Private Circuits: Securing Hardware against Probing Attacks

Yuval Ishai[1], Amit Sahai[2], and David Wagner[3]

[1] Technion — Israel Institute of Technology,[†] yuvali@cs.technion.ac.il
[2] Princeton University, sahai@cs.princeton.edu
[3] University of California, Berkeley, daw@cs.berkeley.edu

Abstract. Can you guarantee secrecy even if an adversary can eavesdrop on your brain? We consider the problem of protecting privacy in *circuits*, when faced with an adversary that can access a bounded number of wires in the circuit. This question is motivated by side channel attacks, which allow an adversary to gain partial access to the inner workings of hardware. Recent work has shown that side channel attacks pose a serious threat to cryptosystems implemented in embedded devices. In this paper, we develop theoretical foundations for security against side channels. In particular, we propose several efficient techniques for building *private circuits* resisting this type of attacks. We initiate a systematic study of the complexity of such private circuits, and in contrast to most prior work in this area provide a formal threat model and give proofs of security for our constructions.

Keywords: cryptanalysis, side channel attacks, provable security, secure multi-party computation, circuit complexity.

1 Introduction

This paper concerns the following fascinating question: Is it possible to maintain secrecy even if an adversary can eavesdrop on your brain? A bit more precisely, can we guarantee privacy when one of the basic assumptions of cryptography breaks down, namely, when the adversary can gain access to the insides of the hardware that is making use of our secrets? We formalize this question in terms of protecting privacy in *circuits*, where an adversary can access a bounded number of wires in the circuit. We initiate the study of this problem and present several efficient techniques for achieving this new type of privacy. Before describing the model and our contribution in more detail, we motivate the problem by providing some necessary background.

1.1 Background

Our understanding of cryptography has made tremendous strides in the past three decades, fueled in large part by the success of analysis- and proof-driven design. Most such work has analyzed algorithms, not implementations: typically one thinks of a cryptosystem as a black box implementing some mathematical function and implicitly assumes the implementation faithfully outputs what the function would (and nothing else). However, in practice implementations are not always a true black box: partial information about internal computations can be leaked (either directly or through *side-channels*), and this may put security at risk.

[†] Work done in part while at Princeton University.

This difference between implementations and algorithms has led to successful attacks on many cryptographic implementations, even where the underlying algorithm was quite sound. For instance, the power consumed during an encryption operation or the time it takes for the operation to complete can leak information about intermediate values during the computation [26, 27], and this has led to practical attacks on smartcards. Electromagnetic radiation [34, 17, 35], compromising emanations [37], crosstalk onto the power line [38, 36], return signals obtained by illuminating electronic equipment [3, 36], magnetic fields [33], cache hit ratios [25, 31], and even sounds given off by rotor machines [24] can similarly give the attacker a window of visibility on internal values calculated during the computation. Also of interest is the *probing attack*, where the attacker places a metal needle on a wire of interest and reads off the value carried along that wire during the smartcard's computation [2]. In general, side channel attacks have proven to be a significant threat to the security of embedded devices.

The failure of proof-driven cryptography to anticipate these risks comes from an implicit assumption in many[1] currently accepted definitions in theoretical cryptography, namely, the *secrecy assumption*. The secrecy assumption states that legitimate participants in a cryptographic computation can keep intermediate values and key material secret during a local computation. For instance, by modeling a chosen-plaintext attack on the encryption scheme E as an algorithm A^{E_k} with oracle access to E_k, we implicitly assume that the device implementing E_k outputs only $E_k(x)$ on input x, and does not leak anything else about the computation of $E_k(x)$. Thus the 'Standard Model' in theoretical cryptography often takes the secrecy assumption for granted, but as we have seen, there are a bevy of ways that the secrecy assumption can fail in real systems.

One possible reaction is to study implementation techniques that ensure the secrecy assumption will always hold. For instance, we can consider adding large capacitors to hide the power consumption, switch to dual-rail logic so that power consumption will be independent of the data, shield the device in a tamper-resistant Faraday cage to prevent information leakage through RF emanations, and so on. Many such hardware countermeasures have been proposed in the literature. However, a limitation of such approaches is that, generally speaking, each such countermeasure must be specially tailored for the set of side channels it is intended to defeat, and one can only plan a defense if one knows in advance what side channels an attacker might try to exploit. Consequently, if the designer cannot predict all possible ways in which information might leak, hardware countermeasures cannot be counted on to defend reliably against side channel attacks.

This leaves reason to be concerned that hardware countermeasures may not be enough on their own to guarantee security. If the attacker discovers a new class of side channel attacks not anticipated by the system designer, all bets are off. Given the wide variety of side channel attacks that have been discovered up till now, this seems like a significant risk: As a general rule of thumb, wherever three or four such vulnerabilities are known, it would be prudent to assume that there may be another, similar but unknown vulnerability lurking in the wings waiting to be discovered. In particular, it is hard to predict what other types of side channel pitfalls might be discovered in the

[1] This implicit assumption is definitely not universal. For instance, the field of secure multi-party computation asks for security even when some parties can be corrupted or observed.

future, and as a result, it is hard to gain confidence that any given implementation will be free of side channels. This is a "risk of the unknown", rather than a known risk[2], and risks of the unknown are the worst kind of risks to assume. Consequently, the secrecy assumption seems optimistic, and we submit that hardware countermeasures may not be the final answer.

A different possible response is to design algorithms that, when implemented, will be inherently robust against side channel attacks. For instance, Daemen and Rijmen proposed replacing each wire of a circuit by two wires, one carrying the original bit and the other its complement [16]; Messerges proposed "data masking", where each value is split into two shares using a 2-out-of-2 secret sharing scheme [28]; Goubin and Patarin suggested a "duplication" method based on similar methods [22]; and many other proposals can be found in the literature. However, none of those schemes have been proven secure, and unsurprisingly, some have since been broken [12, 15]. This experience suggests that the field needs to be put on solid theoretical foundations. For obvious reasons, we would prefer a principled approach that has been proven secure over an ad-hoc countermeasure.

1.2 Our Contribution

In this paper, we take on this challenge. Working in the context of Boolean circuits, we show how to implement cryptosystems (or any algorithm) in a way that can tolerate the presence of a large class of side channel attacks without loss of security. In particular, we show how to transform any circuit implementing some cryptographic algorithm into another, larger circuit that implements the same functionality but that will remain secure even if the attacker can observe up to any t internal bits produced during the computation within one clock cycle.

As a result, our constructions provide a generic defense against probing attacks. They are generic in the sense that we defend against a large class of attacks. To defend against information leakage, we do not need to know how the information might leak; rather, we only need to predict how much information might leak or at what rate. Our constructions are also generic in the sense that they apply to any cryptosystem of interest: rather than trying to secure just, say, AES encryption, we show that any circuit whatsoever can be made robust against probing attacks.

Also, we emphasize that our constructions are provably secure. We develop a formal model of the adversary, propose definitions of security against probing attacks, and prove that our constructions meet these definitions. This puts the field on a principled theoretical footing and removes fears that our proposals might be broken by cryptanalysis.

OUR MODEL. Ideally, we would like to achieve security against an all-powerful attacker, *i.e.* one that can observe every internal value produced during the computation. However, this task is generally impossible to achieve, as follows from the impossibility of obfuscation [4]. Instead, we settle for achieving security against adversaries that are limited in their power to observe the computation. There are many ways we could consider limiting the adversary, but in this paper we choose a simple metric: a t-limited adversary is one that can observe at most t wires of the circuit within a certain time

[2] We thank Mark Miller for introducing us to this turn of phrase.

Applies to	Privacy type	Size	Sec.	Comments
any circuit	perfect	$O(nt^2)$	§4	our basic scheme
PRG circuits	computational	$O(nt)$	§6	only applies to pseudorandom generators
any circuit	computational	$O(nt^2) + \tilde{O}(t^3)$	§6	derandomized version of basic scheme
any circuit	statistical	$\tilde{O}(nt)$	§5	
any circuit	statistical	$\tilde{O}((w+t)d)$	§5	layered circuit of width w and depth d

Table 1. A summary of our main results. Here n denotes the size of the original circuit and t the number of adversarial probes we wish to tolerate. All uses of $O()$ notation hide small constants. We use $\tilde{O}()$ to hide large constants, polylogarithmic factors, or polynomials in a security parameter.

period (such as during one clock cycle).[3] We believe this is a reasonable restriction, as most side channels give the attacker only partial information about the computation. In particular, in probing attacks the cost of micro-probing equipment is directly related to the number of needles one can manipulate at one time—a station with five probes is considerably more expensive than one with only a single probe—and so an attacker is limited in the number of wires that can be observed at any one time. Consequently, the value t is a good measure of the cost of a probing attack. We refer the reader to Section 2 for a more detailed treatment of the model, in particular for the useful case of *stateful* circuits which carry state information from one invocation to the next.

Our model can be compared to that of Chari, et al., who took a first step by analyzing k-out-of-k secret sharing in a model where the attacker can obtain a noisy view of all circuit elements [12], with applications to security against power analysis. That work, however, did not provide security against probing attacks or other side channels where the attacker can view any t wires of his choosing, and our constructions are quite different from theirs. Also of relevance are works on exposure-resilient functions and all-or-nothing transforms (e.g. [9]), which attempt to efficiently secure *storage* (but not computation) against probing attacks, and work on oblivious RAM (cf. [21]) aimed at protecting *software* by hiding the access pattern of a (trusted) CPU.

MPC ON SILICON? There is an interesting relation between the problem we study and that of secure multi-party computation (MPC). In some sense, our contribution may be viewed as a novel application of MPC techniques to the design of secure hardware. We would like to stress, however, that our focus and goals are quite different from the traditional ones in the MPC literature, and that our main results are not derived from state-of-the-art results in this area. We refer the reader to Appendix A for a detailed discussion of the relation between our problem and the MPC problem.

OUR RESULTS. Our basic results are as follows. We show that any circuit with n gates can be transformed into a circuit of size $O(nt^2)$ that is perfectly secure against all probing attacks leaking up to t bits at a time (see Section 4). This general transformation increases circuit size by a factor of $O(t^2)$, but for some specific cryptosystems we can

[3] By default, we allow the adversary to adaptively move its t probes between time periods, but not *within* a time period. See Section 2 for more details.

do better. For PRG's, we can find constructions that yield an $O(nt)$ transformed circuit size, rather than $O(nt^2)$ (Section 6). Finally, we present statistically private transformations which significantly improve the asymptotic efficiency of previous constructions, but whose concrete efficiency becomes better only when t is quite large. See Table 1 for a summary of the main results. Additional results, such as a trading circuit size for increased latency, will be included in the full version of this paper.

We do not know how practical our constructions will be. However, our results already show that the cost of security is not too high. Since many cryptosystems can be implemented quite efficiently in hardware (e.g., $n \approx 10^3$ or 10^4 gates), and since our use of big-$O()$ notation typically does not hide any large constants, it seems that security using our techniques is within the reach of modern systems. We leave a more thorough performance analysis to others.

2 Definitions

Circuits. We will examine probing attacks in the setting of Boolean circuits. A *deterministic circuit* C is a directed acyclic graph whose vertices are Boolean gates and whose edges are wires. We will assume without loss of generality that every gate has fan-in at most 2 and fan-out at most 3. A *randomized circuit* is a circuit augmented with random-bit gates. A *random-bit gate* is a gate with fan-in 0 that produces a random bit and sends it along its output wire; the bit is selected uniformly and independently of everything else afresh for each invocation of the circuit.

The size of a circuit (usually denoted by n) is defined as the number of gates and its depth is the length of the longest path from an input to an output. We will sometimes consider a width-w depth-d *layered* circuit, where the underlying graph is a depth-d layered graph with at most w wires connecting two adjacent layers.

A *stateful circuit* is a circuit augmented with memory cells. A *memory cell* is a stateful gate with fan-in 1: on any invocation of the circuit, it outputs the previous input to the gate, and stores the current input for the next invocation. Thus, memory cells act as delay elements. We extend the usual definition of a circuit by allowing stateful circuits to possibly contain cycles, so long as every cycle traverses at least one memory cell. When specifying a stateful circuit, we must also specify an initial state for the memory cells. When C denotes a circuit with memory cells and s_0 an initial state for the memory cells, we write $C[s_0]$ for the circuit C with memory cells initially filled with s_0. Stateful circuits can also have external input and output wires. For instance, in an AES circuit the internal memory cells contain the secret key, the input wires a plaintext, and the output wires produce the corresponding ciphertext.

We define two distinct notions of security, for stateless and stateful circuits. While we view the stateful model as more interesting from an application point of view, the stateless model is somewhat cleaner and solutions for this model are used as the basis for solutions for the stateful model.

Privacy for stateful circuits. Let T be an efficiently computable *randomized* algorithm mapping a (stateful) circuit C along with an initial state s_0 to a (stateful) circuit C' along with an initial state s'_0. We say that T is a *t-private stateful transformer* if it satisfies soundness and privacy, defined as follows:

SOUNDNESS. The input-output functionality of C initialized with s_0 is indistinguishable from that of C' initialized with s'_0. This should hold for any sequence of invocations on an arbitrary sequence of inputs. In other words, $C[s_0]$ and $C'[s'_0]$ are indistinguishable to an *interactive* distinguisher.

PRIVACY. We require that C' be private against a t-*limited* interactive adversary. Specifically, the adversary is given access to C' initialized with s'_0 as its internal state. Then, the adversary may invoke C' multiple times, adaptively choosing the inputs based on the observed outputs. Prior to each invocation, the adversary may fix an arbitrary set of t internal wires to which it will gain access in that invocation. We stress that while this choice may be adaptive between invocations, i.e., may depend on the outputs and on wire values observed in previous invocations, the adversary is assumed to be too slow to move its probes while the values propagate through the circuit.[4] To define privacy against such a t-limited adversary, we require the existence of a *simulator* which can simulate the adversary's view using only a black-box access to C', i.e., without having access to any internal wires.[5]

Note that randomization is vital for stateful transformers, for otherwise it is impossible to hide the initial state from the adversary. However, apart from the (trusted) randomized initialization, the circuit C' may be deterministic.

We distinguish between three types of transformers: *perfect, statistical,* and *computational*, corresponding to the quality of indistinguishability in the soundness requirement and the type of emulation provided by the simulator. For the latter two types of security, we assume that T is also given a *security parameter* k in terms of which the indistinguishability is defined and the complexity of T is measured.

Privacy for stateless circuits. In contrast to the stateful case, where inputs and outputs are considered public and it is only the internal state that is hidden, privacy for stateless circuits should keep both inputs and outputs hidden in every invocation. To make this possible, we allow the use of a randomized *input encoder* I and an *output decoder* O, a pair of circuits whose internal wires cannot be probed by the adversary. Both I and O should be independent of the circuit C being transformed, and will typically require a small number of gates to compute. Thus, they may be thought of as being implemented by expensive tamper-resistant hardware components. A private stateless transformer can now be defined similarly to the stateful case.

Let T be an efficiently computable *deterministic* function mapping a stateless circuit C to a stateless circuit C', and let I, O be as above. We say that (T, I, O) is a t-*private stateless transformer* if it satisfies soundness and privacy, defined as follows:

SOUNDNESS. The input-output functionality of $O \circ C' \circ I$ (i.e., the iterated application of I, C', O in that order) is indistinguishable from that of C. Note that in the deterministic case this implies functional equivalence.

[4] Most of our constructions are in fact secure even against a fully adaptive adversary, that can also move its probes *within* an invocation, as long as the total number of probes in each invocation does not exceed t.

[5] In a case where C is randomized, the adversary's view should be simulated jointly with the circuit's outputs. This is necessary to capture information learned about the outputs.

PRIVACY. We require that the view of any t-limited adversary, which attacks $O \circ C' \circ I$ by probing at most t wires in C', can be simulated from scratch, i.e. without access to any wire in the circuit. As in the stateful case, the identity of the probed wires has to be chosen in advance by the adversary.

3 Perfect Privacy for Stateless Circuits

In this section we present our first construction for protecting privacy in stateless circuits. In the next section we will show how to use this to achieve protection for the more useful model of stateful circuits, where the contents of memory are to be protected.

Similarly to interactive protocols for secure multi-party computation (e.g., [6, 20]), our construction makes use of a simple secret-sharing scheme. The new twist in the circuit setting is that the atomic unit of information observable by the adversary is any *intermediate* computation rather than an entire party in the protocol setting. We achieve our result through a careful choice of intermediate computations, which allows us to obtain privacy without losing efficiency. The constants involved in the result we present here are quite small, and this construction may be of practical value. We now establish:

Theorem 1. *There exists a perfectly t-private stateless transformer (T, I, O) such that T maps any stateless circuit C of size n and depth d to a randomized stateless circuit of size $O(nt^2)$ and depth $O(d \log t)$.*

Proof. For simplicity, we focus on the case that C is deterministic. We start by describing the construction of the transformer (T, I, O). Let[6] $m = 2t$.

INPUT ENCODER I: Each binary input x is mapped to $m + 1$ binary values: First, m random binary values $r_1, \ldots, r_m$ are chosen using m random-bit gates. The encoding is then these m random values together with $r_{m+1} = x \oplus r_1 \oplus \cdots \oplus r_m$. The circuit I computes the encoding of each input bit independently in this way.

OUTPUT DECODER O: Corresponding to each output bit of C will be $m + 1$ bits $y_1, \ldots, y_{m+1}$ produced by $T(C)$. The associated output bit of C computed by O will be $y_1 \oplus \cdots \oplus y_{m+1}$.

CIRCUIT TRANSFORMER T: Assume without loss of generality that the circuit C consists of only NOT and AND gates. We will construct a transformed circuit C', maintaining the invariant that corresponding to each wire in C will be $m + 1$ wires in C' carrying an additive $m + 1$ out of $m + 1$ secret sharing of the value on that wire of C. The circuit C' is obtained by transforming the gates of C as follows.

For a NOT gate acting on a wire w, we merely take the $m + 1$ wires $w_1, \ldots, w_{m+1}$ associated with w in C', and put a NOT gate on w_1.

Consider an AND gate in C with inputs a, b and output c. In C', we will have corresponding wires $a_1, \ldots, a_{m+1}$ and $b_1, \ldots, b_{m+1}$. Recall that $a = \sum_i a_i \bmod 2$ and $b = \sum_i b_i \bmod 2$. Thus $c = (a \text{ AND } b) = \sum_{i,j} a_i b_j \bmod 2$. The difficulty is in computing shares of c by grouping together elements from the summation so that t intermediate values do not reveal any information to the adversary. We now describe our

[6] Note that there is a way to slightly modify this construction which requires $m = t$ instead of $m = 2t$. See below.

technique for doing so: In the transformation of this gate, we first compute intermediate values $z_{i,j}$ for $i \neq j$. For each $1 \leq i < j \leq m+1$, we introduce a random-bit gate producing a random bit $z_{i,j}$. Then we compute $z_{j,i} = (z_{i,j} \oplus a_i b_j) \oplus a_j b_i$. Note that individually each $z_{i,j}$ is distributed uniformly, but any pair $z_{i,j}$ and $z_{j,i}$ depend on a_i, a_j, b_i, and b_j. Now, we compute the output bits $c_1, \ldots, c_m$ in C' of this AND gate in C as

$$c_i = a_i b_i \oplus \bigoplus_{j \neq i} z_{i,j}.$$

In this way, each AND gate in C is expanded to a "gadget" of $O(m^2)$ gates in C', and the gadgets in C' are connected in the same way that the AND gates of C are connected. The resulting circuit, call it C', is the transformed version of C produced by T: *i.e.*, we define $T(C) = C'$, with C' as above. This completes the description of T.

Clearly, this construction preserves the functionality of the original circuit. To prove t-privacy, we must show how to simulate the view of the t-limited adversary without knowing the input values for C. The simulation will proceed by running the adversary, and providing it with answers to its t queries. We will show that the distribution of answers our simulation provides is *identical* to the distribution the adversary would obtain in a real attack on C'.

The simplest description of the simulator is just this: Answer all adversary queries based on the evaluation of the circuit C' when fed uniform and independent bits as input. In order to prove that this simulation works, we give a different description of the simulator. We first describe the simulator for a circuit C consisting of a single AND gate, and then extend the proof and simulation to the general case.

SIMULATION FOR A SINGLE GATE. Let C' be the transformed circuit, consisting of a single gadget, with input wires $\{a_i\}$ and $\{b_i\}$, and outputs $\{c_i\}$. Recall that in a true evaluation of C', the a_i's and b_i's are additive secret shares with the property that any m shares from the a_i's are distributed as uniform independent random bits, and similarly for the b_i's. We will argue that a perfect simulation of the adversary's query responses is possible based on knowledge of m or fewer shares from the a_i's and the b_i's. Since such a collection of shares is distributed uniformly, this will establish our result.

Suppose an adversary corrupts wires $w_1, \ldots, w_t$ in C'. We will define a set $I \subset [m+1]$ of indices such that the joint distribution of values assigned to the wires w_h (for any specific inputs a and b to the original circuit C) can be perfectly and efficiently simulated given the values of $a|_I := (a_i)_{i \in I}$ and $b|_I$. As mentioned above, the values $a|_I$ and $b|_I$, in turn, can be perfectly simulated by picking them uniformly and independently at random, as long as $|I| \leq m$. Hence, it suffices to describe a procedure for constructing the set I and simulating the values of the t corrupted wires w_h given $a|_I$ and $b|_I$. We describe such a procedure now.

1. Initially, I is empty and all w_h are unassigned.
2. For every wire w_h of the form $a_i, b_i, a_i b_i, z_{i,j}$ (for any $i \neq j$), or a sum of values of the above form (including c_i as a special case), add i to I. Note that this covers

all wires in C' except for wires corresponding to $a_i b_j$ or $z_{i,j} \oplus a_i b_j$ for some $i \neq j$. For such wires, add both i and j to I.[7]

3. Now that the set I has been determined—and note that since there are at most t wires w_h, the cardinality of I can be at most $m = 2t$—we show how to complete a perfect simulation of the values on w_h using only the values $a|_I$ and $b|_I$. Assign values to the $z_{i,j}$ as follows:
   - If $i \notin I$ (regardless of j), then $z_{i,j}$ does not enter into the computation for any w_h. Thus, its value can be left unassigned.
   - If $i \in I$, but $j \notin I$, then $z_{i,j}$ is assigned a random independent value. Analysis: Note that if $i < j$ this is what would have happened in the real circuit C'. If $i > j$, however, we are making use of the fact that by construction, $z_{j,i}$ will never be used in the computation of any w_h. Hence we can treat $z_{i,j}$ as a uniformly random and independent value.
   - If both $i \in I$ and $j \in I$, then we have access to a_i, a_j, b_i, and b_j. Thus, we compute $z_{i,j}$ and $z_{j,i}$ exactly as they would have been computed in the actual circuit C'; *i.e.*, one of them (say $z_{j,i}$) is assigned a random value and the other $z_{i,j}$ is assigned $z_{j,i} \oplus a_i b_j \oplus a_j b_i$.
4. For every wire w_h of the form $a_i, b_i, a_i b_i, z_{i,j}$ (for any $i \neq j$), or a sum of values of the above form (including c_i as a special case), we know that $i \in I$, and all the needed values of $z_{i,j}$ have already been assigned in a perfect simulation. Thus, w_h can be computed in a perfect simulation.
5. The only types of wires remaining are $w_h = a_i b_j$ or $w_h = z_{i,j} \oplus a_i b_j$. But by Step 2, both $i, j \in I$, and by Step 3, $z_{i,j}$ has been assigned, thus the value of w_h can be simulated perfectly.
6. Note that all c_i values for $i \in I$ can be simulated perfectly by the argument above. This completes the simulation and the argument of correctness.

SIMULATION FOR A GENERAL CIRCUIT. The simulation for a general transformed circuit C' proceeds very similarly to the above. First, examining each gadget g in C', we compute the set I. Note that since a total of t wires can be corrupted throughout the circuit C', the size of the set I will still be bounded by m. Next we perform the simulation as above, working our way from the inputs of C' to the outputs. Note that by the observation in Step 6 above, we maintain the invariant that for each gadget g, the shares of the inputs to g with indices belonging to I are perfectly simulated. Thus, inductively, the values of all corrupted wires in C' are simulated perfectly.

RE-RANDOMIZED OUTPUTS. We observe that as long as every output of C' has passed through one AND gadget (if this is not the case, we can artificially AND an output bit with itself), then for each original output bit, the encoded outputs are m-wise independent even given the entire encoding of the inputs. This can be used to prove that the

[7] We note that by changing the construction slightly, namely by computing $(a_i + r)b_j$ and rb_j where r is a fresh random value, we could have avoided increasing I by 2 indices rather than just 1 for any single wire observed by the adversary. This would have allowed us to choose $m = t$ rather than $m = 2t$ as we have chosen now.

construction is in fact secure against a stronger type of adversary who may observe at most t' wires in *each gadget*, where $t' = \Omega(t)$.[8]

IMPROVEMENT IN RANDOMNESS USE. In the conference proceedings version of this paper, it was incorrectly[9] claimed that the same randomness (*i.e.*, the choices of $z_{i,j}$ for $i < j$) could be used in all the gadgets of the above construction, reducing the number of random bits to $O(t^2)$. Instead, the randomness complexity can be reduced to $\mathrm{poly}(t, \log |C|)$ by following the approach of [11] for reducing the randomness complexity in MPC protocols via limited independence. Concretely, this is done by first modifying the above construction of C' so that the values of each t wires in C' depend on at most $\ell = \mathrm{poly}(t)$ randomness gates. Then, the $O(|C'|)$ randomness gates can be emulated by a "locally random" low entropy source obtained by taking the exclusive-or of $t + 1$ independent outputs of an ℓ-wise independent pseudorandom generator. We leave open the question of obtaining tighter bounds on the randomness complexity of private circuits.

UNPROTECTED INPUTS AND OUTPUTS. We have described the construction above for protecting *all* inputs and outputs. It is easy to modify the construction so that certain inputs and outputs are unencoded, and may be observed by both the adversary and the simulator. This is useful in the stateful model, discussed next.

4 Perfect Privacy for Stateful Circuits

In this section we show how to achieve privacy in the stateful model, as defined in Section 2. This model is perhaps much more natural and realistic than the stateless model we considered in the previous section; however, as we show below, achieving privacy in this model is easy once privacy has been achieved in the stateless model.

Our goal is to transform a stateful circuit C into a t-private stateful circuit C' by using a privacy transformer for the stateless case. We now describe the construction. Recall that a stateless privacy transformer must encode the input in some way; we assume that the output is encoded using the same encoding. We also assume that the stateless transformer enjoys the *re-randomized outputs* property, namely that the output encoding for each original output bit is t-wise independent even given all encodings of input bits. Let us refer to the encoding of the stateless transformer as $E_t(x)$, where t is the privacy threshold of the stateless transformer, and x is the input being transformed. We represent each memory cell in C using the same representation. Relying on our stateless transformer as a building block, a stateful transformer $T = (T_C, T_s)$ can proceed as follows. The memory x of C is stored in C' in encoded form $E_{2t}(x)$.[10] C' will work by considering the transformed memory $E_{2t}(x)$ as an input to the original circuit C, which is transformed using the stateless $2t$-privacy transformation. We also modify

[8] The ratio between t and t' depends on the maximal fan-out of C (which we fixed to 3 by default). This dependence can be eliminated by slightly modifying the construction.

[9] We thank Jean-Sebastien Coron for pointing out this error.

[10] Note that the use of $2t$ as a threshold is critical, since the adversary could observe t bits of the inputs to the memory at the end of one clock cycle, and then another t bits of the outputs of the memory in the next clock cycle; in this way the adversary would observe $2t$ bits of the state of the memory.

C so that the next state of the memory is always an output. Then, these encoded outputs are fed back into memory for the next clock cycle. The regular inputs and outputs of C are unprotected, and need not be encoded. This completes the description of T_C.

A simulation argument proving the correctness of this transformer proceeds very similarly to the stateless case analyzed above. In fact, a sequence of invocations of a stateful circuit may be unwound into a larger stateless circuit with an equivalent functionality. Here, the initial state is viewed as a hidden input, and the final state as a hidden output. Thus, the security proof for the stateful case essentially reduces to that of the stateless case. [11] In the "unwound" circuit, the adversary can corrupt up to t wires in each of the concatenated circuits Q produced by the stateless transformation. The simulation proof proceeds exactly as before; the additional corruptions do not obstruct the proof because of the re-randomization property: the outputs of one Q are t-wise independent conditioned on all the values of the inputs to Q; thus in order to provide a full joint simulation of the entire unwound circuit, we need only be able to recover a bounded set of inputs from each component Q. To summarize, we have shown:

Theorem 2. *There exists a perfectly t-private stateful circuit transformer which maps any stateful circuit C of size n and depth d to a randomized stateful circuit of size $O(nt^2)$ and depth $O(d \log t)$.*

5 Statistically Private Transformers

In this section we obtain statistically-private transformers which improve the previous constructions when the privacy threshold t is large. For the description and analysis of these transformers, it is convenient to rely on the following notion of *average-case* security.

Definition 1. *A circuit transformer $T = T(C, k)$ is said to be (statistically) p-*private in the average case *if $C' = T(C, k)$ is statistically private against an adversary which corrupts each wire in C' with independent probability p. That is, the joint distribution of the random set of corrupted wires and the values observed by the adversary can be simulated up to a $k^{-\omega(1)}$ statistical distance.*

We note that a p-adversary as above is roughly the same as an adversary that corrupts a uniformly random subset of $p|C'|$ wires in C'. Intuitively, average-case privacy in this sense should be easier to realize than the standard (worst-case) notion of privacy. Indeed, the circuit transformer from the previous section with k additive shares (i.e., $m = k$) is perfectly private with respect to any adversary corrupting $k/4$ wires in each gadget. It follows that the view of an adversary corrupting each wire with probability, say, $1/(10k)$ can be perfectly simulated except with negligible failure probability. Thus, we have:

Lemma 1. *There exists a circuit transformer $T(C, k)$ producing a circuit C' of size $O(k^2|C|)$, such that T is $\Omega(1/k)$-private in the average case.*

[11] One technical difference between the two models is that the inputs and outputs in the stateful model are known to the adversary. However, these values are given to the simulator "for free" and can thus be easily incorporated into the simulation.

In contrast, achieving *worst-case* privacy against an adversary corrupting $\Omega(|C'|/k)$ of the wires in C' appears to be much harder; in particular, the constructions from the previous section are very far from achieving this when $|C'| \gg k$. The key idea underlying the asymptotic improvements in this section is the following *reduction* from worst-case privacy to average-case privacy.

We start with an efficient circuit transformer T guaranteeing p-privacy in the average case. We then transform its output $C' = T(C, k)$ into a larger circuit $\tilde{C}'$, which in a sense may be viewed as a "sparse" implementation of C'. The circuit $\tilde{C}'$ will carry out the same computation performed by C' in essentially the same way, but will effectively utilize only a small random subset of its wires; all remaining wires of $\tilde{C}'$ will be independent of the inputs and thus rendered useless to the adversary. We stress that the subset of useful wires in $\tilde{C}'$ will only be determined during the invocation of $\tilde{C}'$ and will therefore be independent of the set of corrupted wires. Hence, for an appropriate choice of parameters, the (worst-case) t-privacy of $\tilde{C}'$ will reduce to the average-case p-privacy of C'.

We will describe two distinct instantiations of the above approach. The first is somewhat simpler, but incurs an $\tilde{O}(t) \cdot k^{O(1)}$ multiplicative blowup to the circuit size (see Remark 1). When $t \gg k$, this already provides an asymptotic improvement over the previous solutions, which incur an $O(t^2)$ overhead. In the second construction, which is only sketched in this abstract, we manage to avoid the dependence on t by amortizing it over multiple gates.

Both instantiations make use of *sorting networks* as a building block. A sorting network is a layered circuit from ℓ integer-valued input wires to ℓ integer-valued output wires, which outputs its input sequence in a sorted order. The internal gates in a sorting network are of a very special type: each such gate, called a comparator, has two inputs and two outputs and returns its pair of inputs in a sorted order. The celebrated AKS network [1] achieves the optimal parameters of $O(\ell \log \ell)$ size and $O(\log \ell)$ depth. However, in terms of practical efficiency it is preferable to use simpler sorting networks, such as Batcher's [5], whose slightly inferior asymptotic complexity ($O(\ell \log^2 \ell)$ size and $O(\log^2 \ell)$ depth) hides much smaller constants.

A gate-by-gate approach. Our initial construction transforms the circuit $C' = T(C, k)$ to a circuit $\tilde{C}'$ as follows. With each wire i of C' there are ℓ wires of $\tilde{C}'$ labeled $(i, 1), \ldots, (i, \ell)$, where the parameter ℓ will be determined later. It is convenient to assume that these wires can carry ternary values from the set $\{0, 1, \$\}$. The execution of $\tilde{C}'$ will maintain the following invariant relative to an execution of C': if wire i of C' carries a value $v_i \in \{0, 1\}$, then the wires $(i, 1), \ldots, (i, \ell)$ will carry the value v_i in a random position (independently of other ℓ-tuples) and the value \$ in the remaining $\ell - 1$ positions. This property can be easily initialized at the inputs level by appropriately defining the input encoder of $\tilde{C}'$. Similarly, the output decoder of $\tilde{C}'$ can be easily obtained from that of C'.

It remains to describe how to emulate a gate of C' while maintaining the above invariant. Suppose that $v_i = v_{i_1} * v_{i_2}$, i.e., the value of wire i in C' is obtained by applying some commutative boolean operation '*' to the values of wires i_1, i_2. We replace this gate in C' with a 2ℓ-input, ℓ-output gadget in $\tilde{C}'$, which first routes the values v_{i_1}, v_{i_2} to two random but *adjacent* positions, and then combines them to form

the output. One should be careful, however, to implement this computation so that even by observing intermediate values, the adversary will not be able to learn more values v_i than it is entitled to. Such an implementation for a gadget is given below.

PREPROCESSING. Let $r, r_1, \ldots, r_\ell$ be $\ell+1$ uniformly random and independent integers from the range $[0, 2^k]$. For each $1 \leq j \leq \ell$, use the values $v_{i_1,j}, v_{i_2,j}$ (of wires (i_1, j) and (i_2, j)) to form a pair $(\mathsf{key}_j, \mathsf{val}_j)$ such that: (1) key_j is set to r_j if $v_{i_1,j} = v_{i_2,j} = \$$ and to r otherwise; (2) val_j is set to \$ if both $v_{i_1,j}, v_{i_2,j}$ are \$, to a bit value b if one of $v_{i_1,j}, v_{i_2,j}$ is b and the other is \$, and to $b_1 * b_2$ if $v_{i_1,j} = b_1$ and $v_{i_2,j} = b_2$.

SORTING. A sorting network is applied to the above ℓ-tuple of pairs using key as the sorting key. Let $(u_1, \ldots, u_\ell)$ denote the ℓ-tuple of symbols val_j sorted according to the keys key_j.

POSTPROCESSING. The jth output $v_{i,j}$ is obtained by looking at u_j, u_{j+1}, u_{j+2}: if $u_j, u_{j+1} \neq \$$ then $v_{i,j} = u_j * u_{j+1}$, if $u_j = u_{j+2} = \$$ and $u_{j+1} \neq \$$ then $v_{i,j} = u_{j+1}$, and otherwise $v_{i,j} = \$$.

Note that each such gadget can be implemented by a circuit of size $\tilde{O}(\ell k)$ and depth $\mathrm{poly}(\log \ell + \log k)$.

To complete the description of $\tilde{C}'$, we describe a (simpler) gadget replacing each random bit gate z in C'. As in the gate gadget, the random bit gadget has ℓ inputs and ℓ outputs. The jth input is a random bit z_j. A random selector $r \in [\ell]$ is used for determining which z_j will appear in the output. Specifically, the jth output is set to z_j if $r = j$ and to \$ otherwise. The cost of implementing this gadget is smaller than that of the gate gadget. Hence, the entire circuit $\tilde{C}'$ has size $\tilde{O}(\ell k n)$ and depth comparable to that of C' (up to polylog factors).

We now establish the relation between the worst-case privacy of $\tilde{C}'$ and the average-case privacy of C'.

Lemma 2. *Suppose that C' is p-private in the average case. Then the circuit $\tilde{C}'$, constructed with $\ell = O(t/p^4)$, is statistically t-private in the worst case.*

Proof sketch: It is convenient to make the adversary slightly stronger by assuming that it may actually probe t *logical*, rather than boolean, wires (i.e., each such wire may contain an integer, a ternary symbol, or a bit). For each compromised wire of $\tilde{C}'$, the adversary can either see some random integer r_i, a \$ symbol, or an actual value v_i of the ith wire of C'.[12] In the latter case, we say that v_i has been *observed*. Let S denote the set of indices i such that v_i has been observed. Note that S is a random variable, where the probability is over the execution of $\tilde{C}'$.

We will argue that for any fixed index set S_0, and for ℓ chosen as in the lemma, we have $\Pr[S_0 \subseteq S] \leq p^{|S_0|}$. Thus, an adversary attacking any fixed set of t wires in $\tilde{C}'$ is not better off than an adversary corrupting each wire of C' independently with probability p.

To make this argument, we pick a subset $S_1 \subseteq S_0$ such that: (1) $|S_1| \geq |S_0|/4$; (2) each value in S_1 is observed with probability (at most) p^4; and (3) the events of

[12] In fact, depending on the exact implementation there may be wires of $\tilde{C}'$ containing information on two values v_i. We ignore this technicality as it does not change the analysis in any substantial way.

observing different values in S_1 are independent. This will make the probability of observing all values in S_1 at most $(p^4)^{|S_0|/4} = p^{|S_0|}$ as required.

We pick S_1 to be a maximal matching in the subgraph of C' induced by the wires in S_0. Since the degree of each vertex in this graph is at most 4, we have $|S_1| \geq |S_0|/4$. It remains to show that S_1 satisfies properties (2) and (3) above.

To prove (2) it suffices to show that for any *fixed* wire of $\tilde{C}'$, the probability that this wire contains a useful value (i.e., contributes to S) is $O(1/\ell)$. (Property (2) would then follow, since by taking the union over all t compromised wires of $\tilde{C}'$, the probability of observing a value of C' is $O(t/\ell) \leq p^4$.) This clearly holds for input wires, by definition of the input encoder, and is maintained through all internal wires in the circuit by a symmetry argument. (In the case of gate gadgets, the argument relies on the fact that each val entry inside a sorting network contains one of the gadget's inputs, rather than some arbitrary combination of these inputs; due to the randomness of the sorting keys, the randomness of the positions of the useful entries is maintained).

It remains to argue that the independence property (3) holds. This follows from the fact that no two wires in S_1 are adjacent to a common gate in C' and from the fact that each gadget in $\tilde{C}'$ uses fresh randomness to shuffle its entries. □

Combining Lemma 2 with Lemma 1, we have:

Theorem 3. *There exists a statistically t-private stateless transformer $(\tilde{T}, \tilde{I}, \tilde{O})$, such that $\tilde{T}(C, k)$ transforms a circuit C of size n to a circuit $\tilde{C}'$ of size $n \cdot \tilde{O}(t) \cdot k^{O(1)}$ (where k is a statistical security parameter). The depth of $\tilde{C}'$ is the same as that of C, up to polylog factors.*

Remark 1. Throughout this section, we view $k^{O(1)}$ and $\mathrm{polylog}(t)$ as being small in comparison to t, and therefore do not attempt to optimize the exact dependence on such factors. We note that all occurrences of $k^{O(1)}$ in the complexity of our constructions (e.g., in Theorem 3) can be replaced by $\mathrm{polylog}(k)$ while still satisfying our asymptotic notion of statistical security.

The above construction (and in particular the analysis of Lemma 2) crucially relies on the assumption that the adversary chooses in advance which t wires to corrupt, independently of the values it observes while invoking $\tilde{C}'$. However, for using this construction in the stateful case we need a somewhat stronger security guarantee. Indeed, since the adversary is allowed to move its t probes before each invocation based on the values it observes in previous invocations, it may gradually build more and more knowledge about the locations of useful values in $\tilde{C}'$. To get around this problem and guarantee sufficient independence between different invocations, it suffices to re-randomize each ℓ-tuple of wires representing the new content of a memory cell by applying a perfectly t-private computation of a random cyclic shift. Using our basic construction, this can be done using $\tilde{O}(\ell t^2)$ additional gates. When the size of the circuit is much larger than the number of states and t, the amortized cost per gate of this randomization step is small.

The above discussion is captured by the following theorem.

Theorem 4. *There exists a statistically t-private stateful transformer $\tilde{T}$, such that $\tilde{T}(C, k)$ maps a circuit C of size n with s memory cells to a circuit $\tilde{C}'$ of size $\tilde{O}(nt + st^3) \cdot k^{O(1)}$. The depth of $\tilde{C}'$ is the same as that of C, up to polylog factors.*

Amortizing the cost over multiple gates. The previous construction is redundant in the sense that it uses a separate gadget, of size $\Omega(t)$, for each gate in the circuit. We briefly sketch a modification of this construction which amortizes the additional cost over multiple gates, effectively eliminating the dependence on t. For the description and analysis of this construction, it is convenient to assume that the circuit is *layered* (see Section 2), and use the following modified notion of average-case security for the layered case.

Definition 2. *Let $T = T(C, k)$ be a* layered *circuit transformer producing a layered circuit C' of width w. Then, T is said to be (statistically) p-secure in the average case if $C' = T(C, k)$ is statistically secure against an adversary which corrupts a* random *subset of pw wires in each layer of C'.*

As before, we will use an average-case p-secure C' to build a worst-case t-secure $\tilde{C}'$. However, instead of representing each wire of C' by an ℓ-tuple of wires, we will now represent an entire *layer* of C' by a corresponding layer in $\tilde{C}'$ consisting of $\ell = \max(w, t/p)$ wires. These wires will contain a random permutation of the w values of C' in ℓ random positions and the symbol \$ in all other positions. Note that typically $w > t/p$, in which case there are no useless \$ entries in this list. However, the above choice of ℓ guarantees that by looking at any fixed set of t positions in the list the adversary will observe a random subset containing at most a p-fraction of the values.

Each value of C' is represented by a pair containing its index and its value. This representation naturally defines the input encoder and output decoder. It remains to show how the above representation can be maintained between subsequent layers. As before, we also need to ensure that each *intermediate* level in the computation of $\tilde{C}'$ contains a random permutation of the useful values, where the randomness of these permutations is independent for levels that are sufficiently far apart. To achieve this, we use an ℓ-input, ℓ-output gadget whose inputs represent the jth level wires in C' and whose outputs represent the $(j+1)$th level wires. The high-level idea is as before, except that we now need to *jointly* route $w/2$ pairs of wires to random adjacent positions, and then combine each pair in the right way.

Using this approach, we can obtain the following theorem:

Theorem 5. *There exists a statistically t-secure stateless transformer $(\tilde{T}, \tilde{I}, \tilde{O})$, such that $\tilde{T}(C, k)$ transforms a layered circuit C of width w and depth d to a circuit $\tilde{C}'$ of width $\tilde{O}(\max w, t) \cdot k^{O(1)}$ and depth $d \cdot \mathrm{polylog}(w, t, k)$.*

An analogous theorem for the stateful model can be derived similarly to Theorem 4.

6 A PRG secure against probing attacks

Next, we will show how to build a deterministic, stateful circuit that will produce pseudorandom output and remain secure even in the presence of probing attacks. In essence, we will be building a PRG that resists probing attacks. Because the resulting circuit is deterministic, this is helpful if true randomness is expensive.

The basic construction is as follows. Let $G : \{0,1\}^\sigma \to \{0,1\}^{(2t+1)\sigma+\lambda}$ be a PRG. We will build a deterministic stateful circuit $C'[s'_0]$ with $(2t+1)\sigma$ bits of internal memory, no inputs, and λ bits of output. $C'[s'_0]$ will be understood as a secure translation of

the 0-input λ-output stateless randomized circuit C whose outputs are each fed by a different random-bit gate. The initial random seed s'_0 will be chosen uniformly at random, and the behavior of the circuit $C'[s'_0]$ on any one invocation is defined as follows:

1. Let $s = (s^1, \ldots, s^{2t+1})$ denote the current state of the memory cells.
2. Set $u := G(s^1) \oplus \cdots \oplus G(s^{2t+1})$. Define s', y by parsing u as $u = (s', y)$.
3. Replace the current state of the memory cells with s', and output y.

It is crucial that the circuit $C'[s'_0]$ contain $2t + 1$ disjoint copies of G, executing in parallel and sharing no wires or gates. Our construction is related to the method for distributed pseudorandomness generation with proactive security from [10]. For lack of space, the proof of Theorem 6 is omitted here.

Theorem 6. *If G is a secure PRG, then the stateful deterministic circuit $C'[s'_0]$ defined above is a computationally t-private transformation of the circuit C defined above.*

Application: eliminating randomness gates. One application for our PRG construction is in eliminating randomness for the stateful circuit transformer of Section 4. Our basic solution for the stateless model, as described earlier, relies on the use of random bit gates within the transformed circuit $T(C)$. An appealing consequence of our probe-resistant PRG is that it allows to dispense with on-line randomness generation: an initial random seed can be coded into the initial state by T and (deterministically) "refreshed" at each invocation of the circuit.

Suppose our transformed circuit $T(C)$ uses λ random-bit gates. Let C_r be a stateless randomized circuit consisting of λ independent random-bit gates, each connected to a different output of C_r. If $C'_r[s'_0]$ is any deterministic stateful circuit that is a secure translation of C_r, then we can replace the random-bit gates of $T(C)$ with the probe-resistant PRG $C'_r[s'_0]$. For instance, the deterministic, stateful PRG of Theorem 6 will do the job nicely. In this way, we can derandomize C' and obtain an efficient stateful, deterministic circuit that is computationally t-private and not too much larger than the original circuit.

7 Concluding Remarks

We have developed theoretical foundations for the problem of securing hardware against side channel attacks, and initiated a systematic study of this problem within our framework. In this initial study we restricted our attention to side channels that can be modelled by *probing* attacks, i.e. whose information leakage depends on a limited number of physical wires. It would be interesting to extend our framework and results to a wider class of realistic attacks. A step in this direction is taken by Micali and Reyzin [29], who put forward a very general model for side channel attacks.

Another natural extension of the problem studied in this work is to allowing additional protection against *fault* attacks [7, 27]. Similarly to our problem, solutions to this more general problem can be based on existing protocols from the MPC literature. However, even the most efficient of these (e.g., [23]) are still quite inefficient to implement on hardware. Obtaining better solutions in this setting, possibly under relaxed notions of security, remains an interesting challenge.

Acknowledgements. We wish to thank Jean-Sebastien Coron, David Molnar and anonymous referees for helpful comments. We also thank an anonymous referee for suggesting the use of the brain metaphor. Amit Sahai acknowledges support from an Alfred P. Sloan Foundation Research Fellowship.

References

1. M. Ajtai , J. Komlos , E. Szemeredi. An $O(n \log n)$ sorting network. In *Proceedings of the 15th STOC*, pp. 1-9, 1983.
2. R. Anderson, M. Kuhn, "Tamper Resistance—A Cautionary Note," *USENIX E-Commerce Workshop*, USENIX Press, 1996, pp.1–11.
3. R. Anderson, M. Kuhn, "Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations," *Proc. 2nd Workshop on Information Hiding*, Springer, 1998.
4. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. CRYPTO 2001, 2001.
5. K. Batcher. Sorting Networks and their Applications. In *Proc. AFiPS Spring Joint Conference*, Vol. 32, 1988, pp. 307-314.
6. M. Ben-Or, S. Goldwasser, and A. Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of 20th STOC*, 1988.
7. D. Boneh, R.A. Demillo, R.J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," *EUROCRYPT'97*, Springer-Verlag, 1997, pp.37–51.
8. R. Canetti. Security and composition of multiparty cryptographic protocols. In *J. of Cryptology*, 13(1), 2000.
9. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz and A. Sahai. Exposure-Resilient Functions and All-or-Nothing Transforms. In *EUROCRYPT 2000*, pages 453-469.
10. R. Canetti and A. Herzberg. Maintaining Security in the Presence of Transient Faults. In *CRYPTO 1994*, pages 425-438.
11. R. Canetti, E. Kushilevitz, R. Ostrovsky, and A. Rosén. Randomness versus Fault-Tolerance. *J. Cryptology* 13(1), 2000.
12. S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks," *CRYPTO'99*, Springer-Verlag, 1999, pp.398–412.
13. D. Chaum, C. Crepeau, and I. Damgård. Multiparty unconditional secure protocols. In *Proc. of 20th STOC*, 1988.
14. R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Proc. of EUROCRYPT '00*.
15. J.-S. Coron, L. Goubin, "On Boolean and Arithmetic Masking against Differential Power Analysis," *CHES'00*, Springer-Verlag, pp.231–237.
16. J. Daemen, V. Rijmen, "Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals," *AES'99*, Mar. 1999.
17. K. Gandolfi, C. Mourtel, F. Olivier, "Electromagnetic Analysis: Concrete Results," *CHES'01*, LNCS 2162, Springer-Verlag, 2001.
18. R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proc. of 17th PODC*, 1998.
19. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *JACM*, 33(4):792–807, October 1986.
20. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game (extended abstract). In *Proc. of 19th STOC*, 1987.
21. O. Goldreich and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *JACM* 43(3): 431-473, 1996.
22. L. Goubin, J. Patarin, "DES and Differential Power Analysis—The Duplication Method," *CHES'99*, Springer-Verlag, 1999, pp.158–172.

23. M. Hirt and U. Maurer. Robustness for free in unconditional multi-party computation. In *Proc. of CRYPTO '01*.
24. D. Kahn, *The Codebreakers*, The MacMillan Company, 1967.
25. J. Kelsey, B. Schneier, D. Wagner, "Side Channel Cryptanalysis of Product Ciphers," *ESORICS'98*, LNCS 1485, Springer-Verlag, 1998.
26. P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," *CRYPTO'96*, Springer-Verlag, 1996, pp.104–113.
27. P. Kocher, J. Jaffe, B. Jun, "Differential Power Analysis," *CRYPTO'99*, Springer-Verlag, 1999, pp.388–397.
28. T.S. Messerges, "Securing the AES Finalists Against Power Analysis Attacks," *FSE'00*, Springer-Verlag, 2000.
29. S. Micali and L. Reyzin. A model for physically observable cryptography. Manuscript, 2003.
30. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proc. of 10th PODC*, 1991.
31. D. Page, "Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel," Tech. report CSTR-02-003, Computer Science Dept., Univ. of Bristol, June 2002.
32. B. Pfitzmann, M. Schunter and M. Waidner, "Secure Reactive Systems", IBM Technical report RZ 3206 (93252), May 2000.
33. J.-J. Quisquater, D. Samyde, "Eddy current for Magnetic Analysis with Active Sensor," *Esmart 2002*, Sept. 2002.
34. J.-J. Quisquater, D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards," *Esmart 2001*, LNCS 2140, Springer-Verlag, 2001.
35. J.R. Rao, P. Rohatgi, "EMpowering Side-Channel Attacks," IACR ePrint 2001/037.
36. US Air Force, *Air Force Systems Security Memorandum 7011—Emission Security Countermeasures Review*, May 1, 1998.
37. W. van Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk," *Computers & Security*, v.4, 1985, pp.269–286.
38. D. Wright, *Spycatcher*, Viking Penguin Inc., 1987.
39. A. C. Yao. How to generate and exchange secrets. In *Proc. of 27th FOCS*, 1986.

A Relation with Secure Multi-Party Computation

The problem studied in this paper is closely related to the problem of secure multi-party computation (MPC), introduced and first studied in [39, 20, 6, 13] and extensively studied thereafter. We begin by explaining the relation between the problems, and then highlight some important differences.

THE MPC MODEL. In the most basic setting for secure MPC, n parties are connected by a complete network of point-to-point channels. Initially, each party holds a local input and an independent random input. The parties' goal is to evaluate some publicly known function f of their inputs while hiding their inputs from each other. To this end, they interact via a prescribed protocol. The protocol proceeds in round, where at each round each party may send a message to every other party based on its input, its random input, and messages received in previous rounds. The protocol terminates at some predetermined round, in which all parties should output the correct value of f. A protocol as above is said to be *t-private* if for any set T of at most t parties, the entire view of T (consisting of their inputs, random inputs, and received messages) reveals no more information about the other parties' inputs than what follows from their own inputs and the value of f. Note that the latter information captures what must *inevitably* be learned. To better correspond to our circuit model, it is convenient to consider a

slightly modified MPC model in which each of the n inputs is initially secret-shared among the parties (say, using n out of n additive sharing), and the output produced by the protocol is also secret-shared in a similar fashion. This allows to realize a stronger and simpler privacy requirement: every collusion of t players learns nothing from their interaction with the remaining players.

RELATION TO PRIVATE CIRCUITS. To illustrate the relation between the MPC model and our circuit model, we focus on the stateless case and ignore some unimportant technicalities. First, we show that any t-private protocol corresponds to some t-private circuit computing the same function. Consider the following "hardware implementation" of an n-party protocol as above. In each round of interaction, each player's local computation is implemented by a separate sub-circuit. When the players interact, each message bit is translated into a wire connecting the corresponding sub-circuits. Note that the t-privacy of the protocol guarantees t-privacy also in the circuit model. Indeed, if an adversary can violate the circuit's privacy by probing t wires, then it could have also violated the protocol's privacy by corrupting some t players who "own" these wires.[13] The converse relation also holds. Suppose that we are given a t-private circuit computing the function f where the fan-in of each gate is at most 2. We use the circuit to define a protocol, in which each gate is owned by a distinct player. The circuit is evaluated by the players in a bottom-up fashion, starting with the encoded inputs and ending with an encoded output, where for each wire a message is sent from its source player to its destination players, and for each gate a local computation is performed by the corresponding player. It is not hard to see that if the circuit is $2t$-private, then the corresponding protocol is t-private. Indeed, a protocol-adversary corrupting t players learns content of at most $2t$ wires in the circuit.

In light of the above, one might expect to obtain the best solutions to our problem via efficient hardware implementations of state-of-the-art protocols from the MPC literature. However, this is not really the case. For instance, the BGW protocol [6] (as well as subsequent optimizations [18, 14]) requires each player to evaluate a degree-t polynomial on $\Theta(t)$ points for each gate of the circuit being evaluated. Consequently, the stateless circuit transformer that can be derived from this protocol is significantly less efficient than our transformer. This state of affairs stems from some major differences in the underlying optimization goals. First, the MPC literature puts much emphasis on tolerating a constant fraction of corrupted players, whereas in our setting the number of corruptions is viewed as being independent of the number of "players" (in particular, we are willing to settle for tolerating a miniscule fraction of corruptions). Second, the MPC setting typically views the *communication complexity* and the *round complexity* as the most important resources to optimize, placing the time complexity only as a third-order optimization goal. In contrast, the main optimization criterion in our case is the *size* of a circuit, which roughly (but not exactly) corresponds to the time complexity of the underlying protocol. Finally, our main (stateful) model is quite nonstandard from the MPC point of view, as it involves extra ingredients such as a one-time trusted precomputation (via the circuit transformer), a mobile adversary (as in [30, 10]), and on-line inputs and outputs (as in [32, 8]). To conclude, the problem we are posing is quite different from that of implementing standard MPC protocols at the hardware level.

[13] Note that a wire corresponding to a message bit is owned by more than one player.