An Information-Theoretic Model for Adaptive Side-Channel Attacks

Boris Köpf
Information Security
ETH Zurich, Switzerland
bkoepf@inf.ethz.ch

David Basin
Information Security
ETH Zurich, Switzerland
basin@inf.ethz.ch

ABSTRACT

We present a model of adaptive side-channel attacks which we combine with information-theoretic metrics to quantify the information revealed to an attacker. This allows us to express an attacker's remaining uncertainty about a secret as a function of the number of side-channel measurements made. We present algorithms and approximation techniques for computing this measure. We also give examples of how they can be used to analyze the resistance of hardware implementations of cryptographic functions to both timing and power attacks.

Categories and Subject Descriptors

D.4.6 [**Software**]: Security and Protection

General Terms

Security

1. INTRODUCTION

Side-channel attacks against cryptographic algorithms aim at breaking cryptography by exploiting information that is revealed by the algorithm's physical execution. Characteristics such as running time [18, 6], cache behavior [28], power consumption [19], and electromagnetic radiation [15, 31] have all been exploited to recover secret keys from implementations of different cryptographic algorithms. Side-channel attacks are now so effective that they pose a real threat to the security of devices like smart-cards, which can be subjected to different kinds of measurements. This threat is not covered by traditional notions of cryptographic security and models for proving resistance against such attacks are only now emerging [25, 37].

Two factors determine whether an attacker can successfully mount a side-channel attack on a system and recover a secret key (or other secret data): first, he must be able to extract information about the key through side-channel measurements. Second, he must be able to effectively recover the key from the extracted information. To prove that a system is resistant to side-channel attacks, one must ensure that no realistic attacker can fulfill both conditions. In theory, key recovery may be computationally infeasible, even if the key is completely revealed in an information-theoretic sense.[1] In practice, however, this is often not the case: keys have been successfully recovered from side-channel information from a broad range of cryptographic algorithms on different platforms, e.g., [6, 8, 10, 19, 28]. When key recovery from available information is feasible, the security of a system depends entirely on the first factor: the quantity of information about the key that can be extracted from the system's side-channels. There are no known results for determining this quantity for a given system with respect to the attackers that can interact with it as in, e.g., [6, 8, 18, 28].

In this paper, we propose a solution to this problem for deterministic systems: we present a model that allows us to express the quantity of information that an adaptive attacker can extract from a system, and we provide algorithms and approximation techniques for computing this quantity.

Our model is based on a definition of attack strategies, which are explicit representations of the adaptive decisions made by an attacker during attacks. We combine attack strategies with information-theoretic entropy measures. This allows us to express the attacker's expected uncertainty about the secret after he has performed a side-channel attack following a given strategy.

Even if the attacker can perform arbitrary off-line analysis on measurement data, his interactions with the system under attack are often expensive or limited and their number needs to be considered when reasoning about a system's vulnerability. For example, the system may refuse multiple interactions with the same agent or bound the number of times it re-uses a secret, such as a session key. By quantifying over all attack strategies of a fixed length n, we express what attackers can, in principle, achieve in n attack steps. We use this to define a function Φ that gives a lower bound on the expected uncertainty about the key as a function of the number of side-channel measurements. Since the bounds given by Φ are information-theoretic, they hold for any kind of analysis technique that a computationally unbounded attacker might apply to analyze the measurements. Note that such strong bounds are realistic. In template attacks [10], the entire information contained in each measurement is effectively exploited for key recovery.

We give algorithms and (exponential) complexity bounds

[1]For example, an RSA public key contains all the information about the corresponding private key.

CCS'07, October 29–November 2, 2007, Alexandria, Virginia, USA.

for computing Φ. Furthermore, we propose two heuristic techniques that reduce this complexity and thereby allow us to estimate a system's vulnerability for keyspace sizes for which the direct computation of Φ is infeasible.

Our approach is parametric in the physical characteristics of the side-channel, which can be described by deterministic hardware models of the target system. In this way, the accuracy of our method only depends on the accuracy of the system model used. Furthermore, our approach accommodates different notions of entropy that correspond to different kinds of brute-force guessing.

Finally, we have implemented our approach and we report on experimental results using the resulting prototype. We have analyzed hardware implementations of cryptographic algorithms for their resistance to timing and power attacks, thereby obtaining the following results: (1) an attacker can extract one operand's Hamming weight from the timing of a direct implementation of integer multiplication, but a more defensive implementation reveals no information; (2) only a few timing measurements are needed to extract the entire exponent information from the finite-field exponentiation algorithm of [14]; and (3) one power trace of a finite-field multiplication algorithm contains all information about one of its operands. These results illustrate the potential of our approach for both detecting possible side-channel attacks and showing their absence.

Overall, our contributions are twofold. Theoretically, we develop a simple model for adaptive side-channel attacks that connects information-theoretic notions of security to models for physical characteristics of hardware. Practically, we show that our model can be applied to nontrivial hardware implementations of cryptographic algorithms and we use it to analyze their vulnerability to power and timing attacks.

The remainder of this paper is structured as follows. In Section 2 we introduce our model of adaptive attacks and in Section 3 we extend it with information-theoretic measures. In Section 4 we give algorithms and complexity bounds for computing these measures and we report on experimental results in Section 5. We present related work and draw conclusions in Sections 6 and 7.

2. THE MODEL

We start by describing the assumptions underlying our model.

2.1 Attackers and Side-Channels

Attack Scenario.

Let K be a finite set of keys, M be a finite set of messages, and D be an arbitrary set. We consider cryptographic functions of type $F : K \times M \to D$, where we assume that F is invoked by two collaborating callers. One caller is an honest agent that provides a secret argument $k \in K$ and the other caller is a malicious agent (the *attacker*) that provides the argument $m \in M$. Examples of F are encryption and decryption functions and MACs.

We assume that the attacker has no access to the values of k and $F(k, m)$, but that he can make physical observations about F's implementation I_F that are associated with the computation of $F(k, m)$. Examples of such observations are the power or the time consumption of I_F during the computation (see [19, 24] and [18, 8, 6, 28], respectively).

Typically, the key k is a long-term secret that remains constant during different calls to F. The malicious agent performs an attack in order to gather information for deducing k or narrowing down its possible values. Such an *attack* consists of a sequence of *attack steps*, each with two parts: A *query* phase in which the attacker decides on a message m and sends it to the system, and a *response* phase in which he observes I_F while it computes $F(k, m)$. The attack is *adaptive* if the attacker can use the observations made during the first n steps to choose the query for the n+1st step. An attack ends if either the honest agent changes the key (assuming the independence of the old and new keys) or if the attacker stops querying the system.

Discrete Side-Channel Measurements.

We assume that the attacker can make one side-channel measurement per invocation of the function F and that no measurement errors occur. Furthermore, we assume that the attacker has full knowledge about the implementation I_F. These strong assumptions are justified as our goal is to give bounds on what side-channel attackers can, in principle, achieve.

Given our assumptions, a *side-channel* is a function $f_{I_F} : K \times M \to O$, where O is the set of possible observations, and f_{I_F} is known to the attacker. We will usually leave I_F implicit and abbreviate f_{I_F} by f.

Example 1. Suppose that F is implemented in synchronous (clocked) hardware and that the attacker is able to determine I_F's running times up to single clock cycles. Then the timing side-channel of I_F can be modeled as a function $f : K \times M \to \mathbb{N}$ that represents the number of clock ticks consumed by an invocation of F. A hardware simulation environment can be used to compute f.

Example 2. Suppose F is given in a description language for synchronous hardware. Power estimation techniques such as [27, 40] can be used to determine a function $f : K \times M \to \mathbb{R}^n$ that estimates an implementation's power consumption during n clock ticks.

If the function f accurately models the side-channel, then any randomness in a physical attacker's measurements is due to noise and the assumption of error-free measurements is a safe worst-case characterization of the attacker's capabilities. One can also derive f from the implementation I_F.

Example 3. Suppose a hardware implementation I_F of F is given. As in template attacks [10], average values of I_F's power consumption for fixed input values k and m can be used to define $f(k, m)$.

As in template attacks, the attacker can use noise models of the target implementation to extract the maximal information from his measurements, that is, the value of f.

2.2 Attack Strategies

An adaptive attacker chooses his queries with the knowledge of previously revealed side-channel information. We use trees to define attack strategies, which capture these adaptive choices. Subsequently, we also formalize non-adaptive attacks, that is, attacks in which the malicious agent gathers all side-channel information before performing any analysis. To begin with, we motivate an abstract view of attack steps, which is the key to the simplicity of our model.

Attacker's Choices and Knowledge.

During the query phase, the attacker decides which message $m \in M$ to query the system with. In the response phase, he learns the value $f(k, m)$. In general, he cannot deduce k from $f(k, m)$. What he can deduce though (assuming full knowledge about the implementation I_F and unbounded computational power) is the set of keys that are coherent with the observation $f(k, m)$. Namely, assuming a fixed f, we say that a key k is *coherent with* $o \in O$ *under* $m \in M$ whenever $f(k, m) = o$ holds. Two keys k and r are *indistinguishable under* m iff $f(r, m) = f(k, m)$. Note that for every $m \in M$, *indistinguishability under* m is an equivalence relation on K. For every $o \in O$, the set of keys that are coherent with o under m forms an equivalence class of indistinguishability under m. The set of keys that are coherent with the attacker's observation under the attacker's input is the set of keys that could possibly have led to this observation; we use this set to represent the attacker's knowledge about the key after an attack step.

Functions as Sets of Partitions.

We now provide an abstract formalization of attack steps. As is standard, a *partition* $P = \{B_1, \ldots, B_r\}$ of K is a set of pairwise disjoint *blocks* with $\bigcup_{i=1}^{r} B_i = K$. Observe that every equivalence relation R on K corresponds to a partition P_R of K, where the equivalence classes of R are the blocks of P_R. In this way, a function $f : K \times M \to O$ gives rise to a set of partitions $\mathcal{P}_f = \{P_m \mid m \in M\}$, where P_m is the partition induced by indistinguishability under m.

In terms of the set of partitions $\mathcal{P}_f$, the two phases of an attack step can be described as follows.

1. In the query phase, the attacker chooses a partition $P \in \mathcal{P}_f$.

2. In the response phase, the system reveals the block $B \subseteq P$ that contains k.

Conversely, given a set of partitions $\mathcal{P}$, one can easily define a (non-unique) function f, with $\mathcal{P}_f = \mathcal{P}$. In this sense, the partition-based and the functional viewpoints are equivalent. Formalizing f in terms of $\mathcal{P}_f$ only abstracts from the concrete values that f takes, which are irrelevant for assessing the information that is revealed by f. For clarity of presentation, we will subsequently focus on the partition-based viewpoint.

For this, we need to introduce additional notation. We say that a partition Q of a set K *refines* a partition P of K (denoted by $Q \sqsubseteq P$) iff every block of Q is contained in some block of P. For $A \subseteq K$, we define the *restriction* of P to A as $\{A \cap B \mid B \in P\}$ and denote it by $A \cap P$. Clearly, $A \cap P$ is a partition of A. For partitions P and Q, we define $P \cap Q$ as the partition $\{A \cap B \mid A \in P, B \in Q\}$. Note that $P \cap Q \sqsubseteq P$ and $P \cap Q \sqsubseteq Q$. We are now ready to generalize from single attack steps to entire attacks.

Formalizing Attack Strategies.

To model adaptive attacks, we proceed as follows. We assume a fixed set of partitions $\mathcal{P}$ of K and we use a tree whose nodes are labeled with subsets of K to formalize the attacker's decisions with respect to his possible observations. In this tree, an attack step is represented by a node together with its children. The label A of the parent node is the set of keys that are coherent with the attacker's observation at this point; hence it represents the basis for the attacker's decision. The labels of the children form a partition of A. We require that this partition is of the form $A \cap P$ for some $P \in \mathcal{P}$. This corresponds to the attacker's choice of a query. By observing the system's response, the attacker learns which successor's block actually contains the key. This node is the starting point for subsequent attack steps. Trees of this form represent attack strategies, which we formalize below.

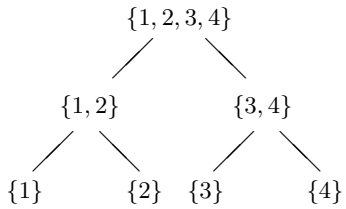

Figure 1: Attack Strategy

Example 4. Let $K = \{1, 2, 3, 4\}$ and consider the set of partitions $\mathcal{P} = \{\{\{1\}, \{2, 3, 4\}\}, \{\{1, 2\}, \{3, 4\}\}, \{\{1, 2, 3\}, \{4\}\}\}$ of K. Suppose the attacker picks $\{\{1, 2\}, \{3, 4\}\}$ as his first query. If the system responds with $\{1, 2\}$, the attacker chooses $\{\{1\}, \{2, 3, 4\}\}$ as his next query. Otherwise, he chooses $\{\{1, 2, 3\}, \{4\}\}$. In this way, he can determine any key in two steps. The corresponding attack strategy is depicted in Figure 1.

Formally, let $T = (V, E)$ be a tree with nodes V and edges $E \subseteq V \times V$. For every node $v \in V$, we denote the set of its successors as $succ(v) = \{w \mid (v, w) \in E\}$. The *height* of a tree T is the length of a longest path in T.

Definition 1. Let $\mathcal{P}$ be a set of partitions of K. An *attack strategy against* $\mathcal{P}$ is a triple (T, r, L), where $T = (V, E)$ is a tree, $r \in V$ is the root, and $L : V \to 2^K$ is a node labeling with the following properties:

1. $L(r) = K$, and

2. for every $v \in V$, there is a $P \in \mathcal{P}$ with $L(v) \cap P = \{L(w) \mid w \in succ(v)\}$.

An attack strategy is of *length* l if T has height l. An *attack* is a path $(r, \ldots, t)$ from the root r to a leaf t of T.

Requirement 1 of Definition 1 expresses that, a priori, every key in K is possibly chosen by the honest agent. Requirement 2 expresses that the labels of the children of each node form a partition of their parent's label and that this partition is obtained by intersecting the label with a $P \in \mathcal{P}$. A simple consequence of requirements 1 and 2 is that the labels of the leaves of an attack strategy partition the label of the root node. This leads to the following definition.

Definition 2. The partition *induced* by the attack strategy $\mathfrak{a} = (T, r, L)$ is the set $\{L(v) \mid v \text{ is a leaf of } T\}$, which we denote by $P_{\mathfrak{a}}$. We define the set of keys that are *coherent with an attack* $a = (r, \ldots, t)$ as $L(t)$.

Observe that this definition of coherence corresponds to our prior definition considering attacks (r, t) of length 1: The keys that are coherent with an observation o under m form

the block $L(t)$ that the system reveals when queried with P_m.

To clearly distinguish between adaptive and non-adaptive attacks, we briefly sketch how the latter can be cast in our model.

Non-adaptive Attack Strategies.

An attack strategy is called *non-adaptive* if the attacker does not have access to the system's responses until the end of the attack. Thus, when choosing a message, he cannot take into account the outcomes of his previous queries. In our model, this corresponds to the attacker choosing the same partition in all nodes at the same level of the attack strategy.

Formally, the *level* of a node $v \in V$ in an attack strategy $\mathfrak{a} = (T, r, L)$, with $T = (V, E)$, is the length of the path from the root r to v. A tree is *full* if all leaves have the same level.

Definition 3. An attack strategy $\mathfrak{a} = (T, r, L)$ is *non-adaptive* iff T is full and for every level i there is a $P_i \in \mathcal{P}$ such that $L(v) \cap P_i = \{L(w) \mid w \in succ(v)\}$, for every v of level i.

Note that we require the tree to be full to exclude observation-dependent termination of attacks. The structure of non-adaptive attacks is simpler than that of adaptive attacks and we can straightforwardly represent the partitions induced.

PROPOSITION 1. *Let $\mathfrak{a}$ be a non-adaptive attack strategy of length l against $\mathcal{P}$. Then we have*

$$P_{\mathfrak{a}} = \bigcap_{i=0}^{l-1} P_i \ ,$$

where $P_i \in \mathcal{P}$ is the partition chosen at level $i \in \{0, \ldots, l-1\}$ of $\mathfrak{a}$.

PROOF. We prove the assertion by induction on the length l of $\mathfrak{a} = (T, r, L)$. If $l = 0$, we have $P_{\mathfrak{a}} = L(r) = K = \bigcap \emptyset$. If $l > 0$, consider the full subtree T' of height $l-1$ of T. We have $P_{\mathfrak{a}} = \{L(w) \mid w \text{ is a leaf of } T\} = \bigcup_v \{L(w) \mid w \in succ(v)\}$, where v ranges over the leaves of T'. By Definition 3 and the induction hypothesis, we conclude $P_{\mathfrak{a}} = \bigcup_v L(v) \cap P_{l-1} = \bigcap_{i=0}^{l-2} P_i \cap P_{l-1} = \bigcap_{i=0}^{l-1} P_i$. □

Observe that, since $\cap$ is commutative, the order of the queries is irrelevant. This is consistent with the intuitive notion of a non-adaptive attack, as the side-channel information is only analyzed when the attack has finished.

In the next section, we will extend the model presented with measures for the quantitative evaluation of attack strategies. Afterwards, we use this quantitative model to give bounds on what attackers can possibly achieve in a given number of attack steps.

3. QUANTITATIVE EVALUATION OF ATTACK STRATEGIES

In Section 2, we used the induced partition $P_{\mathfrak{a}}$ to represent what an attacker learns about the key by following an attack strategy $\mathfrak{a}$. Intuitively, the attacker obtains more information (or equivalently, reduces the uncertainty) about the key as $P_{\mathfrak{a}}$ is refined. Information-theoretic entropy measures can be used to express the remaining uncertainty. Focusing on the remaining entropy, instead of the attacker's information gain, provides a concrete, meaningful measure that quantifies the attacker's effort for key recovery by brute-force guessing under the worst-case assumption that he can actually determine the set of keys that are coherent with his observations during the attack. The viewpoints are informally related by the equation *initial uncertainty = information gain + remaining uncertainty*, which we will make explicit in the following.

3.1 Measures of Uncertainty

We now introduce three entropy measures, which correspond to different notions of resistance against brute-force guessing. Presenting these different measures serves two purposes. First, it accommodates the fact that different types of guesses and different notions of success for brute-force guessing correspond to partially incomparable notions of entropy [22, 7, 30]. Second, it demonstrates how the possibilistic model presented in Section 2 can serve as a basis for a variety of probabilistic extensions.

In the following, assume a probability measure p is given on K and is known to the attacker. For a random variable $X : K \to \mathcal{X}$ with range $\mathcal{X}$, we define $p_X : \mathcal{X} \to \mathbb{R}$ as $p_X(x) = \sum_{k \in X^{-1}(x)} p(k)$, which in the literature is often denoted by $p(X = x)$. For a partition P of K, there are two variables of particular interest. The first is the random variable U that models the random choice of a key in K according to p (i.e., $U = id_K$). The second is the random variable V_P that represents the choice of the enclosing block (i.e., $V_P : K \to P$, where $k \in V_P(k)$). For an attack strategy $\mathfrak{a}$, we abbreviate $V_{P_{\mathfrak{a}}}$ by $V_{\mathfrak{a}}$.

Shannon Entropy.

The *(Shannon) entropy* [35] of a random variable $X : K \to \mathcal{X}$ is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p_X(x) \log_2 p_X(x) \ .$$

The entropy is a lower bound for the average number of bits required for representing the results of independent repetitions of the experiment associated with X. Thus, in terms of guessing, the entropy $H(X)$ is a lower bound for the average number of binary questions that need to be asked to determine X's value [7].

Given another random variable $Y : K \to \mathcal{Y}$, $H(X|Y = y)$ denotes the entropy of X given $Y = y$, that is, with respect to the distribution $p_{X|Y=y}$. The *conditional entropy* $H(X|Y)$ of X given Y is defined as the expected value of $H(X|Y = y)$ over all $y \in \mathcal{Y}$, namely,

$$H(X|Y) = \sum_{y \in \mathcal{Y}} p_Y(y) H(X|Y = y) \ .$$

Entropy and conditional entropy are related by the equation $H(XY) = H(Y) + H(X|Y)$, where XY is the random variable defined as $XY(k) = (X(k), Y(k))$.

Consider now an attack strategy $\mathfrak{a}$ and the corresponding variables U and $V_{\mathfrak{a}}$. $H(U)$ is the attacker's initial uncertainty about the key and $H(U|V_{\mathfrak{a}} = B)$ is the attacker's remaining uncertainty about the key after learning the key's enclosing block $B \in P_{\mathfrak{a}}$. $H(U|V_{\mathfrak{a}})$ is the attacker's expected remaining uncertainty about the key after performing an attack with strategy $\mathfrak{a}$. As the value of $V_{\mathfrak{a}}$ is determined from U, we have $H(UV_{\mathfrak{a}}) = H(U)$. The equation $H(U) = H(V_{\mathfrak{a}}) + H(U|V_{\mathfrak{a}})$

is the formal counterpart of the informal equation given at the start of this section.

Guessing Entropy.

The guessing entropy of a random variable X is the average number of questions of the kind "does $X = x$ hold" that must be asked to guess X's value correctly [22].

As we assume p to be public, the optimal procedure is to try each of the possible values in order of their decreasing probabilities. W.l.o.g., let $\mathcal{X}$ be indexed such that $p_X(x_i) \geq p_X(x_j)$, whenever $i \leq j$. Then the *guessing entropy* $G(X)$ of X is defined as $G(X) = \sum_{1 \leq i \leq |\mathcal{X}|} i\, p_X(x_i)$. Analogously to the conditional Shannon entropy, one defines the *conditional guessing entropy* $G(X|Y)$ as

$$G(X|Y) = \sum_{y \in \mathcal{Y}} p_Y(y) G(X|Y = y) \ .$$

$G(X|Y)$ represents the expected number of optimal guesses needed to determine X when the value of Y is already known. Hence, $G(U|V_{\mathfrak{a}})$ is a lower bound on the expected number of off-line guesses that an attacker must perform for key recovery after having carried out a side-channel attack with strategy $\mathfrak{a}$.

Marginal Guesswork.

For a fixed $\alpha \in [0, 1]$, the marginal guesswork of a random variable X quantifies the number of questions of the kind "does $X = x$ hold" that must be asked until X's value is correctly determined with a chance of success given by α [30]. Again, w.l.o.g. let $\mathcal{X}$ be indexed such that $p_X(x_i) \geq p_X(x_j)$, whenever $i \leq j$. Then the (α)-*marginal guesswork* of X is defined as

$$W_\alpha(X) = \min\{j \mid \sum_{1 \leq i \leq j} p_X(x_i) \geq \alpha\} \ .$$

We define the *conditional marginal guesswork* $W_\alpha(X|Y)$ analogously to the conditional entropy. As before, $W_\alpha(U|V_{\mathfrak{a}})$ is a lower bound on the expected number of guesses that an attacker needs to perform in order to determine the secret with a success probability of more than α after having carried out a side-channel attack with strategy $\mathfrak{a}$.

Uniform Distributions.

If p is uniformly distributed, one can derive simple explicit formulae for the entropy measures presented so far.

PROPOSITION 2. *Let $\mathfrak{a}$ be an attack strategy with $P_{\mathfrak{a}} = \{B_1, \ldots, B_r\}$, $|B_i| = n_i$, and $|K| = n$. If p is uniformly distributed, then*

1. $H(U|V_{\mathfrak{a}}) = \frac{1}{n} \sum_{i=1}^{r} n_i \log n_i$,
2. $G(U|V_{\mathfrak{a}}) = \frac{1}{2n} \sum_{i=1}^{r} n_i^2 + \frac{1}{2}$, *and*
3. $W_\alpha(U|V_{\mathfrak{a}}) = \frac{1}{n} \sum_{i=1}^{r} n_i \lceil \alpha n_i \rceil$.

PROOF. For the proof of 2.2, observe that $G(U|V_{\mathfrak{a}}) = \sum_{i=1}^{r} \frac{n_i}{n} \sum_{j=1}^{n_i} j \frac{1}{n_i} = \frac{1}{n} \sum_{i=1}^{r} \frac{(n_i+1)n_i}{2}$. We conclude that $G(U|V_{\mathfrak{a}}) = \frac{1}{2n} \sum_{i=1}^{r} n_i^2 + \frac{1}{2}$. The other cases are similarly straightforward. □

While there are clear connections between the entropy measures in the uniform case, there is no general relationship between them for arbitrary probability distributions. Massey [22] shows that one can give lower bounds for $G(X)$ in terms of $H(X)$, but that there are no general upper bounds for $G(X)$ in terms of $H(X)$. Pliam [30] shows that there can be no general inequality between marginal guesswork and Shannon entropy.

Worst Case Entropy Measures.

All entropy measures presented so far are average case measures. We use the example of guessing entropy to illustrate this and to show how they can be adapted to accommodate stronger, worst case guarantees.

The conditional guessing entropy $G(U|V_{\mathfrak{a}})$ weights each value $G(U|V_{\mathfrak{a}} = B)$ by the probability that a randomly chosen key from K is contained in $B \in P_{\mathfrak{a}}$. As $G(U|V_{\mathfrak{a}} = B)$ measures the difficulty of guessing a key if its enclosing block B is known, $G(U|V_{\mathfrak{a}})$ quantifies whether keys are, on the average, hard to guess after an attack with strategy $\mathfrak{a}$.

Our model also accommodates entropy measures for a worst case analysis, in the sense that they quantify the guessing effort for the keys in K that are easiest to guess. To capture this, we define the *minimal guessing entropy* $\hat{G}(U|V_{\mathfrak{a}})$ of U given $V_{\mathfrak{a}}$ as $\hat{G}(U|V_{\mathfrak{a}}) = \min\{G(U|V_{\mathfrak{a}} = B) \mid B \in P_{\mathfrak{a}}\}$. The value $\hat{G}(U|V_{\mathfrak{a}})$ is a lower bound on the expected guessing effort for the weakest keys in K.

The following example illustrates the difference between worst case and average case entropy measures.

Example 5. Consider a set of uniformly distributed keys $K = \{1, \ldots, 10\}$ and the partitions $P = \{\{1\}, \{2, \ldots, 10\}\}$ and $Q = \{\{1\}, \ldots, \{10\}\}$. We have $\hat{G}(U|V_P) = 1$, which reflects that there exists a key that is trivial to guess with knowledge of its enclosing block in P. The conditional guessing entropy yields $G(U|V_P) = 4.6$ which reflects that, on the average, 4.6 guesses are still necessary for key recovery. Note that $\hat{G}(U|V_P) = \hat{G}(U|V_Q)$ and that $G(U|V_Q) = 1 < G(U|V_P)$. That is, only the average case measure can distinguish between the partitions P and Q.

Ultimately, it will depend on the application whether worst case or average case measures are appropriate. For the remainder of this paper, we will focus solely on average case measures, as they are better suited for distinguishing between partitions. All of our technical results, however, carry over to the worst case versions with only minor modifications.

Given entropy measures for evaluating attack strategies, we can now define attack optimality and give bounds for what an attacker can, in principle, achieve by performing a side-channel attack.

3.2 Measuring the Resistance to Optimal Attacks

There is a trade-off between the number of attack steps taken and the attacker's uncertainty about the key. More side-channel measurements imply less uncertainty, which entails fewer guesses. In the following, we give a formal account of this for the entropy measures introduced. We then define a function $\Phi_{\mathcal{E}}$ that is parameterized by an entropy measure $\mathcal{E} \in \{H, G, W_\alpha\}$ and whose value is the expected remaining uncertainty about the key after n steps of an optimal attack strategy. As we will show, $\Phi_{\mathcal{E}}$ can be used for assessing an implementation's vulnerability to side-channel attacks.

When assessing the vulnerability of an implementation to active side-channel attacks, we make the worst case assumption that the attacker proceeds optimally. A strategy is optimal if an attacker who follows it can expect to have less uncertainty about the key than with any other strategy of the same length.

Definition 4. Let $\mathfrak{a} = (T, r, L)$ be an attack strategy of length l against a set of partitions $\mathcal{P}$ of K. We call $\mathfrak{a}$ *optimal with respect to* $\mathcal{E} \in \{H, G, W_\alpha\}$ iff $\mathcal{E}(U|V_\mathfrak{a}) \leq \mathcal{E}(U|V_\mathfrak{b})$ holds for all attack strategies $\mathfrak{b}$ against $\mathcal{P}$ of length l.

Next, we define the expected remaining uncertainty as a function of the number of attack steps taken by an optimal attacker. In this way, we relate two important aspects of a system's vulnerability. Namely, how much information can an attacker obtain and how many queries he needs for this.

Definition 5. Let $\mathcal{P}$ be a set of partitions of K and let $\mathcal{E} \in \{H, G, W_\alpha\}$. We define the *resistance* $\Phi_\mathcal{E}$ to an attack against $\mathcal{P}$ by

$$\Phi_\mathcal{E}(n) = \mathcal{E}(U|V_\mathfrak{a}) \ ,$$

where $\mathfrak{a}$ is an optimal attack of length n with respect to $\mathcal{E}$.

We now formally justify the intuition that more attack steps lead to less uncertainty about the key. In particular, we prove that $\Phi_\mathcal{E}$ decreases monotonously. As notation, we say that an attack strategy $\mathfrak{a} = (T, r, L)$ is the *prefix* of an attack strategy $\mathfrak{b} = (T', r', L')$ if T is a subtree of T', $r = r'$, and if L and L' coincide on T. We denote this by $\mathfrak{a} \leq \mathfrak{b}$.

PROPOSITION 3. *Let $\mathcal{E} \in \{H, G, W_\alpha\}$ be an entropy measure and let $\mathfrak{a}$ and $\mathfrak{b}$ be attack strategies.*

1. *$\mathfrak{a} \leq \mathfrak{b}$ implies $\mathcal{E}(U|V_\mathfrak{a}) \geq \mathcal{E}(U|V_\mathfrak{b})$.*
2. *For all $n \in \mathbb{N}$, we have $\Phi_\mathcal{E}(n) \geq \Phi_\mathcal{E}(n+1)$.*

PROOF. We prove 3.1 for the case of the guessing entropy G. Consider a partition P of K. It is easy to see that $G(U|V_P) = \sum_{B \in P} \sum_{i=1}^{|B|} i \ p_U(x_i^B)$, where the elements x^B of block B are indexed in order of their decreasing probabilities. Observe that the probabilities in the sum do not depend on P, but that the indices of the elements decrease as P is refined. As $\mathfrak{a} \leq \mathfrak{b}$ implies $P_\mathfrak{a} \sqsupseteq P_\mathfrak{b}$, 3.1 follows. Assertion 3.2 is a simple consequence of 3.1. □

4. AUTOMATED VULNERABILITY ANALYSIS

In the following, we first show that $\Phi_\mathcal{E}$ is computable for $\mathcal{E} \in \{H, G, W_\alpha\}$ and we give algorithms and complexity bounds. The bounds are exponential and render direct computation infeasible. We then present a greedy heuristic for approximating $\Phi_\mathcal{E}$ to address this problem.

Throughout this section, let $\mathcal{P}$ be a set of partitions of K and let $r \geq 2$ be the maximum number of blocks of a partition in $\mathcal{P}$, i.e., $r = \max\{|P| \mid P \in \mathcal{P}\}$. We assume that partitions are represented using standard disjoint-set data structures with operations UNION and FIND (see, e.g., [13]). Furthermore, we assume that O and K are ordered sets for which two elements can be compared in $\mathcal{O}(1)$. It is not difficult to see that, given a function $f : K \times M \to O$, one can build disjoint-set data structures for $\mathcal{P}_f$ in time $\mathcal{O}(|M|\,|K| \log |K|)$, under the assumption that f can be computed in time $\mathcal{O}(1)$.

4.1 Computing $\Phi_\mathcal{E}$

We begin by establishing an upper bound on the number of attack strategies of a given length; we will use this later when we compute $\Phi_\mathcal{E}$ by enumerating strategies.

LEMMA 1. *The number of attack strategies of length n against $\mathcal{P}$ is bounded from above by $|M|^{\frac{r^n-1}{r-1}}$. Furthermore, every attack strategy of length n can be encoded by an r^n-tuple over $\{1, \ldots, |M|\}$.*

PROOF. A straightforward inductive argument shows that the partition induced by an attack strategy of length n has at most r^n blocks. We prove the claimed bound by induction on n. For $n = 0$, the bound is clearly valid. Assume now that there are at most $|M|^{\frac{r^n-1}{r-1}}$ attack strategies of length n. Each such attack strategy can be extended to an attack strategy of length $n+1$ by assigning one of the $|M|$ partitions to every block of the induced partition. There are at most r^n blocks, so there are at most $|M|^{r^n}$ possible extensions. In total, there are at most $|M|^{\frac{r^n-1}{r-1}} \cdot |M|^{r^n} = |M|^{\frac{r^{n+1}-1}{r-1}}$ attack strategies of length $n+1$, which concludes our inductive proof. Now observe that the choices of partitions at level j can be encoded by a r^j-tuple $(i_{j,1}, \ldots, i_{j,r^j})$ over $\{1, \ldots, |M|\}$. As $\sum_{j=0}^{n-1} r^j = \frac{r^n-1}{r-1} \leq r^n$, the entire strategy can be encoded by a r^n-tuple. □

Computing $\Phi_\mathcal{E}(n)$ requires identifying an optimal attack of length n. We may compute $\Phi_\mathcal{E}(n)$ directly by brute force: enumerate all attack strategies and compute $\mathcal{E}$ for each induced partition. This algorithm yields an upper bound for the complexity of computing $\Phi_\mathcal{E}$.

THEOREM 1. *The value $\Phi_\mathcal{E}(n)$ can be computed in time*

$$\mathcal{O}(n\,|M|^{r^n}|K|\,\log|K|)$$

under the assumption that $\mathcal{E}$ can be computed in time $\mathcal{O}(|K|)$.

PROOF. Let $(i_0; \ldots; i_{n-1,1}, \ldots, i_{n-1,r^{n-1}})$, with $1 \leq i_j \leq |M|$, represent an attack strategy $\mathfrak{a}$ of length n, where the choices of partitions at each level are encoded as in the proof of Lemma 1 and where the individual levels are separated by ";". Iterate over all $k \in K$. For each k, call FIND(k) on the representation of partition i_0 to obtain the index j of k's enclosing block in P_{i_0}. Use FIND(k) to obtain k's block in $P_{i_{1,j}}$. Repeat this procedure until k's block in the partition at depth n is determined. Save these n block indices in a list and store it in an array I at index k. Performing this procedure for all $k \in K$ has time complexity $\mathcal{O}(n\,|K| \log |K|)$. Two keys are in the same block of the partition induced by $\mathfrak{a}$ if and only if their corresponding index lists coincide. To obtain the equivalence classes, sort K according to the lexicographic order given by the lists in I in $\mathcal{O}(n\,|K| \log |K|)$, which dominates the running time for evaluating $\mathcal{E}$ on the resulting partition. Performing this procedure for all attack strategies yields an overall running time of $\mathcal{O}(n\,|M|^{r^n}|K|\,\log|K|)$. □

4.2 Approximating $\Phi_\mathcal{E}$

Brute-force computation of $\Phi_\mathcal{E}$ requires time doubly exponential in the number of attack steps and is hence infeasible even for small parameter sizes. To address this problem, we present a more efficient greedy heuristic and describe properties that help us approximate $\Phi_\mathcal{E}$.

A Greedy Heuristic.

Consider an attacker who has performed a number of attack steps against a set of partitions $\mathcal{P}$ and has narrowed down the set of possible keys to a subset $A \subseteq K$. A greedy choice for the subsequent query is a partition $P \in \mathcal{P}$ that minimizes the remaining entropy of $A \cap P$. To formalize this, consider the random variable $U_A = id_A$ that models the random choice of a key according to the conditional probability distribution $p(\cdot|A)$, and the random variable $V_{P\cap A} : A \to P \cap A$ that models the choice of the enclosing block in $P \cap A$.

Definition 6. An attack strategy $\mathfrak{a} = (T, r, L)$ against $\mathcal{P}$, with $T = (V, E)$, is *greedy with respect to* $\mathcal{E} \in \{H, G, W_\alpha\}$ iff for every $v \in V$ and all $P, Q \in \mathcal{P}$, $\{L(w) \mid w \in succ(v)\} = L(v) \cap P$ implies $\mathcal{E}(U_A|V_{P\cap A}) \leq \mathcal{E}(U_A|V_{Q\cap A})$.

We next define an approximation $\hat{\Phi}_\mathcal{E}$ of $\Phi_\mathcal{E}$ based on the partition induced by a greedy strategy. Note that greedy strategies are not unique and that the induced partitions of two greedy strategies of the same length need not even have the same entropy. Hence to define an approximation $\hat{\Phi}_\mathcal{E}$ we assume a fixed greedy strategy $\mathfrak{a}$ of sufficient length l whose underlying tree is full. For all $n \leq l$, we denote the full prefix of $\mathfrak{a}$ with length n by $\mathfrak{a}(n)$. We define $\hat{\Phi}^\mathfrak{a}_\mathcal{E}$ as $\hat{\Phi}^\mathfrak{a}_\mathcal{E}(n) = \mathcal{E}(U|V_{\mathfrak{a}(n)})$, for all $n \leq l$. We only use $\mathfrak{a}$ as an artifact to consistently resolve the nondeterminism of greedy strategies of different lengths. From now on, we assume that a greedy strategy $\mathfrak{a}$ of sufficient length is fixed and write $\hat{\Phi}_\mathcal{E}$ instead of $\hat{\Phi}^\mathfrak{a}_\mathcal{E}$.

THEOREM 2. *The value $\hat{\Phi}_\mathcal{E}(n)$ can be computed in time*

$$\mathcal{O}(n\, r\, |M|\, |K|^2)\ ,$$

under the assumption that $\mathcal{E}$ can be computed in time $\mathcal{O}(|K|)$.

PROOF. For computing intersections of partitions, we assume a list representation of the blocks of every partition, in which every list is ordered with respect to the order on K. This can be extracted from the given disjoint-set data structures in time $\mathcal{O}(|M|\, |K|^2)$. For a fixed subset of K that is represented as an ordered list, a greedy refinement can then be computed by intersecting it with each of the (at most r) blocks of each of the $|M|$ partitions. As the set representations are ordered, this can be done in time $\mathcal{O}(r\, |M|\, |K|)$. As the number of blocks in every partition of K is bounded by $|K|$, computing n greedy steps can be done in time $\mathcal{O}(n\, r\, |M|\, |K|^2)$. □

We state next several inequalities between the values of $\Phi_\mathcal{E}$ and $\hat{\Phi}_\mathcal{E}$, which we will later use when interpreting our experimental results.

Relating $\Phi_\mathcal{E}$ and $\hat{\Phi}_\mathcal{E}$.

The definition of a greedy strategy begs the question of whether greedy strategies are also optimal. The following example illustrates that this is not the case in general.

Example 6. Consider the set of partitions $\mathcal{P} = \{\{\{1\}, \{2\}, \{3, 4, 5\}\}, \{\{1\}, \{2, 3, 4\}, \{5\}\}, \{\{1, 2, 3\}, \{4, 5\}\}\}$, a uniform distribution, and the guessing entropy as a measure. A greedy strategy refines K to $\{\{1\}, \{2\}, \{3, 4, 5\}\}$ in a first step, and to $\{\{1\}, \{2\}, \{3, 4\}, \{5\}\}$ in a second step. Optimally, however, one would first pick $\{\{1, 2, 3\}, \{4, 5\}\}$ and refine it to $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$ in a second step.

```
greedy :: [Part k] -> Int -> [k] -> Part k
greedy f n keys = app n (greedystep f) [keys]

greedystep :: [Part k] -> Part k -> Part k
greedystep f pt = concat (map refine pt)
  where refine b = minimumBy order (restrict b f)
```

Figure 2: Computing $\hat{\Phi}_\mathcal{E}$ in Haskell

Although Example 6 implies that $\hat{\Phi}_\mathcal{E}$ and $\Phi_\mathcal{E}$ do not coincide in general, we can establish the following relationships.

PROPOSITION 4. *For $\mathcal{E} \in \{H, G, W_\alpha\}$, we have*

1. $\hat{\Phi}_\mathcal{E}(1) = \Phi_\mathcal{E}(1)$,
2. *for all $n \in \mathbb{N}$, $\hat{\Phi}_\mathcal{E}(n) \geq \Phi_\mathcal{E}(n)$, and*
3. *if $\hat{\Phi}_\mathcal{E}(n) = \hat{\Phi}_\mathcal{E}(n+1)$, then we have $\Phi_\mathcal{E}(n') = \hat{\Phi}_\mathcal{E}(n') = \hat{\Phi}_\mathcal{E}(n)$, for all $n' \geq n$.*

PROOF. Assertions 1 and 2 follow directly from Definitions 4 and 6. For Assertion 3, let $\mathfrak{a}$ be the greedy strategy underlying the definition of $\hat{\Phi}_\mathcal{E}$. $\hat{\Phi}_\mathcal{E}(n) = \hat{\Phi}_\mathcal{E}(n+1)$ implies that $P_{\mathfrak{a}(n)}$ cannot be refined by intersection with a partition from $\mathcal{P}$, hence $P_{\mathfrak{a}(n)} = \bigcap_{P\in\mathcal{P}} P$, which refines every partition that can be induced by intersection of elements from $\mathcal{P}$. □

We will make use of Proposition 4 in our experiments. 4.2 shows that an implementation that is shown to be vulnerable when analyzed with $\hat{\Phi}_\mathcal{E}$ must also be vulnerable with respect to $\Phi_\mathcal{E}$. 4.3 implies that if $\hat{\Phi}_\mathcal{E}$ levels off, then so does $\Phi_\mathcal{E}$, and their values coincide. Hence we do not need to compute $\Phi_\mathcal{E}$ for arguments beyond this point.

4.3 An Implementation

For our experiments we have implemented $\hat{\Phi}_\mathcal{E}$ in HASKELL [5]. We have chosen simplicity over efficiency, forgoing sophisticated data structures and optimizations. Instead, we represent sets as lists and partitions as lists of lists and recursively compute greedy refinements of partitions. The core routines are given in Figure 2.

The function `greedy` takes as arguments a list of keys, a list of partitions `f` of the list `keys`, and an integer `n`. It refines the trivial partition `[keys]` by `n`-fold application of a greedy refinement step through `app`. The refinement step is implemented in `greedystep`, where each partition `pt` is refined by greedily refining each individual block. This is done in `refine`, which maps each block to its partition with minimal rank among those obtained by restricting the elements of `f` to `b` with `restrict`. The rank of a partition is given by the function `order`, which can be instantiated to $\mathcal{E} \in \{H, G, W_\alpha\}$. Applying `order` to the result of `greedy` yields $\hat{\Phi}_\mathcal{E}$. The simplicity of this implementation shows that the automation of our techniques is indeed straightforward.

5. EXPERIMENTS

In this section, we report on case studies analyzing implementations of different cryptographic algorithms with respect to their vulnerability to timing and power attacks. We focus on implementations in synchronous hardware as, in this setting, time and power consumption are relatively easy to determine.

As examples, we analyze the timing behavior of circuits for multiplying integers and for exponentiation in finite fields $\mathbb{F}_{2^w}$. We also analyze the power consumption of a (constant-time) circuit for multiplication in $\mathbb{F}_{2^w}$. Exponentiation and multiplication over $\mathbb{F}_{2^w}$ are relevant, for example, in the generalized ElGamal encryption scheme, where decryption consists of exponentiation followed by multiplication [23].

In the remainder of this section, we use the guessing entropy G as a measure of uncertainty and we abbreviate Φ_G by Φ and $\hat{\Phi}_G$ by $\hat{\Phi}$, respectively. We assume a uniform probability distribution in our experiments and compute the remaining uncertainty with the formula given in Proposition 2.2.

5.1 Realization

Goals and Limitations.

Our goal is to compute bounds on the information that realistic implementations may leak to active side-channel attackers.

Computing Φ using the algorithm from Theorem 1 is expensive. The time required is doubly exponential in the number of attack steps, and the sizes of the keyspace and the message space are exponential in the number of bits used to represent keys and messages, respectively. Hence, we cannot feasibly compute Φ for large parameter sizes.

We use two approximation techniques to address this problem.

1. We approximate Φ by $\hat{\Phi}$. We will see that $\hat{\Phi}$ matches Φ on our example data, although this does not hold in general (see Example 6).

2. We parameterize each algorithm by the bit-width w of its operands. Our working assumption is that regularity in the values of Φ for $w \in \{2, \ldots, w_{\max}\}$ reflects the structural similarity of the parameterized algorithms. This allows us to extrapolate to values of w beyond $w_{\max}$. To make this explicit, we will write Φ^w to denote that Φ is computed on w-bit operands.

Using both techniques, we can estimate $\Phi^w(n)$ for values of w and n for which direct computation is infeasible.

Time and Power Estimation with Gezel.

We use the hardware description language Gezel [33] to describe and simulate circuits. Synchronous circuits are modeled in Gezel as automata, where one transition corresponds to one clock cycle. The Gezel environment comes with a compiler that maps circuit descriptions into a synthesizeable subset of Vhdl. The translation is *cycle-true* in that it preserves the circuit's timing behavior within the granularity of clock cycles. In this way, the timing-guarantees obtained by formal analysis translate to silicon implementations.

Precisely estimating a circuit's power consumption is not possible at this level of abstraction as it depends on the physics of the semiconductor technology used. One needs to employ technology-dependent power models for accurate predictions during simulation. In this paper, we take a simple, technology-independent approach that is provided by the Gezel environment to approximate a circuit's power consumption: we count the number of bit transitions during each cycle. The rationale behind this is that, e.g., in CMOS technology, the power dissipation of a signal that remains constant is negligible compared to a signal that changes. Counting the number of bit transitions thus provides approximate information about the actual power consumption and we will use it for our analysis. It is straightforward to replace this simple measure with those given by more realistic models. In this way, the precision of our analysis is only bounded by the precision of the available power models.

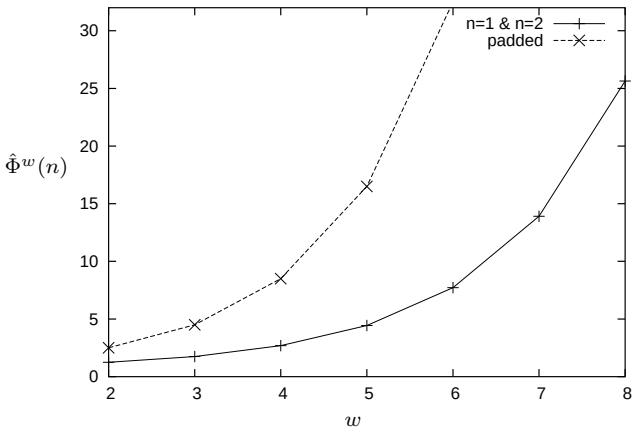

Figure 3: Integer Multiplication

Setup.

For each algorithm and each bit-width $w \in \{2, \ldots, 8\}$, we use the Gezel simulator to build value tables for the side-channel $f : \{0,1\}^w \times \{0,1\}^w \to O$. For timing analysis, we use $O = \mathbb{N}$ to represent the number of clock ticks until termination. For power analysis, we use $O = \mathbb{N}^d$ to represent the toggle count in each of the d clock cycles.

5.2 Results

In this section, we present our experimental results and discuss their implications.

Timing Attacks against Integer Multiplication.

We represent a natural number $k < 2^w$ as a sequence of w bits k_i, with $k = \Sigma_{i=0}^{w-1} k_i 2^i$. To multiply two natural numbers m and k, the product $m \cdot \Sigma_{i=0}^{w-1} k_i 2^i$ can be expanded to $(\ldots((k_{w-1} \cdot m) \cdot 2 + k_{w-2} \cdot m) \cdot 2 + \ldots) \cdot 2 + k_0 \cdot m$, which can easily be turned into an algorithm. Starting with $p = 0$, one iterates over all the bits of k, beginning with the most significant bit. If $k_i = 1$, one updates p by adding m and then doubling p's value. Alternatively, if $k_i = 0$, one updates p by just doubling its value. At the end of the loop, $p = m \cdot k$. In our implementation, the doubling and addition operations each take one clock cycle. Hence, the running time reflects the number of 1-bits in k, that is, k's Hamming weight. For illustration purposes, we use k as the key and m as the message. For the interpretation of Figure 3, first observe that $\hat{\Phi}^w(1) = \hat{\Phi}^w(2)$ holds. Hence, by Proposition 4, the graph actually depicts Φ^w.

There are two conclusions to be drawn from Figure 3. First, the circuit's timing behavior depends on the number of 1-bits in the key. This leads to the hypothesis that the Hamming weight of the key is revealed or, equivalently, that two keys are indistinguishable iff they have the same Hamming weight. The equivalence class of w-bit arguments with

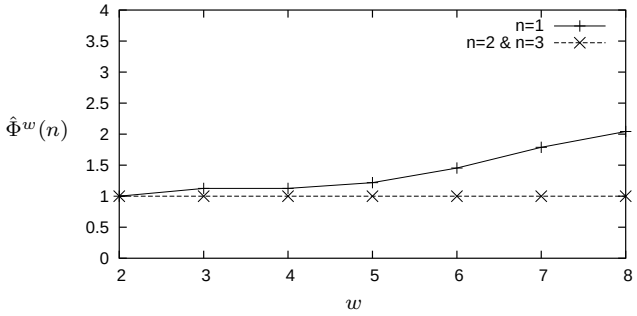

Figure 4: Finite-Field Exponentiation

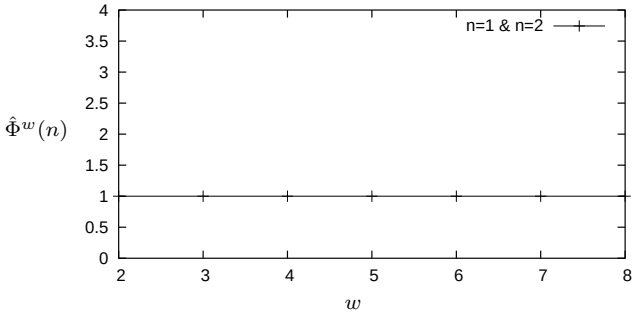

Figure 5: Finite-Field Multiplication

Hamming weight l has precisely $\binom{w}{l}$ elements. Hence, by Proposition 2, the conditional guessing entropy for the corresponding partition is given by $\frac{1}{2^{w+1}} \sum_{l=0}^{w} \binom{w}{l}^2 + \frac{1}{2}$. The values computed using this expression match the solid curve in Figure 3, which supports our hypothesis and confirms a result from [20].

Second, Figure 3 shows that a single side-channel measurement is enough to extract the maximal information revealed by the circuit's timing behavior. This follows as $\Phi^w(1)$ and $\Phi^w(2)$ coincide and is due to the fact that the circuit's running time is independent of the message. It is out of the scope of information-flow analysis, as in [20], to reason about the number of measurements needed to obtain such information.

We have also implemented and analyzed a variant of the integer multiplication algorithm described above, where we introduced a dummy computation step whenever no addition operation takes place. In this way, the algorithm's timing behavior does not leak any information about the input parameters. This is reflected by the dashed line in Figure 3, which matches the guessing entropy for a key without side-channel information, given by $0.5(2^w + 1)$.

Timing Attacks against Exponentiation in $\mathbb{F}_{2^w}$.

We analyzed a GEZEL implementation of the finite-field exponentiation algorithm from [14]. It takes two arguments, a basis m and an exponent k, and it computes m^k in $\mathbb{F}_{2^w}$. The algorithm is based on similar expansions as the integer multiplication algorithm in the previous example, but is more complex due to nested loops. To interpret Figure 4, observe that $\Phi^w(1) = \hat{\Phi}^w(1)$ and $\hat{\Phi}^w \geq \Phi^w$ follow from Proposition 4. We conclude that one timing measurement reveals a quantity of information larger than that contained in the Hamming weight, but that it does not completely determine the key. A second measurement, however, can reveal all remaining key information.

Power Attacks against Multiplication in $\mathbb{F}_{2^w}$.

We analyzed the power leakage of the finite-field multiplication circuit from the GEZEL package. It runs in constant time and we analyzed the power traces given by counting bit transitions, as previously explained. As in the case of integer multiplication, we chose one operand to be secret and one to be public. Figure 5 shows that the entire secret parameter is determined by one power trace. A silicon implementation with similar power consumption will hence be vulnerable to power attacks.

Scaling-Up.

The algorithms presented in Section 4 rely on the complete enumeration of the keyspace and therefore do not scale.[2] However, our data exhibits regularity and we can successfully extrapolate to larger bit-widths. Under our working assumption that this regularity reflects the structural similarity of the parameterized algorithms, we conclude that the interpretations given for each algorithm hold regardless of the implementation's bit-width.

In all three examples, the number of attack steps performed is surprisingly low compared to the sample size used in many published attacks, e.g., [6, 8, 18, 19]. This is due to the fact that noise is typically dealt with by increasing the number of measurements made. Template attacks [10] use noise models to extract the maximum information from every measurement and they demonstrate that key recovery from only a few measurements is indeed possible.

6. RELATED WORK

There has been substantial work in information-flow security on detecting or quantifying information leaks, but the results are only partially applicable to the problem of analyzing how much information an adaptive side-channel attacker can extract from a system. Early approaches focus on quantifying the capacity of covert channels between processes in multi-user systems [26, 39, 16]. The models predate the first published side-channel attack against cryptography [18] and are so general that it is unclear whether and how they could be instantiated to address the problem at hand. Di Pierro et al. [29] show how to quantify the number of statistical tests an observer needs to perform to distinguish two processes in a probabilistic concurrent language. Lowe [21] quantifies information flow in a possibilistic process algebra by counting the number of distinguishable behaviors. Clarkson et al. [12] develop a model for reasoning about an adaptive attacker's beliefs about the secret, which may also be wrong.

The information measures proposed by Clark et al. [11] are closest to ours. The authors relate observational equivalence to random variables and use Shannon entropy to quantify the information flow. However, their measure captures the information gain of a passive observer instead of an active attacker: the public input to the system is chosen

[2]The computation of $\hat{\Phi}^8(2)$ for finite-field exponentiation took 40 minutes on a 2.4 GHz machine with 3 gigabytes of RAM.

with respect to a probability distribution and is not under the attacker's control.

Several approaches in language-based security use security type systems to detect timing side-channels in both sequential and multithreaded settings, see [1, 2, 17] and [36, 32], respectively. A successful type check implies that an attacker cannot gain any information about the secret, even if he exhaustively queries the system. However, such strong guarantees are of unclear significance in the absence of realistic timing models for high-level languages. Information-flow analyses at the hardware level [38, 20] are based on more realistic assumptions about the system, but do not model adaptive attackers.

There is a large body of work on side-channel cryptanalysis, in particular on attacks and countermeasures. However, models and theoretical bounds on what side-channel attackers can achieve are only now emerging. Chari et al. [9] are the first to present methods for proving hardware implementations secure. They propose a generic countermeasure for power attacks and prove that it resists a given number of side-channel measurements. Schindler et al. [34] propose a stochastic model for improving the efficiency of key extraction. However, they do not give bounds on what can, in principle, be achieved by their techniques. Micali et al. [25] propose physically observable cryptography, a mathematical model that aims at providing provably secure cryptography on hardware that is only partially shielded. Their model has recently been specialized by Standaert et al. [37], who show how assumptions on the computational capabilities of an attacker can be combined with leakage functions that measure the information that is revealed by the system's side-channels. Our model could be used to solve the open problem of instantiating these leakage functions. A detailed investigation of this is the subject of future work.

7. CONCLUSIONS AND FUTURE WORK

We have presented a quantitative model for reasoning about adaptive side-channel attacks. It allows us to express an attacker's remaining uncertainty about a secret as a function of the number of his side-channel measurements. This function provides a relevant metric for assessing a system's vulnerability to side-channel attacks.

On the theoretical side, our model of adaptive attacks provides a connection between information-theoretic notions of security and physical models of hardware. Its simplicity is reflected in the three line program (see Section 4.3) that implements this connection. On the practical side, we have applied our model to automatically derive meaningful assertions about the resistance of hardware implementations to adaptive side-channel attacks.

As ongoing work, we are extending our model with statistical techniques for entropy estimation [3, 4]. This allows us to approximate Φ for larger bit-widths. Our initial experiments are encouraging: we are able to confirm that the presented integer multiplication algorithm reveals one operand's Hamming weight—for implementations with 100 bits per operand and with an error of less than 1%. However, the existing confidence intervals for this estimation are too large for practical use and, as future work, we hope to improve them.

8. REFERENCES

[1] J. Agat. Transforming out Timing Leaks. In *Proc. POPL '00*, pages 40–53. ACM.

[2] G. Barthe, T. Rezk, and M. Warnier. Preventing Timing Leaks Through Transactional Branching Instructions. In *Proc. QAPL '05*, ENTCS, pages 33–55. Elsevier.

[3] G. Basharin. On a Statistical Estimate for the Entropy of a Sequence of Independent Random Variables. *Theory Probab. Appl.*, 47:333–336, 1959.

[4] T. Batu, S. Dasgupta, R. Kumar, and R. Rubinfeld. The complexity of approximating entropy. In *Proc. STOC '02*, pages 678–687. ACM, 2002.

[5] R. Bird. *Introduction to Functional Programming using Haskell*. Prentice Hall, second edition, 1998.

[6] D. Boneh and D. Brumley. Remote Timing Attacks are Practical. In *Proc. USENIX Security Symposium '03*.

[7] C. Cachin. Entropy Measures and Unconditional Security in Cryptography. PhD thesis, ETH Zürich, 1997.

[8] J. Cathalo, F. Koeune, and J.-J. Quisquater. A New Type of Timing Attack: Application to GPS. In *Proc. CARDIS '03*, LNCS 2779, pages 291–303. Springer.

[9] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Proc. CRYPTO '99*, LNCS 1666, pages 398–412. Springer.

[10] S. Chari, J. R. Rao, and P. Rohatgi. Template Attacks. In *Proc. CHES '02*, LNCS 2523, pages 13–28. Springer.

[11] D. Clark, S. Hunt, and P. Malacaria. Quantitative Information Flow, Relations and Polymorphic Types. *J. Log. Comput.*, 18(2):181–199, 2005.

[12] M. Clarkson, A. Myers, and F. Schneider. Belief in Information Flow. In *Proc. CSFW '05*, pages 31– 45. IEEE.

[13] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, second edition, 2001.

[14] M. Davio, J. P. Deschamps, and A. Thayse. *Digital Systems with Algorithm Implementation*. John Wiley & Sons, Inc., 1983.

[15] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *Proc. CHES '01*, LNCS 2162, pages 251–261. Springer.

[16] J. W. Gray. Toward a Mathematical Foundation for Information Flow Security. *JCS*, 1(3-4):255–294, 1992.

[17] D. Hedin and D. Sands. Timing Aware Information Flow Security for a JavaCard-like Bytecode. In *BYTECODE '05*, ENTCS. Elsevier.

[18] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proc. CRYPTO '96*, LNCS 1109, pages 104–113. Springer.

[19] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proc. CRYPTO '99*, LNCS 1666, pages 388–397. Springer.

[20] B. Köpf and D. Basin. Timing-Sensitive Information Flow Analysis for Synchronous Systems. In *Proc. ESORICS '06*, LNCS 4189, pages 243–262. Springer.

[21] G. Lowe. Quantifying Information Flow. In *Proc. CSFW '02*, pages 18–31. IEEE.

[22] J. L. Massey. Guessing and Entropy. In *Proc. IEEE Int. Symp. on Info. Th. '94*, page 204. IEEE.

[23] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[24] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Power Analysis Attacks of Modular Exponentiation in Smartcards. In *Proc. CHES '99*, LNCS 1717, pages 144–157. Springer.

[25] S. Micali and L. Reyzin. Physically Observable Cryptography (Extended Abstract). In *Proc. TCC '04*, LNCS 2951, pages 278–296. Springer.

[26] J. K. Millen. Covert Channel Capacity. In *Proc. IEEE Symp. on Security and Privacy '87*, pages 60–66. IEEE.

[27] F. N. Najm. A Survey of Power Estimation Techniques in VLSI Circuits. *IEEE Transactions on VLSI Systems*, 2(4):446–455, 1994.

[28] D. A. Osvik, A. Shamir, and E. Tromer. Cache Attacks and Countermeasures: the Case of AES. In *Proc. CT-RSA '06*, LNCS 3860, pages 1–20. Springer.

[29] A. D. Pierro, C. Hankin, and H. Wiklicky. Approximate Non-Interference. In *Proc. CSFW '02*, pages 3–17. IEEE.

[30] J. O. Pliam. On the Incomparability of Entropy and Marginal Guesswork in Brute-Force Attacks. In *Proc. INDOCRYPT '00*, LNCS 1977, pages 67–79. Springer.

[31] J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Couter-Measures for Smard Cards. In *Proc. E-smart '01*, LNCS 2140, pages 200–210. Springer.

[32] A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. CSFW '00*, pages 200–215. IEEE.

[33] P. Schaumont, D. Ching, and I. Verbauwhede. An Interactive Codesign Environment for Domain-Specific Coprocessors. *ACM Transactions on Design Automation for Electronic Systems*, 11(1):70–87, 2006.

[34] W. Schindler, K. Lemke, and C. Paar. A Stochastic Model for Differential Side-Channel Cryptanalysis. In *Proc. CHES '05*, LNCS 3659, pages 30–46. Springer.

[35] C. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[36] G. Smith and D. Volpano. Secure Information Flow in a Multi-Threaded Imperative Language. In *Proc. POPL '98*, pages 355–364. ACM.

[37] F.-X. Standaert, E.Peeters, C. Archambeau, and J.-J. Quisquater. Towards Security Limits in Side-Channel Attacks. In *Proc. CHES '06*, LNCS 4249, pages 30–45. Springer.

[38] T. Tolstrup. Language-based Security for VHDL. PhD thesis, Technical University of Denmark, 2007.

[39] J. Wittbold and D. Johnson. Information flow in nondeterministic systems. In *Proc. IEEE Symp. on Security and Privacy '90*, pages 144–161. IEEE.

[40] L. Zhong, S. Ravi, A. Raghunathan, and N. Jha. Power Estimation Techniques for Cycle-Accurate Functional Descriptions of Hardware. In *Proc. ICCAD '04*, pages 668– 675. ACM.