Tamper Resilient Circuits: The Adversary at the Gates

Aggelos Kiayias and Yiannis Tselekounis

Department of Informatics and Telecommunications,
National and Kapodistrian University of Athens

Abstract. We initiate the investigation of *gate*-tampering attacks against cryptographic circuits. Our model is motivated by the plausibility of tampering directly with circuit gates and by the increasing use of *tamper resilient gates* among the known constructions that are shown to be resilient against *wire-tampering* adversaries. We prove that gate-tampering is *strictly* stronger than wire-tampering. On the one hand, we show that there is a gate-tampering strategy that perfectly simulates any given wire-tampering strategy. On the other, we construct families of circuits over which it is impossible for any wire-tampering attacker to simulate a certain gate-tampering attack (that we explicitly construct). We also provide a tamper resilience impossibility result that applies to both gate and wire tampering adversaries and relates the amount of tampering to the depth of the circuit. Finally, we show that defending against gate-tampering attacks is feasible by appropriately abstracting and analyzing the circuit compiler of Ishai et al. [18] in a manner which may be of independent interest. Specifically, we first introduce a class of compilers that, assuming certain well defined tamper resilience characteristics against a specific class of attackers, can be shown to produce tamper resilient circuits against that same class of attackers. Then, we describe a compiler in this class for which we prove that it possesses the necessary tamper-resilience characteristics against gate-tampering attackers.

Keywords: tamper resilient circuits, attack modeling.

1 Introduction

Traditionally, cryptographic algorithms are designed under the assumption that adversaries have *black box* access to the algorithms' implementation and private input. In this setting, the adversary chooses an input, supplies the algorithm with it, receives the corresponding output, and it is not allowed to alter the algorithm's internals during its execution. This mode of interaction is usually being modeled as a security game (e.g., chosen-ciphertext attack against an encryption scheme or chosen message attack against a digital signature) and the underlying cryptographic scheme is proven secure based on it. In reality though, besides observing the algorithms' input-output behaviour, an adversary may also land physical attacks on the algorithm's implementation. For instance, she may learn the secret key of an encryption scheme by measuring the power consumed by the device during the encryption operation [23], or by measuring the time needed for the encryption to complete [22]. Besides *passive* attacks, the class of *active* attacks includes inducing faults to the computation [4,6,22], exposing the device to electromagnetic radiation [14,28,29], and several others [17,24,21,1,7,30]. Such attacks have proven to be a significant threat to the real-world security of cryptographic implementations.

1.1 Related Work and Motivation

The work of [18] followed by [11,9] undertook the difficult task of modeling and defending against adversaries that tamper directly with the implementation circuit. In this setting the adversary is given access to a circuit equipped with secret data stored in private memory; it is allowed to modify a bounded number of circuit wires and/or memory gates in each circuit invocation. The objective is to suitably modify the circuit operation so that tampering gives no (or -at least-bounded) advantage to the adversary.

In [18] the adversary is allowed to tamper with a bounded number of wires or memory gates in each computation, and for each component she may *set* its value to 1, *reset* it to 0, or *toggle* its value. The tampering effect can be persistent, i.e., if the value of a circuit wire or memory gate is modified during one run, it remains modified for all subsequent runs. Hence, the adversary can tamper with the entire circuit by persistently tampering with a bounded number of wires in each run. The proposed compiler, which is parameterized by $t \in \mathbb{N}$, transforms any boolean circuit C into C', where C' realizes the same functionality with C and is secure against adversaries who tamper with up to t of its wires in each computation, i.e., any adversary who tampers with up to t circuit wires of C' in one circuit invocation, cannot learn anything more about the circuit's private information than an adversary having *black-box* access to C. Formally, this notion is captured by the following *simulation-based* security definition: for every PPT adversary $\mathcal{A}$ tampering with C', there exists a simulator $\mathcal{S}$ having black-box access to C such that the output distribution of $\mathcal{A}$ and $\mathcal{S}$ are indistinguishable. The construction is based on a randomized secret sharing scheme which shares the bit-value of a wire in C among k wires, and then introduces redundancy by making $2kt$ copies of each wire, where k denotes the security parameter. The randomized encoding guarantees that any tampering with C' will produce an invalid encoding with high probability, triggering the circuit's self-destruction mechanism that erases the circuit's secret memory. Since this mechanism is also prone to tampering, the adversary could try to deactivate it so as to tamper with the rest of the circuit while keeping the secret state intact. In order to prevent such a scenario, C' incorporates an error-propagation mechanism which permeates the circuit and propagates errors induced by tampering attacks. The size of C' is larger than C by a factor of $O(t \cdot \log^3(\frac{1}{\epsilon}))$ where ϵ represents the simulation error.

In [11] the authors consider a different adversarial model, in which the adversary is allowed to tamper with every circuit wire, but each tampering attempt fails with probability $\delta \geq 0$ (*noisy* tampering). Moreover, they put forward a relaxed security definition in which the simulator does not have black-box access to the circuit, but requires logarithmically many bits of information about the circuit's secret memory. The resulting circuit is augmented by a $O(\delta^{-1}\log(\frac{1}{\epsilon}))$ factor for the simulation to fail with probability at most ϵ. Furthermore, it uses no randomness during execution and consists of subcircuits which perform computations over *Manchester encodings*, which encode a single bit into four bits. For each subcircuit, the compiler randomly encodes the 0,1-bits to elements in $\{0,1\}^4\backslash\{0000\}$, and employs *tamper-proof* gates that handle computations over these encodings. If the inputs are invalid, the gates output 0000 and the error propagates to the self-destruction mechanism, which is similar to the one employed in [18] but uses some additional tamper-proof gadgets. Besides error propagation and memory erasure, the self-destruction mechanism verifies that all subcircuits produce consistent outputs. Hence, in order to alter the computation effectively, an adversary needs to tamper with all k subcircuits in a way such that (i) all attacks produce valid, probably different due to randomization, encodings, and (ii) the encodings must produce the same decoded output. As it is proved in [11], this happens with negligible probability in k.

The adversarial model considered in [9] is similar to [18]. The main difference is that now persistent tampering is not allowed on circuit wires and, similarly, to [11] the simulator is allowed a logarithmic amount of leakage from the computation. Regarding the construction, [9] combines *error-correcting codes* and *probabilistically checkable proofs of proximity* (PCPP) in the following way: the circuit's secret state $\mathbf{s}$ and input $\mathbf{x}$ are encoded into $\mathbf{S}$ and $\mathbf{X}$, respectively. Then the transformed circuit computes[1] $y = C_{\mathbf{s}}(\mathbf{x})$ and a PCPP proof π for the validity of the tuple $(y, \mathbf{S} \circ \mathbf{X})$ with respect to the error-correcting code and C. The proof is verified by polynomially many verifiers who output 1 in case of validity, and 0 otherwise, and their output (i) is fed to a (tamper-proof) AND gate with *unbounded* fan-in and fan-out that erases the circuit's secret state if a verifier rejects π, and (ii) together with y they feed a (tamper-proof) AND gate with unbounded fan-in and one output wire, which is the circuit's output wire. If one of the verifiers outputs 0, then the circuit outputs the zero bit. The resulting circuit size is polynomial on the input circuit but a constant ratio of wire tampering can be tolerated.

Besides tampering against circuits' wires or memory gates, some works consider adversaries who tamper exclusively with memory gates [15,25,10,8]. In [15] the authors give an impossibility result on tamper resilience by showing that without using secure hardware an adversary can extract the circuit's private information, by sequentially setting or resetting the memory bits and observing the tampering effects on the circuit output. Apparently, [18,11,9] circumvent the impossibility result by erasing the private information in case of error detection. In [25] the authors consider adversaries who tamper and probe with the circuits' private memory and they give an impossibility result for circuits that do not have access to a source of random bits, with respect to both tampering and probing attacks. [10] introduces the notion of non-malleable codes. Such codes ensure that any adversary who tampers with a codeword, with respect to some specific

[1] Their construction [9] considers circuits that output one bit.

class of tampering functions, will either lead the decoding algorithm to output $\perp$, or output a codeword which is irrelevant to the original word. Moreover, they show how to construct non-malleable codes for specific classes of tampering functions. Finally, in [8] the authors introduce the notion of Built-in Tamper Resilience, which defines security for cryptographic protocols where some of the parties are implemented by hardware tokens that resist tampering attacks.

The above state of the art in tamper resilient circuits suggests a fundamental issue that is a source of theoretical motivation for our work. While tampering circuit wires seems to be a strong adversarial model, recent constructions do heavily exploit tamper-proof gates (e.g., the gate with unbounded high fan-in in [9]). This suggests that *tampering gates directly* might be an even stronger (and possibly in some cases even more plausible) adversarial model; how do wire and gate adversaries fare against each other? and is it possible to protect against both? what are the upper bounds in terms of amount of tampering that can be tolerated? Our work initiates the investigation of gate-tampering attacks and takes steps towards answering all those questions as explained below.

Besides its theoretical interest, our work is also motivated by practical attacks on circuit gates. For example, in [30] it is explained how illumination of a target transistor can cause it to conduct. Transistors are used to implement logical gates so such an optical probe attack will amount to gate tampering in a circuit (effectively changing the gate for another gate). Beyond that, fault injection in the SRAM of an FPGA also results to switching the computation of the FPGA circuit (because the program of the FPGA is stored in memory).

1.2 Our Contributions

Impossibility Results. Informally, the main idea behind our impossibility result (section 3) is the following: we define the notion of *non-triviality* of a cryptographic circuit which attempts to capture the essence of a meaningful cryptographic implementation. According to non-triviality, for every PPT algorithm $\mathcal{A}$ and circuit C with private memory $\mathbf{s}$, where C implements some cryptographic functionality, $\mathcal{A}$ should not be able to learn $\mathbf{s}$, while having black box access to C (observe that if $\mathcal{A}$ learns $\mathbf{s}$ then the implementation becomes obsolete as $\mathcal{A}$ can simulate it). Then we prove that any circuit C that satisfies non-triviality possesses necessarily a *weakly unpredictable bit*, i.e., there exists a secret state bit that cannot be extracted with probability very close to 1, while having black box access to C. Now, let d be the depth of C and assume it consists of gates with fan-in at most 2. If the adversary is allowed to tamper with up to d circuit components (we prove our result for either wires or gates), there exists a strategy that extracts the circuit's unpredictable bit with probability equal to 1. The impossibility result follows from this, since any simulator with black-box access to C only has no capability to predict the unpredictable bitÂă as good as the tampering adversary. The main observation here is that for any $d \in \mathbf{N}$, and every compiler T that receives a circuit C and produces a circuit C' of depth at most d, T cannot be secure against an adversary who tampers with d circuit gates or wires, *regardless of the size of* C'.

It is worth contrasting our result to that of [15], where the authors consider an adaptive adversary who is capable of tampering with private memory bits; by correlating circuit outputs to tampering set/reset operations within the secret-state they show that the whole secret state can be reconstructed. This suggests that the only way to attain tamper resilience of the secret-state is by employing internal integrity detection mechanisms and have the circuit self-destruct in any case of fault detection. With our impossibility result however we show that simulation will inevitably fail even in the presence of error-detection and self-destruction mechanisms in case we allow tampering with up to d circuit components (wires or gates), where d is the circuit's depth. This underlines the strength of tampering with circuit components vs. tampering the secret state.

Gate Adversaries Are Strictly Stronger than Wire Adversaries. We proceed to explore the relationship between gate and wire adversaries. In section 4 we first prove that any tampering attack on up to t circuit wires can be simulated by an adversary who tampers with up to t circuit gates, i.e., for every circuit $C_{\mathbf{s}}$ and any PPT adversary $\mathcal{A}$ who tampers with up to t wires of $C_{\mathbf{s}}$, there exists a PPT adversary $\mathcal{A}'$ who tampers with up to t circuit gates, such that the output of $\mathcal{A}$ and $\mathcal{A}'$ are exactly the same. Then we proceed to prove the other direction which is the most technically involved. Note that in the presence of unbounded fan-out (or fan-in) gates in a circuit it is clear that a gate adversary has an advantage since a wire adversary may be incapable of controlling sufficiently many wires to modify the behavior of the gate. However we demonstrate that even w.r.t. to bounded fan-in/fan-out circuits, gate adversary are strictly stronger. Specifically, we show that there exist a family of circuits $\widetilde{C}_{\tilde{\mathbf{s}}}$ parameterized by n, t and a PPT adversary $\mathcal{A}$ who tampers with up to n circuit gates, such that for all PPT adversaries $\mathcal{A}'$ who tamper with up to t circuit wires (where t can be arbitrarily larger than n), $\mathcal{A}'$ fails to simulate $\mathcal{A}$. Intuitively, the idea behind our proof is the following. We construct a circuit that has a "critical area" comprised of n AND-gates. The input to the critical area is provided by a sub-circuit (referred to as C_1) that implements a PRF, a digital signature and a counter. The output of the critical area is fed to a second sub-circuit (referred to as C_2) that calculates a digital signature and a second counter. The key point is that a gate-adversary can transform all AND-gates of the critical area into XOR-gates. This enables the gate-attacker to produce a circuit output with a certain specific distribution that is verifiable in polynomial-time. The main technical difficulty is assembling the circuits C_1, C_2 suitably so that we can show that no matter what the wire-attacker does, it is incapable of simulating the distribution produced by the gate-attacker. Note that the wire-attacker is fully capable of controlling the input to the C_2 sub-circuit (by tampering with all the output wires of the critical area). Hence by running the circuit several times, the wire-attacker can attempt to learn the proper output distribution of the critical region and feed it to C_2. In our explicit construction, by appropriately assembling the main ingredients of each sub-circuit (PRF, digital signatures and counters) we show that there exists no efficient wire-tampering strategy that simulates the gate-tampering strategy assuming the security of the PRF and the digital signature. The circuit family we construct is executable an unbounded number of times (by either attacker). If one restricts the number of times the implementation can be executed by the tampering attackers (by having the implementation always self-destruct by design after one invocation) then the circuit family can be simplified (due to lack of space we provide this construction in the full version of the paper).

Tamper Resilience against Gate Adversaries. Given our separation result, the question that remains is whether it is possible to defend cryptographic implementations against gate adversaries. Towards that direction, we show (section 5) that gate attackers compromise the security of [18] by effectively eliminating the circuit's randomness, when it is produced by randomness gates, and then we prove that if we substitute those gates with pseudo-random generators, then [18] can be shown to be secure against gate adversaries. Based on [18], we give a general characterization (Definition 10) of a secure class of compilers and we use it in order to present our result in a self-contained way. The way we present our positive result may also be of independent interest: first, we define a class of compilers (Definition 10) that have the property that if they have certain tamper resilience characteristics against an arbitrary class of adversaries, then the circuits that they produce are tamper-resilient against that class of adversaries. Seen in this light, the result of [18] is a specific instance that belongs to this class of compilers that satisfies the basic tamper resilience characteristics of the class against appropriately bounded *wire* adversaries. We proceed analogously to prove that the circuit transformation that removes the randomness gates satisfies the necessary tamper resilience characteristics against appropriately bounded *gate* adversaries. The resulting compiler produces circuits of comparable size to those of [18] however the parameter t in our case reflects the bound on the gate-tampering adversary.

2 Preliminaries

Definition 1 (Circuit). *A Boolean circuit C, over a set of gates $\mathcal{G}$, with n input bits and q output bits, is a directed acyclic graph $C = (V, E)$, such that every $v \in V$ belongs to one of the following sets of nodes:*

- *V_I (**Input**): For all $v \in V_I$, the indegree is zero, the outdegree is greater than 1, and each v represents one input bit. We label these nodes by $\mathsf{i}_1, \ldots, \mathsf{i}_n$.*
- *$V_{\mathcal{G}}$ (**Gate**): Each $v \in V_{\mathcal{G}}$ represents a logic gate in $\mathcal{G}$, and its indegree is equal to the arity of the logic gate. The outdegree is at least 1.*
- *V_O (**Output**): For all $v \in V_O$, the indegree is one and the outdegree is zero, and each v represents one output bit. We label these nodes by $\mathsf{o}_1, \ldots, \mathsf{o}_q$.*

The cardinalities of the sets defined above are n, s and q respectively. Finally, the edges of the graph represent the wires of the circuit. The set of all circuits with respect to a set of gates $\mathcal{G}$, will be denoted by $\mathcal{C}_{\mathcal{G}}$.

The circuit's *private memory* is considered as a *peripheral* component and consists of m additional gates. The value m is the memory's storage capacity in bits.[2] Formally,

Definition 2 (Circuit with Private memory). *A graph C is a circuit with private memory provided that its set of vertices can be partitioned into two sets V, V' such that (i) the graph $C \setminus V'$ is a DAG conforming to definition 1, (ii) there are no edges between nodes in V', (iii) each vertex $v \in V'$ possesses at most one incoming and exactly one outgoing edge. The set V' represents the memory gates of C; in this case, we refer to C as a circuit with private memory V' and we also denote by E' the edges of C that are incident to V'. We denote $|V'| = m$.*

From now on we will use the terms *Circuit* and *Graph* interchangeably, as well as the terms wire/edge.

Definition 3 (Computation). *Let C be a circuit with private memory V' that contains $\mathbf{s} \in \{0,1\}^m$, and let $\mathbf{x} \in \{0,1\}^n$ be the circuit input. The computation $\mathbf{y} = C_{\mathbf{s}}(\mathbf{x})$ consists of the following steps:*

i. ***Memory access****: for each $v \in V'$, the value of v propagates to the outgoing edge.*
ii. ***Breadth-first traversal of** C:*
   - *Assign to each input node i_j, for $j \in [n]$, the value x_j and propagate x_j to its outgoing wires.*
   - *For $v \in V_G$ (gate node), apply the boolean function that corresponds to v on the incoming edges and propagate the result to the outgoing edges.*
   - *Every output node o_i, for $i \in [m]$, evaluates to the incoming value, and determines the value y_i.*

iii. ***Memory update****: every $v \in V'$ is updated according to its incoming value.*

Informally, in each *computation* the circuit receives an input $\mathbf{x}$, produces an output $\mathbf{y}$, and it may also update its private memory from $\mathbf{s}$ to some value $\mathbf{s}'$. From now on, a circuit with secret state $\mathbf{s}$ will be denoted by $C_{\mathbf{s}}$, or by C, if there is no need to refer to $\mathbf{s}$ explicitly.

In the following, we give a generalization of the above definition for multiple rounds and we formally define the adversarial models we consider.

Definition 4 (v-round computation). *Let C be a circuit and* $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ *a pair of random variables. A v-round circuit execution w.r.t.* $\mathbf{Env}$ *is a random variable $(\mathbf{v}, \mathcal{A}^{C(\cdot)}(\mathbf{v}))$ s.t. $\mathcal{A}$ is a polynomial-time algorithm that is allowed to submit v queries to the circuit which is initialized to state* $\mathbf{s}$ *and in each query it performs a calculation over its input as in Definition 3.*

[2] In many circumstances we refer to private memory using the terms *secret state* or *private state*.

In the above definition, $\mathbf{v}$ represents all public information related to private memory $\mathbf{s}$. For instance, if C implements the decryption algorithm of a public-key encryption scheme with secret key $\mathbf{s}$, then $\mathbf{v}$ should contain information such as the length of $\mathbf{s}$ and the corresponding public-key. Now we define the adversary of [18].

Definition 5 (t-wire tampered computation). *Let C be a circuit with private memory V' that contains $\mathbf{s} \in \{0,1\}^m$, and let $\mathbf{x} \in \{0,1\}^n$ be the circuit input. The t-wire tampered computation $\mathbf{y} = C_{\mathbf{s}}^{\mathcal{T}}(\mathbf{x})$ for some tampering strategy $\mathcal{T}$ is defined as follows.*

1. *$\mathcal{T}$ is a set of up to t triples of the form (α, e, p), where e is an edge of C, and α may be one of the following tampering attacks:*
   - ***Set**: set the value of e to 1,*
   - ***Reset**: set the value of e to 0,*
   - ***Toggle**: flip the value of e,*

   and if $p \in \{0,1\}$ is set to 1 then the attack is persistent, i.e., the modification made by α is preserved in all subsequent computations. For non-persistent attacks we write $(\alpha, e, 0)$ or just (α, e).
2. *The computation of the circuit is executed as in definition 3 taking into account the tampering instructions of $\mathcal{T}$.*

Next we define gate-tampered computation.

Definition 6 (t-gate tampered computation). *Let C be a circuit with private memory V' that contains $\mathbf{s} \in \{0,1\}^m$, and let $\mathbf{x} \in \{0,1\}^n$ be the circuit input. The t-gate tampered computation $\mathbf{y} = C_{\mathbf{s}}^{\mathcal{T}}(\mathbf{x})$ for some tampering strategy $\mathcal{T}$ is defined as follows.*

1. *$\mathcal{T}$ is a set of up to t triples of the form (f, g, p), where $g \in V_{\mathcal{G}} \cup V'$, f can be any function $f : \{0,1\}^l \to \{0,1\}$, where l is the arity of the gate represented by g, and p defines persistent (or not) attacks as in definition 5. The tampering substitutes the gate functionality of g to be the function f.*
2. *The computation of the circuit is executed as in definition 3 taking into account the tampering instructions of $\mathcal{T}$.*

Definition 7 (v-round wire (resp. gate) tampered computation). *Let C be a circuit and $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ a pair of random variables. A v-round tampered computation w.r.t. $\mathbf{Env}$ is a random variable $(\mathbf{v}, \mathcal{A}^{C^{*}(\cdot)}(\mathbf{v}))$ s.t. $\mathcal{A}$ is a polynomial-time algorithm that is allowed to submit v queries to the circuit which is initialized to state $\mathbf{s}$ and in the i-th query it performs a wire (resp. gate) tampered computation according to tampering instructions $\mathcal{T}_i$ specified by $\mathcal{A}$. Note that the computation respects persistent tampering as specified by $\mathcal{A}$.*

Notation. We will denote wire and gate adversaries respectively by $\mathcal{A}_{\mathsf{w}}$ and $\mathcal{A}_{\mathsf{g}}$. Moreover, $\mathcal{A}_{\mathsf{w}}^t$ (resp. $\mathcal{A}_{\mathsf{g}}^t$) denotes a wire (resp. gate) adversary who tampers with t circuit wires (resp. gates). The output of a single-round wire (resp. gate) adversary $\mathcal{A}_{\mathsf{w}}$ (resp. $\mathcal{A}_{\mathsf{g}}$) with strategy $\mathcal{T}_{\mathsf{w}}$ (resp. $\mathcal{T}_{\mathsf{g}}$) who performs a tampered computation on C is denoted by $\mathcal{A}_{\mathsf{w}}^{[C,\mathcal{T}_{\mathsf{w}}]}$ (resp. $\mathcal{A}_{\mathsf{g}}^{[C,\mathcal{T}_{\mathsf{g}}]}$). The output of a multi-round adversary A_{g} (resp. A_{w}) is denoted by $\mathcal{A}_{\mathsf{g}}^{\mathcal{C}^*(\cdot)}$ (resp. $\mathcal{A}_{\mathsf{w}}^{\mathcal{C}^*(\cdot)}$). For $n \in \mathbb{N}$, $[n]$ is the set $\{1, \dots, n\}$. The statistical distance between two random variables X, Y, with range $\mathcal{D}$, is denoted by $\Delta(X, Y)$, i.e., $\Delta(X,Y) = \frac{1}{2}\sum_{u \in \mathcal{D}} |\Pr[X = u] - \Pr[Y = u]|$. Finally, if D is a distribution over $\mathcal{D}$, $X \sim D$ indicates that variable X follows distribution D.

3 Impossibility Results

In this section we prove that for any *non-trivial* cryptographic device implemented by some circuit $C \in \mathcal{C}_{\mathcal{G}}$ of depth d, where $\mathcal{G}$ contains boolean logic gates, tamper-resilience is impossible (i) when wire adversaries land $d(k-1)$ non-persistent tampering attacks on the wires of C, where k is the maximum fan-in of the elements in $\mathcal{G}$, and (ii) when gate adversaries non-persistently tamper with d gates of C. In order to do so, we define the notion of *non-triviality*, which characterizes meaningful implementations and then we prove that every non-trivial circuit C possesses a *weakly unpredictable bit* (Lemma 1). Then we define an adversarial strategy $\mathcal{T}$, such that for any $\mathbf{x} \in \{0,1\}^n$, $C^{\mathcal{T}}(\mathbf{x})$ is statistically far from the output of $\mathcal{S}^{C(\cdot)}$, for any PPT algorithm $\mathcal{S}$.

A non-trivial cryptographic device is one that contains a circuit for which no adversary can produce its entire secret-state in polynomial-time when allowed black-box access to it. Formally,

Definition 8 (non-triviality property). *Let* $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ *be a pair of random variables, and let* $C_{\mathbf{s}}$ *be a circuit with secret state* $\mathbf{s}$. *We say that* $C_{\mathbf{s}}$ *satisfies the non-triviality property w.r.t. environment* $\mathbf{Env}$ *if for every PPT algorithm* $\mathcal{A}$ *there exists a non-negligible function* $f(m)$ *such that*

$$\Pr[\mathcal{A}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}) = \mathbf{s}] < 1 - f(m).$$

The above definition is a necessary property from a cryptographic point of view, since its negation implies that the device can be replicated with only black-box access. Thus any attacker can render it redundant by recovering its secret state and instantiating it from scratch. We then focus on specific bits of the secret state. We define a bit to be weakly unpredictable if predicting its value always involves a non-negligible error given black-box access to the device.

Definition 9 (weakly unpredictable bit). *Let* $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ *be a pair of random variables, and* $C_{\mathbf{s}}$ *be a circuit with secret state* $\mathbf{s}$. *Then* $C_{\mathbf{s}}$ *possesses a weakly unpredictable bit w.r.t. environment* $\mathbf{Env}$ *if there exists an index* i, $1 \leq i \leq m$, *such that for every PPT algorithm* $\mathcal{A}$ *there exists a non-negligible function* $\delta(m)$ *such that*

$$\Pr[\mathcal{A}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}) = s_i] < 1 - \delta(m).$$

Armed with the above definitions we demonstrate that any non-trivial circuit possesses at least one weakly unpredictable bit.

Lemma 1. *Let* $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ *be a pair of random variables. Then for every circuit* $C_{\mathbf{s}}$*, if* $C_{\mathbf{s}}$ *satisfies the non-triviality property w.r.t. enviroment* $\mathbf{Env}$*, then* $C_{\mathbf{s}}$ *possesses a weakly unpredictable bit, again w.r.t.* $\mathbf{Env}$.

The above lemma is proved[3] by contradiction: we consider a circuit $C_{\mathbf{s}}$ which satisfies the non-triviality property and none of its bits is a weakly unpredictable bit. Then we construct an algorithm which extracts $\mathbf{s}$ with probability $1 - f(m)$, for some non-negligible function $f(m)$.

We next give the impossibility result for circuits that consist of standard AND, NOT and OR gates, and for the case of wire adversaries. The impossibility is via a construction: we design a specific single round adversary $\mathcal{A}$ that is non-simulatable in polynomial time. The main idea behind the construction is exploiting the tampering instructions so that we correlate the output of the circuit with the weakly unpredictable bit contained in the secret state. Concretely, in the proof of theorem 1 we define a wire adversary $\mathcal{A}_{\mathsf{w}}$ who acts as follows: she targets the weakly unpredictable bit s_i and a path $\mathcal{P}$ from the i-th memory gate to one output gate, say y_j. Then she chooses a tampering strategy $\mathcal{T}_{\mathsf{w}}$ on wires which ensures that s_i remains unchanged during its pass through the circuit gates. For instance, if at some point the circuit computes $g(s_i, x)$, where x is an input or secret state bit, then (i) if g is an AND (resp. OR) gate then $\mathcal{A}_{\mathsf{w}}$ *sets* (resp. *resets*) the wire that carries x, and the circuit computes $\wedge(s_i, 1) = s_i$ (resp. $\vee(s_i, 0) = s_i$). After defining $\mathcal{T}_{\mathsf{w}}$, $\mathcal{A}_{\mathsf{w}}$ executes $C_{\mathbf{s}}^{\mathcal{T}_{\mathsf{w}}}(\mathbf{x})$ for some $\mathbf{x} \in \{0,1\}^n$ of her choice and outputs s_i. The challenge for $\mathcal{S}$ is to output the unpredictable bit with probability close to 1, while having only black box access to $C_{\mathbf{s}}$.

Theorem 1. *(Wire Adversaries - Binary Fan-in) Let* $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ *be a pair of random variables. Then for every circuit* $C_{\mathbf{s}} \in \mathcal{C}_{\mathcal{G}}$ *of depth* $d \in \mathbb{N}$*, where* $\mathcal{G} = \{\wedge, \vee, \neg\}$ *and* $\mathbf{s} \in \{0,1\}^m$*, that satisfies the non-triviality property w.r.t.* $\mathbf{Env}$*, there exists a single round adversary* $\mathcal{A}_{\mathsf{w}}$ *with strategy* $\mathcal{T}_{\mathsf{w}}$*, where* $|\mathcal{T}_{\mathsf{w}}| \leq d$*, such that for every PPT simulator* $\mathcal{S}$ *having black-box access to* $C_{\mathbf{s}}$*, it holds that* $\Delta(\mathcal{S}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}), \mathcal{A}_{\mathsf{w}}^{[C_{\mathbf{s}}, \mathcal{T}_{\mathsf{w}}]}(\mathbf{v})) \geq f(m)$*, for some non-negligible function* $f(m)$.

The above theorem also holds for circuits that contain NAND gates, when the adversary is allowed to tamper with $2d$ circuit wires. Concretely, the adversarial strategy against $g(s_i, x)$, where g is a NAND gate, is the following: $\mathcal{A}_{\mathsf{w}}$ *resets* the wire that carries x and *toggles* the wire that carries s_i. The next corollary generalizes the above theorem for circuits that consist of gates with fan-in greater than two. Consider for example an AND gate with fan-in k, which receives the weakly unpredictable bit, s_i, on some of its input wires. If the adversary *sets* the $k-1$ remaining wires, then the gate outputs s_i.

Corollary 1. *(Wire Adversaries - Arbitrary Fan-in) Let* $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ *be a pair of random variables. Then for every circuit* $C_{\mathbf{s}} \in \mathcal{C}_{\mathcal{G}}$ *of depth* $d \in \mathbb{N}$*, where*

[3] For detailed proofs see the paper's full version.

$\mathbf{s} \in \{0,1\}^m$ and $\mathcal{G} = \{\wedge, \vee, \neg\}$ with bounded fan-in k, that satisfies the non-triviality property, there exists a single round adversary $\mathcal{A}_{\mathsf{w}}$ with strategy $\mathcal{T}_{\mathsf{w}}$, where $|\mathcal{T}_{\mathsf{w}}| \leq d(k-1)$, such that for every PPT simulator $\mathcal{S}$ having black-box access to $C_{\mathbf{s}}$, $\Delta(\mathcal{S}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}), \mathcal{A}_{\mathsf{w}}^{[C_{\mathbf{s}}, \mathcal{T}_{\mathsf{w}}]}(\mathbf{v})) \geq f(m)$, for some non-negligible function $f(m)$.

Finally, we give an impossibility result with respect to gate adversaries. The main idea behind $\mathcal{A}_{\mathsf{g}}$'s strategy in the following theorem, e.g. against an AND gate with fan-in k that receives the weakly unpredictable bit s_i, is the following: $\mathcal{A}_{\mathsf{g}}$ substitutes the AND gate with the function that projects the value of the incoming wire that carries s_i to all outgoing wires.

Theorem 2. *(Gate Adversaries - Arbitrary Fan-in) Let $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ be a pair of random variables. Then for every circuit $C_{\mathbf{s}} \in \mathcal{C}_{\mathcal{G}}$ of depth $d \in \mathbb{N}$, where $\mathbf{s} \in \{0,1\}^m$ and $\mathcal{G} = \{\wedge, \vee, \neg\}$ with bounded fan-in k, that satisfies the non-triviality property, there exists a single round adversary $\mathcal{A}_{\mathsf{g}}$ with strategy $\mathcal{T}_{\mathsf{g}}$, where $|\mathcal{T}_{\mathsf{g}}| \leq d$, such that for every PPT simulator $\mathcal{S}$ having black-box access to $C_{\mathbf{s}}$, $\Delta(\mathcal{S}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}), \mathcal{A}_{\mathsf{g}}^{[C_{\mathbf{s}}, \mathcal{T}_{\mathsf{g}}]}(\mathbf{v})) \geq f(m)$, for some non-negligible function $f(m)$.*

Notice here that $|\mathcal{T}_{\mathsf{g}}|$ does not depend on k.

4 Wire vs. Gate Adversaries

In this section we investigate the relation between wire and gate adversaries of Definitions 5 and 6, respectively. Concretely, we prove that for any boolean circuit $C_{\mathbf{s}}$ and PPT adversary $\mathcal{A}_{\mathsf{w}}^t$ with strategy $\mathcal{T}_{\mathsf{w}}$, $t \in \mathbb{N}$, there exists a PPT adversary $\mathcal{A}_{\mathsf{g}}^t$ with strategy $\mathcal{T}_{\mathsf{g}}$, such that for any tampering action in $\mathcal{T}_{\mathsf{w}}$, there exists an action in $\mathcal{T}_{\mathsf{g}}$ that produces the same tampering effect (Theorem 3). Then we show that the other direction does *not* hold, i.e., we prove that gate adversaries are strictly stronger than wire ones (Theorem 4).

Wire adversaries are subsumed by Gate adversaries. We show that any wire tampering strategy is possible to be simulated by a suitable gate tampering strategy.

Now we briefly discuss the main idea behind Theorem 3, i.e., we describe how the gate adversary simulates tampering attacks on circuit wires w.r.t. to an AND gate g with two input wires, x, y, and one output wire w. So, if $\mathcal{A}_{\mathsf{w}}^t$, $t \geq 3$, *resets* x or y, then $\mathcal{A}_{\mathsf{g}}^{t'}$, $t' \geq 1$, replaces g with the zero function, and if $\mathcal{A}_{\mathsf{w}}^t$ *sets* both x and y, then the gate adversary replaces g with $f(x,y) = 1$. On the other hand, if $\mathcal{A}_{\mathsf{w}}^t$ *sets* x (resp. y) then $\mathcal{A}_{\mathsf{g}}^t$ substitutes g with $f(x,y) = y$ (resp. $f(x,y) = x$). The other cases consider more complex tampering combinations on input and output wires, as well as, tampering with memory gates, and they can be similarly dealt with.

Theorem 3. *Let $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ be a pair of random variables. For every circuit C with gates $\mathcal{G} = \{\wedge, \vee, \neg\}$, $t \in \mathbb{N}$, and any v-round PPT wire adversary $\mathcal{A}_{\mathsf{w}}^t$, there exists a v-round gate adversary $\mathcal{A}_{\mathsf{g}}^l$, for some $l \in \mathbb{N}$, $l \leq t$, such that $\mathcal{A}_{\mathsf{w}}^{C_{\mathbf{s}}^*(\cdot)}(\mathbf{v})$ is identically distributed to $\mathcal{A}_{\mathsf{g}}^{C_{\mathbf{s}}^*(\cdot)}(\mathbf{v})$.*

Gate adversaries are stronger than wire adversaries. Consider a PPT gate adversary $\mathcal{A}_{\mathsf{g}}^t$, $t = 1$, who tampers with an AND or an OR gate g that consists of two input wires x, y, and a single output wire w, by replacing g with some g' that implements one of the 16 possible binary boolean functions[4] $f_i : \{0,1\}^2 \to \{0,1\}$, $i \in [16]$. For each f_i, $i \in [16] \backslash \{7, 10\}$, we give a tampering strategy for $\mathcal{A}_{\mathsf{w}}^t$, $t \leq 3$, that simulates the tampering effect of f_i, for both AND and OR gates. Here we abbreviate the attacks *set, reset* and *toggle* by S, R and T, respectively.

In the following, the variables x, y, w, will denote both circuit wires, as well as their bit-values.

Table 1. All boolean functions from $\{0,1\}^2$ to $\{0,1\}$ and $\mathcal{A}_{\mathsf{w}}^t$'s tampering strategy

	$(0, 0)$	$(0, 1)$	$(1, 0)$	$(1, 1)$	Repr. 1	Repr. 2	$\mathcal{A}_{\mathsf{w}}^t$'s strategy – AND gate	$\mathcal{A}_{\mathsf{w}}^t$'s strategy – OR gate
f_1	1	1	1	1	$1 \wedge 1$	$1 \vee y$	$((\mathsf{S}, x), (\mathsf{S}, y))$	(S, x)
f_2	1	1	1	0	$\neg(x \wedge y)$	$(\neg x \vee \neg y)$	(T, w)	$((\mathsf{T}, x), (\mathsf{T}, y))$
f_3	1	1	0	1	$\neg(x \wedge \neg y)$	$\neg x \vee y$	$((\mathsf{T}, y), (\mathsf{T}, w))$	(T, x)
f_4	1	1	0	0	$\neg x \wedge 1$	$\neg x \vee 0$	$((\mathsf{T}, x), (\mathsf{S}, y))$	$((\mathsf{T}, x), (\mathsf{R}, y))$
f_5	1	0	1	1	$\neg(\neg x \wedge y)$	$x \vee \neg y$	$((\mathsf{T}, x), (\mathsf{T}, w))$	(T, y)
f_6	1	0	1	0	$\neg y \wedge 1$	$\neg y \vee 0$	$((\mathsf{T}, y), (\mathsf{S}, x))$	$((\mathsf{T}, y), (\mathsf{R}, x))$
f_7	1	0	0	1	$x == y$	$(x \wedge y) \vee (\neg x \wedge \neg y)$	—	—
f_8	1	0	0	0	$\neg x \wedge \neg y$	$\neg(x \vee y)$	$((\mathsf{T}, x), (\mathsf{T}, y))$	(T, w)
f_9	0	1	1	1	$\neg(\neg x \wedge \neg y)$	$x \vee y$	$((\mathsf{T}, x), (\mathsf{T}, y), (\mathsf{T}, w))$	*No action*
f_{10}	0	1	1	0	$x \neq y$	$(x \wedge \neg y) \vee (\neg x \wedge y)$	—	—
f_{11}	0	1	0	1	$1 \wedge y$	$0 \vee y$	(S, x)	(R, x)
f_{12}	0	1	0	0	$\neg x \wedge y$	$\neg(x \vee \neg y)$	(T, x)	$((\mathsf{T}, y), (\mathsf{T}, w)$
f_{13}	0	0	1	1	$x \wedge 1$	$x \vee 0$	(S, y)	(R, y)
f_{14}	0	0	1	0	$x \wedge \neg y$	$\neg(\neg x \vee y)$	(T, y)	$((\mathsf{T}, x), (\mathsf{T}, w)$
f_{15}	0	0	0	1	$x \wedge y$	$\neg(\neg x \vee \neg y)$	*No action*	$((\mathsf{T}, x), (\mathsf{T}, y), (\mathsf{T}, w))$
f_{16}	0	0	0	0	$0 \wedge y$	$0 \vee 0$	(R, x)	$((\mathsf{R}, x), (\mathsf{R}, y))$

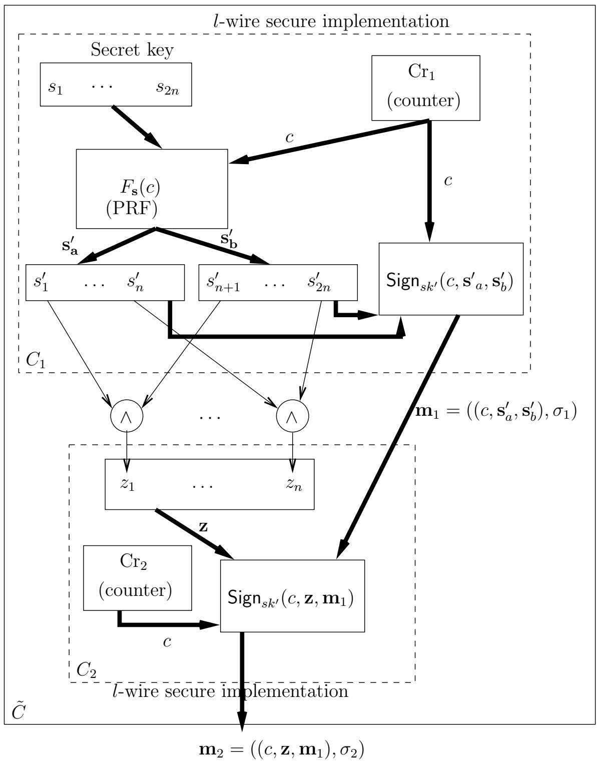
We observe that there is no tampering strategy for $\mathcal{A}_{\mathsf{w}}^t$ consisting of *set, reset* or *toggle* attacks on x, y and w, that simulates the tampering effect of $f_7(x, y) = (x == y)$ (NXOR) and $f_{10}(x, y) = (x \neq y)$ (XOR). We use this observation as a key idea behind theorem 4 which provides a "qualitative" separation between the two classes of adversaries.

In the following we prove that for any n, l, $k \in \mathbb{N}$, there exist a circuit $\tilde{C}$ whose size depends on n, l, k, and a PPT adversary $\mathcal{A}_{\mathsf{g}}^n$, such that for all PPT adversaries $\mathcal{A}_{\mathsf{w}}^t$, where $t \leq l$, $\mathcal{A}_{\mathsf{w}}^t$ fails to simulate the view of $\mathcal{A}_{\mathsf{g}}^n$ while interacting with $\tilde{C}$. Our construction for the counterexample circuit $\tilde{C}$ is presented in Figure 1. It consists of two subcircuits, C_1, C_2, which will be protected against adversaries who tamper with up to l of their wires (l-wire secure). C_1 is the secure transformation of some circuit C_1' which implements a pseudorandom function $F_{\mathbf{s}}(x)$, together with a counter (Cr_a) and a signing algorithm ($\mathsf{Sign}_{sk'}$) of a digital signature scheme $\mathit{\Pi}$ =(Gen,Sign,Vrfy) with secret key sk', $|sk'| = 2n$. C_1 computes $F_{\mathbf{s}}(c)$ and produces two n-bit strings $\mathbf{s}_a'$ and $\mathbf{s}_b'$. Here, c denotes the current counter value and the computation is based on the secret $\mathbf{s}$. Afterwards, the computation σ_1 =$\mathsf{Sign}_{sk'}(c, \mathbf{s}_a', \mathbf{s}_b')$ takes place and $\mathbf{m}_1 = ((c, \mathbf{s}_a', \mathbf{s}_b'), \sigma_1)$ is given as input to C_2, which is the l-wire secure transformation of a circuit C_2'. Furthermore, the two n-bit strings $\mathbf{s}_a'$ and $\mathbf{s}_b'$, are given as input to the AND gates

[4] For clarity, and besides f_7 and f_{10}, we give the functions' representation by logic formulas with respect to both $\wedge$ (Repr. 1) and $\vee$ (Repr. 2) operators.

which compute $\mathbf{s}'_a \wedge \mathbf{s}'_b$. The result $\mathbf{z}$ is given as input to C_2 which implements another instantiation of the signing algorithm on input $\mathbf{z}$ and the counter (Cr_2). Eventually C_2 computes $\sigma_2 = \mathsf{Sign}_{sk'}(c, \mathbf{z}, \mathbf{m}_1)$ and outputs $\mathbf{m}_2 = ((c, \mathbf{z}, \mathbf{m}_1), \sigma_2)$. Notice that Cr_1 and Cr_2 produce the same output in every round and their initial value is zero.

In order to construct the l-wire secure circuits C_1, C_2, we employ the compiler of [18]. This compiler receives the security parameter k, the maximum number of tampering attacks l, C'_1, and outputs C_1. In the same way we transform C'_2 to C_2. Since [18] considers *reversible* NOT gates, i.e., gates on which any tampering action on either side propagates to the other side as well, the NOT gates of C_1, C_2, are also reversible. The final circuit $\tilde{C}$ is the composition of C_1, C_2, as shown in Figure 1. Now, the area between C_1 and C_2 (we call this the *critical*



l-wire secure implementation
Secret key
$s_1 \ \cdots \ s_{2n}$
Cr_1
(counter)
c
$F_\mathbf{s}(c)$
(PRF)
c
$\mathbf{s}'_\mathbf{a}$
$\mathbf{s}'_\mathbf{b}$
$s'_1 \ \ldots \ s'_n$
$s'_{n+1} \ \ldots \ s'_{2n}$
$\mathsf{Sign}_{sk'}(c, \mathrm{s}'_a, \mathrm{s}'_b)$
C_1
$\wedge$
$\cdots$
$\wedge$
$\mathbf{m}_1 = ((c, \mathrm{s}'_a, \mathrm{s}'_b), \sigma_1)$
$z_1 \ \cdots \ z_n$
$\mathbf{z}$
Cr_2
(counter)
$\mathsf{Sign}_{sk'}(c, \mathbf{z}, \mathbf{m}_1)$
c
C_2
l-wire secure implementation
$\tilde{C}$
$\mathbf{m}_2 = ((c, \mathbf{z}, \mathbf{m}_1), \sigma_2)$

Fig. 1. The circuit $\tilde{C}$ for the separation theorem

area) is the part of C that the gate adversary will tamper with by substituting each AND gate with an XOR gate. This will be the main challenge for the wire adversary and its reason to fail the simulation. Specifically, in order to succeed in the simulation the wire adversary should produce two valid signatures σ_1 and

σ_2 on the messages $(c, \mathbf{s}'_a, \mathbf{s}'_b)$ and $(c, \mathbf{z}, \mathbf{m}_1)$ where c is an integer representing the number of rounds the circuit has been executed and $\mathbf{z} = \mathbf{s}'_a \oplus \mathbf{s}'_b$. Now observe that in normal execution the value $\mathbf{z}$ is defined as $\mathbf{s}'_a \wedge \mathbf{s}'_b$ and it is infeasible for the wire adversary to simulate XOR gates using wire tampering directly in the critical area. We emphasize that even by fully controlling the input $\mathbf{z}$ to the second circuit C_2 (and thus entirely circumventing the difficulty of manipulating the $\wedge$ gates) the wire adversary is insufficient since it will have to provide a valid signature in order to execute a proper C_2 evaluation and the only way such a string can come to its possession is via a previous round of circuit execution; this will make the counter found inside each of the two signatures of the final output to carry different values and thus be detectable as a failed simulation attempt.

Using the above logic we now proceed as follows. For the circuit that we have described we consider a simple one-round gate adversary $\mathcal{A}^n_{\mathsf{g}}$ that tampers with the gates in the critical area transforming them into XOR gates and then returns the output of the circuit. Then we show that there exists a polynomial-time distinguisher that given any wire-adversary operating on the same circuit for any polynomial number of rounds is capable of essentially always telling apart the output of the wire adversary from the output of the gate adversary. The impossibility result follows: the knowledge gained by the gate adversary from interacting with the circuit (just once!) is impossible to be derived by any wire adversary (no matter the number of rounds it is allowed to run the circuit).

In the following, the circuit defined above is called $\tilde{C}_{\tilde{\mathbf{s}}}$ with parameters n, k, $l \in \mathbb{N}$, where $\tilde{\mathbf{s}}$ denotes its secret state. Now we define a distinguisher D w.r.t. $\tilde{C}_{\tilde{\mathbf{s}}}$, which receives the public information $\mathbf{v}$ related to $\tilde{\mathbf{s}}$ and $\mathcal{A}^{\tilde{C}^*_{\tilde{\mathbf{s}}}(\cdot)}$, for some tampering adversary $\mathcal{A}$, and distinguishes the output of the gate adversary from the output of the wire adversary.

Distinguisher $D(\mathbf{v}, \mathbf{m}_2)$ w.r.t. $\tilde{C}_{\tilde{\mathbf{s}}}$:
Distinguisher precondition: The environment variable $\mathbf{Env} = (\tilde{\mathbf{s}}, \mathbf{v})$ where $\tilde{\mathbf{s}}$ determines the secret-state of $\tilde{C}$ is such that $\mathbf{v}$ consists of the public key pk of the digital signature Π and $\tilde{\mathbf{s}}$ contains two copies of the secret key of Π, sk', the secret-key of the PRF and the two counters initialized to 0.

Verification: On input $\mathbf{m}_2 = ((c', \mathbf{d}, \mathbf{m}_1), \sigma_2)$, where $\mathbf{m}_1 = ((c, \mathbf{d}_a, \mathbf{d}_b), \sigma_1)$:

$$\begin{array}{rcl}
\text{if } \mathsf{Vrfy}_{\mathsf{pk}}((c', \mathbf{d}, \mathbf{m}_1), \sigma_2) = 0, & & \text{output } 0, \\
\text{else if } \mathsf{Vrfy}_{\mathsf{pk}}((c, \mathbf{d}_a, \mathbf{d}_b), \sigma_1)) = 0, & & \text{output } 0, \\
\text{else if} & \mathbf{d} \neq \mathbf{d}_a \oplus \mathbf{d}_b, & \text{output } 0, \\
\text{else if} & c' \neq c, & \text{output } 0, \\
\text{else} & \text{output } 1. &
\end{array}$$

Fig. 2. The distinguisher D

Theorem 4. *For all l, $k \in \mathbb{N}$, polynomial in n, for the circuit $\tilde{C}_{\tilde{\mathsf{s}}}$ of Figure 1 with parameters n, k, l and* **Env** *as in Figure 2, there exists 1-round gate adversary $\mathcal{A}_{\mathsf{g}}^{n}$ such that for every (multi-round) PPT wire adversary $\mathcal{A}_{\mathsf{w}}^{l}$ it holds for the distinguisher D defined in Figure 2:*

$$|\Pr[D(\mathbf{v}, \mathcal{A}_{\mathsf{w}}^{\tilde{C}^*_{\tilde{\mathsf{s}}}(\cdot)}) = 1] - \Pr[D(\mathbf{v}, \mathcal{A}_{\mathsf{g}}^{\tilde{C}^*_{\tilde{\mathsf{s}}}(\cdot)}) = 1]| = 1 - \mathsf{negl}(\mathsf{n}).$$

In the above theorem, the circuit $\tilde{C}_{\tilde{\mathsf{s}}}$ which distinguishes wire and gate adversaries has a persistent private state which is a random cryptographic key and is operational for an unbounded number of invocations. If one accepts more restricted circuits to be used as counterexamples for separation, specifically circuits that self-destruct after one invocation, we can simplify the separation result via a much simpler circuit. For more details we refer the reader to the paper's full version.

5 Protecting against Gate Adversaries

5.1 Properties That Ensure Security

The following definition generalizes the properties of the compiler presented in [18] and formalizes the functionality for the main parts of the transformed circuit. Definition 10 is a versatile tool for providing tamper-resilient compilers that may be of independent interest. The logic is as follows: we define a (t, k)-secure circuit compiler to be a mapping that produces a circuit accompanied with certain distributions and gate encodings. Specifically the compiler substitutes each wire of the given circuit with a wire-bundle and each gate with a gate that operates over wire-bundles. Within each wire-bundle a specific probability distribution is supposed to exist that encodes probabilistically the 0's and 1's of the original circuit. We note that in the definition below we purposefully leave the exact nature of the class of tampering attacks undetermined.

Definition 10 ((t, k)-secure circuit compiler). *For every t, $k \in \mathbb{N}$, the mapping T over circuits $C \in \mathcal{C}_{\mathcal{G}}$ with n input bits and m output bits where $\mathcal{G} = \{\wedge, \neg\}$ and n, $m \in \mathbb{N}$ and memory strings* $\mathbf{s}$,

$$(C, \mathbf{s}) \to \langle \mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_\perp \rangle, \langle C_\wedge, C_\neg \rangle, \left\langle C_{\mathsf{enc}}, C_{\mathsf{dec}}, \hat{C}, \mathbf{s}', C_{\mathsf{cascade}} \right\rangle$$

is a (t, k)-secure circuit compiler if the circuit $C'_{\mathbf{s}'} = C_{\mathsf{dec}} \circ C_{\mathsf{cascade}} \circ \hat{C}_{\mathbf{s}'} \circ C_{\mathsf{enc}}$ realizes the same functionality with $C_{\mathbf{s}}$ and for any PPT adversary $\mathcal{A}$ with strategy $\mathcal{T}$, where $|\mathcal{T}| \leq t$, the following hold

1. ***(Encoding)*** *$\mathcal{D}_0$, $\mathcal{D}_1$ are distributions of strings in $\{0,1\}^p$, which correspond to valid encodings of the bits 0 and 1, respectively. The length of the encoding, p, depends on the security parameter k and also on t. Moreover, let S_i be the support set of $\mathcal{D}_i$, for $i \in \{0,1\}$. Then, the aforementioned distributions must satisfy the following properties:*

(a) $S_0 \cap S_1 = \emptyset$. *The set of invalid encodings is* $S_\perp = \{0,1\}^p \backslash (S_0 \cup S_1)$.

(b) Each tampering attack against a circuit component that affects a wire-bundle that contains either a sample from $\mathcal{D}_0$ *or* $\mathcal{D}_1$ *may* (i) *leave the value unchanged, or* (ii) *produce an element in* $S_\perp$. *Moreover, there is an efficient way to predict the effect of the tampering (as a distribution over the two events* (i) *and* (ii)*).*

*2. (**Encoder-decoder**) The circuit* C_{enc} *for each input bit* 0 *(resp. input bit* 1*) samples* $\mathcal{D}_0$ *(resp.* $\mathcal{D}_1$*). Moreover, for any* $\mathbf{x} \in \{0,1\}^n$ *the distribution of* $C^{\mathcal{T}}_{\mathsf{enc}}(\mathbf{x})$ *is predictable given the tampering strategy* $\mathcal{T}$ *and* $\mathbf{x}$. C_{dec} *is a deterministic circuit which maps any element of* S_0 *to* 0 *and any element of* S_1 *to* 1.

*3. (**Circuit gates**) The secret state of* C, $\mathbf{s}$, *is substituted by* $\mathbf{s}'$, *where* $\mathbf{s}'$ *is the encoding of* $\mathbf{s}$. *Additionally, every gate in* C *with functionality* $f \in \{\wedge, \neg\}$, n' *input wires and* m' *output wires, is being substituted with the circuit* C_f *with* pn' *input wires and* pm' *output wires. Every wire of* C *is substituted by a bundle of wires* $\mathbf{w}$, *which carries an element in* $S_0 \cup S_1$.
The resuling circuit is $\hat{C}$ *and the following hold:*

*(a) (**Correctness**) For* $i, j \in \{0,1\}$, *if* $\mathbf{x} \sim \mathcal{D}_i$, $\mathbf{y} \sim \mathcal{D}_j$ *then it holds that* $C_\wedge(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}_{\wedge(i,j)}$ *and* $C_\neg(\mathbf{x}) \sim \mathcal{D}_{\neg i}$.

*(b) (**Error propagation**) If* $\mathbf{x} \in S_\perp$ *or* $\mathbf{y} \in S_\perp$, *then* $C_\wedge(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}_\perp$ *and* $C^{\mathcal{T}}_\wedge(\mathbf{x}, \mathbf{y}) \in S_\perp$. *The case for* $C_\neg$ *is similar.*

*(c) (**Simulatability**) For* $i, j \in \{0,1\}$, $\mathbf{x} \sim \mathcal{D}_i$, $\mathbf{y} \sim \mathcal{D}_j$, *one of the following must hold:* (i) $C^{\mathcal{T}}_\wedge(\mathbf{x}, \mathbf{y}) = C_\wedge(\mathbf{x}, \mathbf{y})$ *or* (ii) $C^{\mathcal{T}}_\wedge(\mathbf{x}, \mathbf{y}) \in S_\perp$. *Moreover, there is an efficient way to predict the effect of the tampering as a distribution over the events* (i) *and* (ii), *given* $\mathcal{T}$. *The case for* $C_\neg$ *is similar.*

*4. (**Error propagation & self destruction**)* C_{cascade} *is a circuit which receives* $(\{0,1\}^p)^{m'}$ *wires and returns output in* $(\{0,1\}^p)^{m'}$, *i.e., it receives* m' *wire-bundles and outputs* m' *wire-bundles. It is applied on the output wire-bundles of* $\hat{C}$ *as well as the wire-bundles of* $\hat{C}$ *that update the circuit's secret state (therefore* $m \leq m' \leq m+q$*). Its purpose is to propagate encoding errors and erase the circuit memory (if needed); it works as follows:*

(a) If for all $i \in \{1, \ldots, m'\}$, $\mathbf{y}_i \in S_0 \cup S_1$, *then (1) for all* $i \in \{1, \ldots, m'\}$, *the* i*-th output wire-bundle of* $C_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$ *is equal to* $\mathbf{y}_i$, *and (2) the output distributions of* $C_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$ *and* $C^{\mathcal{T}}_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$ *are simulatable given* $\mathcal{T}$ *and* $C_{\mathbf{s}}(\mathbf{x})$, *where* $\mathbf{x} \in \{0,1\}^n$ *denotes the circuit input.*

(b) If there exists $i \in \{1, \ldots, m'\}$, *s.t.* $\mathbf{y}_i \in S_\perp$, *then, (1) all output wire-bundles of* $C_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'}))$ *will be distributed according to* $\mathcal{D}_\perp$, *(2) all output wire-bundles of* $C^{\mathcal{T}}_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$ *will be in* $S_\perp$, *and (3) the distribution of all output wire-bundles of* $C^{\mathcal{T}}_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$ *will be simulatable given the tampering strategy* $\mathcal{T}$ *and* $C_{\mathbf{s}}(\mathbf{x})$.

5.2 Tamper-Resilient Circuits against Gate Adversaries

Now we give a high level overview of [18] casted as an instance of Definition 10, and we define a gate adversary that compromises its security by attacking randomness gates. Then we prove that by substituting randomness gates with PRNGs, we receive a (t, k)-secure circuit compiler against gate adversaries who tamper with up to t circuit gates. Finally, we prove security for any compiler that satisfies the properties of Definition 10.

A high level description. In [18] the authors consider an encoding in which each input or secret state bit, say x, in the original circuit is encoded into a string of $2k^2t$ bits $(r_1^{2kt}||\ldots||r_k^{2kt})$, where each r_i is a random bit, $i \in [k-1]$, and $r_k = x \oplus r_1 \oplus \ldots \oplus r_{k-1}$. Here, k denotes the security parameter and t is the upper bound on the number of wires that the adversary may tamper with in each computation. The resulting encoding is handled by circuits that implement the functionality of the atomic AND and NOT gates, perform computations over encoded values and satisfy the properties of Definition 10 against wire adversaries. Concretely, let C be a circuit, x an input bit to C and s a secret state bit, and assume that some part of C computes $z = x \wedge s$. According to the aforementioned encoding, the transformed circuit C' encodes x to $(r_1^{2kt}||\ldots||r_k^{2kt})$, where r_i, $i \in [k-1]$, is the output of a randomness gate with fan-out $(2kt)$, [5] and computes $\mathbf{z} = C_\wedge(\mathbf{x}, \mathbf{e})$, using a subcircuit $C_\wedge$ that handles the encoded circuit values and "securely" implements the AND gate. Here, $\mathbf{e}$ and $\mathbf{z}$ denote the encoded version of s and z, respectively, and $\mathbf{z} = (z_1^{2kt}||\ldots||z_k^{2kt})$ is the output of $C_\wedge$ which satisfies $z_i = r_i s_i \oplus \bigoplus_{j \neq i} R_{i,j}$, for $1 \leq i < j \leq k$, $R_{i,j}$ is the output of a randomness gate with fan-out a multiple of $2kt$ and $R_{j,i} = (R_{i,j} \oplus r_i s_j) \oplus r_j b_i$. The number of randomness gates employed by $C_\wedge$ is $\frac{k(k-1)}{2}$. Observe that the value of each wire in the original circuit is shared among k wires and each one of them is replicated $2kt$ times, i.e., each "bundle" consists of k "subbundles" with $2kt$ wires each. The negation of an encoding $\mathbf{e}$ is computed by a circuit $C_\neg$ which consists of $2kt$ NOT gates that simply negate one of the subbundles of $\mathbf{e}$. The whole transformation is the composition of three compilers, and the above description refers to the the second compiler, say T_{rand}. The third compiler replaces randomness gates with circuits that generate pseudo-random bits.

Fact: The compiler of [18] conforms to Definition 10. Let $\mathcal{A}_{\mathsf{w}}^t$ be a wire adversary for C', which is the t-secure transformation of C with respect to T_{rand}, and let s be a secret state bit of C. As we discussed above, s is encoded into $\mathbf{e} = (e_1^{2kt}||\ldots||e_k^{2kt})$, where each e_i, $i \in [k-1]$, is a random bit, and $e_k = s \oplus e_1 \oplus \ldots \oplus e_{k-1}$. Let us consider what happens if $\mathcal{A}_{\mathsf{w}}^t$ tampers with up to t wires of C', where t can be greater than k, and moreover, assume that she tampers with at most $k-1$ different "subbundles" that carry randomized shares of the value s. In such a scenario, the size of each subbundle, which is $2kt$, and the randomization of the carrying values ensure that the adversary may leave the value of each subbundle unchanged or she may alter the value of up to t of its wires, in which case she instantly produces an invalid encoding. Moreover, the effect of the tampering is simulatable in the following way. The simulator simulates the output of the

[5] Besides the $2kt$ wires employed by the encoding, some extra copies of r_i^{2kt} are needed for computing r_k^{2kt}, $i \in [k-1]$.

randomness gates by producing her own randomness, and then she decides the effect of the tampering without touching the distribution of s. On the other hand, if the adversary tampers with all subbundles, and since the randomization on the circuit's signals ensures that each tampering attack produces a fault with constant probability, the simulator knows that the probability that none of the attacks produce an invalid encoding is exponentially small in k. Therefore, with all but negligible (in k) probability an error is induced and propagated by the following circuit components: the cascade phase (Property 4) and the circuits that implement the standard gates of the original circuit (Property 3). Since $C_\wedge$ and $C_\neg$ also produce randomized shares, a similar argument gives us simulatability against adversaries who tamper with such encodings.

Reversible gates. As we have already discussed in section 4, [18] assumes reversible NOT gates. As in [18], in this section we will consider *reversible tampering*, i.e., the adversary who tampers with a reversible NOT gate produces a tampering effect that propagates to the gate's incoming wire (note also that w.r.t. NOT gates the wire and gate adversaries are equivalent).

The compiler T_{rand} is insecure against gate tampering. Let x, s, z and $\mathbf{x}$, $\mathbf{e}$, $\mathbf{z}$ be the values defined above and consider an adversary who (i) sets to zero the $k-1$ randomness gates $R_{i,i+1}$, for $i \in [k-1]$, that lie on $C_\wedge$, (ii) sets to zero the $k-1$ randomness gates that lie on C_{enc} and produce the randomness which is used to encode an input bit x into $\mathbf{x} = (r_1^{2kt}||\dots||r_k^{2kt})$, and (iii) tampers with a gate that outputs z_k. Apparently, the $2(k-1)$ attacks on the randomness gates are fully simulatable. Nevertheless, we have $z_i = 0$, for $i \in [k-1]$ and $z_k = x \cdot s$. Hence, in order to simulate the attack on the gate that outputs z_k, the simulator has to make a "guess" on s and the simulation breaks. Notice, that since we consider persistent tampering, an adversary $\mathcal{A}_{\mathsf{g}}^t$, with $t < k$, can land the aforementioned attack in $2\lceil \frac{k}{t} \rceil$ rounds by tampering with t circuit gates in each round. In general any persistent gate adversary may completely eliminate the circuit's randomness, and the second stage compiler T_{rand} of [18] collapses when subjected to this gate adversary attack. Now, we describe how to circumvent such attacks.

In the full version of the paper we describe how to substitute randomness gates with pseudo-random generators, and then we prove that the resulting compiler, named T_{comp}, satisfies the properties of Definition 10 against gate adversaries. Here we give the intuition on why this construction retains its properties against gate adversaries. The key idea is that eliminating randomness gates effectively removes the advantage of the gate adversary. This is the case because all other gates employed by T_{rand} even those whose fan-out is somehow big (and hence may be thought to be higher value targets for a gate attack), lead to different wire-subbundles. Therefore, a gate adversary that induces a fault will spread the fault to multiple circuit gates. The circuit's defense mechanisms of [18] will then be able to detect the invalid encodings with high probability.

Theorem 5. *For every t, $k \in \mathbb{N}$, the compiler T_{comp} is a (t,k)-secure circuit compiler per definition 10 w.r.t. the class of PPT gate attackers $\mathcal{A}_{\mathsf{g}}^t$.*

The final theorem (which may be of independent interest) states that any compiler which satisfies the properties of definition 10, produces tamper resilient circuits with respect to the standard simulation based security definition.

Theorem 6. *Let $C_{\mathbf{s}}$ any boolean circuit, T_{comp} a (t, k)-secure circuit compiler, t, $k \in \mathbb{N}$, and let $C'_{\mathbf{s}'}$ be the secure transformation of $C_{\mathbf{s}}$ w.r.t. T_{comp}. Then for every tampering adversary $\mathcal{A}$ for which definition 10 applies, there exists a simulator $\mathcal{S}$ such that $\Delta(\mathcal{S}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}), \mathcal{A}^{C'^{*}_{\mathbf{s}'}(\cdot)}(\mathbf{v}))$ is negligible in k.*

The proofs of the above theorems are given in the full version of this paper.

Acknowledgements. This research was supported by ERC project CODAMODA. The second author was also supported by EU FP7 project UaESMC.

References

1. Anderson, R., Kuhn, M.: Tamper resistance-a cautionary note. In: Proceedings of the Second Usenix Workshop on Electronic Commerce, vol. 2, pp. 1–11 (1996)
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (Im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
3. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.: Robust pcps of proximity, shorter pcps and applications to coding. In: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, pp. 1–10. ACM (2004)
4. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
5. Blömer, J., Seifert, J.-P.: Fault based cryptanalysis of the advanced encryption standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003)
6. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
7. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of eliminating errors in cryptographic computations. Journal of cryptology 14(2), 101–119 (2001)
8. Choi, S.G., Kiayias, A., Malkin, T.: BiTR: Built-in tamper resilience. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 740–758. Springer, Heidelberg (2011)
9. Dachman-Soled, D., Kalai, Y.T.: Securing circuits against constant-rate tampering. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 533–551. Springer, Heidelberg (2012)
10. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: ICS, pp. 434–452. Tsinghua University Press (2010)
11. Faust, S., Pietrzak, K., Venturi, D.: Tamper-proof circuits: How to trade leakage for tamper-resilience. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 391–402. Springer, Heidelberg (2011)
12. Gács, P., Gál, A.: Lower bounds for the complexity of reliable boolean circuits with noisy gates. IEEE Transactions on Information Theory 40(2), 579–583 (1994)

13. Gal, A., Szegedy, M.: Fault tolerant circuits and probabilistically checkable proofs. In: Proceedings of Tenth Annual IEEE Structure in Complexity Theory Conference, pp. 65–73. IEEE (1995)
14. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
15. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
16. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, pp. 218–229. ACM (1987)
17. Govindavajhala, S., Appel, A.W.: Using memory errors to attack a virtual machine. In: Proceedings of the 2003 Symposium on Security and Privacy, pp. 154–165. IEEE (2003)
18. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
19. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
20. Katz, J., Lindell, Y.: Introduction to modern cryptography. Chapman & Hall (2008)
21. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Side channel cryptanalysis of product ciphers. In: Quisquater, J.-J., Deswarte, Y., Meadows, C., Gollmann, D. (eds.) ESORICS 1998. LNCS, vol. 1485, pp. 97–110. Springer, Heidelberg (1998)
22. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
23. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
24. Kuhn, M.G., Anderson, R.J.: Soft tempest: Hidden data transmission using electromagnetic emanations. In: Aucsmith, D. (ed.) IH 1998. LNCS, vol. 1525, pp. 124–142. Springer, Heidelberg (1998)
25. Liu, F.-H., Lysyanskaya, A.: Algorithmic tamper-proof security under probing attacks. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 106–120. Springer, Heidelberg (2010)
26. Micali, S., Reyzin, L.: Physically observable cryptography. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
27. Pippenger, N.: On networks of noisy gates. In: 26th Annual Symposium on Foundations of Computer Science, pp. 30–38. IEEE (1985)
28. Quisquater, J.-J., Samyde, D.: Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
29. Rao, J.R., Rohatgi, P.: Empowering side-channel attacks. IACR ePrint, 37 (2001)
30. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Heidelberg (2003)