Non-Uniform Distributions in Quantitative Information-Flow

Michael Backes
Saarland University &
MPI-SWS
Saarbrücken, Germany
backes@mpi-sws.org

Matthias Berg
Saarland University
Saarbrücken, Germany
berg@cs.uni-saarland.de

Boris Köpf
IMDEA Software
Madrid, Spain
boris.koepf@imdea.org

ABSTRACT

Quantitative information-flow analysis (QIF) determines the amount of information that a program leaks about its secret inputs. For this, QIF requires an assumption about the distribution of the secret inputs. Existing techniques either consider the worst-case over a (sub-)set of all input distributions and thereby over-approximate the amount of leaked information; or they are tailored to reasoning about uniformly distributed inputs and are hence not directly applicable to non-uniform use-cases; or they deal with explicitly represented distributions, for which suitable abstraction techniques are only now emerging. In this paper we propose a novel approach for a precise QIF with respect to non-uniform input distributions: We present a reduction technique that transforms the problem of QIF w.r.t. non-uniform distributions into the problem of QIF for the uniform case. This reduction enables us to directly apply existing techniques for uniform QIF to the non-uniform case. We furthermore show that quantitative information flow is robust with respect to variations of the input distribution. This result allows us to perform QIF based on approximate input distributions, which can significantly simplify the analysis. Finally, we perform a case study where we illustrate our techniques by using them to analyze an integrity check on non-uniformly distributed PINs, as they are used for banking.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Information flow controls*; H.1.1 [**Models and Principles**]: Systems and Information Theory—*Information theory*

General Terms

Security

Keywords

Quantitative information flow

1. INTRODUCTION

The goal of an information-flow analysis is to keep track of sensitive information during computation. If a program does not expose any information about its secret inputs to unauthorized parties, it has secure information flow, a property that is often formalized as noninterference. In many cases, achieving noninterference is expensive, impossible, or simply unnecessary: Many systems remain secure as long as the amount of exposed secret information is sufficiently small. Consider for example a password checker. A failed login attempt reveals some information about the secret password. However, for well-chosen passwords, the amount of leaked information is so small that a failed login-attempt will not compromise the security of the system.

Quantitative information-flow analysis (QIF) is a technique for establishing bounds on the information that is leaked by a program. The insights that QIF provides go beyond the binary output of Boolean approaches, such as non-interference analyzers. This makes QIF an attractive tool to support gradual development processes, even without explicitly specified policies. Furthermore, because information-theory forms the foundation of QIF, the quantities that QIF delivers can be directly associated with operational security guarantees, such as lower bounds for the expected effort of uncovering secrets by exhaustive search.

Technically, a quantitative information-flow analysis requires an assumption about the probability distribution of the confidential inputs. Existing approaches deal with this assumption in four fundamentally different ways. We briefly present all four alternatives and discuss their implications on applicability and automation of quantitative information-flow analyses.

The first kind of approach focuses on computing the channel capacity, which is the maximum leakage with respect to all possible input distributions [7,8,17,23,24,27,31]. Maximizing over all possible input distributions is a safe, but often overly pessimistic assumption: Consider a password checker with two possible observable outcomes, `succeed` and `fail`. The capacity of the channel from secret passwords to observables is 1 bit, corresponding to a distribution that assigns probability 0.5 to both outcomes. A naive security analysis will infer that an n-bit password can be leaked in as few as n login attempts and conclude that the system is insecure. However, if the passwords are well-chosen (e.g. drawn uniformly from a large set), a login attempt will reveal much less than one bit of information, which is the reason why the password checker is in fact secure.

The second kind of approach considers the maximum leakage with respect to a subset of possible input distributions, where the subset is specified in terms of bounds on the entropy of the input variables [9, 11]. While entropy bounds are an attractive way of specifying interesting subsets of input distributions, precise reasoning about the leakage of programs in terms of such bounds turns out to be challenging. In particular, deriving tight bounds for the leakage of programs with loops in terms of entropy bounds for their

ASIACCS '11, March 22–24, 2011, Hong Kong, China.

input variables is still an open problem.

The third kind of approach analyzes programs with respect to uniformly distributed inputs [2, 20, 22]. Under the uniformity assumption, computing the information-theoretic characteristics of deterministic programs can be reduced to computing the programs' preimages and determining their numbers or sizes [20]. It has been shown that these tasks can be performed using symbolic [2] and randomized algorithms [22], allowing one to analyze large state-spaces with precision guarantees. However, the applicability of those techniques has so far been restricted to domains with uniformly distributed inputs.

Finally, the fourth kind of approach analyzes programs with respect to an arbitrary, but fixed, probability distribution on the secret inputs [12, 25, 30]. The first automated approach delivers precise results [30], but is limited to programs with small state-spaces due to the explicit representation of the input distribution. An abstraction technique [29] addresses this problem by partitioning the (totally ordered) domain into intervals, on which a piecewise uniform distributions is assumed. However, it is an open problem how to choose the initial partition of the domain in order to allow for an analysis that is precise and efficient at the same time.

In summary, it has been an open problem to perform quantitative information-flow analysis with non-uniform distributions in a precise, scalable, and general way. In this paper, we make the following contributions towards this goal.

Our first contribution is a technique for reducing the problem of QIF with respect to non-uniform distributions to the problem of QIF with respect to uniform distributions. Our reduction enables one to directly apply existing tools for uniform QIF [2, 20, 22] to the non-uniform case. The main idea of the reduction is to represent a non-uniform distribution in terms of a generator program that receives uniformly distributed input and produces outputs according to the desired distribution. We exhibit and prove a connection between the information-flow properties of the target program and the sequential composition of the target program with the generator program. This connection enables us to analyze the composed program with respect to uniform inputs, and to draw conclusions about the information flow of the target program with respect to the non-uniform distribution. Our reduction is motivated by a number of examples that occur in practice. For example, the (non-uniformly distributed) PINs used in electronic banking are derived from uniform bit-strings, e.g., using decimalization techniques [13]. Another example are the keys of a public-key cryptosystem, which are typically produced by key generation algorithms that operate on uniformly distributed input.

Our second contribution is to show that QIF is robust with respect to small variations in the input distribution. This allows us to replace actual input distributions with approximate distributions. Based on the quality of the approximation, we give upper and lower bounds on the error this approximation introduces in the analysis. Focusing on approximate distributions can simplify the information-flow analysis, e.g. by allowing to replace "almost uniform" distributions by uniform distributions.

Finally, we give examples of how our two results can be used for the quantitative information-flow analysis of realistic systems. We use our reduction technique to estimate the information leaked by an integrity check on non-uniformly distributed PINs, and we use our robustness result to bound the error that is introduced by assuming uniformly distributed PINs.

The paper is organized as follows. In Section 2 we introduce basic notions of information flow. The reduction of non-uniform analysis to the uniform case is shown in Section 3. The robustness results are presented in Section 4. Section 5 contains our experiments where we apply our results to analyze an integrity check on non-uniformly distributed PINs. We present related work in Section 6 and conclude in Section 7.

2. PRELIMINARIES

2.1 Programs

A program $P = (I, F, R)$ is a triple consisting of a set of *initial states* I, a set of *final states* F, and a transition relation $R \subseteq I \times F$. We consider programs that implement total functions, i.e., we require that for all $s \in I$ there is exactly one $s' \in F$ with $(s, s') \in R$, and we use the shorthand notation $P(s) = s'$ for $(s, s') \in R$.

Given a final state $s' \in F$, we define its *preimage* $P^{-1}(s')$ to be the set of all input states from which s' is reachable, i.e.,

$$P^{-1}(s') \quad \equiv \quad \{s \mid (s, s') \in R\} \,.$$

The preimage of an unreachable state is the empty set.

2.2 Qualitative Information Flow

We assume that the initial state of each computation is secret. We consider an attacker that knows the program, in particular its transition relation, and the final state of each computation.

We characterize partial knowledge about the elements of I in terms of *partitions* of I, i.e., in terms of a family $\{B_1, \ldots, B_r\}$ of pairwise disjoint *blocks* such that $\bigcup_{i=1}^{r} B_i = I$. A partition of I models that each $s \in I$ is known up to its enclosing block B_i. We compare partitions using the (im-)precision order $\sqsubseteq$ defined by

$$\begin{aligned} &\{B_1, \ldots, B_r\} \sqsubseteq \{B'_1, \ldots, B'_{r'}\} \\ &\quad \equiv \quad \forall i \in \{1, \ldots, r\}\ \exists j \in \{1, \ldots, r'\} : B_i \subseteq B'_j \,. \end{aligned}$$

The knowledge gained by an attacker about initial states of computations of the program P by observing their final states is given by the partition Π that consists of the preimages of reachable final states, i.e.,

$$\Pi \quad \equiv \quad \{P^{-1}(s') \mid s' \in F\} \,.$$

The partition $\{I\}$ consisting of a single block corresponds to the case where no information leaks, and $\{\{s\} \mid s \in I\}$ where each block is a singleton set captures the case that P fully discloses its input. Partitions Π with $\{\{s\} \mid s \in I\} \sqsubset \Pi \sqsubset \{I\}$ capture that P leaks partial information about its input.

More generally, one can assume that initial and final states are pairs of *high* and *low* components, i.e., $I = I_H \times I_L$ and $F = F_H \times F_L$, and that the observer can access the low components of the initial and final states of a given computation. For a low input l and a low output l' we then define the *low-preimage* $P_l^{-1}(l')$ of l' to be the set of all high components of input states with low component l, from which a final state with low component l' is reachable, i.e.

$$P_l^{-1}(l') \quad \equiv \quad \{h \mid \exists h' \in F_H : \ ((h, l), (h', l')) \in R\} \,.$$

As before, we can characterize the knowledge an attacker gains about the high components of initial states states in terms of a partition of I_H. However, the exact shape of this partition strongly depends on the role of the low input. For example, when the low input is controlled by an attacker who can exhaustively run the program with all possible low inputs, then the knowledge the attacker gains about the high input is characterized by partition corresponding to the intersection of all low-preimages, i.e.,

$$\Pi \quad \equiv \quad \bigcap_{l \in I_L} \{P_l^{-1}(l') \mid l' \in F_L\} \,,$$

where the intersection of partitions Π_1, Π_2 is defined by pairwise intersection of their blocks, i.e. $\Pi_1 \cap \Pi_2 = \{A \cap B \mid A \in \Pi_1, B \in \Pi_2\}$.

Several approaches in the literature consider weaker attackers, e.g. those that run the program with a single, fixed low input [25], or those that can observe a bounded number of program runs with adaptively chosen low inputs [20]. While the precise definition of Π depends on the considered attacker model, the characterization of attacker knowledge in terms of a partition (or an equivalence relation) is universal.

The results of this paper rely only on such a partition-based characterization of attacker knowledge and can hence be used in conjunction with all of the aforementioned attacker models. For the sake of presentation, we focus on the simplest scenario, namely programs in which the entire initial state is high, and the entire final state is low (and hence $\Pi = \{P^{-1}(s') \mid s' \in F\}$).

2.3 Quantitative Information Flow

We use information theory to characterize the information that P reveals about its input. This characterization has the advantage of being compact and easy to compare. Moreover, it yields concise interpretations in terms of the effort needed to determine P's input from the revealed information, e.g., by exhaustive search.

We begin by introducing necessary notation. Let A be a finite set and $p\colon A \to \mathbb{R}$ a probability distribution. For a random variable $X\colon A \to B$, we define $p_X\colon B \to \mathbb{R}$ as $p_X(x) = \sum_{a \in X^{-1}(x)} p(a)$, which we will also denote by $\Pr(X = x)$.

The *(Shannon) entropy* [34] $H(X) = -\sum_{x \in B} p_X(x) \log_2 p_X(x)$ of X 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 questions with binary outcome that need to be asked to determine X's value [6]. Given another random variable $Y\colon A \to C$, we write $H(X|Y = y)$ for the entropy of X given that the value of Y is y. The *conditional entropy* $H(X|Y)$ of X given Y is the expected value of $H(X|Y = y)$ over all $y \in C$; it captures the remaining uncertainty about X when Y is observed.

For analyzing programs, we assume a probability distribution p on I and we suppose that it is known to the attacker. For analyzing the program P, we define two random variables. The first random variable $D\colon I \to I$ models the choice of an input in I, i.e., D is the identity $D(s) = s$. The second random variable captures the input-output behavior of P. We overload notation and also denote it by P. Formally, we define $P\colon I \to F$ by $P(s) = s'$ whenever $(s, s') \in R$.

The conditional entropy $H(D|P)$ captures the remaining uncertainty about the program's input when the output is observed. We will use $H(D|P)$ as a measure of information flow in this paper, because it is associated with operational security guarantees: one can give lower bounds for the effort for determining a secret by exhaustive search in terms of $H(D|P)$, see [21, 26].

We mention for completeness that several approaches in the literature (e.g. [11, 25, 30]) focus on computing the *mutual information* $I(D; P)$ between the input and the output of a program, which is defined as the reduction in uncertainty about the input when the output is observed, i.e. $I(D; P) = H(D) - H(D|P)$. For given $H(D)$, the value of $I(D; P)$ can be immediately be derived from that of $H(D|P)$, and vice versa. We present our results in terms of the remaining uncertainty $H(D|P)$ because of its more direct connection to operational security guarantees.

3. NON-UNIFORM QUANTITATIVE INFORMATION-FLOW ANALYSIS

In this section we first show how to reduce the problem of QIF w.r.t. non-uniform distributions to the problem of QIF w.r.t. uniform distributions. We then show how this reduction allows us to leverage existing QIF techniques for programs with uniformly distributed inputs for the QIF-analysis of programs with arbitrarily distributed inputs.

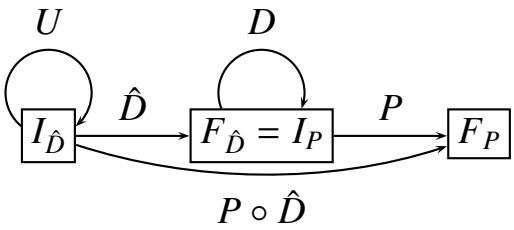

Figure 1: Overview of the random variables required for reducing non-uniform QIF analysis to the uniform case. The input to P is distributed according to the variable D. The input to $\hat{D}$ is given by the uniformly distributed random variable U; the output distribution of $\hat{D}$ matches that of D.

3.1 Reducing the Non-uniform Case to the Uniform Case

The main idea behind our reduction is to express a probability distribution p as a program $\hat{D}$ that takes input that is uniformly distributed and produces output that is distributed according to p. We prove an assertion that connects the remaining uncertainty about the inputs of the sequentially composed program $\hat{D}; P$ (with respect to uniform distributions) to the remaining uncertainty about the inputs of the program P with respect to the distribution p.

Our reduction is motivated by a number of examples that occur in practice. For example, the Personal Identification Numbers (PINs) used in electronic banking are often not uniformly distributed, but derived from uniform bitstrings using decimalization techniques [13] (We will apply our techniques to analyze a program handling such PINs in Section 5). Another example are the keys of a public-key cryptosystem, which are typically not uniformly distributed bitstrings. However, they are produced by a key generation algorithm that operates on uniformly distributed input. More generally, a large number of randomized algorithms expect uniformly distributed randomness. E.g. in complexity theory, randomized algorithms are based on probabilistic Turing machines that work with uniformly distributed random tapes. For a language-based perspective on distribution generators, see [32].

Formally, let $P = (I_P, F_P, R_P)$ be a program and p an arbitrary distribution on I_P. Let $\hat{D} = (I_{\hat{D}}, F_{\hat{D}}, R_{\hat{D}})$ be a program that maps to P's initial states, i.e., $F_{\hat{D}} = I_P$, and let u be the uniform distribution on $I_{\hat{D}}$. We require that the distribution produced by $\hat{D}$ matches the distribution on P's inputs, i.e., $u_{\hat{D}} = p$. We define the random variables D and U as the identity functions on I_P and $I_{\hat{D}}$, respectively, and we use them for modeling the choice of an input according to p and u, respectively. Figure 1 depicts these mappings and their connections.

The setup is chosen such that the uncertainty about the output of the composed program $P \circ \hat{D}$ matches the uncertainty about the output of P, i.e. $H(P \circ \hat{D}) = H(P)$. Similarly, we have $H(\hat{D}) = H(D)$. As a consequence, we can express the remaining uncertainty about the input of P in terms of a difference between the remaining uncertainties about the (uniformly distributed) inputs of $P \circ \hat{D}$ and $\hat{D}$.

Lemma 1. *Let $P, \hat{D}, D, U$ be as defined in Section 3.1. Then*

$$H(D|P) = H(U|P \circ \hat{D}) - H(U|\hat{D}).$$

Proof. The output of P is determined by the output of D, namely $H(P, D) = H(D)$. Therefore it holds $H(D|P) = H(D) - H(P)$, and hence by construction

$$H(D|P) = H(\hat{D}) - H(P \circ \hat{D}) \,. \tag{1}$$

Similarly, the outputs of $\hat{D}$ and $P \circ \hat{D}$ are determined by the output of U, hence

$$H(U|P \circ \hat{D}) - H(U|\hat{D}) \;=\; H(U) - H(P \circ \hat{D}) - (H(U) - H(\hat{D})) \,. \tag{2}$$

The assertion then follows by combining (1) and (2). □

Lemma 1 shows how the remaining uncertainty about the (non-uniform) input of P can be expressed as a difference of remaining uncertainties about (uniform) inputs of $\hat{D}$ and $P \circ \hat{D}$. In the following, we show how this result can be exploited for automating the quantitative information-flow analysis w.r.t. non-uniform distributions using established tools for uniform QIF.

3.2 Automation of QIF for Non-uniform Distributions

We summarize two kinds of techniques for automatically analyzing the information-flow of programs with respect to uniform distributions. The first kind of technique allows for the accurate, but possibly expensive QIF of a given program [2, 20], and the second kind of technique uses a randomized algorithm for obtaining approximate results with quality guarantees [22]. As we will show next, Lemma 1 allows one to use both kinds of techniques for analyzing programs with respect to non-uniform distribution.

3.2.1 Accurate Quantification

The following proposition from [20] connects the combinatorial characteristics of the partition Π_Q induced by a program Q with the remaining uncertainty about the uniformly distributed input of Q.

Proposition 1 (see [20]). *Let $U = id$ be uniformly distributed and let Q be a program taking input distributed according to U. Then*

$$H(U|Q) \;=\; \frac{1}{\#(I_Q)} \sum_{B \in \Pi_Q} \#(B) \log_2 \#(B) \,.$$

Proposition 1 can be turned into an algorithm for computing $H(U|Q)$: Enumerate all blocks B in the partition Π_Q, determine their sizes, and use these data for computing $H(U|Q)$. The algorithm described in [20] uses this approach for a partition Π that reflects the knowledge gained by an attacker that can adaptively provide input to the program. The algorithm described in [2] extracts a logical representation of Π by computing weakest preconditions and employs model counting techniques for determining the sizes of individual blocks from this logical representation.

The following theorem enables us to directly apply both techniques to programs with non-uniform input distributions.

Theorem 1. *Let $P, \hat{D}, D$ be as defined in Section 3.1. Then*

$$H(D|P) = \frac{1}{\#(I_{\hat{D}})} \left(\sum_{B \in \Pi_{P \circ \hat{D}}} \#(B) \log_2 \#(B) - \sum_{B' \in \Pi_{\hat{D}}} \#(B') \log_2 \#(B') \right).$$

Proof. The statement is obtained by applying Proposition 1 to both terms on the right hand side of Lemma 1. □

3.2.2 Randomized Quantification

The direct computation of $H(D|P)$ on basis of Theorem 1 requires the enumeration of all blocks in the partitions Π_P and $\Pi_{P \circ \hat{D}}$, respectively. Each partition may have as many elements as I_P, which severely limits scalability. The following proposition from [22] is an extension of a result from [3] and addresses this limitation: it implies that, for uniformly distributed inputs, one can give tight bounds for $H(U|Q)$ by considering only a small subset of randomly chosen blocks.

Proposition 2 (see [22]). *Let $U = id$ be uniformly distributed and let Q be a program taking input distributed according to U. Let $B_1, \ldots, B_n$ be drawn randomly from Π_Q with respect to the distribution $p(B) = \frac{\#(B)}{\#(I_Q)}$. Then*

$$\frac{1}{n} \sum_{i=1}^{n} \log \#(B_i) - \delta \;\le\; H(U|Q) \;\le\; \frac{1}{n} \sum_{i=1}^{n} \log \#(B_i) + \delta$$

holds with probability of more than $1 - \frac{(\log \#(\Pi_Q))^2}{n\delta^2}$.

As described in [22], Proposition 2 can be turned into a randomized algorithm for quantitative information-flow analysis. To this end, observe that the random choice of blocks can be implemented by executing the program on a (uniformly chosen) input $s \in I_Q$ and determining the preimage $B = Q^{-1}(s')$ of $s' = Q(s)$. If this preimage is represented by a logical assertion, one can compute the size $\#(B)$ of B using model counting techniques [15]. In this way, $H(U|Q)$ can be approximated with high confidence levels using a number of samples n that is only polylogarithmic in the size of the state space.

The following theorem enables us to leverage these techniques for analyzing programs with non-uniform inputs.

Theorem 2. *Let $P, \hat{D}, D$ be as defined in Section 3.1. Let $B_1, \ldots, B_n$ be drawn randomly from $\Pi_{P \circ \hat{D}}$ and let $B'_1, \ldots, B'_n$ be drawn randomly from $\Pi_{\hat{D}}$ with respect to the distribution $p(B) = \frac{\#(B)}{\#(I_{\hat{D}})}$. Then*

$$\frac{1}{n} \sum_{i=1}^{n} \log \frac{\#(B_i)}{\#(B'_i)} - 2\delta \;\le\; H(D|P) \;\le\; \frac{1}{n} \sum_{i=1}^{n} \log \frac{\#(B_i)}{\#(B'_i)} + 2\delta$$

with a probability of more than $\left(1 - \frac{(\log \#(\Pi_{\hat{D}}))^2}{n\delta^2}\right)^2$.

Proof. Apply Proposition 2 to both terms on the right hand side of Lemma 1. For the confidence levels, observe that the blocks B_i are drawn independently from the blocks B'_i, hence the probabilities that the inequalities hold multiply. Observing that $\#(\Pi_{\hat{D}}) \ge \#(\Pi_{P \circ \hat{D}})$, we replace the larger probability by the smaller one, which concludes this proof. □

Finally, the exact computation of the blocks B_i can be prohibitively expensive. Fortunately, one can avoid this expensive computation by resorting to under- and over-approximations $\underline{B_i}$ and $\overline{B_i}$ of B_i, i.e. subsets of initial states with $\underline{B_i} \subseteq B_i \subseteq \overline{B_i}$. The computation of $\underline{B_i}$ and $\overline{B_i}$ can be done using existing techniques for symbolic execution and abstract interpretation, see [22]. In this paper, we simply assume the existence of such approximations.

Corollary 1. *Let $B_1, \ldots, B_n$ and $B'_1, \ldots, B'_n$ be chosen as in Theorem 2. Let $\underline{B_i} \subseteq B_i \subseteq \overline{B_i}$ and $\underline{B'_i} \subseteq B'_i \subseteq \overline{B'_i}$, for all $i \in \{1, \ldots n\}$. Then*

$$\frac{1}{n} \sum_{i=1}^{n} \log \frac{\#(\underline{B_i})}{\#(\overline{B'_i})} - 2\delta \;\le\; H(D|P) \;\le\; \frac{1}{n} \sum_{i=1}^{n} \log \frac{\#(\overline{B_i})}{\#(\underline{B'_i})} + 2\delta$$

with a probability of more than $\left(1 - \frac{(\log \#(\Pi_{\hat{D}}))^2}{n\delta^2}\right)^2$.

Corollary 1 follows directly from Theorem 2 by replacing all blocks that occur in the numerator (denominator) of the right (left) hand side and on the denominator (numerator) on the left (right) hand side by their over-(under-)approximating counterparts.

4. ROBUSTNESS

In this section, we show that the remaining uncertainty about a secret is robust with respect to small variations in the input distribution. This allows us to replace actual input distributions with approximate distributions. Based on the quality of the approximation, we give upper and lower bounds on the error this approximation introduces in the analysis. Focusing on approximate distributions can simplify the information-flow analysis, e.g. by allowing to replace "almost uniform" distributions by uniform distributions.

We say that two distributions p and q on some set S are γ-*close* if the probabilities they assign to each value differ at most by a factor of γ.

$$p \overset{\gamma}{\approx} q \quad \equiv \quad \forall x \in S : \ \frac{1}{\gamma}\, q(x) \leq p(x) \leq \gamma\, q(x)$$

In the following we will consider a random variable with respect to different probability distributions on its input domain. We introduce the notation X^p emphasize that we consider variable X with respect to the underlying distribution p. The following lemma states that, for γ-close distributions p and q, the distributions of X^p and X^q are also γ-close.

Lemma 2. *Let X be a random variable and let p and q be distributions on the domain of X. Then $p \overset{\gamma}{\approx} q$ implies $p_{X^p} \overset{\gamma}{\approx} p_{X^q}$.*

Proof. For all x we have

$$\frac{1}{\gamma} \cdot p_{X^q}(x) = \sum_{a \in X^{-1}(x)} \frac{1}{\gamma} \cdot q(a) \leq \sum_{a \in X^{-1}(x)} p(a) = p_{X^p}(x)\,.$$

The proof for the upper bound follows along the same lines. □

We next show that the entropy of a random variable is robust with respect to small changes in its input distribution. Formally, we show that for two random variables X and Y with γ-close distributions, i.e., $p_X \overset{\gamma}{\approx} p_Y$, the Shannon entropy $H(X)$ can be bounded in terms of the entropy $H(Y)$.

Lemma 3. *Let X and Y be random variables with $p_X \overset{\gamma}{\approx} p_Y$. Then we have*

$$H(X) \begin{cases} \leq \gamma \cdot H(Y) + \gamma \log_2 \gamma \\ \geq \frac{1}{\gamma} \cdot H(Y) - \frac{\log_2 \gamma}{\gamma} \end{cases} .$$

Proof. $H(X) = -\sum_x p_X(x) \log_2 p_X(x) \overset{(*)}{\leq} -\sum_x \gamma \cdot p_Y(x) \log_2 \frac{p_Y(x)}{\gamma} = \gamma \cdot H(Y) + \gamma \log_2 \gamma,$
where $(*)$ follows from $p_X \overset{\gamma}{\approx} p_Y$. The proof of the lower bound is analogous. □

We can use Lemma 3 together with Lemma 2 to obtain bounds on the remaining uncertainty of a program for distribution p from an analysis with respect to a distribution q with $p \overset{\gamma}{\approx} q$.

Theorem 3. *Let p and q be distributions with $p \overset{\gamma}{\approx} q$. Then we have*

$$H(D^p|P^p) \begin{cases} \leq \gamma \cdot H(D^q) - \frac{1}{\gamma} \cdot H(P^q) + \log_2 \gamma \left(\gamma + \frac{1}{\gamma}\right) \\ \geq \frac{1}{\gamma} \cdot H(D^q) - \gamma \cdot H(P^q) - \log_2 \gamma \left(\gamma + \frac{1}{\gamma}\right) \end{cases} .$$

Proof. Observe that $H(D^p|P^p) = H(D^p) - H(P^p)$ because P^p is determined by D^p. Since $p \overset{\gamma}{\approx} q$, Lemma 2 yields $p_{D^p} \overset{\gamma}{\approx} p_{D^q}$. Applying Lemma 3 yields the assertion. □

In Section 5.2 we show an application of Theorem 3, where we analyze a program with respect to a uniformly distributed q in order to derive bounds for the real, almost uniform, distribution p.

5. CASE STUDY

In this section we illustrate the techniques described in the previous sections. Namely, we will give an example of how an analysis with respect to non-uniform distributions can be reduced to the uniform case, which we handle using existing tool support [20]. Furthermore, we give an example of how our robustness result can be used to estimate the error introduced by replacing in the analysis an almost uniform distribution by a uniform one.

We consider a program for checking the integrity of Personal Identification Numbers (PINs) as used in electronic banking. Previous formal analyses of this program [19, 37] assume uniformly distributed PINs; they are not fully accurate because PIN generation methods typically produce a skewed distribution. Using the techniques presented in this paper, we perform the first formal analysis that takes this skew into account.

We analyze the integrity check with respect to PINs that stem from two different PIN generation algorithms. The first generation algorithm is easily expressed as a program, and we will use the techniques developed in Section 3 to perform a precise non-uniform QIF. The second generation algorithm produces PINs that are almost uniformly distributed, and we will use the techniques developed in Section 4 to perform an approximate QIF of the integrity check program. We begin by describing the integrity check and its use in practice.

5.1 PIN Integrity Check

When a customer authenticates himself at an Automated Teller Machine (ATM), he enters his PIN. This PIN is then sent to his bank for verification [1, 4]. Before sending, the PIN is XORed with the customer's Personal Account Number (PAN) and encrypted using a symmetric cryptosystem. In case the ATM cannot communicate directly with the customer's bank, the encrypted PIN⊕PAN will pass through a series of switches. Each of these switches decrypts and extracts the PIN, checks it for integrity, and re-encrypts the PIN using a key that is shared with the next switch. All operations on the PIN are performed in dedicated tamper-resistant Hardware Security Modules (HSMs), which protect the communication keys and the PINs even if the switch is compromised.

Unfortunately, HSMs fail to fulfill this purpose because the outcome of the PIN integrity check leaks information about the value of the PIN [13]: Upon receiving an encrypted pin, the HSM decrypts and XORs the result with a given account number to extract the PIN. The HSM then performs a simple integrity check on the PIN, namely it checks whether all PIN digits are < 10. Clearly, this check will succeed if the given account number is the customers PAN. However, the protocol does not forbid the use of the integrity check with an arbitrary account number PAN', in which case the HSM will reveal whether PIN⊕PAN⊕PAN' is a valid PIN. As the PAN itself is not secret, the integrity check can be seen as an oracle that, on input m, reveals whether all digits of PIN⊕m are < 10.

We model the integrity check of a single PIN digit as a program $P = (I_P, F_P, R_P)$ with $I_P = \{0, \ldots, 9\}$, $F_P = \{0, 1\}$, and

$$P(s) \equiv s \oplus m < 10\,,$$

where $m \in \{0, \ldots, F\}$ is fixed. For example, for $m = F$, the integrity check is equivalent to the condition $s \geq 6$.

5.2 Non-uniform PINs from Decimalization Tables

The PIN generation algorithm described in [5] works as follows: In a first step the customer's account number is encrypted using DES under a fixed PIN derivation key. In a second step, the ciphertext is converted into a hexadecimal number. The first 4 digits of this number are taken and decimalized. The decimalization leaves digits $0-9$ unchanged, and maps $A-F$ to $0-5$.

We assume DES to be an ideal cipher, i.e., a random permutation. This assumption is known as the *Ideal Cipher Model* [35] and is commonly used in cryptography to abstract from the inner details of block ciphers. In this model, the output of the DES encryption is uniformly distributed and we can characterize the skew of the PIN as described in Section 3. More precisely, we capture the PIN generation algorithm as a program $\hat{D}$ which, given uniformly distributed input (i.e. the result of encrypting the PAN with DES), computes a PIN as described above. The purpose of this section is to illustrate our reduction to uniform distributions. For clarity of presentation, we will focus on a simplified scenario with one-digit PINs, i.e., $I_{\hat{D}} = \{0, \ldots, F\}$, $F_{\hat{D}} = \{0, \ldots, 9\}$, and

$$\hat{D}(s) = s \bmod 10$$

For computing $H(D|P)$ for $m = F$ using Theorem 1, we need to determine the partitions $\Pi_{\hat{D}}$ and $\Pi_{P \circ \hat{D}}$ induced on $I_{\hat{D}}$ by $\hat{D}$ and $P \circ \hat{D}$, respectively. It is easy to see that $\Pi_{\hat{D}}$ consists of blocks of values that are equal modulo 10 and that $\Pi_{P \circ \hat{D}}$ combines the blocks from $\Pi_{\hat{D}}$ with values < 6 or ≥ 6 modulo 10, respectively.

$$\Pi_{\hat{D}} = \{\{0, A\},\ \{1, B\},\ \{2, C\},\ \{3, D\},\ \{4, E\},\ \{5, F\},\ \{6\},\ \{7\},\ \{8\},\ \{9\}\}$$
$$\Pi_{P \circ \hat{D}} = \{\{0, 1, 2, 3, 4, 5, A, B, C, D, E, F\},\ \{6, 7, 8, 9\}\}$$

We apply Theorem 1 to this data and obtain

$$\begin{aligned} & H(D|P) \\ &= \frac{1}{\#(I_{\hat{D}})} \left(\sum_{B \in \Pi_{P \circ \hat{D}}} \#(B) \log_2 \#(B) - \sum_{B' \in \Pi_{\hat{D}}} \#(B') \log_2 \#(B') \right) \\ &= \frac{1}{16} \left((12 \log_2(12) + 4 \log_2(4)) - (6 \cdot 2 \log_2(2) + 4 \cdot 1 \log_2(1)) \right) \\ &\approx 2.4387 \end{aligned}$$

This result shows that, after a integrity check with account number PAN$\oplus F$, there are 2.4387 bits of uncertainty left about a single digit. A sightly more complex analysis (whose details we omit) reveals that, for a 4 pin digit, one check with PAN$\oplus FFFF$ leaves 12.9631 bits of uncertainty.

5.3 Automated Analysis of Adaptive Attacks

The analysis presented above considers an adversary that performs the PIN integrity check using a single fixed input $m = F$. In practice, however, an attacker can repeatedly perform PIN integrity checks with different values of m, thereby further narrowing down the possible values of the PIN. For assessing the security of a system it is necessary to take such repeated queries into account. We briefly report on experimental results where we use existing automated techniques for reasoning about this kind of attack.

The basis for our analysis is the formal model for knowledge refinement in adaptive attacks described in [19, 20]. In this model, each attack strategy induces a partition on the set of secret inputs. An attack strategy of n steps is *optimal* if the remaining uncertainty about the secret after an attack is minimal among all possible attack strategies of n steps. Here, the remaining uncertainty is computed from the attack strategy's induced partition using Proposition 1. As a consequence, the automatic tool presented in [19, 20] requires that the inputs are uniformly distributed. Using our reduction techniques, we leverage this restriction: We can simply apply the tool to $P \circ \hat{D}$ and obtain $H(U|P \circ \hat{D})$. We then use Theorem 1 to obtain $H(D|P)$ from $H(U|P \circ \hat{D})$ and $H(U|\hat{D})$ (which is a constant). The results of the analysis are depicted in Figure 2. The value of $H(\hat{D}|P) = 1$ in the last row accounts for the fact that a single PIN digit can can be narrowed down to two equally likely alternative values.

5.4 Almost Uniform PINs

The Interbank PIN generation algorithm [18] works as follows.[1] In a first step, the PAN is encrypted, yielding a string of 16 hexadecimal numbers. This string is scanned from left to right, and the first four decimal digits that are encountered are used as the PIN. If there are less than 4 decimal digits in the string, A is subtracted from each digit in the string, and the process is repeated until four decimal digits are found. This second scan ignores the positions in the string that already yielded decimal digits in the first round. [2]

We use the techniques presented in Section 4 for analyzing the PIN integrity check with respect to PINs that are generated according to the Interbank PIN generation algorithm. To this end, we first determine γ such that the PIN distribution is γ-close to the uniform distribution. Then we perform a uniform analysis of the PIN integrity check and use Theorem 3 with γ to estimate the uncertainty about a PIN drawn from the skewed distribution.

For an analysis of the Interbank PIN distribution, let the random variable X denote the number of decimal digits in the hexadecimal string obtained by encrypting the account number. We assume this string to be uniformly distributed. $\Pr(X = k)$ can be calculated as follows:

$$\Pr(X = k) = \binom{16}{k} \left(\frac{5}{8}\right)^k \left(\frac{3}{8}\right)^{16-k}$$

The probability that there are at least 4 decimal digits in the string is

$$\Pr(X \geq 4) = 1 - \sum_{k=0}^{3} \Pr(X = k) \approx 0.9995\ ,$$

in which case the resulting PINs are uniformly distributed.

Let the random variable Y denote the output of the generation algorithm, i.e., the generated PIN. Then

$$\begin{aligned} \Pr(Y = a) &= \Pr(Y = a | X \geq 4) \Pr(X \geq 4) + \Pr(Y = a | X \leq 3) \Pr(X \leq 3) \\ &= \Pr(X \geq 4) \cdot 10^{-4} + \underbrace{\Pr(Y = a | X \leq 3)}_{\leq 6^{-4}} \Pr(X \leq 3) \end{aligned}$$

We set $\gamma = \Pr(X \geq 4) + \left(\frac{5}{3}\right)^4 \Pr(X \leq 3) \approx 1.003$. A simple calculation shows that $\frac{1}{\gamma} < \Pr(X \geq 4)$, which gives us the following lower bound on $\Pr(Y = a)$:

$$\Pr(Y = a) \geq \Pr(X \geq 4) \cdot 10^{-4} > \frac{1}{\gamma} \cdot 10^{-4}$$

Similarly, we can obtain an upper bound on $\Pr(Y = a)$ as follows:

$$\Pr(Y = a) \leq \Pr(X \geq 4) \cdot 10^{-4} + \Pr(X \leq 3) \cdot 6^{-4} = \gamma \cdot 10^{-4}.$$

[1] We thank Graham Steel for pointing us to this example.
[2] Additionally, a PIN of 0000 is replaced by 0100. We will ignore this detail in our analysis.

#Steps	$\Pi_{P \circ \hat{D}}$	$H(U\|P \circ \hat{D})$	$H(\hat{D}\|P)$
0	`[[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]]`	4	3.25
1	`[[0,1,8,9,10,11],[2,3,4,5,6,7,12,13,14,15]]`	3.05	2.30
2	`[[0,1,10,11],[8,9],[2,3,12,13],[4,5,6,7,14,15]]`	2.09	1.34
3	`[[0,1,10,11],[8,9],[2,3,12,13],[4,5,14,15],[6,7]]`	1.75	1.0

Figure 2: Results of automatically analyzing the PIN integrity check with respect to multiple runs of an adaptive attacker.

We conclude that the distribution of the PINs is γ-close to the uniform distribution u, i.e., $p_Y \overset{\gamma}{\approx} u$.

Consider again the PIN integrity check program P from Section 5.1, generalized to 4 PINs, i.e., $I_P = \{0, \ldots, 9\}^4$, $F_P = \{0, 1\}$, and

$$P(s_1 s_2 s_3 s_4) = \bigwedge_{i=1\ldots4} s_i \oplus m_i < 10$$

Theorem 3 gives the following formula to bound $H(D^{p_Y}|P^{p_Y})$:

$$H(D^{p_Y}|P^{p_Y}) \leq \gamma \cdot H(D^u) - \frac{1}{\gamma} \cdot H(P^u) + \log_2 \gamma \left(\gamma + \frac{1}{\gamma}\right)$$

We set $m = FFFF$, which results in the following upper bound:

$$\begin{aligned} & H(D^{p_Y}|P^{p_Y}) \\ & \leq \gamma \cdot \log_2 10^4 + \frac{1}{\gamma} \cdot \left(\frac{4^4}{10^4} \log_2 \frac{4^4}{10^4} + \frac{10^4 - 4^4}{10^4} \log_2 \frac{10^4 - 4^4}{10^4} \right) \\ & \quad + \log_2 \gamma \left(\gamma + \frac{1}{\gamma}\right) \\ & \approx 13.1654 \end{aligned}$$

A lower bound of $H(D^{p_Y}|P^{p_Y}) \geq 13.0664$ follows along the same lines.

The small delta between the upper and lower bounds shows that the uniform analysis is (almost) precise for PINs generated according to the Interbank algorithm. We conclude by comparing this result with the result of the analysis with respect to PINs generated using decimalization tables presented in Section 5.2 where, for 4 digit PINs, we obtained a remaining uncertainty of 12.9631 bits. The difference between the remaining uncertainties gives an account of the security that is gained by using a better (i.e., less skewed) PIN generation algorithm.

6. RELATED WORK

Denning is the first to quantify information flow in terms of the reduction in uncertainty about a program variable [14]. Millen [28] and Gray [16] use information theory to derive bounds on the transmission of information between processes in multi-user systems. Lowe [24] shows that the channel capacity of a program can be over-approximated by the number of possible behaviors. The channel capacity corresponds to the maximal leakage w.r.t. to any input distribution and hence is an over-approximation of the information that is actually revealed.

Clark, Hunt, and Malacaria [10] connect equivalence relations to quantitative information flow, and propose the first type system for statically deriving quantitative bounds on the information that a program leaks [11]. The analysis assumes as input (upper and lower) bounds on the entropy of the input variables and delivers corresponding (upper and lower) bounds for the leakage of the program. For loops with high guards, the analysis always reports complete leakage of the guard.

Malacaria [25] shows how to characterize the leakage of loops in terms of the loop's output and the number of iterations. Closely related on this approach, Mu and Clark [30] propose a precise, automatic QIF based on a distribution transformer semantics, which can deal with non-uniform input distributions. Their approach relies on an explicit representation of the probability distribution transformed by the program (and hence the set of initial states), which prevents the direct application to programs with large state spaces. The problem is mitigated by an interval-based abstraction proposed in [29]. The abstraction splits a totally ordered domain into intervals, each of which assumed to be uniformly distributed. In our approach, the probability distribution is represented in terms of preimages of a generating program, which offers the possibility of a symbolic treatment of large state spaces.

Köpf and Basin [20] show how to compute partitions on the secret input that represent what an attacker can learn in an adaptive attack. Backes, Köpf, and Rybalchenko [2] show how to determine the partitions corresponding to the information (with respect to a non-adaptive attacker) that a program leaks by computing weakest preconditions. Both approaches rely on counting the number and the sizes of the preimages in order to quantify the remaining uncertainty about the input w.r.t. uniform distributions. When used in conjunction with these approaches, the ideas presented in this paper can be used to weaken the requirement of a uniform distribution.

Köpf and Rybalchenko [22] propose approximation and randomization techniques to approximate the remaining uncertainty about a program's inputs for programs with unbounded loops. Their approach relies on approximating the sizes of blocks (but without their complete enumeration) and it delivers bounds w.r.t. uniformly distributed inputs. As we have shown, the reduction presented in this paper can be used for extending the techniques to programs with non-uniform input distributions.

McCamant and Ernst propose a dynamic taint analysis for quantifying information flow [27]. Their method does not assume a particular input distribution and provides over-approximations of the leaked information along a particular path. However, it does not yield guarantees for all program paths, which is important for security analysis. Newsome, McCamant, and Song [31] also use the feasible outputs along single program paths as bounds for channel capacity (i.e. the maximal leakage w.r.t. to all possible input distributions), and they apply a number of heuristics to approximate upper bounds on the number of reachable states of a program.

Chatzikokolakis, Chothia, and Guha [7] use sampling to build up a statistical system model. Based on this model, they compute the channel capacity, i.e. the maximum leakage w.r.t. all possible input distributions.

DiPierro, Hankin, and Wiklicky [33] consider probabilistic processes with given input distributions and (instead of information theory) use the distance of the produced output distributions to quantify information flow.

Clarkson, Myers, and Schneider [12] use non-uniform input distributions to model adversaries beliefs, which they update according to the program semantics. They do not discuss techniques for automation or abstraction.

Smith [36] proposes min-entropy as an alternative measure of

information flow. Min-entropy gives bounds on the probability of guessing a secret in one attempt, whereas Shannon-entropy gives bounds on the average number of guesses required for determining a secret. The investigation of a reduction from non-uniform to uniform QIF for min-entropy remains future work.

7. CONCLUSIONS AND FUTURE WORK

We have considered the problem of quantifying the information-flow in programs with respect to non-uniform input distributions. We have made the following contributions to solve the problem. First, we have shown how the problem of non-uniform QIF can be reduced to the uniform case. To this end, we represented the non-uniform input distribution as a program that receives uniform input, and we sequentially composed it with the target program. We have proved a connection between the information-theoretic characteristics of the target program and its composition with the distribution generator. This connection enables us to perform a precise non-uniform analysis using existing QIF techniques for the uniform case. Second, we have shown that the result of a QIF is robust with respect to small variations in the input distribution. This result shows that we can estimate the information-theoretic characteristics of a program by considering an approximate input distribution. This is useful in cases where the input distribution can only be approximated or an approximation simplifies the analysis. Finally, we have performed a case-study where we illustrated both techniques and demonstrated their usefulness in practice.

Acknowledgments.

Boris Köpf's research was partially done while at MPI-SWS and is partially supported by FP7-ICT Project NESSoS (256980), by FP7-PEOPLE-COFUND Project AMAROUT (229599), and by Comunidad de Madrid Program PROMETIDOS-CM (S2009TIC-1465).

8. REFERENCES

[1] American National Standards Institute. Banking - Personal Identification Number Management and Security - Part 1: PIN protection principles and techniques for online PIN verification in ATM & POS systems. ANSI X9.8-1, 2003.

[2] M. Backes, B. Köpf, and A. Rybalchenko. Automatic Discovery and Quantification of Information Leaks. In *Proc. 30th IEEE Symposium on Security and Privacy (S& P '09)*, pages 141–153. IEEE, 2009.

[3] T. Batu, S. Dasgupta, R. Kumar, and R. Rubinfeld. The complexity of approximating entropy. In *Proc. 34th Symposium on the Theory of Computing (STOC '02)*, pages 678–687. ACM, 2002.

[4] O. Berkman and O. M. Ostrovsky. The unbearable lightness of pin cracking. In *Financial Cryptography (FC '07)*, volume 4886 of *LNCS*, pages 224–238. Springer, 2008.

[5] M. Bond and P. Zieliński. Decimalisation table attacks for PIN cracking. Technical Report UCAM-CL-TR-560, University of Cambridge, Computer Laboratory, Feb. 2003.

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

[7] K. Chatzikokolakis, T. Chothia, and A. Guha. Statistical Measurement of Information Leakage. In *Proc. 16th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '10)*, LNCS 6015, pages 390–404. Springer, 2010.

[8] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Anonymity protocols as noisy channels. *Inf. Comput.*, 206(2-4):378–401, 2008.

[9] D. Clark, S. Hunt, and P. Malacaria. Quantitative Analysis of the Leakage of Confidential Data. *Electr. Notes Theor. Comput. Sci.*, 59(3), 2001.

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

[11] D. Clark, S. Hunt, and P. Malacaria. A static analysis for quantifying information flow in a simple imperative language. *Journal of Computer Security*, 15(3):321–371, 2007.

[12] M. R. Clarkson, A. C. Myers, and F. B. Schneider. Belief in Information Flow. In *Proc. IEEE Computer Security Foundations Workshop (CSFW '05)*, pages 31–45. IEEE, 2005.

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

[14] D. E. Denning. *Cryptography and Data Security*. Addison-Wesley, 1982.

[15] C. Gomez, A. Sabharwal, and B. Selman. Chapter 20: Model counting. In *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

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

[17] J. Heusser and P. Malacaria. Quantifying information leaks in software. In *Proc. Annual Computer Security Applications Conference (ACSAC '10)*. ACM, 2010.

[18] IBM Corporation. Interbank pin generation algorithm. `https://publib.boulder.ibm.com/infocenter/zos/v1r9/topic/com.ibm.zos.r9.csfb400/inbkal.htm`.

[19] B. Köpf and D. Basin. Automatically Deriving Information-theoretic Bounds for Adaptive Side-channel Attacks. *Journal of Computer Security (to appear)*.

[20] B. Köpf and D. Basin. An Information-Theoretic Model for Adaptive Side-Channel Attacks. In *Proc. ACM Conference on Computer and Communications Security (CCS '07)*, pages 286–296. ACM, 2007.

[21] B. Köpf and M. Dürmuth. A Provably Secure and Efficient Countermeasure against Timing Attacks. In *Proc. 22rd IEEE Computer Security Foundations Symposium (CSF '09)*, pages 324–335. IEEE, 2009.

[22] B. Köpf and A. Rybalchenko. Approximation and Randomization for Quantitative Information-Flow Analysis. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF '10)*, pages 3–14. IEEE, 2010.

[23] B. Köpf and G. Smith. Vulnerability Bounds and Leakage Resilience of Blinded Cryptography under Timing Attacks. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF '10)*, pages 44–56. IEEE, 2010.

[24] G. Lowe. Quantifying Information Flow. In *Proc. IEEE Computer Security Foundations Workshop (CSFW '02)*, pages 18–31. IEEE, 2002.

[25] P. Malacaria. Risk assessment of security threats for looping constructs. *Journal of Computer Security*, 18(2):191–228, 2010.

[26] J. L. Massey. Guessing and Entropy. In *Proc. IEEE International Symposium on Information Theory (ISIT '94)*, page 204. IEEE, 1994.

[27] S. McCamant and M. D. Ernst. Quantitative information flow as network flow capacity. In *Proc. Conf. on Programming*

Language Design and Implementation (PLDI '08), pages 193–205. ACM, 2008.

[28] J. K. Millen. Covert Channel Capacity. In *Proc. IEEE Symposium on Security and Privacy (S&P '87)*, pages 60–66. IEEE, 1987.

[29] C. Mu and D. Clark. An Interval-based Abstraction for Quantifying Information Flow. *ENTCS*, 253(3):119–141, 2009.

[30] C. Mu and D. Clark. Quantitative Analysis of Secure Information Flow via Probabilistic Semantics. In *Proc. 4th International Conference on Availability, Reliability and Security (ARES '09)*, pages 49–57. IEEE, 2009.

[31] J. Newsome, S. McCamant, and D. Song. Measuring Channel Capacity to Distinguish Undue Influence. In *Proc. 4th ACM Workshop on Programming Languages and Analysis for Security (PLAS '09)*. ACM, 2009.

[32] S. Park, F. Pfenning, and S. Thrun. A Probabilistic Language based upon Sampling Functions. In *Proc. ACM Symposium on Principles of Programming Languages (POPL '05)*, 2005.

[33] A. D. Pierro, C. Hankin, and H. Wiklicky. Approximate Non-Interference. In *Proc. IEEE Computer Security Foundations Workshop (CSFW '02)*, pages 3–17. IEEE, 2002.

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

[35] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[36] G. Smith. On the foundations of quantitative information flow. In *Proc. Intl. Conference of Foundations of Software Science and Computation Structures (FoSSaCS '09)*, LNCS 5504, pages 288–302. Springer, 2009.

[37] G. Steel. Formal analysis of PIN block attacks. *Theoretical Computer Science*, 367(1-2):257–270, Nov. 2006.