A Provably Secure And Efficient Countermeasure Against Timing Attacks

Boris Köpf
MPI-SWS
bkoepf@mpi-sws.org

Markus Dürmuth
Saarland University
duermuth@cs.uni-sb.de

***Abstract*—We show that the amount of information about the key that an unknown-message attacker can extract from a deterministic side-channel is bounded from above by $|O| \log_2(n+1)$ bits, where n is the number of side-channel measurements and O is the set of possible observations. We use this bound to derive a novel countermeasure against timing attacks, where the strength of the security guarantee can be freely traded for the resulting performance penalty. We give algorithms that efficiently and optimally adjust this trade-off for given constraints on the side-channel leakage or on the efficiency of the cryptosystem. Finally, we perform a case-study that shows that applying our countermeasure leads to implementations with minor performance overhead and formal security guarantees.**

I. Introduction

Side-channel attacks threaten the security of cryptographic algorithms by exploiting information that is revealed by the algorithm's physical execution. Characteristics such as running time [17], cache behavior [27], power consumption [18], and electromagnetic radiation [12], [29] have all been exploited to recover secret keys from implementations of different cryptographic algorithms. In distributed environments such as the internet, timing attacks are maybe the most daunting kind of side-channel attack: Timing can be measured and exploited remotely [4], opening the door for a potentially large number of attackers.

A number of countermeasures against timing attacks have been proposed [17], [4], most notably input blinding and constant-time implementations.

To describe *input blinding*, we use RSA as an example. Input blinding relies on the fact that all known timing attacks against RSA decryption require that the attacker be able to obtain a large number of pairs of ciphertexts and corresponding execution times. Blinding randomizes each ciphertext before decryption. In this way, the attacker's timing measurements are decorrelated from the ciphertexts, rendering all known timing attacks ineffective. Blinding has become the state-of-the-art countermeasure against timing attacks and is implemented in a wide range of crypto-libraries. However, it is still unclear whether blinding offers any kind of completeness in the sense that it defeats all possible timing attacks [17]. A recent result even shows that a blinded implementation may leak the entire key information if the number of measurements is sufficiently large [2].

In contrast, *constant-time* implementations defeat all timing attacks. However, this security guarantee comes at the price of a large performance penalty, as the system assumes its worst-case execution time on all inputs. For many resource-critical application domains, such a performance penalty is not acceptable. Moreover, for many platforms, constant-time software is hard to write and maintain. As a consequence, constant-time software is not widely used in practical implementations of cryptosystems.

In general (see also the section on related work), existing countermeasures against timing attacks are either not fully practical (e.g., in terms of the resulting performance penalty) or not known to be sound (i.e., they are not backed up by any formal security guarantee). It has been an open problem to devise countermeasures that satisfy both requirements. In this paper, we propose a solution to this open problem for systems with deterministic timing behavior.

The key to our solution is the insight that if a system with deterministic timing behavior is run on inputs that have been previously blinded, the amount of information about the key that is leaked by the system's timing behavior is bounded from above by

$$|O| \log_2(n+1)$$

bits, where O is the set of possible execution times, and n is the number of side-channel measurements made. We formally establish this bound, and we use it to derive a lower bound on the expected effort for determining the key by exhaustive search after a side-channel attack.

Motivated by this result, we propose the combination of blinding and bucketing as a novel countermeasure against timing attacks. *Bucketing* is the discretization of a system's execution times such that the results of the computation are only returned at a small number of fixed points in time. More precisely, bucketing partitions a system's possible execution times into intervals (*buckets*) of variable length, where, for each execution time, one waits until the enclosing bucket's upper bound before returning the result of the computation.

Bucketing reduces the number of possible timing observations (and hence improves the bound on the leaked information) at the cost of the system's performance. The resulting trade-off between security and performance can be adjusted by choosing the number of buckets and their bounds. To this end, we give an algorithm that, for a given number of buckets (i.e., a desired security guarantee), computes a bucketing such that the system's resulting performance penalty is minimal. Moreover, we give an algorithm that computes the minimal number of buckets (i.e., the best possible security guarantee) under the constraint that the resulting performance penalty does not exceed a given bound.

We have implemented our algorithms and have applied them to evaluate the effect of combining bucketing and blinding on the performance and security of a realistic example. As a case study, we analyze an implementation of an algorithm for 1024 bit RSA decryption. Our results show that meaningful security guarantees can be obtained with only a minor performance overhead: If a bucketing of 5 buckets is applied to the implementation, more than 2^{40} timing measurements are necessary for recovering an expected amount of 200 key bits (alternatively, 2^{40} side-channel measurements reduce our lower bound for the expected effort for determining the key by exhaustive search by a factor of 2^{200}). The performance overhead imposed by the bucketing amounts to less than 0.7% with respect to the blinded implementation without bucketing.

In summary, our contributions are both theoretical and practical. On the theoretical side, we prove a bound on the amount of information that a side-channel attacker can extract from a blinded implementation in a given number of measurements. This bound leads to a countermeasure against timing attacks, where the strength of the security guarantee can be freely traded for the resulting performance penalty. On the practical side, we give algorithms that efficiently and optimally adjust this trade-off for given constraints on the side-channel leakage or on the efficiency of the cryptosystem. Finally, we perform a case-study that shows that the combination of blinding and bucketing leads to implementations with minor performance overhead and formal security guarantees.

The remainder of this paper is structured as follows. In Section II, we present a measure for the amount of information that is leaked by a blinded implementation. In Section III, we prove a bound on this quantity and give an interpretation of this bound in terms of guessing. In Section IV, we present algorithms for computing optimal bucketings. In Section V, we present experimental results. We present related work and conclude in Sections VI and VII, respectively.

II. Preliminaries

In this section, we introduce our model of side-channels and an information-theoretic measure that captures the side-channel leakage of blinded implementations. To begin with, we recall some basic information theory. See [9] for a textbook on the subject.

A. Information Theory Basics

Let A be a finite set and $p\colon A \to \mathbb{R}$ a probability distribution. For a random variable $\mathcal{X}\colon A \to X$, we define $p_{\mathcal{X}}\colon X \to \mathbb{R}$ as $p_{\mathcal{X}}(x) = \sum_{a \in \mathcal{X}^{-1}(x)} p(a)$, which is often denoted by $p(\mathcal{X} = x)$ in the literature.

The *(Shannon) entropy* of a random variable $\mathcal{X}\colon A \to X$ is defined as

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

The entropy is a lower bound on the average number of bits required to represent the results of independent repetitions of the experiment associated with $\mathcal{X}$. Thus, in terms of guessing, the entropy $H(\mathcal{X})$ is a lower bound on the average number of binary questions that need to be asked to determine $\mathcal{X}$'s value [5]. The entropy is nonnegative, and reaches its maximal value of $H(\mathcal{X}) = \log_2(|X|)$ when $\mathcal{X}$ is uniformly distributed.

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

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

Entropy and conditional entropy are related by the equation $H(\mathcal{X}\mathcal{Y}) = H(\mathcal{Y}) + H(\mathcal{X}|\mathcal{Y})$, where $\mathcal{X}\mathcal{Y}$ is the random variable defined as $\mathcal{X}\mathcal{Y}(k) = (\mathcal{X}(k), \mathcal{Y}(k))$. The *mutual information*

$$I(\mathcal{X}; \mathcal{Y}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y})$$

captures the expected reduction in uncertainty about $\mathcal{X}$ when one learns the value of $\mathcal{Y}$. The mutual information is measured in bits.

B. Side-Channels

Let K be a finite set of keys, M be a finite set of messages, and D be an arbitrary set. We consider systems that compute functions of type $F\colon K \times M \to D$, and we assume that the attacker can make physical observations about F's implementation I_F that are associated with the computation of $F(k, m)$. We assume that the attacker can make one observation of I_F per call to F and that no measurement errors occur. Formally, a *side-channel* is a function $f_{I_F}\colon K \times M \to O$, where O denotes the set of possible observations. We assume that the attacker has full knowledge of the implementation I_F; in particular we assume that 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 ticks. 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 consumed by an invocation of I_F. This modeling also applies to software implementations of F on simple microprocessors without caches and pipelines.*

Example 2. *For implementations on microprocessors with performance-enhancing features such as pipelines and caches, the timing behavior of I_F is not necessarily a function of the inputs to F. However, if the processor is forced into a fixed initial state before each call to F (e.g., by filling up the pipeline and the cache), and if there are no interferences such as interrupts and preemptions, the execution time becomes a function of the inputs to F.*

Our modeling of side-channels also encompasses simple models of power consumption, e.g. the Hamming weight model [21]. However, these models abstract away certain electrical effects, and formal bounds derived on their basis do not imply security from attackers that exploit the omitted effects. The practical implications of applying our model to power analysis requires further investigation. In this paper, we focus on timing side-channels.

C. Unknown-Message Attacks

In a side-channel attack, a malicious agent gathers side-channel observations $f(k, m_1), \ldots, f(k, m_n)$ for deducing k or narrowing down its possible values. Depending on the attack scenario, the attacker might additionally be able to see or choose the messages $m_i \in M$: An attack is *chosen-message* if the attacker can choose $m_i \in M$; an attack is *known-message* if the attacker can observe but cannot influence the choice of $m_i \in M$; an attack is *unknown-message* if the attacker cannot observe $m_i \in M$.

As previously mentioned, all known timing attacks against RSA decryption require that the attacker be able to obtain a large number of pairs of ciphertexts and corresponding execution times. Input blinding randomizes each ciphertext before decryption, thus turning chosen- and known-message attacks into unknown-message attacks.

Example 3. *Consider an RSA decryption $x = m^k \mod N$, where m is a ciphertext chosen by the attacker, x is the plaintext, $N = p \cdot q$ is the modulus, and k with $k \cdot e = 1 \mod \varphi(N)$ is the secret key. In the* blinding *phase, one picks a random r that is relatively prime to N and computes $m \cdot r^e \mod N$. The result of the decryption is $(m \cdot r^e)^k = x \cdot r \mod N$, which yields x after* unblinding*, i.e., after multiplication with $r^{-1} \mod N$.*

Input blinding techniques are available for many common cryptographic algorithms, including ElGamal and Diffie-Hellman. While the mathematical details of these techniques depend on the algebraic properties of the individual cryptosystems, the notion of an unknown-message attack provides a common abstraction for attacks against blinded implementations.

D. Side-Channel Leakage in Unknown-Message Attacks

We review the information measure from [2] that captures the side-channel leakage in unknown-message attacks. For this, let $p_K\colon K \to \mathbb{R}$ and $p_M\colon M \to \mathbb{R}$ be probability distributions and let the random variables $\mathcal{K} = id_K$, $\mathcal{M} = id_M$ model the choice of keys and messages, respectively; we assume that p_M and p_K are known to the attacker.

For $n \in N$, we define the random variable $\mathcal{O}_n\colon K \times M^n \to O^n$ by $\mathcal{O}_n(k, m_1, \ldots, m_n) = (f(k, m_1), \ldots, f(k, m_n))$, where

$$p_{K \times M^n}(k, m_1, \ldots, m_n) = p_K(k) \prod_{i=1}^{n} p_M(m_i)$$

is the probability distribution on $K \times M^n$. The definition of the random variable $\mathcal{O}_n$ captures that k remains fixed over all invocations of f, while the messages $m_1, \ldots, m_n$ are chosen independently.

Example 4. *Consider again the blinded RSA ciphertext* $m \cdot r^e \mod N$ *from Example 3. If* m *is relatively prime to* N*, the multiplication with* m *and the exponentiation with* e *are permutations on* $\mathbb{Z}_N^\times$*. Hence, for independent and uniformly distributed values of* r*, the blinded inputs to the decryption are also independent and uniformly distributed. Note that choosing a ciphertext* m *that is not relatively prime to* N *corresponds to guessing one of the prime factors of* $N = p \cdot q$*. We can hence safely capture the effect of RSA input blinding by assuming a uniform distribution* p_M*.*

An unknown-message attacker making n side-channel observations $\mathcal{O}_n$ may learn information about the value of $\mathcal{K}$, i.e., about the secret key. This information can be expressed as the reduction in uncertainty about the value of $\mathcal{K}$, i.e., $I(\mathcal{K}; \mathcal{O}_n) = H(\mathcal{K}) - H(\mathcal{K}|\mathcal{O}_n)$.

Note that $I(\mathcal{K}; \mathcal{O}_n)$ captures only the information leaked by the time required for computing F, but does not capture the timing behavior of the blinding and unblinding steps. We assume that these steps do not introduce additional timing leaks.

III. Bounds on the Side-Channel Leakage

In this section, we prove our main result, which is an upper bound on the amount of information that an unknown-message attacker can extract from a side-channel in a given number of measurements. We give the proof idea in Section III-A and formalize it in Section III-B. In Section III-C, we give an interpretation of this bound in terms of the remaining effort for correctly guessing a key.

A. Proof Idea

The number of possible observations that can be made in n measurements is bounded from above by $|O|^n$. Hence, $\log_2(|O|^n) = n \log_2(|O|)$ is an upper bound for the information that is contained in n side-channel measurements. Note that this crude bound does not yet imply any useful security guarantees.

For a fixed key, however, the individual observations are independent and identically distributed. As a consequence, their ordering is irrelevant: only the relative frequency with which each observation occurs carries information about the key. The number of relative frequencies of observations, however, is bounded from above by $(n+1)^{|O|}$, which leads to our $|O| \log_2(n+1)$ upper bound on the leaked information.

B. Formal Proof

Formally, the *type* t_o of a sequence $o = (o_1, \ldots, o_n) \in O^n$ is the relative frequency with which each element of O occurs in o. Let $T_n = \{t_o \mid o \in O^n\}$ be the set of types of sequences of length n of elements of O. The function $\mathcal{T}_n \colon O^n \to T_n$ defined by $o \mapsto t_o$ is a random variable where $p_{\mathcal{T}_n}(t) = p_{\mathcal{O}_n}(\{o \in O^n \mid t_o = t\})$. We denote by $|t_o|$ the number of observation vectors of type t_o, i.e., $|t_o| = |\{o' \in O^n \mid t_{o'} = t_o\}|$.

The following bound on the size of T_n is fundamental to the (information-theoretic) method of types, see e.g. [10].

Lemma 1. $|T_n| \leq (n+1)^{|O|}$

Proof: Each type $t_o \in T_n$ can be represented by a vector in $\{0, \ldots, n\}^{|O|}$, where each component of the vector corresponds to the number of occurences of a particular observation in o. The assertion then follows from $|\{0, \ldots, n\}^{|O|}| = (n+1)^{|O|}$. ∎

Lemma 2 shows that a vector of observations o contains as much information (i.e., leads to the same reduction in uncertainty) about the key as the type t_o.

Lemma 2. $H(\mathcal{K}|\mathcal{O}_n = o) = H(\mathcal{K}|\mathcal{T}_n = t_o)$

Proof: $H(\mathcal{K}|\mathcal{O}_n = o)$ is defined by $-\sum_{k \in K} p_{\mathcal{K}|\mathcal{O}_n=o}(k) \log_2 p_{\mathcal{K}|\mathcal{O}_n=o}(k)$, hence it suffices to show that $p_{\mathcal{K}|\mathcal{O}_n=o} = p_{\mathcal{K}|\mathcal{T}_n=t_o}$. For a fixed k, the individual observations in $o = (o_1, \ldots, o_n)$ are independent, hence

$$p_{\mathcal{O}_n|K=k}(o) = \prod_{i=1}^{n} p_{\mathcal{O}_1|K=k}(o_i)\,.$$

In particular, this implies that $p_{\mathcal{O}_n|K=k}(o) = p_{\mathcal{O}_n|K=k}(o')$ whenever $t_o = t_{o'}$. Hence

$$\begin{aligned} p_{\mathcal{T}_n|\mathcal{K}=k}(t_o) &= \sum_{o' : t_o = t_{o'}} p_{\mathcal{O}_n|\mathcal{K}=k}(o') \\ &= |t_o|\, p_{\mathcal{O}_n|\mathcal{K}=k}(o)\,. \end{aligned}$$

We further have

$$\begin{aligned} p_{\mathcal{K}|\mathcal{T}=t_o}(k) &= \frac{p_{\mathcal{T}|\mathcal{K}=k}(t_o) p_{\mathcal{K}}(k)}{p_{\mathcal{T}}(t_o)} \\ &= \frac{p_{\mathcal{T}|\mathcal{K}=k}(t_o) p_{\mathcal{K}}(k)}{\sum_{k' \in K} p_{\mathcal{T}|\mathcal{K}=k'}(t_o) p_{\mathcal{K}}(k')} \\ &= \frac{|t_o| p_{\mathcal{O}_n|\mathcal{K}=k}(o) p_{\mathcal{K}}(k)}{\sum_{k' \in K} |t_o| p_{\mathcal{O}_n|\mathcal{K}=k'}(o) p_{\mathcal{K}}(k')} \\ &= \frac{p_{\mathcal{O}_n|\mathcal{K}=k}(o) p_{\mathcal{K}}(k)}{p_{\mathcal{O}_n}(o)} \\ &= p_{\mathcal{K}|\mathcal{O}_n=o}(k)\ , \end{aligned}$$

from which the statement follows. ∎

Lemma 3 shows that the random variable $\mathcal{O}_n$ contains as much information about the key as the random variable $\mathcal{T}_n$.

Lemma 3. $H(\mathcal{K}|\mathcal{O}_n) = H(\mathcal{K}|\mathcal{T}_n)$

Proof:

$$\begin{aligned} H(\mathcal{K}|\mathcal{O}_n) &= \sum_{o\in O^n} p_{\mathcal{O}_n}(o) H(\mathcal{K}|\mathcal{O}_n = o) \\ &= \sum_{t\in T_n} \sum_{o:t_o=t} p_{\mathcal{O}_n}(o) H(\mathcal{K}|\mathcal{O}_n = o) \\ &\overset{(*)}{=} \sum_{t\in T_n} p_{\mathcal{T}}(t) H(\mathcal{K}|\mathcal{T}_n = t) \\ &= H(\mathcal{K}|\mathcal{T}_n) \end{aligned}$$

where step $(*)$ follows from Lemma 2. ∎

We are now ready to give the proof of our main result, which is an upper bound on the amount of information that an unknown-message side-channel attacker can extract in a given number of measurements.

Theorem 1. *Let $f\colon K \times M \to O$ be a side-channel and let $\mathcal{K}$ and $\mathcal{O}_n$ be defined as above. Then we have*

$$I(\mathcal{K}; \mathcal{O}_n) \leq |O| \log_2(n+1) \ .$$

Proof:

$$\begin{aligned} I(\mathcal{K}; \mathcal{O}_n) &= H(\mathcal{K}) - H(\mathcal{K}|\mathcal{O}_n) && (1)\\ &= H(\mathcal{K}) - H(\mathcal{K}|\mathcal{T}_n) && (2)\\ &= H(\mathcal{T}_n) - H(\mathcal{T}_n|\mathcal{K}) && (3)\\ &\leq H(\mathcal{T}_n) && (4)\\ &\leq \log_2 |T_n| && (5)\\ &\leq |O| \log_2(n+1) \ , && (6) \end{aligned}$$

where (1) follows from the definition of the mutual information, (2) follows from Lemma 3, (3) follows from the chain rule for the conditional entropy, (4) follows from the non-negativity of the entropy, (5) is basic information theory, and (6) follows from Lemma 1. ∎

Theorem 1 characterizes the available side-channel information in terms of a bound on the expected number of leaked bits. Below, we give a characterization of this information in terms of the remaining effort for correctly guessing a key.

C. The Effort for Guessing Keys after a Side-channel Attack

In this section, we give a lower bound on the attacker's expected effort for guessing the correct key after n side-channel measurements.

The basis for our proof is a result by Massey [22] that shows that the average number of questions of the form "Does $\mathcal{X} = x$ hold" that must be asked to guess the value of a random variable $\mathcal{X}$ with $H(\mathcal{X}) \geq 2$ is bounded from below by $2^{H(\mathcal{X})-2} + 1$. We extend this bound to a bound in terms of the conditional Shannon entropy and apply the result to Theorem 1.

To this end, observe that the optimal guessing strategy is to try each of the possible values of $\mathcal{X}\colon A \to X$ in order of their decreasing probabilities. Without loss of generality, let X be indexed such that $p_{\mathcal{X}}(x_i) \geq p_{\mathcal{X}}(x_j)$, whenever $i \leq j$. Then the *guessing entropy* $G(\mathcal{X})$ of $\mathcal{X}$ is defined as $G(\mathcal{X}) = \sum_{1\leq i\leq |X|} i\, p_{\mathcal{X}}(x_i)$. One defines the *conditional guessing entropy* $G(\mathcal{X}|\mathcal{Y})$ as

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

The conditional guessing entropy captures the expected number of guesses needed to determine $\mathcal{X}$ when the value of $\mathcal{Y}$ is already known [5].

The following lemma generalizes Massey's entropic lower bound on the guessing entropy to a lower bound on the conditional guessing entropy.

Lemma 4. *Let $\mathcal{X}, \mathcal{Y}$ be random variables. Then*

$$G(\mathcal{X}|\mathcal{Y}) = 2^{H(\mathcal{X}|\mathcal{Y})-2} \ .$$

Proof: We have $G(\mathcal{X}) \geq 2^{H(\mathcal{X})-2} + 1$ whenever $H(\mathcal{X}) \geq 2$. Note that $G(\mathcal{X}) \geq 1$ always holds, hence we have $G(\mathcal{X}) \geq 2^{H(\mathcal{X})-2}$ without any restrictions on $H(\mathcal{X})$. We conclude

$$\begin{aligned} G(\mathcal{X}|\mathcal{Y}) &= E(G(\mathcal{X}|\mathcal{Y} = y) \\ &\geq E(2^{H(\mathcal{X}|\mathcal{Y}=y)-2}) \\ &\overset{(*)}{\geq} 2^{E(H(\mathcal{X}|\mathcal{Y}=y))-2} \\ &= 2^{H(\mathcal{X}|\mathcal{Y})-2} \ , \end{aligned}$$

where E denotes the expected value over $y \in Y$ and where $(*)$ follows from Jensen's inequality. ∎

The following corollary of Theorem 1 shows that the information contained in n side-channel measurements lowers the bound for the effort for guessing the correct key by a factor of $(n+1)^{|O|}$.

Corollary 1. *Let $\mathcal{K}$ and $\mathcal{O}_n$ be defined as above. Then*

$$G(\mathcal{K}|\mathcal{O}_n) \geq \frac{2^{H(\mathcal{K})}}{4\,(n+1)^{|O|}} \ .$$

Proof: We have

$$\begin{aligned} G(\mathcal{K}|\mathcal{O}_n) &\overset{(*)}{\geq} 2^{H(\mathcal{K}|\mathcal{O}_n)-2} \\ &\overset{(**)}{\geq} 2^{H(\mathcal{K})-|O|\log_2(n+1)-2} \ , \end{aligned}$$

where $(*)$ follows from Lemma 4 and $(**)$ follows from Theorem 1. ∎

The guessing entropy captures the average effort for determining a secret by exhaustive search. For expressing worst-case bounds, alternative measures such as the min-entropy are required [30]. We leave the investigation of min-entropic bounds for the side-channel leakage in unknown-message attacks to future work.

IV. An Adjustable Countermeasure Against Timing Attacks

A. Bucketing

Reducing the size of O improves the bound on the leaked information given by Theorem 1. In the case of timing, such a reduction can be achieved by a discretization of the possible execution times, which we call *bucketing*. Bucketing partitions a system's possible execution times into intervals (which we call *buckets*) of variable length, where, for each execution time, one waits until the enclosing bucket's upper bound before returning the result of the computation.

Clearly, bucketing leads to a performance penalty. In this section we show how, for a given $r \in \mathbb{N}$, one can compute a bucketing into r buckets such that the resulting performance penalty is minimal. Moreover, we show how, for a given $\epsilon > 0$, one can compute a bucketing with a minimal number of buckets, such that the performance penalty when applying this bucketing is below ϵ.

Our algorithms take as input a sample S of execution times of I_F and do not require precise knowledge about f_{I_F}. The accuracy of the estimated performance penalty, however, depends on the accuracy with which S approximates the average execution time of I_F (In Section V-C, we show how to estimate this accuracy). By contrast, the validity of the obtained security guarantee depends only on the correct number of buckets and does not rely on the accuracy of S. For the number of buckets to be correct, it suffices that S contains an upper bound for the worst-case execution time of I_F. For many embedded processors, such upper bounds can be obtained by automated worst-case execution time analysis [32].

B. Trading Security for Performance

In the remainder of this section, we assume that $O = \{o_1, \ldots, o_m\}$ is a set of positive reals that is indexed in ascending order, i.e. $o_i \leq o_j$ whenever $i < j$. We denote the set $\{o_1, \ldots, o_t\}$ by O_t.

A *sample* S is a multiset of observations. Formally, S is a function $S\colon O \to \mathbb{N}$, where $S(o)$ represents the number of occurrences of o in S. We denote the relative frequency with which $o \in O$ occurs in S by $p_S(o)$, i.e.,

$$p_S(o) = \frac{S(o)}{\sum_{o \in O} S(o)} \ .$$

We extend p_S to sets of observations in the natural way.

An *(r-)bucketing* B of O_t is a vector $B = (\hat{o}_1, \ldots, \hat{o}_r)$ of observations, with $\hat{o}_r = o_t$ and $\hat{o}_i \leq \hat{o}_j$ whenever $i < j$. Note that it would suffice to require that $\hat{o}_r \geq o_t$ and later prove that equality holds for optimal bucketings. We avoid this detour by definition.

For $i \in \{1, \ldots, r\}$, we define the *bucket* b_i by

$$b_i = \{o \in O \mid \hat{o}_{i-1} < o \leq \hat{o}_i\} \ ,$$

where we set $\hat{o}_0 = -1$. For a bucketing B of O_t and a sample S, we define the *average of S with respect to B* by

$$avg(S, B) = \sum_{i=1}^{r} \hat{o}_i \ p_S(b_i). \tag{7}$$

An r-bucketing B of O_t is *S-optimal* if $avg(S, B) \leq avg(S, B')$ for every r-bucketing B' of O_t.

In our application, S models a sample of execution times of the target system and B represents a sequence of time boundaries. If a computation takes time o, we wait for the minimal time boundary $\hat{o}_i$ with $\hat{o}_i \geq o$ until we return the result of the computation. This modified system has an average running time of $avg(S, B)$. The *performance penalty* $pen(S, B)$ resulting from applying a bucketing B of O to S is given by

$$pen(S, B) = avg(S, B) - \sum_{o \in S} o \ p_S(o) \ .$$

Example 5. *The sequence $(o_1, \ldots, o_m)$ is a bucketing of O with*

$$avg(S, (o_1, \ldots, o_m)) = \sum_{o \in O} o \ p_S(o) \, .$$

This value is the average value of the sample S, which leads to a performance penalty of 0 and illustrates that the execution time does not increase if the bucketing is as fine-grained as the set of possible observations.

Example 6. *The singleton sequence (o_m) is a 1-bucketing of O with*

$$avg(S, (o_m)) = o_m \, .$$

This reflects that a system with bucketing (o_m) requires time o_m for all inputs, and hence the average and the worst-case execution times coincide.

We consider the following two optimization problems related to bucketings.

Definition 1 (Minimal Performance Penalty Problem)**.** *Given a sample S of observations and $r \in \mathbb{N}$, the* Minimal Performance Penalty Problem *is to find an r-bucketing of O with minimal performance penalty with respect to S. Formally, the goal is to find an r-bucketing B of O, such that*

$$avg(S, B) \leq avg(S, B')$$

for all r-bucketings B' of O.

Definition 2 (Minimal Information Loss Problem)**.** *Given a sample S of observations and $\epsilon > 0$, the* Minimal Information Loss Problem *is to find a bucketing that minimizes the system's information loss through timing behavior under the constraint that the resulting performance penalty does not exceed ϵ . Formally, the goal is to find an r-bucketing B of O with minimal r and*

$$pen(S, B) \leq \epsilon \ . \tag{8}$$

Below, we present efficient algorithms for both problems.

C. Computing Optimal Bucketings

In this section, we propose efficient algorithms for the Minimal Performance Penalty Problem and for the Minimal Information Loss Problem stated above. For this, we leverage ideas from computing optimal histograms [16].

1) An Algorithm for the Minimal Performance Penalty Problem: The following observation is fundamental for our algorithm: Every S-optimal bucketing of O_i contains a S-optimal bucketing for some O_j with $j < i$. Formally, if the bucketing $B = (\hat{o}_1, \ldots, \hat{o}_r)$ is optimal for O_i, then the bucketing $(\hat{o}_1, \ldots, \hat{o}_{r-1})$ is optimal for O_j, where $o_j = \hat{o}_{r-1}$. Assume to the contrary that $(\hat{o}_1, \ldots, \hat{o}_{r-1})$ is not optimal for O_j. Then one can pick the S-optimal $(r-1)$-bucketing of O_j and add one bucket for the observations $o_{j+1}, \ldots, o_i$. The result is an r-bucketing B' of O_i, with $avg(S, B') < avg(S, B)$, which contradicts the optimality of B for S. A consequence of this so-called *optimal substructure property* is that we can use dynamic programming (see, e.g., [9]) to construct optimal bucketings of a set O_i from optimal bucketings of subsets $O_j \subseteq O_i$.

To this end, we define the function a, where the intuition is that $a(i, r)$ captures the value $avg(S, B)$ for an S-optimal r-bucketing B of O_i.

$$a(i, 1) = o_i \sum_{o \leq o_i} p_S(o)$$

$$a(i, r) = \min_{1 \leq j \leq i} a(j, r-1) + o_i \sum_{o_j < o \leq o_i} p_S(o)$$

The following lemma formalizes our intuition about a.

Lemma 5. *Let S be a sample and $B = (\hat{o}_1, \ldots, \hat{o}_r)$ be a bucketing of O_i. Then*

$$a(i, r) \leq avg(S, B). \tag{9}$$

Moreover, there exists an r-bucketing of O_i for which equality holds in (9).

Proof: We prove the assertion by induction on r. For $r = 1$, there is only one r-bucketing of O_i, namely (o_i). We have $a(i, 1) = o_i \sum_{o \leq o_i} p_S(o)$, which proves the assertion. For $r > 1$, it follows by induction that $a(j, r-1) = avg(S, B)$, where B is an S-optimal $(r-1)$-bucketing of O_j. By adding one bucket with boundary o_i and weight $\sum_{o_j < o \leq o_i} p_S(o)$, one can extend B to an r-bucketing B' of O_i with

$$avg(S, B') = avg(S, B) + o_i \sum_{o_j < o \leq o_i} p_S(o) \ .$$

Note that B' is not necessarily S-optimal for O_i. However, as previously observed, an S-optimal r-bucketing B^* of O_i contains an optimal $(r-1)$-bucketing of some O_{j^*} with $j^* < i$. By definition of $a(i, r)$, this j^* is found and, hence, $a(i, r) = avg(S, B^*)$. ∎

Lemma 5 implies that $a(|O|, r) = avg(S, B)$ for an optimal r-bucketing B of O. For computing a, one can build up a table of size $|O| \times r$. For computing each entry, at most $|O|$ lookups are necessary (The values $\sum_{o \leq o_i} p_S(o)$ can be precomputed for $i \in \{1, \ldots, |O|\}$). A concrete bucketing can then be obtained from this table by backtracking, which yields the following proposition.

Proposition 1. *For a sample S and a number of buckets r, the Minimal Performance Penalty Problem can be solved in time $\mathcal{O}(r\,|O|^2)$.*

As we show next, a similar approach can be taken for solving the Minimal Information Loss Problem.

2) An Algorithm for the Minimal Information Loss Problem: We present an algorithm for the Minimal Information Loss Problem. For this, we re-use the function a defined in Section IV-C1. As before, we use dynamic programming to build a value-table representation of a, where the difference to the algorithm for

```
1   procedure Exp(x, k, N)
2   input x, k, N ∈ ℕ
3   output x^k mod N
4   begin
5       p ← MontMul(1, 1)
6       x ← MontMul(x, 1)
7       for i = n − 1 downto 0 do
8           p ← MontMul(p, p)
9           if k[i] = 1 then
10              p ← MontMul(p, x)
11      od
12      p ← MontMul(p, R^2)
13      return p
14  end.
```

Figure 1. Pseudo-code of the exponentiation algorithm, where **MontMul** stands for Montgomery Multiplication. The operations in lines 5, 6, and 12 transform the operands to Montgomery form and back, respectively, and the loop in lines 7–11 performs the actual exponentiation.

the Minimal Performance Penalty Problem is that the minimal length r of a bucketing B with $pen(S, B) \leq \epsilon$ is not known a priori. However, as Example 5 shows, there is a bucketing of length $|O|$ with a performance penalty of 0. Hence r is bounded from above by $|O|$, which leads to an $\mathcal{O}(|O|^3)$-algorithm.

Proposition 2. *For a sample S and $\epsilon > 0$, the Minimal Information Loss Problem can be solved in time $\mathcal{O}(|O|^3)$.*

This direct approach to the Minimal Information Loss Problem is efficient enough for our experiments, however, it is likely that the complexity bounds can be further improved, e.g., along the lines of [16].

V. Experimental Results

In this section, we perform a case study where we evaluate the influence of bucketing on the performance and the security of a realistic example. To this end, we apply the algorithms presented in Section IV to compute optimal bucketings for an implementation of 1024 bit RSA decryption.

A. RSA Implementation and Timing Model

We consider the RSA implementation in Figure 1, in which the modular exponentiation is performed by square-and-multiply, and where each modular multiplication is carried out using Montgomery's algorithm [26], which is a common choice in practice. It is important to note that Montgomery multiplication is not constant-time, because of so-called *extra reductions* that must be performed for some operands.

Our analysis is based on a timing model that captures a simple 32-bit microprocessor that can perform word-level additions and multiplications within single clock cycles. Using standard algorithms for multi-precision arithmetic [23], a Montgomery multiplication of two 1024-bit integers can then be performed within (roughly) $(1024/32)^2 = 1024$ clock cycles, and an extra reduction step can be performed within (roughly) $1024/32 = 32$ clock cycles. We hence assume that $t_{mul} = 1024$ and $t_{red} = 32$, where t_{mul} and t_{red} are the numbers of clock cycles consumed by a multi-precision multiplication and an extra reduction step, respectively.

An upper bound t_{max} on the worst-case execution time of our implementation is given by

$$t_{max} = (2 \cdot 1024 + 3)(t_{mul} + t_{red}) = 2\,165\,856\ ,$$

which corresponds to an execution in which all possible multiplications and extra reductions are carried out.

B. The Effect of Bucketing on Security and Performance

The black curve in Figure 2 shows the distribution of the execution times of our RSA implementation, based on 2^{19} randomly sampled inputs. The average execution time of this implementation is $1\,586\,211$ clock ticks.

The (centers of the) gray bars and the dashed vertical line depict the boundaries of an optimal 5-bucketing computed with our algorithm, i.e, a solution to the Minimal Performance Penalty Problem. The average execution time for the system with this bucketing is $1\,596\,726$ clock ticks, which corresponds to an overhead of only 0.67% with respect to the implementation without bucketing.

Theorem 1 shows that a 5-bucketing is coarse enough for obtaining meaningful security guarantees: At least $2^{200/5}$ timing measurements are necessary for learning an expected amount of 200 key bits. Alternatively, Corollary 1 shows that $2^{200/5}$ timing measurements reduce the lower bound for the attacker's expected effort for correctly guessing the key by a factor of at most 2^{200}. A modern Smartcard such as Infineon's SLE 66CX366PE can perform an RSA decryption in 26ms [15] (see also [13]). For such a system, the time for performing these 2^{40} measurements amounts to more than 900 years.

We conclude that adding bucketing to a blinded implementation implies only a minor performance overhead and yields meaningful formal security guarantees.

Figure 3 depicts how the average execution time decreases as the number of buckets grows. This decrease is most notable between bucketings of 1 and 2 buckets,

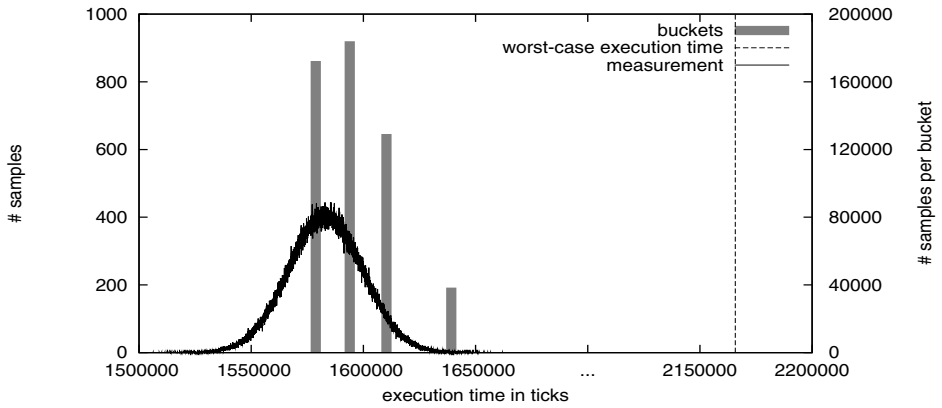

Figure 2. Execution time in clock cycles of 1024-bit RSA exponentiation for 2^{19} samples. The mean for the original distribution is 1 586 211, the mean for the bucketized distribution is 1 596 726.

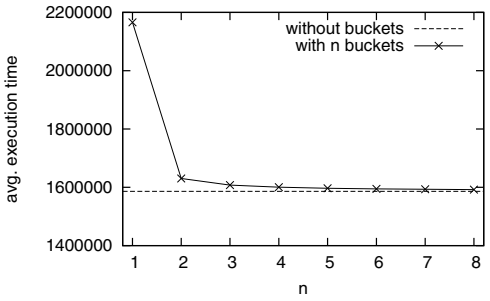

Figure 3. Expected execution time vs. number of buckets for 1024-bit RSA exponentiation for 2^{19} samples.

which illustrates the performance gain of a 2-bucketing with respect to a constant-time implementation. Observe that this strong performance gain does not give the whole picture, because our analysis does not consider the overhead introduced by the blinding and unblinding steps. If the mask r^e can be precomputed, however, this overhead will be small.

C. Error Estimations

When computing bucketings, we approximate the distribution of side-channel observations by a sample S of timing measurements. As discussed in Section IV, the quality of this approximation does not affect the obtained security guarantees. However, the quality of the approximation affects the precision of the estimated performance penalty. We next show how this precision can be determined.

Formally, we derive a bound on

$$|\,pen(S,B) - pen(S_{\mathcal{O}},B)|\ ,$$

where $S_{\mathcal{O}}$ denotes an "idealized sample" of $\mathcal{O}_1$, i.e. a sample with $p_{S_{\mathcal{O}}} = p_{\mathcal{O}_1}$. To this end, let $B = (\hat{o}_1,\ldots,\hat{o}_r)$, let

$$\Delta(p_{\mathcal{O}_1},p_S) = \frac{1}{2}\sum_{x\in O}|p_{\mathcal{O}_1}(x) - p_S(x)|$$

denote the statistical distance of $p_{\mathcal{O}_1}$ and p_S, and let E denote the expected value. We have

$$\begin{aligned}
&|\,pen(S,B) - pen(S_{\mathcal{O}},B)| \\
&= \left|\left(\sum_{i=1}^{r}\hat{o}_i \sum_{\hat{o}_{i-1}<x\le\hat{o}_i} p_S(x) - E(p_S)\right)\right. \\
&\qquad \left. - \left(\sum_{i=1}^{r}\hat{o}_i \sum_{\hat{o}_{i-1}<x\le\hat{o}_i} p_{\mathcal{O}_1}(x) - E(p_{\mathcal{O}_1})\right)\right| \\
&\le \sum_{i=1}^{r}\hat{o}_i \sum_{\hat{o}_{i-1}<x\le\hat{o}_i} |p_S(x) - p_{\mathcal{O}_1}(x)| \\
&\qquad + |E(p_S) - E(p_{\mathcal{O}_1})| \\
&\le 4\cdot\hat{o}_r\cdot\Delta(p_S,p_{\mathcal{O}_1})\ .
\end{aligned}$$

For our example, we instantiate $\hat{o}_r$ with the worst-case execution time of 2 165 856 clock cycles. We have computed upper bounds on $\Delta(p_S,p_{\mathcal{O}_1})$ in terms of the size of O, which we estimate as follows: All variations in the execution time are multiples of 32 clock cycles, hence it suffices to count the number of variations. Only the multiplication in line 10 and the extra reductions in lines 5, 6, 8, 10, and 12 account for potential variations. Summing up, we see that

$$|O| \le 1024\cdot(32+1+1)+3 = 34\,819\ .$$

Our computation of bounds on the statistical distance of $p_{\mathcal{O}_1}$ and p_S in terms of $|O|$ uses Chernoff bounds and the McDiarmid Inequality. The actual computation is tedious and we omit it for better readability. The result

is that a sample of 2^{19} execution times is sufficient to guarantee an approximation error of less than 0.01 with a confidence of more than 0.99. With this, we obtain

$$|\, pen(S, B) - pen(S_{\mathcal{O}}, B)| < 64\,976$$

clock cycles, which amounts to less than 4.1% of the expected execution time. Note that these error bounds depend only on $|O|$ and $\hat{o}_r$ and hold for arbitrary systems with these parameters. The timing behavior of our case-study shows strong regularity, and we expect that this can be exploited for deriving much tighter error bounds.

VI. Related Work

Our results are based on the model of side-channels from [19] and the measure for information-flow in unknown-message attacks from [2]. The exact computation of this measure for a given system requires the enumeration of all possible inputs and does not scale. Our new result implies that, by combining bucketing and blinding, such a costly analysis can be entirely avoided.

A number of quantitative information-flow measures have been proposed in the literature, e.g. [20], [7], [6]. The measure proposed by Clark et al. [6] is closely related to the measure used in this paper; however, it does not capture multiple computations with the same key and is hence not applicable to the analysis of side-channel attacks.

Several approaches in language-based security use type systems to detect [14] or eliminate timing side-channels[1], [3]. If a program successfully type-checks, then an attacker cannot gain any information about the secret, even if he exhaustively runs the program on all possible public inputs. A quantitative, language-based approach to mitigate timing attacks is proposed in [28]. The mitigation relies on probabilistic padding, which also leads to a trade-off between security and performance. Applying these language-based approaches requires restrictive programming and precise knowledge about the time consumption of the individual instructions on the underlying machine. By contrast, our approach only requires that the system's execution time be a function of the inputs to the system.

Code transformations to eliminate the influence of secrets on the control flow of a program have been proposed in [25], [8], and they have been applied to eliminate timing leaks in code for modern multi-purpose processors. As shown in [25], the resulting performance overhead can be large compared to that introduced by blinding (and hence to that introduced by our countermeasure).

Standaert et al. propose a framework for the evaluation of side-channel attacks [31], where they use two largely independent metrics for the evaluation of systems. The information-theoretic metric captures 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 of applying a given key recovery strategy to the measurement data. In this way, an analysis with the model of [31] yields assertions about the effectiveness of a particular kind of attack, but not necessarily universal bounds.

Micali and Reyzin [24] propose *physically observable cryptography*, a mathematical model that aims at providing provably secure cryptography on hardware that is only partially shielded. Using a similar approach, Dziembowski and Pietrzak [11] recently obtained the first positive results, where they construct a stream cipher that is provably secure in the presence of arbitrary leakage functions with output of logarithmic length (assuming ordinary PRFs) or of a constant fraction of state-bits (assuming the existence of exponentially hard PRFs). It will be interesting to see how efficient their constructions work in practice, and whether they extend to public-key cryptography. Our approach is more concrete than that of [11] in that we prove the security of a specific countermeasure against timing attacks. This countermeasure, however, can readily be applied to existing cryptosystems.

The term *bucketing* and the idea of computing optimal bucketings by dynamic programming is inspired by work on computing optimal histograms of probability distributions [16]. In [16], the buckets partition a set of values (which corresponds to the observations in our model), where each value has a given frequency (which corresponds to the number of occurences in a sample). However, there is no direct correspondence between the notions of optimality of bucketings in [16] and in our approach, which requires our development in Section IV-C.

The discretization of the execution time into multiples of a fixed time quantum has been proposed in [4]. However, it was observed that, for obtaining security guarantees, the execution time for all all decryption operations must lie within a single time quantum. This use of discretization corresponds to the requirement that the decryption operation is constant-time. Bucketing weakens this requirement and hence improves the performance of the system. Our results show that this improvement can be achieved while retaining formal security guarantees.

VII. Conclusions

We have presented a provably secure and efficient countermeasure against timing attacks, where the strength of the obtained security guarantee can be freely traded for the resulting performance overhead. We have given algorithms that efficiently and optimally adjust this trade-off for given constraints on the side-channel leakage or on the efficiency of the cryptosystem. Finally, we have performed a case-study that shows that our countermeasure leads to implementations with minor performance overhead and meaningful security guarantees.

Our results can be directly applied to implementations of cryptographic algorithms on systems with deterministic timing behavior, such as simple embedded systems and cryptographic coprocessors, and we believe that our work can have practical impact on this important application domain.

As future work, we will investigate whether our results extend to systems with nondeterministic timing behavior. In particular, we plan to investigate the impact of concurrency and uncertain initial cache state on the information leakage of blinded implementations. Finally, any improvement on the information-theoretic bounds will have a direct impact on the security guarantees delivered by our countermeasure.

References

[1] Johan Agat. Transforming out timing leaks. In *Proc. POPL '00*, pages 40–53. ACM, 2000.

[2] Michael Backes and Boris Köpf. Formally bounding the side-channel leakage in unknown-message attacks. In *Proc. ESORICS '08*, volume 5283 of *LNCS*, pages 517–532. Springer, 2008.

[3] Gilles Barthe, Tamara Rezk, and Martijn Warnier. Preventing timing leaks through transactional branching instructions. In *Proc. QAPL '05*, ENTCS, pages 33–55. Elsevier, 2005.

[4] Dan Boneh and David Brumley. Remote timing attacks are practical. In *Proc. USENIX Security '03*, 2003.

[5] Christian Cachin. Entropy measures and unconditional security in cryptography. PhD thesis, ETH Zürich, 1997.

[6] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative information flow, relations and polymorphic types. *J. Log. Comput.*, 18(2):181–199, 2005.

[7] M.R. Clarkson, A.C. Myers, and F.B. Schneider. Belief in information flow. In *Proc. CSFW '05*, pages 31– 45. IEEE, 2005.

[8] Bart Coppens, Ingrid Verbauwhede, Koen De Bosschere, and Bjord De Sutter. Practical mitigation for timing-based side-channel attacks on modern x86 processors. In *Proc. IEEE SSP '09 (to appear)*. IEEE, 2009.

[9] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, second edition, 2001.

[10] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, second edition, 2006.

[11] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *Proc. FOCS '08*, 2008.

[12] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Proc. CHES '01*, volume 2162 of *LNCS*, pages 251–261. Springer, 2001.

[13] Helena Handschuh and Pascal Paillier. Smart card crypto-coprocessors for public-key cryptography. In *Proc. CARDIS 2000*, volume 1820 of *LNCS*, pages 386–394. Springer, 2000.

[14] Daniel Hedin and David Sands. Timing aware information flow security for a JavaCard-like Bytecode. In *BYTECODE '05*, ENTCS. Elsevier, 2005.

[15] Infineon. Datasheet for the SLE 66CX366PE Chip Card & Security ICs. Online at http://www.infineon.com.

[16] H. V. Jagadish, Viswanath Poosala, Nick Koudas, Ken Sevcik, S. Muthukrishnan, and Torsten Suel. Optimal histograms with quality guarantees. In *Proc. VLDB*, pages 275–286, 1998.

[17] Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Proc. CRYPTO '96*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

[18] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Proc. CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.

[19] Boris Köpf and David Basin. An information-theoretic model for adaptive side-channel attacks. In *Proc. CCS '07*, pages 286–296. ACM, 2007.

[20] Gavin Lowe. Quantifying information flow. In *Proc. CSFW '02*, pages 18–31. IEEE, 2002.

[21] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[22] James L. Massey. Guessing and Entropy. In *Proc. IEEE Int. Symp. on Info. Th. '94*, page 204. IEEE, 1994.

[23] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[24] Silvio Micali and Leonid Reyzin. Physically observable cryptography (Extended Abstract). In *Proc. TCC '04*, volume 2951 of *LNCS*, pages 278–296. Springer, 2004.

[25] David Molnar, Matt Piotrowski, David Schultz, and David Wagner. The program counter security model: Automatic detection and removal of control-flow side channel attacks. In *Proc. ICISC '05*, volume 3935 of *LNCS*, pages 156–168. Springer, 2006.

[26] Peter Montgomery. Multiplication without trial division. *Math. Computation*, 44:519–521, 1985.

[27] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: the case of AES. In *Proc. CT-RSA '06*, volume 3860 of *LNCS*, pages 1–20. Springer, 2006.

[28] Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Quantifying timing leaks and cost optimisation. In *Proc. ICICS '08*, volume 5308 of *LNCS*, pages 81–96. Springer, 2008.

[29] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and couter-measures for smard cards. In *Proc. E-smart '01*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.

[30] Geoffrey Smith. On the foundations of quantitative information flow. In *Proc. FoSSaCS '09*, volume 3935 of *LNCS 5504*, pages 288–302. Springer, 2009.

[31] Francois-Xavier Standaert, Tal G. Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. Cryptology ePrint Archive, Report 2006/139, 2006.

[32] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, and Per Stenström. The worst-case execution-time problem—overview of methods and survey of tools. *Trans. on Embedded Computing Sys.*, 7(3):1–53, 2008.