Automatically Deriving Information-theoretic Bounds for Adaptive Side-channel Attacks

Boris Köpf
MPI-SWS
Martin-Luther-Strasse 12
D-66111 Saarbrücken
Germany
bkoepf@mpi-sws.org

David Basin
ETH Zurich
Haldeneggsteig 4
CH-8092 Zürich
Switzerland
basin@inf.ethz.ch

Abstract

We present a model of adaptive attacks which we combine with information-theoretic metrics to quantify the information revealed to an adaptive adversary. This enables us to express an adversary's remaining uncertainty about a secret as a function of the number of interactions with the system under attack. We present algorithms and approximation methods for computing this function. The main application area for our approach is the analysis of side-channels in cryptographic algorithms and we give examples of how it can be used to characterize the vulnerability of hardware implementations to timing and power attacks. We also show the generality of our approach by using it to quantify the information leaked by a security protocol.

1 Introduction

Problem Statement 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 [11, 26], cache behavior [41], power consumption [27], and electromagnetic radiation [22, 44] 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 a number of alternative models for proving resistance against such attacks have been proposed [14, 37, 51].

Two factors determine whether an adversary can successfully mount a side-channel attack on a system and recover a secret key (or other secret data). First, the adversary must be able to gather sufficient information about the key from his side-channel measurements. Second, the adversary must have sufficient

computational resources to recover the key from this information. To prove that a system is resistant to side-channel attacks, one must therefore ensure that no realistic adversary fulfills both conditions.

In theory, the recovery of a secret may be computationally infeasible, even if the secret is completely revealed in an information-theoretic sense. For example, an RSA public key contains all the information about the corresponding private key, but it is considered infeasible to derive the private key from this information. For side-channel attacks, however, computational complexity is typically not a barrier: side-channels are properties of individual implementations and are, in general, objects of unknown mathematical structure. In particular, side-channels are typically not associated with any computational security guarantees. Indeed, past experience shows that keys can been effectively recovered from side-channel information from a broad range of cryptographic algorithms on different platforms, e.g., [11, 13, 15, 27, 41]. In the absence of computational security, the security of a system against side-channel attacks depends entirely on the amount of secret information that the side-channel reveals.

Intuitively, increasing the number of side-channel measurements yields more information about the secret key. Moreover, in many attack scenarios, the adversary can increase the information he obtains from his measurements by influencing the system's computations, for example, by adaptively choosing the ciphertexts that the system decrypts. However, the adversary's interactions with the system are often expensive or limited, and their number needs to be considered when reasoning about a system's vulnerability to side-channel attacks. For example, the system may bound the number of times it re-uses a secret, such as a session key, or it may refuse multiple interactions with the same agent.

These observations suggest characterizing the security of a system in terms of the quantity of secret information that an adaptive adversary can extract with a given number of side-channel measurements. This quantity is a worst-case bound for what a realistic, computationally bounded adversary can deduce about the key. Defining and computing this bound for a given system has been an open problem prior to this work.[1]

Approach In this paper, we propose a solution to this problem for deterministic systems. We develop a model that enables us to express the quantity of information that an adaptive adversary can extract from a system and we present an algorithm and approximation methods 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 adversary during attacks. We combine attack strategies with information-theoretic entropy measures; this enables us to express the adversary's expected uncertainty about the secret after he has performed an attack, following a given strategy. By quantifying over all attack strategies of a fixed length n, we express what adversaries can, in principle, achieve in n attack steps. We use this to define a function Φ that

[1] This paper extends and generalizes the results of [30].

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 adversary might apply to analyze the measurements.

We give an algorithm with double-exponential time complexity for computing Φ. Furthermore, we present two heuristic methods that reduce this complexity and thereby enable us to estimate a system's vulnerability for state-space sizes for which the direct computation of Φ is infeasible.

Our approach is parametric in the model of the system under attack. For analyzing side-channels, we use a deterministic model of the physical characteristics of the target implementation. For analyzing security protocols, we use a model of the computations performed by an agent in a given protocol step. Furthermore, our approach accommodates different entropy measures. As we will show, this enables us to derive both worst-case and average-case guarantees for the remaining guessing effort for recovering the key after an attack.

Finally, we have implemented our approach and we report on experimental results using our prototype. We have analyzed hardware implementations of cryptographic algorithms for their resistance to timing and power attacks, obtaining the following results.

1. An adversary 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 from the finite-field exponentiation algorithm of [20].
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 both for detecting possible side-channel attacks and for showing their absence.

We have also applied our model to analyze a simple protocol describing the interaction of applications with the API of a hardware security module. Our analysis answers an open question about the rate at which API error messages leak a secret PIN. This example illustrates that our approach can also be used to analyze information leaks beyond those implemented by physical side-channels.

Contributions Our main contributions are twofold. Theoretically, we develop a simple, but powerful, model for adaptive attacks that connects information-theoretic notions of security to models for the physical characteristics of computations. 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.

Outline 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 in Section 6 and draw conclusions in Section 7.

2 A Model of Adaptive Attacks

In this section, we present a formal model of adaptive side-channel attacks. The model formalizes the information that can by gained by an adversary who can measure different properties of the computation of cryptographic functions.

An example is an adversary who tries to deduce the secret key used by an implementation of a decryption algorithm. Here we would assume that the adversary may run the implementation on arbitrary ciphertexts and measure the time required for decryption. Hence, he can collect pairs of ciphertexts and corresponding execution times, which he can analyze to reconstruct the key, see e.g., [11, 26]. Moreover, his strategy for selecting ciphertexts may be adaptive in that the ciphertexts are chosen with respect to the results of previous measurements. Note that we are modeling the adversary's information gain from the side-channel information alone; the plaintext returned by the decryption algorithm plays no role here.

2.1 Adversaries and Side-Channels

Attack Scenario We consider systems that compute functions of type $F\colon K \times M \to D$, where K is a finite set of keys, M is a finite set of messages, and D is an arbitrary set. We assume that the input to F is provided by two different callers. One caller is an honest agent that provides a secret argument $k \in K$ and the other caller is a malicious agent (the *adversary*) that provides the argument $m \in M$. Examples of F are encryption and decryption functions and MACs.

We assume that the adversary 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 consumed or the time used by I_F during the computation, see [27, 36] and [11, 13, 26, 41], 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 gain information for deducing k or narrowing down the range of its possible values. Such an *attack* consists of a sequence of *attack steps*, each with two parts:

1. a *query* phase in which the adversary chooses a message m and sends it to the system, and

2. a *response* phase in which he observes I_F while the system computes $F(k, m)$.

The attack is *adaptive* if the adversary 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 adversary stops querying the system.

Modeling Side-Channels We assume that the adversary can make one side-channel observation per invocation of the function F and that this observation is determined by the inputs to F. Furthermore, we assume that the adversary has full knowledge about the implementation I_F.

Formally, 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 adversary. We will usually leave I_F implicit and abbreviate f_{I_F} by f. Modeling the adversary's observations as a function of the system's input corresponds to the assumption that the physical characteristics of I_F are deterministic and that any randomness in a physical adversary's measurements is input-independent noise. Knowledge of the values of f then corresponds to the assumption of error-free measurements and is a safe worst-case characterization of an adversary's measurement capabilities. This strong assumption is justified as our goal is to give bounds on what any side-channel adversary can, in principle, achieve.

Examples 1 and 2 below show how timing and power side-channels can be modeled, respectively. Example 3 illustrates how models of side-channels can be built by sampling.

Example 1. Suppose that F is implemented in synchronous (clocked) hardware and that the adversary 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 $f\colon K \times M \to \mathbb{N}$ that represents the number of clock ticks taken for the computation of F. A hardware simulation environment can be used to compute f. $\Diamond$

Example 2. Suppose F is given in a description language for synchronous hardware. Power estimation techniques such as [39, 55] can be used to derive a function $f\colon K \times M \to \mathbb{R}^n$ that estimates an implementation's power consumption during n clock ticks. $\Diamond$

Example 3. Suppose a hardware implementation I_F of F is given. We can use average values of I_F's time or power consumption for fixed input values k and m to define $f(k, m)$. $\Diamond$

The target of our information-theoretic analysis is the side-channel f, and we assume that the adversary is oblivious to the result $F(k, m)$ of the computation. However, knowledge of the values of F can be explicitly considered in the analysis.

Example 4. Consider $F\colon K \times M \to D$ and a side-channel $f\colon K \times M \to O$ of I_F. One can model that the adversary has access to the values of F by considering the function $F \times f\colon K \times M \to D \times O$ defined by $(F \times f)(k, m) = (F(k, m), f(k, m))$ as the side-channel. $\Diamond$

Note that the security of many cryptographic functions F relies on computational assumptions rather than on information-theoretic considerations. Subjecting F to an information-theoretic analysis may therefore lead to overly pessimistic results. We return to this point and its implications in Section 3.3.

2.2 Attack Strategies

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

Adversary's Choices and Knowledge During the query phase, the adversary chooses a message $m \in M$ and sends it to the system. 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 sufficient 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 k' are *indistinguishable under m* if $f(k, m) = f(k', m)$. 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 adversary's observation under the adversary's input is the set of all keys that could have led to this observation; we use this set to represent the adversary'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 $f\colon K \times M \to O$ 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.

Example 5. Let $K = \{1, 2, 3, 4\}$, $M = \{1, 2, 3\}$, and $O = \{0, 1\}$. Now consider the function $f\colon K \times M \to O$ defined by

$$f(k, m) = \begin{cases} 1 & \text{if } k \leq m \\ 0 & \text{otherwise}\,. \end{cases}$$

Then f gives rise to the set of partitions $\mathcal{P}_f = \{\{\{1\}, \{2, 3, 4\}\}, \{\{1, 2\}, \{3, 4\}\}, \{\{1, 2, 3\}, \{4\}\}\}$ of K, where the mth partition in $\mathcal{P}_f$ corresponds to 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 adversary 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.

To reason about partitions, we introduce additional notation. We say that a partition Q of a set K *refines* a partition P of K (denoted by $Q \sqsubseteq P$) if 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 $\{A \cap B \mid A \in P, B \in Q\}$. Clearly, $P \cap Q \sqsubseteq P$ 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 vertices are labeled with subsets of K to model the adversary's decisions with respect to his possible observations.

In this tree, the label of each vertex is the set of keys that are coherent with the adversary's observations at this point. The root is labeled with K. This models that, before the attack, every $k \in K$ is a possible key candidate. An attack step is represented by a vertex together with its children. The label A of the parent node represents the basis for the adversary'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 adversary's choice of P as a query. By observing the system's response, the adversary learns which block of the successors actually contains the key. This vertex is the starting point for subsequent attack steps. With this formalization of an attack strategy, an actual attack corresponds to a path from the root in this tree.

Example 6. 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 that the adversary picks $\{\{1, 2\}, \{3, 4\}\}$ as his first query. If the system responds $\{1, 2\}$, the adversary 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 on the left-hand side of Figure 1. In contrast, suppose that the adversary picks $\{\{1\}, \{2, 3, 4\}\}$ as his first query. A response of $\{1\}$ determines the key. Otherwise, suppose the adversary chooses $\{\{1, 2\}, \{3, 4\}\}$ as his subsequent query. A response of $\{2\}$ determines the key. Otherwise, a further attack step with query $\{\{1, 2, 3\}, \{4\}\}$ is required to distinguish the keys 3 and 4. The corresponding attack strategy is depicted on the right-hand side of Figure 1. ◊

Formally, let $T = (V, E)$ be a tree with vertices V, edges $E \subseteq V \times V$, and

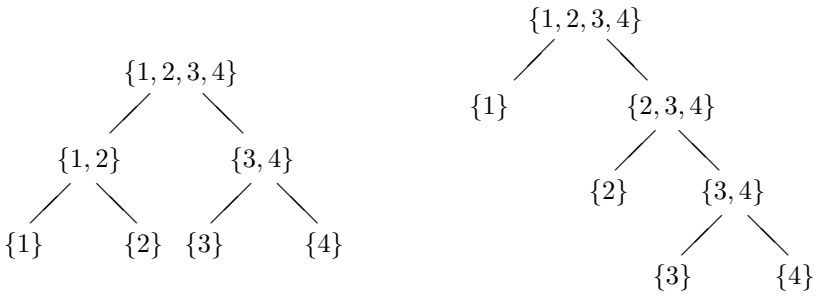

Figure 1: Attack Strategies

root $v^* \in V$. The *height* of a tree T is the length of a longest path from the root to a leaf in T.

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

1. $L(v^*) = K$, and
2. for every $v \in V$, there is a $P \in \mathcal{P}$ with $L(v) \cap P = \{L(w) \mid (v, w) \in E\}$.

An attack strategy is of *length* l if T has height l. An *attack* is a path $(v^*, \ldots, t)$ in T that starts at the root v^* 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 vertex 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 vertex. This leads to the following definition.

Definition 2. The partition *induced* by the attack strategy $\mathfrak{a} = (T, v^*, L)$ 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* $a = (v^*, \ldots, t)$ as $L(t)$.

Observe that this definition of coherence corresponds to our prior definition considering attacks (v^*, 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 sketch how the latter can be cast in our model.

Non-adaptive Attack Strategies An attack strategy is *non-adaptive* if the adversary 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 adversary choosing the same partition in all vertices at the same level of the attack strategy.

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

Definition 3. An attack strategy $\mathfrak{a} = (T, v^*, L)$ is *non-adaptive* if T is full and for every level i there is a $P_i \in \mathcal{P}$ such that $L(v) \cap P_i = \{L(w) \mid (v, w) \in E\}$, 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 give explicit representations of the induced partitions.

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 this assertion by induction on the length l of $\mathfrak{a} = (T, v^*, L)$. If $l = 0$, we have $P_{\mathfrak{a}} = L(v^*) = K = \bigcap \emptyset$. If $l > 0$, consider the full subtree T' of height $l-1$ of T. We have $P_{\mathfrak{a}} = \{L(w) \mid w \text{ is a leaf of } T\} = \bigcup_v \{L(w) \mid (v, w) \in E\}$, where v ranges over the leaves of T'. By Definition 1 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 coherent with the intuitive notion of a non-adaptive attack, as the side-channel information is only analyzed when the attack has finished.

Adaptive attacks are strictly more powerful than non-adaptive attacks in the sense that they can induce finer partitions in a given number of attack steps. To see this, consider again the set of partitions $\mathcal{P}$ from Example 6. The adaptive strategy depicted on the left-hand side of Figure 1 determines every key in two attack steps. It is not difficult to see that this cannot be achieved by any non-adaptive attack strategy in the same number of steps.

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 adversaries 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 adversary learns about the key by following an attack strategy $\mathfrak{a}$. Intuitively, the adversary 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 this remaining uncertainty. Focusing on the remaining entropy instead

of the adversary's information gain provides a concrete, meaningful measure that quantifies the adversary'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

$$\textit{initial uncertainty} = \textit{information gain} + \textit{remaining uncertainty},$$

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 [12, 34, 48]. Second, it demonstrates how the possibilistic model presented in Section 2 can serve as a basis for different probabilistic extensions.

In the following, assume a probability measure p is given on K and is known to the adversary. For a random variable $X : K \to \mathcal{X}$ with range $\mathcal{X}$, we define $p_X : \mathcal{X} \to [0,1]$ 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 random variables of particular interest. The first is the random variable U that models the 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* [47] 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 [12].

Given another random variable $Y : K \to \mathcal{Y}$, one denotes by $H(X|Y = y)$ 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 adversary's initial uncertainty about the key and $H(U|V_\mathfrak{a} = B)$ is the adversary'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 adversary's expected remaining uncertainty about the key after performing an attack with strategy $\mathfrak{a}$. As the value of $V_\mathfrak{a}$ is determined by that of 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 [34]. The difference between guessing entropy and Shannon entropy is that Shannon entropy quantifies the effort for determining the value of X by asking arbitrary binary questions, whereas the guessing entropy quantifies this effort when the questions are restricted to equality tests.

As we assume p to be public, the optimal way to guess the value of X is to try each of the possible values in order of their decreasing probabilities. Without loss of generality, 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

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

This represents the expected number of 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 adversary must perform for key recovery after having carried out a side-channel attack with strategy $\mathfrak{a}$.

Min-Entropy The min-entropy captures the probability of correctly determining the value of a random variable X in a single guess. From this probability, it is straightforward to derive bounds on the probability for correctly determining the value of X in an arbitrary number of guesses. The success probability of a single guess can also be estimated using the conditional Shannon entropy, e.g. using Fano's inequality. However, as pointed out in [48], this estimation is not always accurate. Hence, it is preferable to use min-entropy to compute the success probability of single guesses.

Formally, the *min-entropy* H_∞ is defined as $H_\infty(X) = -\log_2 (\max_{x \in \mathcal{X}} p_X(x))$. The *conditional min-entropy* $H_\infty(X|Y)$ is defined by

$$H_\infty(X|Y) = -\log_2 \left(\sum_{y \in \mathcal{Y}} p_Y(y) \max_{x \in \mathcal{X}} p_{X|Y=y}(x) \right) \ .$$

This quantifies the expected probability of correctly determining the secret in one guess after having observed the outcome of Y [48]. As before, $H_\infty(U|V_\mathfrak{a})$ quantifies the expected probability of correctly determining the secret in one guess after having carried out a side-channel attack with strategy $\mathfrak{a}$.

Relationships between Entropy Measures The presented entropy measures are related as follows. Massey [34] 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 $H(X)$. Min-entropy and Shannon entropy are related by $H_\infty(X) \leq H(X)$, with equality if X is uniformly distributed. For the random variable X with $p_X(1) = \frac{1}{2}$ and $p_X(i) = \frac{1}{2n}$, for $i \in \{2, \ldots, n\}$, we obtain $H(X) = \frac{1}{2}\log_2 4n$ and $H_\infty(X) = \log_2 2$. This example shows that there can be no general upper bound for $H(X)$ in terms of $H_\infty(X)$.

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. $H(U|V_\mathfrak{a}) = \frac{1}{n}\sum_{i=1}^{r} n_i \log_2 n_i$,
2. $G(U|V_\mathfrak{a}) = \frac{1}{2n}\sum_{i=1}^{r} n_i^2 + \frac{1}{2}$, *and*
3. $H_\infty(U|V_\mathfrak{a}) = \log \frac{n}{r}$.

Proof.

1. The entropy of a uniformly distributed random variable with finite range $\mathcal{X}$ is given by $\log_2 |\mathcal{X}|$ (see, e.g. [3]). Consequently, $H(U|V_\mathfrak{a} = B_i) = \log_2 n_i$ and $H(U|V_\mathfrak{a}) = \sum_{i=1}^{r} \frac{n_i}{n} H(U|V_\mathfrak{a} = B_i) = \frac{1}{n}\sum_{i=1}^{r} n_i \log_2 n_i$.
2. We have $G(U|V_\mathfrak{a}) = \sum_{i=1}^{r} \frac{n_i}{n} G(U|V_\mathfrak{a} = B_i) = \sum_{i=1}^{r} \frac{n_i}{n} \sum_{j=1}^{n_i} j \frac{1}{n_i} = \frac{1}{n}\sum_{i=1}^{r} \frac{(n_i+1)n_i}{2} = \frac{1}{2n}\sum_{i=1}^{r} n_i^2 + \frac{1}{2}$.
3. See [48] for a proof of this assertion.

□

Worst-Case Entropy Measures All of the 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* $\widehat{G}(U|V_{\mathfrak{a}})$ of U given $V_{\mathfrak{a}}$ as $\widehat{G}(U|V_{\mathfrak{a}}) = \min\{G(U|V_{\mathfrak{a}} = B) \mid B \in P_{\mathfrak{a}}\}$. The value $\widehat{G}(U|V_{\mathfrak{a}})$ 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 7. 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 $\widehat{G}(U|V_P) = 1$, which reflects that there exists a key that is trivial to guess given knowledge of its enclosing block in P. The conditional guessing entropy yields $G(U|V_P) = 4.6$ which reflects that, on the average, still 4.6 guesses are necessary for key recovery. Note that $\widehat{G}(U|V_P) = \widehat{G}(U|V_Q)$ 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. $\Diamond$

Ultimately, it depends 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 optimal attacks and give bounds for what an adversary 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 and the adversary's uncertainty about the key. More side-channel measurements reduce the uncertainty, which means that fewer guesses are needed. 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 $\mathcal{E} \in \{H, G, H_\infty\}$ 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 to assess 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 adversary proceeds optimally. A strategy is optimal if an adversary who follows it can expect to have less uncertainty about the key than with any other strategy of the same length.

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

Next, we define the expected remaining uncertainty as a function of the number of attack steps taken by an optimal adversary. In this way, we relate

two important aspects of a system's vulnerability: how much information an adversary can obtain and how many queries he needs for this.

Definition 5. Let $\mathcal{P}$ be a set of partitions of K and let $\mathcal{E} \in \{H, G, H_\infty\}$. 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 $\mathfrak{a} = (T, v^*, L)$ is the *prefix* of an attack strategy $\mathfrak{b} = (T', v', L')$ if T is a subtree of T', $v^* = v'$, and L and L' coincide on T. We denote this by $\mathfrak{a} \leq \mathfrak{b}$.

Proposition 3. *Let $\mathcal{E} \in \{H, G, H_\infty\}$ 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. To prove 3.1, first observe that $\mathfrak{a} \leq \mathfrak{b}$ implies $P_\mathfrak{a} \sqsupseteq P_\mathfrak{b}$. Hence it suffices to show that $\mathcal{E}(U|V_P)$ decreases as P is refined. We show this for $\mathcal{E} \in \{H, G, H_\infty\}$.

For $\mathcal{E} = H$, recall that $H(U) = H(U|V_P) + H(V_P)$. For $P_\mathfrak{a} \sqsupseteq P_\mathfrak{b}$ the value of $V_\mathfrak{a}$ is determined by the value of $V_\mathfrak{b}$, hence $H(V_\mathfrak{a}) \leq H(V_\mathfrak{b})$. Hence, $H(U|V_\mathfrak{a}) \geq H(U|V_\mathfrak{b})$ follows.

For $\mathcal{E} = G$, consider a partition P of K. We have

$$G(U|V_P) = \sum_{B \in P} p_V(B) \sum_{i=1}^{|B|} i \ p_{U|V=B}(x_i^B) = \sum_{B \in P} \sum_{i=1}^{|B|} i \ p_U(x_i^B) \ ,$$

where the elements x^B of block B are indexed in order of their decreasing probabilities. Observe that the probabilities in the sum on the right-hand side are independent of the partition P and remain constant as P is refined. In contrast, the indices of the elements decrease as P is refined: the blocks become smaller, but the relative ordering within each block is preserved. Hence, the assertion follows.

For $\mathcal{E} = H_\infty$, consider a partition P of K. As V_P is determined by U, we have $H_\infty(U|V_P) = -\log_2 \left(\sum_{B \in P} \max_{k \in B} p_U(k) \right)$ (see [48]). The values of p_U and the relative ordering of probabilities within each block remain constant as the partition is refined, but the number of blocks increases. Hence, the assertion follows.

Finally, Assertion 3.2 is a simple consequence of 3.1. □

3.3 Computational vs. Information-theoretic Security

The function $\Phi_{\mathcal{E}}$ characterizes the security of a system in terms of information-theoretic entropy measures. This abstracts from the computational cost of recovering the secret from the available information, which may range from trivial to infeasible. We illustrate this with two examples from cryptography. In both examples, the full key information is revealed, but only in the first example is the system insecure with respect to adversaries that are computationally bounded.

Example 8. Consider the one-time pad $\oplus\colon K \times M \to D$, where $K = M = D = \{0,1\}^w$ for some $w \in \mathbb{N}$ and suppose that the adversary can provide messages m and observe the output $k \oplus m$. Then m and $k \oplus m$ contain all the information about k, that is $\Phi_H(1) = 0$. Moreover, k can be efficiently obtained from these values by computing $k = (k \oplus m) \oplus m$. ◊

Example 9. Let $K = \{0, \ldots, d-1\}$ and let $O = \{g^0, g^1, \ldots, g^{d-1}\}$ be a finite cyclic group of order d, with generator g. Define $f\colon K \times M \to O$ as $f(k,m) = g^k$. Then $f(k,m)$ and g contain all the information about k, that is $\Phi_H(1) = 0$. However, there is no known efficient algorithm for computing k from this information, that is, for computing discrete logarithms. ◊

Example 9 illustrates that the availability of secret information does not necessarily entail that a system is vulnerable to computationally bounded adversaries. Hence, an information-theoretic analysis may lead to overly pessimistic assertions about a system's security. However, this is typically not the case for the side-channels arising in practice: many attacks [11, 13, 15, 27, 41] show that computational limitations are typically not the limiting factor for recovering keys. Hence, information-theoretic bounds are indeed realistic.

4 Automated Vulnerability Analysis

In the following, we first analyze the time complexity of computing $\Phi_{\mathcal{E}}$ by brute-force. The resulting bounds are double exponential and hence infeasible for realistic parameter sizes. Afterwards we present a more efficient heuristic algorithm that addresses this problem.

Throughout this section, let $\mathcal{P}$ be a set of partitions of K and let $r > 2$ 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., [19]). 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 $f\colon K \times M \to O$, 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 argument shows that the partition induced by an attack strategy of length n has at most r^n blocks. We prove Lemma 1 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 on the time-complexity for 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 ";". We iterate over all $k \in K$. For each k, we call $\textsc{Find}(k)$ on the representation of the partition i_0 to obtain the index j of k's enclosing block in P_{i_1}. Use $\textsc{Find}(k)$ 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, we 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 algorithm based on a greedy heuristic, and we exhibit formal connections between $\Phi_{\mathcal{E}}$ and the function computed by the heuristic algorithm.

A Greedy Heuristic Consider an adversary 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 $V_{P\cap A} \colon A \to P \cap A$ that models the choice of the enclosing block in $P \cap A$.

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

We next define an approximation $\widehat{\Phi}_{\mathcal{E}}$ 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 $\widehat{\Phi}_{\mathcal{E}}$ 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 $\widehat{\Phi}^{\mathfrak{a}}_{\mathcal{E}}$ as $\widehat{\Phi}^{\mathfrak{a}}_{\mathcal{E}}(n) = \mathcal{E}(U|V_{\mathfrak{a}(n)})$, for all $n \leq l$. We only use $\mathfrak{a}$ 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 $\widehat{\Phi}_{\mathcal{E}}$ instead of $\widehat{\Phi}^{\mathfrak{a}}_{\mathcal{E}}$.

Theorem 2. *The value* $\widehat{\Phi}_{\mathcal{E}}(n)$ *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 next exhibit inequalities between the values of $\Phi_{\mathcal{E}}$ and $\widehat{\Phi}_{\mathcal{E}}$, which we will later use when interpreting our experimental results.

Relating $\Phi_{\mathcal{E}}$ and $\widehat{\Phi}_{\mathcal{E}}$ The definition of a greedy strategy raises the question of whether such strategies are also optimal. The following example illustrates that this is not the case in general.

Example 10. 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. $\Diamond$

As the next example[2] shows, the approximation provided by a greedy strategy can be arbitrarily bad.

Example 11. Let $K = (\{0,1\}^{2n})^J \times \{0,1\}^n$ and $M = J \cup \{0,1\}^n \cup \{*\}$, where J is a finite set disjoint from $\{*\}$ and $\{0,1\}^n$. Define $f(k,m)$ as follows. For $m \in J$, $f(k,m)$ returns the mth component of k, i.e., a vector of $2n$ bits. For $m = *$, $f(k,m)$ returns the last n bits of k. Finally, for $m \in \{0,1\}^n$, $f(k,m) = k$ if the last n bits of k are equal to m, and $f(k,m) = *$ otherwise. The intuition behind the definition of f is the following. Choosing an element of J as a query reveals $2n$ bits of k, whereas choosing $m = *$ or $m \in \{0,1\}^n$ as queries reveals less information about k, as confirmed by a simple calculation. Consequently, a greedy adversary will iteratively query f with all messages from J to extract the first $|J|\,2n$ bits of k, and will then choose $*$ as a query to obtain the last n bits of k. In contrast, an optimal adversary will first query the system with $*$ and learn the last n bits of k. In the second step, the adversary will then use these n bits as a the input to f, thereby obtaining the full key. $\Diamond$

Although Example 11 shows that the difference between $\widehat{\Phi}_{\mathcal{E}}$ and $\Phi_{\mathcal{E}}$ can be arbitrarily large in general, we can establish the following relationships.

Proposition 4. *For $\mathcal{E} \in \{H, G, H_\infty\}$, we have*

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

[2]The authors thank Dominique Unruh for suggesting this example.

```
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 $\widehat{\Phi}_{\mathcal{E}}$ in Haskell

We will make use of Proposition 4 in our experiments. Proposition 4.1 shows that the first greedy choice is optimal. Proposition 4.2 implies that an implementation that is shown to be vulnerable when analyzed with $\widehat{\Phi}_{\mathcal{E}}$ is also vulnerable with respect to $\Phi_{\mathcal{E}}$. Proposition 4.3 implies that if $\widehat{\Phi}_{\mathcal{E}}$ 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 both $\Phi_{\mathcal{E}}$ and $\widehat{\Phi}_{\mathcal{E}}$ in Haskell [9]. Here, we present our implementation of $\widehat{\Phi}_{\mathcal{E}}$. 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 for computing $\widehat{\Phi}_{\mathcal{E}}$ 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 the `n`-fold application of a greedy refinement step using `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 $\mathcal{E} \in \{H, G, H_\infty\}$. Applying `order` to the result of `greedy` yields $\widehat{\Phi}_{\mathcal{E}}$. The simplicity of this implementation shows that an automation of our methods is indeed straightforward.

5 Experiments

We now report on case studies where we automatically derive bounds on the information that security protocols and hardware implementations of cryptographic algorithms leak to adaptive adversaries. Throughout this section, we use the guessing entropy G as a measure of uncertainty. The reason for this choice is that the guessing entropy has a direct interpretation in terms of an attacker's effort for recovering the key by exhaustive search and, hence, in terms of security. We abbreviate Φ_G by Φ and $\widehat{\Phi}_G$ by $\widehat{\Phi}$, 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 Side-Channel Attacks

We first apply our methods to analyze implementations of different cryptographic algorithms with respect to their vulnerability to timing and power attacks. Our methods can be applied to any implementation with a deterministic side-channel model. As a proof of concept, we focus on implementations in synchronous hardware since, 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 [35].

5.1.1 Setup

Approximating Φ 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 key space 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 methods to address this problem.

1. We approximate Φ by $\widehat{\Phi}$. We will see that $\widehat{\Phi}$ matches Φ on our example data, although this does not hold in general (see Example 10).

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

For a fixed n, we depict the values of $\widehat{\Phi}^w(n)$ along the w-axis. If this graph exhibits regularity, we can extrapolate to larger values of k; this reflects the working assumption that the graph's regularity reflects the structural similarity of the parameterized algorithms. Using both methods, 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 [46] 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 signal 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 toggles. Counting signal transitions thus provides approximate information about the actual power consumption, and we will use it for our proof of concept.

Note that formal bounds based on counting signal transitions must be interpreted with care: practical power attacks often exploit technology-dependent electrical effects that are not captured by this simple measure. However, it is straightforward to replace this measure by 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.

Defining f 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 $f\colon \{0,1\}^w \times \{0,1\}^w \to O$. For timing analysis, we use $O = \mathbb{N}$ to represent the number of clock ticks required for the algorithm to terminate. For power analysis, we use $O = \mathbb{N}^d$ to represent the number of signal transitions in each of the d clock cycles.

5.1.2 Results

Timing Attacks against Shift-and-add 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 $\widehat{\Phi}^w(1) = \widehat{\Phi}^w(2)$ 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 if and only if they have the same Hamming weight. The equivalence class of w-bit arguments with Hamming weight l has precisely

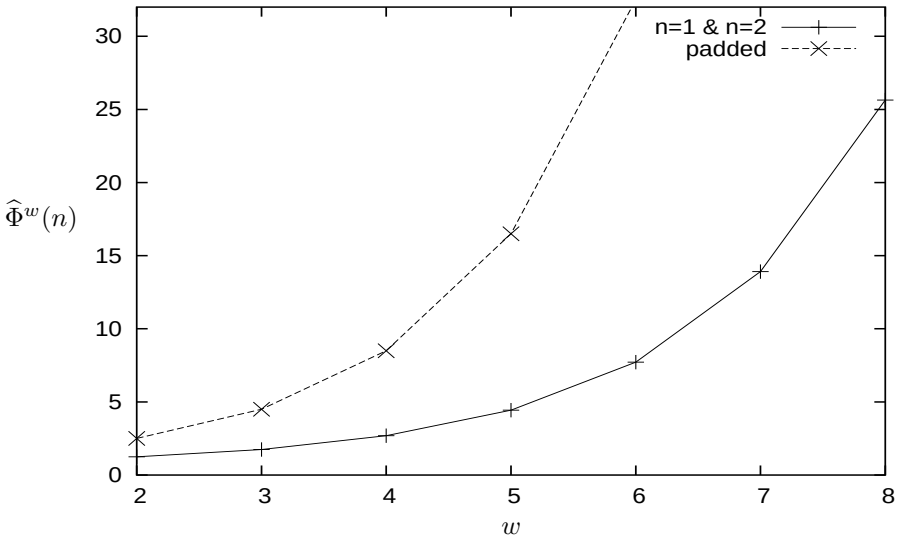

Figure 3: Integer Multiplication

$\binom{w}{l}$ 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 [29].

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 [29], 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 [20]. It takes two arguments, the 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. The results of the analysis are depicted in Figure 4. To interpret the graph, observe that $\Phi^w(1) = \widehat{\Phi}^w(1)$ and $\widehat{\Phi}^w \geq \Phi^w$ 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

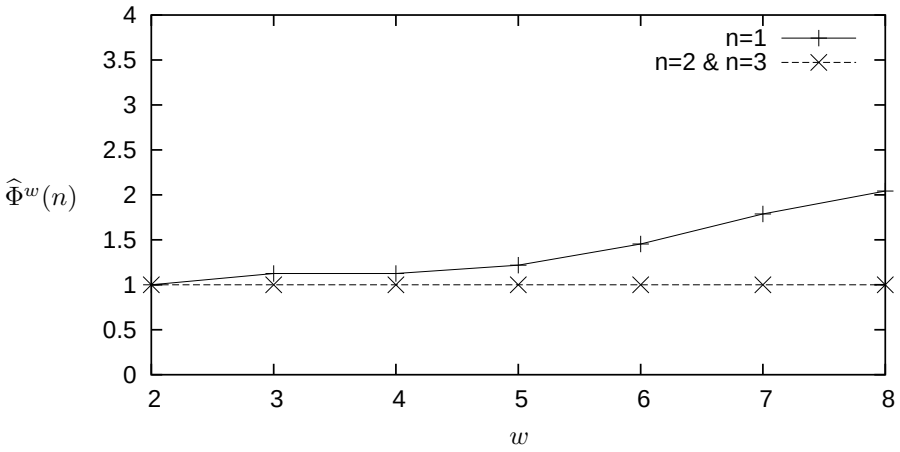

Figure 4: Finite-Field Exponentiation

second measurement, however, suffices to 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 signal 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 key space and therefore do not scale.[3] However, our data shows 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., [11, 13, 26, 27]. This is due to the fact that noise in the measurements is typically dealt with by increasing their number. Template attacks [15] 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.

[3] With precomputed value tables for the time consumption of the finite-field exponentiation algorithm, the computation of $\widehat{\Phi}^8(2)$ took 40 minutes on a 2.4 GHz machine with 3 gigabytes of RAM.

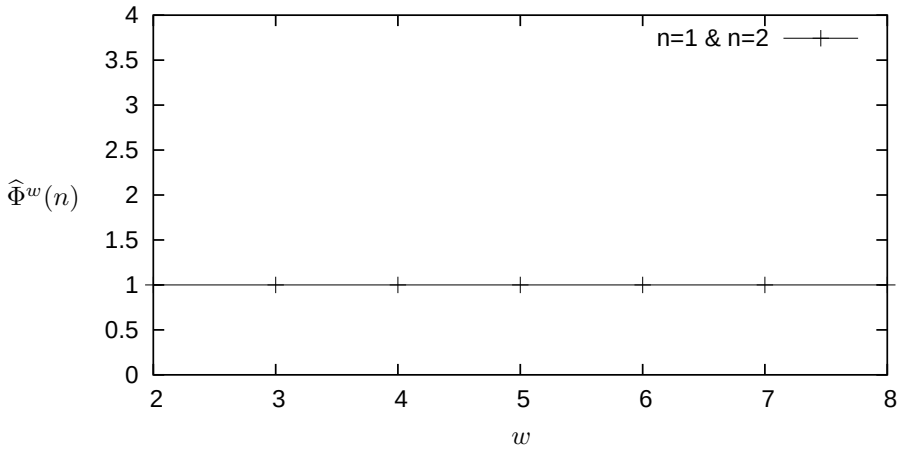

Figure 5: Finite-Field Multiplication

5.2 Protocol Attacks

We report on a case study where we use our methods for the quantitative analysis of a security protocol. The conceptual difference to the results just presented is that we now analyze the protocol design rather than the implementation.

The protocol we analyze stems from the Application Programming Interface (API) of a hardware security module for Automated Teller Machines (ATMs). The attack is well-known [18], but a formal quantitative analysis was, until now, lacking.

5.2.1 The PIN Integrity Check Protocol

When performing bank transactions using ATMs, customers are typically required to authenticate themselves with a Personal Identification Number (PIN). For verification, the PIN is transferred from the ATM to the customer's bank. If there is no direct communication between the ATM and the customer's bank, the PIN must be passed through intermediate network switches.

To protect the PIN during the transfer, it is sent as an encrypted PIN block. The ISO-0 standard describes such a PIN block format: each digit of the PIN is XORed with a digit of the corresponding account number, where digits are represented by 4 bits.[4] The ISO-1, ISO-2, and ISO-3 standards describe alternative PIN block formats. Each pair of adjacent switches shares a so-called transport key, which is used for encrypting the PIN block transfer between them.

Upon reception, a switch decrypts the PIN block. To pass the result on to the next switch, the PIN block may need to be transformed into another format. For this, the PIN is extracted from the PIN block and checked for integrity, which is explained below. This operation (and other operations that involve secret keys

[4]See, e.g., [8] for details.

and unencrypted PINs) are performed by Hardware Security Modules (HSMs), which are tamper-resistant cryptographic devices. HSMs are trusted, but they are controlled by applications that run on the potentially compromised switches. Hence, calls to the HSM's API must not leak any information about the secret keys or the PINs that the HSM processes.

A number of documented attacks exhibit major flaws in the PIN processing API of HSMs [2, 8, 10, 18]. For example, the so-called *PIN format attack* recovers the PIN by inducing errors in the integrity check of the reformatting step. Clulow et al. [10] formalize the involved API calls in terms of a simple protocol between the caller and the HSM, which we depict in Figure 6. Here,

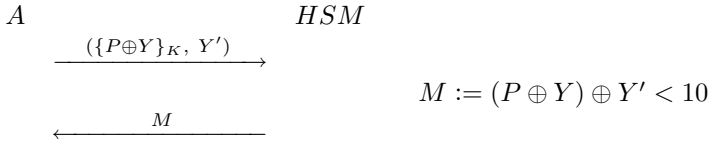

Figure 6: The PIN Integrity Check Protocol

HSM represents the HSM, A represents the application that controls the HSM, and K is a secret key shared between A and HSM. The right-arrow represents a function call and the left-arrow represents the return value. The protocol formalizes the process of extracting the PIN from an encrypted ISO-0 buffer and applying a simple integrity check, where it is verified that each of the PIN's digits is a decimal number. For simplicity, the protocol is formalized for single digit PINs and account numbers.

As a first step, the encrypted PIN buffer $\{P \oplus Y\}_K$ and the account number Y' are provided to the HSM. The HSM decrypts the buffer with K and XORs it with Y', that is, it computes $P' = (P \oplus Y) \oplus Y'$. The HSM then verifies that P' is a decimal number, i.e. that $P' < 10$, and returns M, the result of this check. An honest caller will provide the correct account number, that is, $Y' = Y$ (and therefore $P' = P$). However, this benign behavior is not enforced by the protocol. By suitably choosing values for Y' and observing the results of the corresponding integrity checks, A can deduce information about P's value.

To quantify the information leaked, we cast the integrity check of a single digit as a function $f\colon K \times M \to \{\texttt{True}, \texttt{False}\}$, where $f(k, m) = k \oplus m < 10$ and $K = M = \{0, \ldots, 15\}$. Here, k represents one digit of the unencrypted PIN buffer (i.e., $P \oplus Y$), and m represents the adversary's choice of a digit of the account number (i.e., Y'). We assume that the adversary knows the correct account number Y and, hence, his uncertainty about $P \oplus Y$ corresponds to his uncertainty about P.

5.2.2 Results

The results of the analysis of the PIN Integrity Check Protocol of a single digit (that is, a 4 bit portion of the PIN buffer) are given in Figure 7.

	$n=0$	$n=1$	$n=2$	$n=3$	$n=4,5,\ldots$
$\Phi(n)$	8.5	4.75	2.75	1.75	1.5

Figure 7: Resistance of the PIN integrity check protocol

For $n=0$, the resistance of 8.5 matches the guessing entropy of $0.5(2^4+1)$ for an 4 bit PIN buffer prior to a run of the PIN integrity protocol. The guessing entropy decreases with the number of interactions until, after only 4 attack steps, it levels off at a value of 1.5. This value corresponds to the expected guessing effort for a random variable with two equally likely alternatives. Indeed, an inspection of the partition induced by an optimal attack shows that the blocks are of the form $\{P, P\oplus 1\}$; hence, the PINs can be determined up to their parity. Although this fact was already known [18], the derivation of quantitative bounds for the leaked information was, until now, an open challenge (see [10]).

6 Related Work

There has been substantial work in information-flow security on detecting or quantifying information leaks. Early approaches focus on quantifying the capacity of covert channels between processes in multi-user systems [23, 38, 54]. The models predate the first published side-channel attack against cryptography [26] and are so general that it is unclear how they could be instantiated to address the problem at hand.

A number of measures have been proposed for quantifying the information flow in concrete programs. Di Pierro et al. [43] quantify the probability of correctly distinguishing two processes in a probabilistic concurrent language. This measure can, in principle, be used to characterize the adversary's ability to distinguish between program runs with different keys. Lowe [32] quantifies information flow in a possibilistic process algebra by counting the number of distinguishable behaviors. This measure is closely related to the information-theoretic notion of channel capacity, which captures the maximal information leakage with respect to all possible probability distributions. The information measures proposed by Clark et al. [16] are closest to ours: the authors relate observational equivalence to random variables and use Shannon entropy to quantify the information flow to a passive observer. None of the measures mentioned above captures adversaries that can adaptively interact with the system.

Clarkson et al. [17] consider adaptive adversaries. However, their measure quantifies the accuracy of beliefs rather than the decrease in uncertainty. This captures attack scenarios in which adversaries have strong prior beliefs about the secrets (e.g., by overhearing a password), but does not seem suitable for analyzing side-channel attacks.

Several approaches in language-based security use security type systems to detect timing side-channels in both sequential and multithreaded settings, see

[1, 5, 24] and [45, 49], respectively. If a program successfully type checks, then an adversary cannot gain any information about the secret, even if he exhaustively runs the program on all possible public inputs. 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 [52, 53] are based on more realistic assumptions about the system, but do not model adaptive adversaries. For formal connections between bisimulation-based security properties and the measure proposed in this paper, see [28].

There is a large body of work on side-channel cryptanalysis, in particular on attacks and countermeasures. Chari et al. [14] 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. Kelsey et al. [25] show that some product ciphers can be broken even if only a small amount of side-channel information is available to the adversary. Micali et al. [37] propose a mathematical model that aims at providing provably secure cryptography in the presence of such bounded side-channel leakage. Using a similar approach, Dziembowski and Pietrzak [21] recently obtained the first positive results, where they construct a stream cipher that is provably secure in the presence of bounded leakage. Recently, Naor and Segev [40] proposed a public-key cryptosystem with similar properties. As ongoing work, we are investigating the integration of our information-theoretic bounds with such leakage-resilient cryptosystems.

The model of Micali et al. has been specialized to a framework for the evaluation of side-channel attacks by Standaert, Malkin, and Yung [51] (henceforth called the SMY-model), with applications described in [33, 42, 50]. The SMY-model uses two largely independent metrics for the evaluation of systems. The information-theoretic metric considers only non-adaptive chosen-message adversaries and is not given a direct interpretation in terms of security. The security metric characterizes the security of a system in terms of the success rate for recovering the correct key when applying a given algorithm (e.g., Bayesian classification) to the measurement data. In this way, an analysis with the SMY-model yields meaningful assertions about the effectiveness of the chosen key recovery algorithm, but not necessarily worst-case bounds.

In contrast to the SMY-model, our metric abstracts from any concrete statistical analysis technique and captures adversaries that can adaptively interact with the system. This enables us to derive sound worst-case bounds for what can, in principle, be achieved in a side-channel attack. Clearly, such formal bounds are practically relevant only if they are based on a valid system model. For power analysis, the number of bit-transitions provides only a rough estimation of the power consumption of a circuit; more precise power models are required for deriving practically relevant security guarantees. For timing analysis, however, the number of clock ticks provides a reasonable, deterministic abstraction of time. For this application domain, our metric offers the advantage of a quantitative bound that is sound with respect to adaptive adversaries and arbitrary statistical analysis techniques.

Finally, the model of side-channels presented in this paper serves as the

basis for the quantitative analysis of systems with respect to unknown-message attacks [4, 31]. By comparing the side-channel leakage in unknown-message attacks to that in adaptive attacks, one can quantify the resistance gained by applying message-blinding, the state-of-the art countermeasure against timing attacks.

7 Conclusions and Future Work

We have presented a quantitative model that we use to measure a system's resistance to adaptive side-channel attacks. On the theoretical side, our model 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 analyze the resistance of realistic systems to adaptive side-channel attacks. We have demonstrated that this analysis can be performed at the push of a button and that the results are easily interpreted in terms of an adversary's remaining guessing effort for recovering the secret. Hence our methods require minimal effort and expertise, which suggests that they can become a valuable tool for the development of security-critical systems.

As ongoing work, we are extending our model with statistical techniques for entropy estimation [6, 7]. 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 that 1%. However, the existing confidence intervals for this estimation are too large for practical use and, as future work, we hope to improve them.

Another subject for future work is to investigate whether Φ can be approximated by language-based techniques, for example, by a security type system. This would enable us to derive bounds for systems with larger or infinite state spaces. Furthermore, it would be interesting to integrate randomness into our model. This would lead to more accurate bounds for attacks in which noise is an important factor, such as remote timing attacks. Finally, it is an open problem to determine information-theoretic bounds for attacks against systems with state. Progress here would enable the quantitative analysis of cache attacks.

References

[1] J. Agat. Transforming out Timing Leaks. In *Proc. 27th ACM Symposium on Principles of Programming Languages (POPL 2000)*, pages 40–53. ACM, 2000.

[2] R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov. Cryptographic Processors - A Survey. *Proceedings of the IEEE*, 94(2):357–369, 2006.

[3] R. B. Ash. *Information Theory.* Dover Publications Inc., 1990.

[4] M. Backes and B. Köpf. Formally Bounding the Side-Channel Leakage in Unknown-Message Attacks. In *Proc. 13th European Symposium on Research in Computer Security (ESORICS 2008)*, volume 5283 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2008.

[5] G. Barthe, T. Rezk, and M. Warnier. Preventing Timing Leaks Through Transactional Branching Instructions. In *Proc. 3rd Workshop on Quantitative Aspects of Programming Lanugages (QAPL 2006)*, Electronic Notes in Theoretical Computer Science (ENTCS), pages 33–55. Elsevier, 2005.

[6] G. Basharin. On a Statistical Estimate for the Entropy of a Sequence of Independent Random Variables. *Theory of Probability and Its Applications*, 47:333–336, 1959.

[7] T. Batu, S. Dasgupta, R. Kumar, and R. Rubinfeld. The complexity of approximating entropy. In *Proc. 34th ACM Symposium Symposium on Theory of Computing (STOC 2002)*, pages 678–687. ACM, 2002.

[8] O. Berkman and O. M. Ostrovsky. The Unbearable Lightness of PIN Cracking. In *Proc. 11th International Conference on Financial Cryptography and Data Security (FC 2007)*, volume 4886 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2007.

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

[10] M. Bond and J. Clulow. Extending Security Protocol Analysis: New Challenges. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 125(1):13–24, 2005.

[11] D. Brumley and D. Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[12] C. Cachin. Entropy Measures and Unconditional Security in Cryptography. Dissertation No. 12187. ETH Zürich, 1997.

[13] J. Cathalo, F. Koeune, and J.-J. Quisquater. A New Type of Timing Attack: Application to GPS. In *Proc. 5th International Workshop of Cryptographic Hardware and Embedded Systems (CHES 2003)*, volume 2779 of *Lecture Notes in Computer Science*, pages 291–303. Springer, 2003.

[14] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Proc. Advances in Cryptology (CRYPTO 1999)*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[15] S. Chari, J. R. Rao, and P. Rohatgi. Template Attacks. In *Proc. 4th International Workshop of Cryptographic Hardware and Embedded Systems (CHES 2002)*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[16] D. Clark, S. Hunt, and P. Malacaria. Quantitative Information Flow, Relations and Polymorphic Types. *Journal of Logic and Computation*, 18(2):181–199, 2005.

[17] M. Clarkson, A. Myers, and F. Schneider. Belief in Information Flow. In *Proc. 18th IEEE Computer Security Foundations Symposium (CSFW 2005)*, pages 31–45. IEEE Computer Society, 2005.

[18] J. Clulow. The Design and Analysis of Cryptographic Application Programming Interfaces for Security Devices. Master's thesis, University of Natal, SA, 2003.

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

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

[21] S. Dziembowski and K. Pietrzak. Leakage-Resilient Cryptography. In *Proc. 49th IEEE Symposium Symposium on Foundations of Computer Science (FOCS 2008)*, pages 293–302. IEEE Computer Society, 2008.

[22] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *Proc. 3rd International Workshop of Cryptographic Hardware and Embedded Systems (CHES 2001)*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

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

[24] D. Hedin and D. Sands. Timing Aware Information Flow Security for a JavaCard-like Bytecode. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 141(1):163–182, 2005.

[25] J. Kelsey, B. Schneier, D. Wagner, and C. Hall. Side Channel Cryptanalysis of Product Ciphers. *Journal of Computer Security (JCS)*, 8(2–3):141–158, 2000.

[26] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proc. Advances in Cryptology (CRYPTO 1996)*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[27] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proc. Advances in Cryptology (CRYPTO 1999)*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[28] B. Köpf. Formal Approaches to Countering Side-Channel Attacks. Dissertation No. 17500. ETH Zürich, 2007.

[29] B. Köpf and D. Basin. Timing-Sensitive Information Flow Analysis for Synchronous Systems. In *Proc. 11th European Symposium on Research in Computer Security (ESORICS 2006)*, volume 4189 of *Lecture Notes in Computer Science*, pages 243–262. Springer, 2006.

[30] B. Köpf and D. Basin. An Information-Theoretic Model for Adaptive Side-Channel Attacks. In *Proc. 14th ACM Conference on Computer and Communication Security (CCS 2007)*, pages 286–296. ACM, 2007.

[31] B. Köpf and M. Dürmuth. A Provably Secure And Efficient Countermeasure Against Timing Attacks. In *Proc. 22nd IEEE Computer Security Foundations Symposium (CSF 2009)*, pages 324–335. IEEE Computer Society, 2009.

[32] G. Lowe. Quantifying Information Flow. In *Proc. 15th IEEE Computer Security Foundations Symposium (CSFW 2002)*, pages 18–31. IEEE Computer Society, 2002.

[33] F. Mace, F.-X. Standaert, and J.-J. Quisquater. An Informtion Theoretic Evaluation of Side-Channel Resistant Logic Styles. In *Proc. 9th International Workshop of Cryptographic Hardware and Embedded Systems (CHES 2007)*, volume 4727 of *Lecture Notes in Computer Science*, pages 427–442. Springer, 2007.

[34] J. L. Massey. Guessing and Entropy. In *Proc. 1994 IEEE Symposium on Information Theory (ISIT 1994)*, page 204. IEEE Computer Society, 1994.

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

[36] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Power Analysis Attacks of Modular Exponentiation in Smartcards. In *Proc. 1st International Workshop of Cryptographic Hardware and Embedded Systems (CHES 1999)*, volume 1717 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 1999.

[37] S. Micali and L. Reyzin. Physically Observable Cryptography (Extended Abstract). In *Proc. First Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[38] J. K. Millen. Covert Channel Capacity. In *Proc. 1987 IEEE Symposium on Security and Privacy (Oakland 1987)*, pages 60–66. IEEE Computer Society, 1987.

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

[40] M. Naor and G. Segev. Public-Key Cryptosystems Resilient to Key Leakage. In *Proc. Advances in Cryptology (CRYPTO 2009)*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, 2009.

[41] D. A. Osvik, A. Shamir, and E. Tromer. Cache Attacks and Countermeasures: the Case of AES. In *Proc. RSA Conference 2006, Cryptographers' Track*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.

[42] C. Petit, F.-X. Standaert, O. Pereira, T. G. Malkin, and M. Yung. A Block Cipher based Pseudo Random Number Generator Secure Against Side-Channel Key Recovery. In *Proc. 2007 ACM Symposium on Information, Computer and Communications Security (ASIACCS 2007)*, pages 56–65. ACM, 2007.

[43] A. D. Pierro, C. Hankin, and H. Wiklicky. Approximate Non-Interference. In *Proc. 15th IEEE Computer Security Foundations Symposium (CSFW 2002)*, pages 3–17. IEEE Computer Society, 2002.

[44] J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Couter-Measures for Smard Cards. In *Proc. International Conference on Research in Smart Cards (E-smart 2001)*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[45] A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. 13th IEEE Computer Security Foundations Symposium (CSFW 2000)*, pages 200–215. IEEE Computer Society, 2000.

[46] 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.

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

[48] G. Smith. On the Foundations of Quantitative Information Flow. In *Proc. 13th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2009)*, volume 5504 of *Lecture Notes in Computer Science*, pages 288–302. Springer, 2009.

[49] G. Smith and D. Volpano. Secure Information Flow in a Multi-Threaded Imperative Language. In *Proc. 25th ACM Symposium on Principles of Programming Languages (POPL 1998)*, pages 355–364. ACM, 1998.

[50] F.-X. Standaert, E.Peeters, C. Archambeau, and J.-J. Quisquater. Towards Security Limits in Side-Channel Attacks. In *Proc. 8th International Workshop of Cryptographic Hardware and Embedded Systems (CHES 2006)*, volume 4249 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 2006.

[51] F.-X. Standaert, T. G. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. Cryptology ePrint Archive, Report 2006/139, 2006.

[52] T. Tolstrup and F. Nielson. Analyzing for Absence of Timing Leaks in VHDL. Presented at 6th International Workshop on Issues in the Theory of Security (WITS 2006), 2006.

[53] T. Tolstrup, F. Nielson, and H. Nielson. Information Flow Analysis for VHDL. In *Proc. 8th International Conference on Parallel Computing Technologies (PaCT 2005)*, volume 3606 of *Lecture Notes in Computer Science*, pages 79–98. Springer, 2005.

[54] J. Wittbold and D. Johnson. Information flow in nondeterministic systems. In *Proc. 1990 IEEE Symposium on Security and Privacy (Oakland 1990)*, pages 144–161. IEEE Computer Society, 1990.

[55] L. Zhong, S. Ravi, A. Raghunathan, and N. Jha. Power Estimation Techniques for Cycle-Accurate Functional Descriptions of Hardware. In *Proc. International Conference on Computer-Aided Design (ICCAD 2004)*, pages 668–675. ACM, 2004.