Reasoning About Distributed Secrets

Nicolás Bordenabe, Annabelle Mciver, Carroll Morgan, Tahiry Rabehaja

Reasoning about distributed secrets

N. Bordenabe[1], A. McIver[1], C. Morgan[2], and T. Rabehaja[1]

[1] Department of Computing, Macquarie University, Sydney
nicolas.bordenabe,annabelle.mciver,tahiry.rabehaja@mq.edu.au
[2] DATA61 & University of New South Wales, Sydney
carroll.morgan@unsw.edu.au

Abstract. In 1977 Tore Dalenius described how partial disclosure about one secret can impact the confidentiality of other correlated secrets, and indeed this phenomenon is well-known in privacy of databases. The aim here is to study this issue in a context of programs with distributed secrets. Moreover, we do not assume that secrets never change, in fact we investigate what happens when they do: we explore how updates to some (but not all) secrets can affect confidentiality elsewhere in the system. We provide methods to compute robust upper bounds on the impact of such information leakages with respect to all distributed secrets. Finally we illustrate our results on a defence against side channels.

Keywords: Quantitative information flow, foundations of security, program semantics, secure refinement.

1 Introduction

This paper concerns information flow when secrets are distributed amongst several agents. For example, let X, Y and Z represent three agents with respective secrets $\mathtt{x}, \mathtt{y}$ and $\mathtt{z}$, where $\mathtt{z} = \mathtt{f}(\mathtt{x}, \mathtt{y})$ for some function $\mathtt{f}$. In this basic scenario, each secret is correlated via a known function ($\mathtt{f}$), so that if something is leaked about one of the secrets, then something is also leaked about the others.

Partial disclosure about one secret leading to collateral disclosure about another is well documented in privacy of statistical databases, and was first addressed by Dalenius [9] who argued that ideal designs for privacy should prevent it: "Nothing about an individual should be learnable from the database that cannot be learned without access to the database". Later he argued the infeasibility of such a strict goal, and more recently Dwork [11] addressed the same concern demonstrating that whenever there is a (known) *correlation* between two pieces of information, anything learned about one piece implies that something might also be learned about the other.

In secure programming generally, i.e. not just read-only databases, this corresponds to leaking information about a secret "high-level" variable $\mathtt{x}$, which then consequentially leaks information about a different high-level variable $\mathtt{z}$ that *does not appear in the program at all*, but is known only via "auxiliary information" to be correlated with the initial value of $\mathtt{x}$ (as in $\mathtt{z} = \mathtt{f}(\mathtt{x}, \mathtt{y})$ mentioned above).

Because of the generality of this programming-language perspective, we call this effect *collateral leakage*.

Our approach is information theoretic where we model secrets as probability distributions $\mathbb{D}\mathcal{X}$ over some secret space $\mathcal{X}$, and we assume that programs are mechanisms which can both update secrets and leak some information about them. Within this setting we study the broader phenomenon of collateral leakage where secrets might be distributed between several non-colluding agents, and where those secrets could be correlated.

We extend our recent work by using Hidden Markov Models (*HMM*'s) in a new way to handle collateral leakage. In particular we study how to analyse the system-wide impact of information leaks caused by program execution on some, but not all of the secrets. These issues have been addressed partially in other work, but here we bring all these results together and take them further. In summary, we do the following:

- We review standard *HMM*'s and show how to view *HMM* matrices as mappings from correlations expressed as distributions in $\mathbb{D}\mathcal{X}^2$ to correlations $\mathbb{D}\mathcal{X}^2$, when the *HMM* is only able to update the second component in the product $\mathcal{X}^2$ (§3.2).
- We show how this unusual view of *HMM*'s –as mechanisms that update correlations– can be used to study the impact of information flow on correlated secrets, even when those secrets are not mentioned in particular program fragments (§4), and we provide methods to calculate exact and approximate leakages. Full proofs of these results can be found at [3].
- We illustrate some of our results on the analysis of a defence against side channels in cryptography (§5).

A particular novelty of this approach is "security refinement" which determines a partial order on *HMM*'s extending previous work [20] to *HMM*'s as correlation transformers. Refinement allows programs to be compared in a compositional manner: this does not seem possible with less general notions of leakage [21].

2 Motivation: correlated passwords

2.1 Changing a password: is it only "fresh", or actually "different"?

The example of Fig. 1 contrasts two users' approaches to updating their passwords; for simplicity each password is just one letter, chosen from $\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$. User Lax may update to any of the three, uniformly at random, including his current one; but User Strict must *change* his password, again uniformly, now choosing from only two of course. Because the original password `X` was uniformly distributed (which we assume for simplicity), in both cases the distribution of its final value is uniform as well; but an important difference, as we will see, is that for Lax the final `X` is independent of the initial `X`, while for Strict it is correlated.

In the second statement, at †, we confront (hostile) information flow: both users suffer an "over the shoulder" attack while logging in with their new password.

`// Password X is initially uniformly distributed over` $\mathcal{X} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$.

"Lax" user		"Strict" user	
`X:∈ [A,B,C]`	*	`X:∈ [X⁺,X⁻]`	$
`leak [X⁺,X⁻]`	†	`leak [X⁺,X⁻]`	†

* `[...]` *is a uniform distribution over* `{...}`*; and* `X:∈` *chooses* `X` *from it.*

$ `X`$^+$ *is the letter following* `X` *in* $\mathcal{X}$ *(wrapping around), and* `X`$^-$ *the preceding.*

† `leak [...]` *chooses a value from the distribution* `[...]` *and passes it to the adversary: she does not know however whether that value was* `X`$^+$ *or* `X`$^-$.

Fig. 1: Updating a password

We imagine that an observer hears a key-click and sees a key that is *not* being pressed: she thus learns one value that the new password definitely isn't.

Fig. 1 illustrates our concerns in a very simple way: the secret (password) is updated, and it is its *final* value the adversary wants to capture (indeed she will not be using its *old* value to hack this account). And yet –as we will see– third party agents can be affected.

One reason that in Fig. 1 it's natural to focus on the final state is that our aim (in program semantics) is to integrate security "correctness" with (ordinary) functional correctness of programs, i.e. to treat the two within the same framework [20]; and since functional correctness (and correctness comparisons, i.e. refinement [22]) is determined wrt. the final values a program produces, we should do the same for security correctness. Indeed it is in both cases the concentration on final values that allows small state-modifying programs, whether secure or not, to be sequentially composed to make larger ones [18–20].

Now the example above was constructed so that the two programs have the same final distribution and the adversary has the same knowledge of it — in both cases she knows exactly one value that the password is not. So are these programs equivalent in terms of their functional- and information-flow behaviour?

Here is where we encounter our criterion. As closed systems with a single secret, the password stored in variable `X`, Lax/Strict are indeed equivalent programs when the initial distribution is uniform. But they are *not* equivalent if we consider correlations between `X` and some *other* variable not mentioned in either, but present in a larger system with secrets distributed amongst other users. For example, suppose our young user selected his password `X` to be "the same as Dad's" that is stored in variable `Z`. And suppose Mum knows he did so.

So Dad says "You'd better change your password, son. Making it the same as someone else's is not safe." But as luck would have it Mum is in the bedroom, later, when Son changes it, and sees one of the two values that it has not become. In that case son Lax would leak no information about Dad's password; but with Strict, Dad's password is twice as likely ($1/2$) to be what Mum saw as it is to be one of the other two values ($1/4$ each). (See (8) below for details.)

Can this simple, almost fanciful example be dismissed? We don't think so: the facts are indisputable, that in the Strict case Mum learns something about

Dad's password but in the Lax case she does not: we return to this in §3.2; and we give a more elaborate example in §5. Have we invented this problem? No: it was recognised by Dalenius [9] and formalised by Dwork [11]. But (we believe) its impact has not been studied in respect of program refinement.

We stress the point that this phenomenon is truly remarkable if placed in the context of rigorous reasoning about programs generally. We have

	(`var X; Lax`) and (`var X; Strict`)	are the same over uniform initial `X` (1)
but	(`var Z; var X:=Z; Lax`)	are different over uniform initial `Z` (2)
and	(`var Z; var X:=Z; Strict`)	

What kind of familiar program algebra would invalidate an equality (1) because of variables added (2) that are not referred to by either program? The semantics of Lax and of Strict *must* differ in (1) as well.

In our extended *HMM* model described next, we show how Lax and of Strict are indeed modelled differently by keeping track of how programs change the *correlation* between initial and final program state. (In this case Son's initial and final passwords.)

3 *HMM*'s: generalising channels for secure refinement

3.1 Systems with distributed secrets

In a system of distributed secrets different secrets are handled by different system commands, possibly by different system components. Our aim here is to study the impact those commands have on all system secrets, whether or not they are part of any particular command. There are two reasons why this is important. The first is related to the issue raised by Dalenius, that a rigorous analysis of security must consider the prospect of the mechanism being executed in an environment where other secrets can be impacted. The second is related to compositionality in program semantics — the semantics must include enough detail so that conclusions drawn from local analysis of program fragments in isolation will be consistent with any emergent behaviours when those fragments are executed in larger contexts.

In our use of *HMM*'s detailed below we concentrate on showing how to use an analysis of an *HMM* in a compositional way — i.e. the analysis not only informs us about the leaks and updates to the variables described in a particular *HMM* matrix, but allows us as well to draw conclusions about leaks of other correlated variables if we consider the *HMM* to be executed as part of a larger system.

Review of the channel model for quantitative information flow Traditional models of information flow use "channels" to model flows in so-called "mechanisms", i.e. stochastic matrices which describe the relationship between secrets of type $\mathcal{X}$ and observations of type $\mathcal{Y}$. We recall first the standard notions of information flow in this setting, which we then extend to *HMM*'s.

Given channel matrix C, the entry C_{xy} is the probability that y is observed given that the secret is x. Channel matrices are *stochastic* meaning that for each x, $\sum_{y:\mathcal{Y}} C_{xy} = 1$. We write $\alpha \mathrel{\rightarrow\!\!\!\!\bullet} \beta$ for the type of stochastic matrices over $\alpha \times \beta$, thus C is of type $\mathcal{X} \mathrel{\rightarrow\!\!\!\!\bullet} \mathcal{Y}$. A fundamental assumption is that the secret x, once set, never changes and the measurements of information flow that the channel model supports thus involve comparisons between the attacker's prior knowledge of the secret, and how that changes to posterior knowledge when observations $\mathcal{Y}$ are taken into account. The prior knowledge is captured by a probability distribution $\pi : \mathbb{D}\mathcal{X}$ which assigns a probability π_x to each possible value $x : \mathcal{X}$; posterior distributions emerge when π is combined with C and are calculated using Bayesian reasoning. For observation y and prior π, the posterior probability that the secret is x given observation y is $C_{xy} \times \pi_x / (\sum_{x' : \mathcal{X}} C_{x'y} \times \pi_{x'})$. The *vulnerability* of a secret wrt. leaks can be assessed by measuring the extent to which the attacker can use the information leaked.

Notions of vulnerability of secrets and leakage of channels Vulnerability is a generalisation of entropy (of distributions), no longer necessarily e.g. Shannon but now others more adapted for secure programming, and whose great variety allows fine-grained control of the significance of the information that might be leaked [2].

Given a state-space $\mathcal{X}$, vulnerability is induced by a *gain function* over that space, typically g of type $\mathbb{G}_{\mathcal{W}}\mathcal{X} = \mathcal{W} \rightarrow \mathcal{X} \rightarrow \mathbb{R}$, for some space of *actions* $w : \mathcal{W}$. When $\mathcal{W}$ is obvious from context, or unimportant, we will omit it and write just $g : \mathbb{G}\mathcal{X}$. Given g and w (but not yet x) the function [3] $g.w$ is of type $\mathcal{X} \rightarrow \mathbb{R}$ and can thus be regarded as a random variable on $\mathcal{X}$. The range of $\mathcal{W}$ models a set of *actions* available to the attacker and the value $g.w.x$ represents his gain if he picks w and the secret's value turns out to be x. His optimal average gain, or *g-vulnerability* is then $V_g[\pi] = \max_{w \in \mathcal{W}} \sum_{x:\mathcal{X}} g.w.x \times \pi_x$.

A particularly simple example is $\mathcal{W}{=}\mathcal{X}$ with $g.w.x = (1 \text{ if } w{=}x \text{ else } 0)$ so that the adversary gains 1 if he guesses correctly and 0 otherwise: we call this particular gain-function g_{id}. A benefit of the more general $\mathcal{W}$'s is that they allow representation of many conventional entropy functions (including even Shannon), thus bringing them all within the same framework [21]. Given a g, prior π and a channel C we can model the *expected conditional g-vulnerability* as the maximal gain wrt. the channel C, or the average of the vulnerabilities of the posteriors:

$$V_g[\pi, C] := \sum_{y:\mathcal{Y}} \max_{w:\mathcal{W}} C_{xy} \times \pi_x \times g.w.x \ . \tag{3}$$

Inspired by mutual information, we can define more general notions of leakage by comparing the g-vulnerability of a prior with the expected conditional vulnerability. The *multiplicative g-leakage* of C wrt prior π and gain function g

[3] We write dot for function application, left associative, so that function g applied to argument w is $g.w$ and then $g.w.x$ is $(g.w)$ applied to x, that is using the Currying technique of functional programming.

is the ratio between the posterior and prior g-vulnerabilities:

$$\mathcal{L}_g(\pi, C) := \quad \log_2 V_g[\pi, C]/V_g[\pi] \quad . \tag{4}$$

The *capacity* of a channel is the supremum of that leakage (4), but varying in its definition depending on whether the supremum is over either gain functions, or priors or both:

$$\mathcal{L}_\forall(\pi, C) := \sup_g \mathcal{L}_g(\pi, C) \ , \ \mathcal{L}_g(\forall, C) := \sup_\pi \mathcal{L}_g(\pi, C) \ , \ \mathcal{L}_\forall(\forall, C) := \sup_{\pi,g} \mathcal{L}_g(\pi, C) \ .$$

Remarkably, it can be shown that $\mathcal{L}_\forall(\forall, C)$ equals $\mathcal{L}_{g_{id}}(\forall, C)$ ("min-capacity"): it is the most robust estimation of leakage, and can always be achieved for the uniform prior [1], making it straightforward to calculate. Moreover $\mathcal{L}_{g_{id}}(\forall, C)$ (also called $\mathcal{ML}(C)$) provides an upper bound for information leakage computed from the traditional Shannon entropy. Thus if $\mathcal{ML}(C)$ is no more than k then this means that no more than k bits are leaked by C for any prior. Capacities are useful because if they can be computed for a given channel C we are able to argue that the channel's leaks are insignificant if its calculated capacity is small.

3.2 *HMM*'s leak information about secrets *and* update them

The original model for *HMM*'s describes a two stage process acting on a secret in $\mathbb{D}\mathcal{X}$: first information about the secret is leaked, and then the value of the secret is updated. The first stage is equivalent to the action of a channel as described above, and the second stage is effected by a Markov transition. A Markov transition is also described by a matrix M (say) so that $M_{xx'}$ is the probability that initial state x will result in final state x'. An *HMM-step* then comprises a channel and a transition together, but acting independently on the initial state: we call C its channel and M its *markov* (lower case), and write $(C{:}M)$ of type $\mathcal{X} \rightarrowtriangle \mathcal{Y}{\times}\mathcal{X}$. Defined $(C{:}M)_{xyx'} = C_{xy}{\times}M_{xx'}$, it is a stochastic matrix with rows $\mathcal{X}$ and columns $\mathcal{Y}{\times}\mathcal{X}$. In this way *HMM*'s generalise both Markov processes and channels, and can therefore model a program that both leaks secrets and updates them. We shall, in particular, study how to use *HMM*'s when there are distributed secrets.

We begin by generalising *HMM* steps in order to model behaviour of programs consisting of multiple leak and assignment steps. We define sequential composition for *HMM*'s which summarises the overall information flow: in fact sequential composition is a natural operator if we are using them to model programs. Let $H^{1,2}$ of the same type $\mathcal{X} \rightarrowtriangle \mathcal{Y}{\times}\mathcal{X}$ be *HMM*'s.The composed type is the $\mathcal{X} \rightarrowtriangle \mathcal{Y}^2{\times}\mathcal{X}$ that takes initial state x to final state x' via some intermediate state x'', leaking information (y^1, y^2) –as it goes– gradually into the set $\mathcal{Y}^2$. We define

$$(H^1; H^2)_{x(y^1y^2)x'} \quad = \quad \sum_{x''} H^1_{xy^1x''}{\times}H^2_{x''y^2x'} \ , \tag{5}$$

and note that it is again stochastic. Sequential compositions are strictly more general than the *HMM*-steps built directly from $(C{:}M)$ — that is, for some

arbitrary composition (5) it does not necessarily correspond to a single step $(C{:}M)$ for some C and M.By keeping track of sequences of observations compositions of *HMM*-steps allow observers to accumulate multiple leaks over time and to draw conclusions based on their amalgamated knowledge.

Returning now to our example at Fig. 1 we illustrate how a program can be modelled as an *HMM*, by composing individual *HMM*steps. Recall the program snippet $\mathtt{X}{:\in}\ [\mathtt{X}^+,\mathtt{X}^-]$; leak $[\mathtt{X}^+,\mathtt{X}^-]$ where first $\mathtt{X}$ is updated and then something is leaked. The first statement $\mathtt{X}{:\in}\ [\mathtt{X}^+,\mathtt{X}^-]$ corresponds this Markov matrix $\mathcal{X}\!\to\!\triangleright\mathcal{X}$:

$$M^{S1}: \qquad \begin{array}{c} \\ \mathtt{A} \\ \mathtt{B} \\ \mathtt{C} \end{array} \begin{array}{c} \begin{array}{ccc} \mathtt{A} & \mathtt{B} & \mathtt{C} \end{array} \\ \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{pmatrix} \end{array}$$

For Strict the output for each initial state is a uniform choice over anything *but* the input. As an *HMM* we write it $(I^c{:}M^{S1})$, where I^c is the identity channel that leaks nothing.

The second statement leak $[\mathtt{X}^+,\mathtt{X}^-]$ corresponds to a channel matrix in $\mathcal{X}\!\to\!\triangleright\mathcal{Y}$, where in fact $\mathcal{Y}{=}\mathcal{X}$ because the observables are of the same type as the state: [4]

$$C^2: \qquad \begin{array}{c} \\ \mathtt{A} \\ \mathtt{B} \\ \mathtt{C} \end{array} \begin{array}{c} \begin{array}{ccc} \circ\mathtt{A} & \circ\mathtt{B} & \circ\mathtt{C} \end{array} \\ \begin{pmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{pmatrix} \end{array}$$

This leaks uniformly any value not equal to the current state, and $\circ\mathtt{A}, \circ\mathtt{B}, \circ\mathtt{C}$ denote the observations. We write it as $(C^2{:}I^m)$, where I^m is the identity Markov process that leaves all states unchanged.

For Strict the result is this *HMM* using (5) to form the composition $(I^c{:}M^{S1});(C^2{:}I^m)$, we can write it as a single matrix $\mathcal{X}\!\to\!\triangleright\mathcal{Y}{\times}\mathcal{X}$:

$$\begin{array}{c} \\ \mathtt{A} \\ \mathtt{B} \\ \mathtt{C} \end{array} \begin{array}{c} \overbrace{\begin{array}{ccc} \mathtt{A} & \mathtt{B} & \mathtt{C} \end{array}}^{\circ\mathtt{A}} \quad \overbrace{\begin{array}{ccc} \mathtt{A} & \mathtt{B} & \mathtt{C} \end{array}}^{\circ\,\mathtt{B}} \quad \overbrace{\begin{array}{ccc} \mathtt{A} & \mathtt{B} & \mathtt{C} \end{array}}^{\circ\,\mathtt{C}} \\ \begin{pmatrix} 0 & 1/4 & 1/4 & 0 & 0 & 1/4 & 0 & 1/4 & 0 \\ 0 & 0 & 1/4 & 1/4 & 0 & 1/4 & 1/4 & 0 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 0 & 1/4 & 1/4 & 0 \end{pmatrix} \end{array}$$

The labels $\circ\mathtt{A}, \circ\mathtt{B}, \circ\mathtt{C}$ denote the observations corresponding to those from C^2. Notice that the rows are *not* identical, because the first *HMM*-step updates the state in a way dependent on its incoming value.

Observe that there is a great deal of information concerning the current and former values of $\mathtt{X}$: for example, if the secret was originally uniformly distributed over the three values, and if $\circ\mathtt{A}$ is observed, then the probability that the initial state was $\mathtt{A}$ but is now B is $1/4$. Preserving this information about initial and final correlations in the semantics is precisely how we can analyse the effect that leaks about $\mathtt{X}$ has on other variables (such as $\mathtt{Z}$ in Fig. 1). We therefore consider *HMM*'s to be transformers not of individual secrets, but rather of secret correlations in $\mathbb{D}\mathcal{X}^2$. When an *HMM* transforms a correlation, the first component remains unchanged, but the second is updated, as before, according to the *HMM*. Thus

[4] Although the matrices C^2 and M^{S1} look the same, they are describing different aspects of the system.

given a correlation between X of type $\mathcal{X}$ and some other secret X' (of the same type) the *HMM* now produces a joint distribution $\mathcal{X}\times\mathcal{Y}\times\mathcal{X}$ which describes the correlation between the (unchanged) X', the observations, *and* the updated X.

Definition 1. *Given an HMM H of type $\mathcal{X}\!\!\rightarrowtriangle\!\mathcal{Y}\times\mathcal{X}$, and a distribution $\Pi \in \mathbb{D}\mathcal{X}^2$, we write $\Pi\rangle H\colon \mathbb{D}\mathcal{X}\times\mathcal{Y}\times\mathcal{X}$ for the joint distribution defined:*

$$(\Pi\rangle H)_{x_0yx} \quad := \quad \sum_{x'\colon \mathcal{X}} \Pi_{x_0x'}\times H_{x'yx} \ . \tag{6}$$

The probability that y is observed is $p_y := \sum_{x_0,x}(\Pi\rangle H)_{x_0yx}$. Given that y is observed, the (posterior) probability that the correlation is now (x_0, x) is defined by: $(\Pi\rangle H)_{x_0yx}/p_y$.

When Π^* is the correlation $\Pi^*_{(x_0,x)} = 1/3$ if and only if $x_0 = x$, then $(\Pi^*\rangle\mathsf{Strict})$ allows us to compute the chance that the correlation between initial and final values, given that A is observed: in that case the chance that initial state was A and the final is now B is $1/4$, but the chance that the final state is the same as the initial is 0.

Next, we extend the definition of refinement of abstract *HMM*'s in closed systems [18, 20] to take correlations into account.

Definition 2. *Let $H^1\colon \mathcal{X} \rightarrowtriangle \mathcal{Y}^1\times\mathcal{X}$ and $H^2\colon \mathcal{X} \rightarrowtriangle \mathcal{Y}^2\times\mathcal{X}$ be HMM's over base type $\mathcal{X}$ with observation types $\mathcal{Y}^1, \mathcal{Y}^2$ respectively. We say that $H^1 \sqsubseteq H^2$ if and only if there is a refinement matrix $R\colon \mathcal{Y}^1 \rightarrowtriangle \mathcal{Y}^2$ such that $H^1 \cdot R = H^2$, where $H^1 \cdot R := \sum_{y\colon \mathcal{Y}^1} H^1_{xy}\times R_y$.* [5]

A special case of a refinement matrix R is given by $y \mapsto y^*$ for fixed observation y^* — this removes all information flow in H so that $H \cdot R$ is maximal in the refinement order and therefore only records state updates. We write $\mathsf{mkv.H}$ for this maximal refinement of H. [6] [7]

Def. 2 has an equivalent formulation in terms of gain functions: $H^1 \sqsubseteq H^2$ means that the gain for an attacker of H^1 will always be at least as high as a gain for an attacker observing H^2, because he can use the extra observations to improve his strategy. Note however that the attacker's gains are related to guessing the *correlation*. Given an *HMM* H and gain function g, we define

$$V_g[\Pi, H] \quad := \quad \sum_{y\colon \mathcal{Y}} \max_{w\colon \mathcal{W}} \sum_{x,x',x''} \Pi_{xx'}\times H_{x'yx''}\times g.w.(x, x'') \ . \tag{7}$$

Theorem 1. *Let H^1, H^2 be HMM's. We have $H^1\sqsubseteq H^2$ if and only if $V_g[\Pi, H^1] \geq V_g[\Pi, H^2]$, for all $g\colon \mathcal{W}\rightarrow\mathcal{X}^2\rightarrow\mathbb{R}$, and $\Pi\colon \mathbb{D}\mathcal{X}^2$.*

[5] We have overloaded matrix multiplication, to mean that the summation is always over the shared state in $M^1 \cdot M^2$.

[6] Notice that the exact value y^* is not important for refinement comparisons.

[7] Def. 2 defines a pre-order on *HMM*'s, but it can be made into a partial order on "abstract *HMM*'s", introduced elsewhere [21].

If we use an initial correlation $\Pi_{(x,x')} = 1/3$ if and only if $x = x'$, we see that $\mathsf{Lax} \not\sqsubseteq \mathsf{Strict}$ since $V_{gcid}[\Pi, \mathsf{Lax}] = 1/6$ whereas $V_{gcid}[\Pi, \mathsf{Strict}] = 1/4$. Here *gcid* corresponds to *gid*, but where $\mathcal{W} = \mathcal{X}^2$. Similarly $\mathsf{Strict} \not\sqsubseteq \mathsf{Lax}$ since $V_{[AA]}[\Pi, \mathsf{Lax}] = 1/9$ but $V_{[AA]}[\Pi, \mathsf{Strict}] = 0$, where $[AA]$ is the gain function which gives 1 only for secret (correlation) (A, A), and 0 for everything else.

Crucially, refinement is compositional for sequential composition.

Lemma 1. *Let $H^1 \sqsubseteq H^2$, then $H^1; H \sqsubseteq H^2; H$ and $H; H^1 \sqsubseteq H; H^2$ for any HMM $H: \mathcal{X} \nrightarrow \mathcal{Y} \times \mathcal{X}$.*

4 Reasoning about distributed, correlated secrets

We return now to a system of distributed secrets described by $\mathcal{X}$ and $\mathcal{Z}$, and we study how to model information flow about $\mathcal{Z}$ when only $\mathcal{X}$ is updated by a program fragment. Given an *HMM* $H : \mathcal{X} \nrightarrow \mathcal{Y} \times \mathcal{X}$ representing such a fragment, we can describe the effect that H has on some initial correlation $\Pi : \mathbb{D}(\mathcal{X} \times \mathcal{Z})$ between $\mathcal{X}$ and $\mathcal{Z}$ by computing the joint distribution $J: \mathbb{D}(\mathcal{Z} \times \mathcal{Y} \times \mathcal{X})$: $J_{zyx} := \sum_{x': \mathcal{X}} \Pi_{zx'} \times H_{x'yx}$. Moreover Thm. 1 implies that if $H \sqsubseteq H'$ then $V_g[\Pi, H^1] \geq V_g[\Pi, H^2]$, for $g: \mathcal{W} \times \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$.

We show next that robust upper bounds for leakage $\mathcal{Z}$ follows from leakage about the initial state of $\mathcal{X}$.

Define $\overleftarrow{\Pi}$ to be the $\mathcal{Z}$-marginal of Π, i.e. $\overleftarrow{\Pi} = \sum_{x: \mathcal{X}} \Pi_{xz}$ and let matrix $\overrightarrow{\Pi}$ in $\mathcal{Z} \nrightarrow \mathcal{X}$ be given by $\overrightarrow{\Pi}_{zx} = \Pi_{xz} / \overleftarrow{\Pi}_z$ if $\overleftarrow{\Pi}_z > 0$ and 0 otherwise. This factors Π into its marginal and a conditional, and indeed $\Pi_{xz} = \overleftarrow{\Pi}_z \times \overrightarrow{\Pi}_{zx}$. Now the matrix multiplication $\overrightarrow{\Pi} \cdot H$ gives an *HMM* of type $\mathcal{Z} \nrightarrow \mathcal{Y} \times \mathcal{X}$. Since we are interested in the leakage about $\mathcal{Z}$ only we can define a channel on $\mathcal{Z}$ alone by forgetting the final value of $\mathcal{X}$. This gives us the "effective collateral channel".

Definition 3. *The* effective collateral channel *of H, written $\mathsf{chn}.H$, is a stochastic matrix of type $\mathcal{X} \nrightarrow \mathcal{Y}$ and defined simply by ignoring the final state: thus $(\mathsf{chn}.H)_{xy} := \sum_{x'} H_{xyx'}$.*

Definition 4. *The collateral leakage resp. capacity of H wrt a prior $\Pi: \mathbb{D}(\mathcal{Z} \times \mathcal{X})$ is the collateral leakage resp. capacity of $\overrightarrow{\Pi} \cdot \mathsf{chn}.H$.*

Refinement of collateral channels and their corresponding *HMM*'s is consistent.

Lemma 2. *If $H^1 \sqsubseteq H^2$ then $V_g[\pi, \mathsf{chn}.H^1] \geq V_g[\pi, \mathsf{chn}.H^2]$ for $g: \mathcal{W} \times \mathcal{X} \to \mathbb{R}$ and $\pi: \mathbb{D}\mathcal{X}$.*

Proof. Define $\Pi^*_{xx'} := \pi_x$ if and only if $x = x'$. Observe now that $V_g[\pi, \mathsf{chn}.H^1] = V_{g^*}[\Pi^*, H]$, where $g^*.w.x.x' = g.w.x.x$. The result now follows by Thm. 1.

Using Def. 3 on the *HMM* $(I^c : M^{S1}); (C^2 : I^m)$ described above, we can calculate the effective collateral channel for the program Strict. It describes the information

leak about the initial state of Son's password only.

$$\mathsf{chn.Strict}: \quad \begin{array}{c} \\ \mathtt{A} \\ \mathtt{B} \\ \mathtt{C} \end{array} \begin{array}{c} \circ\mathtt{A} \ \circ\mathtt{B} \ \circ\mathtt{C} \\ \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} \end{array} \tag{8}$$

We see now clearly that Mum's best guess for Son's initial setting of his password is to guess the value she observes; she has now learned something about Dad's *current* password.

The simplicity of Defns. 3,4 conceals that it can be difficult in practice to calculate the collateral leakage of an *HMM*. One reason is that a model for initial state only is *not compositional* i.e. $\mathsf{chn.}(H^1;H^2)$ cannot be calculated from just $\mathsf{chn.}H^1$ and $\mathsf{chn.}H^2$ alone. This implies that if H is expressed as a sequential composition of many smaller ones, e.g. if we have $H = H^1;H^2;\cdots;H^N$, still the final states of the *intermediate* H's must be retained, not only to form the composition, but because the overall y observation from H comprises all the smaller observations $y^1\cdots y^N$ with each y^{n+1} being determined by the final state $(x')^n$ of the H^n just before — we can abstract only at the very end.

In the special case however where each H is an *HMM*-step $(C^n{:}M^n)$, the calculation of the effective channel can be somewhat decomposed.

Lemma 3. *Let H be an HMM and $(C{:}M)$ an HMM-step. Then*

$$\begin{array}{ll} \mathsf{chn.}\,(C{:}M) & = C \\ \mathsf{chn.}\,((C{:}M);H) & = C \parallel (M{\cdot}\mathsf{chn.}H) \end{array}$$

where in general $(C^1\|C^2)_{x,(y^1y^2)} = C^1_{xy^1}\times C^2_{xy^2}$ is parallel composition of channels. The M cannot be discarded, since it affects the prior of the "tail" H of the sequential composition.

We see in the example above that we do not have to construct the full *HMM*, and then reduce it as we did at (8) but instead just perform the matrix multiplication of its components, so that $\mathsf{chn.Strict} = M^{S1}{\cdot}C^2$. Even with Lem. 3, in general $\mathsf{chn.}H$ can be challenging to compute because its size (given by the number of columns in the stochastic matrix representation) grows exponentially with the number of single-step *HMM*'s, in the definition of H, that have non-trivial channel portions. We give an example of such a calculation in §5 (fast exponentiation for cryptography).

On the other hand, if we want to compute only the collateral *capacity*, we can obtain at least an upper bound at considerably less cost, without the need to compute $\mathsf{chn.}H$ exactly. The following provides an upper bound for $\mathcal{L}_\forall(\forall, \mathsf{chn.}H)$, and requires only linear resources.

Lemma 4. *For any H let $\mathsf{CCap.}H$ be defined*

$$\begin{array}{lll} \mathsf{CCap.}\,(C{:}M) & = \mathcal{L}_\forall(\forall, C) & \text{if } H{=}(C{:}M) \\ \mathsf{CCap.}\,((C{:}M);H') & = \mathcal{L}_\forall(\forall, C) + \min(\mathcal{L}_\forall(\forall, M), \mathsf{CCap.}H') & \text{if } H{=}(C{:}M);H' \end{array}$$

Then $\mathcal{L}_\forall(\forall, \mathsf{chn.}H) \le \mathsf{CCap.}H$ with the stochastic matrix M treated as a channel.

In fact Lem. 4 provides a very robust estimate of the collateral capacity of an *HMM*, since it does not mention $\mathcal{Z}$ or the correlating Π. And it is the best possible general bound, achieving equality for some examples, e.g. Fig. 1: CCap for Strict is $\log(3/2)$, and for Lax it is $\log 1 = 0$, confirming that Strict leaks some information about correlated secrets whereas Lax leaks nothing. Both these values are equal to the calculated leakages in their given scenarios. It is also easy to calculate since for any channel we have from [1] that $\mathcal{L}_\forall(\forall, C) = \mathcal{L}_{g_{id}}(\Upsilon_\mathcal{X}, C)$, where $\Upsilon_\mathcal{X}$ is the uniform prior on $\mathcal{X}$.

Lemma 5. *Let $H, H' : \mathcal{X} \twoheadrightarrow \mathcal{Y}\times\mathcal{X}$ be HMMs. $\mathsf{CCap}.H{=}0 \Rightarrow \mathsf{CCap}.(H'; H){=}\mathsf{CCap}.H'$.*

Proof. From Thm. 4 the unfolding of the recursive definition for $\mathsf{CCap}.(H'; H)$ will eventually yield a minimum between non-negative terms which include $\mathsf{CCap}.H$. The result now follows, because this allows a simplification to $\mathsf{CCap}.H'$.

In cases where we know the correlation Π (and thus $\mathcal{Z}$), we can compute the collateral capacity by identifying the optimising gain function in Def. 4.

Theorem 2. *Given H and $\Pi : \mathbb{D}(\mathcal{Z}\times\mathcal{X})$ with $\overleftarrow{\Pi}, \overrightarrow{\Pi}$ resp. the marginals of Π on $\mathcal{Z}, \mathcal{X}$; define conditional $\overrightarrow{\overleftarrow{\Pi}}$ as above. There exists $\hat{g} : \mathbb{G}_\mathcal{Z}\mathcal{Z}$ and $\hat{g}^\Pi : \mathbb{G}_\mathcal{Z}\mathcal{X}$ such that*

$$\mathcal{L}_\forall(\overleftarrow{\Pi}, \overrightarrow{\overleftarrow{\Pi}}{\cdot}\mathsf{chn}.H) \quad = \quad \mathcal{L}_{\hat{g}}(\overleftarrow{\Pi}, \overrightarrow{\overleftarrow{\Pi}}{\cdot}\mathsf{chn}.H) \quad = \quad \mathcal{L}_{\hat{g}^\Pi}(\overrightarrow{\Pi}, \mathsf{chn}.H)\ .$$

This shows that it is possible to construct the gain-function that maximizes the collateral capacity, and even allows its exact calculation. Moreover, it also shows that the collateral capacity of H wrt. $\mathcal{Z}$ can be understood as regular g-leakage of H wrt. the *initial* state of $\mathcal{X}$.

The next theorem is more general, and gives an upper bound over all possible correlations: it is determined by the extremal leakage of the initial prior $\pi : \mathbb{D}\mathcal{X}$, thus easy to calculate [1].

Theorem 3. *Given H and Π as above, $\Upsilon_\mathcal{X}$ is uniform on $\mathcal{X}$, then*

$$\mathcal{L}_\forall(\overleftarrow{\pi}, \overrightarrow{\overleftarrow{\Pi}}{\cdot}\mathsf{chn}.H) \quad \leq \quad \mathcal{L}_{g_{id}}(\Upsilon_\mathcal{X}, \mathsf{chn}.H)\ , \quad \textit{where } \overleftarrow{\pi}, \overrightarrow{\overleftarrow{\Pi}} \textit{ are as defined in Thm. 2.}$$

Note that when $\mathtt{X}$,$\mathtt{Z}$ are completely correlated, i.e. when $\overleftarrow{\overrightarrow{\Pi}}$ and $\overrightarrow{\overleftarrow{\Pi}}$ are both the identity, the inequality in Thm. 3 becomes equality. Finally we note that the separation of information flow from state updates sometimes does simplify sequences of *HMM* steps, when either the $\mathsf{chn}.H$ or $\mathsf{mkv}.\mathsf{H}$ contains no probabilistic updates, then the channel can be approximated by $(\mathsf{chn}.H : \mathsf{mkv}.\mathsf{H})$. We provide details at [3].

5 Case study: side channel analysis

Keys for public-key cryptography are best if independent, but that is not to say that they necessarily are: recently [16] discovered an unexpected sharing of the

prime numbers used to generate them. Although that discovery concerned public keys (and hence also the private keys), it makes the point that we cannot assume not-yet-discovered correlations do not exist between private keys alone. That motivates our example here, the collateral leakage from a fast-exponentiation algorithm that might compromise *someone else's* private key.

```
// B for base, the cleartext; E for exponent, the key: precondition is B,E >= 0,0 .
// P for power, the ciphertext.
P:= 1
while E!=0  // Invariant is P*(B^E) = b^e, where b,e are initial values of B,E .
   D:∈ [2,3,5]               // D for divisor; uniform choice from {2,3,5}.
   R:= E mod D;              // R for remainder.
   if R!=0 then P:= P*B^R fi // Side-channel: is E divisible exactly by D ?
   B:= B^D                   // D is small: assume no side-channel here.
   E:= E div D               // State update of E here. (No side-channel.)
end
// Now P=b^e and E=0: but what has an adversary learned about the initial e ?
```

Although our state comprises B,E,P,D,R we concentrate only on the secrecy of E. In particular, we are not trying to discover B or P in this case; and D,R are of no external significance afterwards anyway.

Fig. 2: Defence against side channel analysis in exponentiation, Exp(B,E)

Fig. 2 implements fast exponentiation, with a random choice of divisor to defend against a side channel that leaks program flow (of a conditional) [24]. Since the program code is public, that leak is effectively of whether divisor D exactly divides the current value of E, which value is steadily decreased by the update at the end of each iteration: thus additional information is leaked every time. In the standard (and fastest) algorithm the divisor D is always 2, but that ultimately leaks E's initial value exactly, one bit (literally) on each iteration. The final value of E is always zero, of no significance; but its initial value represents collateral leakage about subsequent use of this same key (obviously), but also the use of other apparently unrelated keys elsewhere [16]. The obfuscating defence is to vary the choice of D unpredictably from one iteration to the next, choosing it secretly from some set $\mathcal{D}$, here $\{2,3,5\}$ although other options are possible. The divisor D itself is not leaked.

Since the output of Exp(B, E) of Fig. 2 is a function of its inputs, and as mentioned above its behaviour can be summarised by a single-step *HMM*, of the form (chn.Exp(B, E):"Output $\mathtt{B}^{\mathtt{E}}$"). Thus our task is to compute chn.Exp(B, E). We modelled the loop as a sequential composition of *HMM*-steps for a fixed number of iterations and used Lem. 3 to construct a channel that captures the leakage of information about the initial state of the program. We assumed that all variables are secret, and that every iteration leaks some information at the if statement "if R!= 0", revealing whether the (hidden) R at that point is 0 or not, and

therefore possibly information about the other variables too. Interesting however, is that although this leak appears to be standard, the obfuscation provided by the choice of R means that overall the calculation of chn.Exp(B, E) shows that the information leak is highly probabilistic, the more randomness provided by $\mathcal{D}$. We provide detailed calculations in [3] of the construction of the *HMM*'s. Although our calculation is wrt. the uniform prior on E, and even though we do not know the extent of any correlation between this key E and others used elsewhere, by using the multiplicative capacity [1, Sec VI.C], we can bound the maximum leakage about the initial value of E with the min-capacity of such a channel. Furthermore, by relying on Thm. 3 we can see that this min-capacity can also be used as a bound on the collateral leakage *with respect to any other secret that might be correlated to* E. For example, if E has 8 bits then a maximum of 3.51 of those bits of E's initial state will be leaked; moreover no more than of 3.51 of bits of *any other secret* Z in the system, that could be correlated to E will be leaked. This is therefore a very robust upper bound on the impact of system-wide leakage.

Size of E	$\mathcal{D}$={2}	$\mathcal{D}$={2, 3}	$\mathcal{D}$={2, 3, 5}
4 bits	4	2.80	2.22
5 bits	5	3.32	2.61
6 bits	6	3.83	2.92
7 bits	7	4.34	3.21
8 bits	8	4.88	3.51

Note that in the case $\mathcal{D}$={2} the whole secret E is leaked. As explained at end §??, that $\mathcal{L}_{g_{id}}$ gives the upper bound $\mathcal{L}_{\forall}$ for all vulnerabilities.

Fig. 3: Collateral leakage in bits wrt E for different $\mathcal{D}$'s, for *Prog* given at Fig. 2.

Our table Fig. 3 confirms that the larger the divisor set $\mathcal{D}$, the less effective is the side channel; and the protection is increased with more bits for E.

6 Related work, conclusions and prospects

In this paper we have studied information leakage in systems where several secrets are distributed across a system. We have demonstrated how to use an *HMM* model to analyse *collateral leakage* in programs where some, but not all, secrets can change. We have shown that when correlations are present across the system, the impact of collateral leakages can still be predicted by local reasoning on program statements that process particular subsets of secrets. Our model represents the first step towards a general method for analysing leakages in distributed contexts.

Our work extends classical analyses of quantitative information flow which assume that secrets do not change. Early approaches to measuring leakage are based on determining a "change in uncertainty" of some "prior" value of the secret — although how to measure the uncertainty differs in each approach. For

example Clark et al [6] use Shannon entropy to estimate the number of bits being leaked; and Clarkson et al [8] model a change in belief. The role of capacity when the prior is not known was stressed by Chatzikokolakis et al [5]. Smith [23] demonstrated the importance of using measures which have some operational significance, and this idea was developed further by introducing the notion of g-leakage to express such significance in a very general way. The partial order used here on *HMM*'s is the same as the g-leakage order explored in by Alvim et al [1], but it appeared also in even earlier work [18]. Unlike information flows that eg only use Shannon entropy, our $\sqsubseteq$ based on gain functions respects composition of programs, making it suitable for equality reasoning in algebras [20].

More recently Marzdiel et al [17] analysed information flow of dynamic secrets, using a model based on probabilistic automata. This reflects a view that in general systems secrets are not necessarily static. In other work [20] we have explored a model based on the analysis of final states only, but it cannot be used to explore general correlations with fresh variables, as we do here.

Clark et al [7] give techniques for static analysis of quantitative information flow based on Shannon entropy for a small while-language. Extended *HMM*'s for modelling side channels have been explored by Karlof and Wagner [14] and Green et al [13] for e.g. key recovery. We note that in §5 our quantitative capacity bounds on side channels are valid even for collateral leakage, when the program is executed as a procedure call. In §5 we observe that there is a relationship between reducing the impact of a side channel and the performance of the algorithm. Others [10] have explored this trade-off between confidentiality and performance using a game theory setting.

Bordenabe and Smith [4] explore collateral leakage in the context of a static secrets, and our work can be seen as a generalisation of their approach when secrets can be updated. Kawamoto et al. [15] also study gain function leakage for complex systems made from components. Their focus is different to ours in that they consider how to decompose a channel in order to compute the leakage more easily; they do not deal with the general problem of collateral leakage.

References

1. Mário S. Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. Additive and multiplicative notions of leakage, and their capacities. In *CSF*, pages 308–322. IEEE, 2014.
2. Mário S. Alvim, Andre Scedrov, and Fred B. Schneider. When not all bits are equal: Worth-based information flow. In *Proc. 3rd Conference on Principles of Security and Trust (POST 2014)*, pages 120–139, 2014.
3. N. Bordenabe, A. McIver, C Morgan, and T. Rabehaja. Compositional security and collateral leakage, 2016. arXiv:1604.04983.
4. Nicolás E. Bordenabe and Geoffrey Smith. Correlated secrets in quantitative information flow. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 93–104, 2016.
5. Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Inf. Comput.*, 206(2-4):378–401, 2008.

6. D. Clark, S. Hunt, and P. Malacaria. Quantitative analysis of the leakage of confidential data. *Electr. Notes Theor. Comp. Sci.*, 59(3):238–251, 2001.
7. David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantified interference for a while language. *Electr. Notes Theor. Comput. Sci.*, 112:149–166, 2005.
8. Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Belief in information flow. In *18th IEEE Computer Security Foundations Workshop, (CSFW-18 2005), 20-22 June 2005, Aix-en-Provence, France*, pages 31–45, 2005.
9. T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15:429–44, 1977.
10. Goran Doychev and Boris Köpf. Rational protection against timing attacks. In *IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, pages 526–536, 2015.
11. Cynthia Dwork. Differential privacy. In *Proc. 33rd International Colloquium on Automata, Languages, and Programming (ICALP 2006)*, pages 1–12, 2006.
12. Barbara Espinoza and Geoffrey Smith. Min-entropy as a resource. *Information and Computation*, 226:57–75, April 2013.
13. P. J. Green, Richard Noad, and Nigel P. Smart. Further Hidden Markov model cryptanalysis. In *Cryptographic Hardware and Embedded Systems - CHES 2005*, pages 61–74, 2005.
14. Chris Karlof and David Wagner. Hidden Markov Model cryptanalysis. In *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, pages 17–34, 2003.
15. Yusuke Kawamoto, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Compositionality Results for Quantitative Information Flow. In QEST 2014, volume 8657 of *LNCS*, pages 368–383. Springer, 2014.
16. Arjen K. Lenstra, James P. Hughes, Maxime Augier, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, Whit is right. Technical report, EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland, 2012.
17. Piotr Mardziel, Mário S. Alvim, Michael W. Hicks, and Michael R. Clarkson. Quantifying information flow for dynamic secrets. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 540–555, 2014.
18. Annabelle McIver, Larissa Meinicke, and Carroll Morgan. Compositional closure for Bayes Risk in probabilistic noninterference. In *ICALP'10*, LNCS volume 6199, pages 223–235, Berlin, Heidelberg, 2010. Springer.
19. Annabelle McIver, Larissa Meinicke, and Carroll Morgan. Hidden-Markov program algebra with iteration. *Mathematical Structures in Computer Science*, 2014.
20. A. McIver, C. Morgan, and T. Rabehaja. Abstract Hidden Markov Models: a monadic account of quantitative information flow. In *Proc. LiCS 2015*, 2015.
21. Annabelle McIver, Carroll Morgan, Geoffrey Smith, Barbara Espinoza, and Larissa Meinicke. Abstract channels and their robust information-leakage ordering. In volume 8414 of *Lecture Notes in Computer Science*, pages 83–102. Springer, 2014.
22. C.C. Morgan. *Programming from Specifications*. Prentice-Hall, second edition, 1994. `web.comlab.ox.ac.uk/oucl/publications/books/PfS/`.
23. Geoffrey Smith. On the foundations of quantitative information flow. In volume 5504 of *Lecture Notes in Computer Science*, pages 288–302, 2009.
24. Colin D. Walter. MIST: an efficient, randomized exponentiation algorithm for resisting power analysis. In *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, pages 53–66, 2002.