Journal of Computer Security 18 (2010) 191–228
DOI 10.3233/JCS-2010-0360
IOS Press

Risk assessment of security threats for looping constructs *

Pasquale Malacaria
School of Electronic Engineering and Computer Science, Queen Mary University of London, London
E-mail: pm@dcs.qmul.ac.uk

There is a clear intuitive connection between the notion of leakage of information in a program and concepts from Information Theory. We explore this connection by interpreting Information Theory as a security risk assessment of programs. Information Theory will then be used to introduce techniques to reason on looping constructs, which are the kind of programs that previous quantitative models failed to satisfactory address. The semantics here introduced allows to describe both the amount and rate of leakage; if either is small enough, then a program might be deemed "secure". Using the semantics we provide an investigation and classification of bounded and unbounded covert channels.

Keywords: Quantitative Information Flow, Information Theory, language based security

1. Introduction

There is a basic conceptual issue that lies at the heart of the foundations of security: The problem is that "secure" programs do leak small amounts of information. An example is a password checking program

```
if (l==h) access else deny
```

where an attacker will gain some information by observing what the output is (by observing `deny` he will learn that his guess `l` was wrong). This makes non-interference[1] [11] based models of security [8,30] problematic; they judge far too many programs to be "insecure". As elegantly put in [26]:

> In most non-interference models, a single bit of compromised information is flagged as a security violation, even if one bit is all that is lost. To be taken seriously, a non-interference violation should imply a more significant loss. Even ..., where timings are not available, and a bit per millisecond is not distinguishable from a bit per fortnight ... a channel that compromises an unbounded amount of information is substantially different from one that cannot.

*Extended and revised version of [16]. Research partially supported by EPSRC grant EP/C009967/1.

[1] Intuitively interference from x to y means changes in x affect the state of y. Non-interference is the lack of intererence.

Of course, using declassification [27] it is still possible to use a non-interference model to limit, rather than eliminate, the areas in a program where information will be leaked. But, non-interference does not itself help us in deciding whether to declassify. Again, [26] raises the question: how we decide that a region is safe to declassify?

To illustrate the problem, consider the following program containing a secure variable h and a public variable l:

```
l = 20; while (h < l) {l = l − 1}
```

The program performs a bounded search for the value of the secret h. Is it safe to declassify that program? One could argue that the decision should depend on the size of the secret; the larger the secret the more declassifiable it becomes. How to give a precise meaning to this argument? Is the previous program secure if h is a 10-bit variable? Is it secure if h is a 16-bit variable? And should not the answer depend also on the attacker's knowledge of the distribution of inputs e.g. if she/he knew that 0 is a much more likely value for h than any other value?

The main objective of the present work is to develop a theory where this kind of questions can be mathematically addressed. To this aim we will develop an Information Theoretical semantics of looping commands. The semantics is quantitative: outcomes are real numbers measuring security properties of programs.

The appeal of Shannon's Information Theory [28] in this context is that it combines the *probability* of an event with the *damage* the happening of that event would cause. In this sense information theory provides a *risk assessment analysis* of language ba ed security.

1.1. Risk assessment

The components of a quantitative risk assessment are the possible losses and the probabilities of these losses. The typical risk assessment formula is

$$\sum_{1 \leqslant i \leqslant n} L_i \, \mu(L_i),$$

where $L_i, 1 \leqslant i \leqslant n$, is the loss (or damage) associated to the event i and $\mu(L_i)$ is the probability of that loss occurring. In probability terms this is the expected value of the random variable L.

In security terms we first need to define what the damage is and then, once identified the events that an attacker can observe, how damaging the occurrence of such an event could be for the security of the system.

This work, following Information Theory, identifies damage caused by an observable event with the information gained about the secret by that observation.

Example. Consic
bit variables and
equally likely). W
ference between t
has happened. Th
the damage. The
of the whole secre
be gaining inform

(1) observe

- probabil
- damage

(2) observe

- probabi
- damage

Combining da

$\frac{1}{4}\log\left(\frac{4}{1}\right)$

an instance of $\sum$

1.2. Contributio

Contribution to j
What is the n
security context'
We formalize
that expected da
bers returned b
security threats.
relate probabili
leakage: this is
A fundament
age, null leakag

Contribution to
This work de
oretical formula
of loops: these

[2]In the paper log

Example. Consider again the password checking program and suppose `l`, `h` are 2-bit variables and the distribution of values of `h` is uniform (all values $0,\ldots,3$ are equally likely). We identify the damage (or loss) associated to an event with the difference between the size of the search space for the secret before and after the event has happened. The more is revealed by an event-the larger the difference-the bigger the damage. The damage for the observation `access` will be gaining information of the whole secret $2 = \log(4)$ bits[2] while the damage for the observation `deny` will be gaining information of one possibility being eliminated. Formally:

(1) `observe access`:
- probability $= \frac{1}{4}$,
- damage $= \log(4) - \log(1) = \log(\frac{4}{1}) = 2$.

(2) `observe deny`:
- probability $= \frac{3}{4}$,
- damage $= \log(4) - \log(3) = \log(\frac{4}{3})$.

Combining damages with probabilities we get the *expected damage*:

$$\frac{1}{4}\log\left(\frac{4}{1}\right) + \frac{3}{4}\log\left(\frac{4}{3}\right)$$

an instance of $\sum p_i \log(\frac{1}{p_i})$, Shannon's entropy formula.

1.2. Contributions

Contribution to foundations

What is the meaning of the numbers obtained using Information Theory in this security context?

We formalize the concept of damage associated to an observable event and show that expected damage and Information Theoretical leakage coincide; hence the numbers returned by the Information Theory analysis represents a risk assessment of security threats. Using this equivalence between leakage and expected damage we relate probability of an attack causing a security damage above a threshold with leakage: this is done by using the celebrated Markov inequality.

A fundamental result of this part is the equivalence between impossibility of damage, null leakage and not interference in the Goguen Meseguer sense [11].

Contribution to reasoning techniques

This work describes tools to compute the leakage in loops; first Information Theoretical formulas characterizing leakage are extracted by the denotational semantics of loops: these formulas are the basis for defining:

[2]In the paper log stands for base 2 logarithm.

1. Channel capacity: the maximum amount of leakage of a loop as a function of the attacker's knowledge of the input.
2. Rate of leakage: the amount of information leaked as a function of the number of iterations of the loop.

These definitions are then used in a classification of loops. This is an attempt to answer questions like:

1. Is the amount of leakage of the loop unbounded as a function of the size of the secret?
2. How does the rate change when the size of the secret changes?

Notice that in sequential programs many unbounded covert channels contain loops; for this reason we claim that a major achievement of this work is the identification of and mathematical reasoning about unbounded covert channels [26]:

> Characterization of unbounded channels is suggested as the kind of goal that would advance the study of this subject, and some creative thought could no doubt suggest others.

To motivate the relevance of this paper in the above contexts some case studies are presented. We hope that by seeing the definitions at work in these cases the reader will be satisfied that the semantics is:

1. Natural: i.e. in most cases agrees with our intuition about what the leakage should be and when it does not it provides new insights.
2. Helpful: i.e. it provides clear answers for situations where the intuition does n't provide answers.
3. General: although some ingenuity is required case by case, the setting is not ad hoc.
4. Innovative: it provides a fresh outlook on reasoning about covert channels in programs in terms of quantitative reasoning.

1.3. Related work

Early works

Pioneering work by Denning [9,10] shows the relevance of Information Theory to the analysis of flow of information in programs. She worked out semantics for assignments and conditionals, and gave persuasive arguments and examples. However, she did not show how to do a semantics of a full, Turing-complete programming language, with loops. As a consequence, some of the examples we consider involving unbounded channels are beyond the theory there.

Further seminal work relating Information Theory and non-interference in computational systems was done by Millen, McLean, Gray [19,20,31]; none of this work however concentrate on programming languages constructs.

Quantitative approaches to covert channel analysis in somewhat different contexts have been proposed by Gray and Syverson [13], Weber [32] and Wittbold [34].

Recent works
In the context of p
ory and non-interfer
a series of papers b
present work is intr
and program variabl
ditional mutual info
of the public input.
and the analysis is o
the loop leaks every
Similar approach
been proposed in di
dessi and Panangad
Boreale uses con
context of process a
of the secret and th
notion of rate of lea
visible action conve
Chatzikokolakis,
text of anonymity p
information of the
lowed to be leaked
Recently these i
frey Smith [29]. W
Non informatio
also recently been
CSP and defines a
time.
Compared with
this paper abstract
drawbacks to this i
models richer than
Di Pierro, Hank
interference in a
[22]. Their approa
of runs necessary
A probabilistic
Clarkson, Myers,
and the revision
believing that the
system by enterin
To the best of
tive reasoning of

ɔf a loop as a function of

a function of the number

ps. This is an attempt to

ınction of the size of the

hanges?

overt channels contain
f this work is the iden-
ɔvert channels [26]:

s the kind of goal that
ative thought could no

; some case studies are
these cases the reader

bout what the leakage

'e the intuition does n't

se, the setting is not ad

ɔut covert channels in

nformation Theory to
out semantics for as-
l examples. However,
ete programming lan-
'e consider involving

interference in com-
1]; none of this work

ıat different contexts
l Wittbold [34].

Recent works

In the context of programming languages the relations between Information Theory and non-interference [11,25] relevant to the present work have been studied in a series of papers by Clark, Hunt, Malacaria [3–5], where the background for the present work is introduced: the main ingredients are an interpretation of programs and program variables in terms of random variables. Leakage is defined as the conditional mutual information of the secret and the program output given knowledge of the public input. These works however concentrate on providing a static analysis and the analysis is over pessimistic w.r.t. loops (if any leakage is possible in a loop, the loop leaks everything).

Similar approaches based on the same information theoretical definitions have been proposed in different context by Boreale [1] and by Chatzikokolakis, Palamidessi and Panangaden [2].

Boreale uses conditional mutual information to measure information leaks in the context of process algebra. He defines leakage as the conditional mutual information of the secret and the process given knowledge of the public data. He also defines a notion of rate of leakage in terms of the maximal number of bits of information per visible action conveyed by an experiment on the studied process.

Chatzikokolakis, Palamidessi and Panangaden use a similar definition in the context of anonymity protocols; anonymity leakage is defined as the conditional mutual information of the anonymity and the observables given knowledge of the data allowed to be leaked "by design" of the protocol.

Recently these information theoretical definitions have been questioned by Geoffrey Smith [29]. We will discuss Smith's criticism in Section 2.5.

Non information theoretical quantitative approaches to non-intereference have also recently been studied; Lowe [15] defines channel capacity in the context of CSP and defines a notion of rate of leakage as the ratio leaked information/elapsed time.

Compared with Boreale and Lowe's definitions the notion of rate presented in this paper abstracts time in a loop as number of iterations. As we will see there are drawbacks to this interpretation but more sophisticated interpretations of time require models richer than language based ones.

Di Pierro, Hankin, Wiklicky propose a probabilistic approach to approximate non-interference in a declarative setting [21] and more recently in distributed systems [22]. Their approach is to measure bisimilarity, roughly speaking the average number of runs necessary for the attacker to distinguish the two processes.

A probabilistic beliefs-based approach to non-interference has been suggested by Clarkson, Myers, Schneider [6]. Their work is centered around the attacker beliefs and the revision of such beliefs following experiments. For example, an attacker believing that the password is A will revise her beliefs if she is denied access to the system by entering A.

To the best of our knowledge this is the first work to provide tools for quantitative reasoning of loops in programming languages. Also, because of the relationship between unbounded covert channels and loops this paper provides an original quantitative analysis for covert channels in the context of programming languages.

1.4. Structure of the work

The article is structured as follows:

- Section 2 reviews some basic definitions from Information Theory and presents an interpretation of program variables and commands in terms of random variables.
- Section 3 relates leakage and expected security damage.
- Section 4 provides a justification of the Information Theoretical measures in this work. This justification is based on Markov inequality applied to the equivalence between leakage and expected security damage.
- Section 5 define an Information Theoretical formula for the leakage of the command `while e M`. From the leakage formula some definitions are derived, like rate of leakage, channel capacity, security, ratio of leakage.
- Based on these definitions Section 5.5 classifies loops according to their leakage and rate of leakage.
- Section 6 provides case studies justifying the usefulness of these notions.

2. Preliminaries

2.1. Entropy, interaction, interference

We begin by reviewing some basic concepts of Information Theory relevant to this work; additional background is readily available both in textbooks [7] and on the web (e.g. the wikipedia entry for Entropy).

Given a space of events with probabilities $P = (p_i)_{i \in N}$ (N a set of indices) the Shannon's entropy is defined as

$$H(P) = -\sum_{i \in N} p_i \log(p_i).$$

It is usually said that this number measures the average uncertainty of the set of events: if there is an event with probability 1 then the entropy will be 0 and if the distribution is uniform, i.e. no event is more likely than any other the entropy is maximal, i.e. $\log(|N|)$. The entropy of a random variable is the entropy of its distribution.

An important property of entropy which we will use says that if we take a partition of the events in a probability space, the entropy of the space can be computed by summing the entropy of the partition with the weighted entropies of the partition

sets. We call this
$S = \{s_{1,1}, \ldots, s_n$

$H(\mu(s_{1,1}), \ldots$

$= H(\mu(S$

where $\mu(S_i) = \sum$
Given two ranc
of all entropies o

$\sum_{Y=y} \mu(Y =$

where $H(X|Y =$
The higher H
to see that if X
$H(X|Y) = H($
Mutual inforn

$I(X;Y) =$

This quantity m
independent iff
Mutual infor
defined unary o
As we will s
be used to quar
relation betwee
defined as:

$I(X;Y|Z$

2.2. *Random v*

The languag
ments, sequen
language a pro
standard and c
language are a
with constants

provides an original quan-
ramming languages.

ıation Theory and presents
s in terms of random vari-

ge.
ı Theoretical measures in
ıality applied to the equiv-
e.
or the leakage of the com-
efinitions are derived, like
kage:
according to their leakage

ss of these notions.

ation Theory relevant to
in textbooks [7] and on

(N a set of indices) the

e uncertainty of the set
entropy will be 0 and if
n any other the entropy
ble is the entropy of its

hat if we take a partition
ce can be computed by
ntropies of the partition

sets. We call this the *partition property*; formally: given a distribution μ over a set $S = \{s_{1,1}, \dots, s_{n,m}\}$ and a partition of S in sets $(S_i)_{1 \leqslant i \leqslant n}$, $S_i = \{s_{i,1}, \dots, s_{i,m}\}$:

$$H(\mu(s_{1,1}), \dots, \mu(s_{n,m}))$$
$$= H(\mu(S_1), \dots, \mu(S_n)) \sum_{i=1}^{n} \mu(S_i) H\left(\frac{\mu(s_{i,1})}{\mu(S_i)}, \dots, \frac{\mu(s_{i,m})}{\mu(S_i)}\right),$$

where $\mu(S_i) = \sum_{1 \leqslant j \leqslant m} \mu(s_{i,j}) > 0$.

Given two random variables X, Y the conditional entropy $H(X|Y)$ is the average of all entropies of X conditioned to a given value for Y, $Y = y$, i.e.,

$$\sum_{Y=y} \mu(Y = y) H(X|Y = y),$$

where $H(X|Y = y) = -\sum_{X=x} \mu(X = x|Y = y) \log(\mu(X = x|Y = y))$.

The higher $H(X|Y)$ is the lower is the correlation between X and Y. It is easy to see that if X is a function of Y, $H(X|Y) = 0$ and if X and Y are independent $H(X|Y) = H(X)$.

Mutual information is defined as

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

This quantity measures the correlation between X and Y. For example, X and Y are independent iff $I(X;Y) = 0$.

Mutual information is a measure of binary *interaction*. In fact so far we have only defined unary or *binary* concepts.

As we will see conditional mutual information, a form of ternary interaction will be used to quantify *interference*. Conditional mutual information measures the correlation between two random variables conditioned on a third random variable; it is defined as:

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) = H(Y|Z) - H(Y|X,Z).$$

2.2. *Random variables and programs*

The language we are considering is a simple imperative language with assignments, sequencing, conditionals and loops. Further in the paper we will add to this language a probabilistic choice operator. Syntax and semantics for the language are standard and can be found in any good textbook, e.g. [33]. The expressions of the language are arithmetic expression, with constants 0, 1, ... and boolean expressions with constants `tt`, `ff`.

Following denotational semantics commands are state transformers, informally maps which change the values of variables in the memory and expressions are maps from the memory to values; we will denote by $[\![M]\!]$ the standard denotational semantics of the program M [33]. We assume there are two input variables h, l: the high (confidential) and low (public) input, and we assume that inputs are equipped with a probability distribution, so we can consider them as random variables (the input is the joint random variable (h, l)). A deterministic program M can hence be seen as a random variable itself, the output random variable where the probability on an output value of the program is the sum of probabilities of all inputs evaluating via M to that value $\mu(M = v) = \sum\{\mu(h = x, l = x') | [\![M]\!](x, x') = v)\}$.

More formally:

1. Our probability space is (Ω, A, μ) where

$$\Omega = \{\sigma \mid \sigma : \{h, l\} \to N\},$$

$A = \mathcal{P}(\Omega)$ (the power set) and μ a probability distribution over Ω.
An element $\sigma \in \Omega$ is a memory state (environment), i.e. a map from names of variables to values.
A state σ is naturally extended to a map from arithmetic expressions to N by

$$\sigma(e(x_1, \ldots, x_n)) = e(\sigma(x_1), \ldots, \sigma(x_n)),$$

i.e. the σ evaluation of an expression is the value obtained by evaluating all variables in the expression according to σ.

2. A random variable M is a partition (an equivalence relation) over Ω[3]. For a command M the equivalence relation would identify all σ which have the same observable state for the command; i.e. $\sigma \equiv_M \tau$ iff $M(\sigma) \restriction_{Ob} = M(\tau) \restriction_{Ob}$. Here we will take as observable the output values of the variable l, i.e. $Ob = l$; for example if M is the command $l = h$ then $\sigma \equiv_M \tau$ iff

$$\sigma_{[l=[\![h]\!]]} \restriction_{Ob} = \tau_{[l=[\![h]\!]]} \restriction_{Ob} \quad \text{iff} \quad \sigma_{[l=[\![h]\!]]} \restriction_{l} = \tau_{[l=[\![h]\!]]} \restriction_{l} .$$

The notation $\sigma_{x=[\![e]\!]}$ means σ where the variable x is evaluated to $[\![e]\!]$. Hence $\sigma_{[l=[\![h]\!]]} \restriction_l = \tau_{[l=[\![h]\!]]} \restriction_l$ holds for any σ, τ which agree (have the same value) on the variable h.
The probability distribution on a command random variable M is defined as

$$\mu(M = \tau') = \sum_{\tau \in \Omega} \{\mu(\tau) \mid M(\tau) \restriction_{Ob} = \tau' \restriction_{Ob}\}.$$

[3]The conventional mathematical definition of a random variable is that of a map from a probability space to a measurable space. In those terms we are considering the kernel of such a map.

If M is a non
equivalence rel
to the new obs
extended with

$\mu(M = \bot$

Instantiating t
ciated to parti

- M is the c
$\sigma_{x=[\![e]\!]} \restriction_{Ob}$
- M is `if e`
if $\sigma(e) =$ t
and $\tau(e) =$
$\tau(e) =$ `ff`

Given a comr

$M^n \equiv$

for the nth it
tion. For exa

$\mu((x =$

3. Similarly we
(we take as
set of states

$\sigma \equiv_e \tau$

$\mu(e =$

for example

$\sigma \equiv_{e_1 =}$

$\mu((e_1$

Given an e
e^n as e wh
For exampl
abbreviatio

If M is a non terminating program the definition of random variable as an equivalence relation still holds; now we will have an additional class associated to the new observable: all non-terminating states; the probability distribution is extended with the clause:

$$\mu(M = \perp) = \sum_{\tau \in \Omega} \{\mu(\tau) | M(\tau) = \perp\}.$$

Instantiating the above definition we get the following random variables associated to particular commands:

- M is the command $x = e$: this is the equivalence relation $\sigma \equiv_{x=e} \tau$ iff $\sigma_{\mathtt{x}=[\![\mathtt{e}]\!]} \restriction_{Ob} = \tau_{\mathtt{x}=[\![\mathtt{e}]\!]} \restriction_{Ob}$.
- M is `if e c else c'`: then $\sigma \equiv_{\mathtt{if\ e\ c\ else\ c'}} \tau$ iff if $\sigma(e) = \mathtt{tt} \neq \mathtt{ff} = \tau(e)$ then $[\![c]\!](\sigma) \restriction_{Ob} = [\![c']\!](\tau) \restriction_{Ob}$ and $\sigma(e) = \tau(e)$ and $\tau(e) = \mathtt{tt}$ implies $\sigma \equiv_c \tau$ and $\tau(e) = \mathtt{ff}$ implies $\sigma \equiv_{c'} \tau$.

Given a command M we will use the random variable

$$M^n \equiv M; \ldots; M$$

for the nth iteration of M. This is a generalization of the sequential composition. For example, $\sigma \equiv_{(x=x+1)^5} \tau$ iff $\sigma \equiv_{x=x+5} \tau$ and

$$\mu((x = x + 1)^5 = \sigma) = \sum \{\mu(\tau) | (x = x + 5)(\tau) \restriction_{Ob} = \sigma \restriction_{Ob}\}.$$

3. Similarly we will have random variables corresponding to boolean expressions (we take as boolean values the integers 0, 1); again an equivalence class is the set of states evaluated to the same (boolean) value:

$$\sigma \equiv_e \tau \Leftrightarrow \sigma(e) = \tau(e),$$
$$\mu(e = \mathtt{tt}) = \sum_{\tau \in \Omega} \{\mu(\tau) | \tau(e) = \mathtt{tt}\},$$

for example, for $e_1 == e_2$

$$\sigma \equiv_{e_1 == e_2} \tau \Leftrightarrow \sigma(e_1) = \sigma(e_2) = \tau(e_1) = \tau(e_2),$$
$$\mu((e_1 == e_2) = \mathtt{tt}) = \sum_{\tau \in \Omega} \{\mu(\tau) | \tau(e_1) = \tau(e_2)\}.$$

Given an expression e guarding a command M we define the random variable e^n as e where the variables in e are evaluated following $n - 1$ iterations of M. For example, if e is $x > 0$, M is $x = x + 1$ then e^3 is $x + 2 > 0$. e is hence an abbreviation for e^1.

2.3. *Defining leakage*

Following [3] and inspired by works by Dennings, McLean, Gray, Millen [9,10, 19,20,31], *interference* (or leakage of confidential information) in a program M is defined as

$$I(o; h|l),$$

i.e. the conditional mutual information between the output o and the high input h of the program given knowledge of the low input l.

Notice:

1. o is just another name for the random variable corresponding to the program seen as a command, i.e. $o = M$.
2. For deterministic programs we have

$$\begin{aligned} I(o; h|l) &= H(o|l) - H(o|h, l) \\ &= H(o|l) - H([\![M]\!](h, l)|h, l) \\ &= H(o|l), \end{aligned}$$

   i.e. interference becomes the uncertainty in the output of the program M given knowledge of the low input.

To see why $H(o|l)$ is not enough for measuring leakage in nondeterministic setting, consider the following simple program: `l = random(0, 1)`, i.e. the output is 0 or equally likely 1. Since the output is independent from the inputs $H(o|l) = H(o)$and $H(o) = 1$. So we would conclude that there is 1 bit of leakage. This is clearly false as there is no secret information in the program. However,

$$I(o; h|l) = H(o|l) - H(o|h, l) = H(o) - H(o) = 1 - 1 = 0.$$

A further simplification to the definition of leakage is provided by considering (when possible) programs where the low inputs are initialized inside the program. In this case the dependency on the variable l disappear and the leakage formula $I(o; h|l)$ for a deterministic program is hence equivalent to $H(o)$.

2.4. *What an attacker can do*

The validity of every security model is constrained by the capability of the attacker in the model. Clearly the model presented here is inadequate to assess how secure a system is against an attacker using firearms to force an employee to reveal a password. Similarly this model does not cater for power consumption attacks where the

amount of power u
associated informa
Basically the att
about the secret, th
has the following c

1. Can choose v
2. Knows the p
3. Knows the c
4. Can observe

Notice howeve
studied in this mo
measure leakage v
tion in multithreac

2.5. *About Smiths*

Geoffrey Smith
Shannon's inform
consider the prog

P = if (h%

and

Q = 1 = h

assuming uniforn
the whole 8k-bit
reveals at any att
the remaining $7k$
From a probal
the second one,
probability $\frac{1}{8}$) fo
However as S
$k + 0.169$ which
Smith hence s
probability that
formalize this qu
His definition
min-entropy is

$\log \frac{\sum_{l \in L}}{\text{ma}}$

amount of power used by the system is used to guess the computational path and the associated information leak of the secret.

Basically the attacker considered here can only use available public information about the secret, the code and public data for the program. In our model the attacker has the following capabilities:

1. Can choose which low inputs to run the program on.
2. Knows the probability distribution of the secret.
3. Knows the code of the program.
4. Can observe the output of the program.

Notice however that, as shown Section 6.1, some kind of timing attacks can be studied in this model. Also in [12] it has been shown that this model can be used to measure leakage when the attacker can observe some intermediate state of computation in multithreaded programs.

2.5. *About Smiths' leakage as min-entropy*

Geoffrey Smith [29] has recently questioned the definition of leakage in terms of Shannon's information theory. His argument is supported by the following example: consider the programs P and Q where all variables are 8k-bits.

$$\mathtt{P} = \mathtt{if\,(h\%8 == 0)\; l = h;\; else\; l = 1;}$$

and

$$\mathtt{Q} = \mathtt{l = h\; \&\; 0^{7k-1}1^{k+1}}$$

assuming uniform distribution on the secret in the first example an attacker will know the whole 8k-bits of the secret with probability $1/8$ whereas the second program reveals at any attempt the last $k + 1$ bits of the secret but doesn't say anything about the remaining $7k - 1$ bits.

From a probabilistic point of view then the first program is a bigger threat than the second one, because even if no bit is guaranteed to be leaked it is likely (with probability $\frac{1}{8}$) for an attacker to guess the secret.

However as Smith points out the leakage of P according to Shannon entropy is $k + 0.169$ which is less than the leakage of Q which is $k + 1$.

Smith hence suggests an alternative measure of leakage based on "the worst-case probability that an adversary A could guess the value of X correctly in one try", and formalize this quantity using Renyi's min-entropy [23].

His definition of leakage for deterministic programs, a formulation of conditional min-entropy is

$$\log \frac{\sum_{l \in L} \max_{h \in H_l} \mu(H = h)}{\max_{h \in H} \mu(H = h)},$$

where

$$H_l = \{h \in H \mid \mu(L = l|H = h) = 1\}.$$

(Notice here we are using Smith's notation; in his notation the variable l corresponds to the output of the program, hence H_l is the set of high inputs h which produce the output l.)

Using Smith's definition the leakage of P becomes

$$\log \frac{(2^{8k-3} + 1)/2^{8k}}{1/2^{8k}} = \log(2^{8k-3} + 1),$$

so the leakage is $\sim 8k - 3$, a huge increase from $k + 0.169$ whereas the leakage of Q stays the same at $k + 1$.

We have two observations about Smith's argument:

A. Let us first consider the program P. That program on average leaks all $8k$ bits of the secret once every eight attempts, so it is reasonable to say that one attempt on average contributes around $\frac{1}{8}$ of the $8k$ bits of the secret to the leakage: henceforth Shannon's leakage $\sim k$ is a reasonable measure of leakage for P. Similarly $k + 1$ is clearly a meaningful number to quantify the leakage of Q, considering the fact that Q releases the last $k + 1$ bits of the secret in any attack. The problem arising from Smith's argument is if these numbers reflects the threats posed by those two programs: the point is that *those threats are very different in nature*. Smith has in mind an attacker model based on "the expected probability that an adversary A could guess the value of X correctly in one try", hence he sees P as posing a bigger threat than Q. In a different model of the attacker however the program Q may in fact represent a bigger threat because $k + 1$ bits are *guaranteed* to be released by running Q once whereas no bits is *guaranteed* to be released by running P once. The problem with security is that we should always be clear about the attacker model we are considering. This paper argue that Shannon's notion of leakage can be seen as a risk-assessment approach to computer security where guaranteed leakage of bits and probabilities of guessing the secret are combined. As is the case in risk management in real situations risk assessment is usually complemented by other models that address specific threats. We see hence Smiths definition as complementing, not antagonizing the definition of leakage presented in this paper.

B. Notice also that the same leakage of $\sim 8k - 3$ for P is also achievable using Shannon's definition; for that is enough to choose, instead of the uniform distribution on the secret the distribution μ such that

$$\mu(h) = \frac{1}{\alpha + 1} \quad \text{if } h\%8 = 0,$$

:ation the variable l corre-
ɜt of high inputs h which

69 whereas the leakage of

n average leaks all $8k$ bits
·le to say that one attempt
:ret to the leakage: hence-
f leakage for P. Similarly
kage of Q, considering the
attack. The problem aris-
he threats posed by those
ērent in nature. Smith has
bility that an adversary A
: sees P as posing a bigger
/er the program Q may in
:ranteed to be released by
ed by running P once. The
ɔout the attacker model we
of leakage can be seen as
guaranteed leakage of bits
.s is the case in risk man-
ılemented by other models
.on as complementing, not
per.

P is also achievable using
of the uniform distribution

$$\mu(h) = \frac{1}{(2^{8k} - \alpha)(\alpha + 1)} \quad \text{if } h\%8 \neq 0, \tag{1}$$

where $\alpha = 2^{8k-3}$. In fact, as shown in [17] this is the channel capacity (maximal leakage) for this program. Hence if we interested in the "worst that can happen", i.e. the maximal possible leakage, the two definitions seems to coincide.

Smith's argument highlights the subtle foundational problems associated to a "universal" theory of security encompassing all attackers models. We strongly agree with his words about "the difficulty of measuring a range of complex threat scenarios precisely using a single number" [29].

3. Leakage as expected security damage

We now define the random variable D as the *damage* associated to observables. In this work we restrict ourselves to observe only output values, hence the observables are associated to the output events of the random variable o.

Given an observable d define $D_{l,h,o}(d)$ as

$$H(h|l) - H(h|o = d, l),$$

i.e. the difference between the secret before and after the observable event d occurred. In other words:

> the damage associated to an event is the change in uncertainty about the secret caused by the happening of that event.

D will denote $D_{l,h,o}$ when no ambiguities arise.

The following proves that Leakage and the expected value of the random variable Damage are equivalent,[4]

Proposition 1. $I(h; o|l) = E(D)$.

Proof. We have

$$\begin{aligned} E(D) &= \sum_d \mu(o = d)(H(h|l) - H(h|o = d, l)) \\ &=_0 H(h|l) - \sum_d \mu(o = d) H(h|o = d, l) \\ &=_1 H(h|l) - H(h|o, l) \\ &=_2 I(h; o|l). \end{aligned}$$

[4]We assume here that observables and low inputs are independent, i.e. $\mu(o = d, l = x) = \mu(o = d)\mu(l = x)$.

The steps are justified as follows:

0. The d do not intervene in $H(h|l)$ so the sum can move to the right.
1. By definition of conditional entropy and independence between observables and low inputs.
2. By definition of conditional mutual information. □

Example. Consider the password program

```
if (h == 0) access else deny
```

with the following probability distribution of the secret:

$$\mu(h=0) = 1/2, \qquad \mu(h=i) = 1/6, \quad i = 1,2,3.$$

The secret is now not 2 bits but $H(\frac{1}{2},\frac{1}{6},\frac{1}{6},\frac{1}{6}) = 1.79248125$ bits. The values of the variable damage are:

$$D(\mathtt{access}) = 1.79248125 - 0 = 1.79248125,$$
$$D(\mathtt{deny}) = 1.79248125 - H\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right) = 1.79248125 - 1.584962501$$
$$= 0.20751875$$

the expected damage is hence

$$E(D) = \frac{1}{2}1.79248125 + \frac{1}{2}0.20751875 = 0.896240625 + 0.103759375 = 1.$$

Notice the difference from the leakage formula for the same program:

$$I(o;h|l) = H(o) = H\big(\mu(o = \mathtt{access}), \mu(o = \mathtt{deny})\big)$$
$$= H\left(\frac{1}{2},\frac{1}{2}\right) = \frac{1}{2}\log 2 + \frac{1}{2}\log 2$$

although as Proposition 1 proves, their values coincide.

Notice also the damage can be negative

This corresponds to events whose happening does actually increase the uncertainty, i.e. after that event happening the system is *more secure*. As an example consider again the password program but suppose now that there are 101 possible values

for the secret; the
0.001. Then the da

$H(h) - H(h$

The reason for th
then the average
secret in a subset
average number c

$G = \sum_{1\leqslant i\leqslant n}$

where the probab
then we get 0.99
first element and
guesses will be (
(the first elemen
guess the secret
attempts!

3.1. A note on p

When a prog
not a simple m
probabilistic op
puted by taking
this idea is use
choices are obs
alent single-thr
scheduler choi
is computed as

4. Markov in

We aim now
damage above
damage we ca

for the secret; the first having probability 0.99 the remaining 100 having probability 0.001. Then the damage for the event $\mathtt{deny}$ is

$$H(h) - H(h|o = \mathtt{deny}) = H\left(\frac{99}{100}, \frac{1}{1000}, \ldots, \frac{1}{1000}\right) - H\left(\frac{1}{100}, \ldots, \frac{1}{100}\right)$$
$$= 1.0109 - 6.643 = -5.6321.$$

The reason for this is that if it almost certain that the password is the first element then the average search for the secret will be much shorter than searching for the secret in a subset with all element having equal probability. In fact by computing the average number of guesses for the original secret using the formula

$$G = \sum_{1 \leqslant i \leqslant n} i\mu(o_i),$$

where the probabilities of the observations $\mu(o_i)$ are decreasing, i.e. $\mu(o_i) \geqslant \mu(o_{i+1})$ then we get $0.99 + (1/1000) \times (103 \times 100/2) = 6.14$ whereas if we eliminate the first element and only have 100 elements equiprobable then the average number of guesses will be $(1/100) \times (101 \times 100/2) = 50.5$. Hence in the case with 101 elements (the first element having probability 0.99) it will take on average only 6 attempts to guess the secret whereas if the first element is removed it will take on average 50 attempts!

3.1. *A note on probabilistic and multithreaded programs*

When a program contains a probabilistic operator, its denotational semantics is not a simple map. In Section 6.7 it will be shown how leakage for programs with probabilistic operators (where the probabilistic choices are observable) can be computed by taking the average leakage over all possible (deterministic) runs. In [12] this idea is used to compute leakage for multithreaded programs where the thread choices are observable: first a multithreaded program is transformed into an equivalent single-threaded program with a probabilistic operator (the operator reflect the scheduler choice of which thread to execute). After this transformation the leakage is computed as the leakage of a looping program with a probabilistic operator.

4. Markov inequality and security

We aim now to relate leakage and the probability that an attack causes a security damage above some threshold. Using the equivalence between leakage and expected damage we can use the celebrated Markov Inequality to study these relationships.

We use $\mathcal{L}$ to refer to the leakage; because of the equivalence between leakage and expected damage (Proposition 1) we can state the Markov[5] inequality as follows:

> The probability of the damage exceeding a constant is less than the leakage divided by the constant.

Formally

$$\mu(D \geqslant c) \leqslant \frac{\mathcal{L}}{c}.$$

Examples:

1. The probability of leaking at least 1 bit is not greater than the leakage.
2. In a program which does a linear search for the secret the probability of leaking the whole secret is bounded by 1.

We can use Markov inequality also to provide a lower bound for the leakage in terms of probability of damage exceeding a threshold:

$$\mathcal{L} \geqslant c\mu(D \geqslant c).$$

Example. In a linear search program we know the probability of the damage being the whole secret (k bits) is 1 so the leakage has to be the whole secret ($E(D) \geqslant k\mu(D \geqslant k) = k.1 = k$).

Using Markov inequality it is easy to prove the following fundamental result:

> impossibility of damage, null leakage and non-interference are equivalent.

Formally:

Proposition 2. *The following are equivalent:*

1. $\mu(D > 0) = 0$.
2. $\mathcal{L} = 0$.
3. *The program is not interfering.*

Proof.

(1 $\Leftarrow$ 2) Suppose $\mathcal{L} = 0$. Then for all $n > 0$, $nE(D) = 0$, hence $\mu(D \geqslant \frac{1}{n}) \leqslant nE(D) = 0$, so for all n, $\mu(D \geqslant \frac{1}{n}) = 0$ and we conclude $\mu(D > 0) = 0$.

(1 $\Rightarrow$ 2) If $\mu(D > 0) = 0$ then $\sum_{D(d)>0} \mu(D(d)) = 0$ hence all terms in $E(D)$ are 0 so $E(D) = 0$, i.e. $\mathcal{L} = 0$.

[5] In this section we assume that the random variable damage is non-negative.

(2 $\Leftrightarrow$ 3) This equivalence was proven in [5]. The idea is that the denotation of a non interfering program is a function which is constant on the high component, i.e. $\forall h, h', l\ f(l,h) = f(l,h')$. This is the same as $H(o|l) = 0$ because all information on the result comes from the low input. □

Other justifications for Information Theoretical measure of interference can be found in the Information Theory literature. A very relevant one is provided by Massey [16,18] who has shown that entropy (and so leakage) can be related to the average number of attempts needed to guess the secret using a dictionary attack. An alternative line of justification in the context of measuring anonymity is provided by Chatzikokolakis, Palamidessi and Panangaden who relate Information Theoretical measure of anonymity with null hypothesis testing [2].

5. Analysis of loops

5.1. *Loops as disjoint union of functions*

Entropy of disjoint union of functions

Consider a function $f : X \to Y$, which is the union of a family of functions $(f_i)_{i\in I}$ with disjoint domains $(\delta f_i)_{i\in I}$, i.e. for each $i, \delta f_i \subseteq X$ is the domain of f_i and $(\delta f_i)_{i\in I}$ is a partition of X.

We will note f by $\sum_{i\in I}(f_i|\delta f_i)$ when we want to stress that $f(x) = f_i(x)$ for the unique i such that $x \in \delta f_i$.

Define $\{[y] = f^{-1}(y) \mid y \in Y\}$; clearly this is also a partition of X, the partition corresponding to the function f. Define the entropy of f as the entropy of its inverse images, i.e. $H(\mu([y_1]), \ldots, \mu([y_n]))$. The aim now is to characterize the entropy of f.

Assume that f is *collision free*, i.e. the family $(f_i)_{i\in I}$ has also disjoint codomains. In that case $(\delta f_i)_{i\in I}$ can also be seen as a partition on the partition $[y] = f^{-1}(y)$: δf_i is the set of all $[y]$ for y in the codomain of f_i. Let us write $[y_1]^j, \ldots, [y_m]^j$ for the classes in δf_j

From now on to ease the notation we will often use events instead of their probability when no confusion arise, for example in a computation $[y]$ will stand for $\mu[y]$ the probability of the event $[y]$, i.e. $\sum\{\mu(x)|x \in [y]\}$. Similarly $H([y_1], \ldots, [y_n])$ will stand for $H(\mu[y_1], \ldots, \mu[y_n])$, etc.

Proposition 3. *For a collision free function f:*

$$H([y_1], \ldots, [y_n]) = H(\delta f_1, \ldots, \delta f_n) + \sum_{j\in I} \delta f_j H\left(\frac{[y_1]^j}{\delta f_j}, \ldots, \frac{[y_m]^j}{\delta f_j}\right).$$

Proof. We use the information theoretical equality

$$H(Y) = H(X) + H(Y|X) - H(X|Y) \tag{2}$$

with $X = (\delta f_i)_{i \in I}, Y = ([y])_{y \in Y} = \{f^{-1}(y) | y \in Y\}$.

We have then the following equalities:

$$\begin{aligned} H(Y) = {} & H(X) + H(Y|X) - H(X|Y) \\ = {} & H((\delta f_i)_{i \in I}) + H(([y])_{y \in Y} | (\delta f_i)_{i \in I}) - H((\delta f_i)_{i \in I} | ([y])_{y \in Y}) \\ =_A {} & H((\delta f_i)_{i \in I}) + H(([y])_{y \in Y} | (\delta f_i)_{i \in I}) - 0 \\ =_B {} & H((\delta f_i)_{i \in I}) + \sum_{j \in I} \delta f_j H\left(\frac{[y_1]^j}{\delta f_j}, \ldots, \frac{[y_m]^j}{\delta f_j}\right), \end{aligned}$$

where:

A is justified by the fact that if there are no collisions given a value $[y]$ there is a unique i such that $[y] \in \delta f_i$, hence $H((\delta f_i)_{i \in I} | ([y])_{y \in Y}) = 0$.

B is justified by the fact that when a particular outcome δf_i is chosen the only possible values for $([y])_{y \in Y}$ are $[y_1]^i, \ldots, [y_m]^i$. □

Let us now consider the case where f has collisions. Remember a collision is a value $y \in Y$ belonging to the image of two or more different functions, i.e. $[y] \cap \delta f_j \neq \emptyset \neq [y] \cap \delta f_i$ for $i \neq j$. In this case let us define Y' as Y extended with enough new elements to eliminate collisions and let $f' : X \to Y'$ be the derived function with no collisions, so f' is the union of the family of functions $(\delta f'_i)_{i \in I}$ with disjoint domain and codomain. f'_i is defined as

$$f'_i(x) = \begin{cases} f_i(x) & \text{if } \forall j \neq i,\ f_i(x) \neq f_j(x), \\ (f_i(x), i) & \text{otherwise}, \end{cases}$$

where $(f_i(x), i)$ are the new elements added to Y. Let us call *disambiguation* of f the function f'.

Let us define $C_f(Y)$ as the set of collisions of f in Y, and write $x_1^y, \ldots, x_m^y$ for the elements of $[y]$. By using again the partition property we have:

Proposition 4.

$$H([y_1], \ldots, [y_n]) = H([y'_1], \ldots, [y'_{n'}]) - \sum_{y \in C_f(Y)} [y] H\left(\frac{x_1^y}{[y]}, \ldots, \frac{x_m^y}{[y]}\right).$$

Proof. We again use the equality (2): we now take

$$X = (\delta f'_i)_{i\in I}, \qquad Y = ([y])_{y\in Y} = \{f^{-1}(y) | y \in Y\},$$

where $(\delta f'_i)_{i\in I}$ is the domain of the disambiguation of f.

We have then

$$\begin{aligned}
H(Y) = \;& H(X) + H(Y|X) - H(X|Y) \\
=\;& H((\delta f'_i)_{i\in I}) + H(([y])_{y\in Y}|(\delta f'_i)_{i\in I}) - H((\delta f'_i)_{i\in I}|([y])_{y\in Y}) \\
=_A\;& H((\delta f'_i)_{i\in I}) + H(([y'])_{y\in Y}|(\delta f'_i)_{i\in I}) - H((\delta f'_i)_{i\in I}|([y])_{y\in Y}) \\
=_B\;& H((\delta f'_i)_{i\in I}) + \sum_{j\in I} \delta f'_j H\left(\frac{[y'_1]^j}{\delta f'_j}, \dots, \frac{[y'_m]^j}{\delta f'_j}\right) \\
& - \sum_{y\in C_f(Y)} [y] H\left(\frac{x^y_1}{[y]}, \dots, \frac{x^y_m}{[y]}\right),
\end{aligned}$$

where

A Is justified by the fact that when a particular $\delta f'_i$ is chosen, $([y])_{y\in Y}$ and $([y'])_{y\in Y}$ have the same set of possible values (and each value has the same probability in $([y])_{y\in Y}$ and $([y'])_{y\in Y}$).

B Is justified by the fact that when a particular outcome $[y]$ for f is fixed there is uncertainty about which $\delta f'_j$ it belongs to only if it is a collision $\{x^y_1, \dots, x^y_m\}$ and in that case the uncertainty is $H(\frac{x^y_1}{[y]}, \dots, \frac{x^y_m}{[y]})$. □

The proposition says that the entropy of a function defined as a union of functions with disjoint domains is given by the entropy of the derived function with no collisions minus the weighted sum of the entropies of the collisions. To ease the notation we can rewrite Proposition 4 as $H(f) = H(f') - \sum H(C_f(Y))$.

Let us consider the simple case illustrated in Fig. 1.

The function f there depicted can be seen as the sum of two functions, one with domain a_1, a_2 and the other with domain b_1, b_2, b_3. There is one collision, i.e. the point whose inverse image is the set a_2, b_1. The entropy of f is $H(a_1, a_2+b_1, b_2+b_3)$.

In the following computation a, c, b, d will abbreviate $a_1 + a_2, a_2 + b_1, b_1 + b_2 + b_3, b_2 + b_3$ respectively. We have then:

$$\begin{aligned}
& H(f') - \sum H(C_f(Y)) \\
&\quad = H(a_1 + a_2, b_1 + b_2 + b_3) \\
&\qquad + aH\left(\frac{a_1}{a}, \frac{a_2}{a}\right) + bH\left(\frac{b_1}{b}, \frac{d}{b}\right) - cH\left(\frac{a_2}{c}, \frac{b_1}{c}\right)
\end{aligned}$$

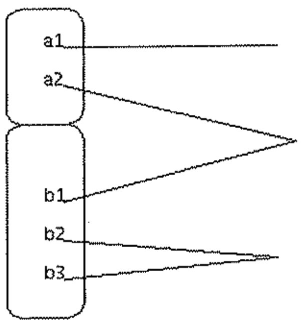


Fig. 1. A collision in a function f.

$$
\begin{aligned}
&= -a\log a - b\log b - a_1\log\frac{a_1}{a} - a_2\log\frac{a_2}{a} - b_1\log\frac{b_1}{b} - d\log\frac{d}{b}\\
&\quad + a_2\log\frac{a_2}{c} + b_1\log\frac{b_1}{c}\\
&= -a\log a - b\log b - a_1\log\frac{a_1}{a} - a_2\left(\log\frac{a_2}{a} - \log\frac{a_2}{c}\right)\\
&\quad - b_1\left(\log\frac{b_1}{b} - \log\frac{b_1}{c}\right) - d\log\frac{d}{b}\\
&= -a\log a - b\log b - a_1\log\frac{a_1}{a} - a_2\log\frac{c}{a} - b_1\log\frac{c}{b} - d\log\frac{d}{b}\\
&= a\log a - b\log b - a_1\log a_1 - a_2\log c - b_1\log c - d\log d + a_1\log a\\
&\quad + a_2\log a + b_1\log b + d\log b\\
&= -a\log a - b\log b - a_1\log a_1 - a_2\log c - b_1\log c - d\log d + a\log a\\
&\quad + b\log b\\
&= -a_1\log a_1 - c\log c - d\log d\\
&= H(a_1, a_2 + b_1, b_2 + b_3).
\end{aligned}
$$

Notice also that Proposition 4 implies that the entropy of f is a lower bound on the entropy of the disambiguation of f.

As a further example let us consider the function $f = f_1 \oplus f_2 \oplus f_3$ defined by

$$
\begin{aligned}
&f_1(x_1) = y_1, \qquad f_1(x_2) = y_2 = f_2(x_3), \qquad f_2(x_4) = y_4,\\
&f_3(x_5) = y_5 = f_3(x_6)
\end{aligned}
$$

and assume uniforr
$H(f)$ we first exten

$f'_1(x_2) = y_2,$

Computing $H(f)$

$H(f) = H(f')$

$= H\Big(\frac{1}{3}$

$= 1.585$

$= 1.918$

5.2. Entropy of loc

Let `while e I`
it as a map

$F = \sum_{0\leqslant i\leqslant n} F$

where n is an upp
disjoint domains: (
that the guard has
of M. The domain

$\{\sigma|M^j(\sigma)(e)$

We can hence re

`while` $e\ M$

where

$e^{(i)} = \begin{cases} e = \\ e = \end{cases}$

and $M^0 =$ `skip.`

$b_1 \log \frac{b_1}{b} - d \log \frac{d}{b}$

$- \log \frac{a_2}{c} \Big)$

$_1 \log \frac{c}{b} - d \log \frac{d}{b}$

$g\, c - d \log d + a_1 \log a$

$\log c - d \log d + a \log a$

y of f is a lower bound on

$f_1 \oplus f_2 \oplus f_3$ defined by

$) = y_4,$

and assume uniform distribution on the inputs. f has one collision y_2 so to compute $H(f)$ we first extend the codomain with a new element y_2' so to have

$$f_1'(x_2) = y_2, \qquad f_2'(x_3) = y_2'.$$

Computing $H(f)$ using Proposition 4 gives:

$$\begin{aligned} H(f) &= H(f') - \sum H(C_f(Y)) \\ &= H\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) + 2\frac{1}{3} H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{1}{3} H(1,0) - \frac{1}{3} H\left(\frac{1}{2}, \frac{1}{2}\right) \\ &= 1.585 + \frac{2}{3} + 0 - \frac{1}{3} \\ &= 1.918. \end{aligned}$$

5.2. *Entropy of loops*

Let `while e M` be a terminating loop. Using denotational semantics we can see it as a map

$$F = \sum_{0 \leqslant i \leqslant n} F_i, \tag{3}$$

where n is an upper bound on the number of iterations of the loop and all $\mathtt{F_i}$ have disjoint domains: each F_i is the map which iterates M i times under the condition that the guard has been true up to that moment and it will be false after ith iteration of M. The domain of F_i is hence defined as

$$\{\sigma | M^j(\sigma)(e) = tt \text{ if } 0 \leqslant j < i \text{ and } M^i(\sigma)(e) = ff\}.$$

We can hence rewrite (3) as

$$\mathtt{while}\, e\, M = \sum_{0 \leqslant i \leqslant n} (M^i | e^{\langle i \rangle}), \tag{4}$$

where

$$e^{\langle i \rangle} = \begin{cases} e = ff & \text{if } i = 0, \\ e = tt, \dots, e^i = tt \wedge e^{i+1} = ff & \text{if } i > 1 \end{cases}$$

and $M^0 = \mathtt{skip}$.

Notice:

1. $e^{\langle i \rangle}$ are events and not random variables.
2. The assumption that n is an upper bound on the number of iterations of the loop implies

$$\sum_{0 \leqslant i \leqslant n} \mu(e^{\langle i \rangle}) = 1.$$

Lemma 1. *The events $e^{\langle 0 \rangle}, \ldots, e^{\langle n \rangle}$ constitute a partition of the set of states.*

Proof. By assumption given any initial state σ the loop will terminate in $\leqslant n$ iterations; exactly one of the $e^{\langle i \rangle}$ must be true for σ, i.e. $\sigma \in e^{\langle i \rangle}$, e.g. for $i > 1$ $\sigma(e) = tt \wedge \cdots \wedge M^i(\sigma)(e) = \mathtt{tt} \wedge M^{i+1}(\sigma)(e) = \mathtt{ff}$.

To prove that this is a partition suppose it is not, i.e. $\sigma \in e^{\langle i \rangle} \cap e^{\langle i+j \rangle}$; then $M^{i+1}\sigma(e) = \mathtt{ff}$ because of $e^{\langle i \rangle}$ and $M^{i+1}\sigma(e) = \mathtt{tt}$ because of $e^{\langle i+j \rangle}$: a contradiction, hence the $e^{\langle i \rangle}$ are disjoint sets, i.e. a partition. □

Proposition 5. *For a collision free loop* `while e M` *bounded by n iterations*

$$H(\mathtt{while\ e\ M}) = H(\mu(e^{\langle 0 \rangle}), \ldots, \mu(e^{\langle n \rangle})) + \sum_{1 \leqslant i \leqslant n} \mu(e^{\langle i \rangle}) H(M^i | e^{\langle i \rangle}).$$

Proof. By Lemma 1, Eq. (4) and Proposition 3. □

Proposition 6. *For a command* `while e M` *bounded by n iterations*

$$H(\mathtt{while\ e\ M}) \leqslant H(\mu(e'^{\langle 0 \rangle}), \ldots, \mu(e'^{\langle n \rangle})) + \sum_{1 \leqslant i \leqslant n} \mu(e'^{\langle i \rangle}) H(M'^i | e'^{\langle i \rangle})$$

with equality iff the loop is collision free.

Proof. In the case of a loop with collisions, following Proposition 4 equality is achieved as follows:

$$\begin{aligned} H(\mathtt{while\ e\ M}) = {} & H(\mu(e'^{\langle 0 \rangle}), \ldots, \mu(e'^{\langle n \rangle})) \\ & + \sum_{1 \leqslant i \leqslant n} \mu(e'^{\langle i \rangle}) H(M'^i | e'^{\langle i \rangle}) \\ & - \sum_{\sigma \in C_{\mathtt{while\ e\ M}}(\Omega)} [\sigma] H\left(\frac{\tau_1^\sigma}{[\sigma]}, \ldots, \frac{\tau_m^\sigma}{[\sigma]}\right). \end{aligned}$$

The term $\sum_{\sigma\in C_{\mathtt{while\ e}\ M}(\Omega)}[\sigma]H(\frac{\tau_1^\sigma}{[\sigma]},\ldots,\frac{\tau_m^\sigma}{[\sigma]})$ is always positive which proves the inequality and is 0 iff there are no possible outputs coming from two possible inputs $\sigma \in e^{\langle i\rangle}$ and $\sigma' \in e^{\langle j\rangle}$, i.e. the loop is collision free. □

Collisions do not present a conceptual change in the framework but add some computational burden; also most loops do not contain collisions; for a collision in a loop to arise two different iteration of the loop should give the same values for all read and written low variables in the loop and the guard should be false on these values. For example, all loops using a counter, a variable taking a different value at each iteration do not contain collisions.

For these reason from now on we will concentrate on collision free loops.

5.3. *Basic definitions*

Definition 1. *Define the leakage of* `while e M` *up to n iterations by*

$$\mathtt{W(e,M)_n} = H\left(\mu(e^{\langle 0\rangle}),\ldots,\mu(e^{\langle n\rangle}), 1 - \sum_{0\leqslant i\leqslant n}\mu(e^{\langle i\rangle})\right) + \sum_{1\leqslant i\leqslant n}\mu(e^{\langle i\rangle})H(M^i|e^{\langle i\rangle}).$$

Proposition 7. $\forall \mathtt{n} \geqslant 0,\ \mathtt{W(e,M)_n} \leqslant \mathtt{W(e,M)_{n+1}}$.

Proof. We only need to prove

$$H(\mu(e^{\langle 0\rangle}),\ldots,\mu(e^{\langle n\rangle}), 1 - \sum_{0\leqslant i\leqslant n}\mu(e^{\langle i\rangle})) \leqslant H(\mu(e^{\langle 0\rangle}),\ldots,\mu(e^{\langle n\rangle}), \mu(e^{\langle n+1\rangle}), 1 - \sum_{0\leqslant i\leqslant n+1}\mu(e^{\langle i\rangle}))$$

which can be rewritten as

$$H(p_1,\ldots,p_n,q_{n+1}+p_{n+1}) \leqslant H(p_1,\ldots,p_n,p_{n+1},q_{n+1})$$

the inequality then follows from

$$\begin{aligned} &H(p_1,\ldots,p_n,p_{n+1},q_{n+1}) \\ &\quad = H(p_1,\ldots,p_n,p_{n+1}+q_{n+1}) \\ &\qquad + (p_{n+1}+q_{n+1})H\left(\frac{p_{n+1}}{p_{n+1}+q_{n+1}},\frac{q_{n+1}}{p_{n+1}+q_{n+1}}\right). \end{aligned}$$ □

The *leakage* of `while e M` is defined as

$$\lim_{n\to\infty} \mathtt{W(e,M)}_n. \tag{5}$$

In the case of a loop with collisions the definition is modified in the obvious way:

$$\lim_{n\to\infty} \mathtt{W'(e,M)}_n - \sum H(C(\mathtt{W'(e,M)})), \tag{6}$$

i.e. we first compute the leakage in the disambiguation of the loop and then we subtract the weighted entropies of the collisions

The *rate of leakage* is

$$\lim_{n\to\infty,\mu(e^{\langle n\rangle})\neq 0} \frac{\mathtt{W(e,M)}_n}{n}.$$

Hence in the case of terminating loops the rate will be the total leakage divided by the number of iterations. This can be considered a rough measure of rate: for example, if the first iteration were to leak all secret and the following billion nothing the rate would still be one billionth of the secret size. However as in our model the attacker can only perform observations on the output and not on intermediate states of the program the chosen definition of rate will give indication of the timing behavior of the channel in that context.

A fundamental concept in Information Theory is *channel capacity*, i.e. the maximum amount of leakage over all possible input distributions, i.e.

$$\max_{\mu} \lim_{n\to\infty} \mathtt{W(e,M)}_n. \tag{7}$$

In our setting we will look for the distribution which will maximize leakage. Informally such a distribution will provide the setting for the most devastating attack: we will refer to this as the *channel distribution.*

Also we will use the term *channel rate* for the rate of leakage of the channel distribution. Again this should be thought of as the average maximal amount of leakage per iteration.

To define rate and channel capacity on the case of collisions the above definitions should be applied on the definition of leakage for loops with collisions.

5.4. Leakage vs security

Consider a simple assignment `l` $=$ `h` where the variables are k-bit variables. We know that the assignment transfers all information from `h` to `l`, so we would be tempted to say that the leakage is k. That is not correct. Suppose `h` is a 3-bit variable (so possible values are $0,\ldots,7$) and suppose the attacker knows `h` is even (so the possible values are $0, 2, 4, 6$). The uncertainty on `h` before executing `l = h` is hence $H(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) = 2$. It follows that the leakage is not 3 bits but

$$H(l = h) = H\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) = 2,$$

i.e. the information of `h`. The *security* of the program is the difference between the uncertainty before execution and the leakage (the uncertainty after execution). Hence security of the previous example of $l = h$ is $2 - 2 = 0$. Notice that when the program reveal everything this notion is invariant w.r.t. the chosen distribution, i.e. while the leakage of $l = h$ will depend on the distribution, its security will always be 0, all that can be revealed is revealed.

Formally security is defined [3] as

$$\mathtt{Sec}(o) = H(h|l) - H(o|l) = H(h|l, o).$$

The last equality is proven as follows:

$$\begin{aligned} H(h|l, o) &= H(h, l, o) - H(l, o) = H(h, l) - H(l, o) \\ &= H(h, l) - H(l) - H(l, o) + H(l) = H(h|l) - H(o|l). \end{aligned}$$

Using arguments similar to the ones presented at the end of Section 2.3 most of the times we will consider the simplified version where there are no dependencies on the low input, i.e. $H(h) - H(o)$. In fact $H(h|l)$ can be reduced to $H(h)$ when the secret is independent of the public input.

Another notion we will use is the *leakage ratio*, i.e.

$$\frac{H(o|l)}{H(h|l)}$$

the amount leaked divided by the maximum amount leakable. This is a number in the interval $[0, 1]$ which measures the percentage of the secret leaked by the program, so the ratio has minimum 0 iff the leakage is 0 and maximum 1 iff all the secret is revealed by the program.

5.5. *Classification of looping channels*

The following classification combines the previous definitions with variations in the size of the secret. For example, a bounded loop is one where even if we were able to increase arbitrarily the size of the secret we would not be able to increase arbitrarily the amount leaked.

For the purposes of this investigation loops are classified as:

a. C-*bounded* if the leakage is upper bounded by a constant C.
b. *Bounded* if the leakage is C-bounded independently of the size (i.e., number of bits) of the secret. It is unbounded otherwise.
c. *Stationary* or *constant rate* if the rate is asymptotically constant in the size of the high input.
d. *Increasing* (resp. decreasing) if the rate is asymptotically increasing (resp decreasing) as a function of the size of the high input.
e. *Mixed* if the rate is not stationary, decreasing or increasing.

Clearly all loops are C-bounded by the size of the secret and by the channel capacity; the interesting thing is to determine better bounds. For example, if we are studying a loop where we know the input distribution has a specific property we may found better bounds than the size of the secret.

From a security analysis point of view the most interesting case is the one of unbounded covert channels, i.e. loops releasing all secret by indirect flows. Notice that a guard cannot leak more than 1 bit so the rate of a covert channel cannot exceed the number of guards in the command.

Notice also that the rate of leakage is loosely related to timing behaviour. In loops with decreasing rate if the size of the secret is doubled each iteration will (on average) reveal less information than each iteration with the original size. We will spell out the timing content of rates in some of the case studies.

6. Case studies

We will now use the previous definitions. The aim is to show that the definitions make sense and the derived classification of channels helps in deciding when a loop is a threat to the security of a program and when is not.

The programs studied are simple examples of common loops: linear, bounded and bitwise search, parity check, etc.

Most of the arguments will use a separation property of the definition of leakage: in fact that definition neatly separates the information flows in the guard and body of a loop, so if there is no leakage in the body (e.g. no high variable appears in the body of the loop) (5) becomes

$$\lim_{n\to\infty} H\left(\mu(e^{\langle 0\rangle}), \ldots, \mu(e^{\langle n\rangle}), 1 - \sum_{0\leqslant i\leqslant n} \mu(e^{\langle i\rangle})\right). \tag{8}$$

On the other side if there is no indirect flow from the guard (e.g. e does not contain any variable affected by high variables) then (5) becomes

$$\lim_{n\to\infty} \sum_{1\leqslant i\leqslant n} \mu(e^{\langle i\rangle}) H(M^i | e^{\langle i\rangle}). \tag{9}$$

ɔnstant C.
ly of the size (i.e., number of

ically constant in the size of

otically increasing (resp de-
t.
creasing.

ecret and by the channel ca-
nds. For example, if we are
s a specific property we may

teresting case is the one of
ret by indirect flows. Notice
covert channel cannot exceed

o timing behaviour. In loops
ch iteration will (on average)
ginal size. We will spell out

s to show that the definitions
elps in deciding when a loop

n loops: linear, bounded and

of the definition of leakage:
ows in the guard and body of
variable appears in the body

. (8)

uard (e.g. e does not contain
es

(9)

Table 1

Summary of analysis for loops; loop i is the loop presented in Section 6.i of the paper

	loop 1	loop 2	loop 3	loop 4	loop 4a	loop 5	loop 6	loop 7
Bound	∞	4.3923	1	16	∞	0	$\log(C)$	∞
Channel rate	$\downarrow$	$=$	$=$	$=$	$=$	$=$	$=$	$=$
Capacity	k	4.3923	1	16	k	0	$\log(C)$	$\frac{k}{2}$
Channel leakage ratio	1	$\leqslant\frac{4.3923}{k}$	$\leqslant\frac{1}{k}$	$\leqslant\frac{16}{k}$	1	0	$\leqslant\frac{\log(C)}{k}$	$\leqslant\frac{1}{2}$

Unless otherwise stated we are assuming uniform distribution for all input random variables (i.e., all input values are equally likely) and that the high input is a k-bit variable assuming possible values $0, \ldots, 2^k - 1$ (i.e., no negative numbers).

A summary of this section results is shown in Table 1[6].

6.1. *An unbounded covert channel with decreasing rate*

Consider

```
l=0;
while (!(l=h)) l=l+1;
```

Under uniform distribution max $\mathtt{W(e,M)}_n$ is achieved by

$$H\big(\mu(e^{\langle 0\rangle}), \ldots, \mu(e^{\langle 2^k-1\rangle})\big) + \sum_{0\leqslant i\leqslant 2^k-1} \mu(e^{\langle i\rangle})H(M^i|e^{\langle i\rangle}).$$

Notice that no high variable appears in the body, so there is no leakage in the body, i.e.

$$\sum_{0\leqslant i\leqslant 2^k-1} \mu(e^{\langle i\rangle})H(M^i|e^{\langle i\rangle}) = 0.$$

We hence only need to study

$$H\big(\mu(e^{\langle 0\rangle}), \ldots, \mu(e^{\langle 2^k-1\rangle})\big)$$

notice now that

$$e^{\langle i\rangle} = \begin{cases} 0 = h & \text{if } i = 0,\\ 0 \neq h, \ldots, i \neq h \wedge i+1 = h & \text{if } i > 0, \end{cases}$$

[6]In the Channel leakage ratio row in the table quantities greater than 1 should be ignored.

hence $\mu(e^{\langle i \rangle}) = \frac{1}{2^k}$. This means

$$H\big(\mu(e^{\langle 0 \rangle}), \dots, \mu(e^{\langle 2^k-1 \rangle})\big) = H\left(\frac{1}{2^k}, \dots, \frac{1}{2^k}\right) = \log(2^k) = k.$$

As expected all k-bit of a variable are leaked in this loop, for all possible k; however to reveal k bits 2^k iterations are required. We conclude that this is an unbounded covert channel with decreasing rate $\frac{k}{2^k}$. To attach a concrete timing meaning to this rate let t_1, t_2 be the time (in milliseconds) taken by the system to evaluate the expression $!(\mathtt{l} = \mathtt{h})$ and to execute the command $\mathtt{l} = \mathtt{l} + 1$ respectively. Then the above program leaks $\frac{k}{2^k}$ bits per $t_1 + t_2$ milliseconds.

Notice that uniform distribution maximizes leakage, i.e. it achieves channel capacity.

Consider, for example, the following input distribution for a 3-bit variable:

$$\mu(0) = \frac{7}{8}, \qquad \mu(1) = \mu(2) = \dots = \mu(7) = \frac{1}{56}.$$

In this case the attacker knows, before the run of the program, that 0 is much more likely than any other number to be the secret, so the amount of information revealed by running the program is below 3 bits (below capacity). In fact we have

$$H\left(\frac{7}{8}, \frac{1}{56}, \dots, \frac{1}{56}\right) = 0.8944838.$$

Notice however that whatever the distribution the security of this program is 0 and leakage ratio 1.

6.2. *A bounded covert channel with constant rate*

```
l = 20; while (h < l) {l = l − 1}
```

After executing the program $\mathtt{l}$ will be $\mathtt{20}$ if $\mathtt{h} \geqslant \mathtt{20}$ and will be $\mathtt{h}$ if $0 \leqslant \mathtt{h} < \mathtt{20}$, i.e. $\mathtt{h}$ will be revealed if it is in the interval $0, \dots, 19$.

The random variables of interest are:

$$M^n \equiv l = 20 - n.$$

The events associated to the guard are:

$$e^{\langle n \rangle} = \begin{cases} h < 20 - n \wedge h \geqslant 20 - (n+1) \equiv h = 20 - (n+1), & n > 0, \\ h \geqslant 20, & n = 0 \end{cases}$$

$$;\Big) = \log(2^k) = k.$$

:his loop, for all possible k; how-
onclude that this is an unbounded
. concrete timing meaning to this
:he system to evaluate the expres-
$+ 1$ respectively. Then the above

cage, i.e. it achieves channel ca-

bution for a 3-bit variable:

$$\frac{1}{56}.$$

the program, that 0 is much more
: amount of information revealed
city). In fact we have

security of this program is 0 and

`if h` $\geqslant$ `20` and will be `h if`
terval $0, \ldots, 19$.

$$h = 20 - (n+1), \quad n > 0, \qquad n = 0$$

and

$$\mu(e^{\langle n\rangle}) = \begin{cases} \frac{2^k-20}{2^k} & \text{if } n = 0,\\ \frac{1}{2^k} & \text{if } 0 < n \leqslant 20,\\ 0 & \text{if } n > 20. \end{cases}$$

Again since the body of the loop does not contain any high variable

$$\sum_{1 \leqslant i \leqslant n} \mu(e^{\langle i\rangle}) H(M^i | e^{\langle i\rangle}) = 0.$$

The leakage is hence given by

$$\begin{aligned} &H(\mu(e^{\langle 0\rangle}), \ldots, \mu(e^{\langle n\rangle}))\\ &\quad = H\left(\frac{2^k-20}{2^k}, \frac{1}{2^k}, \ldots, \frac{1}{2^k}, 0, \ldots, 0\right)\\ &\quad = -\frac{2^k-20}{2^k}\log\left(\frac{2^k-20}{2^k}\right) - 20\left(\frac{1}{2^k}\log\left(\frac{1}{2^k}\right)\right). \end{aligned}$$

This function is plotted in Fig. 2 for $k = 6, \ldots, 16$. The interesting thing in the graph is how it shows that for k around 6 bits the program is unsafe (more than 2.2 bits of leakage) whereas for k from 14 upwards the program is safe (around 0 bits of leakage).

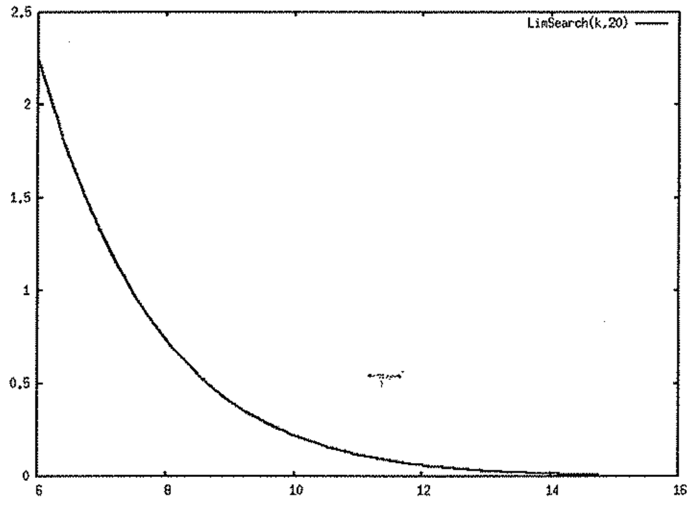


Fig. 2. Leakage in `l=20; while (h < l) {l=l-1}`.

Notice that the uniform distribution is not the channel distribution. The capacity of this channel is 4.3923 and is achieved by the distribution where the only values with non-zero probability for h are in the range $0, \ldots, 20$ and have uniform distribution[7].

Notice that the channel distribution ignores values of h higher than 20, so the channel rate is constant $\frac{4.3923}{21} = 0.2091$. We conclude that this is a bounded covert channel with decreasing rate.

6.3. *A 1-bounded channel with constant rate*

Consider the following program

```
h=BigFile;
i=0;
l=0;
while (i<N)
{
          l= Xor(h[i],l);
          i=i+1;
          }
```

This program take a large confidential file and performs a parity check, i.e. write in l the `Xor` of the first `N` bits of the file. The n-ary `Xor` function returns 1 if its argument has an odd number of 1s and 0 otherwise. This is a yes/no answer so its entropy has maximum 1 which is achieved by the uniform distribution. Hence

$$H(M^n|e^{\langle n\rangle}) = H(h[0] \oplus \cdots \oplus h[n-1]) = 1.$$

Notice that

$$e^{\langle n\rangle} \equiv n < N \wedge n+1 \geqslant N$$

henceforth

$$\mu(e^{\langle i\rangle}) = 0 \quad \text{if } i \neq N-1 \text{ and } \mu(e^{\langle N-1\rangle}) = 1.$$

We deduce the leakage is:

$$H(\mu(e^{\langle 0\rangle}), \ldots, \mu(e^{\langle n\rangle})) + \sum_{1 \leqslant i \leqslant n} \mu(e^{\langle i\rangle}) H(M^i|e^{\langle i\rangle})$$
$$= 0 + \mu(e^{\langle N\rangle}) H(M^N|e^{\langle N\rangle}) = 1.$$

[7] We are ignoring the case where $k < 5$ where the capacity is less than 4.3923.

This is a 1-bou
that if the number
inserting in the se
`size(h)` $=$ k the
decreasing rate $\frac{1}{k}$
Again there are
one where values
values.

6.4. *A 16-bounde*

Consider the pr

```
i
w
```

Here the guard
only need to use

```
int m
```

To compute
i.e. `high` $>=$ m
$\mathtt{m} = 2^{16-n}$) and
The variables

$M^n \equiv$ `lo`
$e^{\langle n\rangle} = 16$
$\mu(e^{\langle n\rangle}) =$

This is a 1-bounded channel with constant rate and capacity 1. Notice however that if the number of iterations were a function of the secret size, for example by inserting in the second line of the program the assignment N = size(h), (where size(h) = k the size of the secret) then it would become a 1 bounded channel with decreasing rate $\frac{1}{k}$ and capacity 1.

Again there are distributions which do not achieve channel capacity, for example one where values of h with odd number of bits equal to 1 are less likely than other values.

6.4. *A 16-bounded stationary channel*

Consider the program

```
int c = 16, low = 0;
while (c >= 0) {
    int m = (int)Math.pow(2,c);
    if (high >= m) {
        low = low + m;
        high = high - m;
    }
    c = c - 1;
}
System.out.println(low);
```

Here the guard of the loop does not contain variables affected by high, hence we only need to use Eq. (9) where M is

```
int m = (int)Math.pow(2,c);
        if (high >= m) {
            low = low + m;
            high = high - m;
        }
        c = c - 1;
```

To compute $H(M^n)$ notice that the nth iteration of M test the nth bit of high, i.e. high $>=$ m is true at the nth iteration iff the nth bit of high is 1 (this is because $\mathrm{m} = 2^{16-\mathrm{n}}$) and copies that bit into low.

The variables of interests are:

$$M^n \equiv \texttt{low} = n - \text{Bits}(\texttt{high}),$$

$$e^{\langle n \rangle} = 16 - n \geqslant 0 \wedge 16 - (n+1) < 0,$$

$$\mu(e^{\langle n \rangle}) = \begin{cases} 1 & \text{if } n = 16, \\ 0 & \text{otherwise.} \end{cases}$$

Because of this the leakage of the guard is 0 and for the total leakage we only need to compute $H(M^{16}|e^{<16>}) = 16$. This mean that the rate is 1.

This is hence an example of a 16-bounded stationary channel. However if we were to replace the first assignment `int c = 16` with `c = size(l)`, i.e.

```
int c = size(l), low = 0;
while (c >= 0) {
    int m = (int)Math.pow(2,c);
    if (high >= m) {
        low = low + m;
        high = high - m;
    }
    c = c - 1;
}
System.out.println(low);
```

then we would have an unbounded stationary channel (assuming that `h,l` be of the same size) with constant channel rate 1.

Again channel capacity is achieved by uniform distribution. For example, a distribution where we already know few bits of `high` will not achieve channel capacity.

6.5. *A never terminating loop*

```
while (0==0)
        low = high;
```

Here $\mu(e^{\langle i\rangle}) = 0$ for all i, hence for all n the formula

$$W(e, M)_n = H\left(\mu(e^{\langle 0\rangle}), \ldots, \mu(e^{\langle n\rangle}), 1 - \sum_{0\leqslant i\leqslant n} \mu(e^{\langle i\rangle})\right) + \sum_{1\leqslant i\leqslant n} \mu(e^{\langle i\rangle}) H(M^i|e^{\langle i\rangle})$$

becomes

$$H(0, \ldots, 0, 1) + \sum_{1\leqslant i\leqslant n} 0H(M^i|e^{\langle i\rangle}) = 0$$

from which we conclude that the leakage, rate and capacity are all 0.

The reason why the program is secure even if the whole secret is assigned to a low variable is that only observations on final states of the command are allowed (none in this case because of non termination); again this is feature of our model where the observer cannot see intermediate values of the computation, in which case this program would leak everything.

the total leakage we only need
rate is 1.
channel. However if we were
= size(l), i.e.

(2,c);

(assuming that h, l be of the

ibution. For example, a distri-
ıot achieve channel capacity.

a

$\mu(e^{\langle i \rangle}))$

ıcity are all 0.
ole secret is assigned to a low
command are allowed (none
feature of our model where
nputation, in which case this

6.6. A may terminating loop

```
l=0;
flag=tt;
       while (flag or l<h)
               {
               if (h<=C) flag=ff;
               l=l+1;
               }
```

This loop will terminate if $h \leqslant C$ and in that case $l = h$.
The event $e^{\langle i \rangle}$ corresponds to $i = h \wedge h \leqslant C$, hence

$$\mu(e^{\langle i \rangle}) = \frac{1}{C}\frac{C}{2^k} = \frac{1}{2^k} \quad \text{if } i \leqslant C.$$

Notice that as the information $h \leqslant C$ is known by knowing $e^{\langle i \rangle}$ we conclude that for all i, $H(M^i|e^{\langle i \rangle}) = 0$.
The leakage of this channel (under uniform distribution) is hence

$$H\left(\frac{1}{2^k}, \dots, \frac{1}{2^k}, \frac{2^k - C}{2^k}\right) = \frac{Ck}{2^k} - \frac{2^k - C}{2^k}\log\left(\frac{2^k - C}{2^k}\right).$$

This function is similar to the one from Section 6.2. Again channel capacity is achieved not by the uniform distribution but from the one where the first C values have probability $\frac{1}{C}$: in that case the program reveal all the secret.
Figure 3 shows the leakage for k between 10 and 20 and C between 400 and 500 under uniform distribution.

6.7. Probabilistic operators

When defining leakage in Section 2.3 it was shown that the conditional entropy $H(o|l)$ would overestimate leakage for a program like

```
l = random(0,1),
```

where random(0,1) a probabilistic operator returning 0 with probability p and 1 with probability $1 - p$.
However we could interpret l = random(0,1) as the program l = x where x is an "additional input" variable taking value 0 with probability p and 1 with probability $1 - p$. Then computing $H(o|l, x)$ gives $H(o|l, x) = H(o|x) = 0$, all uncertainty in the output comes from "the random" x so it can be eliminated by conditioning on it.

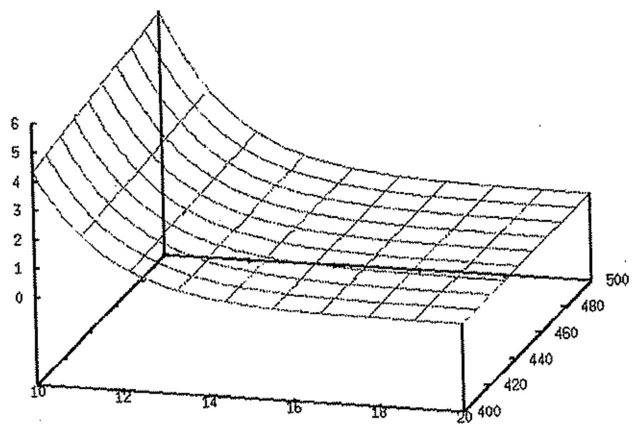


Fig. 3. Leakage for program in Section 6.6.

This suggests that an analysis of probabilistic programs can be developed by introducing a new random variable to cater for the probabilistic operator; the leakage formula becomes $H(o|l, x)$; the effect of this formula is to subtract from the uncertainty in the output the uncertainty coming from the low input and from the probabilistic operator; i.e. the uncertainty in $H(o|l, x)$ comes from the secret.

This approach works for a restricted class of probabilistic programs: the ones where the probabilistic choices are "observables"; it does not work for all probabilistic programs, because sometimes randomness is "injected" in a program so to confounding the attacker. In that case the general definition of leakage from Section 2.3 ought to be used.

As usual we can simplify the formula $H(o|l, x)$ to $H(o|x)$ by considering the low inputs to be initialized in the program as shown at the end of Section 2.3.

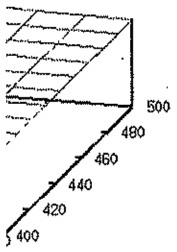
In the cases of loops using a probabilistic operator we take X as a stream of bits; the ith bit in the stream is the ith outcome of the operator.

We can compute the leakage of probabilistic programs by using the definition of conditional entropy

$$H(o|x) = \sum \mu(x = x_i)H(o|x = x_i).$$

As an example consider the program P

```
int i=0;  low = 0;
while (i<size(high)) {
    if (Coin[i]==0)
        low[i] = high[i];
        i=i+1;
}
System.out.println(low);
```



ı 6.6.

ns can be developed by intro-
ıtic operator; the leakage for-
subtract from the uncertainty
ıt and from the probabilistic
ıe secret.
abilistic programs: the ones
loes not work for all proba-
injected" in a program so to
nition of leakage from Sec-

$(o|x)$ by considering the low
nd of Section 2.3.
'e take X as a stream of bits;
or.
ns by using the definition of

where Coin is a stream of bits such that $\texttt{Coin[i]} = 0$ with probability p_i. Then at the end of the program the ith bit of high will be copied in low with probability p_i.

To compute the leakage of the program, i.e. $H(P|\texttt{Coin})$ we proceed as follow:

1. Compute, using Eq. (5), the entropies $H(P_{s_1}), \ldots, H(P_{s_n})$ where $H(P_{s_i})$ is the above program where the vector Coin is instantiated to a specific sequence s_i.
2. Compute $\sum \mu(s_i)H(P_{s_i}) = H(P|\texttt{Coin})$.

Given a stream s_i and high a k-bit variable, the bits of high copied in low are those corresponding to the positions in s_1 with value 0. For example, if high is a 4-bit variable and $s_i = 1001\ldots$ then low will be the sequence $0h[1]h[2]0$. The leakage of $H(P_{s_i})$ = number of 0s in s_i.

For example, if we assume high, Coin are uniformly distributed, i.e. any bit in high, Coin has $1/2$ chance of being 0 or 1 and high is a 4-bit variable then there will be 4 sequences with 1 zero, 6 with 2 zeros, 4 with 3 zeros and 1 with 4 zeros (the general formula is $\frac{k!}{(k-i)!i!}$ where i is the number of zeros). The leakage will hence be

$$\frac{4}{16} + \frac{6}{16}2 + \frac{4}{16}3 + \frac{1}{16}4 = \frac{1}{2} + \frac{3}{2} = 2$$

the general formula being

$$\sum_{1\leqslant i\leqslant k} \frac{k!}{(k-i)!i!} i \frac{1}{2^k} = \sum_{1\leqslant i\leqslant k} \frac{k!}{(k-i)!i!} i \frac{1}{2^i}\frac{1}{2^{k-i}} = \frac{k}{2}.$$

This is hence an unbounded channel leaking $\frac{k}{2}$ bits with rate $\frac{1}{2}$.

Notice that in the presence of probabilistic operators all definitions introduced, leakage, rate, channel, leakage ratio have an additional parameter, i.e. the distribution on this random choice input. The leakage for the above program given $\texttt{Coin[i]} = 0$ with probability p is pk. This is obtained by the expected value of the binomial distribution:

$$\sum_{1\leqslant i\leqslant k} \frac{k!}{(k-i)!i!} ip^i(1-p)^{k-i} = pk.$$

For example, by changing the distribution in Coin such that for all i, $\texttt{Coin[i]} = 0$ with probability 1 the above program become the unbounded stationary channel studied in Section 6.4 whereas if for all i, $\texttt{Coin[i]} = 0$ with probability 0 the above program become secure.

7. Conclusion and further work

The central point of this work has been to provide an Information Theoretical semantics of leakage in loops. The theory consists of several notions: absolute leakage, rate of leakage, channel capacity, and leakage ratio. We have given a classification of loops with the aim to determine which loop presents a security threat, and then presented several case studies in an attempt to show that the definitions and classification are useful in individuating security threats and are natural.

We believe that the ideas in this paper could provide a springboard for further applications of Information Theory in security and programming languages. Already this work has been used to quantify security of Multi-threaded programs [12].

Some directions for investigation are the following:

1. *Static Analysis*. This work could pave the way for more powerful static analyses based on Information Theory. As the case studies show the analysis requires some ingenuity, for example to determine which events the $e^{\langle i \rangle}$ represents. This reasoning usually involves the ability to detect interaction between several random variables. It may be possible that by combining techniques from theorem proving, model checking and quantitative static analysis like [4,5] some reasonable static analysis may be built. The central point, though, is that with a precise semantics of loops in place, we have a reference semantics that potential abstract domains should over-approximate, in which case loops could be soundly analyzed via fixed-point iteration.
2. *Timing Attacks*. As already noted, there is some information about timing in the notion of rate of leakage, rate being an indication of the average time needed to release some information; for example, a low rate suggests little amount of secret is released in each iteration, a decreasing rate indicates that the channel take longer to transmit information as the size of the secret increases. However many timing attacks are not covered in our current model, for example, those whose study requires intermediate states of execution to be *observable*; hence more work is required to address important issues in timing attacks.
3. *Concurrency, non determinism*. Integrating this work with a concurrency framework could open the way to the analysis of interesting protocols.
4. *Separation Logic*. O'Hearn, Reynolds and Isthiaq [14,24] have introduced a logic to reason about heaps based on some sort of non-interference between different parts of the code. Quantified interference may suggest a weaker separation logic which could be interesting to explore.

Acknowledgments

The author is very grateful to Fabrizio Smeraldi, Sebastian Hunt, Peter O'Hearn, Geoffrey Smith and the anonymous referees for very useful comments on this work. This research was supported by the EPSRC grant EP/C009967/1 Quantitative Information Flow.

References

[1] M. Boreale, Quan
Notes in Compute
[2] K. Chatzikokolak
Information and (
[3] D. Clark, S. Hunt
tronic Notes in Th
[4] D. Clark, S. Hunt
in Theoretical Co
[5] D. Clark, S. Hunt
Journal of Logic .
[6] M.R. Clarkson,
Computer Securi
45.
[7] T.M. Cover and J
[8] D. Bell and L. L
Technical Report
[9] D.E.R. Denning,
(1976), 236–243.
[10] D.E.R. Denning,
[11] J. Goguen and J.
and Privacy, IEE
[12] H. Chen and P. M
2007 ACM Work
2007, pp. 31–41.
[13] J.W. Gray III an
Distributed Com
[14] S. Isthiaq and P.
London, January
[15] G. Lowe, Quanti
of Critical Syste
[16] P. Malacaria, As
Principles of Pr
[17] P. Malacaria an
observational m
and Analysis for
[18] J.L. Massey, Gu
Trondheim, Nor
[19] J. McLean, Secu
Security and Pr
[20] J. Millen, Cove
Privacy, IEEE C
[21] A. Di Pierro, C.
Electronic Note.
eds, Elsevier, 20

Ve have given a classification
ıts a security threat, and then
ıat the definitions and classifi-
ıre natural.
·ide a springboard for further
gramming languages. Already
:hreaded programs [12].

ɔr more powerful static analy-
lies show the analysis requires
ɜvents the $e^{(i)}$ represents. This
teraction between several ran-
ning techniques from theorem
analysis like [4,5] some rea-
l point, though, is that with a
:ference semantics that poten-
in which case loops could be

ıformation about timing in the
ɔn of the average time needed
rate suggests little amount of
rate indicates that the channel
the secret increases. However
ɜnt model, for example, those
ution to be *observable*; hence
ːs in timing attacks.
is work with a concurrency
: interesting protocols.
aq [14,24] have introduced a
t of non-interference between
:e may suggest a weaker sepa-
ɜ.

ɜbastian Hunt, Peter O'Hearn,
ıseful comments on this work.
Ͻ009967/1 Quantitative Infor-

References

[1] M. Boreale, Quantifying information leakage in process calculi, in: *Proceedings of ICALP*, Lecture Notes in Computer Science, Vol. 4052, Springer, Berlin, 2006, pp. 119–131.

[2] K. Chatzikokolakis, C. Palamidessi and P. Panangaden, Anonymity protocols as noisy channels, *Information and Computation* **206** (2008), 378–401.

[3] D. Clark, S. Hunt and P. Malacaria, Quantitative analysis of the leakage of confidential data, *Electronic Notes in Theoretical Computer Science* **59** (2001).

[4] D. Clark, S. Hunt and P. Malacaria, Quantified interference for a while language, *Electronic Notes in Theoretical Computer Science* **112** (2005), 149–166.

[5] D. Clark, S. Hunt and P. Malacaria, Quantitative information flow, relations and polymorphic types, *Journal of Logic and Computation* **18**(2) (2005), 181–199.

[6] M.R. Clarkson, A.C. Myers and F.B. Schneider, Belief in information flow, in: *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW 18)*, IEEE Computer Society Press, 2005, pp. 31–45.

[7] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley-Interscience, 1991.

[8] D. Bell and L. LaPadula, Secure computer systems: Unified exposition and multics interpretation, Technical Report MTR-2997, MITRE Corporation, 1997.

[9] D.E.R. Denning, A lattice model of secure information flow, *Communications of the ACM* **19**(5) (1976), 236–243.

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

[11] J. Goguen and J. Meseguer, Security policies and security models, in: *IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, 1982, pp. 11–20.

[12] H. Chen and P. Malacaria, Quantitative analysis of leakage for multi-threaded programs, in: *Proc. 2007 ACM Workshop on Programming Languages and Analysis for Security*, San Diego, CA, June 2007, pp. 31–41.

[13] J.W. Gray III and P.F. Syverson, A logical approach to multilevel security of probabilistic systems, *Distributed Computing* **11**(2) (1998), 73–90.

[14] S. Isthiaq and P.W. O'Hearn, BI as an assertion language for mutable data structures, in: *28th POPL*, London, January 2001, pp. 14–26.

[15] G. Lowe, Quantifying information flow, in: *Proceedings of the Workshop on Automated Verification of Critical Systems*, 2001.

[16] P. Malacaria, Assessing security threat of looping constructs, in: *Proc. 34th ACM Symposium on Principles of Programming Languages*, Nice, France, January 2007, pp. 225–235.

[17] P. Malacaria and H. Chen, Lagrange multipliers and maximum information leakage in different observational models, in: *Proc. PLAS08: ACM SIGPLAN Workshop on Programming Languages and Analysis for Security*, Tucson, AZ, USA, June 2008, pp. 135–146.

[18] J.L. Massey, Guessing and entropy, in: *Proc. IEEE International Symposium on Information Theory*, Trondheim, Norway, 1994, p. 204.

[19] J. McLean, Security models and information flow, in: *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1990, pp. 180–187.

[20] J. Millen, Covert channel capacity, in: *Proc. 1987 IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press, 1987, pp. 60–66.

[21] A. Di Pierro, C. Hankin and H. Wiklicky, Probabilistic confinement in a declarative framework, in: *Electronic Notes in Theoretical Computer Science*, Vol. 48, A. Dovier, M.Ch. Meo and A. Omicini, eds, Elsevier, 2001, pp. 1–23.

[22] A. Di Pierro, C. Hankin and H. Wiklicky, Quantitative static analysis of distributed systems, *Journal of Functional Programming* **15**(5) (2005), 703–749.

[23] A. Rényi, On measures of information and entropy, in: *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability*, Berkley, CA, 1960, pp. 547–561.

[24] J. Reynolds, Separation logic: A logic for shared mutable data structures, 2002.

[25] J.C. Reynolds, Syntactic control of interference, in: *Conf. Record 5th ACM Symp. on Principles of Programming Languages*, Tucson, AZ, 1978, pp. 39–46.

[26] P.Y.A. Ryan, J. McLean, J. Millen and V. Gilgor, Non-interference, who needs it?, in: *Proceedings of the 14th IEEE Security Foundations Workshop*, Cape Breton, NS, Canada, June 2001, IEEE, 2001, p. 237.

[27] A. Sabelfeld and D. Sands, Dimensions and principles of declassification, in: *Proceedings of the 18th IEEE Computer Security Foundations Workshop*, Cambridge, England, 2005, Computer Society Press, IEEE, 2005, pp. 255–269.

[28] C. Shannon, A mathematical theory of communication, *The Bell System Technical Journal* **27** (1948), 379–423 and 623–656.

[29] G. Smith, On the foundations of quantitative information flow, Manuscript, July 2008.

[30] D. Volpano and G. Smith, A type-based approach to program security, in: *Proceedings of TAPSOFT '97 (Colloquium on Formal Approaches in Software Engineering)*, Lille, France, 1997, Lecture Notes in Computer Science, Vol. 1214, Springer, 1997, pp. 607–621.

[31] J.W. Gray, III, Toward a mathematical foundation for information flow security, in: *Proc. 1991 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1991, pp. 21–34.

[32] D.G. Weber, Quantitative hookup security for covert channel analysis, in: *Proceedings of the 1988 Workshop on the Foundations of Computer Security*, Fanconia, NH, USA, 1988, pp. 58–71.

[33] G. Winskel, *The Formal Semantics of Programming Languages: An Introduction*, MIT Press, Cambridge, MA, USA, 1993.

[34] T. Wittbold, Network of covert channels, in: *Proceedings of the 1990 Workshop on the Foundations of Computer Security*, 1990.

Journal of Computer
DOI 10.3233/JCS-2(
IOS Press

Detecting

Chiara Bodei
[a] *Dipartimento di*
E-mails: {chiara,
[b] *Dipartimento di*
E-mail: brodo@u
[c] *Informatics and*
Plads bldg 321, L
E-mail: hg@imm

A *type flaw att*
a field in a messa
extension of the
terms. We develo
possible behavio
during the protoc
occur. In the sam
by forcing some
type violations a
necessary to enfo
risk of having ty
to a number of s
complexity of th

Keywords: Secu

1. Introduct

At a high
value, such a
concrete leve
corresponden
be non-trivia
on the types
intruder that
message in p

*A prelimina
FET-GC II Inte
**Correspon