Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases

Sebastian Faust[1] ⋆, Tal Rabin[2], Leonid Reyzin[3]⋆⋆,
Eran Tromer[4]⋆⋆⋆ and Vinod Vaikuntanathan[2]

[1] K.U. Leuven ESAT-COSIC/IBBT
[2] IBM Research
[3] Boston University
[4] MIT

Abstract. Physical computational devices leak side-channel information that may, and often does, reveal secret internal states. We present a *general* transformation that compiles any circuit into a new, functionally equivalent circuit which is resilient against well-defined classes of leakage. Our construction requires a *small*, *stateless* and *computation-independent* leak-proof component that draws random elements from a fixed distribution. In essence, we reduce the problem of shielding arbitrarily complex circuits to the problem of shielding a single, simple component.

Our approach is based on modeling the adversary as a powerful observer that inspects the device via a limited measurement apparatus. We allow the apparatus to access all the bits of the computation (except those inside the leak-proof component) and the amount of leaked information to grow unbounded over time. However, we assume that the apparatus is limited either in its computational ability (namely, it lacks the ability to decode certain linear encodings and outputs a limited number of bits per iteration), or its precision (each observed bit is flipped with some probability). While our results apply in general to such leakage classes, in particular, we obtain security against:

- *Constant depth circuits leakage*, where the measurement apparatus can be implemented by an AC^0 circuit (namely, a constant depth circuit composed of NOT gates and unbounded fan-in AND and OR gates), or an $\mathsf{ACC}^0[p]$ circuit (which is the same as AC^0, except that it also uses $\mathtt{MOD}_p$ gates) which outputs a limited number of bits.
- *Noisy leakage*, where the measurement apparatus reveals all the bits of the state of the circuit, perturbed by independent binomial noise. Namely, each bit of the computation is perturbed with probability p, and remains unchanged with probability $1 - p$.

⋆ Supported in part by Microsoft Research through its PhD Scholarship Programme, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and FWO grant G.0225.07.

⋆⋆ Supported in part by NSF grants CNS-0831281 and CNS-0546614.

⋆⋆⋆ Supported by NSF CyberTrust grant CNS-0808907 and AFRL grant FA8750-08-1-0088.

1 Introduction

The best of cryptographic algorithms are insecure when their implementations inadvertently reveal secrets to an eavesdropping adversary. Even when the software is flawless, practical computational devices leak information via numerous side channels, including electromagnetic radiation (visible and otherwise) [30,23], timing [7], power consumption [22], acoustic emanations [33], and numerous effects at the system architecture level (e.g., cache attacks [5,26,27]). Leaked information is even more easily accessible when the computational device is at the hands of an adversary, as is often the case for many modern devices such as smart-cards, TPM chips and (potentially stolen) mobile phones and laptops. Reducing such information leakage has proven excruciatingly difficult and costly, and its complete elimination is nowhere in sight.

There has lately been a growing amount of interest in coming up with precise definitions of security against side-channel attacks and in designing cryptographic algorithms that withstand these attacks (e.g., [24,19,28,17,11,8,29,3,25,9] and others). Micali and Reyzin [24] were the first to propose a general model of side-channel attacks. They model a side-channel attacker as a two part entity – the first is the *measurement apparatus* that performs measurements on the physical state of the device. This is done on behalf of the second entity which is the *adversarial observer*. The observer is assumed to be computationally powerful (e.g., polynomial-time or even unbounded), and takes as input the measurements of the apparatus. Thus, the power of the adversarial observer is primarily constrained by the quality of the information provided by the measurement apparatus.

It is interesting to note that even though computational devices leak abundantly, many side channel attacks are hard to carry out and some devices remain unbroken. This is due to the fact that *useful* measurements can often be difficult to realize in practice. Physical measurement apparatuses typically produce a "shallow" or "noisy" measurement of the state of the object, by combining some of its salient physical properties in a simple way. The measurement consists of a limited amount of information, obtained as a simple leakage function applied to the physical state of the device; any in-depth analysis happens only in the form of post-processing by the observer (rather than in the measurement apparatus).

In this work, we follow the paradigm of Ishai, Sahai, and Wagner [19] who construct a *general transformation* from any cryptographic algorithm into one that is functionally equivalent, but also leakage-resilient. The particular class of leakage functions they consider is the class of spatially local measurement functions, namely functions that read and output at most t bits of information. In particular, the leakage functions are completely oblivious of a large portion of the circuit's state.

In contrast, we are interested in security against global measurements, which are often easier to carry out than localized measurements that require a focus on specific wires or memory cells; in many side-channel attacks, the main practical difficulty for the attacker lies precisely in obtaining high spatial resolution and accuracy. Furthermore, global measurements are typically also more informative than local measurements. The question that motivates our work is whether, analogously to [19], we can construct a general circuit transformation that tolerates global side-channel measurements.

1.1 Our Results

Similar to Ishai et al. [19], we present a general transformation for arbitrary circuits that makes them resilient against certain classes of leakage. We now explain what these classes of leakage are and describe our techniques.

Measurement Apparatus. As in most prior work, the measurement apparatus in our model is not allowed to access some (very limited) portions of the computation. It can observe the rest of the computation, and return either a "computationally bounded" or a "noisy" function of the entire state.[5] Specifically, the measurement apparatus is modeled as computing either of the following types of *leakage functions*:

- a *computationally-bounded leakage function* f applied to the state of the device and all intermediate results that occur during the computation. The class of functions $\mathcal{L}$ from which f can be chosen models the practical limitations of the physical experimental setting available to the attacker. For example, $\mathcal{L}$ may consist of all functions computable by circuits of small depth.

  For the computational limitation to be meaningful, the function must also be limited in its output length (otherwise, the measurement apparatus could simply leak the entire state by "computing" the identity function).
- a *noisy leakage function*, where the measurement apparatus returns the accessed bit with probability $1 - p$ and flips it with probability p. The measurement apparatus can potentially access *all the bits of the computation* this way.

There are specific components of the circuit that we consider to be leak-free. We diverge from previous solutions by requiring that these components be *simple, stateless and computation-independent.* By this, we mean that the complexity of implementing the leak-free component is independent of the complexity of the computed function, and that it neither holds secrets nor maintains state. In particular, the leak-free component cannot hold the secret data used in the computation.

Specifically, our leak-free components, which we call *opaque gates*, are defined as follows. The opaque gate has *no inputs* and it outputs an element sampled according to a fixed distribution which is independent of the computation being

[5] When we refer to the state of a computation, we mean all the intermediate values produced during the computation on a particular input. Once this computation is done, the intermediate state is erased to make room for new computations. Thus, the leakage function can access all the bits of the current computation, but *not the past computations.* In fact, this is necessary to achieve security.

carried out. For example, an opaque gate that we consider is one that samples t uniformly random bits subject to the condition that they have even parity.

The leakage function cannot observe the innards of the opaque gate, but it can observe the wires going into and coming out of it. Although the requirement of a leak-free component is a strong one, the leak-free components we require are minimal in many senses:

1. It is a fixed standardized functionality which can be designed and validated once and added to one's VLSI "cell library" — which is far better than having to devise separate protection mechanisms for every circuit of interest.
2. It has no secret keys, no inputs and no internal state, i.e., it is independent of the computation in the circuit and merely samples from a distribution.
3. Alternatively, because we only need samples from a distribution, we can have the opaque "gate" simply read them one by one from a precomputed list. Thus, it suffices to have leak-proof one-time storage (a consumable "tape roll") instead of leak-proof computation. This is a viable option if the computation is performed only a bounded number of times.

Many variations of the leak-proof component assumption have been made in the literature. We highlight some of these works below.

- The "Oblivious RAM" model of Goldreich and Ostrovsky [15,16] considered memory to be leaky and the computation to be on a leak-free secure processor which stores a long-term secret key.
- The model of Micali and Reyzin [24] (and subsequent works [11,29,12]) reversed these roles: they assume that the memory cells that are not accessed during a computation step do not affect the observable leakage from that stage and cannot be measured by the apparatus. They called it the "only computations leaks" assumption.[6]
- The model of Goldwasser et al. [17] (which, although presented in the one-time programs setting, can be transformed into the leakage-resilient setting) relaxes the assumption of Micali and Reyzin, assuming only that some *read-only* memory (which holds secrets correlated to the computation) is leak-free if it is not "touched". The circuit, however, can only be executed a single time (or more generally, a bounded number of times).

The adversarial observer is all-powerful, and in each invocation of the circuit, it comes up with an input to the circuit as well as a leakage function, and obtains the output of the computation (on the given input), together with the leakage. The adversary decides which leakage function to use in a particular invocation *adaptively*, depending on all the information it received so far. We design circuit transformations that withstand such adversaries, and obtain the following main results.

[6] [11,29] point out that this requirement can be somewhat relaxed – it suffices that leakage of memory that is not used is independent of the leakage from computation.

Theorem 1 (Informal). *Let t be a (statistical) security parameter. There are circuit transformations that convert any (possibly stateful) circuit C into a circuit $\widehat{C}$ that is resilient against the following leakage functions:*

- *Constant-depth AC^0 circuits whose output length in each invocation is bounded by $t^{1-\delta}$, for any $\delta > 0$, and whose output length over the course of time is unbounded.*
- *Noisy measurements that leak the entire state of the circuit in each invocation, where each bit flipped independently with probability p, for any constant $p \in (0, 1/2]$.*

In both cases, the size of the transformed circuit $\widehat{C}$ is larger than the size of the original circuit C by a factor of $O(t^2)$.

Both results follow from a more general transformation that protects against *any* leakage class, provided that it has an associated encoding scheme (See Theorem 2 for details). We should note that although AC^0 is not a particularly strong class of functions, it is strong enough to allow for measuring approximate Hamming weight of the values on the wires [2]: something routinely measured by side-channel attacks in practice.

1.2 Overview of the Techniques

To protect against the kinds of information leakage described above, we encode the computation in a way that prevents the powerful computing observer from gaining additional information about the computation. We show that, indeed, for certain classes of leakage, *any* computation can be so encoded: namely, we give a method for transforming arbitrary circuits into new circuits, which are still leaky but whose leakage is useless to the attacker (in the sense of offering no advantage over black-box access to the original circuit's functionality).

More precisely, given any linear secret sharing scheme Π and a leakage class $\mathcal{L}$ which cannot decode Π[7], we show an explicit construction that transforms any circuit C into a circuit $\widehat{C}$ that is resilient against leakage in $\mathcal{L}$.

The gist of the construction is to encode every wire of C into a bundle of wires in $\widehat{C}$ using Π, where each wire in the bundle carries a single share. Similarly to Ishai et al. [19], we transform each gate in C into a gadget in $\widehat{C}$ which operates on encoded bundles. The gadgets are carefully constructed to use Π internally in a way that looks "essentially random" to leakage functions in $\mathcal{L}$, and we show that this implies that the whole content of the transformed circuit remains "essentially random" to a leakage in $\mathcal{L}$. Hence, the adversary gets no advantage from his observation of the leakage; formally, this is captured by a simulation-based definition.

An important contribution of this work is a general technique for proving security of leakage-resilient circuit transformations. Namely, we capture a strong

[7] Technically, the requirement that we make for the class $\mathcal{L}$ is a little bit stronger then not being able to decode.

notion of leakage-resilience for circuits or parts thereof, by saying that they are *reconstructible* if there exist certain efficient simulators for their internal wires that fool the leakage class. We then show a *composition lemma*: if all parts of a circuit are reconstructible then so is the whole circuit. This implies security of the transformation. Thus, security of the overall transformation is reduced to the reconstructibility of the individual gadgets. Our specific results using linear secret-sharing schemes follow this route, and other transformations can be built by devising different gate gadgets and merely showing that each is reconstructible by itself.

Other Related Approaches. Recently, starting from the work of Akavia et al. [3], several results have appeared that show security against adversaries that learn arbitrary functions of the secret state of a device *without requiring leak-free components* (see [3,4,9,21,25] and the references therein). All these constructions assume that the total leakage does not exceed the size of the secret key; in contrast, the total leakage in our case can be unbounded (subject only to the condition that in every time period, it is bounded). Furthermore, these works design specific cryptographic primitives such as encryption and signatures, whereas we focus on a general leakage-resilient transformation.

Standaert et al. [35] consider security against particular attacks such as Hamming weight attacks and analyze in [28] the security of a block-cipher based construction of a pseudorandom number generator.

2 Preliminaries and Definitions

Notation. Throughout the paper, we let t denote the security parameter. For $n \in \mathbb{N}$, let $[1, n]$ denote the set of integers $\{1, \ldots, n\}$. We denote function composition by $f \circ g : x \mapsto f(g(x))$. If $\mathcal{L}_1$ and $\mathcal{L}_2$ are two sets of functions, then $\mathcal{L}_2 \circ \mathcal{L}_1$ is a set of functions $\{f \circ g \mid f \in \mathcal{L}_2, g \in \mathcal{L}_1\}$. Vectors, denoted $\boldsymbol{v} = (v_1, \ldots, v_n)$, will be treated as column vectors.

If $\mathcal{D}$ is a probability distribution, then the notation $d \leftarrow \mathcal{D}$ means that the random variable d is drawn from $\mathcal{D}$. (If D is a set with no distribution specified, then by default we assume the uniform distribution.) If $\mathcal{D}$ is a randomized algorithm, then $d \leftarrow \mathcal{D}(x)$ denotes the output of $\mathcal{D}$ on input x. The notation $\mathcal{D} \equiv \mathcal{D}'$ means the distributions $\mathcal{D}$ and $\mathcal{D}'$ are identical.

Circuits. We consider circuits whose wires carry elements of an arbitrary finite field $\mathcal{K}$; in particular, we may set $\mathcal{K} = GF(2)$ to speak of a Boolean circuit. We consider circuits composed of the following gates operating on elements of $\mathcal{K}$ (in addition to the input, output, and memory gates): $\oplus, \ominus$, and $\odot$ (which compute, respectively, the sum, difference, and product in $\mathcal{K}$, of their two inputs), the "coin flip" gate \$ (which has no inputs and produces a random independently chosen element of $\mathcal{K}$), and for every $\alpha \in \mathcal{K}$, the constant gate $\texttt{const}_\alpha$ (which has no inputs and simply outputs α). Fanout is handled by a special `copy` gate that takes as input a single value and outputs two copies. Notice that `copy` gates compute the identity function (pass-through wires) and are present mainly for notational convenience.

For a circuit C containing w wires, a *wire assignment to* C is a string in $\mathcal{K}^w$, where each element represents a value on a wire of C. By $\mathcal{W}_C(X)$, we denote a distribution of wire assignments that is induced when a circuit C is being evaluated on an input X (in particular, if C is deterministic, then $\mathcal{W}_C(X)$ has only one element in its support). By $\mathcal{W}_C(X|Y)$, we denote the same distribution conditioned on the fact that the output of $C(X)$ was Y.

Two classes of circuits figure prominently in this paper.

- The first class of circuits is $\mathsf{SHALLOW}(d, s)$, the class of all deterministic circuits (i.e., ones without \$ gates) that have at most s $\oplus, \ominus$, and $\odot$ gates that are arranged at most d deep (i.e., the longest path in the circuit has at most d such gates on it).[8]
- The second is a class that contains a single *probabilistic* circuit $\mathcal{N}_p$ that gets as input a string $\boldsymbol{v}$, and outputs $\boldsymbol{w} = \boldsymbol{v} \oplus \boldsymbol{r}$, where each bit of $\boldsymbol{r}$ is *independently* 1 with probability p, and 0 with probability $1 - p$.

Stateful Circuits. A *stateful* circuit additionally contains memory gates, which have a single incoming edge and any number of outgoing edges.[9] Memory gates maintain state: at any clock cycle, a memory gate sends its current state down its outgoing edges and updates it according to the value of its incoming edge. Any cycle in the circuit must contain at least one memory gate.

The state of all memory gates at clock cycle i is denoted by M_i, with M_0 denoting the initial state. Inputs to and outputs from clock cycle i are denoted, respectively, by x_i and y_i. When a circuit is run in state M_{i-1} on input x_i, the computation will result in a wire assignment $\mathcal{W}_i$; the circuit will output y_i and the memory gates will be in a new state M_i. We will denote this by $(y_i, M_i, \mathcal{W}_i) \Lleftarrow C[M_{i-1}](x_i)$.

2.1 Leakage-Resilient Circuit Transformation

In this work, we construct a circuit transformation that takes as input a circuit and outputs a functionally equivalent, and yet, leakage-resilient circuit. Our definition generalizes the notion of a private transformation from Ishai, Sahai and Wagner [19]. For readers familiar with the model of Ishai et al., we note that the main difference is that whereas they speak of a "t-private transformation" that is secure against observers who can access at most t wires, we consider the general notion of a "$\mathcal{L}$-secure transformation" that is secure against observers who can evaluate any leakage function f within a class $\mathcal{L}$. One can recover the definition of Ishai et al. from our definition by simply letting $\mathcal{L}$ be the class of functions that output a subset of their input bits.

In order to understand our definition, it helps to keep the following scenario in mind. Imagine a circuit that has a secret stored within it (possibly in an encoded form) and it uses the secret together with a (public) input to come up

[8] Note that `copy` and $\mathtt{const}_\alpha$ gates do not count towards the depth d or the size s.

[9] Formally, our notion of a stateful circuit is essentially the same as the one in [19].

with an output; the encoding of the secret itself may get modified during the computation. For example, the circuit may implement a block cipher or the RSA signing algorithm, where the keys are secret. An adversarial observer (who we denote OBS) gets to interact with the circuit and the measurement apparatus by iterating the following process polynomially many times, in an adaptive manner: choosing an input for the circuit and a leakage function for the measurement apparatus, and receiving the output of the circuit on the chosen input and the physical leakage from the measurement apparatus. We would like to make sure that the ability to observe physical leakage does not help the observer: that is, the observer learns nothing more about the state of the circuit from the leakage than it could have learnt from input-output access.

Circuit Transformer. A circuit transformer TR takes as input a security parameter t, a circuit C, and an initial state M_0 and produces a new circuit $\widehat{C}$ and new initial state $\widehat{M_0}$.[10] We require the transformer to be *sound*: for all C and M_0, $C[M_0]$ should behave identically to $\widehat{C}[\widehat{M_0}]$. By "behave identically" we mean that for any number of clock cycles q and any set of inputs $x_1, x_2, \ldots, x_q$ (one for each clock cycle) the distribution of the outputs $y_1, y_2, \ldots, y_q$ is the same for C starting at state M_0 and $\widehat{C}$ starting at state $\widehat{M_0}$.

Security. We want to ensure that the transformed circuit leaks no useful information to an observer other than what the observer could have obtained by input-output access to the circuit's functionality. We define an $(\mathcal{L}, \tau, q)$*-observer* OBS to be an algorithm that: [11]

- Queries the circuit q times with inputs x_i, and receives the outputs y_i.
- For each execution of the circuit (say, with input x_i), chooses a leakage function $f \in \mathcal{L}$, and obtains $f(\mathcal{W}_C(x_i))$. That is, the leakage function f takes as input the circuit's wire assignment on input x_i, and outputs the resulting leakage.
- Runs for at most τ steps (not including the computation by the leakage function itself).

The observer makes the choice of which leakage function to use in a particular execution *adaptively*, depending on all the information it has received so far. To formalize that such an observer learns nothing useful, we show the existence of a simulator SIM, and prove that anything the observer learns can also be learned by SIM which only sees inputs and outputs of the circuit.

Consider the following two experiments that start with some circuit C in state M_0, and allow it to run for q iterations. In both experiments, we assume

[10] Throughout this paper, we use the hat notation $\hat{\square}$ (reminiscent of the proverbial "tinfoil hat") to designate circuit or components that are transformed for leakage-resilience.

[11] The number of observations q, the observer's running time τ, and various other running times and success probabilities are all parameterized by a security parameter t, which is given as input to the transformation TR. For readability, we will omit t from most of our discussion.

that OBS and SIM are stateful, namely, they remember their state from one invocation to the next.

```
$\mathsf{Exp}^{\mathrm{real}}_{\mathsf{TR}}(\mathsf{OBS}, \mathcal{L}, q, C, M_0)$:
$(\widehat{C}, \widehat{M}_0) \leftarrow \mathsf{TR}(C, M_0)$
$(x_1, f_1) \leftarrow \mathsf{OBS}(\widehat{C})$, with $f_1 \in \mathcal{L}$
For $i = 1$ to $q - 1$
    $(y_i, \widehat{M}_i, \mathcal{W}_i) \Lleftarrow \widehat{C}[\widehat{M}_{i-1}](x_i)$;
    $(x_{i+1}, f_{i+1}) \leftarrow \mathsf{OBS}(y_i, f_i(\mathcal{W}_i))$
$(y_q, M_q, \mathcal{W}_q) \Lleftarrow \widehat{C}[\widehat{M}_{q-1}](x_q)$;
Return output of $\mathsf{OBS}(y_q, f_q(\mathcal{W}_q))$.
```

```
$\mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}(\mathsf{SIM}, \mathsf{OBS}, q, C, M_0)$:
$(\widehat{C}, \widehat{M}_0) \leftarrow \mathsf{TR}(C, M_0)$
$(x_1, f_1) \leftarrow \mathsf{OBS}(\widehat{C})$, with $f_1 \in \mathcal{L}$
For $i = 1$ to $q - 1$
    $(y_i, M_i) \leftarrow C[M_{i-1}](x_i)$
    $\Lambda_i \leftarrow \mathsf{SIM}(x_i, y_i, f_i)$, with $\Lambda_i$ being the leakage
    $(x_{i+1}, f_{i+1}) \leftarrow \mathsf{OBS}(y_i, \Lambda_i)$
$(y_q, M_q) \leftarrow C[M_{q-1}](x_q)$;
$\Lambda_q \leftarrow \mathsf{SIM}(x_q, y_q, f_q)$
Return output of $\mathsf{OBS}(y_q, \Lambda_q)$.
```

The definition below says that the transformed circuit is leakage-resilient if the outputs of the two experiments above are indistinguishable.

Definition 1. *Let $\mathcal{L}$ be a class of circuits, and let $\tau = \tau(t)$, $\tau' = \tau'(t)$, $q = q(t)$ and $\epsilon = \epsilon(t)$ be functions of the security parameter t. A circuit transformer* TR *is said to be $(\mathcal{L}, \tau, \tau', q, \epsilon)$-secure if for every $(\mathcal{L}, \tau, q)$-observer* OBS*, there is a simulator* SIM *that runs in time τ' such that for all circuits C and all initial states M_0,*

$$\left|\Pr[\mathsf{Exp}^{\mathrm{real}}_{\mathsf{TR}}(\mathsf{OBS}, \mathcal{L}, q, C, M_0) = 1] - \Pr[\mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}(\mathsf{SIM}, \mathsf{OBS}, q, C, M_0) = 1]\right| \leq \epsilon,$$

where the probabilities are taken over all the coin tosses involved in the experiments. We refer to a circuit transformer being $\mathcal{L}$-secure, as a shorthand for saying that it is $(\mathcal{L}, \mathsf{poly}(t), \mathsf{poly}(t), \mathsf{poly}(t), \mathsf{negl}(t))$-secure in the above sense.

Remark. We note that a stronger result is obtained when $\mathcal{L}$, τ and q are as large as possible (as it allows for more leakage functions, and stronger observers), when τ' is as close as possible to τ, and when the distinguishing advantage ϵ is as small as possible (because either of these indicate a tighter simulation).

3 Circuit Transformation from Linear Secret-Sharing

Our main result states that if there exists a linear encoding scheme for elements of any field $\mathcal{K}$ (taking a single element to t elements) for which encodings of any two values are indistinguishable by functions in a class $\mathcal{L}$, then there exists a circuit transformation that is secure against a slightly less powerful leakage class $\mathcal{L}_{\mathsf{TR}}$. (Jumping ahead, we remark that the leakage class $\mathcal{L}$ is essentially the same as the class $\mathcal{L}_{\mathsf{TR}}$ "augmented with" a depth-3 circuit of size $O(t^2)$).

We now describe the main elements in the circuit transformation.

Encoding for the wires. Our transformation can be based on any *linear encoding scheme* $\Pi = (\mathsf{Enc}, \mathsf{Dec})$, which maps a single element of $\mathcal{K}$ to a vector in $\mathcal{K}^t$ and back. In the simplest case of $\mathcal{K} = \mathsf{GF}(2)$, an encoding of a bit x is a random string of t bits whose exclusive-or is x. More generally, for security parameter t, a linear encoding scheme Π is defined by a *decoding vector* $\boldsymbol{r} = (r_1, \ldots, r_t) \in \mathcal{K}^t$ and the decoding function $\mathsf{Dec} : (y_1, \ldots, y_t) \mapsto \sum_i y_i r_i = \boldsymbol{r}^{\intercal}\boldsymbol{y}$. Enc is a (probabilistic) algorithm that, on input x, chooses uniformly at random an element of $\mathsf{Dec}^{-1}(x)$.

Linear encoding schemes include the aforementioned parity encoding, as well as any threshold or non-threshold linear secret sharing scheme, e.g., [32,6,20].

We need the notion of leakage-indistinguishability of an encoding scheme which, roughly speaking, formalizes what it means for an encoding of two values to be indistinguishable in the presence of leakage. In conjunction with formalizing this notion, let us first introduce a more general definition that speaks about leakage-indistinguishability of two distributions.

Definition 2. *Two distributions X and Y are said to be $(\mathcal{L}, p, \tau, \epsilon)$-*leakage-indistinguishable*, if for any observer OBS, running in time τ and making at most p queries to its oracle where each query f is a function in $\mathcal{L}$,*

$$|\Pr[x \leftarrow X; \mathsf{OBS}^{\mathsf{Eval}(x,\cdot)}(1^t) = 1] - \Pr[y \leftarrow Y; \mathsf{OBS}^{\mathsf{Eval}(y,\cdot)}(1^t) = 1| \leq \epsilon,$$

where $\mathsf{Eval}(x, \cdot)$ takes as input a leakage function f and outputs $f(x)$.

We say that an encoding scheme *Π is $(\mathcal{L}, p, \tau, \epsilon)$-*leakage-indistinguishable *if for any $a, b \in \mathcal{K}$ the two distributions $\mathsf{Enc}(a)$ and $\mathsf{Enc}(b)$ are $(\mathcal{L}, p, \tau, \epsilon)$-leakage-indistinguishable. If $\tau = \mathsf{poly}(t)$ and $\epsilon = \mathsf{negl}(t)$, then we abbreviate this to $(\mathcal{L}, p)$-leakage-indistinguishable.*

Opaque gates. In our scheme, the transformed circuit $\widehat{C}$ is built of the same gate types as the original circuit, with the addition of a new *opaque* gate denoted $\mathcal{O}$. As mentioned in the introduction, the $\mathcal{O}$ gate has *no inputs*, and outputs an encoding sampled from the distribution $\mathsf{Enc}(0)$. Crucially, while the wires coming out of this gate can be observed by the leakage function, we assume that its internals do not leak (we show how to somewhat relax this condition in the full version). For the case of $\mathcal{K} = \mathsf{GF}(2)$ our leak-free component can be implemented by a circuit that works as follows: generate t random bits $b_0, \ldots, b_{t-1}$ and output the bits $c_i := b_i \oplus b_{i+1 \bmod t}$ for $0 \leq i \leq t-1$.

As mentioned in the introduction, our leak-free component is minimal in many senses; the only sense in which it is not minimal is that its size is proportional to the security parameter t. Improving on this is left as an important open problem.

We now state our main theorem. The rest of this section describes the transformation, and the next section contains an overview of the proof of security.[12]

Theorem 2. *Let t be the security parameter, and let $\mathcal{L}_{\mathsf{TR}}$ be some class of leakage functions. If there exists a linear encoding scheme Π that is $(\mathcal{L}_{\Pi}, 2)$-leakage-indistinguishable, then there exists a circuit transformation TR that is $\mathcal{L}_{\mathsf{TR}}$-secure provided that:*

$$\mathcal{L}_{\Pi} \supseteq \mathcal{L}_{\mathsf{TR}} \circ \mathsf{SHALLOW}(3, O(t^2))$$

[12] A complete statement of the theorem keeps track of other parameters such as the running-time of the observer as well as the simulator, and the distinguishing advantage. We postpone the more detailed theorem statement to the full version.

The transformation increases the size of each multiplication gate by a factor of $O(t^2)$ and the rest of the circuit by a factor of $O(t)$, where the constants hidden in $O(\cdot)$ are small.

3.1 The Transformation for Stateless Circuits

We will first describe our transformation for circuits without any memory gates, which we call, like in [19], *stateless circuits*. We then show how to extend the transformation to general (i.e., stateful) circuits.

Given a stateless circuit C, our transformation TR produces the transformed circuit $\widehat{C}$ as follows (see Figure 1 for an example). Each wire w in C is replaced by a *wire bundle* in $\widehat{C}$, consisting of t wires $\boldsymbol{w} = (w_1, \ldots, w_t)$, that carry an encoding of w. Each gate is transformed into a *gadget*, built out of gates, which takes encodings and outputs encodings. Crucially, note that the internals of these gadgets may leak. The gadgets themselves are described in Figure 2.

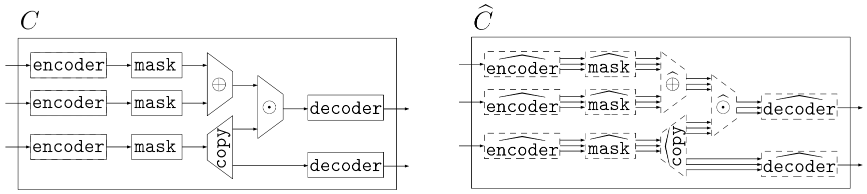

Fig. 1. Example of a circuit C for the function $(a, b, c) \mapsto ((a \oplus b) \odot c, c)$, and the corresponding transformed circuit $\widehat{C}$. Three parallel lines denote encoding (t wires). Dashed borders indicate a gadgets, whose internal wires leak. Note that in C, the special gates `encoder`, `decoder`, `mask` and `copy` are just the identity and are present for notational convenience.

Since our gadgets operate on encoded values, $\widehat{C}$ needs to have a subcircuit at the beginning that encodes the inputs and another subcircuit at the end that decodes the outputs. However, in our proofs, we want to be able to also reason about transformed circuits without encoding and decoding. Thus, we do not require that every transformed circuit $\widehat{C}$ should have such encoding and decoding. Instead, we introduce artificial input and output gates that can be part of C for syntactic purposes. If such gates are present (as they would be on any "complete" circuit that one would actually wish to transform), then $\widehat{C}$ will include input encoding and output decoding. If they are not, then $\widehat{C}$ will operate on already encoded inputs and produce encoded outputs.

More precisely, if we wish for $\widehat{C}$ to include input encoding and output decoding, then the circuit C given to TR must have two special gates in sequence on every input wire: an `encoder` gate followed by a `mask` gate, both of which are simply the identity. Also, on every output wire there must be a special `decoder` gate, which is also the identity. These special gates must not appear anywhere

```
**Transformation** $c \leftarrow a \odot b \;\Rightarrow\; \boldsymbol{c} \leftarrow \boldsymbol{a} \widehat{\odot} \boldsymbol{b}$:
    Compute the $t \times t$ matrix
        $B \leftarrow \boldsymbol{a}\boldsymbol{b}^{\mathsf{T}} = (a_i b_j)_{1 \le i,j \le t}$ using $t^2$ $\odot$ gates
    Compute the $t \times t$ matrix $S$
        where each column of $S$ is output by $\mathcal{O}$
    $U \leftarrow B + S$ (using $t^2$ $\oplus$ gates)
    Decode each row of $U$ using $t-1$ $\oplus$ gates,
        $t$ $\odot$ gates, and $t$ $\mathtt{const}_\alpha$ gates
        to obtain $\boldsymbol{q} \leftarrow U\boldsymbol{r}$,
        where $\boldsymbol{r}$ is the decoding vector
        (it does not matter how this decoding is
        performed as long as there are $O(t)$ wires
        in the decoding subcircuit and each one
        carries some linear combination of the
        wires being decoded, plus possibly a
        constant)
    $\boldsymbol{o} \leftarrow \mathcal{O}$
    $\boldsymbol{c} \leftarrow \boldsymbol{q} + \boldsymbol{o}$ (using $t$ $\oplus$ gates)
```

```
**Transformation** $c \leftarrow \$ \;\Rightarrow\; \boldsymbol{c} \leftarrow \widehat{\$}$:
    $c_i \leftarrow \$$    for $i \in [1, t]$
    Output $\boldsymbol{c}$
```

```
**Transformation** $c \leftarrow a \oplus b \;\Rightarrow\; \boldsymbol{c} \leftarrow \boldsymbol{a} \widehat{\oplus} \boldsymbol{b}$
            (or $c \leftarrow a \ominus b \;\Rightarrow\; \boldsymbol{c} \leftarrow \boldsymbol{a} \widehat{\ominus} \boldsymbol{b}$):
    $\boldsymbol{q} \leftarrow \boldsymbol{a} + \boldsymbol{b}$ (or $\boldsymbol{q} \leftarrow \boldsymbol{a} - \boldsymbol{b}$)
            using $t$ $\oplus$ (or $\ominus$) gates
    $\boldsymbol{o} \leftarrow \mathcal{O}$
    $\boldsymbol{c} \leftarrow \boldsymbol{q} + \boldsymbol{o}$ (using $t$ $\oplus$ gates)
```

```
**Transformation** $b \leftarrow \mathtt{mask}(a) \;\Rightarrow\; \boldsymbol{b} \leftarrow \widehat{\mathtt{mask}}(\boldsymbol{a})$
    $\boldsymbol{o} \leftarrow \mathcal{O}$
    $\boldsymbol{b} \leftarrow \boldsymbol{a} + \boldsymbol{o}$ (using $t$ $\oplus$ gates)
```

```
**Transformation** $a \leftarrow \mathtt{const}_\alpha \;\Rightarrow\; \boldsymbol{a} \leftarrow \widehat{\mathtt{const}}_\alpha$,
        for any $\alpha \in \mathcal{K}$
    Let $\boldsymbol{\alpha}$ be a fixed arbitrary encoding of $\alpha$.
    $\boldsymbol{o} \leftarrow \mathcal{O}$
    $\boldsymbol{a} \leftarrow \boldsymbol{\alpha} + \boldsymbol{o}$ (using $t$ $\oplus$ gates)
```

```
**Gadget** $(\boldsymbol{b}, \boldsymbol{c}) \leftarrow \widehat{\mathtt{copy}}(\boldsymbol{a})$
    $\boldsymbol{o_1} \leftarrow \mathcal{O}$, $\boldsymbol{o_2} \leftarrow \mathcal{O}$
    $\boldsymbol{b} \leftarrow \boldsymbol{a} + \boldsymbol{o_1}$ (using $t$ $\oplus$ gates)
    $\boldsymbol{c} \leftarrow \boldsymbol{a} + \boldsymbol{o_2}$ (using $t$ $\oplus$ gates)
```

Fig. 2. Gadgets used in the stateless circuit transformation TR.

else in C. In $\widehat{C}$ each `encoder` gate is replaced by an $\widehat{\mathtt{encoder}}$ gadget which performs encoding (see below), each `decoder` gate is replaced by a $\widehat{\mathtt{decoder}}$ gadget that performs decoding (see below), and each `mask` gate is replaced by a $\widehat{\mathtt{mask}}$ gadget (that is needed for security and is described in Figure 2).

The $\widehat{\mathtt{encoder}}$ gadget takes an input $a \in \mathcal{K}$ and outputs an encoding (i.e., a wire bundle) $\boldsymbol{a} \in \mathcal{K}^t$ of a. The encoding can be chosen arbitrarily from the support of $\mathsf{Enc}(a)$: $\boldsymbol{a} = (r_1^{-1}a, 0, \ldots, 0)$. The $\widehat{\mathtt{decoder}}$ gadget takes an encoding (i.e., a wire bundle) $\boldsymbol{a} \in \mathcal{K}^t$ of a and outputs $a \leftarrow \mathsf{Dec}(\boldsymbol{a})$. This is computed by a decoding circuit with just $\mathtt{const}_\alpha$, $\oplus$, and $\odot$ gates. The operation of all the gadgets is described in 2. For the soundness of our transformation, we refer the reader to the full version.

Incidentally, observe that because every gadget other than $\widehat{\mathtt{encoder}}$ or $\widehat{\mathtt{decoder}}$ ends with a masking by an output of $\mathcal{O}$,[13] and wire bundles do not fan-out (instead, they go through the $\widehat{\mathtt{copy}}$ gadget), each connecting wire bundle carries an encoding of its value that is chosen *uniformly and independently of all the wires in the transformed circuit.* This fact, together with the construction of the gadgets, is what enables the simulation.

Handling Stateful Circuits. To augment the above stateless circuit transformation to a full circuit transformation, we have to explain how to transform the initial state M_0 and what to do with each memory gate. The initial state is

[13] One can instead define the basic gadgets as not including this masking with $\mathcal{O}$, and instead place a `mask` gate on every wire. The resulting transformation is similar.

replaced by a randomly chosen encoding $\mathsf{Enc}(M_0)$. Each memory gate is replaced by a gadget that consists of t memory gates to store the encoding followed by a $\widehat{\texttt{mask}}$ gadget to guarantee re-randomization of the state.[14]

4 Proof of Security

Conceptually, the proof of security for the circuit transformation in Section 3 proceeds in two steps. First, consider a mental experiment where each gadget in the transformed circuit $\widehat{C}$ is *perfectly opaque*. Namely, the only wires that the observer OBS can "see" are the *external wires* of the gadgets that connect the output of a gadget to the input of another gadget (these are exactly the wires that carry encodings of the values in the circuit C). The wires internal to the gadgets are off-limits to OBS. Once in this (imaginary) world, we use the first key property of our gadgets, namely

Re-randomizing: The output of each gadget in $\widehat{C}$ is a *uniformly random* encoding of the output of the corresponding gate in C.[15]

Letting $w_1, \ldots, w_m$ denote the values of the wires in C, the re-randomizing property says that the wire-bundles in $\widehat{C}$ that are external to the gadgets are distributed like $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m)$ where the $\boldsymbol{w}_i \leftarrow \mathsf{Enc}(w_i)$ are *random and independent* encodings of the bit w_i.

The simulator does not know the value w_i (because it does not know the secret state in the circuit), but will simulate it with a random encoding of a random value w'_i. Now, the leakage indistinguishability of the encoding scheme tells us that given the leakage from any of these encodings (individually), it is hard to tell if the underlying value is w_i or w'_i. By a hybrid argument, the same holds for a vector of *independent* encodings of m values as well, which is what the simulator uses.

Before we declare victory (in this imaginary world), let us look a little more closely at the hybrid argument. At each hybrid step, we will prove indistinguishability by a reduction to the security of the encoding scheme. In other words, we will show by reduction that if OBS equipped with functions from $\mathcal{L}_{\mathsf{TR}}$ can distinguish two hybrid wire distributions, then some adversary OBS_{Π}, equipped with functions from a slightly larger class $\mathcal{L}_{\Pi}$, can distinguish two encodings. Given an encoding, our reduction will need to fake the remaining wires of the circuit and give them as input to the function from $\mathcal{L}_{\mathsf{TR}}$.

Efficiency of such a reduction is particularly important. If OBS specifies a leakage function $f \in \mathcal{L}_{\mathsf{TR}}$ for $\widehat{C}$, then OBS_{Π} will specify its own leakage function

[14] Masking the output of the memory gadget has two reasons: first, we want to allow the total leakage to be much larger than the size of the state, and second, we want to allow the adversary to choose leakage functions adaptively.

[15] Of course, given the values of the internal wires of the gadgets as well, the outputs of the gadgets are not independent encodings any more. But, note that we are still in the mental experiment where the observer does not get to see the internals of the gadgets.

f_{Π} for the encoding and return its result to OBS. This leakage function f_{Π} has to fake (in a way that will look real to f and OBS) all the wires of $\widehat{C}$ before it can invoke f. At the same time, f_{Π} should not be much more complex than f, because our result is more meaningful when difference between the power of $\mathcal{L}_{\Pi}$ and the power of $\mathcal{L}_{\mathsf{TR}}$ is smaller. The main trick is for OBS_{Π} to hardwire as much as possible into f_{Π}, so that when f_{Π} observes the encoding, it has to do very little work before it can invoke f. In fact, in this imaginary situation, all the remaining wires can be hardwired into f_{Π} because of independence of encodings, so f_{Π} has to simply invoke f on its input wires and hardwired values.

The second step in the proof is to move from the mental experiment to the real world, where the internals of the gadgets also leak. Unlike in the mental experiment, where the values of all wire bundles were independent, values of wires inside a gadget are correlated to its input and output wire bundles. Thus, they cannot be hardwired into f_{Π}. Nor can they be computed by f_{Π}, because the complexity of the gadgets is too high.

Handling this problem requires invoking the second key property of the gadgets, namely:

Reconstructibility: We say that a pair of strings (X, Y) is *plausible for* $\widehat{G}$ if $\widehat{G}$ might output Y on input X. For every gadget $\widehat{G}$, there exists a distribution $\mathsf{REC}_{\widehat{G}}$ over low-complexity functions $\mathcal{R}$, which takes as input X, Y and produces a *simulated distribution* of the internal wires of $\widehat{G}$. If for any plausible X, Y this distribution is $(\mathcal{L}, \tau, \epsilon)$-leakage-indistinguishable from the actual distribution of the internal wires of $\widehat{G}$ (conditioned on X and Y), then we say that $\widehat{G}$ is $(\mathcal{L}, \tau, \epsilon)$-reconstructible by $\mathcal{R}$, and call $\mathsf{REC}_{\widehat{G}}$ a $(\mathcal{L}, \tau, \epsilon)$-reconstructor.

In the following we will often omit the parameters τ and ϵ.

We use this property to handle leakage from gadgets. Given reconstructors for each single gadget we can show that a transformed circuit that is encoding-based (i.e. the gadgets operate on encodings) and composed of reconstructible gadgets is secure according to Definition 1. On a high-level we will replace each gadget with its reconstructor in addition to replacing connecting wire bundles with random encodings. The proof that the simulation is indistinguishable requires first doing a hybrid argument over gadgets as they are replaced by reconstructors one-by-one, and then modifying the hybrid argument over the wires described above. In the hybrid argument over the wires, f_{Π} can have hardwired values for every wire in the circuit except the gadgets connected to the challenge encoding, which will be computed by f_{Π} using the low-complexity function given by the reconstructor. This allows for a very efficient reduction. The formal statement of the composition lemma is given in Lemma 3.

Let us now move on to building reconstructors for two simple gadgets.

4.1 Reconstructors for Single Gadgets

We present proof sketches for the reconstructibility of the $\widehat{\oplus}$ and $\widehat{\odot}$ gadget.

Lemma 1 **($\widehat{\oplus}$ and $\widehat{\ominus}$ gadgets are reconstructible).** *For any class of circuits $\mathcal{L}$, the $\widehat{\oplus}$ and $\widehat{\ominus}$ gadgets are $(\mathcal{L}, \infty, 0)$-reconstructible, where the reconstructor can be computed by* $\mathsf{SHALLOW}(2, O(t))$.

Proof. In this sketch we will do the proof only for $\widehat{\oplus}$. The reconstructor $\mathsf{REC}_{\widehat{\oplus}}$ is the distribution whose only support is the following circuit $R_{\widehat{\oplus}}$. On inputs (X, Y) where $X = (\boldsymbol{a}, \boldsymbol{b})$ (i.e., the desired input of the $\widehat{\oplus}$ gate), and $Y = (\boldsymbol{c})$ (i.e., its desired output), $R_{\widehat{\oplus}}$ assigns the wires of $\widehat{\oplus}$ to $\boldsymbol{q} \leftarrow \boldsymbol{a} \oplus \boldsymbol{b}$ and $\boldsymbol{o} \leftarrow \boldsymbol{c} \ominus \boldsymbol{q}$.

If X, Y are chosen as in the definition of a reconstructor, then the resulting output of $R_{\widehat{\oplus}}$ is identically distributed to the wire distribution $\mathcal{W}_{\widehat{\oplus}}(X|Y)$, since in both cases $\boldsymbol{o}$ takes the only possible consistent value $\boldsymbol{o} \leftarrow \boldsymbol{c} \ominus \boldsymbol{q}$. Notice that $R_{\widehat{\oplus}}$ can be computed by a circuit of depth 2 because on inputs X, Y it first computes $\boldsymbol{q} \leftarrow \boldsymbol{a} \oplus \boldsymbol{b}$ and based on that $\boldsymbol{o} \leftarrow \boldsymbol{c} \ominus \boldsymbol{q}$. The $\ominus$ and $\oplus$ gates above operate only on single field elements, so $R_{\widehat{\oplus}}$ requires $O(t)$ size. □

Let us now give a proof sketch for the $\widehat{\odot}$ reconstructor. Notice that the main technical difficulty is the fact that our simulation has to be shallow whereas the real $\widehat{\odot}$ gadget is already a deep circuit. In the following, let $\mathcal{K} = \mathsf{GF}(2)$.

Lemma 2 **($\widehat{\odot}$ is reconstructible).** *Let $\mathcal{L}_{\widehat{\odot}}$ be a class of functions, and assume that the encoding Π is $\mathcal{L}_{\Pi}$-leakage-indistinguishable, where $\mathcal{L}_{\Pi} \supseteq \mathcal{L}_{\widehat{\odot}} \circ$* $\mathsf{SHALLOW}(2, O(t^2))$. *Then, the $\widehat{\odot}$ gadget is $\mathcal{L}_{\widehat{\odot}}$-reconstructible, where the reconstructor can be computed by* $\mathsf{SHALLOW}(2, O(t^2))$.

Proof (sketch). The reconstructor $\mathsf{REC}_{\widehat{\odot}}$ takes as inputs (X, Y), where $X = (\boldsymbol{a}, \boldsymbol{b})$, and $Y = (\boldsymbol{c})$ and is defined as follows:

1. Sample U uniformly from $\mathcal{K}^{t \times t}$ and compute the values on the wires in the subcircuits for the computation of $\boldsymbol{q}$. Hard-wire the results as $R_{\widehat{\odot}}$'s outputs.
2. On input X, $R_{\widehat{\odot}}$ computes the matrix $B \leftarrow (a_i \odot b_j)_{i,j}, i, j \in [1, t]$ and outputs it as part of the wire assignment.
3. $R_{\widehat{\odot}}$ computes $S \leftarrow B - U$ and $\boldsymbol{o} \leftarrow \boldsymbol{c} - \boldsymbol{q}$.

$\mathsf{REC}_{\widehat{\odot}}$ has size $O(t^2)$ (because it needs to compute matrices B and S) and depth 2, because S is computed from B, that in turn has been computed from the inputs.

It remains to show that the distribution $R_{\widehat{\odot}}(X, Y)$ produced by the reconstructor and the actual wire distribution $\mathcal{W}_{\widehat{\odot}}(X|Y)$ are leakage-indistinguishable by leakage functions in $\mathcal{L}_{\widehat{\odot}}$. Since U is computed as $B + S$ it suffices to show that S can be replaced by a matrix sampled uniformly at random from $\mathcal{K}^{t \times t}$.

We prove it by a hybrid argument and define hybrids $\mathcal{W}^{\ell}_{\widehat{\odot}}(X|Y)$ ($\ell \in [0, t]$) as $\mathcal{W}_{\widehat{\odot}}(X|Y)$, except that for the first ℓ columns of S the elements are drawn uniformly from $\mathcal{K}$. We show the leakage-indistinguishability between two consecutive hybrids by a reduction to the encoding leakage-indistinguishability. As part of this reduction we build the observer OBS_{Π} that runs $\mathsf{OBS}_{\widehat{\odot}}$ and has to answer its leakage queries $f_{\widehat{\odot}} \in \mathcal{L}_{\widehat{\odot}}$. OBS_{Π} runs $f_{\widehat{\odot}}$ as part of its own leakage function $f_{\Pi} \in \mathcal{L}_{\Pi}$. However, f_{Π} only expects a single target encoding $\boldsymbol{e}$ as input, whereas functions from $\mathcal{L}_{\widehat{\odot}}$ expect a full wire assignment for $\widehat{\odot}$. Thus, before f_Π runs $f_{\widehat{\odot}}$, a wire simulator f_S, computes a wire assignment for $\widehat{\odot}$ given only the target encoding $\boldsymbol{e}$. To keep the reduction tight (and our result meaningful), f_S has to be very simple; i.e. we use the input $\boldsymbol{e}$ as little as possible and hard-wire most of the values of the wires of $\widehat{\odot}$ into f_S. For any X, Y:

1. From X compute $B = (a_i b_j)_{i,j\in[1,t]}$ and hard-wire $\boldsymbol{a}, \boldsymbol{b}, B$ into f_S.
2. Hard-wire the columns $1 \dots \ell - 1$ to random encodings and $\ell + 1 \dots t$ to $\mathsf{Enc}(0)$. The ℓth column is filled with the challenge encoding $\boldsymbol{e}$.
3. Hard-wire all elements of $U = B + S$ into f_S except for the ℓth column. For the ℓth column, compute for each $i \in [1, t]$, the value $U_{i,\ell} \leftarrow B_{i,\ell} + e_i$.
4. The wires in the decoding sub-circuits to compute $\boldsymbol{q}$ from U carry the $\oplus$ of some row $\{U_{i,j}\}_j$. If a wire in the sub-circuit does not depend on $U_{i,\ell}$ (i.e., the input to f_S), then pre-compute its value and hard-wire the intermediate result. On the other hand, if it depends on $U_{i,\ell} = B_{i,\ell} + e_i$, then pre-compute a partial sum except the term that depends on e_i and hard-wire the result. On input $\boldsymbol{e}$, f_S computes the missing outputs by $\oplus$-ing the relevant parts of $\boldsymbol{e}$.
5. With fixed Y and $\boldsymbol{q}$ from (3) compute $\boldsymbol{o} \leftarrow Y - \boldsymbol{q}$ and output it.

It is not difficult to check that f_S outputs a valid wire assignment for $\widehat{\odot}$ that is either distributed as $\mathcal{W}^{\ell-1}_{\widehat{\odot}}(X|Y)$ or $\mathcal{W}^{\ell}_{\widehat{\odot}}(X|Y)$. If $\boldsymbol{e}$ is drawn from $\mathsf{Enc}(0)$, then the ℓth column of S is assigned an encoding drawn from $\mathsf{Enc}(0)$. Since all the other wires are computed honestly using either hard-wired values or the input $\boldsymbol{e}$, $f_S(\mathsf{Enc}(0))$ and $\mathcal{W}^{\ell-1}_{\widehat{\odot}}(X|Y)$ are distributed identically. If $\boldsymbol{e} \leftarrow \mathsf{Enc}(x)$, for $x \in \mathcal{K}$, then the ℓth column of S is assigned an encoding drawn from $\mathsf{Enc}(x)$, hence, we get that $f_S(\mathsf{Enc}(x))$ and $\mathcal{W}^{\ell}_{\widehat{\odot}}(X|Y)$ are distributed identically. Since f_S needs to compute the ℓth column of U, the values in the decoding sub-circuits, and from $\boldsymbol{q}$ the value of $\boldsymbol{o}$, $f_S \in \mathsf{SHALLOW}(2, O(t^2))$. Together with the t hybrids, we get that $\mathcal{W}_{\widehat{\odot}}(X|Y)$ and $R_{\widehat{\odot}}(X, Y)$ are $(\mathcal{L}_{\widehat{\odot}}, t\epsilon)$-leakage-indistinguishable, if Π is $(\mathcal{L}_\Pi, \epsilon)$-leakage-indistinguishable (where $\mathcal{L}_\Pi \supseteq \mathcal{L}_{\widehat{\odot}} \circ \mathsf{SHALLOW}(2, O(t^2))$). □

The rerandomizing property of the simple gadgets follows immediately from the fact that every gadget's output is masked by the output of $\mathcal{O}$.

4.2 Security of Full Circuit Transformation

Until now we showed that individual gadgets are re-randomizing, and reconstructible. The following central lemma, that is proved in the full version, states how to compose reconstructors for single gadgets to yield a reconstructor for the entire circuit.

Lemma 3 (Composition Lemma). *Let $\mathcal{L}_{\widehat{C}}$ be some set of leakage functions and $\epsilon_\Pi > 0, \tau_\Pi > 0, t > 0$. Let Π be $(\mathcal{L}_\Pi, \tau_\Pi, \epsilon_\Pi)$-leakage-indistinguishable. Let C be a stateless circuit of size s, without* `encoder` *or* `decoder` *gates with k_I inputs and k_O outputs. Then the transformed circuit $\widehat{C}$ is rerandomizing and $(\mathcal{L}_{\widehat{C}}, \tau_{\widehat{C}}, \epsilon_{\widehat{C}})$-reconstructible by* $\mathsf{SHALLOW}(2, (k_\mathrm{I} + k_\mathrm{O})O(t^2))$ *where $\mathcal{L}_\Pi = \mathcal{L}_{\widehat{C}} \circ$* $\mathsf{SHALLOW}(3, O(t^2))$*, $\epsilon_{\widehat{C}} = \epsilon_\Pi s(t + 2)$, and $\tau_{\widehat{C}} = \tau_\Pi - O(st^2)$.*

There is one caveat that remains in proving security according to Definition 1: the $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$ gadget are not reconstructible, however, the simulator can easily include them into his simulation since the inputs and outputs of these gadgets are known.

We would like to make a final remark: the circuit transformation that we discussed so far are based on any linear encoding scheme, however, the proof techniques that we introduced along the way are more general. Note that Lemma 3 relies essentially on the fact that the gate gadgets are rerandomizing and reconstructible. One can obtain an analogously result using *any* (not necessarily linear) encoding scheme and a corresponding set of sound gate gadgets that are rerandomizing and reconstructible. We refer the interested reader to the full version.

5 Security against Constant Depth Leakage

In this section, we show how to use the general circuit transformation from Section 3 to achieve security against leakage functions that can be computed by constant-depth circuits.

5.1 AC^0 Leakage

The first leakage class we consider is AC^0, the class of constant-depth, polynomial-size circuits formed out of NOT gates and unbounded fan-in AND and OR gates. Let $\mathcal{C}(d, s, \lambda)$ denote the class of AND-OR-NOT Boolean circuits with depth d, size s and λ bits of output.

The Encoding. The encoding we use in this case is the parity encoding. The (randomized) parity encoding of a bit b is a sequence of bits $(b_1, \ldots, b_t)$ which are uniformly random subject to the condition that their parity is the bit b. This encoding can be computed in many different ways, for example, as:

ENC(b): Generate bits $b_1, \ldots, b_{t-1}$ uniformly at random, and set $b_t := b \oplus \bigoplus_{i=1}^{t-1} b_i$.

Obviously, the decoding function for the parity encoding is simply the parity function, namely the function that outputs the exclusive-or of the t bits in the encoding.

The parity encoding is hard to decode for AC^0 circuits. The classical result of Håstad [18] (which builds on [1,14]), translated to our definition, states that the parity encoding of the bits 0 and 1 are indistinguishable by circuits in the class $\mathcal{C}(d, 2^{t^{1/d}}, 1)$ for any constant d. This protects against AC^0 circuits that output 1 bit. Using a recent result of Dubrov and Ishai [10, Theorem 3.4], we can protect against the circuit class $\mathcal{C}(d, e^{O(t^\delta/d)}, t^{1-\delta})$ for any $0 < \delta < 1$, namely AC^0 circuits that output up to $t^{1-\delta}$ bits.

We obtain the following theorem by instantiating Theorem 2 with the parity encoding, and using the above observations about the leakage-indistinguishability of the parity encoding against AC^0 circuits. The reader is referred to the full version for a tight statement and a formal proof of security.

Theorem 3. *Let t be the security parameter, and $0 < \delta < 1$, and $d \in \mathbb{N}$ be constants. Then, there exists a circuit transformation that is $\mathcal{L}_{\mathsf{AC}^0,d,\delta}$-secure where $\mathcal{L}_{\mathsf{AC}^0,d,\delta} = \mathcal{C}(d-4, e^{O(t^\delta)/d}, t^{1-\delta})$ is the class of all Boolean AND-OR-NOT circuits of depth at most $d-4$, size at most $e^{O(t^\delta)/d}$ and output length at most $t^{1-\delta}$.*

In particular, the theorem states that the transformation is secure against AC^0 circuits (constant depth, polynomial-size circuits) that output at most $t^{1-\delta}$ bits, *for any constant* $\delta > 0$.

5.2 $\mathsf{ACC}^0[q]$ leakage

A natural way to extend the class of leakage functions from AC^0 to something more general is to allow the leakage function to have parity gates. Clearly, such circuits can decode the parity encoding, but are there still other linear encoding schemes that cannot be decoded by even such circuits? It turns out that such encodings indeed exist. For any integer q, let $\mathtt{MOD}_q$ be the gate that outputs 0 if the sum of its inputs is 0 modulo q, and 1 otherwise. The class $\mathcal{C}_{\mathtt{MOD}\text{-}q}(d, s, \lambda)$ is defined to be the functions computable by circuits made of $\mathtt{NOT}$ gates and unbounded fan-in $\mathtt{AND}, \mathtt{OR}$ and $\mathtt{MOD}_q$ gates, with depth d, size s and output length λ. For example, letting $q = 2$, we get the class of depth d circuits that include parity gates as well.

The encoding scheme we use in this case is the mod-q' encoding scheme, for some q' that is co-prime to q, defined analogously to the parity encoding scheme in Section 5.1. By a result of Razborov and Smolensky [31,34], for any distinct primes q' and q, the mod-q' encoding is leakage-indistinguishable for functions in the class $\mathcal{C}_{\mathtt{MOD}\text{-}q}(O(1), \mathsf{poly}(t), 1)$, i.e., $\mathsf{ACC}^0[q]$ circuits with output length 1. Since the mod-q' encoding is linear, we can apply Theorem 2 to get a secure circuit transformation.

6 Security against Noisy Leakage

So far, we considered leakage classes that are constrained in terms of their computational power and output length. In this section, we consider the noisy leakage model, where the leakage consists of the values of *all the wires* in the circuit, except that each bit is flipped with some probability $p \in [0, 1/2]$. More precisely, the class of noisy leakage functions is represented by the circuit class $\mathcal{L} = \{\mathcal{N}_p\}_{p \in [0,1/2]}$, where each circuit $\mathcal{N}_p$ is probabilistic, and is defined as follows: Let B_p be the binomial distribution with parameter p which outputs 1 with probability p and 0 otherwise. Then, $\mathcal{N}_p(\boldsymbol{x}) = \boldsymbol{x} \oplus \boldsymbol{b}$, where each bit b_i is drawn from the distribution B_p and the different b_i are independent.

Ideally, we would hope that the circuit transformation in Section 3 provides security against noisy leakage as well. However, this turns out to be false, and in fact, there is an explicit attack against the transformation in Section 3 (as well as the circuit transformation of Ishai et al. [19]) in the presence of noisy leakage, even when the noise is very small.

We outline the basic idea of the attack here. Specifically, the attack is against the construction of the multiplication gadget $\widehat{\odot}$ in Figure 2. The gadget takes as input two encodings $\boldsymbol{a}$ and $\boldsymbol{b}$ and first computes the t^2 bits $\{a_i \wedge b_j : i, j \in [t]\}$. Consider the first t bits $(a_1 \wedge b_1, \ldots, a_1 \wedge b_t)$. If $a_1 = 0$, then all these bits are 0, whereas if $a_1 = 1$, then roughly half of them are 1. Given such disparity, the observer can determine whether a_1 is 0 or 1, even if he is given a noisy version of these t bits (for any noise parameter $p < 1/2$). Proceeding in a similar way, he can reconstruct all the bits a_i, and thus the input bit a itself. The fundamental reason why this attack works is that the construction of the $\widehat{\odot}$ gadget in Figure 2 has *high input locality*, namely it accesses the input bits a large number of times.

6.1 A New Circuit Transformation against Noisy Leakage

We construct a new circuit transformation against noisy leakage. The transformation proceeds in the same way as in Section 3, except for the construction of the multiplication gadget $\widehat{\odot}$. The new construction of the multiplication gadget avoids the attack outlined below, and is constructed using a new opaque gate that we call $\mathcal{M}$ (in addition to the opaque gate $\mathcal{O}$). We stress that the opaque gate $\mathcal{M}$ that we design and use, inherits the main characteristics of the opaque gate $\mathcal{O}$ in that it is *stateless*, and *independent of the computation*. In other words, $\mathcal{M}$ simply produces samples from a fixed distribution.

In what follows, we describe the specification of the opaque gate $\mathcal{M}$ as well as the construction of the $\widehat{\odot}$ gadget.

The Opaque Gate $\boldsymbol{\mathcal{M}}$. The opaque gate $\mathcal{M}$ is probabilistic, takes no inputs and operates in the following way: Sample $2t$ uniformly random 0-sharings $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t \leftarrow \mathcal{O}$ and $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t \leftarrow \mathcal{O}$. Let $\mathbf{R}$ and $\mathbf{S}$ be the following two $t \times t$ matrices:

$$\mathbf{R} = \begin{pmatrix} \boldsymbol{r}_1 \\ \vdots \\ \bigoplus_{j=1}^{i} \boldsymbol{r}_j \\ \vdots \\ \bigoplus_{j=1}^{t} \boldsymbol{r}_j \end{pmatrix} \text{ and } \mathbf{S} = \begin{pmatrix} \boldsymbol{s}_1 \\ \vdots \\ \bigoplus_{j=1}^{i} \boldsymbol{s}_j \\ \vdots \\ \bigoplus_{j=1}^{t} \boldsymbol{s}_j \end{pmatrix}$$

Let $R_{i,j}$ (resp. $S_{i,j}$) denote the $(i,j)^{th}$ entry of the matrix $\mathbf{R}$ (resp. $\mathbf{S}$). Define $\mathbf{R} \otimes \mathbf{S}$ to be the "inner product of the matrices $\mathbf{R}$ and $\mathbf{S}$", when written out as bit-strings. That is,

$$\mathbf{R} \otimes \mathbf{S} = \bigoplus_{i,j} R_{i,j} S_{i,j}$$

The output of the opaque gate $\mathcal{M}$ is the tuple $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_t, u)$ where $u = \mathbf{R} \otimes \mathbf{S}^t$, the inner product of the matrices $\mathbf{R}$ and *the transpose of* $\mathbf{S}$.

The new Multiplication Gadget $\widehat{\odot}$. The operation of the multiplication gadget $\widehat{\odot}$ proceeds in two stages.

- The first stage uses a gadget $\widehat{\mathsf{mult}}$ that takes as input two encodings $\boldsymbol{a} = (a^{(1)}, \ldots, a^{(t)})$ and $\boldsymbol{b} = (b^{(1)}, \ldots, b^{(t)})$, and outputs a *longer encoding* $\boldsymbol{q} = (q^{(1,1)}, \ldots, q^{(t,t)})$ of size t^2.
- The second stage "compresses" this longer encoding into an encoding $\boldsymbol{c} = (c^{(1)}, \ldots, c^{(t)})$, using a gadget $\widehat{\mathsf{compress}}$.

We first describe how the (sub-)gadget $\widehat{\mathsf{mult}}$ works.

1. First, generate $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_t, u) \leftarrow \mathcal{M}$.
2. Define $\boldsymbol{a}_0 := \boldsymbol{a}$ and $\boldsymbol{b}_0 := \boldsymbol{b}$. Compute the encodings $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ iteratively as follows. For $1 \leq i \leq t$, set
$$\boldsymbol{a}_i = \boldsymbol{a}_{i-1} \oplus \boldsymbol{r}_i, \text{ and } \boldsymbol{b}_i = \boldsymbol{b}_{i-1} \oplus \boldsymbol{s}_i$$
3. Let $a_i^{(j)}$ (resp. $b_i^{(j)}$) denote the j^{th} bit of the vector $\boldsymbol{a}_i$ (resp. $\boldsymbol{b}_i$). Output $\boldsymbol{q} = (q^{(1,1)}, \ldots, q^{(t,t)})$ defined as follows:
$$q^{(i,j)} = \begin{cases} a_1^{(1)} \wedge b_1^{(1)} \oplus u & \text{if } (i,j) = (1,1) \\ a_i^{(j)} \wedge b_j^{(i)} & \text{otherwise} \end{cases}$$
(Note the asymmetry in the evaluation, namely the bit $a_i^{(j)}$ is multiplied with the bit $b_j^{(i)}$, where the subscript and the superscript are switched; this asymmetry is intentional, and indeed, crucial to the correctness).
4. Generate $\mathbf{z} \leftarrow \mathcal{O}_{t^2}$ (thus, $\mathbf{z}$ is a uniformly random t^2-bit string whose entries xor to 0). Output $\boldsymbol{w} := \boldsymbol{q} \oplus \mathbf{z}$.

Now, we invoke the $\widehat{\mathsf{compress}}$ gadget on the output of the $\widehat{\mathsf{mult}}$ gadget. The $\widehat{\mathsf{compress}}$ gadget takes t^2 bits $(q^{(1,1)}, \ldots, q^{(t,t)})$ and outputs t bits $(c^{(1)}, \ldots, c^{(t)})$ such that $\bigoplus_{i,j} q^{(i,j)} = \bigoplus_i c^{(i)}$. The construction of the $\widehat{\mathsf{compress}}$ gadget proceeds in the following way.

1. Split the bits $q^{(i,j)}$ into t blocks of t bits each.
2. Construct a tree of $\widehat{\oplus}$ gadgets that takes as input t *blocks* of t bits each, and outputs *one block* of t bits. (The structure of the tree can be arbitrary.) Apply the tree to the bits $q^{(i,j)}$ and call $\boldsymbol{c} = (c^{(1)}, \ldots, c^{(t)})$ the output.

The correctness of the $\widehat{\odot}$ gadget can be verified by a simple computation, and is omitted. The efficiency of implementation is practically the same as that in 3. Namely, the transformation converts a circuit of size s into another circuit of size $O(s \cdot t^2)$, where t is the security parameter. We now outline the main ideas behind the proof of security of the new transformation against noisy leakage.

Outline of the Security Proof. As in Section 3, the proof proceeds in two steps. First, we show that the gadgets are re-randomizing and reconstructible. In other words, this says that the internals of a gadget reveal no more useful information than its inputs and output. Secondly, we apply a general version of the Composition Lemma (Lemma 3) to conclude that since each individual gadget is re-randomizing and reconstructible, the entire circuit transformation is leakage-resilient. We describe these two steps in a little more detail below.

It is easy to see that the gadgets are re-randomizing. The key difference from Section 3 is that in the proof of reconstructibility, we are not concerned about the *computational efficiency* of the reconstructor, but rather the *number of times the reconstructor accesses its input*. This is a consequence of the fact that the larger the number of noisy copies of an encoding $\boldsymbol{e}$ (with independent binomial noise) the observer sees, the easier it is for him to tell if $\boldsymbol{e}$ is an encoding of 0 or 1. Thus, the bulk of the effort in the design of the circuit transformation as well as the reconstructor is in ensuring that the inputs and the intermediate values are "touched" as few times as possible. The technical heart of the proof (similar to the theorems of [13,18,10] for the AC^0 case) is a lemma which states that for any constant c and any fixed vectors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_c$, the distribution of $(\mathcal{N}_p(\boldsymbol{e} \oplus \boldsymbol{f}_1), \ldots, \mathcal{N}_p(\boldsymbol{e} \oplus \boldsymbol{f}_c))$ when $\boldsymbol{e}$ is an encoding of 0 or 1 are statistically close. We refer the reader to the full version for the design of the reconstructors and the formal proof.

Acknowledgments. We thank Debajyoti Bera, Shafi Goldwasser, Yuval Ishai, Moni Naor, Ran Raz and Ronen Shaltiel for helpful discussions, and the anonymous Eurocrypt reviewers for their detailed comments.

References

1. Miklos Ajtai. $\sum_1^1$-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):48, 1983.
2. Miklós Ajtai. Approximate counting with uniform constant-depth circuits, 1993.
3. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.
4. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.
5. Daniel J. Bernstein. Cache-timing attacks on AES. `http://cr.yp.to/papers.html#cachetiming`, 2005.
6. G.R. Blakley. Safeguarding cryptographic keys. 48:313–317, 1979.
7. David Brumley and Dan Boneh. Remote timing attacks are practical. *Comput. Netw.*, 48(5):701–716, 2005.
8. Francesco Davì and Stefan Dziembowski. Leakage-resilient storage. Cryptology ePrint Archive, Report 2009/399, 2009. `http://eprint.iacr.org/`.
9. Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC '09*, pages 621–630. ACM, 2009.
10. Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *STOC '06*, pages 711–720. ACM, 2006.
11. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS '08*, pages 293–302. IEEE Computer Society, 2008.
12. Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.
13. Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *SFCS '81: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, pages 260–270, Washington, DC, USA, 1981. IEEE Computer Society.

14. Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.
15. Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *STOC*, pages 182–194, 1987.
16. Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.
17. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *CRYPTO*, pages 39–56, 2008.
18. Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20, 1986.
19. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO'03*, pages 463–481, 2003.
20. Mauricio Karchmer and Avi Wigderson. On span programs. In *Structure in Complexity Theory Conference*, pages 102–111, 1993.
21. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.
22. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.
23. Markus G. Kuhn. *Compromising emanations: eavesdropping risks of computer displays.* PhD thesis, University of Cambridge, 2003. Technical Report UCAM-CL-TR-577.
24. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC'04*, pages 278–296, 2004.
25. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.
26. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of AES. In *CT-RSA*, pages 1–20, 2006.
27. Colin Percival. Cache missing for fun and profit. presented at BSDCan 2005, Ottawa, 2005; see `http://www.daemonology.net/hyperthreading-considered-harmful`, 2005.
28. Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In *ASIACCS*, pages 56–65, 2008.
29. Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.
30. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.
31. Alexander Razborov. Lower bounds for the size of circuits of bounded depth with basis and, xor. Math. Notes of the Academy of Science of the USSR 41, 1987.
32. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
33. Adi Shamir and Eran Tromer. Acoustic cryptanalysis: on nosy people and noisy machines. presented at the Eurocrypt 2004 rump session; see `http://tromer.org/acoustic`, 2004.
34. Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *STOC*, pages 77–82, 1987.
35. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT*, pages 443–461, 2009.