}(I_2^{(t)}) \right)$$

As a consequence the sum of the outputs of the leaky product circuits is also $\mathbf{s}$, that is,

$$\mathbf{s} = \sum_{t \in L_{\mathbf{d}}} \tilde{\sigma}(T_0^{(t)})$$

Since the rest of the circuit $C^{(Q)}$ simply adds the outputs of the leaky product circuits we get $\mathbf{s} = \tilde{\sigma}(T_0^{(\mathbf{b}_{\bar{m}-1})}) = \bar{\sigma}_{\mathtt{out}}(T_0)$, which completes the proof of the statement about the functionality of the circuit $C^{(Q)}$. *Q.E.D.*(Lemma 32)

Definition. For each $C \in \Phi$ we define a polynomial system of equations G_C over the field C. In each case G_C is the canonical polynomial system of equations of the corresponding circuit type, that is, of the splitting circuits, leaky product circuits, or the addition circuits. $\square$

Lemma 33 *Under the standard assumption described in* (53) the circuit $C^{(Q)} = \mathbf{C}_{m,G}^{(\times,+)}$ is $(\theta, \varepsilon_1, p)$-cylindrical, where $p = e^{-c_1 \bar{m}^{\frac{1}{2}}}$.

Proof. We have to define a selection function $\mathcal{S}$ such that the adversary $\mathcal{Y} = \mathtt{adv}_{C^{(Q)}}(\varepsilon_1, \mathcal{S})$ satisfies conditions (11) and (12) of the definition of a $(\theta, \varepsilon_1, p)$-cylindrical adversary.

For the following definition recall that for an arbitrary set Y, $\mathcal{X}(Y)$ denotes the set of all output nodes of $C_{\mathtt{end}}$ which do not belong to $\mathtt{exit}(Y \cap \mathtt{set}(C_{\mathtt{end}}), C_{\mathtt{end}})$, where $C_{\mathtt{end}}$ is the copying circuit which produces the output of the last addition circuit $\varphi(\mathbf{b}_{\bar{m}-1})$ in the circuit $C^{(Q)}$.

Definition. 1. $\Phi_\times$ will denote the set of all $C \in \Phi$, such that C is a leaky product circuit.

2. For an arbitrary set $X \subseteq \mathtt{set}(C^{(Q)})$, let $\mathcal{U}_X$ be the set of all $C \in \Phi_\times$ such that C is not percolative with respect to $\mathtt{full}(X) \cap \mathtt{set}(C)$ and G_C. We define $\mathcal{S}(X)$ by

$$\mathcal{S}(X) = X \cup \mathcal{X}(X) \cup \bigcup \{ \mathtt{set}(C) \backslash (\mathtt{detin}(C) \cup \mathtt{outset}(C)) \mid C \in \mathcal{U}_X \}$$

Let $\mathcal{Y} = \mathtt{adv}_{C^{(Q)}}(\varepsilon_1, \mathcal{S})$. $\square$

Assume now that D is a random subset of $\mathtt{set}(C^{(Q)})$ so that for all $x \in \mathtt{set}(C^{(Q)})$, $\mathtt{prob}(x \in D) \leq \varepsilon_1$ and the various events $x \in D$ are mutually independent. The definition of $\mathcal{S}(X)$, the fact that the copying circuit is $(\theta, \varepsilon_1, \frac{p}{2})$-cylindric (Lemma 12) and Lemma 18 together imply that with a probability of at least $1 - \frac{p}{2}$ condition (11) is satisfied by $C{:=}C^{(Q)}$, $X{:=}D$, $\mathcal{S}$, and $\mathcal{Y}$.

We want to show that with a probability of at least $1 - \frac{p}{2}$ with respect to the choice of the random set D, condition (12) is also satisfied with the same choices of the parameters. For the proof of this fact we will use Lemma 7. According to Lemma 7 it is sufficient to show that

Lemma 34 *Under the standard assumption the following holds. Assume $f \in \mathtt{func}_\theta(I, F)$, $\mathcal{G} = \mathtt{domain}(f)$, where I is the input block sequence of $C^{(Q)}$. Then with a probability of at least $1 - \frac{p}{2}$, for the randomization of the set D, the block input of $C^{(Q)}$ is invisible from the set $\mathbf{X}$, where*

$$\mathbf{X} = \mathcal{G} \cup \mathtt{full}\left(D \cup \mathcal{X}(D) \cup \bigcup\{\mathtt{set}(C)\backslash(\mathtt{detin}(C) \cup \mathtt{outset}(C)) \mid C \in \mathcal{U}_D\}\right)$$

Proof of Lemma 34. We need several definitions and lemmas for this proof.

Definition. For all sets $A \subseteq \mathtt{set}(C^{(Q)})$, $\Phi_A^{\neg\mathtt{perc}}$ will denote the set of all $C \in \Phi$ so that C is not percolative with respect to $A \cap \mathtt{set}(C)$ and G_C. □

Proposition 15 *Under the standard assumption, with a probability of at least $1 - e^{-2c_1 m}$ with respect to the randomization of the set D, we have $\Phi_{\mathbf{X}}^{\neg\mathtt{perc}} \cap (\Phi \backslash \Phi_\times) = \emptyset$.*

Moreover, the events $C \in \Phi_D^{\neg\mathtt{perc}}$, for $C \in \Phi_\times$, are mutually independent, and for each fixed $C \in \Phi_\times$, we have $\mathtt{prob}(C \in \Phi_D^{\neg\mathtt{perc}}) \leq \theta$.

Proof. For the proof of the first statement let $\tilde{c} > 0$ so that $\theta \ll \tilde{c} \ll d, \alpha$. Clearly for all $C \in \Phi\backslash\Phi_\times$ we have $\mathbf{X} \cap \mathtt{set}(C) \subseteq (\mathcal{G} \cup \mathcal{X}(\mathtt{full}(D)) \cup \mathtt{full}(D)) \cap \mathtt{set}(C)$. Therefore the assumption about $\mathcal{G}$ the definition of $\mathcal{X}(\mathtt{full}(D))$ and Chernoff's inequality applied for the number of elements of $D \cap \mathtt{set}(C)$ implies that $|\mathbf{X} \cap \mathtt{set}(C)| < \tilde{c}m$ with a probability of at least $1 - e^{-3c_1 m}$. Therefore applying Lemma 13, Lemma 24, or Lemma 26 according to the type (copying, splitting, or addition) of the circuit C, we get that C is $\mathbf{X} \cap \mathtt{set}(C), G_C$ percolative. With a probability of at least $1 - e^{-3c_1}$ this holds for all $1 - e^{-2c_2}$ which implies the first statement.

In the second statement of the lemma the independence of the events $C \in \Phi_\times$ follows from the fact that the sets $\mathtt{set}(C)$ for $C \in \Phi_\times$ are pairwise disjoint, and therefore the choices of the various elements of D in them are mutually independent. The inequality is a consequence of Lemma 29 *Q.E.D.*(Proposition 15)

Proof of Lemma 34 continued. Assume that $g \in \mathtt{func}(\mathbf{X}, F)$ and $\delta \in \mathtt{func}(\mathtt{block}_{\mathtt{in}}(C^{(Q)}), F)$. (Since $\mathtt{block}_{\mathtt{in}}(C^{(Q)})$ is the set of input blocks of $C^{(Q)}$, such a δ is a function defined on a set with 3 elements, which correspond to the variables of the polynomial $x_0(x_1 + x_2)$.) We will denote by $N_{\delta,g}$ the number of solutions of the system $\mathcal{I}(\delta) \cup \mathcal{E}(g)$ defined for the circuit $C^{(Q)}$. We have to prove that for all $g \in \mathtt{func}(\mathbf{X}, F)$ and for all $\delta, \delta' \in \mathtt{func}(\mathtt{block}_{\mathtt{in}}(C^{(Q)}), F)$, we have $N_{\delta,g} = N_{\delta',g}$. Assume that a $g \in \mathtt{func}(\mathbf{X}, F)$ is fixed. Because of the symmetry in δ and δ', it is sufficient to prove the following:

(66) *for all $\delta, \delta' \in \mathtt{func}(\mathtt{block}_{\mathtt{in}}(C^{(Q)}), F)$, $N_{\delta,g} > 0$ implies that $N_{\delta,g} = N_{\delta',g}$.*

For the proof of (66) assume that a $\delta \in \mathtt{func}(\mathtt{block}_{\mathtt{in}}(C^{(Q)}), F)$ is fixed with $N_{\delta,g} > 0$, and $\delta' \in \mathtt{func}(\mathtt{block}_{\mathtt{in}}(C^{(Q)}), F)$ is arbitrary. We prove statement (66) in two steps. As a first step we show that

Lemma 35 *$N_{\delta,g} > 0$ implies $N_{\delta',g} > 0$*

Using this lemma, we will show that Lemma 7 is applicable for the systems of equations $\mathcal{I}(\delta) \cup \mathcal{E}(g)$ and $\mathcal{I}(\delta') \cup \mathcal{E}(g)$.

Proof of Lemma 35. We prove the lemma by using Lemma 31, with the following choices of the parameters. The compatible family of circuits $Q = \langle \Phi, \leq_\Phi \rangle$, the polynomial systems G_C for $C \in \Phi$, and the set $\mathbf{X}$ has been already defined. Let $\Phi_{\mathbf{X}}$ be the set of all elements $C \in \Phi$ so that C is percolative with respect to $\mathbf{X} \cap \mathtt{set}(C)$ and G_C. Since $N_{g,\delta} > 0$ there exists a $\sigma \in \mathtt{func}(\mathtt{detin}(C^{(Q)}), F)$ so that $\bar{\sigma}_{\mathtt{in}} = \delta$. (Recall that $\bar{\sigma}_{\mathtt{in}}$ is the function which assigns to each input block, the sum of the values of σ on the elements of that input block.) We fix a σ with this property. We want to define functions λ, λ_C so that they satisfy condition (61) of Lemma 31. We will do that in a way that the function $\lambda \in \mathtt{func}(\Psi, F)$ restricted to $\mathtt{block}_{\mathtt{in}}(C^{(Q)})$ will be identical to δ', and for all $C \in \Phi \backslash \Phi_{\mathbf{X}}^{\neg\mathtt{perc}}$, we have $\lambda_C = \overline{(\sigma_C)}_{\mathtt{io}}$. This will guarantee that if $\rho \in \mathtt{func}(\mathtt{inset}(C^{(Q)}), F)$ is the function whose existence is stated in Lemma 31 then $z_h{:=}\rho(h)$ is a solution of $\mathcal{I}(\delta') \cup \mathcal{E}(g)$.

The existence of the function $\lambda \in \mathtt{func}(\Psi, F)$ that we need for the application of Lemma 31 will be guaranteed by the following proposition.

Proposition 16 *There exists a function $\lambda \in \mathtt{func}(\Psi, F)$ such that the following conditions are satisfied:*

(67) *λ_C satisfies the system G_C for all $C \in \Phi$, where λ_C is the restriction of λ to the set $\Psi_C \cup \mathtt{outset}(C)$.*

(68) *$\lambda_C = \overline{(\sigma_C)}_{\mathtt{io}}$ for all $C \in \Phi_{\mathbf{X}}^{\neg\mathtt{perc}}$.*

(69) *$\lambda(t_0) = \delta'(I_0)$, $\lambda(\mathbf{l}t_0) = \delta'(I_1)$, $\lambda(\mathbf{r}t_0) = \delta'(I_2)$*

For the construction of the function λ whose existence was claimed in the proposition we will need the following simple fact. (Recall that for all $C \in \Phi$, Ψ_C is the set consisting of all of the deterministic input blocks of C and all of the output blocks of C, and $\Psi = \bigcup_{C \in \Phi} \Psi_C$.)

Proposition 17 *Assume that Φ_0 is an upward closed subset of the set Φ with respect to the partial ordering $\leq_\Phi$, and μ is a function with values in F so that*

(70) *$\mathtt{domain}(\mu) = \bigcup \{\Psi_C \mid C \in \Phi_0\}$*

(71) *for all $C \in \Phi$, μ satisfies G_C.*

Then μ has an extension μ_1 to Ψ, so that, for all $C \in \Phi$, μ_1 satisfies the system G_C.

Proof. Since Φ is finite, it is sufficient to show that if $\Phi_0 \neq \Phi$, then there exists $C_1 \in \Phi \backslash \Phi_0$ and an extension μ_1 of μ to $\mathtt{domain}(\mu) \cup \Psi_{C_1}$, such that μ_1 satisfies G_{C_1}, and $\Phi_1 = \Phi_0 \cup \{C_1\}$ is an upward closed subset of Φ. Let C_1 be a maximal element of $\Phi \backslash \Phi_0$ with respect to the partial ordering $\leq_\Phi$. The maximality of C_1 and the definition of a compatible family of circuits implies, that if J is an output block of C_1, then for all $C \in \Phi_0$, J is neither an input block nor an output block of C. Therefore $J \notin \mathtt{domain}(\mu)$. Now we define an input τ for C_1, (including both the deterministic and probabilistic inputs), so that for each $I \in \mathtt{detin}(C_1)$, $\sum_{x\in I} \tau(x) = \mu(I)$. Apart from these equalities τ is arbitrary, in particular its values on the probabilistic input nodes can be chosen arbitrarily. The input τ defines an evaluation function $\chi_\tau^{(C_1)}$ on $\mathtt{set}(C_1)$. We define μ_1 by $\mu_1(J) = \sum_{x\in J} \chi_\tau^{(C_1)}(x)$ for all $J \in \Psi_{C_1}$. Since the circuit C_1 satisfies the system G_{C_1}, so does the function the function μ_1. *Q.E.D.*(Proposition 17)

Definition. The set of all $t \in L_{\mathbf{d}-1} \cup L_{\mathbf{d}-2}$ such that there exist $t' \in L_{\mathbf{d}-2}$ and $a \in L_{\mathbf{d}}$ with $t \leq_T t'$, $a <_T t'$, and $\varphi(a) \in \Phi_{\mathbf{X}}^{\neg\mathtt{perc}}$ will be denoted by $R_{\mathbf{X}}$. □

Proof of Proposition 16. First we define a function $\kappa' \in \mathtt{func}(R_{\mathbf{X}}, F)$. For each $t \in R_{\mathbf{X}}$, $\kappa'(t) = \tilde{\sigma}(I_0^{(t)}) = \sum\{\chi_\sigma^{(C(Q))}(x) \mid x \in I_0^{(t)}\}$.

Now we use Lemma 2 with $d{:=}\mathbf{d}-1$, $T{:=}T\backslash L_{\mathbf{d}}$, $A{:=}R_{\mathbf{X}}$, $A'{:=}R_{\mathbf{X}} \cap L_{\mathbf{d}-2}$, $\lambda'{:=}\kappa'$. According to Proposition 15, the events $t \in R_{\mathbf{X}}$, for all $t \in L_{\mathbf{d}-2}$, are mutually independent, therefore Lemma 2 is applicable. Using the upper bound on the probability of the events $t \in R_{\mathbf{X}}$ given in Lemma 15, we get the following. For an absolute constant $\hat{c}$, we have, that with a probability of at least $1 - 2^{-\hat{c}|\log \varepsilon_1|\sqrt{m}} \geq 1 - 2^{-\sqrt{m}} = 1 - 2^{-m^{\frac{3}{2}}}$ for the randomization of D, there exists a well-balanced function κ on $T\backslash L_{\mathbf{d}}$, so that κ is an extension of κ', and

(72) $\kappa(t_0) = \delta'(I_0)$, $\kappa(\mathbf{l}t_0) = \delta'(I_1)$, $\kappa(\mathbf{r}t_0) = \delta'(I_2)$.

Let Φ_0 be the set of all $C \in \Phi$, so that either there exists an $x \in T\backslash(L_{\mathbf{d}} \cup L_{\mathbf{d}-1} \cup L_{\mathbf{d}-2})\}$ with $C = \varphi(x)$ or there exists a $C' \in \Phi_{\mathbf{X}}^{\neg\mathtt{perc}}$ so that $C' \leq_\Phi C$.

We define a function μ on $M = \bigcup\{\Psi_C \mid C \in \Phi_0\}$. Assume that $J \in M$. If J is an input or output block of some $C = \varphi(x)$ with $x \in T\backslash(L_{\mathbf{d}} \cup L_{\mathbf{d}-1} \cup L_{\mathbf{d}-2})\}$ then $J \in \Psi_{\varphi(x),\mathtt{in}}$ for some $x \in T\backslash L_{\mathbf{d}} = \mathtt{domain}(\kappa)$. In that case we define μ by $\mu(J) = \kappa(\varphi(x))$, where $J \in \Psi_{\varphi(x),\mathtt{in}}$, $x \in T\backslash L_{\mathbf{d}}$.

Suppose now that $J \in M$, but for all $x \in T\backslash(L_{\mathbf{d}} \cup L_{\mathbf{d}-1} \cup L_{\mathbf{d}-2})$, J is not an input block or output block of $\varphi(x)$. By the definition of Φ_0 this is possible only if J is an input block or an output block of a $C \in \Phi_{\mathbf{X}}^{\neg\mathtt{perc}}$. In that case let $\mu(J) = \tilde{\sigma}(J)$. (Recall that by definition $\tilde{\sigma}(J) = \overline{(\sigma_C)}_{\mathtt{io}}(S) = \sum_{x\in S} \chi_\sigma^{(C)}(x)$, where S is an arbitrary input or output block of a circuit $C \in \Phi$.) We claim that μ satisfies the system G_C for all $C \in \Phi_0$. If $C = \varphi(x)$ with $x \in T\backslash(L_{\mathbf{d}} \cup L_{\mathbf{d}-1} \cup L_{\mathbf{d}-2})$ then this is a consequence of the fact that κ is well balanced, and the equations defining "λ is well-balanced", are identical to the equations of the canonical system of a splitting circuit. For the remaining circuits $C \in M$, by the definition of κ' and κ, we have that $\mu(J) = \tilde{\sigma}(J)$ for all $J \in \Psi_C$ therefore we get that μ satisfies the system $G_C = 0$ since $\tilde{\sigma}$ satisfies it.

We have shown that the set Φ_0 and the function μ satisfy the requirements of Proposition 17. Therefore there exists an extension μ_1 of μ onto $\bigcup\{\Psi_C \mid C \in \Phi\}$ with the properties described in Proposition 17. We define λ by $\lambda = \mu'$. We show now that the conditions of Proposition 16 are satisfied by the function λ.

Condition (67). This is a consequence of the fact that $\mu_1 = \lambda$ satisfied condition (71) of Proposition 17.

Condition (68). For all input and output blocks of a circuit $C \in \Phi_{\mathbf{X}}^{\neg\mathtt{perc}}$, we have $\lambda(J) = \mu_1(J) = \mu(J)$. The definition of the function μ implies that for such a block J we have $\mu(J) = \tilde{\sigma}(J)$, which implies our statement.

Condition (69). On the three input blocks I involved in this condition we have $\lambda(I) = \mu(I) = \kappa(t)$, where I is an input block of $\varphi(t)$. Therefore condition (72) implies our statement. *Q.E.D.*(Proposition 16)

Proof of Lemma 35 continued. Proposition 16 guarantees the existence of a function $\lambda \in \mathtt{func}(\Psi, F)$. Therefore we can apply Lemma 31 with the choice of parameters described at the beginning of the proof of Lemma 35 and get a solution for the system $\mathcal{I}(\delta') \cup \mathcal{E}(g)$. *Q.E.D.*(Lemma 35)

Recall that the circuit $\mathbf{C}_m^{(\times,+)}$ has three input blocks and so the system $\mathcal{I}(\delta)$ consists of three equations.

Definition. We write the system of equations $\mathcal{I}(\delta) \cup \mathcal{E}(g)$ in the form of $q_\alpha = \hat{\varphi}(\alpha)$, $\alpha \in E$, where

(a) $E = \{0,1,2\} \cup \mathbf{X}$ (we assume that $\{0,1,2\} \cap \mathbf{X} = \emptyset$),

(b) for all $\alpha \in E$, q_α is a polynomial whose constant term is 0,

(c) $q_i = P_i$ and $\hat{\varphi}(i) = \delta(i)$ for $i = 0,1,2$,

(d) for all $a \in \mathbf{X}$, the polynomials $q_a - \hat{\varphi}(a)$ and $p_a^{(g)} - g(a)$ are identical. □

Remark. Condition (d) of the definition above does not imply $\hat{\varphi}(a) = g(a)$, since $p_a^{(g)}$ may have a nonzero constant term. □

Definition. We define a function $\psi(\alpha)$ on the set E by: "for all $a \in \mathbf{X}$, $\psi(a) = \hat{\varphi}(a)$, and for all $i \in \{0,1,2\}$, $\psi(i) = \delta'(i)$". Clearly the systems $\mathcal{I}(\delta') \cup \mathcal{E}(g)$ and $q_\alpha = \psi(\alpha)$, $\alpha \in E$ are identical. □

Lemma 36 *The conditions of Lemma 7 are satisfied by the polynomials q_α, $\alpha \in E$ defined above. Consequently the number of solutions of the systems $\mathcal{I}(\delta) \cup \mathcal{E}(g)$ and $\mathcal{I}(\delta') \cup \mathcal{E}(g)$ are identical.*

Proof. The requirement that the constant terms of the polynomials q_α, $\alpha \in E$ are all zeros is included in the definitions of these polynomials. U will denote the set of indeterminates occurring in the polynomials q_α, $\alpha \in E$. Now we define the sets E_0, E_1, U_0, U_1. For each leaky product circuit $C \in \Phi_\times$, let Y_C be the unique product gate of C.

Let H_0 be the set of all input nodes h of $C^{(Q)}$, including both probabilistic and deterministic input nodes, with the property that $h > Y_C$ for a suitably chosen $C \in \Phi_\times$ with respect to the partial ordering of the nodes of the circuit $C^{(Q)}$. U_0 is defined by $U_0 = \{z_h \mid h \in H_0\}$.

Let $\bar{\mathbf{X}}$ be the set of all $a \in \mathbf{X}$, so that there exists a $C \in \Phi_\times$ with $Y_C < a$. We define E_0 by $E_0 = \bar{\mathbf{X}} \cup \{0,1,2\}$. Finally $E_1 = E \backslash E_0$, $U_1 = U \backslash U_0$. We will show that the conditions of Lemma 7 are satisfied by these choices. Condition (35) is an immediate consequence of the definitions of U_0, U_1, E_0 and E_1. To show that the conditions hold as well we need the following.

Proposition 18 *Suppose that "a" is a gate of the circuit $C^{(Q)}$. "a" is a product gate iff $a = Y_C$ for a suitably chosen $C \in \Phi_\times$. Moreover if t is a nonlinear term of some q_α, $\alpha \in E$, then $\alpha \in \mathbf{X}$, and there exists a $C \in \Phi_\times$ so that $\alpha \le Y_C$, and $p_{Y_C}^{(g)}$ contains a term t' so that $t = \iota t'$ for some $\iota \in F$.*

Proof. The first statement of the proposition is an immediate consequence of the definition of $C^{(Q)}$. Assume now that t is a nonlinear term of some q_α. The fact that P_0, P_1, P_2 have only linear terms implies that $\alpha \in \mathbf{X}$, and therefore q_α and $p^{(g)}_\alpha$ may differ only in their constant terms. Consequently, $p^{(g)}_\alpha$ contains the nonlinear term t. If α is not a product gate, then, according to the recursive definition of $p^{(g)}_a$, there exists a $b \in \texttt{set}(C^{(Q)})$ so that $a \prec b$ (in the partial ordering of $C^{(Q)}$) and there exists a term $\bar{t}$ of $p^{(g)}_b$ so that $t = \beta\bar{t}$ for some nonzero $\beta \in F$. (This is a consequence of the fact that the only gates in this part of the circuit are $+, \times(-1)$). Therefore we can define a path, going upward starting from a, so that for each element b_i of this path the polynomial $p^{(g)}_{b_i}$ contains a term t_i so that $t = \beta_i\bar{t}_i$ for some $\beta_i \in F\backslash\{0\}$. The path must end in a node with a product gate, that is, in an node on Y_C for some $C \in \Phi_\times$. *Q.E.D.*(Proposition 18)

Proof of Lemma 36 continued. We show now that the remaining conditions of Lemma 7 are satisfied by the sets E, E_0, U, U_0 defined earlier.

Condition (36). Assume that $\alpha \in E_0$. By the definition of E_0 we have that $\alpha \in \{0,1,2\} \cup \bar{\mathbf{X}}$. If $\alpha \in 0,1,2$ then for each indeterminate z_h in q_α, h is a deterministic input node of $C^{(Q)}$ and so $h \geq C$, for some $C \in \Phi_\times$ and therefore $z_h \in U_0$. If $\alpha \in \bar{\mathbf{X}}$ then the definition of $\bar{\mathbf{X}}$ implies the existence of a $C \in \Phi_\times$ with $Y_C < a$ and so for all indeterminates z_h in q_a we have $Y_C < a \leq z_h$ and so $z_h \in U_0$. In both cases Proposition 18 implies that q_α has linear terms only.

Condition (37). Assume that t is a nonlinear term of q_α for some $\alpha \in E_1$. Then, by Proposition 18 there exists a $C \in \Phi_\times$ a term $t' \in p^{(g)}_{Y_C}$ and a $\iota \in F$, so that $t = \iota t'$. Therefore if z_h is an indeterminate in t then it also occurs in t' and consequently $h > Y_C$, which implies $z_h \in U_0$.

Condition (38). We can use Lemma 31 to show that the condition is satisfied. To make the lemma applicable to the present case we define a new compatible family of circuits $\tilde{Q}$. Namely, for each $C \in \Phi_\times$ we cut C into two parts C_0 and C_1 so that C_0 is a total sum circuit, C_1 is a reverse sum circuit and the only output node of C_0 is identical to the only input node of C_1. Let $\tilde{\Phi}$ be the set

$$(\Phi\backslash\Phi_\times) \cup \{C_i \mid C \in \Phi_\times, i \in \{0,1\}\}$$

The partial ordering $\leq_{\tilde{\Phi}}$ is defined in the following way. $\leq_{\tilde{\Phi}}$ is the unique partial ordering so that: (a) for all $C \in \Phi_\times$, $C_0 < C_1$, (b) for all $C, D \in \Phi_\times$, $i, j \in \{0,1\}$ if $C \neq D$ then C_i, D_j are incomparable, and (c) for all $x, y \in \tilde{\Phi}$ if at least one of the elements x, y is not of the form C_i for suitably chosen $C \in \Phi_\times$, $i \in \{0,1\}$, then $x \leq_{\tilde{\Phi}} y$ iff $x \leq_\Phi y$. $C^{(\tilde{Q})}$ will denote the circuit defined by the family $\tilde{\Phi}$. Clearly $C^{(Q)} = C^{(\tilde{Q})}$, that is, the two circuits have the same nodes with the same type of gates on each of them, we only got them through different constructions. (More precisely they may be only isomorphic depending on the definition of the process of "identification" of nodes in their constructions.) For each $C \in \tilde{\Phi}_\times$ and for all $i \in \{0,1\}$, the canonical system of the circuit C_i as defined for total sum, and reverse sum circuits, (that is, e.g., the total sum circuit the system which says that the sum of the inputs is the output) will be denoted by G_{C_i}. We define a set $\tilde{\mathbf{X}}$ by $\tilde{\mathbf{X}} = (\mathbf{X}\backslash\Phi_\times) \cup \bigcup\{\{C_0, C_1\} \mid C \in \mathbf{X}\}$.

Later we will use the following statement that follows easily from the definition of percolativity:

(73) *if $C \in \Phi$ and C is percolative with respect $\mathbf{X} \cap C$ and G_C, then for all $i \in \{0,1\}$, C_i is also percolative with respect to $\mathbf{X} \cap C_i$ and G_{C_i}*

Assume now that the system $\mathcal{E} = \{q_\alpha = \hat{\varphi}(\alpha) \mid \alpha \in E\}$ has at least one solution, and we have to show that every solution of the system $\mathcal{E}_0 = \{q_\alpha = \hat{\varphi}(\alpha) \mid \alpha \in E_0\}$ can be extended into a solution of $\mathcal{E}$.

We want to use Lemma 31 in the proof of condition of (38) with $Q{:=}\tilde{Q}$, $A{:=}\tilde{\mathbf{X}}$. We define the functions σ, λ whose existence is stated in the assumptions of Lemma 31 in the following way.

The definition of σ. Since $\mathcal{E}$ has a solution there exists a $\sigma \in \mathtt{func}(\mathtt{inset}(C^{(\tilde{Q})}), E)$ so that $z_h = \sigma(h)$ is a solution of $\mathcal{E}$.

Let $\mathbf{H}$ be the set of all $h \in \mathtt{inset}(C^{(\tilde{Q})})$, so that h is either in a splitting circuit or a total sum circuit. Assume that $z_h = \tau(h)$, $h \in \mathbf{H}$ is a solution, for $\mathcal{E}_0$ that we want to extend into a solution of $\mathcal{E}$. λ and λ_C will depend on τ.

The definitions of λ *and* λ_C. We define a function $\lambda \in \mathtt{func}(\bigcup\{\mathtt{block_{io}}(C) \mid C \in \tilde{\Phi}\}, F)$ with the properties required by Lemma 31. For the definition of λ we extend τ from the set $\mathbf{H}$ to every element of the set $\mathtt{inset}(C^{(\tilde{Q})})$ with values in F.

(74) *The extension of* τ *is defined by the equations of* $\mathcal{E}_1$ *for all* $h \in \mathtt{inset}(C^{(\tilde{Q})})\backslash\mathbf{H}$ *if each* $h \in \mathtt{inset}(C)$ *for some* $C \in \tilde{\mathbf{X}}$.

For the remaining elements h of $\mathtt{inset}(C^{(\tilde{Q})})\backslash\mathbf{H}$ the extension is arbitrary. This extension of τ will be denoted by $\bar{\tau}$. For each $J \in \bigcup\{\mathtt{block_{io}}(C) \mid C \in \tilde{\Phi}\}$ we define $\lambda(J)$ by $\lambda(J) = \sum_{x\in J} \chi_{\bar{\tau}}^{C^{(\tilde{Q})}}$. We define λ_C as the restriction of λ onto $\mathtt{block_{io}}(C)$, for each $C \in \tilde{\Phi}$. Since λ was defined from an evaluation function, the function λ_C satisfies G_C for all $C \in \tilde{\Phi}$ therefore condition (61) of Lemma 31 satisfies with $\Phi{:=}\tilde{\Phi}$.

Condition (62) is a consequence of definition of the extension of τ described in condition (74). Conditions (60),(60),(61) of Lemma 31 are immediate consequences of the definitions.

From the evaluation ρ whose existence is guaranteed by Lemma 31 we get the solution of $\mathcal{E}$ by $z_h = \chi_\rho(h)$. This completes the proof of condition (38) of Lemma 7

Now we apply Lemma 7 to the systems and $\mathcal{I}(\delta') \cup \mathcal{E}(g)$. By Lemma 35 if at least one of these systems has a solution then both of them have at least one solution, consequently Lemma 7 implies that the number of their solutions are equal. *Q.E.D.*(Lemma 36)

Proof of Lemma 34 continued. According to Lemma 36 the systems $\mathcal{I}(\delta) \cup \mathcal{E}(g)$ and $\mathcal{I}(\delta') \cup \mathcal{E}(g)$ has the same number of solutions which implies that the block input of $C^{(Q)}$ is invisible from $\mathbf{X}$. *Q.E.D.*(Lemma 34)

Proof of Lemma 33 continued. We have shown in Lemma 34 that the block input of $C^{(Q)}$ is invisible from the set $\mathbf{X} \supseteq \mathcal{G} \cup D$, therefore by Lemma 6 it is also invisible from the set $\mathcal{G} \cup D$. *Q.E.D.*(Lemma 33)

This completes the proof of Lemma 3 as well.

8 Leaking Machines

8.1 Motivation

Assume that M is a machine, we may think of a RAM, or a Turing machine, or a pointer machine, which may get input, may provide output, and while it works an adversary gets some information about what is happening inside the machine. For example in case of a RAM, the adversary may get at certain times the contents of the memory cells involved in the instruction executed at that time. The adversary in possession of this partial information about the working of the machine tries to guess what is its input. Our goal is to describe certain possible adversaries and find ways to hide most of the information contained in the input from them in an efficient way. This can be done in certain cases by replacing the program P_0 of the machine by another program P_1, which gives the same output, but works in a way which reveals less useful information to the adversary than the original program P_0. Depending on the adversary there may be certain type of information, that cannot be hidden without making the program P_1 too inefficient. E.g., for many reasonable choice of an adversary the total amount of time/memory needed to compute the correct output cannot be hidden efficiently. In such a case we will assume that these parameters, the total time and the total amount of memory used by the program, are fixed, and our goal will be to hide any other information from the adversary.

For the formulation of our results and especially for the proof, we will need many different computational models. Therefore, first we give a very general definition of a probabilistic machine with information leaking to an adversary. All of the specific computational models that we will use later will be a special case of this one. In this general model, that we will call a leaking machine, only the following constituents of a machine will be given. (a) a distribution which determines the output of the machine if an input is given, and (b) for each possible input a a distribution of the information (as a $0, 1$ sequence) that reaches the adversary if the machine gets input a. More precisely we will give the joint distribution of the two distributions described in (a) and (b).

These concepts in themselves of course do not tell how the computation is performed. Still for many definitions and proofs, particularly when we reduce the problem of hiding information, from one computational model to another one, these notions will be sufficient.

One of the basic problems in defining precisely our general goal of hiding information is, that certain type of information, as we have mentioned, cannot be hidden efficiently, so we have to concede it to the adversary, and try to hide only extra information, that is information which does not follow from the conceded one. We will formulate this in the following way. We will define two adversaries. Adversary 1 will get every information that cannot be efficiently hidden, in each case we will tell explicitly what is it. Adversary 2 will get all the information that Adversary 1 gets and also some extra information which will be defined in each case. Our results usually will state that the computation can be performed in a way, perhaps in another slightly larger machine, so that the two machines compute the same function and at the end of the computation Adversary 2 will not know essentially more than Adversary 1. This last concept will be formulated in terms of indistinguishability. We will essentially require that for each fixed pair a, b of possible inputs, if Adversary 1 and Adversary 2 are trying to guess what was the input, knowing that it was chosen with uniform distribution from the set $\{a, b\}$, and knowing the information that reached them during the computation, then the probability that Adversary 2 gives the right answer can be only larger by ε than the same probability for Adversary 1, where ε is negligible.

8.2 Definition of a leaking machine

Definition. $\{0,1\}^{<\infty}$ will denote the set of all finite $0,1$-sequences. ω will denote the set of all natural numbers. Suppose that A, B are sets. $\texttt{func}(A, B)$ will denote the set of all functions defined in A and with values in B. If $a = \langle a_1, ..., a_i \rangle$, $b = \langle b_1, ..., b_j \rangle$ are finite sequences, then their concatenation the sequence $\langle a_1, ..., a_i, b_1, ..., b_j \rangle$ will be denoted by $a \circ b$. $\Box$

Definition. 1. The pair $M = \langle A, \wp \rangle$ will be called a leaking machine if it satisfies the following conditions.

(a) A is a finite subset of $\{0,1\}^{<\infty}$,

(b) $\wp$ is a function defined on A whose value on each $a \in A$ will be denoted by $\wp_a$, and $\wp_a$ is a probability measure defined on the σ-algebra consisting of all subsets of $\{0,1\}^{<\infty} \times \{0,1\}^{<\infty}$.

2. Assume that $M = \langle A, \wp \rangle$ is a leaking machine and $\langle \vartheta, \xi \rangle$ is a pair of functions each defined on the set A. For each $a \in A$, ϑ_a denotes the value of ϑ on a and ξ_a denotes the value of ξ on a. Suppose further that for each $a \in A$, both ϑ_a and ξ_a are random variables with values in $\{0,1\}^{<\infty}$, and their joint distribution on $\{0,1\}^{<\infty} \times \{0,1\}^{<\infty}$ is $\wp_a$. Then we will say that the pair $\langle \vartheta, \xi \rangle$ is a realization of the leaking machine M. $\Box$

Remark. The intuitive meaning of the leaking machine $M = \langle A, \wp \rangle$ is the following. We may think that M is a probabilistic machine, e.g., a RAM, which may get an element of A as an input, and the possible outputs of the machine are elements of the set $\subseteq \{0,1\}^{<\infty}$. Since the machine is probabilistic, the input a does not determine uniquely the output, only its distribution on $\{0,1\}^{<\infty}$. We assume that there is an adversary who is watching the machine and gets some information about it. This information depends on the input $a \in A$, the working of the machine, including its probabilistic steps, and other probabilistic events outside the machine which may influence what the adversary can observe about the behavior of the machine. The information which can reach the adversary is an element of a set $D \subseteq \{0,1\}^{<\infty}$. The joint distribution of the output and the information reaching the adversary at input a is the distribution $\wp_a$ on $\{0,1\}^{<\infty} \times \{0,1\}^{<\infty}$. In other words, if at input a the output is the value of the random variable ϑ_a, and the information that the adversary gets is the value of the random variable ξ_a, then $\wp_a$ is the distribution of the pair $\langle \vartheta_a, \xi_a \rangle$.

In the definition of a leaking machine we did not describe, how the machine performs the computation. The probability distributions, which define a leaking machine, describe only how the input determines the output of the machine and what is the information that reaches the adversary. We will use however the concept of a leaking machine in cases when the output is computed from the input in some specific computational model e.g., by a RAM. That is, ϑ_a will be the output of a (possibly probabilistic) RAM at input a, and ξ_a will be usually defined also in terms of the RAM. E.g., if our goal is oblivious simulation, then ξ_a can be the memory access pattern of the RAM during the computation at input a. $\Box$

Definition. 1. If μ, ν are probability measures on the same σ-algebra $\mathcal{A}$, then the distance of μ and ν is $\sup\{|\mu(B) - \nu(B)| + |\mu(D) - \nu(D)|\}$ taken for all $B, D \in \mathcal{A}$ with $B \cap D = \emptyset$. $\Box$

Definition. Assume that $M = \langle A, \wp \rangle$ is a leaking machine, $\langle \vartheta, \xi \rangle$ is a realization of M, $a, b \in A$, and $\varepsilon > 0$. We will say that the adversary can distinguish a from b with a bias greater than ε, if the following holds. There exists a function S defined on $\{0,1\}^{<\infty}$, so that if we take a random element h from the set $\{a, b\}$ with uniform distribution, and a random value x of the random variable ξ_h is given, then $\texttt{prob}(S(x) = h) > \frac{1}{2} + \varepsilon$. $\texttt{bias}_M(a, b)$ will denote the largest real δ, so that for all positive $\varepsilon < \delta$, the adversary can distinguish a from b with a bias greater than ε. $\Box$

Definition. Assume that $M = \langle A, \wp \rangle$ is a leaking machine and $\langle \vartheta, \xi \rangle$ is a realization of M. For each $a \in A$ the distribution of ϑ_a on $\{0,1\}^{<\infty}$ will be denoted by $\bar{\wp}_a$. $\Box$

Definition. Assume that $M_i = \langle A_i, \wp^{(i)} \rangle$, $i = 0, 1$ are leaking machines, for $i = 0, 1$, and $\varepsilon > 0$. We will say that the machine M_1 simulates the machine M_0 with with an error of at most ε, or M_1 is an ε-simulation of M_0, if the following two conditions are satisfied:

(75) $A_0 = A_1$, *and for all* $a \in A_0$, *the distance of* $\bar{\wp}_a^{(0)}$ *and* $\bar{\wp}_a^{(1)}$ *is at most* ε,

(76) *for all* $a, b \in A_0$ *we have* $\texttt{bias}_{M_1}(a, b) \leq \texttt{bias}_{M_0}(a, b) + \varepsilon$. $\Box$

Remark. The motivation for the definition of ε-simulation is the following. We may think that M_0 and M_1 are random access machines which get their inputs from the same set $A = A_0 = A_1$, and $\varepsilon > 0$ is so small that a single event with probability at most ε can be disregarded. M_1 simulates M_0 with an error of at most ε in the sense that M_1 at the input a it gives essentially the same output (in a probabilistic sense) than M_0, and the adversary of M_1 does not gain essentially more information about the input than the adversary of M_0. The meaning of the word "essentially" in both cases depends on ε as formulated in condition (75) and (76) of the definition. Therefore, we have that from the point of view of getting the correct output and hiding the input from the adversary, machine M_1 is essentially as good as machine M_0. This notion will be useful when the adversary of machine M_0 is relatively weak so it is easy to understand what can the adversary of M_0 learn about the input. If M_1 is an ε-simulation of M_0 and M_1 has a possibly much stronger adversary, then we have shown that the computation done by M_1 is just as secure as the computation on M_0 in spite of its stronger adversary.

Another way of using the notion ε-simulation will be of applying Lemma 37, formulated below, repeatedly to a sequence of machines and showing this way that the last machine in the sequence is essentially as secure than the first one. $\Box$

Lemma 37 *Assume that* $M_i = \langle A_i, \wp_i \rangle$, $i = 0, 1, 2$ *are leaking machines, and* M_i *is an* ε_i*-simulation of* M_{i-1} *for* $i = 1, 2$. *Then the machine* M_2 *is an* $\varepsilon_1 + \varepsilon_2$*-simulation of the machine* M_0.

Proof. We have to show that conditions (75) and (76) of the the definition of ε-simulation hold for M_0 and M_2.

Condition (75). Since M_i is an ε_i-simulation of M_{i-1} for $i = 1, 2$, we have $A_0 = A_1$ and $A_1 = A_2$ therefore $A_0 = A_2$. For the same reasons, for all $a \in A_0$, we have $\texttt{distance}(\bar{\wp}_a^{(0)}, \bar{\wp}_a^{(1)}) \leq \varepsilon_1$ and $\texttt{distance}(\bar{\wp}_a^{(1)}, \bar{\wp}_a^{(2)}) \leq \varepsilon_2$, consequently the triangle inequality for the distance between distributions imply that $\texttt{distance}(\bar{\wp}_a^{(0)}, \bar{\wp}_a^{(2)}) \leq \varepsilon_1 + \varepsilon_2$.

Condition (76). Assume that $a, b \in A_0 = A_1 = A_2$. Since machine M_i ε_i-simulates machine M_{i-1} for $i = 1, 2$ we have $\texttt{bias}_{M_1}(a, b) \leq \texttt{bias}_{M_0}(a, b) + \varepsilon_1$ and $\texttt{bias}_{M_2}(a, b) \leq \texttt{bias}_{M_1}(a, b) + \varepsilon_2$ therefore $\texttt{bias}_{M_2}(a, b) \leq \texttt{bias}_{M_0}(a, b) + \varepsilon_1 + \varepsilon_2$. *Q.E.D.*(Lemma 37)

9 RAMs as Leaking Machines

9.1 Random access machines

We use a von Neumann type random access machine, where data and program are not distinguished. For the definition for such a machine see e.g., in [1] the random access stored program (RASP) machines, in the modified form where the contents of the memory cells are not arbitrary integers, but integers in the interval $[0, 2^q - 1]$.

For each positive integer $q \geq 10$, $\mathcal{M}_q$ is a machine with 2^q memory cells each containing a sequence of $0, 1$ bits of length q. These cells will be called $\texttt{cell}(0), \texttt{cell}(1), \ldots, \texttt{cell}(2^q - 1)$. We will consider the sequences contained in the cells as the binary representations of natural numbers from the interval $[0, 2^q - 1]$, so if we say that a cell contains the natural number i, we mean it contains its binary representation. (We may restrict the number of memory cells if needed, by simply saying that a particular program is using only the first m cells for some $m < 2^q$. Therefore our requirement that the machine has $2^q - 1$ memory cells, is not a real restriction). The state of the machine at each time is a function which assigns to each of memory cells its content.

$\mathcal{M}_q$ has γ_0 instructions. The names or encodings of these instructions are integers in $[0, 2^q - 1]$. We will describe below a finite list of instructions and assume that their names, as natural numbers, are fixed independently of q.

$\texttt{cell}(0)$ is called the accumulator of the machine and $\texttt{cell}(1)$ the instruction pointer. (The choice of these particular cells for the mentioned roles have no significance.)

The machine $\mathcal{M}_q$ has six types of instructions. (a) arithmetic operations, (b) an instruction to generate a random number, (c) instructions moving data between the memory cells, (d) control transfer instructions, which determine which instruction will be executed next, (e) input/output instructions, and (f) the halt instruction to terminate the execution of the program. We will define each of these types later in more detail.

The machine $\mathcal{M}_q$ is working in cycles, each cycle counts as one time unit. In each cycle it does the following. It checks the content of the instruction pointer. Its content is interpreted as an address, of a memory cell, say, number i. Then the machine executes the instruction whose name is in cell i. An instruction may have parameters (for the sake of simplicity we assume that each instruction has at most one parameter). A parameter typically is the address of a memory cell. The content of cell $i + 1$ is considered as the parameter of the instruction in cell number i. The machine executes the instruction with the indicated parameter and then, if it is not a control transfer instruction, it increases the content of the instruction pointer by 2. If it is a control transfer instruction then the instruction defines the new content of the instruction pointer. We will say that *a memory cell or register is involved in an instruction* if its content is used by the machine to execute the instruction, or the result of the instruction is placed in it. We will use this concept by considering an adversary who wants to get some information about what the machine is doing, and at each instruction knows which memory cells/registers are involved in the instructions, but does not know their contents.

(a) The arithmetic instructions are $+, \times, -, \lfloor x/y \rfloor$, the constants $0, 1$, and $2^q - 1$. (According to our assumption even the name of the instruction $2^q - 1$ is independent of q.) In case of the arithmetic operations of the form $f(x, y)$, at the time when the machine reads the instruction, which is in `cell`(i), x must be in the accumulator, and y must be in the memory cell whose address is the parameter of the instruction, that is, y is in `cell`(a) where a is the content of `cell`$(i+1)$. The result appears in the accumulator. In the case of the constants, the result of the instruction, that is, the value of the constant in binary form, appears in the accumulator (and the value of the parameter is irrelevant).

(b) an instruction to generate a random number. A random integer from the interval $[0, 2^q-1]$ appears in the accumulator, the value of the parameter is irrelevant.

(c) instructions moving data between the memory cells. Read instruction: if the value of the parameter is a, then the instruction puts the content of `cell`(a) into the accumulator. Write instruction: if the value of the parameter is a, then the instruction puts the content of the accumulator into `cell`(a).

(d) control transfer instructions. GOTO X instruction. If the value of the parameter is X then the content of the instruction pointer is changed into a. "IF $X = 0$ THEN GOTO Y" instruction. If the content of the accumulator is 0 then the content of the instruction pointer is changed into Y, where Y is the value of the parameter, otherwise the value of the instruction pointer is increased by 2. "IF $X > 0$ THEN GOTO Y". If the content of the accumulator is greater than 0, then the content of the instruction pointer is changed into Y, where Y is the value of the parameter, otherwise the value of the instruction pointer is increased by 2.

(e) input/output instructions. INPUT instruction. The input is written in the accumulator. OUTPUT instruction. The value of the accumulator is given as output. In both cases the value of the parameter is irrelevant.

(f) HALT instruction. Terminates the execution of the program.

The first few memory cells sometimes will be also called registers. (Intuitively this corresponds to the CPU of a computer.)

A state of the machine $\mathcal{M}_q$, as we have indicated earlier, is a function which assigns to each of it cells a possible content. A history of the machine $\mathcal{M}_q$ is a function defined on an initital segment I of the natural numbers which assigns a state $S(t)$ to each $t \in I$ with the following property. Suppose that $t, t+1 \in I$ and in $S(t)$ the content of the instruction pointer is a. If a is the name of an instruction different of the input instruction and the random number generator instruction, then we get $S(t+1)$ from $S(t)$, by executing the instruction a with the values contained in the memory cells of $\mathcal{M}_q$ defined by $S(t)$. If a is the input or random number generator instruction then $S(t+1)$ must be a state that is obtained from $S(t)$ by executing instruction a with the values of the memory cells described in $S(t)$, and with a suitably chosen value of the input or the generated random number.

Remark. Usually we will assume that before the machine starts to work, a program a of constant length is placed in the memory. We will call this the starting program. We may think that this is a small program, whose role is to write in the memory a larger program b and data for b. When we will consider an adversary who wants to get some information about what the machine is doing, we will assume that b is known to the adversary. Because of this, the exact way as b gets into the machine is irrelevant. $\Box$

Definition. Assume that $v = \langle v_0, v_1, ..., v_{l-1}\rangle$ is a sequence of natural numbers each in the interval $[0, 2^{q-1}]$. We consider a machine $\mathcal{M}_q$ with at least l memory cells and with the following initial state. `cell`(i) contains the integer v_i for all $i = 0, 1, ...l-1$ and `cell`(i) contains the integer 0 for all $i > l-1$. The machine $\mathcal{M}_q$ with this initial state will be denoted by $\mathcal{M}_q[a]$ and will called the machine $\mathcal{M}_q$ with the initial program a. □

Remark. Sometimes an initial program a of $\mathcal{M}_q$ will be given in the form of $a = P \circ w$ where P is a program, (that is, a finite sequence of natural numbers) which can be executed for all sufficiently large q in the machine $\mathcal{M}_q$, and w contains data depending on q. For example we may want to tell to a fixed program P the following parameters: n, the number of memory cells of the machine and, t, the total time that can be used by the program. In this case the starting program will be of the form $P \circ \langle n, t\rangle$ where P is fixed independently of q but n and t may depend on q.□

Definition. The machine $\mathcal{M}_q$ with n memory cells will be denoted by $\mathcal{M}_{q,n}$. The machine $\mathcal{M}_{q,n}$ with a limit t on the time it can use will be denoted by $\mathcal{M}_{q,n,t}$. Such a machine stops a time t whatever would be its next instruction. □

9.2 Leaking machines based on $\mathcal{M}_q$.

9.2.1 The standard leaking RAM with benign adversary

We will define a leaking machine M by describing a realization $\langle \vartheta, \xi\rangle$ of M. For the definition of ϑ and ξ we will use a machine $\mathcal{M}_{q,n,t}$. The leaking machine M that we will define below will depend on the following parameters. (a) q, the word length in the machine $\mathcal{M}_q$, (b) n, the number of memory cells used in $\mathcal{M}_q$, (c) t, the total amount of time that can be used by $\mathcal{M}_q$ for the computation, (d) P, the initial program that resides in $\mathcal{M}_q$ when the computation starts.

Definition. Assume that q, n, t are positive integers, $q > 10$, $n < 2^q, t < 2^q$ and P is a sequence $\langle p_0, \ldots, p_{k-1}\rangle$ where each p_i is an integer in $[0, 2^q - 1]$. We define a leaking machine $M = M[q, n, t, P] = \langle A, \wp\rangle$ which will be called the standard leaking RAM with benign adversary, with word length q, memory size n, total time t and initial program P. A will be the set of all $0, 1$ sequences of length at most qt. We will also refer to this leaking machine as the standard leaking RAM. We define the distributions $\wp_a$ by describing a realization $\langle \vartheta, \xi\rangle$ of the machine M.

The definition of ϑ_a. Assume that $a \in A$ and we consider the machine $\mathcal{M}_{q,n,t}[P]$, that is, the machine $\mathcal{M}_{q,n,t}$ with the initial program P. Let a' be the $0, 1$-sequence of length exactly qt, that we get from a by adding as many 0s to its end as needed. We suppose that the input a' is in a buffer and each time when the machine $\mathcal{M}_{q,n,t}$ asks for an input it gets a word which consists of the next q elements of the $0, 1$ sequence a'. The machine $\mathcal{M}_q$ has only n memory cells if an instruction is given involving a memory cell with address larger then $n-1$, then the machine halts. While $\mathcal{M}_{q,n,t}$ is working with the input described above, it may give outputs at certain times. At each time the given output is a word of length q. Let g be the concatenation of these $0, 1$-sequences into a single $0, 1$-sequence, in the order as the outputs occurred. This $0, 1$ sequence g will be the value of the random variable ϑ_a. The randomness of ϑ_a is determined by the probabilistic steps of $\mathcal{M}_{q,n,t}$.

The definition of ξ_a. Assume that $a \in A$. The value of ξ_a is a finite $0, 1$-sequence which encodes following information. (a) q, t, n, P, (b) the time sequence $t_0, \ldots, t_j$, when the machine $\mathcal{M}_{q,n,t}$ asks for an input, and (c) the time sequence $t'_0, \ldots, t'_l$ when $\mathcal{M}_{q,n,t}$ gives an output. The choice of the encoding is irrelevant, but the information given in (a),(b), and (c) must uniquely determine ξ_a. $\square$

9.2.2 Leaking RAM with access-pattern adversary

In this section we define a leaking machine which differs form the standard leaking machine at two points (a) its adversary is not the benign adversary but an adversary who at each time knows which instruction is executed and which memory cells are involved in the executed instructions including their roles in it and (b) the starting program P is of special form so that the execution of the program has a cyclical structure as described below. The motivation for this definition is that the result of [3] is equivalent to the statement that such a leaking machine ε'-simulates the standard leaking machine with the benign adversary, where $\varepsilon' = tn^{-\log n}$. First we describe the special properties of the starting program P that we require from such a leaking machine.

Definition. The starting program P is called cyclical if there exists positive integers α, β with β, α so that for all positive integers q, n, t, $q > 10$, $n > \beta$, if the machine $\mathcal{M}_{q,n,t}$ starts with the program $P \circ \langle n, t\rangle$, then for all $\tau \in \omega$ if the machine is running at time τ and $\tau \not\equiv -1 \pmod{\alpha}$ then condition (77) is satisfied, while if $\tau \equiv -1 \pmod{\alpha}$ then condition (78) is satisfied:

(77) (The $\tau \not\equiv -1 \pmod{\alpha}$ case.) *The instruction executed by the machine at time τ is one of the following types: read instruction, write instruction, arithmetic instruction, random number generating instruction. In the execution of such an instruction at time τ only memory cells* $\texttt{cell}(i)$ *with $i < \beta$ are involved. Moreover the name of the instruction executed at time τ and the memory cells involved in it, together with their roles in the instruction, depend only on the residue class of τ modulo α.*

(78) (The $\tau \equiv -1 \pmod{\alpha}$ case.) *At time τ one of the following instructions are executed: the input instruction, the output instruction, the read instruction, the write instruction, the HALT instruction.* $\square$

Remark. The last requirement in condition (77) that "the name of the instruction executed at time τ ... depend only on the residue class of τ modulo α" motivates the expression *cyclical* starting program.

Definition. Assume that q, n, t are positive integers, $q > 10$, $n < 2^q, t < 2^q$ and $P = \langle p_0, \ldots, p_{k-1}\rangle$ is a cyclical initial program. We define a leaking machine $M^{(\text{a.p.})} = M^{(\text{a.p.})}[q, n, t, P] = \langle A, \wp\rangle$ which will be called the leaking RAM with access-pattern adversary, with word length q, memory size n, total time t and initial program P. A will be the set of all $0, 1$ sequences of length at most qt. We define the distributions $\wp_a$ by describing a realization $\langle \vartheta, \xi\rangle$ of the machine M.

The definition of ϑ_a. For each $a \in A$, ϑ_a is identical to the random variable ϑ_a defined for the standard leaking RAM with identical parameters q, n, t, P.

The definition of ξ_a. Assume that $a \in A$. The value of ξ_a is a finite $0, 1$-sequence which encodes the following information. (a) q, t, n, P, (b) for each time τ the name of the instruction which was executed at time τ and the addresses of the memory cells which were involved in the instruction together with their roles in the instruction. (a), and (b) must uniquely determine ξ_a. $\square$

9.2.3 Leaking RAM with parity encoded i/o

The next definition describes a random variable $\pi_{u,q,m,s}$ depending on a $0,1$-sequence u, and positive integers q, m, s. The value of $\pi_{u,q,m,s}$, is a $0,1$-sequence τ, consisting of blocks of length q, each representing in binary form with q bits an element of a $0,1$ sequence σ. σ is the concatenation of blocks of lengths m. Each of these blocks encodes an element of the sequence u: the parity of 1s in the block is even iff the corresponding element of u is 0. The parameter s limits the total length of τ, therefore some elements of u, may not be encoded at all. The random variable $\pi_{u,q,m,s}$ will do the preprocessing of the input for a RAM $\mathcal{M}_q$, that is, it encodes each element of the sequence u by the parity of a $0,1$-sequence of length m. Moreover the elements of this last $0,1$ sequence are represented in binary form by q-bits, which guarantees that at each input instruction of the standard RAM gets only one element of the sequence. This is needed, since otherwise an adversary could get too much information from reading a single input.

Definition. Suppose that q, m, s are positive integers, and $u = \langle u_0, \ldots, u_{j-1} \rangle$ is a finite $0,1$-sequence. We define a random variable $\pi_{u,q,m,s}$. For each fixed $i = 0, 1, \ldots, j-1$ let $v^{(i)} = \langle v_{0,i}, \ldots, v_{m-1,i} \rangle$ be a random $0,1$-sequence of length m so that $\sum_{k=0}^{m-1} v_{k,i} \equiv u_i \pmod 2$, with uniform distribution on the set of all sequences with this property. Assume further that the random variables $v^{(0)}, \ldots, v^{(j-1)}$ are mutually independent. Let $\sigma = \langle \sigma_0, \sigma_1, \ldots, \sigma_{mj-1} \rangle$ be the concatenation of the sequences $v^{(0)}, \ldots, v^{(j-1)}$ in this order. For each σ_i let $\bar{\sigma}_i$ be the binary form of the natural number σ_i given as a $0,1$-sequence of length q. $\langle \tau_0, \tau_1, ..., \tau_{qmj-1} \rangle$ will be the concatenations of the sequences $\bar{\sigma}_0, \sigma_1, ..., \bar{\sigma}_{mj-1}$ in this order. If $k = \min\{qmj-1, s\}$, then the sequence $\tau = \langle \tau_0, \ldots, \tau_{k-1} \rangle$ is the value of the random variable $\pi_{u,q,m,s}$. $\square$

The function $\mathcal{F}_m$, defined below, will do the postprocessing of the output.

Definition. Assume that m is a positive integer. We define a function $\mathcal{F}_m$ on the set of all finite $0,1$ sequences. Assume that $\sigma = \langle \sigma_0, \sigma_1, \ldots, \sigma_{j-1} \rangle$ is a finite $0,1$ sequence. $\mathcal{F}_m(\sigma)$ will be a $0,1$-sequence $u = \langle u_0, \ldots, u_{k-1} \rangle$ where $k = \lceil \frac{j}{m} \rceil$, and for all $i = 0, 1, \ldots, k-1$, u_i is the unique element of the set $\{0,1\}$ so that $u_i \equiv \sum_{r=ik}^{(i+1)k-1} \sigma_r \pmod 2$, where for $r > j-1$ we have by definition $\sigma_r = 0$. $\square$

In the following we define a leaking machine M' which determines its output using an extension of the machine $\mathcal{M}_{q,n,t}[P]$, by adding extra preprocessing of the input and postprocessing of the output, both depending on a parameter m. The adversary will be stronger, than the adversary of the standard RAM. In particular, the adversary, at certain times, will be able to see the contents of the memory cells involved in the instruction executed by $\mathcal{M}_{q,n,t}[P]$ at that time. The adversary will use a probabilistic and adaptive strategy H defined below, to select these times. We will consider only in strategies which guarantee some limit on the number of occasions when the adversary can get this extra information.

Definition. A strategy for the adversary is a function H which assigns to each pair $\langle i, x \rangle \in \omega \times \{0,1\}^{<\infty}$ a random variable $H_{i,x}$ so that the random variables in the set $\texttt{set}(H) = \{H_{i,x} \mid i \in \omega, x \in \{0,1\}^{<\infty}\}$ are mutually independent and each takes its values in $\{0,1\}$. $\square$

Remark. Assume that a machine $\mathcal{M}_q$ is working and an adversary may get some information about the states of the machine. For this the adversary will use a strategy $H_{i,x}$ in the following way. The adversary may get the contents of the memory cells involved in the instruction executed at certain times. The adversary has to select the times, when this happens, these will be called the compromised times. Suppose that at time s the information that reached the adversary till that time, is the $0,1$-sequence $g(s)$. Then the adversary takes a random value of the random variable $H_{s,g(s)}$, independently of the randomizations done in the machine and the earlier decisions of the adversary. If the value of $H_{s,g(s)}$ is 1, then the adversary gets the contents of the memory cells involved in the instruction executed at time s, if $H_{s,g(s)} = 0$ then the adversary does not get this information. $\Box$

Definition. Assume that q, n, t, m are positive integers, $q > 10$, $n < 2^q, t < 2^q$, P is a sequence $\langle p_0, \ldots, p_{k-1}\rangle$ where each p_i is an integer in $[0, 2^q - 1]$, and H is a strategy for the adversary. We define a leaking machine $M' = M'[q, n, t, P, m, H] = \langle A, \wp\rangle$, that will be called a parity encoded RAM with word length q, memory size n, total time t, initial program P, parity encoding length m, and adversary strategy H. The input set A will be the set of all $0,1$ sequences of length at most qt. We define the distributions $\wp_a$ by describing a realization $\langle \vartheta, \xi\rangle$ of the machine M'.

The definition of ϑ_a. Assume that $a \in A$. Let $\bar{a}$ be a random value of $\pi_{a,q,m,t}$, and let $\langle \vartheta^{(M)}, \xi^{(M)}\rangle$ be a realization of the leaking machine $M[q, n, t, P \circ \langle m\rangle]$ defined earlier. ϑ_a is defined as $\mathcal{F}_m(\vartheta^{(M)}_{\bar{a}})$. (That is, by the preprocessing we get $\bar{a}$ for the input a, then the the machine $\mathcal{M}_{q,n,t}[P]$, provides its output at input $\bar{a}$ as the value of the random variable $\vartheta^{(M)}_{\bar{a}}$, and finally by postprocessing of this output we get $\mathcal{F}_m(\vartheta^{(M)}_{\bar{a}})$.)

The definition of ξ_a. Assume that $a \in A$. The value of ξ_a is a finite $0,1$-sequence which encodes the following information. This information will be given by the values of a function $\mathbf{g}$ defined on $\{0, 1, \ldots, t-1\}$, with the intuitive meaning that the information that reaches the adversary before time s is encoded by $\mathbf{g}(s)$, where the time is the time of machine $\mathcal{M}_{q,n,t}[P]$ while it computes the value of $\vartheta_{\bar{a}}$. (Naturally $\mathbf{g}(s)$ also depends on the history of the machine $\mathcal{M}_{q,n,t}[P]$ before time s and the decisions of the adversary made before this time.) We define $\mathbf{g}(s)$ by recursion of s. $\mathbf{g}(0)$ contains q, n, t, m, P. Assume that $s \in \{0, 1, \ldots, t-1\}$ and $\mathbf{g}(0), \ldots, \mathbf{g}(s-1)$ has been already defined. We define now $\mathbf{g}(s)$. The following information will be encoded in $\mathbf{g}(s)$.

(a) s, the name of the instruction that was executed by $\mathcal{M}_{q,n,t}[P]$ at time s, during the computation value of $\vartheta_{\bar{a}}(M)$, and the addresses of the memory cells involved in the instruction executed at time s together with their roles in the instruction,

(b) if $H_{s,\mathbf{g}(s)} = 1$, then the contents of the memory cells whose addresses were given in (a) with the indication which content belongs to which address. Such an s will be called a compromised time.

As in the case of $M[q, n, t, P]$ the choice of the encoding is irrelevant, but the information described above must uniquely determine ξ_a. $\Box$

Remark. We defined the random variables $H_{i,x}$ for all $i \in \omega$, $x \in \{0,1\}^{<\infty}$, instead of defining $H_{i,x}$ only to those pairs i, x where the adversary may need the value of $H_{i,x}$. This does not cause any problems since a strategy defined on only certain pairs $\langle i, x\rangle \in \omega \times \{0,1\}^{<\infty}$ always can be extended in a trivial way to the whole set $\omega \times \{0,1\}^{<\infty}$. $\Box$

Definition. 1. Assume that H is a strategy for the adversary. We will say that H is an $\langle \varepsilon, m\rangle$-moderate strategy for the adversary of the parity encoded leaking machine M, if for each time interval I of length m, and for each history of the leaking machine M the number of compromised times in I is at most εm. In other words, if $\mathbf{g}(s)$ encodes the information that reached the adversary before time s, then there exists at most εm integers i of the interval I, so that $H_{i,\mathbf{g}(i)} = 1$.

2. We will say that the adversary strategy $H_{i,x}$ is ε-random, if for all $i \in \omega$, $x \in \{0,1\}^{<\infty}$ we have $\mathtt{prob}(H_{i,x} = 1) \le \varepsilon$. $\square$

Theorem 7 *There exists $c_0, c_1, c_2, c_3 > 0$ such that for all sufficiently small $\varepsilon > 0$ and for all program P_0 there exists a program P_1 with the following properties. Suppose that q, n, t, m are integers, $q > 10$, $n \le 2^q$, $t \le 2^q$, $m \le n$. Then for all leaking machines M and M' if M, M' satisfy conditions* (79) and (80) below, then M' is an ε' simulation of M, where $\varepsilon' = c_0 \max\{tn^{-\log n}, te^{-c_1 m}\}$.

(79) *M is a standard leaking RAM with the benign adversary, with word length q, memory size n, total time t, and initial program $P_0 \circ \langle n, t\rangle$,*

(80) *M' is a parity encoded RAM with world length q, memory size $n(\log n)^{c_2} m^{c_3}$, total time $t(\log n)^{c_2} m^{c_3}$, initial program $P_0 \circ \langle m, n, t\rangle$, and parity encoding length m, with an $\langle \varepsilon, m\rangle$-moderate adversary.*

We will first prove the theorem in the modified form where in condition (80) we replace the expression "$\langle \varepsilon, m\rangle$-moderate strategy" by "$\varepsilon$-random strategy".

10 The composite machine and a related leaking machine

We have already defined the composite machine in section 3.5, and using that machine we define now a leaking machine $M_C = \langle A, \wp\rangle$ that we will call the composite leaking machine. As in the earlier definitions of leaking machines we define M_C by defining a realization $\langle \vartheta, \xi\rangle$ of it. In the earlier examples we defined the random variables ϑ_a through computation done by the RAM $\mathcal{M}_q$. In this case instead of $\mathcal{M}_q$ the composite machine will do the computation.

Definition. Assume that q, c, n, t are positive integers, $q > 10$, $n < 2^q, t < 2^q$, P is a sequence $\langle p_0, \ldots, p_{c-1}\rangle$ where p_i is an integer $\in [0, 2^{q-1}]$ for $i = 0, 1, ..., c-1$, and H is a strategy for the adversary. We define a leaking machine $M_C = \langle A, \wp\rangle$, that will be called the composite leaking machine with word length q, *RAM*-size c, memory size n, total time t, initial program P. A will be the set of all $0, 1$ sequences of length at most t. We define the distributions $\wp_a$ by describing a realization $\langle \vartheta, \xi\rangle$ of the machine M_C.

The definition of ϑ_a. Assume that $a \in A$ and the machine $\bar{\mathcal{M}}_{q,c}$ starts to work in a state where for all $i = 0, 1, \ldots, c-1$ the content of the memory cell i is p_i. We suppose that the input a is in a buffer and each time when the machine $\mathcal{A}_n$ asks for an input it gets a bit which is the next element of the $0, 1$ sequence a. While $\mathcal{C}_{q,c,n}$ is working with the input described above, $\mathcal{A}_n$ may give outputs at certain times. At each time the given output is a bit. Let ρ be the $0, 1$-sequence, containing these bits in the order as they were given as outputs. This $0, 1$ sequence ρ will be the value of the random variable ϑ_a. The randomness of ϑ_a is determined by the probabilistic steps (the instruction RANDOM) of $\mathcal{A}_n$.

The definition of ξ_a. The adversary gets the state of the machine $\bar{\mathcal{M}}_{q,c}$ at each time, while the composite machine $\mathcal{C}_{q,c,n} = \langle \mathcal{A}_n, \bar{\mathcal{M}}_{q,c} \rangle$ computes the value of ϑ_a. $\Box$

Remark. The state of $\bar{\mathcal{M}}_{q,c}$ is a function which assigns to each memory cell to its content. Therefore the adversary knows exactly what happens in $\bar{\mathcal{M}}_{q,c}$ but does not get directly any information about the machine $\mathcal{A}_k$. The state of $\bar{\mathcal{M}}_{q,c}$ however determines at each time, the name of the instruction performed in $\mathcal{A}_n$, and the memory cells are involved in it. $\Box$

The proof of Theorem 7 will be based on the repeated use of Lemma 37. Namely we will construct leaking machines M_0, M_1, M_2, M_3, M_4 so that if M, M' are the leaking machines of Theorem 7, then $M_0 = M$, $M_4 = M'$ and M_{i+1} is an $\frac{\varepsilon'}{4}$-simulation of M_i. The total time and memory size in each step will grow by at most of a factor of $(m \log n)^{c'}$, where c' is a constant, so altogether their increase remain below the bound required in Theorem 7. The following Theorem **A** is an immediate consequence of the results of [2]. In the two lemmas formulated after Theorem **A** we define M_0, M_2, M_3 and M_4 and formulate the statements that M_{i+1} is an ε' simulation of M_i.

Definition. In the following the word program will mean a finite sequence of natural numbers, which can be interpreted as a program for $\mathcal{M}_q$ for all sufficiently large integers q. $\Box$

Theorem A. *There exists $c_2 > 0, c_3 > 0$ such that for all programs P_0, there a program P_1 with the following properties. Suppose that $q > 10$, n, t are positive integers, $n \leq 2^q$, $t \leq 2^q$. For all leaking machines M_0 and M_2, if M_0, M_1 satisfy conditions* (81) and (82) below, then M_1 is an ε'-simulation of M_0, where $\varepsilon' = tn^{-\log n}$.

(81) *M_0 is a standard leaking RAM with benign adversary, word length q, memory size n, total time t, and initial program $P_0 \circ \langle n, t \rangle$,*

(82) *M_1 is a leaking RAM with an access-pattern adversary, word length q, memory size $n(\log n)^{c_2}$, total time $tn(\log n)^{c_3}$, and initial program $P_1 \circ \langle n, t \rangle$*

For the proof of this theorem see [2]. The fact that the initial program P_1 can be chosen in a way that it is cyclical can be proved in the following way. Assume that the simulated protected CPU, that we will call simply CPU, consists of the first β memory cells of $\mathcal{M}_q$, where β is a constant. The work of the CPU is organized into cycles of length α, where α is a constant. In each cycle the first $\alpha - 1$ instructions involves only memory cells from the CPU and the last instruction is used for communication with the outside word and with the memory unit. As it is explained in the Remark after Lemma 13 in [2] we may assume that the first $\alpha - 1$ operations are either arithmetic/read/write operations or the random number generator instruction and this way P_1 satisfies the requirements of the definition of a cyclical initial program.

Lemma 38 *There exists $c_2 > 0, c_3 > 0$ such that for all programs P_0, there exist a $c > 0$ and a program P_1 such that for all sufficiently small $\varepsilon > 0$ the following holds. Suppose that q, n, t are integers, $q > 10$, $n \leq 2^q$, $t \leq 2^q$. For all leaking machines M_1 and M_2, if M_1, M_2 satisfy conditions* (79) and (80) below, then M_2 is an 0-simulation of M_1.

(83) *M_1 is a leaking RAM with an access-pattern adversary, word length q, memory size n, total time t, and initial program $P_0 \circ \langle n, t \rangle$*

(84) *M_2 is a composite machine with word length q, RAM size c, memory size at most $n(\log n)^{c_2}$, total time at most $\lfloor t(\log n)^{c_3} \rfloor$, and initial program $P_1 \circ \langle n, t \rangle$.*

Lemma 39 *There exists $c_1 > 0, c_2 > 0, c_3 > 0$ such that for all programs P_0, there exists a $c > 0$ and a program P_1 such that for all sufficiently small $\varepsilon > 0$ the following holds. Suppose that q, n, t, m are integers, $q > 10$, $n \leq 2^q$, $t \leq 2^q$, $m \leq n$. For all leaking machines M_2 and M_3, if M_2, M_3 satisfy conditions* (79) and (80) below, then M_3 is an ε'-simulation of M_2, where $\varepsilon' = te^{-c_1 m}$.

(85) *M_2 is a composite machine with word length q, RAM size c, memory size n, total time t, initial program $P_1 \circ \langle n, t \rangle$.*

(86) *M_3 is a parity encoded RAM with world length q, memory size $n(\log n)^{c_2} m^{c_3}$, total time $t(\log n)^{c_2} m^{c_3}$, initial program $P_2 \circ \langle n, t, m \rangle$, and parity encoding length m, with an ε-random adversary.*

Lemma 40 *There exists $c_2, c_3, c_4, c_5 > 0$ so that for all programs P_2 there exists a program P_3 such that for all sufficiently small $\varepsilon > 0$ the following holds. Suppose that q, n, t, m are integers, $q > 10$, $n \leq 2^q$, $t \leq 2^q$, $m \leq n$. For all leaking machines M_3 and M_4, if M_3, M_4 satisfy conditions* (79) and (80) below, then M_4 is an ε'-simulation of M_3, where $\varepsilon' = te^{-c_5 m}$.

(87) *M_3 is a parity encoded RAM with world length q, memory size n, total time t, initial program $P_2 \circ \langle n, t, m \rangle$, and parity encoding length m, with an ε-random adversary.*

(88) *M_4 is a parity encoded RAM with world length q, memory size $n + c_4$, total time $m^{c_4} t$, initial program $P_3 \circ \langle n, t, m \rangle$, and parity encoding length m, with an $\langle \varepsilon, m \rangle$-moderate adversary.*

11 Proof of Lemma AP2

Assume that M_1 is a leaking RAM with an access-pattern adversary, with word length q, memory size n, total time t, and initial program $\langle P_0 \circ \langle n, t \rangle \rangle$. The composite machine M_2, simulating M_1, will work in the following way. The memory size k of the semi-RAM $\mathcal{A}_k$ will be $k = qn + (\log n)^{c'}$, where $c' > 0$ is a constant. This way each bit stored in the memory of M_1 can be represented by the content of a memory cell x_l of $\mathcal{A}_k$. E.g. the jth bit of memory cell `cell`(i) of M_1 can be stored in x_{2+qi+j}. The term 2 is needed to keep the memory cells x_0, x_1, of $\mathcal{A}_k$ free, since they have a special role in the instructions of $\mathcal{A}_k$. The remaining $(\log n)^{c'}$ bits of $\mathcal{A}_k$ will be used as a working memory for various tasks.

Let α, β be the integer whose existence is stated in the definition of a cyclical starting program applied to P_1. According to condition (77) of a cyclical starting program, the sequence of the names and parameters of the instructions executed at times $s\alpha, s\alpha + 1, ..., (s + 1)\alpha - 2$ for $s = 0, 1, ...$ by the machine M_1 do not depend on s. The names of these insructions and parameters are all natural numbers which remain below a constant bound, since by condition (77) all of the memory cells involved in these instructions are among the first β cells. Let S be a sequence encoding these information. It is possible to write the starting program P_2 in a way that it contains the sequence S, and therefore when M_2 starts, then the memory of $\bar{\mathcal{M}}_{q,c}$ contains the sequence S. Therefore we may write the program P_1 in a way that $\bar{\mathcal{M}}_{q,c}$ changes the contents of the bits in the cells x_i in the same way as the corresponding instruction from the sequence S would change it in the memory of M_1, that is, $\bar{\mathcal{M}}_{q,c}$ maintains a correct representation of the memory of M_1 in $\mathcal{A}_k$ all the time.

Of course a single instruction usually requires the change of $\bar{c}\log n$ bits for some constant $\bar{c}$. Since condition (77) restricts the instructions in the sequence S into read, write, arithmetic, and random number generating instructions, $\bar{\mathcal{M}}_{q,c}$ is able to give $\mathcal{A}_k$ instructions which inside the memory of $\mathcal{A}_k$ performs this instructions using computations with individual bits. The time of the computation used to execute each individual instruction of S may be a constant power of $\log n$ but not more. For the random number generating instruction the RANDOM instruction of $\mathcal{A}_k$ is used. During the computation a needed for a single instruction of S the mentioned working memory of $\mathcal{A}_k$ consisting of $(\log n)^{c'}$ bits is used. This way the computation done by M_1 is simulated by M_2 for each time τ with $\tau \not\equiv -1 \pmod{\alpha}$.

Assume now that $\tau \equiv -1 \pmod{\alpha}$. P_2 is written in a way that $\bar{\mathcal{M}}_{q,c}$ counts the time of M_1 so it knows that such a τ has been reached. The content of the memory of M_1 is represented in $\mathcal{A}_k$. First $\bar{\mathcal{M}}_{q,c}$ determines the content κ of the instruction pointer of M_1 at time τ. In this case "determines" means that κ must be available as the content of a memory cell of $\bar{\mathcal{M}}_{q,c}$. This can be done by using the input instruction of $\bar{\mathcal{M}}_{q,c}$ which puts the content of x_0 into `cell`(0). This way bit by bit $\bar{\mathcal{M}}_{q,c}$ may recreate the content of of the instruction pointer of M_1 as a single integer. (The number of M_2 instructions used for this task will be power constant of $\log n$.) Being the the instruction pointer of M_1 available in the memory of $\bar{\mathcal{M}}_{q,c}$, now $\bar{\mathcal{M}}_{q,c}$ may determine which instruction out of the five possibilities must be executed and what are the corresponding parameters (if any), using the input instruction of $\bar{\mathcal{M}}_{q,c}$ the same way as in the case of the instruction pointer.

In the possession of the name of the instruction y and its parameters, the machine $\bar{\mathcal{M}}_{q,c}$, using its output instruction, may change the representation of M_1 in $\mathcal{A}_k$ in the way as the instruction y would do it in M_1. Since the instruction is limited to the four choices listed in condition (78) this does not involve anything else then moving individual bits in the memory of $\mathcal{A}_k$ and possibly perform the INPUT or OUTPUT instructions of $\mathcal{A}_k$ or the HALT instruction of $\bar{\mathcal{M}}_{q,c}$. All of the described activity increases the time of simulation, compared to the computation done by M_1, by only a factor of $(\log n)^{c''}$, and needs no more extra working memory than the one described earlier. This completes the description of the machine M_2.

In the described simulation there is no possibility for errors that is if $\wp_a^{(i)}$ is the distribution of the output for input a for machine M_i, $i = 1, 2$ then the distance of $\wp_a^{(0)}$ and $\wp_a^{(1)}$ is 0. We claim that during this simulation the access-pattern adversary of M_1 gets exactly the same information as the adversary of the composite machine. Indeed the adversary for the composite machine knows the content of the memory cells of $\bar{\mathcal{M}}_{q,c}$ all the time. These contents determine which instruction was simulated in $\mathcal{A}_k$ and the memory cells involved in it together with their roles. No other information reaches $\bar{\mathcal{M}}_{q,c}$ since the input instruction which passes information from $\mathcal{A}_k$ to $\bar{\mathcal{M}}_{q,c}$ was used only to transfer the bits of the items mentioned above. Moreover the timing of all of the instructions done by $\bar{\mathcal{M}}_{q,c}$ is uniquely determined by the mentioned information which is available for the access-pattern adversary. Therefore $\texttt{bias}_{M_1}(a,b) = \texttt{bias}_{M_2}(a,b)$ for all inputs a, b. Consequently M_2 is a 0-simulation of M_1. *Q.E.D.*(Lemma 38)

12 Proof of Lemma 39

According to the definition of a parity encoded RAM the realization of the random variable $\vartheta_a^{(M_3)} = \vartheta_a^{(M_3')}$ will be computed on a machine $\mathcal{M}_q = \mathcal{M}_q[P_2 \circ \langle n, t, m \rangle]$ which gets each bit a_i of the input a in the form of a $0, 1$ sequence of length m whose parity is a_i. The program P_2 will be defined later.

The computation done by $\mathcal{M}_q$, while computing the values of $\vartheta_a = \vartheta'_a$, will be organized in the following way. A block B of mn memory cells will be used to represent each state of the machine $\mathcal{A}_n$ during this computation, which simulates the computation done by the composite machine. The memory cells in B are partitioned into n blocks $B_0, \ldots, B_{n-1}$ each containing exactly m memory cells. The content of each of these memory cells will be always 0 or 1. For each $i = 0, 1, \ldots, n-1$, $S_r(i) \in \{0, 1\}$ will denote the least nonnegative residue of the sum of the contents of the memory cells in B_i modulo 2, at $\mathcal{M}_q$-time r. The state of $\mathcal{A}_n$ at $\mathcal{A}_n$-time r is a function $\varphi_r \in \texttt{func}(\{0, 1, \ldots, n-1\}, \{0, 1\})$, where $\varphi_r(i)$ is the content of memory cell x_i of $\mathcal{A}_n$ at $\mathcal{A}_n$-time r. Such a state of $\mathcal{A}_n$ will be represented in $\mathcal{M}_q$ by the contents of the memory cells in $B = \bigcup_{i=0}^{n} B_i$. Namely, if the representation of the state φ_r of $\mathcal{A}_n$ is completed in $\mathcal{M}_q$ by $\mathcal{M}_q$-time $\lambda(r)$, then $\varphi_r(i) = S_{\lambda(r)}(i)$ for $i = 0, 1, \ldots, n-1$. The RAM part $\bar{\mathcal{M}}_{q,c}$ of the composite machine will be directly represented in the memory of the machine $\mathcal{M}_q$ in c consecutive memory cells. The set of these cells will be denoted by Φ_0. (Naturally Φ_0 and B are disjoint).

During the proof we will use the following notation: $\hat{P}_1 = P_1 \circ \langle n, t \rangle$ and $\hat{P}_2 = P_2 \circ \langle n, t, m \rangle$, where P_1 is given by M_2 and we have to define P_2 in a way that satisfies the requirements of the lemma.

The initial program $\hat{P}_2$ will be given such that $\mathcal{M}_q$ works in the following way. $\mathcal{M}_q$ starts to execute the program $\hat{P}_1$ the same way as it would have been executed in the machine $\bar{\mathcal{M}}_{q,c}$. $\hat{P}_1$ is executed the same way as in $\bar{\mathcal{M}}_{q,c}$ but using the memory cells in Φ_0, with the exception of the output and the input instructions. Recall that an output of $\bar{\mathcal{M}}_{q,c}$ in the composite machine is always an instruction of $\mathcal{A}_n$ (possibly together with a parameter), and this instruction, say u, is immediately executed in $\mathcal{A}_n$. Therefore when $\hat{P}_2$ notices that $\hat{P}_1$ gives an output, then $\hat{P}_2$ executes the corresponding instruction on the representation of $\mathcal{A}_n$, stored in $B = \bigcup_{i=0}^{n-1} B_i$. That is, if, after executing u in $\mathcal{A}_n$, the memory cell x_i of $\mathcal{A}_n$ at $\mathcal{A}_n$-time r contains is $\varphi_r(i)$, then $\hat{P}_2$ has to change the contents of the memory cells in B to reflect the change of state in $\mathcal{A}_n$. As we have indicated already, the $\mathcal{M}_q$-time when this change is completed will be denoted by $\lambda(r)$. The new contents of the memory cells in B must satisfy the equations $\varphi_r(i) = S_{\lambda(r)}(i)$ for $i = 0, 1, \ldots, n-1$.

If $\hat{P}_1$ executes an input instruction then, simulating the definition of the input instruction in $\bar{\mathcal{M}}_{q,c}$, $\hat{P}_2$ does the following. $\hat{P}_2$, using the memory cells in B, determines what is the content of the memory cell x_0 of $\mathcal{A}_n$ according to the current representation of the memory of $\mathcal{A}_n$ in B, and then writes this value in the first memory cell of Φ_0 which represents cell 0 of $\bar{\mathcal{M}}_{q,c}$.

It is easy to define the program $\hat{P}_2$ which accomplishes this task in itself. We want to do it however in a way, that an ε_1-random adversary, does not gain essentially more information about the input than the adversary of the composite machine, during the corresponding computation.

In order to make the definitions and the proof simpler, we assume the following. When $\hat{P}_2$ starts to work, it writes a random $0,1$ bit into each memory cell of B, so that these random bits are chosen independently and with uniform distribution from $\{0,1\}$. (We may avoid this initialization, by treating always separately those memory cells of B which has not been used yet and those which has been already used.)

Definition. 1. According to our earlier definition, the block of memory cells B consists of n consecutive smaller blocks $B_0,\ldots,B_{n-1}$ each containing m memory cells. The sequence $\langle B_0,...,B_{n-1}\rangle$ will be denoted by $\mathcal{B}$. The memory cells in B_i will be denoted by $b_{i,0},b_{i,1},\ldots,b_{i,m-1}$ in their natural order. When we say that the contents of the memory cells in B have an ε-cylindrical distribution we mean that they have an ε-cylindrical distribution with respect to the sequence $\mathcal{B}$. $\Box$

Now we complete the definition of the program $\hat{P}_2$ with the properties described above. Assume that while simulating the composite machine, the program $\hat{P}_2$ took the machine $\mathcal{M}_q$ in a state representing the state of $\mathcal{A}_n$ at $\mathcal{A}_n$-time s. By the definition of the function λ, such a representation is completed by $\mathcal{M}_q$-time $\lambda(s)$, and we have $\varphi_s(i) = S_{\lambda(r)}(i)$ for $i = 0,1,\ldots,n-1$. Now $\hat{P}_2$ continues its simulation of the machine $\bar{\mathcal{M}}_{q,c}$. Using the representation of $\bar{\mathcal{M}}_{q,c}$ given in the memory cells of Φ_0, $\hat{P}_2$ continues this until it reaches an output instruction or an input instruction.

First we consider the case of an output instruction. Recall that an output instruction of $\bar{\mathcal{M}}_{q,c}$ is used to give an instruction to $\mathcal{A}_n$, possibly together with a parameter. Therefore $\hat{P}_2$ knows which instruction will be executed by $\mathcal{A}_n$ at $\mathcal{A}_n$-time $r+1$. We describe now for each instruction separately how will $\hat{P}_2$ change the contents of the memory cells of B so that when this change is finished by $\mathcal{M}_q$-time $\lambda(r+1)$ the contents of these cells represent the state φ_{r+1} of $\mathcal{A}_n$, that is, $\varphi_{r+1}(i) = S_{\lambda(r+1)}(i)$ for $i = 0,1,\ldots,n-1$.

We describe now the action of $\hat{P}_2$ in $\mathcal{M}_q$ corresponding to each instruction of $\mathcal{A}_n$. In some of these instructions we will say that $\hat{P}_2$ computes the output of one of the circuits C_i, $i = 0,1,2,3$ defined in Lemma 3, for a given input. We need now only the $F = F_2$ special case of Lemma 3. $\hat{P}_2$ evaluates the circuit C_i in the following way. When $\hat{P}_2$ starts to work, it computes $\texttt{graph}(C_i)$ together with its labeling for $i = 0,1,2,3$. $\hat{P}_2$ also decides an order in which the gates of the circuit C_i will be used for its evaluation. $\hat{P}_2$ stores the rank of each node according to this ordering. Another, arbitrarily chosen, ordering $\leq_{\texttt{edge}}$ is fixed for the set of the all edges of $\texttt{graph}(C_i)$. $\hat{P}_2$ stores the rank of each edge according to the ordering $\leq_{\texttt{edge}}$. All of this is done only once, and, according to Lemma 3, the required time is polynomial in m and the results can be stored in a polynomial number of memory cells of $\mathcal{M}_q$. Assume that for each node x of the graph C_i, $\hat{P}_2$ reserves a memory cell $\mathbf{n}_x$, where the output of the gate corresponding to this node will be written during an evaluation of the circuit C_i. All of this initial activity of $\hat{P}_2$ is uniquely determined by the choice of the circuits C_i independently of the input, so even if it is completely known to the M_3-adversary it does not give him any additional knowledge compared to the M_2-adversary.

All of the circuits C_i have one or two input blocks, we consider now only the two input blocks situation, the circuits with one input block can be handled in a similar way. In each case when $\hat{P}_2$ will have to evaluate a circuit C_i, the bits of the two input blocks will be given as the contents of the blocks of memory cells B_j and B_k for some $j, k \in \{0, 1, ..., n-1\}$, that is in the cells $b_{j,0}, \dots, b_{j,m-1}$ and $b_{k,0}, \dots, b_{k,m-1}$. $\hat{P}_2$ copies the contents of these cells to the corresponding input nodes of C_i, that is, to the cells $\mathbf{n}_x$, where x runs over the deterministic input nodes of C_i. After copying the content of the cell $b_{j,s}$, and $b_{k,s}$, $s = 0, 1, ..., m-1$, $\hat{P}_2$ immediately erases their contents by writing 0s into them.

The circuits C_i may have probabilistic input nodes as well. For each probabilistic input node x of C_i, $\hat{P}_2$ generates a random $0, 1$-bit with the random generator of $\mathcal{M}_q$ and writes it into $\mathbf{n}_x$. After that $\hat{P}_2$ evaluates all of the gates of the circuit in the usual way. Suppose that x is a gate so that edges are pointing from y and z to x. Then, while evaluating x, $\hat{P}_2$ reads the contents of $\mathbf{n}_y$, $\mathbf{n}_z$ performs the mod 2 operation corresponding to the gate x and writes the result into $\mathbf{n}_x$. Therefore, to evaluate gate x, program $\hat{P}_2$ needs only a constant number of instructions. These instructions will be performed in a time interval J_x of constant length. These intervals clearly can be chosen with the following properties:

(89) *there exists an absolute constant $c' > 0$, so that for each gate x, we have $|J_x| \leq c'$.*

(90) *the intervals J_x are determined independently of the input of the circuit C_i, and in each interval J_x if the value associated with a node $y \in \mathtt{set}(C_i)$ is used then $y \in \mathtt{full}(\{x\})$.*

Immediately after a node x is used for the last time, $\hat{P}_2$ erases the content of $\mathbf{n}_x$. $\hat{P}_2$ knows that it was the last time that x is used since the ranks in the ordering $\leq_{\mathtt{edge}}$ can be computed by $\hat{P}_2$, in constant time.

Now we describe what $\hat{P}_2$ does at various $\mathcal{A}_n$ instructions.

$\mathcal{A}_n$-instruction INPUT. $\hat{P}_2$ asks for input m times, and as soon as the jth input bit arrives, $\hat{P}_2$ writes it in the memory cell $b_{0,j}$. This guarantees that after this procedure the m bit is in block B_0 representing the single bit content of memory cell x_0 in $\mathcal{A}_n$.

Instruction OUTPUT. $\hat{P}_2$ evaluates the circuit C_0 using the content of memory cells $b_{0,0}, \dots, b_{0,m-1}$ as input. The first output block will be the new content of the memory cells $b_{0,0}, \dots, b_{0,m-1}$. (The second output block is not used.) For each $j = 0, 1, ..., m-1$, when during the described process, $\hat{P}_2$ accesses first the memory cell $b_{0,j}$, it gives gives its content as output.

Instruction WRITE i. $\hat{P}_2$ evaluates the circuit C_0 using the content of memory cells $b_{0,0}, \dots, b_{0,m-1}$ as input. The first output block will be the new content of the memory cells $b_{0,0}, \dots, b_{0,m-1}$ the second output block will be the new content of $b_{i,0}, \dots, b_{i,m-1}$

Instruction READ i. The same as instruction WRITE i, only the roles of blocks B_0 and B_i are reversed.

Instruction NEGATION. $\hat{P}_2$ adds 1 (in the field F_2) to the content of $b_{0,0}$. After that $\hat{P}_2$ evaluates the circuit C_0 using the content of memory cells $b_{0,0}, \dots, b_{0,m-1}$ as input. The first output block will be the new content of the memory cells $b_{0,0}, \dots, b_{0,m-1}$. (The second output block is not used.)

Instruction AND. $\hat{P}_2$ evaluates the circuit C_2 using the content of memory cells $b_{1,0}, \dots, b_{1,m-1}$ as the input in the first input block and the contents of the memory cells $b_{0,0}, \dots, b_{0,m-1}$ in the second input block. The first output block will be the new content of the memory cells $b_{1,0}, \dots, b_{1,m-1}$ the second output block will be the new content of $b_{0,0}, \dots, b_{0,m-1}$

Instruction EXCLUSIVE OR. Same as instruction AND, but using circuit C_3.

Instruction RANDOM. For each $j = 0, 1, \ldots, m-1$, $\hat{P}_2$ selects a random $0, 1$ bit using the random number generator of $\mathcal{M}_q$ and then writes the result in cell $b_{0,j}$.

Instruction REFRESH. $\hat{P}_2$ evaluates the circuit C_0 using the content of memory cells $b_{0,0}, \ldots, b_{0,m-1}$ as input. The first output block will be the new content of the memory cells $b_{0,0}, \ldots, b_{0,m-1}$. (The second output block is not used.)

This completes the description of the action of $\hat{P}_2$ when it notices that $\hat{P}_1$ executes an output instruction. Suppose now that the $\hat{P}_2$ notices that $\hat{P}_1$ executes an input instruction. In this case $\hat{P}_2$ adds the content of the memory cells $b_{0,0}, \ldots, b_{0,m-1}$ modulo 2 and writes the result into the first cell of Φ_0 (which represents cell 0 of $\bar{\mathcal{M}}_{q,c}$). After that $\hat{P}_2$ executes a REFRESH instruction as described above. This completes the definition of the program $\hat{P}_2$. Now we can complete the proof of Lemma 39.

Definition. Assume that and adversary $\mathcal{X}$ for a machine M has the knowledge $\kappa(i)$ (represented as a $0, 1$-sequence) at time i, and η is a random variable. We will say that the distribution of η from the point of view of the adversary $\mathcal{X}$ at time i is D, if the conditional distribution of η conditioned by the knowledge of the adversary at time i is D. More precisely let κ_0 be a possible value of $\kappa(i)$. We will say that the distribution of the random variable η from the point of view of the adversary $\mathcal{X}$ with knowledge κ_0 at time i is D, if the conditional distribution of η with the condition that $\kappa(i) = \kappa_0$ is D. When we will say that "with a probability of p the distribution of η from the point of view of adversary $\mathcal{X}$ at time i has property P", we will mean, that with a probability p, for the randomizations by the machine M and the adversary $\mathcal{X}$ together, $\kappa(i)$ takes a value κ_0 so that the distribution of η, from the point of view of the adversary $\mathcal{X}$, with knowledge κ_0 at time i, has property P. $\Box$

We want to describe the distribution of the contents of the memory cells of $\mathcal{M}_q$, from the point of view of the M_3-adversary, that is, the ε_1-random adversary, at $\mathcal{M}_q$-time $\lambda(i)$ for all $i = 0, 1, \ldots$. It will be easier to describe this conditional distribution if we add some extra knowledge to the knowledge of the M_3-adversary. We will call this new adversary, whose knowledge will be defined later, the strong M_3-adversary. We will show that the statement of Lemma 39 holds even if we consider the machine M_3 with the strong M_3-adversary, who always knows at least as much as the original M_3-adversary. Clearly this will imply the lemma in its original form.

In the following description we will use several real numbers as parameters. The adversary will be an ε_1-random adversary. We will show that the contents of certain memory cells in the machine maintain an ε_0-cylindrical distribution from the point of view of the adversary with a probability of at least $p = e^{-c_1 m}$. We will assume that $0 < \varepsilon_1 \ll c_1 \ll \varepsilon_0 \ll 1$.

The definition of the strong M_3-adversary. We define the knowledge of the strong M_3-adversary at $\mathcal{M}_q$-time $\lambda(i)$ by recursion on i. We want to define this knowledge in a way that

(91) *the knowledge of the strong M_3-adversary at $\mathcal{M}_q$-time $\lambda(i)$ implies the knowledge of the M_2-adversary at $\mathcal{A}_n$-time i.*

In addition to this we will require also the following. The distribution of the contents of the memory cells from the point of view from the strong M_3-adversary at $\mathcal{M}_q$-time i satisfies the following conditions with a probability of at least $1 - ie^{-c_1 m}$:

(92) *The contents of all of the memory cells of $\mathcal{M}_q$ outside B at $\mathcal{M}_q$-time $\lambda(i)$ are uniquely determined, that is, each has a fixed value which is taken with probability* 1,

(93) *the contents of the memory cells in B at $\mathcal{M}_q$-time i has ε_0 cylindrical distribution with respect to the sequence $\mathcal{B}$.*

(94) *The distribution of the input of the machine M_2 till $\mathcal{A}_n$-time i from the point of view of the M_2-adversary is the same as its distribution from the point of view of the strong M_3-adversary.*

The recursive definition is the following. Assume that the knowledge of the strong M_3-adversary has been already defined so that for conditions (91), (92), (93), are satisfied with $i := i - 1$. We include all of the knowledge of the M_2 adversary at time i into the knowledge of the strong M_3-adversary at time $\lambda(i)$. Consequently condition (91) is satisfied.

The definition of $\hat{P}_2$ implies that between time $\lambda(i-1)$ and $\lambda(i)$, $\hat{P}_2$ first simulates $\bar{\mathcal{M}}_{q,c}$ using the memory cells of Φ_0 until it reaches an input or an output instruction. The $\mathcal{M}_q$-time, immediately before such an input or output instruction is executed, will be denoted by $\lambda_1(i)$. Till $\lambda_1(i)$, during the computation by $\hat{P}_2$, the memory cells of B are not used at all. Since this part of the computation is deterministic, whatever happens is uniquely determined by the contents of the memory cells of $\mathcal{M}_q$ outside of B at $\mathcal{M}_q$-time $\lambda(i-1)$. Therefore, by condition (92), the knowledge of the strong M_3-adversary at $\mathcal{M}_q$-time $\lambda(i-1)$ uniquely determines the contents of the memory cells of $\mathcal{M}_q$ outside of the set B at time λ_{i-1}.

To continue the definition of the knowledge of the strong M_3-adversary, we have to distinguish several cases according to the input or output instruction that is reached by program $\hat{P}_2$ at $\mathcal{M}_q$-time $\lambda_1(i) + 1$.

Case I. *An output instruction (of $\hat{P}_1$ working in $\bar{\mathcal{M}}_{q,c}$) is reached by $\hat{P}_2$ at $\mathcal{M}_q$-time $\lambda_1(i)+1$, so that while simulating the corresponding $\mathcal{A}_n$ instruction $\hat{P}_2$ has to evaluate one of the circuits C_i of Lemma 3 as described in the definition of $\hat{P}_2$.* In this case the inductive assumption (93) implies that with a probability of at least $(i-1)e^{-c_1 m}$ from the point of view of the strong M_3-adversary at $\mathcal{M}_q$-time $\lambda(i-1)$, the contents of $\mathcal{B}$ has ε_0 cylindrical distribution with respect to $\mathcal{B}$. Let $\bar{G}$, be the base of this ε_0-cylindrical distribution. Therefore using the fact that the circuits C_j of Lemma 3 are $(\varepsilon_0, \varepsilon_1, e^{-c_1 m})$-cylindrical, we get the following. The input of the circuit C_j used in this step also has ε_0-cylindrical distribution (from the point of view of the strong M_3-adversary at time $\lambda_1(i)$) since we get it form the distribution of the contents of B, by using only one or two blocks. Let G be the restriction of $\bar{G}$ to these blocks. According to the definition of an $(\varepsilon_0, \varepsilon_1, p)$-cylindrical circuit, there exists a set $X \subseteq \texttt{set}(C_j)$ so that if D is the set of nodes x of C_j so that at least one elements of the time interval J_x is compromised and the strong adversary gets the output of the gate at each node in the set $Y = G \cup \texttt{full}(D) \cup X$, then the distribution of the output of the circuit C_j from the point of view of the M_3-adversary at time $\lambda(i)$ is ε_0-cylindrical. (We are using here properties (89) and (90) from the definition of intervals J_x.) Naturally the output of the circuit C_j corresponds only at most two elements of the sequence $\mathcal{B}$. However the contents of the cells in the remaining elements of $\mathcal{B}$ has not been used during the evaluation of C_j so the independence requirements for the ε_0-cylindricity for the contents of the cells in B with respect to $\mathcal{B}$ remain valid. Therefore condition (93) holds for i as well, with the definition of the knowledge of the strong adversary that we described.

We also note that all of the partial results for the evaluation of the circuit C_j is erased by $\hat{P}_2$ during the evaluation of C_j and therefore, in the present "Case I.", the contents of all of the memory cells of $\mathcal{M}_q$ not contained in B remain uniquely determined by the knowledge of the strong M_3-adversary at $\mathcal{M}_q$-time $\lambda(i-1)$ and so by its knowledge at $\mathcal{M}_q$-time $\lambda(i)$ as well.

Case II. *An output instruction (of $\hat{P}_1$ working in $\bar{\mathcal{M}}_{q,c}$) is reached by $\hat{P}_2$ at $\mathcal{M}_q$-time $\lambda_1(i)+1$, so that the corresponding $\mathcal{A}_n$ instruction that $\hat{P}_2$ has to simulate is the instruction* INPUT. In this case, by definition, the strong M_3-adversary gets the same new information as the ε_1-random adversary. In this case the incoming bits are random with uniform distribution with the condition that their sum is the corresponding input bit of M_2. This obviously remains true even from the point of view of the strong M_3-adversary since the randomizations in the preprocessing remains hidden from the ε_1-random adversary. Therefore conditions (91), (92) (93), (94) are trivially satisfied. (The handle of the new $(\varepsilon_0, \varepsilon_1, p)$ cylindrical distribution in block B_0 can be the function determined by the contents of those memory cells $b_{0,j}$, $j = 0, 1, ..., m-1$ which were accessed during a compromised instruction.)

Case III. *An output instruction (of $\hat{P}_1$ working in $\bar{\mathcal{M}}_{q,c}$) is reached by $\hat{P}_2$ at $\mathcal{M}_q$-time $\lambda_1(i)+1$, so that the corresponding $\mathcal{A}_n$ instruction that $\hat{P}_2$ has to simulate is the instruction* RANDOM. Similar to Case II.

Case IV. *An input instruction (of $\hat{P}_1$ working in $\bar{\mathcal{M}}_{q,c}$) is reached b y $\hat{P}_2$ at $\mathcal{M}_q$-time $\lambda_1(i)+1$.* In this case the sum (in F_2) a of the contents of the memory cells $b_{0,0}, ..., b_{0,m-1}$ will be copied into the cell of Φ_0 representing $\texttt{cell}(0)$ of $\bar{\mathcal{M}}_{q,c}$. a will be known to the M_2-adversary and so by definition to the M_3-adversary and the strong M_3-adversary. According to the definition of $\hat{P}_2$ the change in Phi_0 is immediately followed by the simulation of a REFRESH instruction of $\mathcal{A}_n$ which uses the circuit C_0 and so in the way as we have described above, restores the $(\varepsilon_0, \varepsilon_1, ie^{-c_1})$-cylindricity of the distribution of the contents of the memory cells in B with respect to $\mathcal{B}$. *Q.E.D.*(Lemma 39)

13 Proof of Lemma 40.

During the proof we will use the following notation: $\hat{P}_2 = P_2 \circ \langle n, t, m\rangle$ and $\hat{P}_3 = P_3 \circ \langle n, t, m\rangle$, where P_2 is given by M_3 and we have to define P_3 in a way that satisfies the requirements of the lemma.

We define the starting program $\hat{P}_3$ in the following way. The machine M_4, which is $\mathcal{M}_q[\hat{P}_3]$ with preprocessing and postprocessing will simulate the machine $\mathcal{M}_q[\hat{P}_2]$, also with preprocessing and postprocessing, where $\mathcal{M}_q[P]$ denotes the machine $\mathcal{M}_q$ with the starting program P.

$\hat{P}_3$ simulates each instruction of $\hat{P}_2$. Between the simulations of two consecutive instructions of $\hat{P}_2$, program $\hat{P}_3$ may have to do other type of computation. To make these interruptions possible, a single instruction of $\hat{P}_2$ will be simulated by several instructions of $\hat{P}_3$. We may assume that each instruction of $\hat{P}_2$ is simulated by exactly $\bar{c}$ instruction of $\hat{P}_3$, where the integer $\bar{c} > 0$ is an absolute constant. We partition the time used by $\hat{P}_3$ into intervals $J_0, J_1, ...$ of length $\bar{c}$. Assume that i is a natural number so that either $i = 0$ or $\hat{P}_3$ has completed the simulation of an instruction of $\hat{P}_2$ in the time interval J_{i-1}. Suppose also that the next instruction of $\hat{P}_2$ to be simulated is x. $\hat{P}_3$ will select an integer $k \geq 1$ and simulate x in the interval J_{i+k}. For the selection of k $\hat{P}_3$ starts to generate a random sequence of integers $g_0, g_1, ... \in [0, m-1]$ according to the following rules. The various integers g_j are taken from the interval $[0, m-1]$ independently and with uniform distribution. g_l is generated in the time interval J_{i+l}, and in this time interval $\hat{P}_3$ checks whether $g_l = 0$. If $g_l = 0$ then $k = l + 1$, $\hat{P}_3$ stops generating the elements of the sequence $g_0, g_1, ...$, and in the time interval $J_{i+k} = J_{i+l+1}$, $\hat{P}_3$ simulates instruction x of $\hat{P}_2$. If $g_l \neq 0$ then $\hat{P}_3$ continues generating the sequence $g_0, g_1, ...$ by generating g_{l+1} in the time interval J_{l+1}. $\hat{P}_3$ continues this until it either determines the value of k, or reaches the limit on its total time. This completes the definition of $\hat{P}_3$. Note that, if the limit on the total time used is larger by a factor of m^2 for M_4 than for M_3, then, with a probability of at least $1 - e^{-c_6}m$, $\hat{P}_3$ will be able to complete the simulation of $\hat{P}_2$ where $c_6 > 0$ is a constant.

We claim that an (ε, m)-moderate adversary $\mathcal{D}$ of $\mathcal{M}_q[\hat{P}_3]$ with high probability gets no more information than a suitably chosen $2\bar{c}\varepsilon$-random adversary $\mathcal{R}$ of $\mathcal{M}_q[\hat{P}_2]$. Assume such an adversary $\mathcal{D}$ is fixed and we define a corresponding adversary $\mathcal{R}$. The definition of $\mathcal{R}$ is the following. From the point of view of $\mathcal{R}$ and instruction x executed by $\hat{P}_2$ will be compromised iff there exists a positive integer x, so that $\hat{P}_3$ simulates x in the time interval J_i and J_i contains at least one compromised time from the point of view of the (ε, m)-moderate adversary $\mathcal{D}$.

We have to show that $\mathcal{R}$ is $2\bar{c}\varepsilon$-random, that is for each fixed instruction x of $\hat{P}_3$ the probability p_x that x is compromised will be at most $2\bar{c}\varepsilon$ from the point of view of $\mathcal{R}$. Assume that either $i = 0$ or the instruction of $\hat{P}_2$ which is before x, has been simulated already by $\hat{P}_3$ in the time interval J_i. We consider the strategy of $\mathcal{D}$ only after time interval J_i and till the time x is simulated. $\mathcal{D}$ may use a probabilistic strategy for choosing the compromised times. We prove first that for each fixed deterministic strategy of $\mathcal{D}$ we have $p_x \leq \varepsilon$, the general case is an immediate consequence of this. Assume now that a deterministic strategy S is fixed for $\mathcal{D}$.

Let X be the set of all natural numbers l with the following property.

(95) *If the information that $\mathcal{D}$ has at the end of time interval J_{l-1} does not contradict to the statement "$g_0, g_1, ..., g_{l-1}$ are all different from 0", then following strategy S, $\mathcal{D}$ will declare an element of the time interval $J_{i+l} \cup J_{i+l+1}$ compromised.*

It is sufficient to show that the probability of $k \in X$ is at most $2\bar{c}\varepsilon$, where x is simulated time interval J_{i+k}. Indeed if an instruction of J_{i+k} is compromised then, since $g_0 \neq 0, ..., g_{k-2} \neq 0$, then $\mathcal{D}$ either declares an element of the interval J_{i+k-1} compromised, so $l \in X$ because of that, or $\mathcal{D}$ does not declare any element of I_{i+k-1} compromised but declares an element of J_{i+k} compromised. In this latter case $g_0 \neq 0, ..., g_{k-2} \neq 0, g_{k-1} \neq 0$ is consistent with the knowledge of $\mathcal{D}$ at the end of interval J_{i+k-1}, and an element of J_{i+k} is compromised, therefore we have again $k \in X$.

We estimate $\mathtt{prob}(k \in X)$ using Bayes' theorem. For all $j = 0, 1, ...$, let A_j be the event that $k \in [j\frac{m}{\bar{c}}, (j+1)\frac{m}{\bar{c}} - 1]$. It is sufficient to show that for each fixed j we have $\mathtt{prob}(k \in X \mid A_j) \leq 2\bar{c}\varepsilon$. The definition of an (ε, m) moderate adversary implies that for each j the time interval $\bigcup\{J_{i+r} \mid r \in [j\frac{m}{\bar{c}}, (j+1)\frac{m}{\bar{c}} - 1]\}$ may contain at most εm compromised instruction and therefore there exists at most $\bar{c}\varepsilon m$ integer $r[j\frac{m}{\bar{c}}, (j+1)\frac{m}{\bar{c}} - 1]$ such that J_{i+r} contains a compromised time. Since for each $l \in X$, either J_{i+l} or J_{i+l+1} contains a compromised time we have that $X \cap [j\frac{m}{\bar{c}}, (j+1)\frac{m}{\bar{c}} - 1] \leq 2\bar{c}\varepsilon m$. Since for each fixed $l \in X$ the probability of $k = l$ is at most $\frac{1}{m}$ (since the probability if $g_j = 0$ is at most $\frac{1}{m}$) we get that $\mathtt{prob}(k \in X \mid A_j) \leq 2\bar{c}\varepsilon \leq 2\bar{c}\varepsilon$. *Q.E.D.*(Lemma 40)

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.

[2] M. Ajtai. Oblivious *RAM*s without cryptographic assumptions. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:28, 2010.

[3] M. Ajtai. Oblivious *RAM*s without cryptogrpahic assumptions. In L. J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 181–190. ACM, 2010.

[4] I. Damgård, S. Meldgaard, and J. B. Nielsen. Perfectly Scure Oblivious *RAM* without Random Oracles. TCC 2011, pages 144-163.

[5] S. Faust, T. Rabin, L. Reyzin, E. Tromer, and V. Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In H. Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, 2010.

[6] O. Goldreich. Towards a theory of software protection and simulation by oblivious *RAM*s. In *STOC, Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, 25-27 May 1987, New York City, NY, USA*, pages 182–194. ACM, 1987.

[7] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious *RAM*s. *J. ACM*, 43(3):431–473, 1996.

[8] S. Goldwasser and G. N. Rothblum. Securing computation against continuous leakage. In T. Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 59–79. Springer, 2010.

[9] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[10] S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In M. Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[11] R. Ostrovsky. Efficient computation on oblivious *RAM*s. In *STOC, Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, 14-16 May 1990, Baltimore, Maryland, USA*, pages 514–523. ACM, 1990.

[12] R. Ostrovsky. *Software Protection and Simulation on Oblivious RAMs.* PhD thesis, MIT, 1992.

[13] N. Pippenger and M. J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.

[14] N. Alon, J. Spencer *The Probabilistic method*, John Wiley & Sons Inc. New York, 1992. 1993.

[15] Sh. Hoory, N. Linial, A. Wigderson, *Expander graphs and their applications*, Bulletin (New series) of the American Mathematical Society, Vol. 43, Number 4, Oct. 2006, pp. 439-561

[16] D. A. Osvik, A. Shamir, and E. Tromer, *Cache Attacks and Countermeasures: the Case of AES*, Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006,
see also http://people.csail.mit.edu/tromer/papers/cache.pdf

[17] C. Rackoff and D. Simon, *Cryptographic defense against traffic analysis*, Proceedings of the 25th STOC, pp., 672 - 681, 1993

List of frequently used symbols related to block circuits.

$\texttt{func}(A,B)$ is the set of all functions defined on A with values in B, p. 25.
$\texttt{size}(C)$ is the number of nodes in the circuit C, p. 25.
$\texttt{set}(C)$ is the set of nodes of the circuit C, p. 25.
$\texttt{inset}(C)$ is the set of input nodes of the circuit C, p. 25.
$\texttt{outset}(C)$ is the set of output nodes of the circuit C, p. 25.
$\chi_\delta^{(C)}$ is the evaluation function of the circuit C at input δ, p. 25.
$\leq_C$ is the natural partial ordering on $\texttt{set}(C)$, where the input nodes are the maximal elements, p. 25.
$\texttt{rand}(C)$ is the set of probabilistic input nodes of a circuit C p. 26.
$\texttt{detin}(C)$ is the set of deterministic input nodes of a circuit C, $\texttt{inset}(C) = \texttt{rand}(C) \cup \texttt{detin}(C)$, p. 26.
$\mathbf{h}_W(a)$ is the sequence whose ith element is $\sum_{x\in W_i} a(x)$, where $a \in \texttt{func}(\bigcup W_i, F)$, p. 26.
$\mathbf{h}_W(f,i) = \sum_{x\in W_i} f_i(x)$, where $f = \langle f_0, ..., f_{s-1},\ f_i \in \rangle W_i, F)$, p. 28.
$\xi_{x,C}$ the ouput of the probabilistic F-circuit as a random variable at input x, p. 26.
$\eta_{a,C}$ the ouput of the probabilistic block F-circuit (icluding preprocessing and postprocessing) as a random variable at original input a, p. 27.
$\texttt{full}(X)$ the set of nodes which either belong to X or immediately above an element of X in $\leq_C$, p. 27.
$\texttt{seq}(W,F)$ is the set of all sequences whose elements ith element is defined on W_i with values in F, p. 28.
$\texttt{extension}_F(f,B,u)$ is the set of all extensions g of f onto B with $\sum_{x\in B} g(x) = u$. p. 28
$\texttt{ext}(H)$ is the natural extension of a distribution on $\texttt{func}(\texttt{detin}(C), F)$ onto $\texttt{func}(\texttt{detin}(C) \cup \texttt{rand}(C), F)$, p. 29.
$\texttt{DET}(b)$ is the restriction of an input b from $\texttt{inset}(C) = \texttt{detin}(C) \cup \texttt{rand}(C)$ onto $\texttt{detin}(C)$, p. 29.
$\texttt{cond}_{\texttt{output}}(H,f)$, $\texttt{cond}_{\texttt{detin}}(H,f)$ if the input b is arriving with distribution H, this is the conditional distribution of the input/output with the condition that $\chi_b^{(C)}$ is identical to f on $\texttt{domain}(f)$, p. 29.
$\texttt{pow}(X)$ is the powerset of X, p. 31.
$\texttt{adv}_C(\varepsilon, \mathcal{S})$ is the ε-random adversary for the circuit C strengthened by the compromised set $\mathcal{S}(X)$, p. 31.
$\mathcal{Y}^+$ is the adversary $\mathcal{Y}$ with the strengthened with the knowledge of the deterministic input, p. 31.
$\mathcal{D}_{\mathcal{X},H}$ is the distribution of the knowledge of adversary $\mathcal{X}$ with input distribution H. $\mathcal{D}_{\mathcal{Y},H,X}$ is the same with the condition that X is the random subset. p. 32.
$\texttt{func}_\theta(A,B)$ set of all functions from $\texttt{func}_\theta(W,B)$, where $W = \langle W_0, ..., W_{s-1}\rangle$ whose domain intersect each W_i in a set of size at most $\theta|W_i|$, p. 32.
p_a, $a \in \texttt{set}(C)$, are the polynomials asscociated with the nodes of the block F-circuit C, p 33.
$P_0, ..., P_{k-1}$ are the input block polynomials, p 33.
$Q_0, ..., Q_{k-1}$ are the output block polynomials, p 33.
$a \ll b$ means a is sufficiently small with respect to b p. 34.

$C_i^{(m)}$ are the block circuits functionally equivalent to the various gates as defined in Lemma 3, p. 34.

$\mathtt{block_{in}}(C)$, $\mathtt{block_{out}}(C)$ set of input/output blocks of C, $\mathtt{block_{io}}(C) = \mathtt{block_{in}}(C) \cup \mathtt{block_{out}}(C)$, p. 39.

$\mathcal{I}(\delta)$ is the system of equations $\{P_i = \delta_i \mid i = 0, 1, ..., k-1\}$, p. 39.

$\mathcal{E}(g)$ is the system of equations $\{p_a = g(a) \mid a \in A\}$, p. 39.

$p_b^{(g)}$, modified form of p_a with $p_b^{(g)} = g(b)$ for all $b \in \mathtt{domain}(g)$, p. 40

$\sigma^{(\mathtt{out})}/\sigma^{(\mathtt{io})}$ for $\sigma \in \mathtt{func}(\mathtt{inset}(C), F)$. It is the restriction of $\chi_\sigma^{(C)}$ to $\mathtt{outset}(C)/\mathtt{detin}(C) \cup \mathtt{outset}(C)$, p. 43.

$\bar{\sigma}^{(\mathtt{in})}(J)/\bar{\sigma}^{(\mathtt{out})}(J)$ is the sum of the values of $\sigma^{(\mathtt{in})}/\sigma^{(\mathtt{out})}$ on the input/output block J, p. 43.

$f \natural g$ is identical to g on $\mathtt{domain}(g)$ and identical to f on $\mathtt{domain}(f)\backslash\mathtt{domain}(g)$, p. 42.

$\mathbf{C}_{m,G}^{(=)}$ is the copying circuit, p. 44.

$\mathtt{pre}_G(C), \mathtt{post}_{G(C)}$: copying circuit is applied before/after the circuit C, p. 56.

$\mathbf{C}_{m,G}^{(=,2)}$ is the doubling circuit, p. 57.

$\mathbf{C}_{m,G}^{(j)}$, $j = 0, 1$ is a block circuit computing the conant j, p. 57.

$\mathbf{C}_{m,G}^{(+)}, \mathbf{C}_{m,G}^{(+)}, \mathbf{C}_{m,G}^{(+)}$ are block circuits computing the correpsonding operations, p. 57.

$\mathbf{C}_{m,G}^{(\mathtt{rand})}$ is the random generator circuit, p. 58.

$\mathbf{S}\alpha$ is the sum of the elements of the sequence $\alpha = \langle \alpha_0, ..., \alpha_{m-1} \rangle$, p. 58.

$\mathbf{C}_{m,G}^{(\mathtt{spl})}$ is the splitting circuit, p. 58.

$\mathtt{sequence}_{m,F}$ is the set of all sequences of length m with elements in F, p. 58.

$\mathbf{C}_{m,G}^{(\mathbf{S})}$ is the total sum circuit, p. 60.

$\mathbf{C}_{m,G}^{(\mathbf{S}^{-1})}$ is the total sum circuit, p. 61.

$\mathbf{C}_{m,G}^{(\mathtt{leaky}\times)}$ is the leaky product circuit, p. 61.

$\mathtt{perc}(\Phi, A, G_x)$ is the set of all $C \in \Phi$ so that C is percolative with respect $A \cap \mathtt{set}(C)$and G_x, p. 61.

$\mathbf{C}_{m,G}^{(+,\times)}$ is a blockcircuit computing $a(b_0 + b_1)$, p. 69.