Tamper Resilient Circuits: The Adversary at the Gates

Aggelos Kiayias[*] Yiannis Tselekounis[*]

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. [IPSW06] 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 [Koc96], or by measuring the time needed for the encryption to complete [KJJ99]. Besides *passive* attacks, the class of *active* attacks includes inducing faults to the computation [BS97, BDL97, KJJ99], exposing the device to electromagnetic radiation [GMO01, QS01, RR01], and several others [GA03, KA98, KSWH98, AK96, BDL01, SA03]. Such attacks have proven to be a significant threat to the real-world security of cryptographic implementations.

1.1 Related work & Motivation

The work of [IPSW06] followed by [FPV11, DSK12] 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.

[*]Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece, {aggelos,tselekounis}@di.uoa.gr. This research was supported by ERC project CODAMODA. The second author was also supported by EU FP7 project UaESMC. The first author was also partly supported by NSF grant 0831306.

In [IPSW06] 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 [FPV11] 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 [IPSW06] 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 [FPV11], this happens with negligible probability in k.

The adversarial model considered in [DSK12] is similar to [IPSW06]. The main difference is that now persistent tampering is not allowed on circuit wires and, similarly, to [FPV11] the simulator is allowed a logarithmic amount of leakage from the computation. Regarding the construction, [DSK12] 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 [GLM+04, LL10, DPW10, CKM11]. In [GLM+04] the authors give an impossibility result on tamper resilience by showing that without using secure hardware an

[1]The construction of [DSK12] considers circuits that output one bit.

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, [IPSW06, FPV11, DSK12] circumvent the impossibility result by erasing the private information in case of error detection. In [LL10] 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. [DPW10] introduces the notion of non-malleable codes. Such codes ensure that any adversary who tampers with a codeword, with respect to some specific 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 [CKM11] 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 [DSK12]). 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 [SA03] 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 bit 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 [GLM$^+$04], 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.

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 [IPSW06] 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 [IPSW06] can be shown to be secure against gate adversaries. Based on [IPSW06], 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 [IPSW06] 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 [IPSW06] 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 $i \in [q]$, 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.*

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 [IPSW06].

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.*

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

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 $\mathcal{A}_{\mathsf{g}}^{C^*(\cdot)}$ (resp. $\mathcal{A}_{\mathsf{w}}^{C^*(\cdot)}$). For $n \in \mathbb{N}$, $[n]$ is the set $\{1, \ldots, n\}$. The statistical distance between two random variables X, Y, with range $\mathcal{D}$, is denoted by $\Delta(X, Y)$, i.e., $\Delta(X, Y) = \frac{1}{2}\sum_{u \in \mathcal{D}} |\Pr[X = u] - \Pr[Y = u]|$. $\Delta(X, Y) \le k$ is also denoted by $X \approx_k Y$, and if D is a distribution over $\mathcal{D}$, $X \sim D$ indicates that variable X follows distribution D. Finally, for any vector $\mathbf{x} = (x_1, \ldots, x_n)$, $[\mathbf{x}]_{\{i \ldots j\}} = (x_i, \ldots, x_j)$, $i, j \in [n]$, $i \le j$.

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 adversary $\mathcal{A}$ with strategy $\mathcal{T}$, such that $\mathcal{A}^{[C,\mathcal{T}]}$ 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.* $\mathbf{Env}$*.*

Proof. The lemma is proved by contradiction: we consider a circuit $C_{\mathbf{s}}$ which satisfies the non-triviality property for some non-negligible function $f(m)$, and none of its bits is a weakly unpredictable bit. Then we construct an algorithm which extracts $\mathbf{s}$ with probability at least $1 - f(m)$. Concretely, let $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ be a pair of random variables and $C_{\mathbf{s}}$ be a circuit with secret state $\mathbf{s}$, where $|\mathbf{s}| = m$, that satisfies the non-triviality property, i.e., for every PPT algorithm $\mathcal{A}$, there exists a non-negligible function $f(m)$ such that $\Pr[\mathcal{A}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}) = \mathbf{s}] < 1 - f(m)$. Here $\mathbf{v}$ contains public information related to $\mathbf{s}$, such as its length. Towards contradiction, suppose that for all $i \in [m]$, there exist PPT algorithms $\mathcal{A}_i$, such that for all non-negligible functions $\delta(m)$, $\Pr[\mathcal{A}_i^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}) = s_i] \geq 1 - \delta(m)$. We define an algorithm $\mathcal{A}^*$ that interacts with $C_{\mathbf{s}}$ in the following way:

i. For $i = 1, \ldots, m$: $\mathcal{A}^*$ executes $\mathcal{A}_i^{C_{\mathbf{s}}(\cdot)}(\mathbf{v})$, and let $s'_i = \mathcal{A}_i^{C_{\mathbf{s}}(\cdot)}(\mathbf{v})$.

ii. $\mathcal{A}^*$ outputs $\mathbf{s}' = (s'_1, \ldots, s'_m)$.

Since $\{\mathcal{A}_i\}_{i=1}^m$ is a family of PPT algorithms, $\mathcal{A}^*$ is also a PPT algorithm and $\Pr[(\mathcal{A}^*(\mathbf{v}))^{C_{\mathbf{s}}(\cdot)} = \mathbf{s}] < 1 - f(m)$ for some non-negligible function $f(m)$ (non-triviality property). Now, let E_i denote the event that $\mathcal{A}_i^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}) \neq s_i$, $1 \leq i \leq m$. Then

$$\begin{aligned}\Pr[(\mathcal{A}^*(\mathbf{v}))^{C_{\mathbf{s}}(\cdot)} \neq \mathbf{s}] &= \Pr[E_1 \cup E_2 \cup \ldots \cup E_m] \\ &\leq \Pr[E_1] + \Pr[E_2] + \ldots + \Pr[E_m] \\ &\leq m\delta(m),\end{aligned}$$

for all non-negligible functions $\delta(m)$. Hence,

$$\begin{aligned}\Pr[(\mathcal{A}^*(\mathbf{v}))^{C_{\mathbf{s}}(\cdot)} = \mathbf{s}] &= 1 - \Pr[(\mathcal{A}^*)^{C_{\mathbf{s}}(\cdot)} \neq \mathbf{s}] \\ &\geq 1 - m\delta(m).\end{aligned}$$

By hypothesis, $f(m)$ is non-negligible. Therefore, $\delta_1(m) = \frac{f(m)}{m}$ is also non-negligible. Since the last inequality holds for all non-negligible functions it clearly holds for $\delta_1(m)$ and we get

$$\begin{aligned}\Pr[(\mathcal{A}^*(\mathbf{v}))^{C_{\mathbf{s}}(\cdot)} = \mathbf{s}] &\geq 1 - m\delta_1(m) \\ &= 1 - f(m),\end{aligned}$$

which breaks the non-triviality property for $C_{\mathbf{s}}$. ■

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* $f(m)$*.*

Proof. Let $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ be a pair of random variables and $C_{\mathbf{s}}$ be a circuit of depth d that satisfies the non-triviality property. Then by Lemma 1, $C_{\mathbf{s}}$ possesses a weakly unpredictable bit, i.e., there exists a secret state bit s_i, $i \in [m]$, such that for every *PPT* algorithm $\mathcal{A}$ there exists a non-negligible function $f(m)$ such that

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

The main idea behind $\mathcal{A}_{\mathsf{w}}$'s strategy is the following: $\mathcal{A}_{\mathsf{w}}$ 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. Afterwards, she executes $C_{\mathbf{s}}^{\mathcal{T}_{\mathsf{w}}}(\mathbf{x})$ for some $\mathbf{x} \in \{0,1\}^n$ of her choice and outputs $y_j = s_i$. The main challenge for $\mathcal{S}$ is to produce the same output bit having black box access to $C_{\mathbf{s}}$.
Concretely, suppose that $\mathcal{P}$ contains the nodes (gates) $\{g_1, \ldots, g_k\}$, where $k \leq d$ and each $g_i \in \{\vee, \wedge, \neg\}$, and let $\{w_1, \ldots, w_k\}$ be the corresponding input wires, i.e., w_i is an input wire to g_i that belongs to $\mathcal{P}$. In case the arity of g_i is 2, the second input wire, i.e., the one that does not belong to $\mathcal{P}$, will be denoted by w_i'. Now its time to define $\mathcal{T}_{\mathsf{w}}$:

Initially $\mathcal{T}_{\mathsf{w}} = \emptyset$. For $i = 1, \ldots, k$:

- if $g_i = \vee$, $\mathcal{T}_{\mathsf{w}} = \mathcal{T}_{\mathsf{w}} \cup (\mathit{Reset}, w_i')$,

- if $g_i = \wedge$, $\mathcal{T}_{\mathsf{w}} = \mathcal{T}_{\mathsf{w}} \cup (\mathit{Set}, w_i')$,

- if $g_i = \neg$, $\mathcal{T}_{\mathsf{w}} = \mathcal{T}_{\mathsf{w}} \cup (\mathit{Toggle}, w_{i+1})$.

Intuitively, if $g_i = \vee$ $(\wedge)$ we cancel its effect on s_i by reseting (setting) the value of w_i' to zero (one). If $g_i = \neg$ we just toggle the value of its outgoing wire. Apparently, $\mathcal{A}_{\mathsf{w}}$ extracts s_i with probability 1. Moreover, since $\mathcal{S}$ is a PPT algorithm and s_i is a weakly unpredictable bit, $\Pr[\mathcal{S}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}) = s_i] < 1 - f(m)$, for some non-negligible function $f(m)$. Hence,

$$\begin{aligned}
\Delta(\mathcal{S}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}), \mathcal{A}_{\mathsf{w}}^{[C_{\mathbf{s}}, \mathcal{T}_{\mathsf{w}}]}(\mathbf{v})) &= \frac{1}{2}\left|\Pr[\mathcal{A}_{\mathsf{w}}^{[C_{\mathbf{s}}, \mathcal{T}_{\mathsf{w}}]}(\mathbf{v}) = s_i] - \Pr[\mathcal{S}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}) = s_i]\right| \\
&+ \frac{1}{2}\left|\Pr[\mathcal{A}_{\mathsf{w}}^{[C_{\mathbf{s}}, \mathcal{T}_{\mathsf{w}}]}(\mathbf{v}) \neq s_i] - \Pr[\mathcal{S}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}) \neq s_i]\right| \\
&> \frac{1}{2}|f(m)| + \frac{1}{2}|f(m)| \\
&= f(m).
\end{aligned}$$

■

The above theorem also holds for circuits that consist of 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: $\mathcal{A}_{\mathsf{w}}$ *sets* 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* $\mathbf{s} \in \{0,1\}^m$ *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* $f(m)$*.*

Proof. The main idea behind the strategy of $\mathcal{A}_{\mathsf{w}}$ is the same as in Theorem 1, with a slight difference: $\mathcal{A}_{\mathsf{w}}$ has to tamper with up to $k-1$ wires for each gate $g_i \in \mathcal{P}$ in order to cancel its effect on s_i. Therefore, $\mathcal{A}_{\mathsf{w}}$ needs to tamper with at most $d(k-1)$ wires in total. ■

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* $f(m)$*.*

Proof. As in Theorem 1, the adversary targets the weakly unpredictable bit s_i and chooses a path $\mathcal{P}$ from the i-th memory gate m_i to an output gate y_j. Suppose that $\mathcal{P}$ contains the gates $\{m_i, g_1, \ldots, g_k, g_{k+1}\}$, where $k \leq d$, $g_{k+1} = y_j$, and let $\{w_1, \ldots, w_{k+1}\}$ be the corresponding input wires, i.e., w_i is an input wire to g_i that belongs to $\mathcal{P}$. Here the adversary tampers with g_i, for $i \in [k]$, in the following way: she substitutes g_i with the function that projects the value of w_i to w_{i+1}. In this way s_i remains unaffected while "traveling" from m_i to y_j while the adversary tampers with up to d circuit gates. ■

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.

Theorem 3. *Let* $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ *be a pair of random variables. For every circuit* C *with gates in* $\mathcal{G} = \{\wedge, \vee, \neg\}$*,* $t \in \mathbb{N}$*, 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* $\mathcal{A}_{\mathsf{g}}^{C_{\mathbf{s}}^*(\cdot)}(\mathbf{v})$*.*

Proof. Let $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ be a pair of random variables, C a circuit in $\mathcal{C}_{\mathcal{G}}$ with private memory V', where $\mathcal{G} = \{\wedge, \vee, \neg\}$, and let $\mathcal{A}_{\mathsf{w}}^t$ be a PPT adversary with strategy $\mathcal{T}_{\mathsf{w}}$, $t \in \mathbb{N}$. We construct an adversary $\mathcal{A}_{\mathsf{g}}^l$, $l \leq t$, with strategy $\mathcal{T}_{\mathsf{g}}$, such that for all $\mathbf{x} \in \{0,1\}^n$, $C_{\mathbf{s}}^{\mathcal{T}_{\mathsf{w}}}(\mathbf{x}) = C_{\mathbf{s}}^{\mathcal{T}_{\mathsf{g}}}(\mathbf{x})$, and if the tampered computation

$C_{\mathbf{s}}^{\mathcal{T}_{\mathsf{w}}}(\mathbf{x})$ updates the private memory from $\mathbf{s}$ to $\mathbf{s}'$, so does $C_{\mathbf{s}}^{\mathcal{T}_{\mathsf{g}}}(\mathbf{x})$, and the result carries from single-round adversaries to v-round ones.

Now assume that $\mathcal{A}_{\mathsf{w}}^t$ executes $C_{\mathbf{s}}^{\mathcal{T}_{\mathsf{w}}}(\mathbf{x})$, for some $\mathbf{x} \in \{0,1\}^n$, and recall that $\mathcal{A}_{\mathsf{w}}^t$ tampers with wires that might be input wires to elements in $V_{\mathcal{G}}$ (circuit gates), V' (memory gates) or V_O (output gates). Let $v \in V_{\mathcal{G}}$, where v represents $g = \wedge$ and let the input wires to g be w_1, $w_2 \in E \cup E'$, i.e., g computes $\wedge(w_1, w_2)$.[3] $\mathcal{A}_{\mathsf{g}}^l$ is aware of $\mathcal{A}_{\mathsf{w}}^t$'s strategy and she is allowed to substitute g with any function in $\{f \mid F : \{0,1\}^2 \to \{0,1\}\}$. We consider the following cases for f w.r.t. $\mathcal{T}_{\mathsf{w}}$:

1. $f(w_1, w_2) = 0$, i.e., f is the zero function if $\mathcal{A}_{\mathsf{w}}^t$ *resets* w_1 or w_2 or both,
2. $f(w_1, w_2) = 1$, i.e., f outputs 1 if $\mathcal{A}_{\mathsf{w}}^t$ *sets* w_1 and w_2,
3. $f(w_1, w_2) = w_2$, if $\mathcal{A}_{\mathsf{w}}^t$ *sets* w_1 (if $\mathcal{A}_{\mathsf{w}}^t$ *sets* w_2, $f(w_1, w_2) = w_1$),
4. $f(w_1, w_2) = (\neg(\vee(w_1, w_2)))$, if $\mathcal{A}_{\mathsf{w}}^t$ *toggles* w_1 and w_2,
5. $f(w_1, w_2) = (\neg(\vee(w_1, \neg w_2)))$, if $\mathcal{A}_{\mathsf{w}}^t$ *toggles* w_1 (if $\mathcal{A}_{\mathsf{w}}^t$ *toggles* w_2, $f(w_1, w_2) = (\neg(\vee(\neg w_1, w_2)))$,
6. $f(w_1, w_2) = (\neg w_2)$, if $\mathcal{A}_{\mathsf{w}}^t$ *sets* w_1 and *toggles* w_2 (if $\mathcal{A}_{\mathsf{w}}^t$ *sets* w_2 and *toggles* w_1, $f(w_1, w_2) = (\neg w_1)$).

The case for $g = \vee$ is quite similar. Now let $g = \neg$ with input wire w. $\mathcal{A}_{\mathsf{g}}^l$ substitutes g with $f \in \{F \mid F : \{0,1\} \to \{0,1\}\}$, where $f(w) = 1$, if $\mathcal{A}_{\mathsf{w}}^t$ *resets* w, $f(w) = 0$, if $\mathcal{A}_{\mathsf{w}}^t$ *sets* w and $f(w) = w$, if $\mathcal{A}_{\mathsf{w}}^t$ *toggles* w.

Let $v \in V'$, i.e., v represents a memory gate g. By definition, g consists of at most one input and one output wire. Clearly, any attack on the input wire of v propagates on its output wire, unless the adversary tampers with the output wire. Now let w be the input wire to g. We consider the following cases:

1. If $\mathcal{A}_{\mathsf{w}}^t$ *sets* the value of w, then $\mathcal{A}_{\mathsf{g}}^l$ substitutes g with $f(w) = 1$.
2. If $\mathcal{A}_{\mathsf{w}}^t$ *resets* the value of w, then $\mathcal{A}_{\mathsf{g}}^l$ applies the function $f(w) = 0$.
3. If $\mathcal{A}_{\mathsf{w}}^t$ *toggles* the value of w, then $\mathcal{A}_{\mathsf{g}}^l$ applies the function $f(w) = \neg w$.

The output wire of v can be considered an input wire to some gate in $V_{\mathcal{G}}$. Therefore, this case is already covered.

Finally, let $v \in V_O$ be an output node and y its value with respect to the tampering made so far. Moreover, let $w \in E$ be its incoming wire. Since $\mathcal{A}_{\mathsf{g}}^l$ knows the strategy of $\mathcal{A}_{\mathsf{w}}^t$, she knows if $\mathcal{A}_{\mathsf{w}}^t$ lands a *Set*, *Reset* or *Toggle* attack and applies $f(w) = 1$, $f(w) = 0$ or $f(w) = \neg(w)$, respectively. This holds for every output bit y_i, $i \in [q]$. Hence, $C_{\mathbf{s}}^{\mathcal{T}_{\mathsf{w}}}(\mathbf{x}) = C_{\mathbf{s}}^{\mathcal{T}_{\mathsf{g}}}(\mathbf{x})$, and since $\mathcal{A}_{\mathsf{w}}^t$ tampers with at most t wires she indirectly affects the functionality of up to t circuit gates, which is the maximum number of gates that $\mathcal{A}_{\mathsf{g}}^l$ needs to tamper with. ■

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.

[3]In the following w will denote both a circuit wire $w \in E \cup E'$, as well as its bit-value.

[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.

	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$	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)	$((\mathsf{R},x),(\mathsf{R},y))$

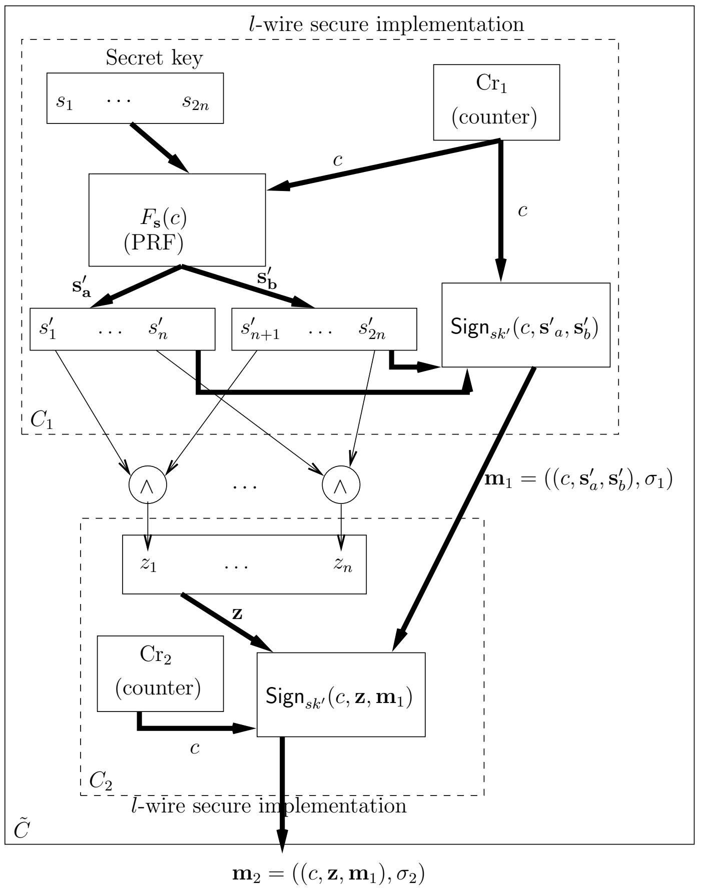
Table 1: All boolean functions from $\{0,1\}^2$ to {0,1} and $\mathcal{A}_{\mathsf{w}}^t$'s tampering strategy.

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_1) and a signing algorithm ($\mathsf{Sign}_{sk'}$) of a digital signature scheme $\Pi =$(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 $\sigma_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 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 σ_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 [IPSW06]. 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 [IPSW06] 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 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



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

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

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 $\mathcal{A}^n_{\mathbf{g}}$ 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 $\mathbf{m}_1 = ((c, \mathbf{d}_a, \mathbf{d}_b), \sigma_1)$:

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

Figure 2: The distinguisher D

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

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

Before proving Theorem 4 we prove the following lemma.

Lemma 2. *Let $F_{\mathbf{s}} : \{0,1\}^{2n} \to \{0,1\}^{2n}$ be any pseudorandom function (PRF), $\mathcal{S} \subseteq \{0,1\}^{2n}$ and $|\mathcal{S}| = q(n)$, where $q(n)$ is polynomial in n. Then for all $\mathbf{x} \in \mathcal{S}$, if $F_{\mathbf{s}}(\mathbf{x}) = \mathbf{r}$, where $\mathbf{r} = \mathbf{r}_a || \mathbf{r}_b$ and $|\mathbf{r}_a| = |\mathbf{r}_b| = n$,*

$$\Pr[\mathbf{r}_a \wedge \mathbf{r}_b = \mathbf{r}_a \oplus \mathbf{r}_b] = \mathsf{negl}(n).$$

Proof. Since $F_{\mathbf{s}}$ is a *PRF* we have that for every PPT algorithm $\mathcal{A}$, there exists a negligible function $\epsilon(n)$

$$\left|\Pr[\mathcal{A}^{F_{\mathbf{s}}(\cdot)}(1^{2n}) = 1] - \Pr[\mathcal{A}^{f(\cdot)}(1^{2n}) = 1]\right| \leq \epsilon(n),$$

where f is a random function from $\{0,1\}^{2n}$ to $\{0,1\}^{2n}$. Now, for any $\mathbf{x}$ in $\{0,1\}^{2n}$, $f(\mathbf{x})$ is uniformly distributed over $\{0,1\}^{2n}$. Let $f(\mathbf{x}) = \mathbf{r}_a||\mathbf{r}_b$. Then, $\mathbf{r}_a$ and $\mathbf{r}_a$ are uniformly distributed over $\{0,1\}^n$, and therefore,

$$\Pr[\mathbf{r}_a \wedge \mathbf{r}_b = \mathbf{r}_a \oplus \mathbf{r}_b] = \frac{1}{4^n}.$$

Towards contradiction, suppose that for some $\mathbf{x}$ in $\mathcal{S}$, $F_{\mathbf{s}}(\mathbf{x}) = \mathbf{r}_a||\mathbf{r}_b$ and

$$\Pr[\mathbf{r}_a \wedge \mathbf{r}_b = \mathbf{r}_a \oplus \mathbf{r}_b] = \frac{q(n)}{4^n} + g(n),$$

where $g(n)$ is a non-negligible function. Now we define a PPT algorithm $\mathcal{B}$ which distinguishes f from $F_{\mathbf{s}}$

Algorithm $\mathcal{B}$:

- Choose $\mathbf{x}$ uniformly at random from $\mathcal{S}$,
- query the oracle using $\mathbf{x}$ and receive $\mathbf{r} = \mathbf{r}_a||\mathbf{r}_b$,
- if $\mathbf{r}_a \wedge \mathbf{r}_b = \mathbf{r}_a \oplus \mathbf{r}_b$ return 1, otherwise return 0.

So,

$$\left|\Pr[\mathcal{B}^{F_{\mathbf{s}}(\cdot)}(1^{2n}) = 1] - \Pr[\mathcal{B}^{f(\cdot)}(1^{2n}) = 1]\right| \geq \left|\frac{1}{q(n)} \cdot \left(\frac{q(n)}{4^n} + g(n)\right) - \frac{1}{4^n}\right| = \frac{g(n)}{q(n)},$$

and therefore, $\mathcal{B}$ distinguishes f from $F_{\mathbf{s}}$ with non-nengligible probability, which contradicts the hypothesis for $F_{\mathbf{s}}$. ■

Now we proceed to the proof of Theorem 4.

Proof. As we have already discussed, $\mathcal{A}^n_{\mathsf{g}}$ acts as follows: she chooses strategy $\mathcal{T}_{\mathsf{g}}$ according to which each AND gate of the critical area is substituted by an XOR gate, and then executes the circuit. Since the computation that takes place on C_1, C_2 is untampered, it is not hard to see that the circuit output, $\mathbf{m}_2 = ((c', \mathbf{d}, \mathbf{m}_1), \sigma_2)$, where $\mathbf{m}_1 = ((c, \mathbf{d}_a, \mathbf{d}_b), \sigma_1)$, satisfies the following properties: (i) σ_2 (resp. σ_1) is a valid signature for $(c', \mathbf{d}, \mathbf{m}_1)$ (resp. $(c, \mathbf{d}_a, \mathbf{d}_b)$) w.r.t. Π, (ii) $c' = c$, and (iii) $\mathbf{d} = \mathbf{d}_a \oplus \mathbf{d}_b$. Hence, by the definition of the distinguisher D, we have that $\Pr[D(\mathbf{v}, \mathcal{A}_{\mathsf{g}}^{\tilde{C}^*_{\tilde{\mathbf{s}}}(\cdot)}) = 1] = 1$.

Before discussing the attack vectors of $\mathcal{A}^l_{\mathsf{w}}$ against $\tilde{C}_{\tilde{\mathbf{s}}}$, we introduce some notational conventions: let $\mathcal{G} = \{g_1, \ldots, g_n\}$ be the set of AND gates that lie on the critical area of the circuit, let $\mathcal{W}_{\mathbf{m}_1}$ be the set of circuit wires that carry $\mathbf{m}_1$, and $\mathcal{W}_{\mathbf{m}_2}$ the corresponding set for $\mathbf{m}_2$. We partition the probability space as follows:

- S: D on input $\mathbf{v}$, $\mathcal{A}_{\mathsf{w}}^{\tilde{C}^*_{\tilde{\mathbf{s}}}(\cdot)}$, outputs 1.
- N: $\mathcal{A}_{\mathsf{w}}^{l}$ outputs $\mathbf{m}_2$ after interacting with $\tilde{C}_{\tilde{\mathbf{s}}}$ for at least one round.
- O: $\mathbf{m}_2$ is the output of a circuit computation, tampered or untampered.[5]
- T: $\mathcal{A}_{\mathsf{w}}^{l}$ tampers with $\tilde{C}_{\tilde{\mathbf{s}}}$.
- T': $\mathcal{A}_{\mathsf{w}}^{l}$ tampers with $\tilde{C}_{\tilde{\mathbf{s}}}$ and induces a fault.
- T_1: $\mathcal{A}_{\mathsf{w}}^{l}$ tampers with the wires in $\mathcal{W}_{\mathbf{m}_1}$ and produces $\mathbf{m}_1' \neq \mathbf{m}_1$, where $\mathbf{m}_1$ is the partial output of the execution of C_1, tampered or not.
- T_1': $\mathcal{A}_{\mathsf{w}}^{l}$ substitutes the current value of the elements in $\mathcal{W}_{\mathbf{m}_1}$ with some $\mathbf{m}_1'$, which is a valid output of C_1 from a preceding round.
- T_2: $\mathcal{A}_{\mathsf{w}}^{l}$ alters the output of the critical area by tampering with the input or output wires of the elements in $\mathcal{G}$.
- T_3: $\mathcal{A}_{\mathsf{w}}^{l}$ tampers with C_j, for some $j \in \{1,2\}$, and induces a fault.
- F: $\mathcal{A}_{\mathsf{w}}^{l}$ forges a valid signature with respect to Π.

The probability of S equals to

$$\Pr[S] = \Pr[S|N] \cdot \Pr[N] + \Pr[S|\neg N] \cdot \Pr[\neg N], \tag{1}$$

and we need to show that $\Pr[S]$ is negligible in n. Clearly,

$$\Pr[S|\neg N] \leq \Pr[F] = \mathsf{negl}(n), \tag{2}$$

since $\mathcal{A}_{\mathsf{w}}^{l}$ does not interact with $\tilde{C}_{\tilde{\mathbf{s}}}$, and therefore, she has to come up with a valid signature forgery. Now we split $\Pr[S|N]$ with respect to O and its complement[6]

$$\Pr[S|N] = \Pr[S|N,O] \cdot \Pr[O] + \Pr[S|N,\neg O] \cdot \Pr[\neg O], \tag{3}$$

and let $X' = [N, \neg O]$ and $X = [N, O]$. Regarding $\Pr[S|X']$, since $\mathbf{m}_2$ is not the output of a circuit computation, tampered or untampered, $\mathcal{A}_{\mathsf{w}}^{l}$ produces a valid output if (i) succeeds in forging a valid signature for each $\mathbf{m}_i$, $i \in \{1,2\}$, with respect to Π, which happens with negligible probability in n, or (ii) she tampers with C_1 and/or C_2, extracts sk' and computes the signatures on her own. Since each C_i is l-wire secure, this happens with negligible probability in k. Hence,

$$\Pr[S|X'] \leq \mathsf{negl}(n). \tag{4}$$

Regarding S given X we have

$$\Pr[S|X] = \Pr[S|X,T] \cdot \Pr[T] + \Pr[S|X,\neg T] \cdot \Pr[\neg T], \tag{5}$$

and furthermore, let $X_1 = [X, T]$, $X_1' = [X, \neg T]$. Then

$$\Pr[S|X_1] = \Pr[S|X_1,T'] \cdot \Pr[T'] + \Pr[S|X_1,\neg T'] \cdot \Pr[\neg T'], \tag{6}$$

and

$$\Pr[S|X_1'] = \Pr[\mathbf{s}_a' \wedge \mathbf{s}_b' = \mathbf{s}_a' \oplus \mathbf{s}_b'] = \mathsf{negl}(n). \tag{7}$$

[5] Tampering with the output wires of $\tilde{C}_{\tilde{\mathbf{s}}}$ reduces to substitution of the circuit output $\mathbf{m}_2$ by some $\mathbf{m}_2'$, i.e., $\mathbf{m}_2'$ is not considered as output of any circuit computation, tampered or not.

[6] $(A \cap B)$ is also denoted by (A, B).

The above follows from Lemma 2, since the untampered PRF output is $\mathbf{s}'_a||\mathbf{s}'_b$. Now let $X_2 = [X_1, T']$. Then

$$\begin{aligned}\Pr[S|X_2] &= \sum_{i=1}^{3}\Pr[S|X_2,T_i]\cdot\Pr[T_i] + \sum_{i=1}^{2}\sum_{j=i+1}^{3}\Pr[S|X_2,T_i,T_j]\cdot\Pr[T_i,T_j] \\ &+ Pr[S|X_2,T_1,T_2,T_3]\cdot\Pr[T_1,T_2,T_3]. \qquad (8)\end{aligned}$$

We expand the above probability summations,

$$\Pr[S|X_2,T_1] = \Pr[S|X_2,T_1,T'_1]\cdot\Pr[T'_1] + \Pr[S|X_2,T_1,\neg T'_1]\cdot\Pr[\neg T'_1] \le \mathsf{negl}(n). \qquad (9)$$

The above inequality holds due to the following: if the adversary substitutes $\mathbf{m}_1$ with $\mathbf{m}'_1$, where $\mathbf{m}'_1$ is taken from a previous round with counter value c', then the circuit output, say $\mathbf{m}'_2$, contains different counter values. Hence, $\Pr[S|X_2,T_1,T'_1] = 0$. On the other hand, if the attacker substitutes $\mathbf{m}_1$ with $\mathbf{m}'_1$, where $\mathbf{m}'_1$ is not the output of a previous computation, then $\Pr[S|X_2,T_1,\neg T'_1] \le \Pr[F] = \mathsf{negl}(n)$.

Regarding T_2, (i) $\mathcal{A}^l_{\mathsf{w}}$ cannot produce the XOR tampering effect by tampering with the input or output wires of the critical area, and (ii) $\mathcal{A}^l_{\mathsf{w}}$ guesses the output of the PRF with negligible probability in n. Hence,

$$\Pr[S|X_2,T_2] = \mathsf{negl}(n). \qquad (10)$$

If $\mathcal{A}^l_{\mathsf{w}}$ induces a fault to C_1 and/or C_2, she learns sk' with negligible probability in k, and triggers the circuit's self destruction mechanism with probability very close to 1. Hence,

$$\Pr[S|X_2,T_3] \le \mathsf{negl}(k). \qquad (11)$$

Now,

$$\begin{aligned}\Pr[S|X_2,T_1,T_2] &= \Pr[S|X_2,T_1,T_2,T'_1]\cdot\Pr[T'_1] + \Pr[S|X_2,T_1,T_2,\neg T'_1]\cdot\Pr[\neg T'_1] \\ &\le 0\cdot\Pr[T'_1] + \Pr[F]\cdot\Pr[\neg T'_1] \le \mathsf{negl}(n). \qquad (12)\end{aligned}$$

In the above inequality, $\Pr[S|X_2,T_1,T_2,T'_1] = 0$ due to the mismatch between the two counter values, and regardless of the adversarial strategy against the critical area. Moreover, $\Pr[S|X_2,T_1,T_2,\neg T'_1] \le \Pr[F]$ since the attacker needs to forge a valid message $\mathbf{m}_1$ w.r.t. to Π, whose counter value would be equal to the counter value of C_2. Using similar arguments we receive

$$\begin{aligned}\Pr[S|X_2,T_1,T_3] &= \Pr[S|X_2,T_1,T_3,T'_1]\cdot\Pr[T'_1] + \Pr[S|X_2,T_1,T_3,\neg T'_1]\cdot\Pr[\neg T'_1] \\ &\le 0\cdot\Pr[T'_1] + \Pr[F]\cdot\Pr[\neg T'_1] \le \mathsf{negl}(n). \qquad (13) \\ \Pr[S|X_2,T_2,T_3] &\le \mathsf{negl}(k), \qquad (14)\end{aligned}$$

and

$$\Pr[S|X_2,T_1,T_2,T_3] \le \mathsf{negl}(k). \qquad (15)$$

Now, by setting the unconditional probabilities in (8) equal to 1, and by (9)-(15) we receive

$$\Pr[S|X_2] \le \mathsf{negl}(n). \qquad (16)$$

Now let $X'_2 = [X_1, \neg T']$. Then,

$$\Pr[S|X'_2] = \Pr[\mathbf{s}'_a \wedge \mathbf{s}'_b = \mathbf{s}'_a \oplus \mathbf{s}'_b] = \mathsf{negl}(n) \text{ (Lemma 2)}. \qquad (17)$$

We set the unconditional probabilities of (6) equal to 1, we apply (16), (17) on it, and we receive

$$\Pr[S|X_1] \le \mathsf{negl}(n). \qquad (18)$$

Hence, by (18), (7) and (5) we get that

$$\Pr[S|X] \le \mathsf{negl}(n), \qquad (19)$$

and by applying (19), (4) on (3) gives

$$\Pr[S|N] \leq \mathsf{negl}(n). \tag{20}$$

Finally, by (20), (2) and (1) we get

$$\Pr[S] \leq \mathsf{negl}(n). \tag{21}$$

■

In the above theorem, the circuit $\tilde{C}_{\bar{\mathbf{s}}}$ which distinguishes wire and gate adversaries has persistent private state and is operational for unbounded number of invocations. If one accepts more restricted circuits to be used as counterexamples, specifically circuits that self-destruct after one invocation, we can simplify the separation result via a much simpler circuit shown in Figure 3. That circuit, say $\bar{C}_{\bar{\mathbf{s}}}$, outputs $2n$ bits, and each output bit is the AND of two private state bits as shown below. Moreover, it employs a self-destruction mechanism, consisting of standard AND gates, that ensures memory erasure in one invocation against adversaries that tamper with t circuit wires. The environment variable $\mathbf{Env} = (\bar{\mathbf{s}}, \mathbf{v})$ w.r.t. $\bar{C}_{\bar{\mathbf{s}}}$ is defined as follows:

1. Sample message $\mathbf{m} \overset{r}{\leftarrow} U_l$, for some $l \in \mathbb{N}$, compute $\sigma = \mathsf{Sign}_{\mathsf{sk}}(\mathbf{m})$ and let $\mathbf{p} = \mathbf{m}||\sigma$, $\mathbf{p} \in \{0,1\}^n$, where $\mathsf{Sign}_{\mathsf{sk}}$ is the signing algorithm of a digital signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}_{\mathsf{sk}}, \mathsf{Vrfy}_{\mathsf{pk}})$.
2. Sample $\mathbf{s}$, $\mathbf{s}' \overset{r}{\leftarrow} U_n$, compute $\mathbf{k} = \mathbf{s} \oplus \mathbf{s}'$ and set $\mathbf{c}' = \mathbf{k} \oplus \mathbf{p}$.
3. Sample $\mathbf{c} \overset{r}{\leftarrow} U_n$ and set $\mathbf{d} = \mathbf{c} \oplus \mathbf{c}'$.
4. Initialize the circuits secret state to $\mathbf{s}||\mathbf{s}'||\mathbf{c}||\mathbf{d}$.

The strategy of the gate adversary is the same as in Theorem 1: $\mathcal{A}^n_{\mathsf{g}}$ tampers with the rightmost n AND gates and substitutes each one of them with an XOR gate. In this way, she receives $\mathbf{k} = \mathbf{s} \oplus \mathbf{s}'$, $\mathbf{c}' = \mathbf{c} \oplus \mathbf{d}$, and a valid signature with respect to Π by computing $\mathbf{p} = \mathbf{c}' \oplus \mathbf{k}$. Hence, the wire adversary has to produce a valid signature in one circuit invocation, by tampering with up to $t = \mathsf{poly}(n)$ circuit wires.

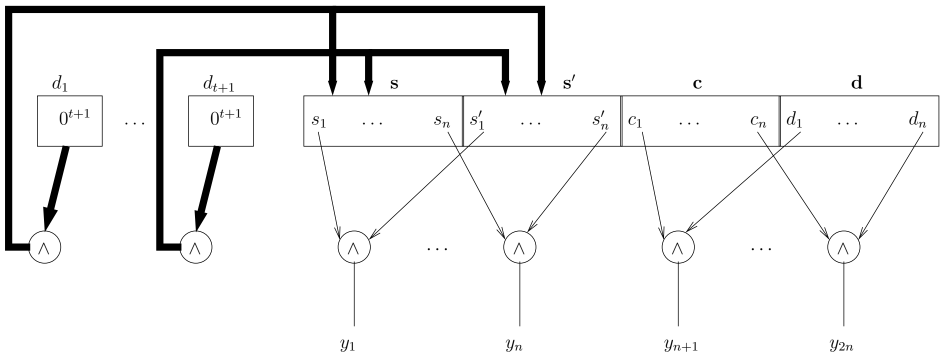


Figure 3: A circuit that self-destructs after one invocation.

5 Defending against Gate Adversaries

5.1 Properties that ensure security

The following definition generalizes the properties of the compiler presented in [IPSW06] 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 *mega-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. More specifically, we consider a tampering attack for a single circuit execution to be a sequence of instructions $\mathcal{T}$ that modify a number of circuit components equal to the cardinality of $\mathcal{T}$ (these may be wires or gates).

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 q output bits where $\mathcal{G} = \{\wedge, \neg\}$ and $n,\ q \in \mathbb{N}$, and memory strings $\mathbf{s}$, $|\mathbf{s}| = m$,*

$$(C, \mathbf{s}) \rightarrow \langle \mathcal{D}_0, \mathcal{D}_1 \rangle, \langle C_{\mathsf{AND}}, C_{\mathsf{NOT}} \rangle, \left\langle C_{\mathsf{enc}}, C_{\mathsf{dec}}, \hat{C}, \mathbf{s}', C_{\mathsf{cascade}} \right\rangle$$

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 the following hold (for any arbitrary tampering strategy $\mathcal{T}$, where $|\mathcal{T}| \leq t$):

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 $\{0,1\}^p \backslash (S_0 \cup S_1)$, is denoted by $S_\perp$.

   (b) Each tampering attack $\mathcal{T}$ against a circuit component that affects a wire-bundle[7] 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, given $\mathcal{T}$ there is an efficient way to predict the effect of the tampering (as a distribution over the two events (i) and (ii)), with all but negligible probability in k, i.e., the stastistical distance between the two distributions is negligible in k.

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 simulatable with all but negligible probability in k, 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 $\mathbf{s}'$, $|\mathbf{s}'| = pm$, where $\mathbf{s}'$ is the encoding of $\mathbf{s}$. Additionally, every gate in C with functionality $f \in \{\wedge, \neg\}$, $n' \in \{1,2\}$ input wires and $q' \geq 1$ output wires, is being substituted with the circuit C_f with pn' input wires and pq' output wires. Every wire of C is substituted by a bundle of wires of size p which during circuit computation will carry an element in $S_0 \cup S_1$.*

   The resulting 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 $C_{\mathsf{AND}}(\mathbf{x},\mathbf{y}) \sim \mathcal{D}_{(i \wedge j)}$ and $C_{\mathsf{NOT}}(\mathbf{x}) \sim \mathcal{D}_{\neg i}$.*

   *(b) (**Error propagation**) If $\mathbf{x} \in S_\perp$ or $\mathbf{y} \in S_\perp$, then $C_{\mathsf{AND}}(\mathbf{x},\mathbf{y}) \in S_\perp$ and $C^{\mathcal{T}}_{\mathsf{AND}}(\mathbf{x},\mathbf{y}) \in S_\perp$. The case for C_{NOT} 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) $C^{\mathcal{T}}_{\mathsf{AND}}(\mathbf{x},\mathbf{y}) \approx C_{\mathsf{AND}}(\mathbf{x},\mathbf{y})$ or (ii) $C^{\mathcal{T}}_{\mathsf{AND}}(\mathbf{x},\mathbf{y}) \in S_\perp$. Moreover, there is an efficient way to predict the effect of the tampering as a distribution over the events (i) and (ii), given $\mathcal{T}$, with all but negligible probability in k. The case for C_{NOT} 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, where $m' = m + q$. The first q input wire-bundles of C_{cascade} are the output wire-bundles of $\hat{C}$, while the bundles $q+1, \ldots, m+q$, feed the private memory of $\hat{C}$. Its purpose is to propagate encoding errors and erase the circuit memory (if needed); it works as follows:*

[7]For example an attack against a single wire or a memory gate.

(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 $\mathbf{y}_i$, (2) the output distributions of $[C_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})]_{\{1\ldots q\}}$ and $[C^{\mathcal{T}}_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})]_{\{1\ldots q\}}$ are simulatable with all but negligible probability in k, given $\mathcal{T}$ and $C_{\mathbf{s}}(\mathbf{x})$, where $\mathbf{x} \in \{0,1\}^n$ denotes the circuit input, and (3) there is an efficient way to predict if there is any wire-bundle in $[C^{\mathcal{T}}_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})]_{\{q+1\ldots m'\}}$ that carries an element in $S_\perp$, with all but negligible probability in k.*

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

Remark. Note that the above compiler will be suitable for multi-round adversaries. Intuitively, any single-round adversary attacking with a strategy $\mathcal{T}$ for which Definition 10 applies, will either produce an invalid encoding, which implies memory erasure and error propagation throughout the entire circuit, or it leaves the computation intact. This is sufficient to enable simulatability against attackers that follow a multi-round strategy as we demonstrate on Theorem 6.

5.2 Tamper-resilient circuits against gate adversaries

Now we give a high level overview of [IPSW06] 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 [IPSW06] 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 $(2kt)$, [8] and computes $\mathbf{z} = C_{\mathsf{AND}}(\mathbf{x}, \mathbf{e})$, using a subcircuit C_{AND} that handles the encoded circuit values and "securely" implements the AND gate. Here, $\mathbf{e} = (e_1^{2kt}|| \ldots ||e_k^{2kt})$ and $\mathbf{z} = (z_1^{2kt}|| \ldots ||z_k^{2kt})$ denote the encoded version of s and z, respectively, and $\mathbf{z}$ is the output of C_{AND} which satisfies $z_i = r_i e_i \oplus \bigoplus_{j \neq i} R_{i,j}$, for $1 \le i < j \le k$, $R_{i,j}$ is the output of a randomness gate with fan-out a multiple of $2kt$ and $R_{j,i} = (R_{i,j} \oplus r_i e_j) \oplus r_j e_i$. The number of randomness gates employed by C_{AND} 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_{NOT} which consists of $2kt$ NOT gates that simply negate one of the subbundles of $\mathbf{e}$. The whole circuit transformation is the composition of three compilers, and the above description refers to the the second compiler, which we will call T_{rand}. The third compiler replaces randomness gates with circuits that generate pseudo-random bits.

Fact: The compiler of [IPSW06] conforms to Definition 10. Let $\mathcal{A}^t_{\mathsf{w}}$ 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 $\mathcal{A}^t_{\mathsf{w}}$ 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

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

$2kt$, 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 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_{AND} and C_{NOT} 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, [IPSW06] assumes reversible NOT gates. As in [IPSW06], 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. Suppose first that $t \geq 2k-1$ and let x, s, z and $\mathbf{x}$, $\mathbf{e}$, $\mathbf{z}$ be the values defined in the high level description above. 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_{AND}, (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 (e.g., sets it to 0). 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. For the case when $t < 2k-1$ notice that since we consider persistent tampering, an adversary $\mathcal{A}_{\mathsf{g}}^t$ can still 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 [IPSW06] collapses when subjected to this gate adversary attack (contrary to this, a wire adversary cannot achieve the same effect since the randomness gates attacked above have a large fan-out and thus cannot be effectively zeroed). Next we describe how to circumvent such attacks via the substitution of randomness gates with logic circuitry that implements a pseudo-random generator (as in the third compiler of [IPSW06]).

Eliminating randomness gates. Consider a circuit C_{prg} which implements a pseudo-random generator initialized with a random seed $\mathbf{m} \in \{0,1\}^n$, that consists of l AND gates and any, polynomial in n, number of NOT gates, and outputs one pseudo-random bit. We modify C_{prg} with respect to t and k as follows:

1. **Secret state:** Each secret state bit, m_i, of $\mathbf{m}$ is replaced by $(r_1^{2kt}|| \dots ||r_k^{2kt})$, where $r_i \xleftarrow{r} \{0,1\}$ for $i \in [k-1]$ and $r_k = m_i \oplus r_1 \oplus \dots \oplus r_{k-1}$. Furthermore, we generate $l \cdot \frac{k(k-1)}{2}$ random bits, and we replicate each bit $2kt$ times. The resulting secret state is of length $O(k^3 tl)$.

2. **Circuit gates:** Each AND and NOT gate is replaced by C_{AND} and C_{NOT}, respectively (as defined above), which perform computations over encoded values. Moreover, the output gate is replaced by $2k^2t$ output gates.

3. **Circuit wires:** Each wire of C is replaced by $2k^2t$ wires that carry encoded bit-values. Finally, we introduce $O(k^3tl)$ wires that feed the C_{AND} subcircuits with the random bits produced in step 1 and stored in the secret state. Recall that in T_{rand}, the randomness needed by C_{AND} is produced by randomness gates. Instead here we feed C_{AND} with the randomness built into the secret state that follows the encoding described above.

4. **Circuit output:** With the modifications made so far, the circuit outputs a pseudo-random bit r which is encoded into $(r_1^{2kt}||\ldots||r_k^{2kt})$. In order to produce the same output with a randomness gate having fan-out equal to $2kt$ we need to compute r^{2kt}, where $r = r_1 \oplus \ldots \oplus r_k$. So, we expand C_{prg} with $k-1$ $\bar{C}_{\mathsf{XOR}}$ gadgets that were introduced in [IPSW06]. Each $\bar{C}_{\mathsf{XOR}}$ is a tamperable circuit which receives $4kt$ input bits, outputs $2kt$ bits and implements the following functionality:
   - $\bar{C}_{\mathsf{XOR}}(0^{2kt}, 0^{2kt}) = 0^{2kt}$,
   - $\bar{C}_{\mathsf{XOR}}(0^{2kt}, 1^{2kt}) = 1^{2kt}$,
   - $\bar{C}_{\mathsf{XOR}}(1^{2kt}, 0^{2kt}) = 1^{2kt}$,
   - $\bar{C}_{\mathsf{XOR}}(1^{2kt}, 1^{2kt}) = 0^{2kt}$,
   - $\bar{C}_{\mathsf{XOR}}(*^{2kt}, *^{2kt}) = 0^{kt}1^{kt}$. Here "$*^{2kt}$" denotes any string of length $2kt$ that is not covered by the above cases.

   The implementation of each output bit of $\bar{C}_{\mathsf{XOR}}$ is realized by a circuit of the form "OR of ANDs" of the input bits or their NOTs, or the "NOT" of such a circuit. Notice how the $\bar{C}_{\mathsf{XOR}}$ gadget propagates invalid encodings, i.e., bit-strings in $\{0,1\}^{2kt}$ that consist of both zero and one bits.

Now, let $\bar{C}_{\mathsf{prg}}$ be the resulting circuit, which apparently outputs r^{2kt}. We replace each randomness gate with the XORed output of k such circuits $\bar{\mathcal{C}}_{\mathsf{prg}} = \{\bar{C}^1_{\mathsf{prg}}, \ldots, \bar{C}^k_{\mathsf{prg}}\}$, which consist of k uniform and independent seeds. So, if an adversary tampers with fewer that k of these circuits, then due to the randomization of their signals there exists a simulation that reproduces the tampering effect. On the other hand, if the adversary attacks all k circuits, she induces a fault with all but negligible probability in k.

Now, let T_{comp} be T_{rand} in which each randomness gate is replaced by the XORed output of the elements in $\bar{\mathcal{C}}_{\mathsf{prg}}$. We now prove that T_{comp} satisfies the properties of Definition 10 against gate adversaries, but before doing so we give the intuition on why this construction somehow 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 [IPSW06] will then be able to detect the invalid encodings with high probability. Now we prove that T_{comp} is a (t,k)-secure compiler against gate adversaries.

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 $\mathcal{A}^t_{\mathsf{g}}$.*

Proof. We prove the theorem's statement by verifying each property of Definition 10 separately. So, let $C_{\mathsf{s}} \in \mathcal{C}_{\mathcal{G}}$ be a circuit, where $\mathcal{G} = \{\wedge, \neg\}$ and let $C'_{\mathsf{s}'}$ be its secure transformation w.r.t. T_{comp}, t and k, where $C'_{\mathsf{s}'} = C_{\mathsf{dec}} \circ C_{\mathsf{cascade}} \circ \hat{C}_{\mathsf{s}'} \circ C_{\mathsf{enc}}$. We need to show that C_{dec}, C_{cascade}, $\hat{C}_{\mathsf{s}'}$ and C_{enc} satisfy the properties of Definition 10 against $\mathcal{A}^t_{\mathsf{g}}$, and we do so by focusing on gates with big fan-out, in which case the gate adversary obtains significant advantage over the wire adversary.

We define S_0 to be the subset of $\{0^{2kt}, 1^{2kt}\}^k$ such that for each $\mathbf{x} = (x_1^{2kt}, \ldots, x_k^{2kt}) \in S_0$, $x_1 \oplus \ldots \oplus x_k = 0$. $\mathcal{D}_0$ is the uniform distribution over S_0. Symmetrically, S_1 is the subset of $\{0^{2kt}, 1^{2kt}\}^k$ such that for each $\mathbf{x} = (x_1^{2kt}, \ldots, x_k^{2kt}) \in S_1$, $x_1 \oplus \ldots \oplus x_k = 1$, and $\mathcal{D}_1$ is the uniform distribution over S_1. We prove next the required properties per Definition 10 (for convenience we do not prove them in the same order they appear in the definition).

i. **Property 2.** The encoding phase of [IPSW06] is a circuit that consists of gates in $\{\vee, \wedge, \neg\}$, and encodes each input bit x_i of C_{s} to $(r_1^{2kt}||\ldots||r_k^{2kt})$, where each r_j, $j \in [k-1]$, is generated by a randomness gate with fan-out equal to a multiple of $2kt$, and $r_k = r_1 \oplus \ldots \oplus r_{k-1} \oplus x_i$. In our setting, r_j^{2kt} is the XOR of the output of k circuits in $\bar{\mathcal{C}}_{\mathsf{prg}_{i,j}} = \{\bar{C}^1_{\mathsf{prg}_{i,j}}, \ldots, \bar{C}^k_{\mathsf{prg}_{i,j}}\}$, with private memories of length $O(k^3tl)$, initialized as described above. Recall that l is the number of AND gates of each pseudo-random generator. Now, let d be the length of the seed of the PRNG before the transformation. For any tampering strategy $\mathcal{T}_{\mathsf{g}}$, with $|\mathcal{T}_{\mathsf{g}}| \le t$, we define a simulator $\mathcal{S}_{\mathsf{enc}}$ such

that for every $\mathbf{x} \in \{0,1\}^n$, $\mathcal{S}_{\mathsf{enc}}(d, \mathcal{T}_{\mathsf{g}}, \mathbf{x}) \approx_k C_{\mathsf{enc}}^{\mathcal{T}_{\mathsf{g}}}(\mathbf{x})$:
Input: d, $\mathcal{T}_{\mathsf{g}}$, $\mathbf{x} = (x_1, \ldots, x_n)$.

(a) (PRNG **private memory sampling**) The simulator produces its own randomness in order to initialize the private memory of the PRNG's: for $i \in [n]$, $j \in [k-1]$, $z \in [k]$, generate $kd + l \cdot \frac{k(k-1)}{2}$ random bits and replicate each bit $2kt$ times. Let $\mathsf{Mem}_{i,j}^z$ be the resulting string, $\mathbf{Mem}_{i,j} = \{\mathsf{Mem}_{i,j}^1, \ldots, \mathsf{Mem}_{i,j}^k\}$ and $\mathsf{MEM}^i = \{\mathbf{Mem}_{i,1}, \ldots, \mathbf{Mem}_{i,k-1}\}$, i.e., MEM^i consists of private memory strings of the PRNG's that produce the $k-1$ random bits needed for the encoding of x_i, as described above.

(b) Output $h(\mathbf{x}, \mathcal{T}_{\mathsf{g}}, (\mathsf{MEM}^1, \ldots, \mathsf{MEM}^n))$, where h is a function whose circuit implementation is the deterministic circuit $\bar{C}_{\mathsf{enc}}$ which is derived by C_{enc} as follows: (i) for $i \in [n]$, $j \in [k-1]$, the private memory of each $\bar{C}_{\mathsf{prg}_{i,j}}^z \in \bar{\mathcal{C}}_{\mathsf{prg}_{i,j}} = \{\bar{C}_{\mathsf{prg}_{i,j}}^1, \ldots, \bar{C}_{\mathsf{prg}_{i,j}}^k\}$, $z \in [k]$, has been initialized with $\mathbf{Mem}_{i,j}$ as discussed in the above step, and (ii) for each $(f, \mathsf{g}) \in \mathcal{T}_{\mathsf{g}}$, g is replaced by g' which implements f. In other words, $h(\mathbf{x}, \mathcal{T}_{\mathsf{g}}, (\mathsf{MEM}^1, \ldots, \mathsf{MEM}^n)) = \bar{C}_{\mathsf{enc}}(\mathbf{x})$, and since the private memories of the PRNGs employed by $\bar{C}_{\mathsf{enc}}(\mathbf{x})$ and $C_{\mathsf{enc}}(\mathbf{x})$ are initialized with bit-strings that follow the same distribution, we have $C_{\mathsf{enc}}^{\mathcal{T}_{\mathsf{g}}}(\mathbf{x}) \approx_k \bar{C}_{\mathsf{enc}}^{\mathcal{T}_{\mathsf{g}}}(\mathbf{x})$.

Informally, $\mathcal{S}_{\mathsf{enc}}$ receives the circuit input $\mathbf{x}$, the adversarial strategy $\mathcal{T}_{\mathsf{g}}$ and the length of seed which is a function the security parameter k, samples the randomness needed by the PRNGs by flipping her own coins, and simulates the tampering effect which depends only on the circuit input and the generated randomness. Notice that in order to effectively eliminate one random bit, the adversary has to tamper with k PRNG circuits. Hence, she produces an invalid encoding with all but negligible probability in k, which propagates through the C_{AND} and C_{NOT} gates to the circuit's output. The replacement of randomness gates prevents the derandomization of the encoded input, which apparently does not affect Property 2 of Definition 10, but it is a major component for Properties 1 and 3. The decoder is just a deterministic circuit with no special properties. Therefore, Property 2 of Definition 10 follows.

ii. **Property 3.** Each input and secret state bit of C is encoded according to the encoding described above. The main computation is being handled by the circuit $\hat{C}_{\mathsf{s}'}$, which is derived by C by replacing each of its wires with $2k^2t$ wires and each atomic gate with a special gate that performs computations over encoded values. Concretely, consider an AND gate of C with input $(x, y) \in \{0,1\}^2$, such that $x \wedge y = z$. Such a gate is substituted by circuit C_{AND} which receives $(x_1^{2kt} || \ldots || x_k^{2kt})$ and $(y_1^{2kt} || \ldots || y_k^{2kt})$ and outputs $(z_1^{2kt} || \ldots || z_k^{2kt})$. Here $z_i = x_i y_i \oplus \bigoplus_{j \neq i} R_{i,j}$, for $1 \leq i < j \leq k$, $R_{i,j}$ is a random bit and $R_{j,i} = (R_{i,j} \oplus x_i y_j) \oplus x_j y_i$, i.e., z_i is one of the k shares of z. In the middle-stage compiler of [IPSW06], each $R_{i,j}$ is the output of a randomness gate with fan-out $O(2kt)$, while now, $R_{i,j}$ comes from the XORed output of k pseudo-random generators with $2kt$ output gates each. As we discussed above, this modification prevents the adversary from eliminating randomness employed by the construction and preserves simulatability. Now, for $1 \leq i < j \leq k$, consider circuits C_i^* that compute $z_i = x_i y_i \oplus \bigoplus_{j \neq i} R_{i,j}$, implemented by standard AND and XOR gates. Recall that in T_{comp} blowing up z_i into z_i^{2kt} is achieved as follows. Each AND gate of C_i^* is substituted by a circuit $\bar{C}_{\mathsf{AND}}$ which receives $4kt$ input bits, outputs $2kt$ bits and realizes the following functionality:

- $\bar{C}_{\mathsf{AND}}(0^{2kt}, 0^{2kt}) = 0^{2kt}$,
- $\bar{C}_{\mathsf{AND}}(0^{2kt}, 1^{2kt}) = 0^{2kt}$,
- $\bar{C}_{\mathsf{AND}}(1^{2kt}, 0^{2kt}) = 0^{2kt}$,
- $\bar{C}_{\mathsf{AND}}(1^{2kt}, 1^{2kt}) = 1^{2kt}$,
- $\bar{C}_{\mathsf{AND}}(*^{2kt}, *^{2kt}) = 0^{kt}1^{kt}$.

Each output bit of $\bar{C}_{\mathsf{AND}}$ is computed by a subcircuit of the form "OR of ANDs" of the input bits or their NOTs, or a "NOT" of such a circuit. Let $\bar{C}_i^*$ be the resulting circuit. The circuit C_{AND} is constructed by k such circuits, where each $\bar{C}_i^*$, $i \in [k]$, computes z_i^{2kt}. Regarding the $\oplus$-gates the circuit $\bar{C}_{\mathsf{XOR}}$ is used that was described in the beginning of this section.

One can easily verify that if $(x, y) \in \{0,1\}^2$ and $x \wedge y = z$, then

$$C_{\mathsf{AND}}((x_1^{2kt}||\ldots||x_k^{2kt}),(y_1^{2kt}||\ldots||y_k^{2kt})) = (z_1^{2kt}||\ldots||z_k^{2kt}),$$

i.e., C_{AND} preserves correctness (Property 3a). Moreover, suppose that one of the inputs, say $\mathbf{x} = \mathbf{x}_1||\ldots||\mathbf{x}_k$, is invalid. Then the $\bar{C}_{\mathsf{AND}}$ and $\bar{C}_{\mathsf{XOR}}$ gadgets that implement C_{AND} guarantee that $C_{\mathsf{AND}}(\mathbf{x},\mathbf{y}) \in S_\perp$. Specifically, if $C_{\mathsf{AND}}(\mathbf{x},\mathbf{y}) = (\mathbf{z}'_1||\ldots||\mathbf{z}'_k)$, then for some $j \in [k]$, $\mathbf{z}_j = 0^{kt}1^{kt}$.

Now suppose at least one of the inputs, say $\mathbf{x}$, belongs to $S_\perp$, and consider a gate adversary who tampers with at most $t-1$ gates of C_{AND}.[9] Since $\mathbf{x} = (\mathbf{x}_1||\ldots||\mathbf{x}_k) \in S_\perp$, there exists some $i \in [k]$ such that $\mathbf{x}_i$ contains both 0-bits and 1-bits. The length of each $\mathbf{x}_i$ is $2kt$ and is given as input to some $\bar{C}_{\mathsf{AND}}$ or $\bar{C}_{\mathsf{XOR}}$ circuit, i.e., the circuits that implement the AND and XOR gates of C_{AND}, respectively. We need to verify that the invalid encoding propagates even if the gate adversary tampers with the internals of those gadgets: recall that each output bit of those gadgets is computed by a subcircuit of the form "OR of ANDs" of the input bits or their NOTs, or a "NOT" of such a circuit. Therefore, we have $2kt$ subcircuits which independently compute $2kt$ output bits. Since the adversary can tamper with up to $t-1$ gates in total, she may alter the functionality of up to $t-1$ of the $2kt$ subcircuits. On input $\mathbf{x}_i$, an untampered $\bar{C}_{\mathsf{AND}}$ gadget would output $0^{kt}1^{kt}$, and therefore, the adversary is far from producing 0^{2kt} or 1^{2kt}. Hence, the error propagates through $\bar{C}_{\mathsf{AND}}$ and $\bar{C}_{\mathsf{XOR}}$, and Property 3b follows for C_{AND}. The case for C_{NOT} is similar.

We prove Property 3c by reviewing the simulator of [IPSW06] against adversaries who tamper with wires inside the circuit gadgets. In [IPSW06], and due to the randomization techniques employed by the circuit, the simulator knows the probability distribution of the input to the gadget and generates samples according to it. Then she decides the tampering effect given the adversary's tampering strategy $\mathcal{T}_{\mathsf{w}}$. The encoding also ensures that each tampering attack against C_{AND} produces an element in $S_\perp$ with constant probability, and therefore, if the attacker tampers with t different C_{AND} circuits, an element in $S_\perp$ is produced with all but negligible probability in k. In our case, any adversary who tampers with up to t circuit gates, affects up to t circuit gadgets and the attacks can be simulated exactly as in [IPSW06]. Notice that the effect of any attack on a circuit that implements a pseudo-random generator is simulatable as in C_{enc}.

iii. **Property 1.** Property 1a of Definition 10 does not depend on the adversarial model and follows directly from [IPSW06] given the above description.

In [IPSW06] the adversary is allowed to tamper with up to t circuit wires in each computation. Let $\mathbf{x} = (x_1^{2kt}||\ldots||x_k^{2kt})$ be the encoding of $x \in \{0,1\}$, where each $x_i \in \{0,1\}$, for $i \in [k]$, is a share of x which incorporates the circuit's randomness (see above). The value x is carried by a circuit wire on C, while $\mathbf{x}$ is carried by a wire-bundle on $\hat{C}_{\mathbf{s}'}$. Any wire adversary needs to tamper with at least kt wires in order to change $\mathbf{x}$ to $\mathbf{x}' \neq \mathbf{x}$, in a way such that $\mathbf{x}' \notin S_\perp$. Hence, if the adversary tampers with up to t wires of the wire-bundle that carries $\mathbf{x}$, she leaves the value of the bundle unchanged or she induces an element in $S_\perp$. Moreover, the randomization employed by the encoding, both for the case in which $\mathbf{x}$ is the encoding of an input or secret state bit, as well as for the case in which $\mathbf{x}$ is the output of a circuit that implements an atomic gate of C (recall C_{AND}, C_{NOT} and the ideas of Property 3), guarantees that (i) each tampering attack on wires that carry $\mathbf{x}$ induces a fault with constant probability and (ii) the simulatability of the tampering effect, which follows by the simulatability of the randomness incorporated in each x_i, given $\mathcal{T}$. If the adversary tampers with wires that lie on distinct subbundles, the probability of inducing a fault increases. The gate adversary does not tamper with the wires that carry $\mathbf{x}$ directly. She instead tampers with the gates that produce $\mathbf{x}$. Hence, we consider the following cases:

- If $\mathbf{x}$ is an encoded secret state bit, and since $\mathbf{x} \in \{0,1\}^{2k^2t}$, then $\mathbf{x}$ is the output of $2k^2t$ memory gates.[10] Hence, the gate adversary may alter up to t bits of $\mathbf{x}$, and apparently, she cannot change the value of $\mathbf{x}$ without producing an element in $S_\perp$.

[9]Since $\mathbf{x} \in S_\perp$, the attacker has already applied at least one tampering action.

[10]Recall that each memory gate has one input and one output wire.

- Let $\mathbf{x} = (\mathbf{x}_1 || \dots || \mathbf{x}_k)$, where $\mathbf{x}_i$ is of length $2kt$ and assume that $\mathbf{x}$ is the output of some C_{AND} circuit. Then each $\mathbf{x}_i$ is the output of some circuit gadget ($\bar{C}_{\mathsf{AND}}$ or $\bar{C}_{\mathsf{XOR}}$) or it comes from a PRNG circuit. By definition of those components, each of their $2kt$ bits is produced by a distinct circuit gate, and the gate adversary may alter up to t of their bits.

Hence, any adversary $\mathcal{A}^t_{\mathsf{g}}$ who can alter up to t bits of $\mathbf{x}$ cannot change the value of $\mathbf{x}$ without producing an element in $S_\perp$, and the simulatability of the tampering effect is proved as in property 3.

iv. **Property 4.** The cascade phase of [IPSW06] is a deterministic circuit which receives m' wire-bundles, outputs m' wire-bundles, and enforces error propagation and memory erasure without employing any randomness. As in $\hat{C}_{\mathbf{s}'}$, C_{cascade} employs special tamperable gadgets, call them C_{check}, that receive $4kt$ input bits, i.e., they receive pairs of subbundles, and output $4kt$ bits. The gadgets realize the following functionality:

- $C_{\mathsf{check}}(0^{2kt}, 0^{2kt}) = (0^{2kt}, 0^{2kt})$,
- $C_{\mathsf{check}}(0^{2kt}, 1^{2kt}) = (0^{2kt}, 1^{2kt})$,
- $C_{\mathsf{check}}(1^{2kt}, 0^{2kt}) = (1^{2kt}, 0^{2kt})$,
- $C_{\mathsf{check}}(1^{2kt}, 1^{2kt}) = (1^{2kt}, 1^{2kt})$,
- $C_{\mathsf{check}}(*^{2kt}, *^{2kt}) = (0^{kt}1^{kt}, 0^{kt}1^{kt})$.

Each output bit of C_{check} is the output of a subcircuit of the form "OR of ANDs" of its input wires or their NOTs or a "NOT" of such a circuit. Hence, a gate adversary who tampers with up to t circuit gates may alter the functionality of up to t subcircuits of C_{check}. Clearly, the attacker cannot modify the value of any bundle without producing an invalid encoding, and moreover, she cannot change any invalid encoding to a valid one since she needs to tamper with at least kt gates. Hence, we derive that both wire and gate adversaries have the same advantage against C_{cascade}, and the proof of Property 4 against gate adversaries follows by [IPSW06]. Here we give the main idea behind the proof.

Suppose the input to $C_{\mathbf{s}}$ is $\mathbf{x} = (x_1, \dots, x_n)$ with encoding $\bar{\mathbf{x}}$, and let $\hat{C}^{\mathcal{T}_{\mathsf{g}}}_{\mathbf{s}'}(\bar{\mathbf{x}}) = (\mathbf{y}_1, \dots, \mathbf{y}_{m'}) = \bar{\mathbf{y}}$ and for all i, $1 \le i \le m'$, $\mathbf{y}_i \in S_0 \cup S_1$, i.e., the output of $\hat{C}_{\mathbf{s}'}(\bar{\mathbf{x}})$ does not contain invalid encodings. By the definitions of C_{cascade} and C_{check}, we have that for any untampered computation $[C_{\mathsf{cascade}}(\bar{\mathbf{y}})]_i = \mathbf{y}_i$, and moreover, the output distribution of the wire-bundles of $C_{\mathsf{cascade}}(\bar{\mathbf{y}})$ that feed C_{dec} is simulatable given black box access to $C_{\mathbf{s}}$: the simulator computes $\mathbf{y} = C_{\mathbf{s}}(\mathbf{x})$, samples her own randomness so as to encode $\mathbf{y}$ into $\bar{\mathbf{y}}'$ and computes $C_{\mathsf{cascade}}(\bar{\mathbf{y}}')$. Since $\bar{\mathbf{y}} \approx_k \bar{\mathbf{y}}'$, we have that $C_{\mathsf{cascade}}(\bar{\mathbf{y}}') \approx_k C_{\mathsf{cascade}}(\bar{\mathbf{y}})$. $C^{\mathcal{T}_{\mathsf{g}}}_{\mathsf{cascade}}(\bar{\mathbf{y}})$ is simulatable in a similar way: the simulator having black box access to $C_{\mathbf{s}}$, receives $\mathbf{y} = C_{\mathbf{s}}(\mathbf{x})$, samples her own randomness so as to encode $\mathbf{y}$ into $\bar{\mathbf{y}}'$. Then she supplies C_{cascade} with $\bar{\mathbf{y}}'$ and applies the adversarial strategy $\mathcal{T}_{\mathsf{g}}$ on it. Regarding the wire-bundles of C_{cascade} that feed the circuit's private memory, for any attack on the gates of C_{cascade} there exists a simulator which decides if the attack produces an invalid encoding. The description of the simulator is similar to the one we described in property 3.

If there exists i, $1 \le i \le m'$, such that $\mathbf{y}_i \in S_\perp$, then by the definition of C_{check}, $[C_{\mathsf{cascade}}(\bar{\mathbf{y}})]_i \in S_\perp$, and more specifically, $[C_{\mathsf{cascade}}(\bar{\mathbf{y}})]_i = (0^{kt}1^{kt})^k$, for all $i \in [m']$. By the description of C_{check} and the structure of C_{cascade} (see [IPSW06]), $[C^{\mathcal{T}_{\mathsf{g}}}_{\mathsf{cascade}}(\bar{\mathbf{y}})]_i \in S_\perp$, for all $i \in [m']$. Since for any tampering action against $\hat{C}_{\mathbf{s}'}$ we have proved the existence of a simulator that predicts the result of the tampering effect as a distribution between producing invalid encodings or not, and due to the error propagation functionality of C_{AND}, C_{NOT}, there exists a simulator $\mathcal{S}$ who "predicts" the output wire-subbundles of $\hat{C}^{\mathcal{T}_{\mathsf{g}}}_{\mathbf{s}'}(\bar{\mathbf{x}})$ that carry invalid encodings. In other words, there exists a simulator $\mathcal{S}$ which receives $\mathcal{T}_{\mathsf{g}}$, $\mathbf{x}$, $\bar{\mathbf{x}}$, and for each i, j outputs 1 iff the j-th wire-subbundle, of the i-th wire bundle carries an invalid encoding. We define a simulator $\mathcal{S}'$ for $C^{\mathcal{T}_{\mathsf{g}}}_{\mathsf{cascade}}(\bar{\mathbf{y}})$ as follows:
Input: $\mathcal{T}_{\mathsf{g}}$, $\mathbf{x}$, $\bar{\mathbf{x}}$.

(a) Run $\mathcal{S}(\mathcal{T}_{\mathsf{g}}, \mathbf{x}, \bar{\mathbf{x}})$ and denote by $V_{i,j}$ its output for each $i \in [m']$ and $j \in [k]$.

(b) For $i \in [m']$, if for all $j \in [k]$, $V_{i,j} = 0$, then generate randomness so as to produce a valid encoding of $[C_{\mathbf{s}}(\mathbf{x})]_i$ and set $\mathbf{y}_i$ equal to that value. Otherwise, set $\mathbf{y}_i$ equal to an invalid encoding by taking into account the adversarial strategy $\mathcal{T}_{\mathsf{g}}$ against $\hat{C}_{\hat{\mathbf{s}}}$.

(c) Output $h'(\mathbf{y}_1, \ldots, \mathbf{y}_n)$, where h' is a function whose circuit implementation is the deterministic circuit $\bar{C}_{\mathsf{cascade}}$ which is derived by C_{cascade} if for each $(f, \mathsf{g}) \in \mathcal{T}_{\mathsf{g}}$, we replace g by g', where g' implements f.

The invalid encoding propagates to the circuit's secret state and output, and Property 4 follows. ■

Security for multiple rounds. Since the attacks we consider are persistent, a multi-round adversary may alter the value of more than kt bits of the encoding by tampering with t circuit gates in each round. However, if the adversary alters the value of t bits of the encoding during a round, due to the randomization of the subbundle's signals, an element in $S_\perp$ is produced with constant probability. Observe that in order for the simulation to fail at least kt bits should be affected; nevertheless, a multi-round adversary will succeed in modifying kt bits only with negligible probability in k.

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* $\mathbf{Env} = (\mathbf{s}, \mathbf{v})$ *be a pair of random variables,* $C_{\mathbf{s}}$ *any boolean circuit,* T *a* (t, k)*-secure circuit compiler for* t, $k \in \mathbb{N}$*, and let* $C'_{\mathbf{s}'}$ *be the secure transformation of* $C_{\mathbf{s}}$ *with respect to* T*. Then for every multi-round tampering adversary* $\mathcal{A}$ *that in each round follows a tampering strategy* $\mathcal{T}$ *with* $|\mathcal{T}| \leq t$ *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 *w.r.t.* $\mathbf{Env}$.

Proof. Let $\mathcal{A}$ be any tampering adversary. We construct a simulator $\mathcal{S}$ such that $\mathcal{S}^{C_{\mathbf{s}}(\cdot)}(\mathbf{v}) \approx_k \mathcal{A}^{C'^{*}_{\mathbf{s}'}(\cdot)}(\mathbf{v})$. We first prove the case of single-round adversaries; with this as a basis we then tackle multi-round adversaries.

Suppose that $\mathcal{A}$ is a single round tampering adversary against $C'_{\mathbf{s}'} = C_{\mathsf{dec}} \circ C_{\mathsf{cascade}} \circ \hat{C}_{\mathbf{s}'} \circ C_{\mathsf{enc}}$, with strategy $\mathcal{T}$ and chosen circuit input $\mathbf{x}$. Notice that we purposefully leave the nature of the tampering attacks undetermined, but we assume their effects do not break the properties of definition 10. Now, by property 2 of definition 10 there exists a simulator $\mathcal{S}_1$ s.t. $\mathcal{S}_1(\mathcal{T}, \mathbf{x}, \mathbf{v}) \approx_k C^{\mathcal{T}}_{\mathsf{enc}}(\mathbf{x})$. We construct a simulator $\mathcal{S}_2$ which outputs a distribution over $\{0, 1\}$, where the "1" output indicates that there exists a wire-bundle that feeds C_{cascade} and carries an invalid encoding, and the "0" output indicates that all wire-bundles that feed C_{cascade} carry valid encodings. We define $\mathcal{S}_2$ as follows:
Input: $\mathcal{S}_2$ receives $\mathcal{T}$, $\mathbf{x}$, $\mathbf{v}$.

1. $\mathcal{S}_2$ executes $\mathcal{S}_1(\mathcal{T}, \mathbf{x}, \mathbf{v})$ to obtain $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Let $\mathcal{I} \subseteq [n]$ be the set of indexes s.t. if $j \in \mathcal{I}$, then $\mathbf{x}_j$ is an invalid encoding, or in other words, the j-th input wire-bundle of $\hat{C}_{\hat{s}}$ carries an invalid encoding. If $\mathcal{I} \neq \emptyset$ then due to property 3b of definition 10 (error propagation), we know that an invalid encoding will reach C_{cascade}. Hence, if $\mathcal{I} \neq \emptyset$, $\mathcal{S}_2$ outputs 1 and terminates, else it proceeds to step 2.

2. Let $G(V, E)$ be the graph that represents $\hat{C}_{\hat{\mathbf{s}}}$, i.e., if $e \in E$, then e represents a wire-bundle in $\hat{C}_{\hat{\mathbf{s}}}$ and for every $v \in V$, v represents a C_{AND}, C_{NOT} or memory gate in $\hat{C}_{\hat{\mathbf{s}}}$. $\mathcal{S}_2$ performs a breadth-first search traversal of G until it finds a circuit component for which there exists a tampering strategy in $\mathcal{T}$ against it. Clearly, any action in $\mathcal{T}$ may be an attack against the circuit outside of the mega-gates (C_{AND}, C_{NOT}) or within such gates. In the first case, the attack may affect either a memory gate or a wire-bundle that contains an encoded bit, i.e., it is an attack against the encoding of values in the circuit. Then, by property 1b of definition 10, for every such action there exists a simulator which given only the tampering action predicts the tampering effect as a distribution between leaving the value of the encoding unchanged and producing an element in $S_\perp$. In the second case, we use property 3c from which we are guaranteed a simulator w.r.t. attacks against the circuit gates within C_{AND} or C_{NOT} that as before will predict whether a fault is produced or not. In the course of the traversal, for each tampering action in $\mathcal{T}$ the simulator $\mathcal{S}_2$ executes the

corresponding simulator. Concretely, let $\mathcal{T}_1 \subseteq \mathcal{T}$ be the strategy of $\mathcal{A}$ that affects solely $\hat{C}_{\hat{\mathsf{s}}}$. For any tampering action $a_i \in \mathcal{T}_1$, there exists a simulator $\bar{\mathcal{S}}_i$ that meets the properties mentioned above. If some $\bar{\mathcal{S}}_i(\mathcal{T}, \mathbf{x}, \bar{\mathbf{x}}, \mathbf{v})$ outputs 1, $i \in |\mathcal{T}_1|$, then $\mathcal{S}_2$ outputs 1 and terminates. Otherwise, it continues the BFS traversal of G having the strategy described above. If for all $i \in |\mathcal{T}_1|$, $\bar{\mathcal{S}}_i$ outputs 0, then $\mathcal{S}_2$ outputs 0 and terminates.

Now we define the simulator for C_{cascade}, $\mathcal{S}_3$:
Input: $\mathcal{S}_3$ receives $\mathcal{T}$, $\mathbf{x}$, $\mathbf{v}$.

1. $\mathcal{S}_3$ executes $\mathcal{S}_2(\mathcal{T}, \mathbf{x}, \mathbf{v})$.

2. If $\mathcal{S}_1(\mathcal{T}, \mathbf{x}, \mathbf{v}) = 0$, then by property 4a of definition 10, there exist simulators $\mathcal{S}^1_{\mathsf{c}}$, $\mathcal{S}^2_{\mathsf{c}}$ and $\mathcal{S}_{\mathsf{mem}}$ s.t. $\mathcal{S}^1_{\mathsf{c}}(C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}) \approx_k C_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$, $\mathcal{S}^2_{\mathsf{c}}(\mathcal{T}, C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}) \approx_k C^{\mathcal{T}}_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$ and $\mathcal{S}_{\mathsf{mem}}(\mathcal{T}, C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}) = 1$ iff an element in $S_\perp$ feeds the circuit's private memory. Here $(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$ denotes the input to C_{cascade} w.r.t. $\hat{C}_{\mathsf{s}'}(\bar{\mathbf{x}})$. If there exists a tampering action in $\mathcal{T}$ against C_{cascade}, then $\mathcal{S}_3$ returns $(\mathcal{S}^2_{\mathsf{c}}(\mathcal{T}, C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}), \mathcal{S}_{\mathsf{mem}}(\mathcal{T}, C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}))$. Otherwise returns $(\mathcal{S}^1_{\mathsf{c}}(C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}), 0)$.

3. If $\mathcal{S}_1(\mathcal{T}, \mathbf{x}, \mathbf{v}) = 1$, then by property 4b of definition 10, there exist simulators $\mathcal{S}^3_{\mathsf{c}}$, $\mathcal{S}^4_{\mathsf{c}}$ s.t. $\mathcal{S}^3_{\mathsf{c}}(C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}) \approx_k C_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$ and $\mathcal{S}^4_{\mathsf{c}}(\mathcal{T}, C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}) \approx_k C^{\mathcal{T}}_{\mathsf{cascade}}(\mathbf{y}_1, \ldots, \mathbf{y}_{m'})$. Again, if there exists a tampering action in $\mathcal{T}$ against C_{cascade}, then $\mathcal{S}_3$ returns $(\mathcal{S}^4_{\mathsf{c}}(\mathcal{T}, C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}), 1)$. Otherwise returns $(\mathcal{S}^3_{\mathsf{c}}(C_{\mathbf{s}}(\mathbf{x}), \mathbf{x}, \mathbf{v}), 1)$.

The single-round simulator $\mathcal{S}_{\mathsf{r}}$ is the composition of the simulators defined above. Concretely, $\mathcal{S}_{\mathsf{r}}$ receives $(\mathcal{T}, \mathbf{x}, \mathbf{v})$, computes $(a, b) = \mathcal{S}_3(\mathcal{T}, \mathbf{x}, \mathbf{v})$, returns $(C_{\mathsf{dec}}(a), b)$, and since the statistical distance between $\mathcal{S}_3(\mathcal{T}, \mathbf{x}, \mathbf{v})$ and the output of C_{cascade} that feeds C_{dec} is negligible in k, and C_{dec} is a stateless deterministic circuit which performs the decoding (hence any tampering attack against C_{dec} can be trivially simulated) we have that ${C'}^{\mathcal{T}}_{\mathsf{s}'}(\mathbf{x}) \approx_k C_{\mathsf{dec}}(a)$.

The result easily extends to the multi-round case since the output of $\mathcal{S}_3$ indicates if an element in $S_\perp$ is fed into the circuit's private memory, during the next computation, the circuit's error propagation mechanism enforces memory erasure, something that ensures that all output wire-bundles of C_{cascade} will carry invalid encodings. ■

References

[AK96] Ross Anderson and Markus Kuhn. Tamper resistance-a cautionary note. In *Proceedings of the second Usenix workshop on electronic commerce*, volume 2, pages 1–11, 1996.

[BDL97] Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of checking cryptographic protocols for faults. In *Advances in CryptologyEUROCRYPT97*, pages 37–51. Springer, 1997.

[BDL01] Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of cryptology*, 14(2):101–119, 2001.

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Advances in CryptologyCRYPTO 2001*, pages 1–18. Springer, 2001.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology CRYPTO'97*, pages 513–525. Springer, 1997.

[BS03] Johannes Blömer and Jean-Pierre Seifert. Fault based cryptanalysis of the advanced encryption standard (aes). In *Financial Cryptography*, pages 162–181. Springer, 2003.

[BSGH+04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust pcps of proximity, shorter pcps and applications to coding. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 2004.

[CKM11] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. Bitr: built-in tamper resilience. In *Advances in Cryptology–ASIACRYPT 2011*, pages 740–758. Springer, 2011.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452. Tsinghua University Press, 2010.

[DSK12] Dana Dachman-Soled and Yael Tauman Kalai. Securing circuits against constant-rate tampering. In *Advances in Cryptology–CRYPTO 2012*, pages 533–551. Springer, 2012.

[FPV11] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In *Automata, Languages and Programming*, pages 391–402. Springer, 2011.

[GA03] Sudhakar Govindavajhala and Andrew W Appel. Using memory errors to attack a virtual machine. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 154–165. IEEE, 2003.

[GG94] Péter Gács and Anna Gál. Lower bounds for the complexity of reliable boolean circuits with noisy gates. *Information Theory, IEEE Transactions on*, 40(2):579–583, 1994.

[GLM$^+$04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *Theory of Cryptography*, pages 258–277. Springer, 2004.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded SystemsCHES 2001*, pages 251–261. Springer, 2001.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[GS95] Anna Gal and Mario Szegedy. Fault tolerant circuits and probabilistically checkable proofs. In *Structure in Complexity Theory Conference, 1995., Proceedings of Tenth Annual IEEE*, pages 65–73. IEEE, 1995.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In *Advances in Cryptology-EUROCRYPT 2006*, pages 308–327. Springer, 2006.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology-CRYPTO 2003*, pages 463–481. Springer, 2003.

[KA98] Markus G Kuhn and Ross J Anderson. Soft tempest: Hidden data transmission using electromagnetic emanations. In *Information Hiding*, pages 124–142. Springer, 1998.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in CryptologyCRYPTO99*, pages 388–397. Springer, 1999.

[KL08] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography.* Chapman & Hall, 2008.

[Koc96] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in CryptologyCRYPTO96*, pages 104–113. Springer, 1996.

[KSWH98] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side channel cryptanalysis of product ciphers. In *Computer SecurityESORICS 98*, pages 97–110. Springer, 1998.

[LL10] Feng-Hao Liu and Anna Lysyanskaya. Algorithmic tamper-proof security under probing attacks. In *Security and Cryptography for Networks*, pages 106–120. Springer, 2010.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *Theory of Cryptography*, pages 278–296. Springer, 2004.

[Pip85] Nicholas Pippenger. On networks of noisy gates. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, pages 30–38. IEEE, 1985.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, pages 200–210. Springer, 2001.

[RR01] Josyula R Rao and Pankaj Rohatgi. Empowering side-channel attacks. *IACR ePrint*, 37, 2001.

[SA03] Sergei P Skorobogatov and Ross J Anderson. Optical fault induction attacks. In *Cryptographic Hardware and Embedded Systems-CHES 2002*, pages 2–12. Springer, 2003.