UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Tiago Melo Tannus

Towards the Quantification of Information Leakage for Dynamic Secrets

Belo Horizonte
2022

Tiago Melo Tannus

Towards the Quantification of Information Leakage for Dynamic Secrets

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Mário Sérgio Alvim

Belo Horizonte
2022

Tiago Melo Tannus

Explorando a Quantificação de Vazamento de Informação sobre Segredos Dinâmicos

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Mário Sérgio Alvim

Belo Horizonte
2022

Tannus, Tiago Melo.

T167t Towards the quantification of information leakage for dynamic secrets [manuscrito] / Tiago Melo Tannus. — 2022.
95 f. il.

Orientador: Mário Sérgio Alvim.
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação
Referências: f. 90-92.

1. Computação – Teses. 2. Fluxo de informação – Teses. 3. Criptografia de dados (Computação) – Teses. 4. Segurança da informação – Teses. 5. Métodos formais – Teses. I. Alvim, Mário Sérgio. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*46 (043)

Ficha Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa CRB 6/1510 Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

TOWARDS THE QUANTIFICATION OF INFORMATION LEAKAGE FOR DYNAMIC SECRETS

TIAGO MELO TANNUS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Mario Sérgio Ferreira Alvim Júnior - Orientador
Departamento de Ciência da Computação - UFMG

Profa. Elaine Gouvêa Pimentel
Departamento de Matemática - UFRN

Prof. Jeroen Antonius Maria van de Graaf
Departamento de Ciência da Computação - UFMG

Prof. Gabriel de Morais Coutinho
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 29 de março de 2022.

Documento assinado eletronicamente por **Mario Sergio Ferreira Alvim Junior**, **Professor do Magistério Superior**, em 31/03/2022, às 16:03, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Elaine Gouvêa Pimentel**, **Usuária Externa**, em 04/04/2022, às 10:21, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Gabriel de Morais Coutinho**, **Professor do Magistério Superior**, em 05/04/2022, às 16:42, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Jeroen Antonius Maria Van de Graaf**, **Professor do Magistério Superior**, em 07/04/2022, às 15:20, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1350826** e o código CRC **7286E366**.

Para vovó Edir.

Acknowledgments

First of all, I would like to thank my mother, Paula, and father, Sérgio. They love me immensely, and made many sacrifices to make this a reality. Without their continuous support throughout all the hardships none of this would be possible.

Second, but no less important, I would like to thank Professor Mário Alvim. It was a pleasure to work with him. The greatest teacher I've ever met in a classroom, always trying to improve. Maybe the most hard working person I know. And also one of the most comprehensive.

I would also like to thank my family for all the emotional support they gave me. Family is the most important thing in life, and I was lucky to have a great family that always supported my choices. I would like to thank my sister, Sara, my girlfriend, Isabelle, my uncle, Pablo, and all the other family members that make me a better person everyday.

I believe that friends are the second most important thing in life. Friends are a different kind of family. I will cite here as many as I can remember. I will definitely forget some of you, but you all have a place in my heart. Mateus, for opening many doors for me; Plinio, for being an schmuck; Benhur, for giving great life advice; Salim, for helping me move; Soneca, for saying that he saw Nelson Piquet winning an F1 championship before he was born; Richard, for turning my head to the left; Avx, for turning my head to the right; Pinarel, for playing quality DotA with me; Brasil, for being crazy; Iza, for controlling Brasil; Lotti, for giving great musical advice; Bolha and Pedro, for giving terrible opinions about soccer; Thiti, a great partner for everyhing; Ramon, another great partner for everything; Henrique, a third great partner for everything; Lucas Renan, a great partner for a walk to McDonalds; Artur, for talking about games; Loucas, for talking about people; João Falcão Moreno, for showing me new technologies and his uncle's movie; Pedro Bustamante, for great anime and manga suggestions; Aquino, for hunting pokemons; Guru, for talking about Galo; Gabriel Henrique Lopes Gomes Alves Nunes, for helping controlling the environment; Joseph, for the late night talks about life; Bruno, for using female deodorant; Rod Quantico, for the defense of the indians; Igor, for questioning things; Bruno Dematttos, for being just a great person; and everyone else that I forgot to mention here.

I also want to thank the Computer Science Department which was an integral part of my life during many years. I had the pleasure to work with great people at CRC - Heitor, Pi, Jean, Manassés, Armstrong, Polly, Clovis, Murilo, and Alison - who were part of the journey. They also gave me a deeper contact with the people from the department

which I learned to admire and respect. Great people like Sônia, Gilmara, David, Ricardo, and many others. I want to thank Professor Gabriel Coutinho, Elaine Pimentel and Jeroen van de Graaf for taking the time to read my thesis and point out important topics so I could improve this work.

I would also like to thank all the teachers I had throughout my academic career, for they taught me many lessons about life. The classroom is one of the most enriching environments in life, if you have the right mindset to enjoy it.

I would also like to thank CAPES for giving me financial support throughout the master's course.

Finally, I would like to thank all brazilian citizens, who paid for my academic education. Brazil is a blessed country with great people that deserve nothing but the best.

"*Fire shall teach you.*"
(Ember Spirit, Dota 2 Hero)

Resumo

O framework de fluxo de informação quantitativo (QIF) apresenta uma modelagem matemática para vários tipos de sistemas com os quais lidamos no dia a dia. Essa modelagem é feita através de matrizes estocásticas, distribuições de probabilidade, e outros modelos matemáticos suplementares. Além disso, esse framekwork é capaz de modelar informações sensíveis que interagem com este sistema. Através desta interação, pode haver vazamento de informação. Um ponto que QIF também é capaz de capturar. Com isso é possível mensurar a vulnerabilidade relacionada a este vazamento.

A literatura original de QIF propõe que o conhecimento à priori de um adversário sobre informações sensíveis pode ser modelado como uma simples distribuição de probabilidade. Recentemente, um novo trabalho expandiu essa ideia para que esse tipo de conhecimento pudesse ser modelado por distrubuições sobre distribuições de probabilidade, conhecidas como hiper-distribuições. Esse trabalho desmembrou o conceito tradicional de vulnerabilidade em dois tipos: vulnerabilidade do segredo, que está associada intrinsicamente ao segredo, e vulnerabilidade de estratégia, que relaciona a vulnerabilidade das diferentes estratégias presentes. Entretanto, nada foi desenvolvido para como o conhecimento a priori interage com um canal. Também não há calculos para vulnerabilidade após o processamento. Consequentemente, não há noção formulada para vazamento de informação.

Essa dissertação apresenta duas maneiras de realizar a interação de uma hiper-distribuição com o canal, além de apresentar uma maneira restrita de cálculos de vulnerabilidade após a interação do conhecimento com o sistema. Isso nos permite medir o vazamento de informação, dada a interação.

Palavras-chave: Fluxo de Informação Quantitativo, Segredos Dinâmicos, Vazamento de Informação, Fundamentos de Segurança da Computação, Métodos Formais.

Abstract

The framework of Quantitative Information Flow (QIF) presents a mathematical modeling for many types of systems which we deal with on a daily basis. This is done through stochastic matrices, probability distributions, and supplementary mathematical apparatus. It also models sensitive information and how that changes after interacting with a system. Through this interaction, there may be information leakage. This is something that QIF is also able to capture. Furthermore, it is possible to measure how vulnerable a piece of information is when it relates to the system.

The original QIF literature proposes that the adversary's prior knowledge about sensitive information can be modelled as a simple probability distribution on secret values. Recently, a new work expanded this idea so that this type of knowledge could be modelled as distributions on distributions on secret values, known as hyper-distributions. This work dismembered the traditional concept of vulnerability into two different types: secret vulnerability, which focuses on the vulnerability of the secret itself, and strategy vulnerability, which focuses on the vulnerability associated with the different strategies that can generate a secret. However, nothing has been said about how this knowledge interacts with a channel. It was also left open the issue of measuring any type of vulnerabilities after the processing. Consequently, there is no notion of leakage of information when hyper distributions represent prior knowledge.

Our work presents two models for the interaction between the prior information modelled as a hyper-distribution and the channel. We also present a way to measure vulnerability after the interaction between knowledge and system. This allows us to measure how much information is accessed by the adversary given this interaction.

Keywords: Quantitative Information Flow, Dynamic Secrets, Information Leakage, Vulnerability Measures, Foundations of Computer Security, Formal Methods.

List of Figures

Contents

Chapter 1

Introduction

We live in a world where technology is advancing quickly. Computational systems are everywhere we can possibly imagine, ranging from the most trivial tasks from day-to-day life to the most complex computational systems. There are many properties that are relevant like correctness, efficiency, transparency, privacy, and many others. Each of these properties has both practical and theoretical relevance. And we would like examine the specifics of one of them: protection of sensitive information. One of its most relevant aspects is privacy.

One may ask how privacy relates to computational systems and what is the relevance of treating such problem. Both of these questions may seem to have straightforward answers, and yet they are extremely important.

We can associate the concept of privacy to computer science. Let us think of a system in a more abstract way. If we represent a system as a "box" where possibly sensitive information is received as an input and then processed, later the system outputs a result. With this way of depicting things, we may want to keep this sensitive information private. A few systems that value privacy are elections, password checkers, bank apps, and others. From this we can argue that privacy in the context of a computational system has the aim to minimize sensitive information leakage after it is processed by a system. Speaking in more general terms, being able to differentiate when information is made private or public [Moore, 2008].

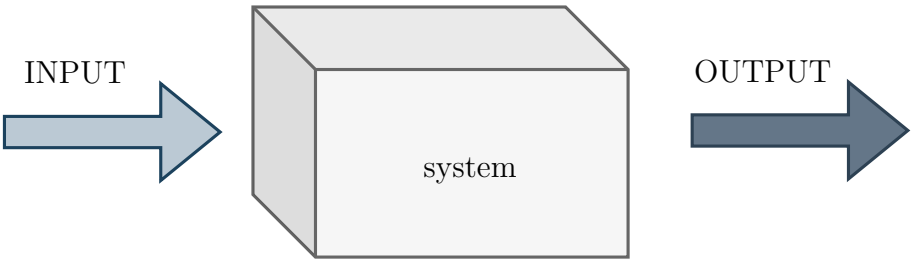

Figure 1.1: Abstraction of a system.

We can interpret Figure 1.1 as if information flows through a system. The challenge related to it is keeping this information private. Arguably it is impossible to not reveal anything sensitive. Take the classical example of a password checker for instance. When

someone types a password on a login screen, on the general case it will necessarily grant access to the account if the password is correct, or deny it if the password is incorrect. Then it is true that some information can be derived from each of the answers of the system. Even knowing that some information is being leaked, we still continue to use this kind of system because we deem it "safe enough" to do so. Therefore there the amount of information exposed is perceived intuitively as acceptably small.

From this acceptance, arises the need to use a mathematical framework that can encapsulate the concepts of the occurrence of information leakage and also quantify it. From this we can define an acceptable threshold for each situation. There is a theoretical framework that captures this general idea: The *Quantitative Information Flow* (QIF)[Alvim et al., 2020]. QIF is a domain of theoretical computer science that aims to study how much information leaks in computational systems, and what can be done to prevent those leaks. Further on this dissertation this framework will be presented with its formalisms. For now we obtain ourselves to the most important concepts.

First let us think about the *user* of the system. The user is the entity that utilizes the system to process its information. This is done by feeding the system a possibly sensitive input. Since we want to treat about privacy, we call this input a *secret*. This secret can be seen as a value that is hidden from all entities involved, other than the user himself. Another important entity is the *adversary*[1]. The adversary tries to discover what was the input given by the user - i.e. the secret. This can be seen as an "invasion of privacy". At first the adversary doesn't know what was the input, but he can observe how the system behaves after the secret is processed.

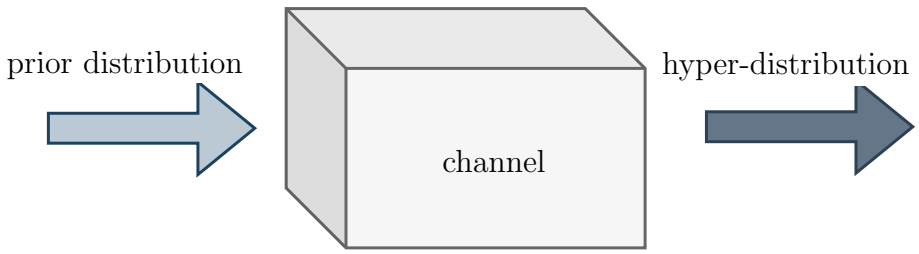

Figure 1.2: Representation of the system using QIF terminology.

The adversary also has a few other pieces of knowledge. She knows that the user has a finite set of secrets from which she can choose from. Her knowledge about how the secret is generated is modelled as simple probability distribution to pick a specific secret as input. This is distribution is called *prior distribution*. After selecting the secret, the user employs a *channel* to process it. The channel is a stochastic matrix that maps the input to an output given a distribution of probabilities. By combining her prior knowledge about the secret with full knowledge about how the system works, the adversary can use the system's outputs to update her knowledge about the secret. This can be modelled as a

[1]From here on, when we use the pronoun "she" we will be referring to the adversary.

posterior distribution that is a probability distribution that represents how likely it is for each of possible secret given a particular output. Finally by weighing the probability of each posterior distributions happening, we obtain a *hyper-distribution*, also called a hyper. A hyper is a probability distribution over probabilities distributions that symbolizes the adversary's knowledge after the system displays the output. The general idea can be visualized on Figure 1.2.

There are two very important concepts related to what was already described, the first is the *vulnerability* of the system, which gives us is how big is the threat of the secret being discovered given an instant. The vulnerability can be measured before the secret goes through the channel by analyzing the prior distribution, which we call prior vulnerability. It can also be measured after it goes through the channel by analyzing the hyper distribution, which is called posterior vulnerability.

The second is the concept of *leakage* that creates an association between the two kinds of vulnerability previously described. This association tells us how much of the vulnerability changed after the secret was processed by the channel. In other words it represents how much information the channel allowed to be learned by the adversary.

To better visualize this concept, we can think of a more concrete example. Imagine that a user is at an ATM machine. His password would represent the secret, the channel is the ATM machine that could accept or deny the input password. After the channels accepts or deny an input, it will necessarily leak some information. Either by accepting the input or rejecting it.

Given a prior vulnerability of a secret, it may seem intuitive that changing the distribution multiple times can improve the security we have about the secret. Which would decrease the vulnerability. Inspired by this idea, Mardziel et al. [2014a] tried to find what would actually happen if you changed the prior multiple times. And the result was counter-intuitive: changing your prior multiple times could actually increase the value of the vulnerability if your strategy for doing so was somehow predictable.

This novel finding motivated a new line of work on this subject. Alvim et al. [2017a] presented a new way of introducing priors: instead of simple distributions, priors were denoted by hypers, aiming to represent the different strategies an user could have to input the secret. This made that the concept of vulnerability had to be broken down into two different parts. *Strategy vulnerability* that addresses the values innate to each prior distribution, and *environmental vulnerability* that handles how each user of the system picks a different prior to make use of.

These later findings are the motivation for this project, Alvim et al. [2017a] modelled the priors as hypers and defined the concepts of prior vulnerability given this modeling. The authors did not expand the concepts for an update function, which would require them to define an operation to push a hyper through a channel. Consequently it was impossible to define concepts for posterior vulnerability and leakage. We want

to use this new concept to explore what happens with the other parts of the system. This different modeling can have different implications on channels, posteriors, posterior vulnerabilities, and, most importantly, leakages, that are one of the focal points of the framework.

1.1 Thesis objectives

The goals of this thesis are the following.

Main objective: Investigate how the recently proposed models of prior knowledge as hyper-distributions can be extended to distinguish leakage about secrets from leakage about strategies.

The QIF framework presents us with the idea that the adversary has some previous knowledge about the system, usually dubbed as prior distribution. This knowledge is then updated, which is usually done by a channel. This update generates a posterior adversarial knowledge, which is usually denoted by a hyper-distribution.

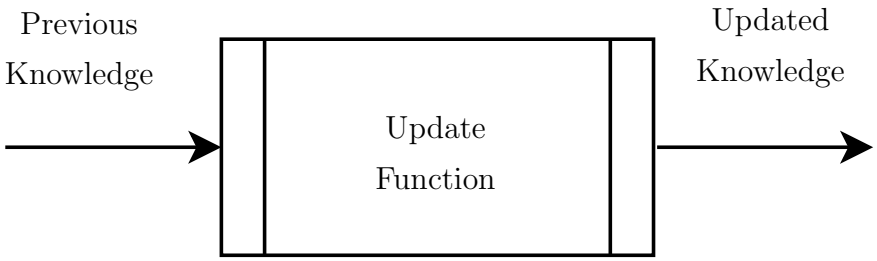

Figure 1.3: Representation of a system with the concept of knowledge.

Recent research on this subject [Alvim et al., 2017a][Alvim et al., 2020, chap. 13, and 14] tried to redefine the classical notion of a prior probability being a single distribution of probability over the secrets. While the former redefined the prior as a hyper, the latter kept the notion of a single distribution being the prior, but allowed channels to receive hypers as inputs without defining their outputs. From these facts we can divide the main objective into two specific objectives..

1.1.1 Specific objective 1: Define knowledge update

Define a new update function similar to the traditional QIF approach. This new update function must be able to receive a hyper as an input and push it through a channel. We also need to define what results from this operation.

1.1.2 Specific objective 2: Define posterior versions of vulnerability measures and of leakage for secret and strategy

Evolve the already established concepts of vulnerability of strategies and vulnerability of secrets, from Alvim et al. [2017a], for the posterior case. This means that we must be able to apply the concepts of vulnerability to the resulting object of the new type from the knowledge update, specified by 1.1.1.

1.2 Contributions

The main contributions of this thesis are the following:

- We define of an *update function* that broadens the default QIF approach, allowing hypers to be used as priors and produce meaningful results after the interaction with a channel. This step gives sequence to the work of Alvim et al. [2017a]. This work is presented on Chapter 3.
- We propose two new metrics for posterior vulnerability for secret and strategy. One that encapsulates the secret and another for the environment. We also define leakage for environments. It is important to note that these metrics are done with a restricted set of guesses. This also extends the previous work from Alvim et al. [2017a]. This work is presented on Chapter 5.
- We provide a revised notation to the work of Alvim et al. [2017a], which aims to make the understanding of the framework of environments clearer. This work is scattered throughout the thesis, but it is presented on Section 2.2 and used extensively on Chapter 5.

1.3 Related Work

Information is an inherent subject to computer science. Keeping information secret is also very important. This is due to the fact that our world has computational systems controlling everything around us. The reliability of these systems depends on who has access to it, thus a secret password can be used to restrict access. These secrets can be held by big organizations or single individuals. Zviran and Haga [1999] describe many characteristics of their use (e.g length, frequency of use, composition, lifetime, etc). These are important factors that may help the users create stronger passwords. Brown et al. [2004] present an interesting study of frequency of personal password usage and how much one's personal characteristics are present on the secret.

Passwords must be strong, that means they must not be easily discovered by third parties. Literature presents us with many ways to create them, for instance Glory et al. [2019] present a complex algorithm to generate strong and unique passwords that will not be easily cracked. The problem is that a complex password may not be recalled by the user with ease. Blocki et al. [2014] argue that spaced repetition associated with a mnemonic technique may enable users to create strong passwords that will also be easily recalled. Although Florêncio et al. [2007] argue that strong passwords are not really necessary to avoid brute-force attacks, they just have to be sufficiently long given that a "'three strikes" rule apply.

One could have the intuition that changing passwords frequently would make them stronger. Some studies, like Adams et al. [1997], seem to endorse this idea. Later on, Mardziel et al. [2014b] had an interesting finding through a theoretical work on the foundations of QIF. They used probabilistic automata to model program execution. This model takes inputs and, through a random function, produce outputs. They aim to capture dynamics of secrets, thus they use strategy functions to generate inputs. This model also presents different types of adversaries: the usual adversary that interacts with the channel on every occasion, and a "wait-adaptative adversary" that can observe the execution of the system and interact when she sees fit. They also proposed an information-theoretic metric for quantifying flow of dynamic secrets. This metric then was used on experiments with both types of adversaries. The metric also generalized the previous metric of leakage for static secrets. Through experimentation they concluded that more frequent change could lead to more leakage. However, that work didn't try to explain why this is possible, or characterize the phenomenon precisely. They only proposed that strategies play a role on this. We present more about this subject on Example 2.19.

More recently, Alvim et al. [2017a] published a study further expanding this idea. They tried to characterize the phenomenon, explicitly modeling strategies as part of the adversary's prior knowledge. This work could distinguish between two kinds of prior vulnerability: environmental prior vulnerability, which focused on quantifying the intrinsic uncertainty about how each strategy generates secrets, and strategy prior vulnerability, which models the uncertainty about the adversarial knowledge of the aggregation of multiple strategies. This will be presented by Defintions 2.21, 2.22, 2.24, and 2.27. This work did not define any type of posterior vulnerability or leakage, and this is the main objective of this thesis.

1.4 Thesis outline

We organize this thesis as follows. In Chapter 2, we present the background knowledge needed to read this work. Mainly introducing QIF concepts and a few other concepts related to it. In Chapter 3 we present two types of knowledge update for the specific

scenario where prior knowledge is represented as a hyper. Chapter 4 presents many attempts on defining posterior vulnerability measures for the case of dynamic secrets. Until we reach Chapter 5, where we present our final definition, for a restricted set of guesses. We later present, in the Appendix A, a few details of a program that calculates the posterior hypers, given a prior hyper and a channel, and also calculates all types of relevant vulnerabilities for this work.

We finally conclude this work on Chapter 6.

Chapter 2

Preliminaries and literature review

In this chapter, we present all essential results in literature for the understanding of this thesis. We introduce the general framework for QIF in Section 2.1, a few extensions of this framework in Section 2.2, and also a type of stochastic matrix named markov in Section 2.3.

2.1 The traditional QIF framework

The quantitative information flow (QIF) framework aims to measure information leakage in a system. This is a relevant problem in computational security with respect to privacy. To achieve the goal of privacy the framework models information and the systems that process it. The approach also shows how information flows when data is processed. The flow may leak sensitive information, which can be used by an adversary for ill intended purposes, which is what we want to avoid.

QIF derives from information theory. The latter is already well defined on literature. This research field is based in works as the one proposed by Shannon [1948].

QIF is a relatively new approach. Early works had many different notations. The fundamentals of this knowledge has been organized in a book, published by Alvim et al. [2020]. This thesis will adopt notations from this book. The symbol (:=) is used, throughout the thesis, for definitions. Additionally we use the pronoun "she" to refer to the adversary, which will be introduced on Section 2.1.1. We also use the symbol $\mathbb{D}$ as a representation of a probability distribution. That is, given a set $\mathcal{A}$, when we state $\mathbb{D}\mathcal{A}$, it means the set of infinitely many probability distributions over the set $\mathcal{A}$.

2.1.1 Secrets and basic notions

One of the major concepts is that of the *secret*, which is used to represent the sensitive information concerning a user. A secret can be used to represent things such as passwords, locations, etc. Even results such as a positive test of an infectious disease can be sensitive with regards to an adversary.

We consider two agents interacting with the secret in some way. The *user* is the one who knows the secret and wants to keep it protected. The *adversary* does not know

the secret value, and wants to infer it by using the clues she has. Before interacting with the system, the adversary knows the probability distribution over possible values the secret can assume. We call this a *prior distribution*, because it represents the adversarial knowledge before any interaction with the system. We consider a probability distribution δ on a finite set $\mathcal{X}$ a function from $\mathcal{X}$ to the interval $[0, 1]$, which we denote as $\delta : \mathcal{X} \mapsto [0, 1]$, such that $\sum_{x \in \mathcal{X}} \delta_x = 1$.

For instance if the secret is a 4-digit numerical password for a bank account, the prior can be represented as an uniform probability of $1/10000$ for each of the 10000 possible digit combinations. We define[1] secret and prior as follows.

Definition 2.1 (Secret and prior)**.** *A* secret *is some piece of information the user wants to protect from the adversary. We usually denote a secret by x, and assume it can take values from a non-empty, finite set $\mathcal{X}$. The knowledge some adversary has about the secret is represented by a probability distribution π on $\mathcal{X}$, called a* prior distribution*, that specifies the probability π_x of every value $x \in \mathcal{X}$.*

Another important concept is that of the support. The support of a distribution is the set of all elements with non-zero probabilities of a distribution. We define the support as follows.

Definition 2.2 (Support of a distribution)**.** *Let $\mathcal{A}$ be a nonempty set. Given a distribution $\pi \in \mathbb{D}\mathcal{A}$, the* support *of π is $\lceil \pi \rceil = \{a \in \mathcal{A} | \pi(a) > 0\}$.*

A point distribution is also a key concept. We define it as follows.

Definition 2.3 (Point distribution)**.** *Given a distribution $\pi \in \mathbb{D}\mathcal{A}$, we call π a point distribution, written $[a]$, when one of the elements of $a \in \mathcal{A}$ has an assigned probability of 1.*

The prior distribution is pivotal to measure how *vulnerable* a secret is before the adversary interacts with the system. It represents the knowledge the adversary has before the system is run. At first she can only take information from the prior. And thus the QIF framework presents us a way to measure the corresponding vulnerability. Taking inspiration from the literature we can use a few already established metrics like *guessing entropy* [Massey, 1994], and *Shannon entropy* [Shannon, 1948].

Guessing entropy measures the expected number of tries an adversary needs to guess the secret. It takes into account that the adversary always makes the optimal choice in a linear search (i.e. guessing the secret making one attempt at the time. Each

[1] Most definitions on Section 2.1 are strongly based from Alvim et al. [2020], with the adaptations necessary to be presented here.

of the attempts asks a question: "Is $x = x'$?", for some $x' \in \mathcal{X}$.). Using this metric, an optimal adversary always tries to guess the exact value of the secret from the most to least probable. Therefore it is an uncertainty measure for her (i.e., the higher its value, the less informative the prior is). Assuming that $\pi_{x_i}s$ are in a non-increasing order, guessing entropy is defined as

$$G(\pi) \quad = \quad \sum_{i=1}^{|\mathcal{X}|} i\pi(x_i).$$

Shannon entropy represents the maximum number of Boolean questions that would have to be asked, over a prior $\pi : \mathbb{D}\mathcal{X}$, by an adversary, so she could identify the secret value (i.e. questions for Shannon entropy may be formulated as: "Does $x \in \mathcal{X}'$?", for some $\mathcal{X}' \subseteq \mathcal{X}$.). In this case the adversary makes queries about features related to the secret, narrowing down the possible set the secret might be in. Using the same example as before, where the secret was a 4-digit password, the adversary might ask: "Does the secret include the number 0?", if the answer is yes, we could exclude all secrets which do not include 0, thus narrowing the set of possible values. This way of thinking also takes us to an uncertainty measure. Shannon entropy is defined as

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

As presented by Smith [2009] both of these metrics may seem promising, but they are not ideal for some scenarios. For instance the simple scenario when an adversary has only one guess, and wants to guess correctly.

Since guessing entropy and Shannon entropy do not cover important cases such as the one we just presented, Smith [2009] and Braun et al. [2009] proposed a new metric called *Bayes Vulnerability*. The idea behind Bayes vulnerability is that the adversary has only one shot at guessing the secret, thus he wants to make the most out of her attempt by guessing what has more probability. It is defined as follows.

Definition 2.4 (Bayes vulnerability)**.** *Given a prior distribution π over a set of secrets $\mathcal{X}$, the Bayes vulnerability is defined as*

$$V_1(\pi) \qquad := \qquad \max_{x \in \mathcal{X}} \ \pi_x.$$

Example 2.5. *We can exemplify a few instances of Bayes vulnerability. Let us think about 3 cases: a coin, a 6-sided die where the secret is whether or not a number smaller or equal to 2 is rolled, and a 20-sided die where the secret is one specific face.*

- *The distribution for the first case can be represented as* $\pi^{coin} = \{1/2, 1/2\}$*, thus the optimal guess is any side of the coin. This results in* $V_1(\pi^{coin}) = 1/2$*, which means the probability that the adversary guesses which side of the coin was tossed is* $1/2$*.*
- *In the second scenario, we can represent the distribution as* $\pi^{6-die} = \{1/3, 2/3\}$*, thus the optimal guess is that the number rolled will be greater than 2. This results in* $V_1(\pi^{6-die}) = 2/3$*, which means the probability that the adversary guesses correctly if the number tossed was lower than 2 is* $2/3$*.*
- *And the last case has a uniform probability distribution with all values equal to* $1/20$*, thus any side of the die is an optimal guess. This results in* $V_1(\pi^{20-die}) = 1/20$*, which means the probability that the adversary guesses which side of the 20-sided die was tossed is* $1/20$*.*

△▽[2]

Bayes vulnerability is a good way to measure vulnerability, since it gives us a very straightforward way to think about it. But there is a nuance, it assumes that the adversary is interested in discovering the entirety of the secret in one try. This is not valid in every scenario. The adversary might be interested in discovering the secret partially, she can be happy to discover the secret in more than one try, or she even may not want to make a guess if she is penalized by guessing wrong.

To cover all of these scenarios Alvim et al. [2012b] proposed the *g-vulnerability framework*. This framework was shown to generalize the Bayes vulnerability, guessing entropy, and shannon entropy. g-vulnerability adds more expressiveness, thus making us able to better depict instances that weren't previously covered.

As stated on Alvim et al. [2020].

> The basic pillar of the g-vulnerability is that the knowledge about a secret X is important only to the extent that it can be exploited by an adversary, enabling her to take some action that rewards her.

That means that if the adversary knows the password for the user's bank account, she can log-in and steal money, or if she knows the exact location of a courier, she can intercept it. Thus to model its operational scenario there is an specific set $\mathcal{W}$ of actions the adversary can take. This set of actions $\mathcal{W}$, as we will call it from now on, can be the same as the set of secrets $\mathcal{X}$ or not.

To model these scenarios, the g-leakage framework presents a function that given the guesses and secrets, rewards the adversary depending on how much he is expected to benefit from an action.. This is defined as the *gain function*.

[2]From now on we will use △▽ to denote the end of an example.

Definition 2.6 (Gain function)**.** *Given a finite set of secrets $\mathcal{X}$ and a set of guesses $\mathcal{W}$, where $\mathcal{X} \neq \emptyset$ and $\mathcal{W} \neq \emptyset$, a* gain function *is a function $g : \mathcal{W} \times \mathcal{X} \to \mathbb{R}$.*

The value given by the gain function $g(w, x)$ represents how much the adversary is rewarded by taking action $w \in \mathcal{W}$ when the secret is $x \in \mathcal{X}$. The values returned by the function can be arbitrary in specific cases, but it is very common to return values in the interval $[0, 1]$. When the value is restricted to this interval, it is straightforward that when $g(w, x) = 0$, the adversary has no gain by choosing w when secret is x. Similarly, when $g(w, x) = 1$, w is an optimal action for secret x. From now on when we will call $\mathbb{G}^1\mathcal{X}$ the set of gain functions restricted to $[0, 1]$.

Now we can define g-vulnerability, which is fundamentally the maximization of the expected gain of an adversary, given a distribution π over the possible values of a secret X.

Definition 2.7 (g-vulnerability)**.** *Given a prior distribution π, a set of secrets $\mathcal{X}$,a gain function $g : \mathcal{X} \times \mathcal{W} \to \mathbb{R}$, and a set of guesses $\mathcal{W}$, the g-vulnerability is a function defined as*

$$V_g(\pi) \quad := \quad \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi_x g(w, x).$$

We can expand on this by using an example from Alvim et al. [2020, chap. 3, p. 25].

Example 2.8. *With $\mathcal{X} = x_1, x_2$ and $\mathcal{W} = w_1, w_2, w_3, w_4, w_5$, let gain function g have the (rather arbitrarily chosen) values shown in the following matrix:*

G	x_1	x_2
w_1	-1.0	1.0
w_2	0.0	0.5
w_3	0.4	0.1
w_4	0.8	-0.9
w_5	0.1	0.2

To compute the value of V_g on, say, $\pi = (0.3, 0.7)$, we must compute the expected gain for each possible action w in $\mathcal{W}$, given by the expression $\sum_{x \in \mathcal{X}} \pi_x g(w, x)$ for each one, to see which of them is best. The results are as follows.

$$\begin{aligned}
\pi_{x_1}(w_1, x_1) + \pi_{x_2} g(w_1, x_2) &= 0.3 \times (-1.0) + 0.7 \times 1.0 &= 0.40 \\
\pi_{x_1}(w_2, x_1) + \pi_{x_2} g(w_2, x_2) &= 0.3 \times 0.0 + 0.7 \times 0.5 &= 0.35 \\
\pi_{x_1}(w_3, x_1) + \pi_{x_2} g(w_3, x_2) &= 0.3 \times 0.4 + 0.7 \times 0.1 &= 0.19 \\
\pi_{x_1}(w_4, x_1) + \pi_{x_2} g(w_4, x_2) &= 0.3 \times 0.8 + 0.7 \times (-0.9) &= -0.39 \\
\pi_{x_1}(w_5, x_1) + \pi_{x_2} g(w_5, x_2) &= 0.3 \times 0.1 + 0.7 \times 0.2 &= 0.17
\end{aligned}$$

Thus we find that w_1 *is the best action and* $V_g(\pi) = 0.4$

△▽

The g-vulnerability framework is very expressive, much more than what we presented on Example 2.8. Depending on which function we choose, we can model a wide variety of scenarios so our measurement of vulnerability is more faithful to reality.

There is a particularly important gain function: the *identity gain function*[Alvim et al., 2020, chap. 3, p. 29], this function depicts the scenario where the adversary only has gain if he guesses the exact value of the secret in one try. The actions here are values that can be guessed. When the adversary guesses right, it has a gain of 1, and if she guesses wrong it has 0 gain.

$$g_{id}(w, x) := \begin{cases} 1, & \text{if } w = x, \\ 0, & \text{if } w \neq x. \end{cases}$$

This gain function has an important implication: it yields Bayes vulnerability.

Theorem 2.9 (Alvim et al. [2020])**.** *Vulnerability under* g_{id} *coincides with Bayes vulnerability, for all* $\pi : \mathbb{D}\mathcal{X}$*:*

$$V_{g_{id}}(\pi) \quad = \quad V_1(\pi) \quad .$$

Proof. Alvim et al. [2020, chap. 3, p. 30]. □

There are many other interesting gain functions, and they can be modelled to describe appropriately numerous different scenarios. Alvim et al. [2016] showed that $V_g's$ are exactly the family of information functions satisfying a set of reasonable properties. Here we restrict ourselves to the relevant gain functions from this scope.

2.1.2 Channels and updates

As we discussed previously the user wants to protect a secret from the adversary when it is being used. The QIF framework depicts the processing of the secret by a system. The system can represent many things such as security protocols or computer programs. This is modeled as a information-theoretic *channel*, which is a probabilistic function from inputs to outputs, usually depicted by a matrix. If we go back to our 4-digit password

being the secret, we can think about the channel as a representation of the ATM machine or the bank app, which will take the password as input for access. The output is to unlock access or not, depending on the correctness of the password.

Definition 2.10 (Channel matrix)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets representing secret input and observable output values respectively. Meaning each x is mapped to a distribution on y. A channel matrix C from $\mathcal{X}$ to $\mathcal{Y}$ is a stochastic matrix, indexed by $\mathcal{X} \times \mathcal{Y}$, where each entry $\mathsf{C}_{x,y}$ represents the conditional probability $p(y|x)$ of the system producing output $y \in \mathcal{Y}$ when input is $x \in \mathcal{X}$.*

Thus a channel matrix has type $\mathcal{X} \times \mathcal{Y} \mapsto [0, 1]$, this can also be represented as $\mathcal{X} \mapsto \mathbb{D}\mathcal{Y}$. This means that each x is mapped to a distribution on $\mathcal{Y}$. To be more concise on definitions, we will adopt the representation $\mathcal{X} \nrightarrow \mathcal{Y}$ for $\mathcal{X} \to \mathbb{D}\mathcal{Y}$. This representation also abstracts unimportant attributes, such as program code and protocol entities, while keeping the information-theoretic properties.

We exemplify this with the channel matrix C (2.1) where each row represents the probability distribution that each secret x_i outputs each observable y_j.

C	y_1	y_2	y_3	y_4
x_1	$1/4$	$1/4$	$1/4$	$1/4$
x_2	$1/2$	$1/3$	$1/6$	0
x_3	0	1	0	0

(2.1)

The channel is a pivotal part of QIF, as it affects directly the adversarial knowledge about the secret X. This happens when we the initial knowledge the adversary has, the prior distribution π, interacts with the channel. For instance, lets suppose the channel above produces the output y_4, the adversary can infer that the secret is x_1. This is because the input x_1 is the only secret able to output y_4. But by making this assertion, we assume that the adversary knows how the channel works.

One special case that will be relevant to us further on this thesis is the null channel [Alvim et al., 2020, p60].

Definition 2.11 (Null Channel $\mathbb{O}$)**.** *The null channel $\mathbb{O}$ is a channel that leaks everything. It is the mapping of $\pi \mapsto \sum_x \pi_x[[x]]$. The corresponding reduced channel matrix is the identity matrix I.*

The adversary, after observing an output y, updates her knowledge about x, from the prior distribution, to a posterior distribution. This can be calculated using the Bayes' theorem

$$p(x|y) \quad = \quad \frac{p(y|x)p(x)}{p(y)} \quad . \tag{2.2}$$

But to achieve this result, there is a process that must be completed. The first step is to calculate the joint distribution. It is defined as follows

Definition 2.12 (Joint distribution)**.** *The joint distribution is defined as the probability of each input-output pair. We denote this by*

$$p_{XY}(x, y) := \pi_x \mathsf{C}_{x,y}.$$

The joint distribution can be represented as a matrix $\mathsf{J} : \mathcal{X} \times \mathcal{Y}$*, where each* $\mathsf{J}_{x,y}$ *represents the probability of the input-output pair* (x, y)*.*

So applying the prior $\pi = \{1/2, 1/3, 1/6\}$ on the channel C, Table 2.1, we obtain the following joint J (2.3). On J, each cell represents the joint probability of each pair (x, y).

J	y_1	y_2	y_3	y_4
x_1	$1/8$	$1/8$	$1/8$	$1/8$
x_2	$1/6$	$1/9$	$1/18$	0
x_3	0	$1/6$	0	0

(2.3)

After achieving the joint distribution, the adversarial knowledge has been updated. But to obtain new information some calculations must be made. Firstly we find the marginal distribution on $\mathcal{Y}$. This marginalization is done by the sum of each column in J. Since $p(y) = \sum_{x:\mathcal{X}} p(x, y)$. For Table 2.3, we get $\{7/24, 29/72, 13/72, 1/8\}$. And then each value $\mathcal{Y}$ can obtain, gives a posterior distribution on X. Calculations follow the Bayes theorem 2.2. Now we can obtain the following matrix.

Posterior	y_1	y_2	y_3	y_4
x_1	$3/7$	$12/29$	$9/13$	1
x_2	$4/7$	$9/29$	$4/13$	0
x_3	0	$8/29$	0	0

(2.4)

The combination of the distribution $p(y)$ and the matrix 2.4 represent the adversarial knowledge after the interaction between C and π. Each row of matrix 2.4 represents that given an output y_j, what is the probability that each secret x_i generated it. This can also be seen as a distribution over distributions, this is what we call a hyper-distribution. We define hyper distribution below.

Definition 2.13 (Hyper distribution)**.** *Given a finite set $\mathcal{X}$ of secret values, a hyper-distribution Δ is a distribution on distributions on $\mathcal{X}$. Thus Δ has type $\mathbb{D}(\mathbb{D}\mathcal{X})$, which can also be written as $\mathbb{D}^2\mathcal{X}$. A hyper-distribution that results from the interaction of a prior distribution π and a channel C is written $[\pi \triangleright C]$, pronounced "pushing π through C".*

We call inner distributions the set of distributions that are part of the support $\lceil\Delta\rceil$ of Δ. Each inner is represented by δ_i. We call the distribution on the inners outer distributions. They are represented by a_i. Thus we can write $[\pi \triangleright C] = \sum_i a_i[\delta^i]$.

The hyper-distribution obtained by pushing $\pi = \{1/2, 1/3, 1/6\}$ on channel 2.1 can be seen below.

$[\pi \triangleright \mathsf{C}]$	$7/24$	$29/72$	$13/72$	$1/8$
x_1	$3/7$	$12/29$	$9/13$	1
x_2	$4/7$	$9/29$	$4/13$	0
x_3	0	$8/29$	0	0

(2.5)

On this hyper we have as the outer distribution $\{7/24, 29/72, 13/72, 1/8\}$. In other words the probability that the observable output is $y_1 = 7/24$, $y_2 = 29/72$, $y_3 = 13/72$, and $y_4 = 1/8$. Given that an specific y_j was observed, each row, which are the inners, contains the associated probability for the x_i that generated the output.

2.1.3 Posterior vulnerability and information leakage

As discussed on Section 2.1, our goal is to measure leakage caused by the system while processing information. So far we have defined a measure for prior vulnerability and how to model a system. All we have to do now is define a metric to verify how vulnerable the information is after interacting with the system, and then compare it with what we had previously.

It is only natural to apply this measurement to the resulting hyper after π is pushed through C, or in other words $[\pi \triangleright \mathsf{C}]$. But how would it be done?

We know that a channel C maps the prior π to a hyper $[\pi \triangleright \mathsf{C}]$, this hyper is represented by possible states of knowledge about $\mathcal{X}$ that the adversary has, what we called inners, and their respective probabilities of happening, what we called outers. One reasonable way is to calculate the value of V_g on each of these inners and combine them somehow.

The combination of these inners can be done if we take the worst case scenario, or the "maximum" value. But this is not precise in some cases, for instance Alvim et al. [2020, chap. 5, p. 72]. Thus the posterior g-vulnerability is defined as the *expected*

value of the g-vulnerabilty over the hyper-distribution, in other words inners weighted by outers. We define it as follows.

Definition 2.14 (Posterior g-vulnerability)**.** *Given a prior distribution π, a gain function $g : \mathcal{X} \times \mathcal{W} \to \mathbb{R}$, and a channel C, the posterior g-vulnerability is defined as*

$$V_g\left[\pi \triangleright C\right] \quad := \quad \sum_i a_i V_g(\delta^i) \quad \textit{where} \quad [\pi \triangleright C] \quad = \quad \sum_i a_i [\delta^i].$$

In other words $V_g\left[\pi \triangleright C\right]$ is the expected value of V_g over $[\pi \triangleright C]$.

Example 2.15. *Here we can look back at Example 2.5, and make the proper calculations to find which is the posterior vulnerability on this case. We will use the bayes gain function, 2.1.1, to make the calculations.*

$[\pi \triangleright \mathsf{C}]$	$^7/_{24}$	$^{29}/_{72}$	$^{13}/_{72}$	$^1/_8$
x_1	$^3/_7$	$^{12}/_{29}$	$^9/_{13}$	1
x_2	$^4/_7$	$^9/_{29}$	$^4/_{13}$	0
x_3	0	$^8/_{29}$	0	0

We have that the posterior g-vulnerability is

$$\begin{aligned} V_g[\pi \triangleright \mathsf{C}] &= {}^7/_{24} \times V_g({}^3/_7, {}^4/_7, 0) + {}^{29}/_{72} \times V_g({}^{12}/_{29}, {}^9/_{29}, {}^8/_{29}) \\ &\quad + {}^{13}/_{72} \times V_g({}^9/_{13}, {}^4/_{13}, 0) + {}^1/_8 \times V_g(1, 0, 0) \\ &= {}^7/_{24} \times {}^4/_7 + {}^{29}/_{72} \times {}^{12}/_{29} + {}^{13}/_{72} \times {}^9/_{13} + {}^1/_8 \times 1 \\ &= {}^1/_6 + {}^1/_6 + {}^1/_8 + {}^1/_8 \\ &= {}^7/_{12}. \end{aligned}$$

△▽

We can finally define the notion of *information leakage*. All we have to do is compare the values of prior vulnerabilities $V_g(\pi)$ and posterior vulnerabilities $V_g\left[\pi \triangleright \mathsf{C}\right]$. This comparison can be done "multiplicatively", focusing on the relative difference, or "additively", focusing on the absolute difference.

Definition 2.16 (Multiplicative and additive g-leakage)**.** *Let $\pi : \mathbb{D}\mathcal{X}$ be a prior distribution on $\mathcal{X}$, let $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ be a channel, the multiplicative g- leakage is defined as*

$$\mathcal{L}_g^{\times}(\pi, C) \quad := \quad \frac{V_g\left[\pi \triangleright C\right]}{V_g(\pi)},$$

and the additive g-leakage is defined as

$$\mathcal{L}_g^{+}(\pi, C) \quad := \quad V_g\left[\pi \triangleright C\right] - V_g(\pi).$$

It is important to notice that multiplicative leakage is ambiguous as the vulnerability can be positive and negative, thus QIF mainly focuses on non-negative vulnerabilities [Alvim et al., 2020, chap. 3.3.1].

Another result presented by Alvim et al. [2012a], is that a posterior vulnerability is always greater than, or equal to, prior vulnerability. This is intuitively true because the adversary cannot lose information about the secret after it is pushed through a channel. The worst that could happen is that the output does not change the information, thus the adversary does not gain any new information but maintains what she has. This is what we call the *theorem of monotonicity*.

Theorem 2.17 (Monotonicity [Alvim et al., 2020])**.** *Pushing a prior through a channel does not decrease vulnerability: for all distributions π in $\mathbb{D}\mathcal{X}$ and channels C we have*

$$V_g\left[\pi \triangleright C\right] \quad \geq \quad V_g(\pi) \quad .$$

Due to Definition 2.17, the additive leakage is always non-negative, also multiplicative leakage is never smaller than 1.

Example 2.18. *Let us consider the prior $\pi = \{1/2, 1/3, 1/6\}$, which we used to obtain the hyper 2.5, and Example 2.15 with its respective $V_g\left[\pi \triangleright \mathsf{C}\right]$ value. We know that $V_g(\pi) = 1/2$, and $V_g\left[\pi \triangleright C\right] = 7/12$. The multiplicative leakage is*

$$\mathcal{L}_g^{\times}(\pi, C) \quad = \quad \frac{V_g\left[\pi \triangleright C\right]}{V_g(\pi)} \quad = \quad \frac{7/12}{1/2} \quad = \quad 7/6,$$

and the additive is

$$\mathcal{L}_g^{+}(\pi, C) \quad = \quad V_g\left[\pi \triangleright C\right] - V_g(\pi) \quad = \quad 7/12 - 1/2 \quad = \quad 1/12.$$

△▽

The QIF framework is extensive and there are many interesting properties, other than showed here. We choose to restrict this review to these concepts, because they are the ones necessary to understand our work. We refer to Alvim et al. [2020] for various examples that can augment this thesis.

2.2 Expanded QIF framework for dynamic secrets

We already presented many basic QIF concepts until this point. It is possible to apply them in different scenarios, some of them previously exposed here. But some scenarios need more expressive models. For instance *dynamic secrets*. Dynamic secrets are what the name says, secrets that can change over different executions. If we think two users who have access to a vault which can be unlocked through a password. For security reasons this password needs to be changed every two weeks. Each user can have two different passwords, and cycle through them throughout the weeks according to each of their strategies. This scenario cannot be easily modelled by a simple prior. Here we need an additional layer of expressiveness.

Previous work already developed foundation for this subject. Mardziel et al. [2014b] presented the motivation behind it. Alvim et al. [2017a] formulated the theoretical framework to represent priors for dynamic secrets. Alvim et al. [2020] presented a work that is helpful when dealing with updates of this different type of prior. On this section we will present the fundamentals of this extension of QIF.

2.2.1 Adding the notion of secrets and environments

There is an intuitive thought that changing the secret multiple times makes it safer. Mardziel et al. [2014b] presented results that challenge this premise. If the user has a fixed strategy to generate a secret, and uses this strategy multiple times, the secret may become more vulnerable than if the original secret is kept the same. They present an interesting example, reproduced below as on Example 2.19. They use a different model, here it is adapted with QIF terminology, making it possible to model the described scenario.

Example 2.19 (Stakeouts and raids [Mardziel et al., 2014b])**.** *Consider that an evil gang has a stash of special goods which is being hidden from the police. They can move this stash around many different buildings. A police officer wants to stop the evil gang, for this he can choose to stakeout or raid a building. When the police uses the stakeout action, they obtain information without alarming the gang but they do not recover the stash or make any arrests. The action of raiding a building is an active attempt to find the stash, thus if the police chooses to do so they may scare the gang, therefore this cannot be used all the time.*

Adapting the notation used on Mardziel et al. [2014b] to QIF we have that secrets are the locations of the stash. The channels are stakeouts, from which the police can make observations of the movement in each building in order to infer information about the location of the secret. After obtaining information through observation of the channel, the police can make an action: raid any building or leave them untouched, which we can interpret as the set of guesses. Gain occurs if the police raid a where the stash is located.

Initially many scenarios were considered: dynamic secrets, which get updated every few time steps, perfect raids, which only are successful exactly when the building raided contains the stash, imperfect raids, which have a small chance of success even when raiding wrong buildings and a small chance of failure when raiding the correct building, etc. All scenarios presented by the authors had the expected behaviors, making the secret more or less vulnerable, following general intuition about secrecy, depending on the execution.

Then the authors extended their previously defined concepts, adding more "structure" to it. On their second formulation Mardziel et al. [2014b, sec. VI-E], the officer should look after n-buildings (in the authors' numeric example, 5), additionally each building would have a number of floors

$$n\text{-}floors = (n\text{-}buildings - 1)!$$

(in the authors' numeric example 24). and each floor would be controlled by a gang. Each of the gangs would have a stash of special goods, and they each had their own way of moving the stash around the buildings. Although their floor number is fixed. We can see the representation of this idea clearer with Figure 2.1. Each gang has a floor on each building, for instance gang g0 has floor 0 on each of the buildings, gang g1, has floor 1 on each of the buildings and so on.

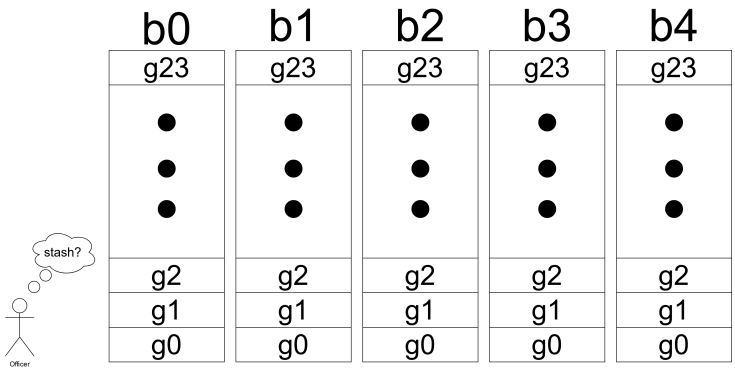

Figure 2.1: Representation of a model for buildings and floors, raided by a police officer.

This changes a few aspects of what we previously described as the QIF modelling of the scenario. Now the secret is not only a building, but also the floor. Each gang has a different strategy to move the stash around, where each strategy is a distinct permutation in-between buildings (e.g. $g_0 = 0, 1, 2, 3, 4$, $g_2 = 0, 3, 4, 1, 2$, $g_3 = 3, 4, 1, 0, 2$). And the channels now are stakeouts that are successful only half of the time. Another important change on this adaptation is that the gain now is the probability of making a raid that is correct, on any gangs' stash. There is also a crucial factor, the strategies of each gang are known to the officer, and that is exactly what causes leakage. Additionally, the authors allowed different amounts of change on the stash location. The results obtained can be seen on Figure 2.2.

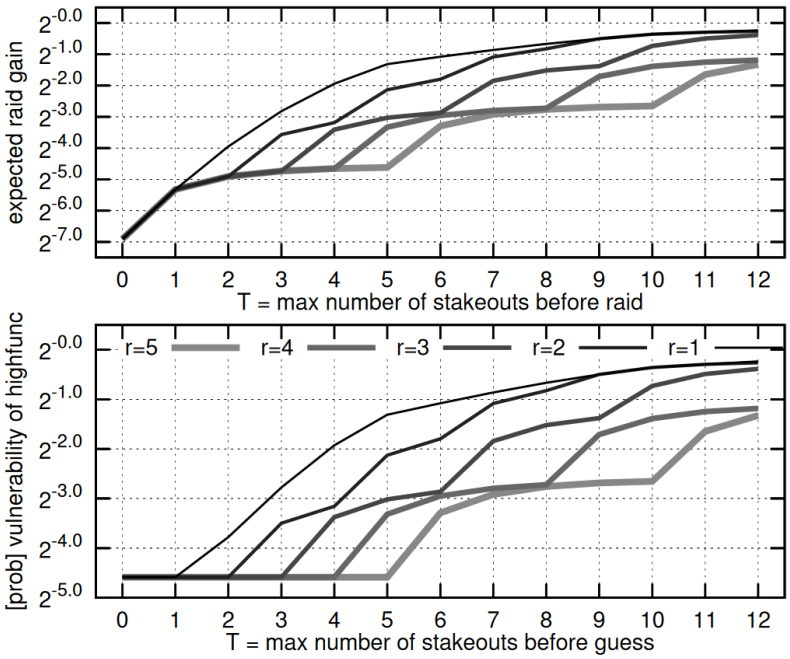

Figure 2.2: Experiment ran by Mardziel et al. [2014b]. with buildings and floors.

Here the parameter r *represents how often the gangs move their stash. So for* $\mathsf{r} = 1$*, the gangs move their stashes every time step, for* $\mathsf{r} = 2$ *, they move every 2 time steps. As we can see on Figure 2.2(top), the more often the stash is moved, the bigger the adversarial expected gain is. We can interpret the adversarial gain here as the chance of a successful raid, which here corresponds to a measure of secret vulnerability. This may contradict our intuition, but it shows that in some cases the more the secret is changed, the more vulnerable it becomes. Figure 2.2(bottom) shows that the adversary also learns the strategy faster,* highfunc *represents the strategy, the more the secret is changed. We can see a direct correlation between those two characteristics.* △▽

We can use what we learned from Example 2.19 to think of a new example. It may seem intuitive at first that if the user has a secret to protect and he changes the secret multiple times, it would make it less vulnerable. For instance, suppose you have access to a laboratory and you have a password for each day, being the respective day of the week. Every day you enter the laboratory for the first time, you have to write on the door the first three letters of your password (i.e MON for Monday, TUE for Tuesday, WED for Wednesday, and so on). If an adversary wants to find out your password, she will have an easier time. If your password was "MONDAY" and you never changed it, it would be harder for the adversary. We can also exemplify this with couriers on a map, this example has ideas derived from the previous, but its modelling is closer to what we are used to on the QIF notation presented here.

Example 2.20. *Suppose we have a map with 5 different cities* $\{x_1, x_2, x_3, x_4, x_5\}$. *Each courier A, B, C, D, E, or F might be on any city, and the adversary wants to find any of them. For this we have a distribution a that represents the possibility of finding each of the couriers* $\{\pi^A, \pi^B, \pi^C, \pi^D, \pi^E, \pi^F\}$.

$1/10$	$1/10$	$2/10$	$3/10$	$2/10$	$1/10$
π^A	π^B	π^C	π^D	π^E	π^F

We then have a set of distributions $\Delta = \{\pi^A, \pi^B, \pi^C, \pi^D, \pi^E, \pi^F\}$ *that presents the distribution that each courier has to move to a city on the map. For instance* π^A *shows that user A equal probability of moving to any of the cities, while* π^B *represents that B can be on city* x_1 *or* x_2 *with equal probability, and won't be anywhere else.*

Δ	π^A	π^B	π^C	π^D	π^E	π^F
x_1	$1/5$	0	$1/3$	$1/2$	$3/4$	$1/3$
x_2	$1/5$	$1/2$	$1/3$	0	$1/4$	0
x_3	$1/5$	0	0	0	0	0
x_4	$1/5$	0	0	0	0	$2/3$
x_5	$1/5$	$1/2$	$1/3$	$1/2$	0	0

Now we can combine both distributions on a single table, which could be interpreted as possible strategies used by defenders, an environment, that we will later define on this section.

Δ	π^A	π^B	π^C	π^D	π^E	π^F
x_1	$1/5$	0	$1/3$	$1/2$	$3/4$	$1/3$
x_2	$1/5$	$1/2$	$1/3$	0	$1/4$	0
x_3	$1/5$	0	0	0	0	0
x_4	$1/5$	0	0	0	0	$2/3$
x_5	$1/5$	$1/2$	$1/3$	$1/2$	0	0
a	$1/10$	$1/10$	$2/10$	$3/10$	$2/10$	$1/10$

This last table looks similar to a hyper that could result from pushing a prior distribution through a channel, but in this case it represents the knowledge of the adversary before any interaction with the system. The adversary knows the all the data. Thus she can calculate the outer probability, which represents single courier to be chosen. And she also knows the inners, the strategy for each of the couriers.

△▽

As we discussed on the beginning of this section, some situations are not faithfully represented by a simple prior distribution $\pi : \mathbb{D}\mathcal{X}$. Some scenarios, like the one from Example 2.20 are better represented by an element of type $\mathbb{D}^2\mathcal{X}$. On these scenarios the

outers represent the probability that an user chooses a specific strategy, while the inners represent the distribution over the secrets for each strategy. This makes us think that in some cases we may need elements of type $\mathbb{D}^n\mathcal{X}$. Alvim et al. [2017a, sec. 6] expanded these ideas and made two important contributions: they showed that an element of type $\mathbb{D}^2\mathcal{X}$ is enough to represent, for leakge purposes, any scenario that $\pi : \mathbb{D}\mathcal{X}$ does not capture; they also defined metrics for prior vulnerability using an element of type $\mathbb{D}^2\mathcal{X}$ as the prior.

As stated previously, Alvim et al. [2017a] formulated the framework to deal with situations where elements of type $\mathbb{D}^2\mathcal{X}$ were needed. The concept was that an user had a context of execution, in which he would get to choose a strategy to use. After his choice, that strategy would be responsible for generating the secret input, which would then interact with the system. We can visualize this concept with Figure 2.3.

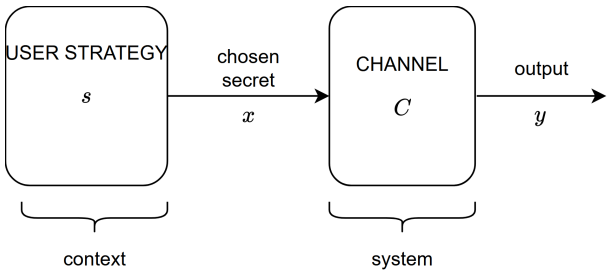

Figure 2.3: Abstraction of a system with a context of execution.

This framework presents many tools to depict different levels of adversarial knowledge about the strategies and its associated probabilities. Her knowledge about the system could be total, partial or concise. Total knowledge is when the adversary knows all the associated probability distributions. Naturally, partial knowledge is when the adversary knows parts of the distributions. Concise, or as we define it π, represents the state of knowledge when the adversary doesn't differentiate between strategies. We will always assume that the adversary has total knowledge about the system, unless stated otherwise.

Two important concepts presented by Alvim et al. [2017a] are those of strategy and environment. Strategy represents the probability of each secret, while environment[3] associates all possible strategies with another distribution of probability.

Definition 2.21 (Strategy and environment)**.** *A strategy, $\sigma \in \mathbb{D}\mathcal{X}$, is a distribution of probability over all possible values of $\mathcal{X}$.*
An environment is a probability distribution over the set of strategies, we can denote an environment by $\Pi \in \mathbb{D}\mathcal{S}$. It can also be represented as $\Pi \in \mathbb{D}^2\mathcal{X}$.

[3]On Alvim et al. [2017a] the notation for environment is En, on this thesis we chose to redefine this as Π to help with more straightforward definitions. Additionally strategies are defined there as π_n, on this thesis we define strategies as σ^n.

We exemplify these concepts with matrices 2.6, here we have two environments, Π^1 and Π^2. They may look similar at first, but they differ greatly when it comes to which strategy will be used.

	σ^1	σ^2	σ^3
x_1	1	0	$1/2$
x_2	0	1	$1/2$
Π^1	$1/2$	$1/2$	0

and

	σ^1	σ^2	σ^3
x_1	1	0	$1/2$
x_2	0	1	$1/2$
Π^2	0	0	1

(2.6)

Both environments behave similarly, we have three columns σ^1, σ^2 and σ^3 that represent the different possible strategies. Each strategy has a different distribution for $\mathcal{X}$. The bottom row is a probability distribution over the possible strategies. This is where Π^1 differs from Π^2, they assign different probabilities for each strategy. These environments can also be represented on a single table, as is shown below.

	σ^1	σ^2	σ^3
x_1	1	0	$1/2$
x_2	0	1	$1/2$
Π^1	$1/2$	$1/2$	0
Π^2	0	0	1

(2.7)

By looking at the matrix 2.7, we can see that the notion of environment generalizes the concept of prior presented on Section 2.1. For instance environment Π^2 has only one distribution possible over the secrets, thus it is a point-hyper. This distribution is $\sigma^3 = \{1/2, 1/2\}$. That is the same representation we use for a simple prior π. In other words a hyper has type $\mathbb{D}^2\mathcal{X}$, but a point-hyper can be represented as a $\mathbb{D}\mathcal{X}$.

It is important to point out that every environment is a hyper, but not every hyper is an environment. This is due to the fact that, in the context of QIF, hypers can represent the values of secrets and their outputs, alternatively environments treat only about secrets.

After defining the concepts of environment and strategy, we can make the distinctions between concise, partial and total knowledge, we define it as follows and illustrate it with Example 2.23.

Definition 2.22 (Concise, Partial and Total Knowledge)**.** *Given an environment* $\Pi : \mathbb{D}^2\mathcal{X}$*, the adversary can have multiple levels of knowledge. The adversary has* total knowledge *if he knows every existing column of the environment.*
She has partial knowledge if she knows a combination of the environment Π*. These combinations are done by merging columns and multiplying their respective probabilities.*
The adversary has concise knowledge if she knows only one column, in other words a simple prior distribution. This is done by merging all the columns of the environment into one, $\pi = [\sum_i \sigma^i]$*.*

Example 2.23 (Alvim et al. [2017a] (sec. 5))**.** *Imagine a system with six users, where users* σ^1,σ^2 *live in state A, users* σ^3,σ^4 *live in state B, users* σ^5,σ^6 *live in state C. If the adversary knows each user by id, he has total knowledge of each user strategy. It can be represented by matrix 2.8.*

Total	σ^1	σ^2	σ^3	σ^4	σ^5	σ^6
x_1	1	0	$1/2$	$1/4$	$3/4$	$1/3$
x_2	0	1	$1/2$	$3/4$	$1/4$	$2/3$
User	$1/10$	$1/10$	$2/10$	$3/10$	$2/10$	$1/10$

(2.8)

If the adversary knows only from which state the user is trying to login, we can represent his knowledge by matrix 2.9, this is done by collapsing the appropriate columns from each of the states.

Partial	σ^A	σ^B	σ^C
x_1	$1/2$	$7/20$	$11/18$
x_2	$1/2$	$13/20$	$7/18$
State	$2/10$	$5/10$	$3/10$

(2.9)

We obtain this by merging σ^1, σ^2 *into* σ^A*, and similarly for the others. Finally if the adversary doesn't know about the users or states, his knowledge is represented by 2.10, which is obtained by collapsing all columns into one.*

Concise	π
x_1	$11/24$
x_2	$13/24$
$[\sigma^\Pi]$	1

(2.10)

Merging columns mean that given a_i, σ^i *and* a_j, σ^j*, the merging is a new inner* $\sigma^i + \sigma^j$ *with its respective outer* $a_i + a_j$*. In other words the adversary may be losing*

information because he cannot differentiate between the strategies represented by σ^i *and* σ^j*.*

□

On traditional QIF we used to treat priors as distributions on $\mathbb{D}\mathcal{X}$, but with the concept of environments we are able to use $\mathbb{D}^2\mathcal{X}$. There is another result on Alvim et al. [2017a, sec. 6] that shows that this generalization also works for $\mathbb{D}^n\mathcal{X}$. Despite this generalization, we can collapse the environment into a simple $\mathbb{D}\mathcal{X}$, the concise prior π, as we mentioned previously. The *concise* is captured by taking the expectation of the inners in an environment, weighted by the outer.

Our main goal in QIF is to measure leakage, and given this new way of describing prior knowledge, we have all the tools we need to define a measure for prior vulnerability. This measurement dismembers the traditional prior vulnerability into two types. One about secrets, even if the strategies are known. The other refers to the uncertainty about the strategy itself. The first type is called *secret vulnerability given an environment*, this measure aims to capture the uncertainty embedded on the secret itself. And it is defined as follows[4].

Definition 2.24 (Secret vulnerability given an environment)**.** *Given a g-vulnerability measure* $V_g : \mathbb{D}\mathcal{X} \to \mathbb{R}$*, the secret vulnerability given an environment* $\Pi : \mathbb{D}^2\mathcal{X}$ *is a function* $V(X \mid \Pi) : \mathbb{D}^2\mathcal{X} \to \mathbb{R}$ *of the environment defined as*

$$V(X \mid \Pi) := \mathop{\mathbb{E}}_{\Pi} V_g.$$

Example 2.25. *Looking back at Table 2.7, we can use Definition 2.24 to measure the prior vulnerability for each environment. Here our* V_g *is Bayes vulnerability.*

$$V(X \mid \Pi^1) = 1/2 V_1(\sigma^1) + 1/2 V_1(\sigma^2) = 1/2 \times 1 + 1/2 \times 1 = 1, \tag{2.11}$$

and

$$V(X \mid \Pi^2) = 1 \times V_1(\sigma^3) = 1/2 \times 1 = 1/2. \tag{2.12}$$

Each of these values represent how vulnerable the secret itself is if outers are known. As we can see $V(X \mid \Pi^1)$ *is high, representing the high vulnerability of the secret. If we look back at matrix 2.7, we can verify that if the adversary knows which strategy is being used on* Π^1 *she learns the secret value. A similar reasoning can be made to verify that* Π^1 *protects the secret more.*

△▽

[4]On Alvim et al. [2017a], the secret vulnerability given an environment is defined as environmental vulnerability $V^{en}(En)$, where En is the environment. This notation is very confusing because it mixes the meanings of different types of vulnerability. Therefore, on this thesis, we adopt the notation presented on Definition 2.24.

From the results from Example 2.25 we can conclude that despite both environments being similar, the value of prior secret vulnerability given an environment may have large differences. We can verify that $V(X \mid \Pi^1) \geq V(X \mid \Pi^2)$, which is expected since Π^1 has randomness on the strategies while Π^2 has its randomness on the secrets. This further endorses the notion of uncertainty being measured on the secret itself. The secret is more vulnerable on Π^1, because if the adversary discovers which strategy is being used, he will know what is the secret. On Π^2, on the other hand, it doesn't matter if the adversary knows which strategy is being used, we will always have some uncertainty related about the secret.

2.2.2 Expanding the concept of vulnerability

As we discussed previously, grounded by Example 2.25, the security of a secret may lie on the lack of knowledge by the adversary on which strategy is being used. The security may also lie within the strategy itself, which Alvim et al. [2017a] call *security by strategy*. As we have defined before, security by strategy is represented by $V(X \mid \Pi)$. Thus a definition to capture the vulnerability within the lack of knowledge about the strategies, or what we call *security by aggregation*, is needed.

Alvim et al. [2017a] defined a metric called strategy vulnerability or $V(S \mid \Pi) : \mathbb{D}^2\mathcal{X} \mapsto \mathbb{R}$. This metric aims to capture two main concepts: represent the certainty of an adversary about which strategy the user will use to generate his secrets; and how predictable is the behavior of an environment. Let us look at the following example to better grasp these concepts.

Example 2.26. *Take into account the following matrix from Alvim et al. [2017a]. This matrix represents four possible strategies* $\{\sigma^1; \sigma^2; \sigma^3; \sigma^4\}$. *There are also the possible environments* $\{\Pi^1, \Pi^2, \Pi^3\}$ *which assign different probabilities to each of the strategies. For* Π^1 *the secret space is* $\mathcal{X} = \{x_1, x_2\}$. Π^1. *It assumes that* σ^1 *will be used half of the time, while* σ^2 *will also be used half of the time.* Π^2 *assumes that* σ^3 *is the only possible strategy. Finally,* Π^3 *assigns* $1/2$ *probability for each* σ^1 *and* σ^4.

	σ^1	σ^2	σ^3	σ^4
x_1	1	0	$1/2$	$9/10$
x_2	0	1	$1/2$	$1/10$
Π^1	$1/2$	$1/2$	0	0
Π^2	0	0	1	0
Π^3	$1/2$	0	0	$1/2$

To follow the premises presented before, it is intuitive to expect that strategy vulnerability should be high on Π^2, *since its a point distribution, therefore the adversary will*

definitely know which strategy is being used. Π^1 *and* Π^3 *behave alike in respect of strategies, but they are different environments, thus we can't have the same values for both. For instance if* Π^1 *produces secret* x_1*, the user will know which strategy is being used,* σ^1*, and this is not the case for* Π^3*. Which takes us to the conclusion that* Π^3 *is more vulnerable in respect to a measurement of strategy vulnerability. So we want a metric that satisfies the following order:* $V(S \mid \Pi^2) > V(S \mid \Pi^3) > V(S \mid \Pi^1)$.

△▽

From this example we can infer that the intuition behind *strategy vulnerability* is that a strategy is known given a concise prior if $V_g(\pi) \approx V(X \mid \Pi)$. The reason being that if these values are similar, the environment must also be similar to it's concise, since another way to read this equation is $V_g(\mathbb{E}\,\Pi) \approx \mathbb{E}_\Pi\, V_g$. This definition reflects correctly wrt. the ordering of Example 2.26 and is defined as follows[5].

Definition 2.27 (Strategy vulnerability given a concise prior)**.** *Given a g-vulnerability measure* $V_g : \mathbb{D}\mathcal{X} \to \mathbb{R}$*, the strategy vulnerability given a concise prior* π*, and an environment* Π *is defined as the ratio*

$$V(S \mid \pi) := \frac{V_g(\pi)}{V(X \mid \Pi)}.$$

By observing definitions 2.24 and 2.27, it is noticeable that they relate to each other. Alvim et al. [2017a] state that when looking at traditional prior vulnerability, we have a notion of *perceived security*. More specifically we can decompose the traditional concept of prior vulnerability, Definition 2.7 into two factors, security by aggregation and security by strategy:

$$\underbrace{V_g(\pi)}_{\text{perceived security}} = \underbrace{V(S \mid \pi)}_{\text{security by agg}} \times \underbrace{V(X \mid \Pi)}_{\text{security by strat}} . \tag{2.13}$$

Equation 2.13 expresses that prior vulnerability can be allocated into strategy vulnerability given a concise prior and secret vulnerability given an environment, with inversely proportional behavior. As we discussed before prior secret vulnerability given an environment is more closely related to secrets than strategy vulnerability given a concise prior, which relates the strategies. Therefore the main goal to protect a secret within this scenario is to *decrease secret vulnerability given an environment* to the cost of increasing strategy vulnerability given a concise prior. This will make the secret safer.

[5]Following the change on Definition 2.24 for prior secret vulnerability given an environment, on this thesis we also adopt a different notation for strategy vulnerability. Here we call it strategy vulnerability given a concise prior, whereas on Alvim et al. [2017a] the notation for strategy vulnerability is $V^{st}(En)$.

2.2.3 Developing on partial knowledge

Definitions concerning environments, so far, have considered an adversary with either concise or total knowledge. But Definition 2.22 already gives us a hint that this may not be the case every time. In fact, on most real-world scenarions the adversary will have partial knowledge. Therefore Alvim et al. [2017a] presents another tool to depict this situation: model of adversarial knowledge.

A *model* of adversarial knowledge is a hyper $M : \mathbb{D}^2\mathcal{X}$ that depicts the knowledge the adversary may have on how secrets are generated. This model can represent many different levels of knowledge about the environment. Each inner σ^n in M represents a strategy the adversary understands as possible. Each outer associated with a inner represents the probability of that strategy being used. The environment Π is representative of an adversary with total knowledge, while π represents the adversary with concise knowledge.

The process of abstraction is done by taking Π and multiplying it by an *aggregation matrix* A that associates the strategies which the adversary cannot distinguish. This aggregation matrix can be deterministic or probabilistic.

Definition 2.28 (Aggregation matrix (p. 14,15 Alvim et al. [2017a]))**.** *The aggregation matrix A is a channel matrix of type $\mathcal{S} \times \mathcal{S}$, in which each entry is $A(i, j)$ is the probability $p(\pi|\mu)$ of an adversary mapping strategy π to μ.*

Following the definition of aggregation matrix, we can now relate this concept to the abstraction of hypers. We define it as follows:

Definition 2.29 (Abstraction of a hyper)**.** *A hyper M is an abstraction of another hyper Π, denoted by $M \sqsubseteq \Pi$, iff $M = \Pi \cdot A$ for some aggregation matrix A.*

We can see on Figure 2.2.3 this update works. At first we have an environment Π that is transformed into a joint matrix J. J is then multiplied by the aggregation matrix A that associates σ^1 and σ^2 at full. Which results in the final model M.

Π

	σ^1	σ^2	σ^3
	$1/2$	$1/3$	$1/6$
x_1	1	0	$1/3$
x_2	0	$1/2$	$1/3$
x_3	0	$1/2$	$1/3$

$\rightarrow$ J

$9/18$	0	$1/18$
0	$3/18$	$1/18$
0	$3/18$	$1/18$

$\cdot$ A

	$\sigma^{1,2}$	σ^3
σ^1	1	0
σ^2	1	0
σ^3	0	1

$=$ M

	$\sigma^{1,2}$	σ^3
	$5/6$	$1/6$
x_1	$3/5$	$1/3$
x_2	$1/5$	$1/3$
x_3	$1/5$	$1/3$

Figure 2.4: Example of Aggregation Matrix

Alongside the notion of abstractions and models, the vulnerabilities proposed by Alvim et al. [2017a] were also enhanced. Prior secret vulnerability given an abstraction is defined as follows:

Definition 2.30 (Prior secret vulnerability given an abstraction)**.** *The vulnerability of the secret in an environment* Π*, when the adversary's model is abstraction* M *is given by*

$$V(X \mid \Pi, M) := \sum_{\pi} \Pi_{\sigma} \sum_{\mu} A(\mu, \pi) \sum_{x} \pi(x) g(w, x)$$

The intuition behind this definition is that the adversary chooses his actions based on the model M, but the actual gain still has to be measured w.r.t. the real strategies from Π.

Prior strategy vulnerability given an abstraction was also redefined.

Definition 2.31 (Prior strategy vulnerability given an abstraction)**.** *Given a g-vulnerability measure* $V_g : \mathbb{D}\mathcal{X} \to \mathbb{R}$*, the strategy vulnerability in an environment* Π*, obtained through an aggregation matrix* A*, is defined as the ratio*

$$V(S \mid \Pi, M) := \frac{V(X \mid \Pi, M)}{V(X \mid \Pi)}.$$

The change here is simpler, given that strategy vulnerability given a concise prior is defined as a function of secret vulnerability given an environment.

2.3 Hidden Markov Models: what they are and how they interact

One of our definitions, Section 4.4, worked with a notion of updating the secret and for this it used markovs, therefore we introduce it here. Markov is a common name within the probability scope. There are many kinds of markov processes. They are stochastic processes that depend only on its current state to predict the next state [Oksendal, 2013]. The most known Markov process is the markov chain. It describes a series of possible events in which the probability of each subsequent event depends only on the state of the previous event [Gagniuc, 2017]. As the Markov chain, there are many others. Our main interest lies on the Hidden Markov models. This type of markov models different types of signals in order to accommodate concurrent processes [Varga and Moore, 1990]. The QIF approach for Hidden Markov models is slightly different, we follow the definitions from Alvim et al. [2020, chap. 13, and 14] and present it on the remainder of this section. From now on we will refer to Hidden Markov Models as only markovs, for simplicity.

These updates are important to understand the process behind the development of our solution. Specifically on Section 4.4 markovs will be used considerably.

2.3.1 A brief introduction on markovs

On QIF a *markov* is a stochastic matrix that updates the secret without producing any observable outputs [Alvim et al., 2020, chap. 13]. Markovs are similar to channels, in the sense that they can be represented as matrices and interact with secrets. The main difference between them is that markovs take the secrets and update them, while channels produce observable outputs. We present the definition for markovs below.

Definition 2.32 (Markov)**.** *Let $\mathcal{X}$ be a finite set representing secret input values. A* markov *matrix M from $\mathcal{X}$ to $\mathcal{X}$ is a stochastic matrix, indexed by $\mathcal{X} \times \mathcal{X}$, where each entry M_{x_i,x_j} represents the probability that the secret is x_i is updated to x_j.*

As Alvim et al. [2020] states, markovs do not leak and channels do not update. A markov takes a prior π and transforms it into a point-hyper on π'. Since the markov only changed the prior, it does not leak. But if $\pi \neq \pi'$ then the markov has updated the secret. A similar thought can be brought to the channel, that takes the prior π to a hyper Δ. If the support of Δ is non-singleton, then the channel has leaked information about the secret.

The update on the prior through the markov is done by multiplying the matrices. If the prior π can be represented as the vector $A = [a_{1j}]$ with dimensions $1 \times n$ and the markov can be represented by the matrix $B = [b_{ij}]$ with dimensions $n \times n$, it is always a square matrix because it maps secrets to secrets, the product AB will be the resulting markov with dimensions $1 \times n$. $AB = [c_{1j}]$, where $c_{1j} = a_{11}b_{1j} + a_{12}b_{2j} + \cdot + a_{1n}b_{nj}$.

We can look at the example below to better grasp these concepts.

Example 2.33. *Suppose that we have 4 possible values a secret can assume $\mathcal{X} = \{1, 2, 3, 4\}$. Their initial probabilities are $\pi = \{1/2, 1/4, 1/4, 0\}$ respectively. Then we can add arbitrary update rules such as:*

- *If the secret is an even number, it will become an odd number with $1/3$ probability for each, and it has $1/3$ probability of not changing.*
- *If the secret is an odd number, it will become 2 with probability $1/3$ and 4 with probability $2/3$.*

Usually the prior vector is represented as a column, but for this operation we will use its transpose. This can be easily represented as the markov below.

$$\begin{array}{|c|cccc|}\hline \pi^T & 1/2 & 1/4 & 1/4 & 0 \\ \hline \end{array} \quad \cdot \quad \begin{array}{|c|cccc|}\hline \mathsf{M} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1/3 & 0 & 2/3 \\ 2 & 1/3 & 1/3 & 1/3 & 0 \\ 3 & 0 & 1/3 & 0 & 2/3 \\ 4 & 1/3 & 0 & 1/3 & 1/3 \\ \hline \end{array} \quad ,$$

in which we can run a round of update. As described before it is the product of the vector π and the matrix M. This operation results on matrix 2.33.

$$[\pi\mathsf{M}] \quad = \quad \begin{array}{|c|cccc|}\hline (\pi^T)' & 1/12 & 1/3 & 1/12 & 1/2 \\ \hline \end{array} \quad .$$

△▽

2.3.2 Markovs models and their interactions with channels

Markovs can interact with channels in two ways, both of them are presented by Alvim et al. [2020, chap 13.4.1]. First we can have a channel leak, then the result is passed on to the markov. This interaction is called *channel then markov*.

The first step on this interaction is the prior π being pushed through C. This leaks some information and the adversary may revise her knowledge on the initial state. The leaked information is represented by the posteriors. Every single observation y has an associated inner p^y and its outer probability p_y, which relies on π and C. After this observation, the adversary takes the initial distribution of M, and deduces what the final distribution σ'_y is to the initial distribution σ^y. The hyper resulting from running $C;M$ on π is composed by all inners σ'_y, each one with its respective outer p_y.

Example 2.34 (Alvim et al. [2020] (chap. 13, p. 235-239))**.** *Suppose we have a prior $\pi : \{1/2, 1/2\}$ and a channel and a markov with the same format*

$$\mathsf{C} \quad = \quad \mathsf{M} \quad = \quad \begin{array}{|cc|}\hline 0 & 1 \\ 1/2 & 1/2 \\ \hline \end{array} \quad .$$

The interaction $\mathsf{C};\mathsf{M}$ on an uniform prior, we have that

$$[(1/2, 1/2) \triangleright \mathsf{C}] \quad = \quad 3/4 * [(2/3, 1/3)] + 1/4 * [(0, 1)] \quad ,$$

then we run M on each of those inners separately, giving

$$[(2/3, 1/3) \triangleright \mathsf{M}] \quad = \quad [(2/3, 1/3)] \quad \textit{and} \quad [(0, 1) \triangleright \mathsf{M}] \quad = \quad [(1/2, 1/2)]$$

and the combining of both hypers, weighted by the outer $[(3/4, 1/4)]$, results in $3/4 * [(5/6, 1/6)] + 1/4 * [(1/2, 1/2)]$.

△▽

The other way that this interaction can happen is to have a markov update the secret, then the channel leak. This interaction is called *markov then channel*.

Here the process is the inverse of the previous one. The prior π first interacts with a markov M. This updates the prior to a new one. The updated prior π' is then pushed through C, which leaks some information.

Example 2.35 (Alvim et al. [2020] (chap. 13, p. 235-240))**.** *As in the previous example, suppose we have a prior* $\pi : \{1/2, 1/2\}$ *and a channel and a markov with the same format*

$$\mathsf{C} \quad = \quad \mathsf{M} \quad = \quad \begin{array}{|c|c|} \hline 0 & 1 \\ \hline 1/2 & 1/2 \\ \hline \end{array} \quad .$$

The interaction $\mathsf{M};\mathsf{C}$ *on an uniform prior, we have that*

$$[(1/2, 1/2) \triangleright \mathsf{M}] \quad = \quad [3/4, 1/4] \quad ,$$

which is obtained by a simple matrix multiplication. Now we conclude pushing the resulting prior π' *through* C*, giving*

$$[(3/4, 1/4) \triangleright \mathsf{C}] \quad = \quad 7/8 * [(6/7, 1/7)] + 1/8 * [(0, 1)] \quad .$$

△▽

With this we conclude the literature review for the understanding of this work.

Chapter 3

Extending on Objectives and Knowledge Update

In this chapter we present the first relevant result of this thesis: the formulation of update functions for the general case where the prior is represented as hyper distributions. Section 3.1.1 introduces the concept of a direct update, where each inner is pushed through the channel independently, and the result is weighted by the outer. Section 3.1.2 introduces a indirect type of update, where the prior hyper distribution is transformed into a simple distribution and then pushed through the channel. This chapter solves the objective proposed on Section 1.1.1.

3.1 Defining knowledge update for dynamic secrets

As we stated previously, our main objective is to verify the leakage when we have the prior knowledge represented by a hyper distribution. As we showed on Section 2.2, literature already give us the tools to represent this type of prior knowledge and how to calculate its respective vulnerabilities. From this, the first step we have to take is to define how the environment will interact with the channel. This interaction will produce a new element that represents the update in knowledge for the adversary. There are two types of update that we used.

3.1.1 Direct Update

The first type of update is what we called *direct update*. It works as a direct interaction of environment and channel. If we look back at the traditional QIF framework, we have a prior $\pi : \mathbb{D}\mathcal{X}$ interacting with a channel matrix $C : \mathcal{X} \to \mathbb{D}\mathcal{Y}$, this then results on a hyper $\Delta : \mathbb{D}(\mathbb{D}\mathcal{X})$. On the novel approach we interact the prior environment $\Pi : \mathbb{D}^2\mathcal{X}$ and $C : \mathcal{X} \to \mathbb{D}\mathcal{Y}$, but it is noticeable that the type of the channel does not change. This can only mean that it will accept the same objects as input. If we look closely at the environment Π, we can say that it is composed by many inners δ^i weighted by an outer a_i. Each of these inners represents a different strategy, which can also be seen as a separate prior. Thus we can take each of these priors and push it through the channel.

By pushing each inner δ^i, we have different resulting hypers. But we still have the outers a_i relating to each of these hypers. From this interaction all we have to do is combine them into a single object. This resulting object will have type $\mathbb{D}^3\mathcal{X}$. We define it as follows.

Definition 3.1 (Environment-through-channel hyper)**.** *Let $\Pi : \mathbb{D}^2\mathcal{X}$ be a hyper on $\mathcal{X}$ such that $\Pi = \sum_i a_i[\pi^i]$, where each $\pi^i : \mathbb{D}\mathcal{X}$ is an inner distribution and each a_i is the probability the outer assigns to π^i. Let $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ be a channel. Then result of pushing Π through C is a 3-hyper (i.e. an object of type $\mathbb{D}^3\mathcal{X}$), represented by $\{\Pi \triangleright \mathsf{C}\}$, it is defined as:*

$$\{\Pi \triangleright \mathsf{C}\} \qquad := \qquad \sum_i a_i[[\pi^i \triangleright \mathsf{C}]]$$

Where each of the brackets [] represent a point hyper distribution. From this we can infer that an environment-through-channel hyper is of type $\mathbb{D}^3\mathcal{X}$.

Example 3.2 (Direct update usage. Part I)**.** *We will exemplify how the update works by pushing the hyper akasha through the identity channel I.*

akasha

σ^1	σ^2	σ^3
$1/3$	$1/3$	$1/3$
1	*0*	*0*
0	*1*	*0*
0	*0*	$1/2$
0	*0*	$1/2$

$\triangleright$

I

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Pushing the distribution σ^1 results as follows:

$$\sigma^1 \triangleright I = \begin{array}{|c|}\hline 1 \\ 0 \\ 0 \\ 0 \\ \hline \end{array} \triangleright I \Rightarrow Joint : \begin{array}{|c|c|c|c|}\hline 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ \hline \end{array} \Rightarrow [\sigma^1 \triangleright I] : \begin{array}{|c|}\hline 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ \hline \end{array}$$

Pushing the distribution σ^2 results as follows:

$$\sigma^2 \triangleright I = \begin{array}{|c|}\hline 0 \\ 1 \\ 0 \\ 0 \\ \hline \end{array} \triangleright I \Rightarrow Joint : \begin{array}{|c|c|c|c|}\hline 0&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ \hline \end{array} \Rightarrow [\sigma^2 \triangleright I] : \begin{array}{|c|}\hline 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ \hline \end{array}$$

Pushing the distribution σ^3 results as follows:

$$\sigma^3 \triangleright I = \begin{array}{|c|}\hline 0 \\ 0 \\ 1/2 \\ 1/2 \\ \hline \end{array} \triangleright I \Rightarrow Joint : \begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 \\ \hline \end{array} \Rightarrow [\sigma^3 \triangleright I] : \begin{array}{|c|c|}\hline 1/2 & 1/2 \\ \hline 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ \hline \end{array}$$

After that we have the following environment-through-channel:

$$\{akasha \triangleright \mathsf{C}\} = \begin{array}{|c|c|c|c|}\hline 1/3 & 1/3 & \multicolumn{2}{c|}{1/3} \\ \hline 1 & 1 & 1/2 & 1/2 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline \end{array}$$

As shown by Alvim et al. [2017a, sec. 6] if your measure of information has type $\mathbb{D}^2\mathcal{X} \to \mathbb{R}$ then you can represent the adversarial knowledge as an object of type $\mathbb{D}^2\mathcal{X}$. If it is of type $\mathbb{D}^n\mathcal{X}$, with $n \geq 3$, you can convert it to an equivalent object of type $\mathbb{D}^2\mathcal{X}$.

Theorem 3.3 (Expressiveness of hypers[Alvim et al., 2017a]). *For every $\pi^n : \mathbb{D}^n\mathcal{X}$, with $n \geq 2$,*

$$V^n(\pi^n) \quad = \quad V^n(\hat{\pi}^2)\,,$$

where $\hat{\pi}^2 : \mathbb{D}^2\mathcal{X}$ is the hyper resulting from marginalizing the joint of π^n w.r.t $Y_2 \times Y_3 \times \cdots Y_{n-1}$.

Proof. Alvim et al. [2017b, p. 29]. □

This result is very important because it allows us to be more expressive when we want to. We have shown with Definition 3.1 that we can represent the posterior knowledge of an adversary with an element of type $\mathbb{D}^3\mathcal{X}$, therefore we can convert it to an object of type $\mathbb{D}^2\mathcal{X}$ and use appropriate measures.

Alvim et al. [2020, chap. 14.3.1], has already shown a way to squash probabilistic functions so they collapse from type $\mathbb{D}^n\mathcal{X}$ to $\mathbb{D}^{n-1}\mathcal{X}$, for $n \geq 1$. Taking advantage of this, we define how our environment-through-channel $\{\Pi \triangleright C\} : \mathbb{D}^3\mathcal{X}$ collapses back into a hyper $[\Pi \triangleright C] : \mathbb{D}^2\mathcal{X}$ as follows.

Definition 3.4 (Squashing of Environment-through-channel)**.** *Let* $\Pi : \mathbb{D}^2\mathcal{X}$ *be a hyper on* $\mathcal{X}$ *such that* $\Pi = \sum_i a_i[\pi^i]$*, where* a_i *is the outer probability of the hyper and* π^i *represents the inner distributions. Let* $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ *be a channel, and let* $\{\Pi \triangleright \mathsf{C}\}$ *be the environment-through-channel, of type* $\mathbb{D}^3\mathcal{X}$*, resulting from pushing* Π *through* C*. Let* μ *be the squashing function of type* $\mathbb{D}^n\mathcal{X} \to \mathbb{D}^{n-1}\mathcal{X}$*, for* $n \geq 1$*, so that for* $\Pi : \mathbb{D}^2\mathcal{X}$ *we have:*

$$(\mu\Delta)_x := \sum_{\delta:\lceil\Delta\rceil} \Delta_\delta \quad \times \quad \delta_x$$

Then the squashing of $\{\Pi \triangleright \mathsf{C}\}$ *can be represented as:*

$$\mu\{\Delta \triangleright \mathsf{C}\}$$

We can also squash an environment of type $\mathbb{D}^2\mathcal{X}$ into a distribution of type $\mathbb{D}\mathcal{X}$. We define it as follows.

Definition 3.5 (Squashing of a hyper)**.** *Let* $\Delta : \mathbb{D}^2\mathcal{X}$ *be a hyper on* $\mathcal{X}$ *such that* $\Delta = \sum_i a_i[\pi^i]$*, where* a_i *is the outer probability of the hyper and* π^i *represents the inner distributions. Let* $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ *be a channel, and let* $\{\Pi \triangleright \mathsf{C}\}$ *be the environment-through-channel, of type* $\mathbb{D}^3\mathcal{X}$*, resulting from pushing* Π *through* C*. Let* μ *be the squashing function of type* $\mathbb{D}^n\mathcal{X} \to \mathbb{D}^{n-1}\mathcal{X}$*, for* $n \geq 1$*. The squashing of* Π *is equal to*

$$\mu\Delta \qquad = \qquad \mu(\mu\{\Delta \triangleright \mathsf{C}\})$$

We can now move to Example 3.6 to better understand how squashing works on both cases.

Example 3.6 (Direct update usage. Part II)**.** *We will exemplify how squashing works by using it on the result of pushing the hyper akasha onto the identity channel. Consider the following environment-through-channel.*

$\{akasha \triangleright \mathsf{C}\}$ =

1/3	1/3	1/3	
1	1	1/2	1/2
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

If we squash it once we get we will take the element of type $\mathbb{D}^3\mathcal{X}$ *to an element* $\mathbb{D}^2\mathcal{X}$*. This is done by a simple multiplication of the outer probabilities of* $\{akasha \triangleright \mathsf{C}\}$ *and the respectives outer probabilities of* $[akasha \triangleright \mathsf{C}]$*.*

$$[akasha \triangleright \mathsf{C}] \quad = \quad \begin{array}{|cccc|}\hline 1/3 & 1/3 & 1/6 & 1/6 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline \end{array}$$

We can squash this 2-hyper once more, by doing it we would obtain an element of type $\mathbb{D}\mathcal{X}$*. This operation takes the outer probabilities of* $[akasha \triangleright \mathsf{C}]$ *and multiplies each of its respective inners. Then we sum all the columns into one.*

$$\pi' \quad = \quad \begin{array}{|c|}\hline 1/3 \\ 1/3 \\ 1/6 \\ 1/6 \\ \hline \end{array}$$

□

This type of update is very important and will be used throughout Chapter 4 for most of the attempts on defining posterior vulnerability measures for dynamic secrets. Including our final formulation.

3.1.2 Indirect Update

The second type of update is called *indirect update*. The strategies and secrets are transformed into an intermediate object, which is then pushed through the channel with standard QIF approach. The result of that is then marginalized so we can have the final results. This update is done by taking the prior environment $\Pi : \mathbb{D}^2\mathcal{X}$ and creating a joint distribution of secrets and strategies. This joint distribution will be of type $\mathbb{D}(\mathcal{X} \times \mathcal{S})$ which can then be pushed through a channel and create a resulting hyper. The channel has to be adapted to receive this joint. It needs to be copied once for each strategy, and its rows combined to each other copy of it. This will be clearer with our next example.

Example 3.7 (Indirect update usage)**.** *Consider the following environment:*

akasha

σ^1	σ^2	σ^3
$1/3$	$1/3$	$1/3$
1	0	0
0	1	0
0	0	$1/2$
0	0	$1/2$

This is then transformed into the join distribution pairing strategies and secrets.

$$\pi_{akasha} \quad = \quad \begin{array}{|c|c|}\hline \sigma^1 x_1 & 1/3 \\ \sigma^1 x_2 & 0 \\ \sigma^1 x_3 & 0 \\ \sigma^1 x_4 & 0 \\ \sigma^2 x_1 & 0 \\ \sigma^2 x_2 & 1/3 \\ \sigma^2 x_3 & 0 \\ \sigma^2 x_4 & 0 \\ \sigma^3 x_1 & 0 \\ \sigma^3 x_2 & 0 \\ \sigma^3 x_3 & 1/6 \\ \sigma^3 x_4 & 1/6 \\ \hline \end{array} \tag{3.1}$$

The adaptation of the channel occurs. For each strategy, we double the rows of the channel. In this case we have an identity channel and three strategies. Therefore we have 3 identity channels concatenated into one.

$$I \quad = \quad \begin{array}{|c|c|c|c|}\hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline \end{array} \quad , \qquad III \quad = \quad \begin{array}{|c|c|c|c|}\hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline \end{array}$$

The resulting hyper is obtained by the simple QIF update method of pushing a prior through a channel.

$$[\pi_{akasha} \triangleright \mathit{III}] \quad = \quad \begin{array}{|c|cccc|} \hline & 1/3 & 1/3 & 1/6 & 1/6 \\ \hline \sigma^1 x_1 & 1 & 0 & 0 & 0 \\ \sigma^1 x_2 & 0 & 0 & 0 & 0 \\ \sigma^1 x_3 & 0 & 0 & 0 & 0 \\ \sigma^1 x_4 & 0 & 0 & 0 & 0 \\ \sigma^2 x_1 & 0 & 0 & 0 & 0 \\ \sigma^2 x_2 & 0 & 1 & 0 & 0 \\ \sigma^2 x_3 & 0 & 0 & 0 & 0 \\ \sigma^2 x_4 & 0 & 0 & 0 & 0 \\ \sigma^3 x_1 & 0 & 0 & 0 & 0 \\ \sigma^3 x_2 & 0 & 0 & 0 & 0 \\ \sigma^3 x_3 & 0 & 0 & 1 & 0 \\ \sigma^3 x_4 & 0 & 0 & 0 & 1 \\ \hline \end{array} \quad .$$

The hyper $[\pi_{akasha} \triangleright \mathit{III}]$ *can be then marginalized for strategies or secrets, to obtain the results we want. The marginalization here is simple to be made, all we have to do is add the rows that represent each aspect.*

For the strategy marginalization we have:

$$[\pi_{akasha} \triangleright \mathit{III}]^{St} \quad = \quad \begin{array}{|c|cccc|} \hline & 1/3 & 1/3 & 1/6 & 1/6 \\ \hline \sigma^1 & 1 & 0 & 0 & 0 \\ \sigma^2 & 0 & 1 & 0 & 0 \\ \sigma^3 & 0 & 0 & 1 & 1 \\ \hline \end{array} \quad ,$$

and for the secret marginalization we have:

$$[\pi_{akasha} \triangleright \mathit{III}]^{Sec} \quad = \quad \begin{array}{|c|cccc|} \hline & 1/3 & 1/3 & 1/6 & 1/6 \\ \hline x_1 & 1 & 0 & 0 & 0 \\ x_2 & 0 & 1 & 0 & 0 \\ x_3 & 0 & 0 & 1 & 0 \\ x_4 & 0 & 0 & 0 & 1 \\ \hline \end{array} \quad .$$

△▽

This second update is will be further used on Section 4.6, which was one of the attempts on defining posterior vulnerability measures.

Both types of update can be used on any environment, but the resulting object of each type has different ways of being interpreted. Therefore, any measure applied to the resulting objects must take that into consideration.

Chapter 4

Attempts of defining measures for posterior secret and strategy vulnerabilities

In the search for a definition for strategy and secret leakage, we made various attempts that failed for a few different reasons: they did not meet our intuition, they failed important properties, and others. Nevertheless, they will be described here because these attempts give us insights into how secret and strategy leakage do behave. They are also helpful in justifying the need for the final definition described in Chapter 5.

In this chapter, we present each of the preliminary attempts on defining a consistent measure for posterior vulnerabilities given the notion of environments. We start on Section 4.1 with a brief introduction on work towards new measure.

In Section 4.2 we use the direct update, described on Section 3.1.1, and try to apply it on the resulting hyper the definitions already made by Alvim et al. [2017a]. We also show why this definition was not appropriate.

With Section 4.3 we show a few re-definitions of the previous vulnerability metrics that did not work.

Section 4.4 tries a novel approach using markovs, which is important to reveal to us the metrics previously defined may not be sensible enough.

Additionally, Section 4.5 tries to work on this sensitivity issue with the proposal of a new gain function.

And finally, in Section 4.6 we attempt a new type of update function, the indirect update from Section 3.1.2, while taking advantage of the new gain function defined previously.

4.1 How to define measures for posterior vulnerability given environments

As discussed in Section 2, there are many possible scenarios that QIF can encapsulate. Special scenarios require special metrics for leakage. If the user has many possible strategies to generate an input secret, modeling vulnerability will be more complex. For this on Alvim et al. [2017a], a metric is proposed to differentiate two main types of vulnerabilities: Strategy and Environmental. They presented versions for the prior case. In this section, we discuss the need for strategy and secret vulnerability measures and present a few attempts on how to define this measure for the posterior case, and so complete the model to provide an assessment of leakage.

As Alvim et al. [2017a] say, strategy vulnerability is a measure of how much the adversary knows about the strategy being used. Therefore this metric must distinguish scenarios where the adversary knows precisely which is the user's strategy from those where he is uncertain about it. Additionally, we should also take into account the amount of uncertainty about it. Alvim et al. [2017a] go further and propose a metric for prior strategy vulnerability, Definition 2.27. But they do not present a model for updates with a hyper distribution representing the adversarial prior knowledge. Thus literature does not have posterior vulnerability defined for this specific case. Likewise Alvim et al. [2017a] propose that secret vulnerability is a measure of how much the adversary knows about the secret itself. Proposing a prior metric for it, but leaving the posterior case for future work.

4.2 Attempt using direct update and the definitions from Alvim et al. [2017a] for prior vulnerability

The standard QIF framework uses different definitions for prior and posterior vulnerabilities. One of the main reasons for that is that the objects used to attain the values have different types. A prior π is used to calculate the prior vulnerability and has type $\mathbb{D}\mathcal{X}$, a hyper Δ is used to calculate the posterior vulnerability and has type $\mathbb{D}^2\mathcal{X}$.

The is different with environments. We can see this more clearly with Figure 4.1 This figure has an initial hyper $\Pi : \mathbb{D}^2\mathcal{X}$, representing an environment, that is pushed through C. This results in an environment-through-hyper $\{\Pi \triangleright \mathsf{C}\}$, which is represented by the resulting set Δ of distributions of type $\mathbb{D}^2\mathcal{X}$ and the distribution over it $\gamma : \mathbb{D}\mathcal{Y}$. An advantage of the framework presented in Section 3.1 is that we can squash $\{\Delta \triangleright \mathsf{C}\}$ so it also becomes of type $\mathbb{D}^2\mathcal{X}$.

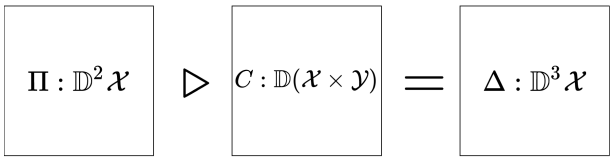

Figure 4.1: Graphical view of the update model from 3.1.1

Using this reasoning, we can take the same metrics for prior vulnerability proposed by Alvim et al. [2017a] and use them on the objects that result from pushing the hyper Π through C. We present these definitions below.

Attempt 4.1 (Posterior strategy vulnerability adapted from Alvim et al. [2017a]). *Let $\Pi : \mathbb{D}^2\mathcal{X}$ be a hyper on $\mathcal{X}$, let $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ be a channel, let $\{a_i\}$ be an outer probability distribution over the inners of $\{\Pi \triangleright \mathsf{C}\}$. The posterior strategy vulnerability, written $V(S \,|\, \Pi \triangleright \mathsf{C})$ is defined as the expected value of $V(S \,|\, \mathsf{C})$ over $\{\Pi \triangleright \mathsf{C}\}$, that is,*

$$V(S \,|\, \Pi \triangleright \mathsf{C}) \quad := \quad \sum_i a_i V(S \,|\, [\sigma^i \triangleright \mathsf{C}]),$$

where $\{\Pi \triangleright \mathsf{C}\} = \sum_i a_i[[\sigma^i \triangleright \mathsf{C}]]$, a_i represents the outer probability of the environment-through-hyper generated, and σ^i represents the inner distributions of the point hypers.

Attempt 4.2 (Posterior secret vulnerability adapted from Alvim et al. [2017a]). *Let $\Pi : \mathbb{D}^2\mathcal{X}$ be a hyper on $\mathcal{X}$, let $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ be a channel, let a_i be an outer probability over the inners of $\{\Pi \triangleright \mathsf{C}\}$. The posterior strategy vulnerability, written $V(X \,|\, \Pi \triangleright \mathsf{C})$ is defined as the expected value of $V(X \,|\, \mathsf{C})$ over $\{\Pi \triangleright \mathsf{C}\}$, that is,*

$$V(X \,|\, \Pi \triangleright \mathsf{C}) \quad := \quad \sum_i a_i V(X \,|\, [\sigma^i \triangleright \mathsf{C}]),$$

where $\{\Pi \triangleright \mathsf{C}\} = \sum_i a_i[[\sigma^i \triangleright \mathsf{C}]]$, a_i represents the outer probability of the environment-through-hyper generated, and σ^i represents the inner distributions of the point hypers.

Example 4.4 shows how both definitions are applied.

Although in principle promising, this re-utilization of definitions did not work. When using the metrics on the hypers Π' the property of monotonicity 2.17 did not hold. Also, a few values intuitively did not pass some of the sanity checks to verify its integrity. This should be clearer with the next example. For this example, we adapt the already established definition of leakage, Definition 2.16, as follows.

Definition 4.3 (Secret and strategy leakage)**.** *Let $\Pi : \mathbb{D}^2\mathcal{X}$ be a hyper on $\mathcal{X}$, let $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ be a channel, the multiplicative strategy leakage given an environment is defined as:*

$$\mathcal{L}^{\times}_{st}(\Pi, \mathsf{C}) \quad := \quad \frac{V(S \,|\, \Pi \triangleright \mathsf{C})}{V(S \,|\, \Pi)} \quad ,$$

the additive strategy leakage given an environment is defined as:

$$\mathcal{L}^{+}_{st}(\Pi, \mathsf{C}) \quad := \quad V(S \,|\, \Pi \triangleright \mathsf{C}) - V(S \,|\, \Pi) \quad ,$$

the multiplicative secret leakage given an environment is defined as:

$$\mathcal{L}^{\times}_{sec}(\Pi, \mathsf{C}) \quad := \quad \frac{V(X \,|\, \Pi \triangleright \mathsf{C})}{V(X \,|\, \Pi)} \quad ,$$

and the additive strategy leakage given an environment is defined as:

$$\mathcal{L}^{+}_{sec}(\Pi, \mathsf{C}) \quad := \quad V(X \,|\, \Pi \triangleright \mathsf{C}) - V(X \,|\, \Pi).$$

Example 4.4 (Calculating $V(S \,|\, \Pi \triangleright \mathsf{C})$ and $V(X \,|\, \Pi \triangleright \mathsf{C})$ without squashing)**.** *Let us think about the following matrix which presents us with three environments: bane, clinkz and dazzle. Each of those will be pushed through the identity channel I.*

	σ^1	σ^2	σ^3	σ^4
x_1	1	0	$1/2$	$9/10$
x_2	0	1	$1/2$	$1/10$
bane	$1/2$	$1/2$	0	0
clinkz	0	0	1	0
dazzle	$1/2$	0	0	$1/2$

$\triangleright$

I	
1	0
0	1

. (4.1)

When we calculate, using the definitions 2.24 and 2.27, we get the following results for each of the environments:

$$\begin{array}{rclcrcl} V(X \,|\, \mathit{bane}) & = & 1 & \text{and} & V(S \,|\, \mathit{bane}) & = & 1/2, \\ V(X \,|\, \mathit{clinkz}) & = & 1/2 & \text{and} & V(S \,|\, \mathit{clinkz}) & = & 1, \\ V(X \,|\, \mathit{dazzle}) & = & 19/20 & \text{and} & V(S \,|\, \mathit{dazzle}) & = & 1. \end{array}$$

The resulting hyper, after squashing according to Theorem 3.4, for each of these environments is represented as follows:

bane

$1/2$	$1/2$
1	0
0	1

,

clinkz

$1/2$	$1/2$
1	0
0	1

,

dazzle

$1/2$	$9/20$	$1/20$
1	1	0
0	0	1

.

Now we can use the new definitions for posterior environmental vulnerability 4.1 and posterior strategy vulnerability 4.2 to make the proper calculations:

$$\begin{array}{rcl@{\quad}l@{\quad}rcl} V(X \,|\, bane \triangleright I) & = & 1 & and & V(S \,|\, bane \triangleright I) & = & 1, \\ V(X \,|\, clinkz \triangleright I) & = & 1 & and & V(S \,|\, clinkz \triangleright I) & = & 1/2, \\ V(X \,|\, dazzle \triangleright I) & = & 1 & and & V(S \,|\, dazzle \triangleright I) & = & 19/20. \end{array}$$

And then we verify the leakage for each case

$$\begin{array}{rcl@{\quad}l@{\quad}rcl} \mathcal{L}^{+}_{sec}(bane, I) & = & 1 - 1 = 0 & and & \mathcal{L}^{+}_{st}(bane, I) & = & 1 - 1/2 = 1/2, \\ \mathcal{L}^{+}_{sec}(clinkz, I) & = & 1 - 1/2 = 1/2 & and & \mathcal{L}^{+}_{st}(clinkz, I) & = & 1/2 - 1 = -1/2, \\ \mathcal{L}^{+}_{sec}(dazzle, I) & = & 1 - 19/20 = 1/20 & and & \mathcal{L}^{+}_{st}(dazzle, I) & = & 19/20 - 1 = -19/20. \end{array}$$

From which we can clearly see that the property of monotonicity is being violated. △▽

From Example 4.4 we can conclude that adapting the definitions from Alvim et al. [2017a], the results for vulnerabilities from pushing Π through C do not respect the property of monotonicity. Therefore they are not satisfactory measures.

Monotonicity does not hold for the non-squashed version as well. Take for instance *dazzle* and its non-squashed and squashed version respectively:

dazzle

1/2	1/2	
1/2	9/10	1/10
1 0	1 0	0 1

,

dazzle

1/2	9/20	1/20
1 0	1 0	0 1

. (4.2)

By making simple calculations we will attain equivalent results. The matrices 4.2 may differ in forms but result in the same values of vulnerability. Therefore we find the same problem with respect to monotonicity. From which we can conclude that Definitions 4.2 and 4.1 do not result in a solution for a vulnerability metric using, either using the non-squashed or squashed versions.

4.3 Attempt defining posterior strategy vulnerability as a function of secret vulnerability

There is an interesting aspect about the hypers that represent the inners from $\{\Pi \triangleright \mathsf{C}\}$. They are not necessarily an environment, in that each $\mathbb{D}\mathcal{X}$ is not a strategy. We can take

as an example the non-squashed version from matrix 4.2 from the previous section. We present them below splitting them into each of its inners.

$$
\begin{array}{c}
[dazzle \triangleright I] \\
\begin{array}{|c||cc|}\hline
1/2 & \multicolumn{2}{c|}{1/2} \\ \hline
1/2 & 9/10 & 1/10 \\ \hline\hline
1 & 1 & 0 \\
0 & 0 & 1 \\ \hline
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
[\sigma^1 \triangleright I] \\
\begin{array}{|c|}\hline
1/2 \\ \hline\hline
1 \\
0 \\ \hline
\end{array}
\end{array}
\quad and \quad
\begin{array}{c}
\{\sigma^4 \triangleright I\} \\
\begin{array}{|cc|}\hline
9/20 & 1/20 \\ \hline\hline
1 & 0 \\
0 & 1 \\ \hline
\end{array}
\end{array}
\quad .
$$

The hypers $[\sigma^1 \triangleright I]$ and $\{\sigma^4 \triangleright I\}$ are the result of pushing σ^1 and σ^2 through channel I, respectively. They are not necessarily environments. This is easy to see if we look at environment *dazzle* before it goes through the channel I.

	σ^1	σ^2	σ^3	σ^4
x_1	1	0	1/2	9/10
x_2	0	1	1/2	1/10
dazzle	1/2	0	0	1/2

,

We can see that the environments have 4 possible strategies. On $[\sigma^1 \triangleright I]$ we only have one strategy depicted, which is different from the environment used for the prior calculations. And it would not make sense to make the same calculations on different objects. The same idea can be used to argue about $\{\sigma^4 \triangleright I\}$.

One could assume that it would make sense to apply metrics for posterior environmental vulnerability on the resulting inners. Starting from that, we redefined posterior strategy vulnerability and made that environmental vulnerability would not change after the environment was pushed through a channel. This comes from the idea that the environment itself does not change from prior to posterior. We, therefore, used the default definition of leakage to evaluate the results, Definition 4.3.

Attempt 4.5 (Posterior strategy vulnerability as a function of secret vulnerability)**.** *Let $\Pi : \mathbb{D}^2\mathcal{X}$ be a hyper on $\mathcal{X}$, let $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ be a channel, let $\{a_i\}$ be an outer probability distribution on the inners of $\{\Pi \triangleright \mathsf{C}\}$. Let π be the concise hyper of each inner of the $\{\Pi \triangleright \mathsf{C}\}$.*

The posterior strategy vulnerability, written $V(S \mid \Pi \triangleright \mathsf{C})$ is defined as the expected value for strategy vulnerability given that each inner was pushed through C over the environmental vulnerability calculated prior to pushing the hyper, that is:

$$V(S \mid \Pi \triangleright \mathsf{C}) \qquad := \qquad \frac{\sum_i a_i V(S \mid [\sigma^i \triangleright \mathsf{C}])}{V(X \mid \Pi)}$$

After formulating this definition we ran many examples with sanity checks. From analyzing the results we found that all the posterior vulnerabilities resulted in 1. So the

values for leakage, despite being in accordance with basic properties, were not adequate. From this, we could conclude that this definition only divides the vulnerability value by itself when calculating leakage. We can see that on Equation 4.3 below.

$$\mathcal{L}^{\times}_{st}(\Pi, \mathsf{C}) \quad = \quad \frac{V^{st}\{\Pi \triangleright \mathsf{C}\}}{V^{st}(\Pi)} \quad = \quad \frac{V^{en}(\Pi)}{\sum_i a_i V(\pi^i)} \times \frac{V(\pi)}{V^{en}(\Pi)} \quad = \quad \frac{V(\pi)}{\sum_i a_i V(\pi^i)} \tag{4.3}$$

From Equation 4.3 we can conclude that $\mathcal{L}^{\times}_{st}(\Pi, \mathsf{C})$, as defined, is the inverse of $V(S \mid \Pi \triangleright \mathsf{C})$. Therefore whenever we calculate leakage we would get 1 as result, indifferent of how the prior was calculated. This approach showed itself not to be informative in practice, thus it would consider every system as equivalent. Following this, we moved on to a new approach.

4.4 Attempt using the update model from McIver and Morgan [Alvim et al., 2020, chap. 13,14]

On Alvim et al. [2020, chap. 13,14] the authors propose a framework that deals with the problem of updating secrets. They also briefly discuss the problem of strategies. There is a special problem proposed by Alvim et al. [2020, exercise. 13.2] on this subject. We present it here with Algorithm 1. It is very important to note that this approach views secrets as a combination of values and strategies, as in a joint distribution.To better understand this algorithm keep in mind that ${}_p\oplus$ is an operator used on QIF to represent a choice between two arguments with p probability. More precisely, $a_p \oplus b$ represents a random choice that takes a with probability p, and b with probability $(1-p)$.

Algorithm 1 Password Histories

```
1:  VAR
2:     password: 0,1                                  - Either 0 or 1.
3:     p,q: [0,1]                                     - Between 0 and 1.
4:
5:  REPEAT N TIMES
6:     PRINT password                                 ← Leak current password.
7:
8:                                                    ↓ Internal choice of new password.
9:                          password := ( IF password=0 THEN 1 p⊕ 0 ELSE 0 q⊕ 1)
10: END.
```

The algorithm shown models a password that is changed repeatedly, this process is done probabilistically based on its current value. Which is something already suggested by Mardziel et al. [2014b]. A question that may be asked is what would happen to the adversarial knowledge after multiple runs. Intuitively, that the adversary would eventually learn about the values of p and q after observing many runs.

This initial thought can be expanded if we create an example for it, thus let us work on that. Imagine we have three different strategies for generating the password:

- random: A new password is generated uniformly. ($p = q = 1/2$).
- not-1: A new password is generated uniformly if its current value is 0, but if it is 1, change it to 0. ($p = 1/2, q = 1$).
- alternate - If the current value is 0, set it to 1, and vice versa. ($p = q = 1$).

We can represent this strategy as follows. First, we need to state that both the current value of the password and the strategy that is being used are represented as pairs - for instance r0 means that the strategy is random and the current value is 0, n1 represents strategy not-1 and the current value 1, and so on. We list each (strategy,password) pair as follows:

- r0: strategy random and current value is 0.
- r1: strategy random and current value is 1.
- n0: strategy not-1 and current value is 0.
- n1: strategy not-1 and current value is 1.
- a0: strategy alternate and current value is 0.
- a1: strategy alternate and current value is 1.

Then we can build the following matrices to represent the adversarial knowledge about the system. We build a markov where each row and each column represent a pair (strategy,secret) this is responsible to update the secret and is depicted as follows:

M	$r0$	$r1$	$n0$	$n1$	$a0$	$a1$
$r0$	$1/2$	$1/2$	0	0	0	0
$r1$	$1/2$	$1/2$	0	0	0	0
$n0$	0	0	$1/2$	$1/2$	0	0
$n1$	0	0	1	0	0	0
$a0$	0	0	0	0	0	1
$a1$	0	0	0	0	1	0

(4.4)

Here each row represents the current state the world is in, and each column represents the next state that the world can change to. For instance, if we are on $r0$, we can move to $r0$ with $1/2$ probability, or to $r1$ with the same probability. The adversary a priori has no knowledge of which secret is being used. Therefore we can represent it as a simple prior with matrix 4.5.

$r0$	$1/6$
$r1$	$1/6$
$n0$	$1/6$
$n1$	$1/6$
$a0$	$1/6$
$a1$	$1/6$

(4.5)

And the channel is depicted by Algorithm 1 on line 9. Below we can have a full overview of the system created.

$r0$	$1/6$
$r1$	$1/6$
$n0$	$1/6$
$n1$	$1/6$
$a0$	$1/6$
$a1$	$1/6$

$\triangleright$

C	0	1
$r0$	0	1
$r1$	1	0
$n0$	0	1
$n1$	1	0
$a0$	0	1
$a1$	1	0

M	$r0$	$r1$	$n0$	$n1$	$a0$	$a1$
$r0$	$1/2$	$1/2$	0	0	0	0
$r1$	$1/2$	$1/2$	0	0	0	0
$n0$	0	0	$1/2$	$1/2$	0	0
$n1$	0	0	1	0	0	0
$a0$	0	0	0	0	0	1
$a1$	0	0	0	0	1	0

This matrix is the representation of the prior 4.5 that will interact with both the channel C and the markov M. The first step of this approach is to make the traditional QIF interaction of pushing the prior through a channel.

$r0$	$1/6$
$r1$	$1/6$
$n0$	$1/6$
$n1$	$1/6$
$a0$	$1/6$
$a1$	$1/6$

$\triangleright$

C	0	1
$r0$	0	1
$r1$	1	0
$n0$	0	1
$n1$	1	0
$a0$	0	1
$a1$	1	0

$=$

$[\pi \triangleright \mathsf{C}]$

$1/2$	$1/2$
0	$1/3$
$1/3$	0
0	$1/3$
$1/3$	0
0	$1/3$
$1/3$	0

After this, we take the resulting hyper and interact it with the markov to update the secret. Section 2.3.2 presents how to make these calculations. This is done by pushing each inner of $[\pi \triangleright C]$ and then combining them weighted by their outer probabilities.

$[\pi \triangleright \mathsf{C}]$

$1/2$	$1/2$
0	$1/3$
$1/3$	0
0	$1/3$
$1/3$	0
0	$1/3$
$1/3$	0

$\triangleright$

M	$r0$	$r1$	$n0$	$n1$	$a0$	$a1$
$r0$	$1/2$	$1/2$	0	0	0	0
$r1$	$1/2$	$1/2$	0	0	0	0
$n0$	0	0	$1/2$	$1/2$	0	0
$n1$	0	0	1	0	0	0
$a0$	0	0	0	0	0	1
$a1$	0	0	0	0	1	0

$=$

$[\pi \triangleright \mathsf{C}|\mathsf{M}]$

$1/2$	$1/2$
$1/6$	$1/6$
$1/6$	$1/6$
$1/3$	$1/6$
0	$1/6$
$1/3$	0
0	$1/3$

Then we repeat the process on the second iteration, this time we push a hyper through the channel. This is done by pushing each column and then weighting it by its outer. This is the direct update from Section 3.1.1.

Δ

$1/2$	$1/2$
$1/6$	$1/6$
$1/6$	$1/6$
$1/3$	$1/6$
0	$1/6$
$1/3$	0
0	$1/3$

$\triangleright$

C	0	1
$r0$	0	1
$r1$	1	0
$n0$	0	1
$n1$	1	0
$a0$	0	1
$a1$	1	0

$=$

$[\Delta \triangleright \mathsf{C}]$

$1/12$	$5/12$	$1/3$	$1/6$
0	$1/5$	0	$1/2$
1	0	$1/4$	0
0	$2/5$	0	$1/2$
0	0	$1/4$	0
0	$2/5$	0	0
0	0	$1/2$	0

And finally, we make the final push of the resulting hyper $[\Delta \triangleright C]$ through the Markov, again with using the intermediate hypers described on Section 2.3.2.

$[\Delta \triangleright \mathsf{C}]$

$1/12$	$5/12$	$1/3$	$1/6$
0	$1/5$	0	$1/2$
1	0	$1/4$	0
0	$2/5$	0	$1/2$
0	0	$1/4$	0
0	$2/5$	0	0
0	0	$1/2$	0

$\triangleright$

m	$r0$	$r1$	$n0$	$n1$	$a0$	$a1$
$r0$	$1/2$	$1/2$	0	0	0	0
$r1$	$1/2$	$1/2$	0	0	0	0
$n0$	0	0	$1/2$	$1/2$	0	0
$n1$	0	0	1	0	0	0
$a0$	0	0	0	0	0	1
$a1$	0	0	0	0	1	0

$=$

$[\Delta \triangleright \mathsf{C}|\mathsf{M}]$

$1/12$	$5/12$	$1/3$	$1/6$
$1/2$	$1/10$	$1/8$	$1/4$
$1/2$	$1/10$	$1/8$	$1/4$
0	$1/5$	$1/4$	$1/4$
0	$1/5$	0	$1/4$
0	0	$1/2$	0
0	$2/5$	0	0

Now we can make calculations for vulnerability and verify if this result can be valuable. Here we use the default Bayes vulnerability. Since our prior is a uniform distribution with $1/6$ probability, $V_g(\pi) = 1/6$. For the posterior case, we have that

$$V_g\,[\Delta \triangleright \mathsf{C}|\mathsf{M}] = 1/12 \times 1/2 + 5/12 \times 2/5 + 1/3 \times 1/2 + 1/6 \times 1/4 = 5/12.$$

If we calculate the leakage we have $\mathcal{L}_g^{\times}(\pi, \mathsf{C}|\mathsf{M}) = 5/12 \div 1/6 = 5/2$. With this result, we can conclude that the adversary would eventually increase his knowledge of the secret

over multiple runs. But despite the promising results we attained, this approach is not sensible enough. Since it fails to grasp big differences between similar distributions. The matrices presented below are one of these scenarios.

$$\begin{array}{c|cc} & \multicolumn{2}{c}{\textit{ember}} \\ & \sigma^1 & \sigma^2 \\ \hline x_1 & {}^{99}/_{100} & {}^{98}/_{100} \\ x_2 & {}^{1}/_{100} & {}^{2}/_{100} \\ \hline & {}^{1}/_{2} & {}^{1}/_{2} \end{array} \quad , \quad \begin{array}{c|cc} & \multicolumn{2}{c}{\textit{huskar}} \\ & \sigma^1 & \sigma^2 \\ \hline x_1 & 1 & 0 \\ x_2 & 0 & 1 \\ \hline & {}^{1}/_{2} & {}^{1}/_{2} \end{array} \quad . \tag{4.6}$$

We can ask ourselves "what is the prior vulnerability on each of the scenarios presented above?". This can be verified with simple calculations with the metrics defined by Alvim et al. [2017a].

$$V(X \mid \textit{ember}) = {}^{99}/_{100} \times {}^{1}/_{2} + {}^{98}/_{100} \times {}^{1}/_{2} = {}^{197}/_{200}.$$

$$V(X \mid \textit{huskar}) = 1 \times {}^{1}/_{2} + 1 \times {}^{1}/_{2} = 1.$$

With *ember*, the strategies differ minimally and the secret will be x_1 most of the time, despite which strategy is chosen. On *huskar* we always get x_1 for σ^1, and x_2 for σ^2. But if we only look at the results, the values for vulnerability are greatly similar. Thus we need a better way to measure a "distance" between strategies. This will allow our results to properly judge similar, but fundamentally different strategies.

Another important point is that if we look only at the outer, the vulnerability could be ${}^{1}/_{2}$ in both cases. This is also very misleading.

The idea that any metric for posterior vulnerability had to be sensible enough to capture subtle differences like the ones presented in matrix 4.6 were *sine qua non* on how we approached our next attempts, Sections 4.5, and 4.6.

4.5 Refined attempt of McIver and Morgan using Manhattan Distance as a gain function

Given what we learned so far, there are a few ways to move forward. One promising idea is to combine the structure already present in the literature and what we learned with 4.4, that is, measure distances between strategies.

The best way to measure the distance between strategies is to define a specific gain function to them. With this gain function defined, we can calculate her gains and the vulnerabilities associated. Given a set of strategies, the adversary can guess which strategy is being used. Her gain will be determined considering the distance between the strategy she chose and the actual strategy.

On this approach we go back to representing secrets as simple values, instead of pairs as we did on Section 4.4.

4.5.1 Finding a suitable gain function

The adversarial guesses here are the strategies, which are probability distributions. Thus a suitable gain function needs to be able to measure the distance between probability distributions. Literature presents us with many ways to measure this type of distance. Examples range from Rényi entropy [Rényi, 1961], Kullback-Liebler divergence [Kullback and Leibler, 1951], Hellinger distance [Cieslak et al., 2012], Bhattacharyya distance [Bhattacharya, 1946], etc.. Some of these suit generic cases and can be broadly applied, others match more particular cases that need to meet certain criteria. We tried a few of these functions and ended up choosing Manhattan distance. This function was chosen because it is easy to understand. It also achieves the results we aimed for vulnerabilities and leakage.

There are a few other ways to measure this distance without logarithmic functions. Two of the most known are total variation[Devroye et al., 1990] and Manhattan distance [Krause, 1973][1]. We made calculations with both distances but we found out that would be better to work with the Manhattan distance only. This is because total variation distance works with a supremum, which causes problems with the optimization that may be used on future developments of the project, Chapter 6.

Definition 4.6 (Manhattan distance)**.** *The* Manhattan distance *between two distributions* $\alpha : \mathbb{D}\mathcal{X}$ *and* $\beta : \mathbb{D}\mathcal{X}$ *of probability given by*

$$mh(\alpha, \beta) := \sum_{p \in \alpha, \beta} |\alpha(p) - \beta(p)|.$$

Having chosen a function to measure the distance, we can now take this to the QIF framework and define a gain function associated with it.

Definition 4.7 (Manhattan distance gain function)**.** *The* Manhattan distance gain function $g_{mh} : \mathbb{D}\mathcal{X} \times \mathbb{D}\mathcal{X} \mapsto 0, 1$ *is given by*

$$g_{mh}(\alpha, \beta) := \frac{2 - mh(\alpha, \beta)}{2}.$$

The thought behind this definition is very intuitive. As any simple metric we have a subtraction of the maximum possible adversarial gain and the distance between the

[1] This distance is also known as Taxicab Geometry or L_1 distance, here we will treat is as Manhattan Distance

strategy chosen by the adversary and the actual strategy. The only tweak that had to be made is to change the subtraction from 1, intuitive, to 2. This is because on the worst-case scenario the Manhattan distance would be 2. Giving us negative values if we kept 1.

With the gain function, Definition 4.7, we can define the prior and posterior vulnerabilities specific to this case.

Attempt 4.8 (Prior Manhattan distance strategy vulnerability)**.** *Given the Manhattan distance gain function 4.7, a strategy $\sigma \in \mathbb{D}\mathcal{X}$, a hyper $\Pi \in \mathbb{D}^2\mathcal{X}$, and a set of guesses $w \in \mathcal{W}$. The prior Manhattan distance strategy vulnerability is defined as*

$$V_{g_{mh}}(S|\Pi) \quad = \quad \max_{w} \sum_{\sigma} \Pi(\sigma) g_{mh}(w, \sigma).$$

Attempt 4.9 (Posterior Manhattan distance strategy vulnerability)**.** *Given the Manhattan distance gain function 4.7, given a strategy $\sigma \in \mathbb{D}\mathcal{X}$, given a hyper $\Pi \in \mathbb{D}^2\mathcal{X}$, given a set of guesses $w \in \mathcal{W}$, and given an environment-through-channel $\{\Pi \triangleright C\} \in \mathbb{D}^3\mathcal{X}$. The posterior Manhattan distance strategy vulnerability is defined as*

$$V_{g_{mh}}(S \mid \Pi \triangleright C) \quad = \quad \sum_{i} a^i \max_{w} \sum_{\sigma} \{\Pi \triangleright C\}\,(\sigma) g_{mh}(w, \sigma).$$

Where a^i represents the i'th outer of the $\{\Pi \triangleright C\}$.

It is important to notice here that we redefined only the strategy vulnerability. This is because using a gain function that compares probability distributions makes little sense when calculating secret vulnerabilities. For this part we keep Definitions 2.24 and 4.2. We can verify through Example 4.4 that they respected monotonicity and had valid results for leakage.

With these definitions made, we can move to an experimental phase to verify if they have interesting results that can be further expanded.

4.5.2 Testing the previous metrics with g_{mh}

After defining the new gain function, we can run sanity checks to verify if it respects the basic properties we need. This was not the case. Results did not follow the intuition they should, which made this simple definition not adequate. Before we present our example there is one important point to be made about the set of guesses $\mathcal{W}$.

To make the proper calculations we would have to consider every possible guess. This would be attained through linear programming, as presented by Alvim et al. [2020,

chap. 5.5.3]. But we can *restrict* the set of guesses to $\mathcal{W} = \{\pi\} \cup \{\pi^i | \sigma^i \in \lceil \Pi \rceil\}$ and still attain meaningful results.

Now we present a few interesting results of sanity checks.

Example 4.10 (Calculating $V_{g_{mh}}(S \,|\, \Pi \triangleright C)$ with g_{mh})**.** *Let us think about the following matrix which presents us with two environments jakiro and lich. These environments will be pushed through the identity channel I.*

	σ^1	σ^2	σ^3
x_1	1	0	$1/2$
x_2	0	1	$1/2$
jakiro	$1/2$	$1/2$	0
lich	0	0	1

$\triangleright$ I =

1	0
0	1

.

Here we have that the set of guesses $\mathcal{W}$, following our restriction, is $\mathcal{W} = \{(1,0),(0,1),(1/2,1/2)$ which we will refer to as w_1, w_2, and w_3 respectively. Note that $\pi = w_3$. Now we can use Definition 4.8 to compute the prior Manhattan Distance Vulnerability. For jakiro we have

$$\begin{aligned}
V_{g_{mh}}(S \,|\, jakiro) &= \max_i \Big(\sum_j jakiro(w_j)\, g_{mh}(w_i, w_j)\Big) \\
&= \max(1/2 \times g(w_1,w_1) + 1/2 \times g(w_1,w_2) + 0 \times g(w_1,w_3); \\
&\qquad 1/2 \times g(w_2,w_1) + 1/2 \times g(w_2,w_2) + 0 \times g(w_2,w_3); \\
&\qquad 1/2 \times g(w_3,w_1) + 1/2 \times g(w_3,w_2) + 0 \times g(w_3,w_3)) \\
&= \max(1/2 \times 2 + 1/2 \times 0; 1/2 \times 0 + 1/2 \times 2; 1/2 \times 1 + 1/2 \times 1) \\
&= \max(1;1;1) \\
&= 1,
\end{aligned}$$

which is intuitive because the entirety of the uncertainty is within the strategy, thus the value of vulnerability is low. For lich we have

$$\begin{aligned}
V_{g_{mh}}(S \,|\, lich) &= \max_i \Big(\sum_j lich(w_j)\, g_{mh}(w_i, w_j)\Big) \\
&= \max(0 \times g(w_1,w_1) + 0 \times g(w_1,w_2) + 1 \times g(w_1,w_3); \\
&\qquad 0 \times g(w_2,w_1) + 0 \times g(w_2,w_2) + 1 \times g(w_2,w_3); \\
&\qquad 0 \times g(w_3,w_1) + 0 \times g(w_3,w_2) + 1 \times g(w_3,w_3)) \\
&= \max(1 \times 1; 1 \times 1; 1 \times 2) \\
&= \max(1;1;2) \\
&= 2,
\end{aligned}$$

which is intuitive because the strategy is already known, making the value for vulnerability high. The resulting hyper for each of these environments is represented as follows:

$$jakiro \quad = \quad lich \quad = \quad \begin{array}{c|c|c|c} & \Delta^1 & \Delta^2 & \Delta^3 \\ & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 1/2 \;\; 1/2 \\ \hline 1 & 1 & 0 & 1 \;\; 0 \\ 0 & 0 & 1 & 0 \;\; 1 \end{array} \quad .$$

Which can be squashed to be represented as

$$jakiro \quad = \quad lich \quad = \quad \begin{array}{c|cc} & 1/2 & 1/2 \\ \hline 1 & 1 & 0 \\ 0 & 0 & 1 \end{array} \quad , \qquad .$$

Now we can use the new definitions for posterior manhattan distance vulnerability 4.9 to make the proper calculations. Since both environments result in the same hyper, the calculation is the same.

$$\begin{aligned} V_{g_{mh}}(S \,|\, jakiro \triangleright I) &= V_{g_{mh}}(S \,|\, lich \triangleright I) \\ &= \sum_i \delta^i \max_w \sum_s \Delta(\sigma) g_{mh}(w, \sigma) \\ &= \sum_i \delta^i V_{g_{mh}}(\Delta^i) \\ &= (0 \times V_{g_{mh}}(\Delta^1) + 0 \times V_{g_{mh}}(\Delta^2) + 1 \times V_{g_{mh}}(\Delta^3) \\ &= V_{g_{mh}}(\Delta^3) \\ &= \max_i (\sum_j a_{\Delta^3} \, g_{mh}(w_i, w_j)) \\ &= \max(1/2 \times g(w_1, w_1) + 1/2 \times g(w_1, w_2); \\ &\qquad 1/2 \times g(w_2, w_1) + 1/2 \times g(w_2, w_2); \\ &\qquad 1/2 \times g(w_3, w_1) + 1/2 \times g(w_3, w_2)) \\ &= \max(1/2 \times 2 + 1/2 \times 0; 1/2 \times 0 + 1/2 \times 2; 1/2 \times 1 + 1/2 \times 1) \\ &= \max(1; 1; 1) \\ &= 1, \end{aligned}$$

And then we verify the leakage for each case

$$\mathcal{L}^{\times}_{g_{mh}}(jakiro, I) \quad = \quad \frac{1}{1} \quad = \quad 1 \qquad and \qquad \mathcal{L}^{\times}_{g_{mh}}(lich, I) \quad = \quad \frac{1}{2} \quad = \quad \frac{1}{2}.$$

From which we can see incoherent values. First and more importantly it violates the property of monotonicity 2.17 on the calculation of $\mathcal{L}^{\times}_{g_{mh}}(lich, I)$*, giving us a leakage smaller than 1. The expected value for* $\mathcal{L}^{\times}_{g_{mh}}(jakiro, I)$ *should also be higher than 1, because at first the adversary does not know the strategy, and after pushing the prior through the channel he learns exactly what the strategy is.*

△▽

Although we attained inconclusive results with this attempt, defining a new gain function made us touch on the aspect of being able to differentiate between similar and distinct strategies. This aspect is pivotal in defining a proper metric for vulnerabilities within different strategies. This paved the way to our next definition that will be presented in Section 4.6 and our final definition presented in Chapter 5.

4.6 Attempt through the generation of a prior from the joint matrix

Our final attempt was a hybrid version. It combined ideas from previous attempts. Here we have the idea presented in Section 4.4 that makes the secrets pairs of strategies and secrets. In other words we will use the joint $\mathbb{D}(\mathcal{S} \times \mathcal{X})$ as the prior. We also use the idea of a distinct gain function, presented in Section 4.5. This helps us measure vulnerabilities with more accuracy. We will present an example throughout this section that can help create the intuition behind the formulation. Illustrating each step along the way.

Firstly we can look at Figure 4.2 to get a sense of how our formulation works. We first have a prior hyper $\Pi : \mathbb{D}^2\mathcal{X}$. This prior has a set $\mathcal{S}$ of strategies and a set $\mathcal{X}$ secrets. Π is then transformed into a $Joint : \mathbb{D}(\mathcal{S} \times \mathcal{X})$ distribution, combining $\mathcal{S}$ and $\mathcal{X}$ in pairs. This *Joint* will then be pushed through a channel that is adjusted accordingly to interact with it. This "pushing" works like in traditional QIF and results on $\Delta : \mathbb{D}^2 y$. This Δ then is marginalized to receive the calculations for the respective attributes: $\mathcal{S}$ and $\mathcal{X}$.

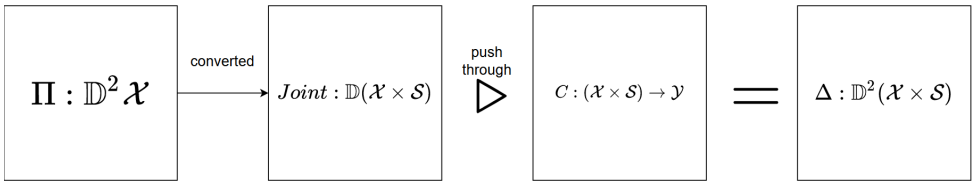

Figure 4.2: Graphical view of the joint approach.

$$mars \quad = \quad \begin{array}{c|cc} & \sigma^1 & \sigma^2 \\ & 1/2 & 1/2 \\ \hline x_1 & 1/2 & 0 \\ x_2 & 1/2 & 0 \\ x_3 & 0 & 1/2 \\ x_4 & 0 & 1/2 \end{array} \tag{4.7}$$

Consider the environment $mars$, presented by matrix 4.7. This environment has strategies $\mathcal{S} = \{\sigma^1, \sigma^2\}$ and secrets $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$. We will use this object to make the measurements for prior strategy vulnerability, taking into account the Manhattan distance gain function 4.7. We define it as follows.

Attempt 4.11 (Prior strategy vulnerability with a prior joint)**.** *Let $\Pi : \mathbb{D}^2\mathcal{X}$ be a hyper on $\mathcal{X}$ such that $\Pi = \sum_i a_i[\sigma^i]$. Where each $\delta^i : \mathbb{D}\mathcal{X}$ is an inner distribution and each a_i is the associated outer distribution over the inners. Also let $\mathcal{W}$ be the set of guesses consisting of each δ^i and π^Π.*
The prior strategy vulnerability, written $V_{g_{mh}}(S|\Pi)$, is defined as

$$V_{g_{mh}}(S|\Pi) := \max_{w \in W} \sum_{i \in |a|} \Pi_{\delta^i} g_{mh}(w, \delta^i),$$

where Π_{δ^i} is the probability of δ^i according to Π.

Since we are using the Manhattan distance gain function 4.7, the same rules for the set of guesses apply here. The restricted set is

$$\mathcal{W}_{Elf} = \{(1/2, 1/2, 0, 0); (0, 0, 1/2, 1/2); (1/4, 1/4, 1/4, 1/4)\} \tag{4.8}$$

With the set of guesses we can calculate the prior strategy vulnerability.

$$\begin{aligned} V_{g_{mh}}(S|mars) &= \max_i (\sum_j mars(w_j)\, g_{mh}(w_i, w_j)) \\ &= \max(1/2 \times g(w_1, w_1) + 1/2 \times g(w_1, w_2); \\ &\qquad 1/2 \times g(w_2, w_1) + 1/2 \times g(w_2, w_2); \\ &\qquad 1/2 \times g(w_3, w_1) + 1/2 \times g(w_3, w_2))) \\ &= \max(1/2 \times 2 + 1/2 \times 0; 1/2 \times 0 + 1/2 \times 2; 1/2 \times 1 + 1/2 \times 1) \\ &= \max(1; 1; 1) \\ &= 1, \end{aligned} \tag{4.9}$$

For the prior secret vulnerability, we take the literature approach defined by Alvim et al. [2017a], which we presented on Definition 2.24. The calculations for each vulnerability follow.

$$\begin{aligned} V_{g_{mh}}(X|mars) &= \mathop{\mathbb{E}}_{mars} V_g. \\ &= 1/2 V_1(\sigma^1) + 1/2 V_1(\sigma^2) \\ &= 1/2 \times 1/2 + 1/2 \times 1/2 \\ &= 1/2 \end{aligned} \tag{4.10}$$

Now we take the idea from Section 4.4 and combine each strategy and secret into pairs. This is done by simply multiplying its probabilities. The hyper representing the prior, matrix 4.7, then becomes a simple probability distribution, shown on matrix 4.11. The result of this operation represents the joint distribution of secrets and strategies.

$$\sigma^{mars} \quad = \quad \begin{array}{|c|c|} \hline \sigma^1 x_1 & 1/4 \\ \sigma^1 x_2 & 1/4 \\ \sigma^1 x_3 & 0 \\ \sigma^1 x_4 & 0 \\ \sigma^2 x_1 & 0 \\ \sigma^2 x_2 & 0 \\ \sigma^2 x_3 & 1/4 \\ \sigma^2 x_4 & 1/4 \\ \hline \end{array} \tag{4.11}$$

Prior σ^{mars} is ready to be pushed through a channel. In this example, we are using channel D, but to take the adapted prior, we have to double the rows of D, constructing channel DD. We can see this transformation on matrix 4.12. This is the indirect update that we presented on Section 3.1.2.

$$D \quad = \quad \begin{array}{|c|c|} \hline 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \hline \end{array} \quad , \qquad DD \quad = \quad \begin{array}{|c|c|} \hline 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \hline \end{array} \tag{4.12}$$

The result of pushing is then

$$[\sigma^{Elf} \triangleright DD] \quad = \quad \begin{array}{|c|c|c|} \hline & 1/2 & 1/2 \\ \hline \sigma^1 x_1 & 1/2 & 0 \\ \sigma^1 x_2 & 1/2 & 0 \\ \sigma^1 x_3 & 0 & 0 \\ \sigma^1 x_4 & 0 & 0 \\ \sigma^2 x_1 & 0 & 0 \\ \sigma^2 x_2 & 0 & 0 \\ \sigma^2 x_3 & 0 & 1/2 \\ \sigma^2 x_4 & 0 & 1/2 \\ \hline \end{array} \quad . \tag{4.13}$$

$[\sigma^{mars} \triangleright DD]$, in this case, is what we called previously Δ. So now we have to make the dismembering for each attribute: Strategies and Secrets. This is done by a simple marginalization.

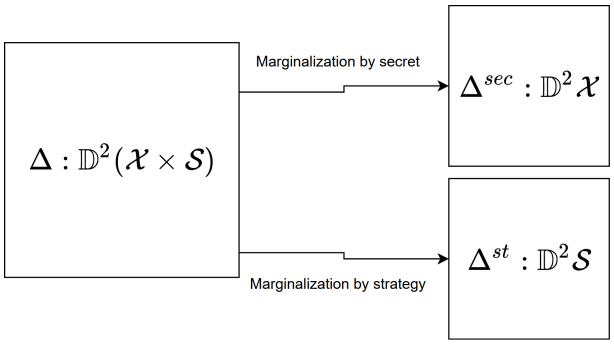

Figure 4.3: Dismembration of Δ.

The resulting marginalization on strategies is

$$[\sigma^{mars} \triangleright DD]^{St} \quad = \quad \begin{array}{|c|c|} \hline 1/2 & 1/2 \\ \hline 1 & 0 \\ 0 & 1 \\ \hline \end{array} \quad , \tag{4.14}$$

and the marginalization for secrets is

$$[\sigma^{mars} \triangleright DD]^{Sec} \quad = \quad \begin{array}{|c|c|} \hline 1/2 & 1/2 \\ \hline 1/2 & 0 \\ 1/2 & 0 \\ 0 & 1/2 \\ 0 & 1/2 \\ \hline \end{array} \quad . \tag{4.15}$$

In each matrix 4.14 and 4.15 we can now calculate the posterior strategy vulnerability. We define it as follows.

Definition 4.12 (Posterior strategy vulnerability with a prior joint)**.** *Let* $[\Pi \triangleright C]_{st}$ *be the resulting hyper of pushing* Π *through* C*. Let* $[\Pi \triangleright C]_{st}$ *be the marginalization on strategies of such hyper. Where each* $\gamma^i : \mathbb{D}\mathcal{Y}$ *is an inner distribution and each* b_i *is the associated outer probability.*
The posterior strategy vulnerability, written $V_{g_{mh}}(S|\Pi \triangleright C)$*, is defined as*

$$V_{g_{mh}}(S|\Pi \triangleright C) := \sum_{y \in \mathcal{Y}} b_y \times \max_{w \in W} \sum_{i \in |a|} \gamma^i g_{mh}(w, \delta^i)$$

After defining both prior and posterior vulnerabilities, it is only natural that we follow with the leakage definition. This definition is straightforward and is very similar to the other types of leakage previously defined in 2.16. It is defined as follows:

Definition 4.13 (Multiplicative strategy leakage and additive strategy leakage)**.** *Let* $\Pi : \mathbb{D}^2\mathcal{X}$ *be a hyper on* $\mathcal{X}$*, let* $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ *be a channel, the multiplicative strategy leakage is defined as:*

$$\mathcal{L}^{\times}_{g_{mh}}(\Pi, C) \quad := \quad \frac{V_{g_{mh}}(S|\Pi \triangleright C)}{V_{g_{mh}}(S|\Pi)}$$

and the additive strategy leakage is defined as:

$$\mathcal{L}^{+}_{g_{mh}}(\Pi, C) \quad := \quad V_{g_{mh}}(S|\Pi \triangleright C) - V_{g_{mh}}(S|\Pi).$$

We already have the value for prior strategy vulnerability from Equation 4.9. With the prior strategy vulnerability in hands, we now look to 4.14 and calculate the posterior strategy vulnerability using Definition 4.12.

$$\begin{aligned} V^{st}_{g_{mh}}[mars \triangleright II] &= \sum_{y \in \mathcal{Y}} b_y \times \max_{w \in W} \sum_{i \in |a|} \gamma^i g_{mh}(w, \delta^i) \\ &= 2. \end{aligned}$$

And now we can verify the leakage using Definition 4.13.

$$\mathcal{L}^{+}_{st}(mars, II)_{g_{mh}} \quad = \quad \frac{2}{1} \quad = \quad 2 \qquad \textit{and} \qquad \mathcal{L}^{\times}_{st}(mars, II)_{g_{mh}} \quad = \quad 2 - 1 \quad = \quad 1. \tag{4.16}$$

There was a big problem with our results. Throughout many tests and experiments, we realized that, as represented by the results from 4.16, the result for $\mathcal{L}^{\times}_{st}(\Pi, C)_{g_{mh}}$ would always result in 2. We ran over five hundred sanity checks. They varied in sample size, channel type, prior type, and different restrictions for the set of guesses. With all of them resulting in the same value.

This made us realize that this formulation was correct but not ideal. We were missing something. The problem was that the adversary had too much power, he always had unrestricted knowledge. Which allowed him to make the most informed decision always, thus he would not gain any new information despite the pushing through the channel. This made us look back to Alvim et al. [2017a, sec. 5]. With this kind of knowledge, the leakage had to be maximum. Therefore we had to improve the model on how he could perceive the environment. This was done using aggregation matrices, which we presented in Section 2.2.3.

Chapter 5

Final definition of posterior strategy and secret vulnerabilities given a model

In this chapter we discuss the main contribution of this thesis: A formulation for posterior vulnerabilities given a model for both secrets and strategies. This formulation allows us to define leakage for the scenario where a hyper is used to represent adversarial prior knowledge [Alvim et al., 2017a]. This chapter addresses the objective proposed in Section 1.1.2.

In Section 5.1 we define our metrics for posterior strategy vulnerability and posterior secret vulnerability. Section 5.2 presents the operational interpretation for the metrics. We also present a proof for monotonicity in Section 5.3. Finally, in Section 5.4 we make comparisons with previous examples to show the validity of our formulation.

5.1 Final definition using aggregation matrices

The previous attempts helped us define a measure well suited for posterior strategy vulnerability. With our first attempt, Section 4.2, we reused definitions proposed by Alvim et al. [2017a]. Recall that it showed us that the resulting element from pushing an environment through a channel is not necessarily an environment, in the sense that it does not necessarily reflect how a secret is generated, although it depicts the adversarial state of knowledge. Given that a new definition was needed.

Our second attempt, presented on Section 4.3, showed us that we have to be careful with said redefinition, because environments and hypers do not represent the same objects. For instance, environments necessarily have type $\mathbb{D}^2\mathcal{X}$, because by definition generate secrets. Hypers can have different types, this is because they can represent many types of knowledge related or unrelated to the secret. Our third attempt, Section 4.4, showed us that only obtaining values respecting monotonicity is not enough: we also have to respect distances between strategies. This idea was explored on our fourth attempt, Section 4.5, which was not sensitive enough, even with the hybrid approach from Section 4.6. This

was because we were missing an important point: *models of partial knowledge*.

Alvim et al. [2017a] introduced the concepts of strategy and the prior vulnerabilities related to it. It also introduced aggregation matrices and models, presented on Section 2.2.3. Their work defined that the adversary only knew a model M, and not necessarily the entire environment Π. Vulnerability was shown to be behave similarly if a model of partial knowledge was introduced on the framework. There were two special cases. The first was when $M = \Pi$, that is if the aggregation matrix was the identity matrix. This represented the adversary with unabridged knowledge, which corresponds to the case in which an adversary can discern the exact strategy generating the secret. The second case was when $M = \pi$, which happened when A was the null channel, Definition 2.11. This case coincides with the traditional QIF approach, in which the adversary knows only one possible strategy of generating secrets.

Alvim et al. [2017a] did not define how to push an environment Π through a channel. We developed this idea on Section 3.1. It is important to define a measure for posterior vulnerability and leakage. A very intuitive way to think is to take the squashed product of pushing Π through C and defining

$$V(X|[M \triangleright C]) \quad = \quad \mathop{\mathbb{E}}_{[M \triangleright C]} V$$

and use Alvim et al. [2017a] method of defining strategy leakage as

$$V(S|[M \triangleright C]) \quad = \quad \frac{V(X|[M \triangleright C])}{V(X|[\pi \triangleright C])}$$

but this can be shown not to work when comparing both leakages, assuming we are using multiplicative leakage:

$$\begin{aligned} \frac{V(X|[M \triangleright C])}{V(X|M)} &= \frac{V(S|[M \triangleright C])}{V(S|M)} \cdot \frac{V(X|[\Pi \triangleright C])}{V(X|\Pi)} \\ &= \frac{V(X|[M \triangleright C])}{V(X|\Pi \triangleright C])} \cdot \frac{V(X|\Pi)}{V(X|M)} \cdot \frac{V(X|[\Pi \triangleright C])}{V(X|\Pi)} \end{aligned} \tag{5.1}$$

because this means that strategy leakage will always be ≤ 1. Since using the perceived leakage equation 2.13, we have that

$$\underbrace{\text{Perceived Leakage}}_{\text{constant}} = \underbrace{\text{Strategy Leakage}}_{\leq 1} \times \underbrace{\text{Secret Leakage}}_{\geq 1}.$$

with all of this in mind we can redefine the prior cases for strategy and secret vulnerability as follows.

Definition 5.1 (Prior secret vulnerability given a model)**.** *Let* $\Pi : \mathbb{D}^2\mathcal{X}$*, where each* $\sigma^i : \mathbb{D}\mathcal{X}$ *represents a possible strategy within* Π*. Let* $M : \mathbb{D}^2\mathcal{X}$ *be a model of adversarial knowledge consistent with* Π*, let* $\mathcal{X}$ *be the set of secrets, let* $\mathcal{W}$ *be the set of guesses, and let* $g : \mathcal{X} \times \mathcal{W} \to \mathbb{R}$ *be a gain function. The* prior secret vulnerability given a model*, written* $V(X \mid M)$*, is defined as:*

$$V(X \mid M) \quad := \quad \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} M_{\sigma'} \sigma'_x g(w, x),$$

where σ' *represents the possible strategies on* M *mapped by an aggregation matrix* A*.*

The intuition behind Definition 5.1 is to assume that the representation for the adversarial knowledge is the model, and thus choose the best guess given the distributions represented by it.

Definition 5.2 (Prior strategy vulnerability given a model)**.** *Let* $\Pi : \mathbb{D}^2\mathcal{X}$*, where each* $\sigma^i : \mathbb{D}\mathcal{X}$ *represents a possible strategy within* Π*. Let* $A : \mathcal{S} \times \mathcal{S}'$ *be an aggregation matrix. Let* $M : \mathbb{D}^2\mathcal{X}$ *be a model of adversarial knowledge obtained by applying* A *to* Π*, let* $\mathcal{X}$ *be the set of secrets, let* $\mathcal{W}$ *be the set of guesses, and let* $g : \mathcal{X} \times \mathcal{W} \to \mathbb{R}$ *be a gain function. The* prior strategy vulnerability given a model*, written* $V(S \mid M)$*, is defined as:*

$$V(S \mid M) \quad := \quad \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma A(\sigma' | \sigma) g_{mh}(w, \sigma)$$

where σ' *represents the possible strategies on* A *which can be mapped to a model* $M : \mathbb{D}^2\mathcal{X}$*.*

To better understand all the calculations, we discuss on the examples of this section how the calculations for these vulnerabilities are made. Example 5.3 presents the context and the calculations for the prior vulnerabilities. It is important to notice that for these tests we restrict the set of guesses to $\mathcal{W} = \mathcal{X}$ in the case of secret vulnerability, and to $\mathcal{W} = \{\pi\} \cup \{\pi^i | \sigma^i \in \lceil \Pi \rceil\} \cup \{\sigma^j \in M\}$ in the case of strategy vulnerabilities.

Example 5.3 (*invoker* Example. Part I)**.** *Suppose we have an environment invoker*

$$invoker \quad = \quad \begin{array}{|c|c|c|} \hline & \sigma^1 & \sigma^2 \\ \hline & {}^9\!/_{10} & {}^1\!/_{10} \\ \hline x_1 & 1 & 0 \\ \hline x_2 & 0 & 1 \\ \hline \end{array} \quad .$$

Suppose the adversarial knowledge is achieved through the aggregation matrix A

$$A \quad = \quad \begin{array}{c|c} & \sigma^A \\ \hline x_1 & 1 \\ x_2 & 1 \end{array} \quad ,$$

The model for the adversarial knowledge is then achieved by $M = \Pi \cdot A$

$$\begin{array}{c|cc} & \sigma^1 & \sigma^2 \\ & 9/10 & 1/10 \\ \hline x_1 & 1 & 0 \\ x_2 & 0 & 1 \end{array} \quad \cdot \quad \begin{array}{c|c} \sigma^A \\ \hline 1 \\ 1 \\ 1 \end{array} \quad = \quad \begin{array}{c|c} \sigma^A \\ \hline 1 \\ 9/10 \\ 1/10 \end{array}$$

Suppose we now push M *through the following* C

$$invoker \quad = \quad \begin{array}{c|cc} & y_1 & y_2 \\ \hline x_1 & 1 & 0 \\ x_2 & 0 & 1 \end{array} \quad .$$

If we make the calculations we can come that the resulting hyper is

$$[invoker \triangleright C] \quad = \quad \begin{array}{c|cc} & \sigma^1 & \sigma^2 \\ & 9/10 & 1/10 \\ \hline x_1 & 1 & 0 \\ x_2 & 0 & 1 \end{array} \quad .$$

We have in hand all the results of how the knowledge updates when we push invoker through C*. Now we need to know what it represents. Firstly we will make the calculations for the prior vulnerabilities.*

$$\begin{aligned} V(S \mid M) &= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma A(\sigma' | \sigma) g_{mh}(w, \sigma) \\ &= max(w = \sigma^1, w = \sigma^2, w = \pi) \\ &= max(((9/10 \times 1 \times 1)_{\sigma^1} + (1/10 \times 1 \times 0)_{\sigma^2})_{w=\sigma^1}, \\ &\qquad ((9/10 \times 1 \times 0)_{\sigma^1} + (1/10 \times 1 \times 1)_{\sigma^2})_{w=\sigma^2}, \\ &\qquad ((9/10 \times 1 \times 9/10)_{\sigma^1} + (1/10 \times 1 \times 1/10)_{\sigma^2})_{w=\pi}), \\ &= max(9/10, 1/10, 41/50) \\ &= 9/10. \end{aligned}$$

△▽

The definitions for posterior vulnerability for secret and strategy follow closely their prior versions. We only add the interaction with the channel to it. This is very similar to what is done by Alvim et al. [2020, theorem 5.7]. Below we define posterior secret vulnerability given a model.

Definition 5.4 (Posterior secret vulnerability given a model)**.** *Let* $\Pi : \mathbb{D}^2\mathcal{X}$*, where each* $\sigma^i : \mathbb{D}\mathcal{X}$ *represents a possible strategy within* Π*. Let* $M : \mathbb{D}^2\mathcal{X}$ *be a model of adversarial knowledge, let* $\mathcal{X}$ *be the set of secrets, let* $\mathcal{W}$ *be the set of guesses, let* $g : \mathcal{X} \times \mathcal{W} \to \mathbb{R}$ *a gain function, and let* $C : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ *be a channel.* The posterior secret vulnerability given a model M, w.r.t. channel C*, written* $V(X \mid M \triangleright C)$*, is defined as:*

$$V(X \mid M \triangleright C) \quad := \quad \sum_{y \in \mathcal{Y}} \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} M_{\sigma'} \sigma'_x C_{xy} g(w, x)$$

where σ' *represents the possible strategies on* M *mapped by an aggregation matrix* A*.*

This definition may seem complex at first, but it is just the adaptation of Definition 2.30 with the interaction between the strategy generated by the model and the channel. In similar fashion to Definition 5.4, we now define posterior strategy vulnerability given a model.

Definition 5.5 (Posterior strategy vulnerability given a model)**.** *Let* $\Pi : \mathbb{D}^2\mathcal{X}$*, where each* $\sigma^i : \mathbb{D}\mathcal{X}$ *represents a possible strategy within* Π*. Let* $A : \mathcal{S} \times \mathcal{S}'$ *be an aggregation matrix, let* $\mathcal{X}$ *be the set of secrets, let* $\mathcal{W}$ *be the set of guesses, let* $g : \mathcal{X} \times \mathcal{W} \to \mathbb{R}$ *a gain function, and let* $C : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ *be a channel.* The posterior strategy vulnerability given a model M, w.r.t. channel C*, written* $V(S \mid M \triangleright C)$*, is defined as:*

$$V(S \mid M \triangleright C) \quad := \quad \sum_{y \in \mathcal{Y}} \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \sum_{x \in \mathcal{X}} \Pi_\sigma A(\sigma' | \sigma) \sigma^x C_{xy} g_{mh}(w, \sigma)$$

where σ' *represents the possible strategies on* A *which can be mapped to a model* $M : \mathbb{D}^2\mathcal{X}$*.*

It is important to notice here, that we keep the restrictions made on the set of guesses which we presented on Section 4.6. Namely, guesses are $\mathcal{W} = \{\pi\} \cup \{\pi^i | \sigma^i \in \lceil \Pi \rceil\} \cup \{\sigma^j \in M\}$.

Example 5.6 (*invoker* Example. Part II)**.** *With our definition of posteriors in hand, we can now make the calculations for posterior vulnerabilities on the context of Example 5.3.*

$$\begin{aligned} V(S \mid M \triangleright C) &= \sum_{y \in \mathcal{Y}} \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \sum_{x \in \mathcal{X}} \Pi_\sigma A(\sigma' | \sigma) \sigma^x C_{xy} g_{mh}(w, \sigma) \\ &= (\sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \sum_{x \in \mathcal{X}} \Pi_\sigma A(\sigma' | \sigma) \sigma^x C_{xy} g_{mh}(w, \sigma))_{y_1} \end{aligned}$$

$$
\begin{aligned}
&+ \Big(\sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \sum_{x \in \mathcal{X}} \Pi_\sigma A(\sigma'|\sigma) \sigma^x C_{xy} g_{mh}(w, \sigma)\Big)_{y_2} \\
&= \Big(\max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \sum_{x \in \mathcal{X}} \Pi_\sigma A(\sigma'|\sigma) \sigma^x C_{xy} g_{mh}(w, \sigma)\Big)_{y_1} \\
&+ \Big(\max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \sum_{x \in \mathcal{X}} \Pi_\sigma A(\sigma'|\sigma) \sigma^x C_{xy} g_{mh}(w, \sigma)\Big)_{y_2} \\
&= \{\max(w = \sigma^1, w = \sigma^2, w = \pi)\}_{y_1} \\
&+ \{\max(w = \sigma^1, w = \sigma^2, w = \pi)\}_{y_2} \\
&= \{\max([((9/10 \times 1 \times 1 \times 1 \times 1)_{x_1} + (9/10 \times 1 \times 0 \times 0 \times 1)_{x_2})_{\sigma^1} \\
&\qquad + ((1/10 \times 1 \times 0 \times 1 \times 0)_{x_1} + (1/10 \times 1 \times 1 \times 0 \times 0)_{x_2})_{\sigma^2}]_{w=\sigma^1}, \\
&\qquad [9/10 \times 1 \times 1 \times 1 \times 0)_{x_1} + (9/10 \times 1 \times 0 \times 0 \times 0)_{x_2})_{\sigma^1} \\
&\qquad + ((1/10 \times 1 \times 0 \times 1 \times 1)_{x_1} + (1/10 \times 1 \times 1 \times 0 \times 1)_{x_2})_{\sigma^2}]_{w=\sigma^2}, \\
&\qquad [9/10 \times 1 \times 1 \times 1 \times 9/10)_{x_1} + (9/10 \times 1 \times 0 \times 0 \times 9/10)_{x_2})_{\sigma^1} \\
&\qquad + ((1/10 \times 1 \times 0 \times 1 \times 1/10)_{x_1} + (1/10 \times 1 \times 1 \times 0 \times 1/10)_{x_2})_{\sigma^2}]_{w=\pi}\}_{y_1} \\
&+ \{\max([((9/10 \times 1 \times 1 \times 0 \times 1)_{x_1} + (9/10 \times 1 \times 0 \times 1 \times 1)_{x_2})_{\sigma^1} \\
&\qquad + ((1/10 \times 1 \times 0 \times 0 \times 0)_{x_1} + (1/10 \times 1 \times 1 \times 1 \times 0)_{x_2})_{\sigma^2}]_{w=\sigma^1}, \\
&\qquad [9/10 \times 1 \times 0 \times 0 \times 0)_{x_1} + (9/10 \times 1 \times 0 \times 1 \times 0)_{x_2})_{\sigma^1} \\
&\qquad + ((1/10 \times 1 \times 0 \times 0 \times 1)_{x_1} + (1/10 \times 1 \times 1 \times 1 \times 1)_{x_2})_{\sigma^2}]_{w=\sigma^2}, \\
&\qquad [9/10 \times 1 \times 1 \times 0 \times 9/10)_{x_1} + (9/10 \times 1 \times 0 \times 1 \times 9/10)_{x_2})_{\sigma^1} \\
&\qquad + ((1/10 \times 1 \times 0 \times 0 \times 1/10)_{x_1} + (1/10 \times 1 \times 1 \times 1 \times 1/10)_{x_2})_{\sigma^2}]_{w=\pi}\}_{y_2} \\
&= \max(9/10, 0, 81/100)_{y_1} + \max(0, 1/10, 1/100)_{y_2} \\
&= 1.
\end{aligned}
$$

With this we conclude the calculations of vulnerabilities. △▽

There were many attempts to define posterior vulnerability for strategies. This final approach was shown to suit our prerequisites. After the formulation of this measure it is natural to define the leakage related to it. This formulation follows what we did previously, but we show here for completeness.

Definition 5.7 (Secret and strategy leakage)**.** *Let* $\Pi : \mathbb{D}^2\mathcal{X}$ *be a hyper on* $\mathcal{X}$*, let* $M : \mathbb{D}^2\mathcal{X}$ *be a model of adversarial knowledge, let* $\mathsf{C} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ *be a channel, the corresponding* multiplicative strategy leakage *is defined as:*

$$\mathcal{L}(S, M, C, \times) \quad := \quad \frac{V(S \mid M \triangleright C)}{V(S \mid M)} \quad ,$$

the corresponding additive strategy leakage *is defined as:*

$$\mathcal{L}(S, M, C, +) \quad := \quad V(S \mid M \triangleright C) - V(S \mid M) \quad ,$$

the corresponding multiplicative secret leakage *is defined as:*

$$\mathcal{L}(X, M, C, \times) \quad := \quad \frac{V(X \mid M \triangleright C)}{V(X \mid M)} \quad ,$$

and the corresponding additive strategy leakage *is defined as:*

$$\mathcal{L}(X, M, C, +) \quad := \quad V(X \mid M \triangleright C) - V(X \mid M).$$

Example 5.8 (invoker Example. Part II)**.** *Now to finalize the calculations for leakage, we can take the result for prior vulnerability from Example 5.3 and the result for posterior vulnerability from Example 5.6.*

$$\begin{aligned} \mathcal{L}_{invoker}(S, M, C, \times) &= V(S \mid M \triangleright C) \div V(S \mid M) \\ &= (1) \div (9/10) \\ &= 10/9. \end{aligned}$$

This example, differently from our previous formulations, presents a meaningful result. The adversary actually learns about the strategy after the prior is pushed through the channel.

These definitions are a step forward. However they were made with a restriction the set of possible strategy guesses strategies. This is not enough to give a complete analysis of the scenario. This is actually an optimization problem. General formulations for QIF are present on Alvim et al. [2020, chap. 5.5.3]. This framework presents a linear programming model to solve QIF optimization problems. Through this, our formulations can be expanded to unlimited guesses in an optimization problem.

5.2 Operational Interpretation

Every information metric needs to state the meaning behind the number. This is what we mean by operational interpretation. For instance the number Shannon entropy gives means how many "yes or no questions" must be asked by the adversary so she discovers

the secret, Bayes vulnerability means the probability of the adversary guessing the secret with only one guess, etc. Here we will give the operational interpretation of Equation 5.4 and Equation 5.5.

We start by providing a formula for the joint distribution of: a real strategy $\sigma : \mathbb{D}\mathcal{X}$ used to generate secrets, a perceived strategy $\sigma' : \mathbb{D}\mathcal{X}$ the adversary believes is being used to generate the secret, the secret $x : \mathcal{X}$, and the observation $y : \mathcal{Y}$. We can represent the inter-relationship among these elements in our model as follows.

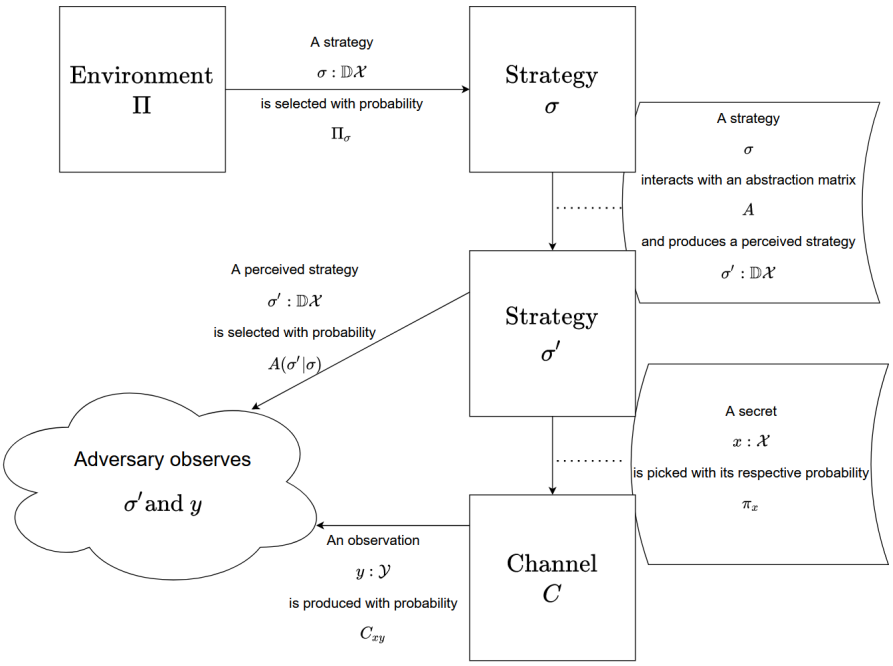

Figure 5.1: Operational significance overview.

Now the joint is given by

$$\begin{aligned} p(\sigma, \sigma', x, y) &= p(\sigma)p(\sigma'|\sigma)p(x|\sigma', \sigma)p(y|x, \sigma', \sigma) && \text{(By the chain rule of probability)} \\ &= \Pi_\sigma A(\sigma'|\sigma)\sigma^x C_{xy} && \text{(By our model. Definition 5.2).} \end{aligned} \tag{5.2}$$

Note that marginals on any combination of σ, σ', x, y can be obtained in the usual way. The result presented by Alvim et al. [2017a] is that $V(X \mid M)$ is the adversary's expected gain if secrets are generated by environment Π, but the adversary believes they are generated by a model M compatible with Π. This is formalized by Proposition 5.9 below

Proposition 5.9 (Operational interpretation of $V(X \mid M)$)**.**

$$V(X \mid M) \quad := \quad \sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma \sum_{\sigma' \in \mathbb{D}\mathcal{X}} A(\sigma'|\sigma) \sum_{x \in \mathcal{X}} \sigma^x g(w^{\sigma'}, x),$$

where $w^{\sigma'} = \underset{w}{argmax} \sum_{x \in \mathcal{X}} \sigma^x g(w^{\sigma'}, x)$.

Proof.

$$\begin{aligned}
\sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma \sum_{\sigma' \in \mathbb{D}\mathcal{X}} A(\sigma'|\sigma) \sum_{x \in \mathcal{X}} \sigma^x g(w^{\sigma'}, x) &= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \sum_{x \in \mathcal{X}} p(\sigma, \sigma', x) g(w^{\sigma'}, x) && \text{(By Equation 5.2)} \\
&= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} M_{\sigma'} \sum_{x \in \mathcal{X}} \sigma'_x g(w^{\sigma'}, x) \sum_{\sigma \in \mathbb{D}\mathcal{X}} p(\sigma|\sigma', x) \\
&= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} M_{\sigma'} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \sigma'_x g(w, x) \\
&= V(X \mid M). && \square
\end{aligned}$$

For $V(S \mid M)$, the operational interpretation is akin to that of $V(X \mid M)$:

Proposition 5.10 (Operational interpretation of $V(S \mid M)$)**.**

$$V(S \mid M) \quad := \quad \sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma \sum_{\sigma' \in \mathbb{D}\mathcal{X}} A(\sigma'|\sigma) g(w^{\sigma'}, \sigma),$$

where $w^{\sigma'} = \underset{w}{argmax} \sum_{\sigma \in \mathbb{D}\mathcal{X}} p(\sigma|\sigma') g(w^{\sigma'}, \sigma)$.

Proof.

$$\begin{aligned}
\sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma \sum_{\sigma' \in \mathbb{D}\mathcal{X}} A(\sigma'|\sigma) g(w^{\sigma'}, \sigma) &= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} p(\sigma, \sigma') g(w^{\sigma'}, \sigma) && \text{(By Equation 5.2)} \\
&= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} p(\sigma') \sum_{\sigma \in \mathbb{D}\mathcal{X}} p(\sigma|\sigma') \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} p(\sigma|\sigma') g(w^{\sigma'}, \sigma) \\
&= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} p(\sigma|\sigma') g(w^{\sigma'}, \sigma) \\
&= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma A(\sigma'|\sigma) g(w, \sigma) \\
&= V(S \mid M). && \square
\end{aligned}$$

The next result shows that when the adversary has concise knowledge, Definition 5.4 recovers the classic definition.

Proposition 5.11. *$V(X \mid \pi \triangleright C)$ collapses into $V_g[\pi \triangleright C]$.*

Proof.

$$\begin{aligned} V(X \mid \pi \triangleright C) &= \sum_{y\in\mathcal{Y}} \sum_{\sigma'\in\mathbb{D}\mathcal{X}} \max_{w\in\mathcal{W}} \sum_{x\in\mathcal{X}} M_{\sigma'}\sigma'_x C_{xy} g(w,x) && \text{(By Definition 5.4)} \\ &= \sum_{y\in\mathcal{Y}} \max_{w\in\mathcal{W}} \sum_{x\in\mathcal{X}} \pi_x C_{xy} g(w,x) && (M_\pi = 1,\ \sigma'_x = \pi_x) \\ &= V_g\left[\pi \triangleright C\right]. && \square \end{aligned}$$

5.3 Monotonicity proofs regarding the vulnerabilities

The most common problem that our previous attempts incurred was disrespecting the property of monotonicity 2.17, which mandates that the posterior vulnerability is always bigger or equal to the prior vulnerability. Therefore here we present a mathematical proof for each of the vulnerabilities we presented on this section. For the secret vulnerability we have that

Theorem 5.12 (Monotonicity of Vulnerability of the secret given a model)**.** *Let $\Pi : \mathbb{D}^2\mathcal{X}$, where each $\sigma^i : \mathbb{D}\mathcal{X}$ represents a possible strategy within Π. Let $M : \mathbb{D}^2\mathcal{X}$ be a model of adversarial knowledge consistent with Π, let $\mathcal{X}$ be the set of secrets, let $\mathcal{W}$ be the set of guesses, let $g : \mathcal{X} \times \mathcal{W} \to \mathbb{R}$ a gain function, and let $C : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ be a channel. Let $V(X \mid M)$ be defined as in Definition 5.1. Let $V(X \mid M \triangleright C)$ be defined as in Definition 5.4, then*

$$V(X \mid M \triangleright C) \quad \geq \quad V(X \mid M) \quad .$$

Proof.

$$\begin{aligned} V(X \mid M \triangleright C) &= \sum_{y\in\mathcal{Y}} \sum_{\sigma'\in\mathbb{D}\mathcal{X}} \max_{w\in\mathcal{W}} \sum_{x\in\mathcal{X}} M_{\sigma'}\sigma'_x C_{xy} g(w,x) && \text{(By Definition 5.4)} \\ &\geq \sum_{\sigma'\in\mathbb{D}\mathcal{X}} \max_{w\in\mathcal{W}} \sum_{x\in\mathcal{X}} M_{\sigma'}\sigma'_x g(w,x) \sum_{y\in\mathcal{Y}} C_{xy} && \text{(All terms are non-negative)} \\ &= \sum_{\sigma'\in\mathbb{D}\mathcal{X}} \max_{w\in\mathcal{W}} \sum_{x\in\mathcal{X}} M_{\sigma'}\sigma'_x g(w,x) && (\sum_{y\in\mathcal{Y}} C_{x,y} = 1) \\ &= V(X \mid M). && \square \end{aligned}$$

Finally for strategy vulnerability we have that

Theorem 5.13 (Monotonicity of vulnerability of the strategy given a model)**.** *Let* $\Pi : \mathbb{D}^2\mathcal{X}$*, where each* $\sigma^i : \mathbb{D}\mathcal{X}$ *represents a possible strategy within* Π*. Let* $A : \mathcal{S} \times \mathcal{S}'$ *be an aggregation matrix, knowledge, let* $\mathcal{X}$ *be the set of secrets, let* $\mathcal{W}$ *be the set of guesses, let* $g : \mathcal{X} \times \mathcal{W} \to \mathbb{R}$ *a gain function, and let* $C :: \mathcal{X} \to \mathbb{D}\mathcal{Y}$ *be a channel. Let* $V(S \mid M)$ *be defined as in Definition 5.2. Let* $V(S \mid M \triangleright C)$ *be defined as in Definition 5.5, then*

$$V(S \mid M \triangleright C) \quad \geq \quad V(S \mid M) \quad .$$

Proof.

$$\begin{aligned}
V(S \mid M \triangleright C) &= \sum_{y \in \mathcal{Y}} \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \sum_{x \in \mathcal{X}} \Pi_\sigma A(\sigma' | \sigma) \sigma^x C_{xy} g_{mh}(w, \sigma) && \text{(By Definition 5.5)} \\
&\geq \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma A(\sigma' | \sigma) g_{mh}(w, \sigma) \sum_{x \in \mathcal{X}} \sigma^x \sum_{y \in \mathcal{Y}} C_{xy} && \text{(All terms are non-negative)} \\
&= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma A(\sigma' | \sigma) g_{mh}(w, \sigma) \sum_{x \in \mathcal{X}} \sigma^x && (\sum_{y \in \mathcal{Y}} C_{x,y} = 1) \\
&= \sum_{\sigma' \in \mathbb{D}\mathcal{X}} \max_{w \in \mathcal{W}} \sum_{\sigma \in \mathbb{D}\mathcal{X}} \Pi_\sigma A(\sigma' | \sigma) g_{mh}(w, \sigma) && (\sum_{x \in \mathcal{X}} \sigma^x = 1)) \\
&= V(S \mid M). && \square
\end{aligned}$$

Having guaranteed monotonicity, we assure that given Definition 5.2 and Definition 5.5, the adversary cannot lose information about the secret. In the worst case, where $V(S \mid M \triangleright C) = V(S \mid M)$, we can deduce that the output will not be give any new information and thus can be disregarded. This property, being for strategies or secrets, is pivotal for the notions of leakage presented by Definition 5.7. It assures us that additive leakage is always non-negative, and multiplicative leakage is never smaller than 1.

Given these results, the new metrics satisfy one of the fundamental properties of QIF for posterior vulnerabilities [Alvim et al., 2020, chap. 11.3], therefore guaranteeing the validity of the measures.

5.4 Comparing older definitions with Definition 5.5

In this section we will show how Definition 5.5 compares to the others, and proves to be more adequate on depicting each scenario.

On Section 4.2, Example 4.4 we obtained negative results for additive leakage on two of the three the cases.

$$\begin{array}{|c|cccc|}\hline & \sigma^1 & \sigma^2 & \sigma^3 & \sigma^4 \\ \hline x_1 & 1 & 0 & 1/2 & 9/10 \\ x_2 & 0 & 1 & 1/2 & 1/10 \\ \hline bane & 1/2 & 1/2 & 0 & 0 \\ clinkz & 0 & 0 & 1 & 0 \\ dazzle & 1/2 & 0 & 0 & 1/2 \\ \hline \end{array} \triangleright \begin{array}{c} I \\ \begin{array}{|cc|}\hline 1 & 0 \\ 0 & 1 \\ \hline \end{array} \end{array} . \tag{5.3}$$

If we apply our latest definition we obtain

$$\begin{aligned} V(S \,|\, bane \triangleright I) &= 2, \\ V(S \,|\, clinkz \triangleright I) &= 1, \\ V(S \,|\, dazzle \triangleright I) &= 199/100. \end{aligned}$$

And then we verify the leakage for each case

$$\begin{aligned} \mathcal{L}_{bane}(S, M, I, \times) &= \frac{2}{2/3} = 3, \\ \mathcal{L}_{clinkz}(S, M, I, \times) &= \frac{1}{1} = 1, \\ \mathcal{L}_{dazzle}(S, M, I, \times) &= \frac{199/100}{1} = 199/100. \end{aligned}$$

which proves to be more suitable just by respecting the property of monotonicity. Additionally, it makes sense that *clinkz* has minimal leakage, since its strategy is already known to the adversary because it is a point hyper. *bane* presents a big leakage, since the strategy will be known to the adversary after the execution. Also *dazzle* represents minimum leakage, because the adversary cannot gain too much information, given the similarities between the strategies.

Section 4.3 did not present a calculation, but argued over the same example as Section 4.2, which we just presented a solution for.

Section 4.4 showed us that there would not be a good distinction between the scenarios presented on Equation 4.6, all systems would be treated as equivalent. If we apply our new approach to these scenarios we can properly distinguish them. The new model has the following results.

$$\begin{aligned} \mathcal{L}_{ember}(S, M, I, \times) &= \frac{199/100}{1} = 199/100, \\ \mathcal{L}_{huskar}(S, M, I, \times) &= \frac{2}{2/3} = 3. \end{aligned}$$

huskar presents a high amount of leakage, which makes sense since the strategy will be revealed to the adversary. Additionally, *ember* presents a low amount of leakage, which follows the premise that little knowledge will be obtained about the strategy after the hyper is pushed through the channel. This example shows that there can be room for improvement, because the leakage here should be even lower. This may be done with a different gain function.

Section 4.5, with Example 4.10 had the problem of having two environments

	σ^1	σ^2	σ^3
x_1	1	0	$1/2$
x_2	0	1	$1/2$
jakiro	$1/2$	$1/2$	0
lich	0	0	1

$\triangleright$ I

1	0
0	1

,

that differ on a fundamental basis present incoherent values with our intuition, there $\mathcal{L}_{jakiro}(S, M, I, \times) = 1$, which present would suggest for instance, that the adversary does not learn anything about the strategy for *jakiro*, which is untrue since he learns exactly what the strategy is. Also for $\mathcal{L}_{lich}(S, M, I, \times) = 2$ suggests that the adversary learns something about *lich* strategy, which cannot be possible since its already known to the adversary. If we apply our novel definition we have that

$$\mathcal{L}_{jakiro}(S, M, I, \times) = \frac{2}{2/3} = 3 \quad \textit{and} \quad \mathcal{L}_{lich}(S, M, I, \times) = \frac{1}{1} = 1.$$

which presents us with more reasonable values.

Section 4.6 was shown not to work because the values were always the same. Our final formulation does not have the same issue. As we presented.

Judging by all of these experiments, we can conclude that the new definitions for both strategy and secret vulnerability present meaningful results w.r.t the intuition behind what we would expect.

Chapter 6

Conclusion

In this thesis, we studied how prior knowledge can be modelled as hypers within the framework of Quantitative Information Flow (QIF). Furthermore we also studied how this novel model of prior knowledge impacts a few other elements within the processing of information by a system.

This issue was brought to light by Alvim et al. [2017a], which also presented an appropriate way to measure prior vulnerabilities when prior knowledge is represented by a hyper. On this thesis, in Chapter 3, we expanded this notion making possible to model posterior knowledge. This was done through two different types of updates.

We also presented a formula to calculate posterior vulnerability within this scenario. One for each type of vulnerability: secret, Definition 5.4, and strategy, Definition 5.5. We were able to test the effectiveness of each definition by restricting the set of guesses possible to the adversary so we obtain feasible calculations.

This study can be further expanded by allowing the calculations for posterior vulnerability to take into consideration an unrestricted set of guesses. This may be an optimization problem. Another way to expand these formulations is to associate it to the concept of Capacity [Alvim et al., 2017a, chap. 7] and work on the implications of this notion when using a hyper as a prior distribution.

It is also possible to improve on the tool presented in Appendex A. This could be done by developing an API for the program so it can be incorporated on other programs. A second improvement to be made is to create an interaction from the calculations made here to the QIF-related language like *kuifje*[1]. One more possible improvement on the tool is to reproduce this code on a faster language, like C++, which could make calculations more efficient.

Finally many experiments can be made with real world data, by using the code we present on Appendex A. One interesting scenario is to compare the results obtained by Mardziel et al. [2014b], who analyzed the Rockyou dataset, with the proposed formulations made in Section 5 of this thesis.

[1]Here is a good kuifje repository: https://github.com/gleisonsdm/kuifje-compiler.

Bibliography

Anne Adams, Martina Angela Sasse, and Peter Lunt. Making passwords secure and usable. In *People and computers XII*, pages 1–19. Springer, 1997.

Mário S Alvim, Miguel E Andrés, and Catuscia Palamidessi. Quantitative information flow in interactive systems. *Journal of Computer Security*, 20(1):3–50, 2012a.

Mario S Alvim, Kostas Chatzikokolakis, Catuscia Palamidessi, and Geoffrey Smith. Measuring information leakage using generalized gain functions. In *2012 IEEE 25th Computer Security Foundations Symposium*, pages 265–279. IEEE, 2012b.

Mário S Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. Axioms for information leakage. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 77–92. IEEE, 2016.

Mário S Alvim, Piotr Mardziel, and Michael Hicks. Quantifying vulnerability of secret generation using hyper-distributions. In *International Conference on Principles of Security and Trust*, pages 26–48. Springer, 2017a. https://homepages.dcc.ufmg.br/ msalvim/publications/2017-POST.pdf.

Mário S Alvim, Konstantinos Chatzikokolakis, Annabelle Mciver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. *The Science of Quantitative Information Flow*. Springer, 2020. ISBN 9783319961293, 9783319961316.

Mário S. Alvim, Piotr Mardziel, and Michael Hicks. Quantifying vulnerability of secret generation using hyper-distributions (extended version), 2017b.

A Bhattacharya. On a measure of divergence between two multinomial populations. 1946.

Jeremiah Blocki, Saranga Komanduri, Lorrie Cranor, and Anupam Datta. Spaced repetition and mnemonics enable recall of multiple strong passwords. *arXiv preprint arXiv:1410.1490*, 2014.

Christelle Braun, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Quantitative notions of leakage for one-try attacks. In *25th Conference of Mathematical Foundations of Programming Semantics*, pages 75–91, 2009.

Alan S Brown, Elisabeth Bracken, Sandy Zoccoli, and King Douglas. Generating and remembering passwords. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 18(6):641–651, 2004.

David A Cieslak, T Ryan Hoens, Nitesh V Chawla, and W Philip Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, 2012.

Luc Devroye, Laszlo Gyorfi, et al. No empirical probability measure can converge in the total variation sense for all distributions. *The Annals of Statistics*, 18(3):1496–1499, 1990.

Dinei Florêncio, Cormac Herley, and Baris Coskun. Do strong web passwords accomplish anything? *HotSec*, 7(6):159, 2007.

Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

Farhana Zaman Glory, Atif Ul Aftab, Olivier Tremblay-Savard, and Noman Mohammed. Strong password generation based on user inputs. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0416–0423. IEEE, 2019.

Eugene F Krause. Taxicab geometry. *The Mathematics Teacher*, 66(8):695–706, 1973.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Piotr Mardziel, Mário S Alvim, and Michael Hicks. Adversary gain vs. defender loss in quantified information flow. In *Workshop on Foundations of Computer Security (FCS)*, 2014a.

Piotr Mardziel, Mário S Alvim, Michael Hicks, and Michael R Clarkson. Quantifying information flow for dynamic secrets. In *2014 IEEE Symposium on Security and Privacy*, pages 540–555. IEEE, 2014b.
https://homepages.dcc.ufmg.br/ msalvim/publications/2014-S&P.pdf.

James L Massey. Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory*, page 204. IEEE, 1994.

Adam Moore. Defining privacy. *Journal of Social Philosophy*, 39(3):411–428, 2008.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

Alfréd Rényi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 4, pages 547–562. University of California Press, 1961.

Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Geoffrey Smith. On the foundations of quantitative information flow. In *International Conference on Foundations of Software Science and Computational Structures*, pages 288–302. Springer, 2009.

A_P Varga and Roger K Moore. Hidden markov model decomposition of speech and noise. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 845–848. IEEE, 1990.

Moshe Zviran and William J Haga. Password security: an empirical study. *Journal of Management Information Systems*, 15(4):161–185, 1999.

Appendix A

A tool for calculating secret and strategy leakage

In this section we describe the an additional contribution of this thesis. We present a tool which is able to make the QIF operations we presented previously. We can take a prior hyper as an input, push it through a channel and receive the squashed version of the posterior hyper, this represents our update function from Section 3.1.1. Additionally we can calculate the vulnerabilities from our final definition, Chapter 5, and the leakages related to them.

A.1 Implementation

Our main goal with the implementation [1] is to be able to calculate the pushing of a hyper through a channel and determine the values for vulnerabilities. We are not worried about efficiency, just correctness. Therefore we choose to implement the code with Python3. Python is a comprehensible language, which makes the code easier to read. We also have many libraries to chose from, which make the code less clunky. Numpy and Fraction are two very important libraries, since many of QIF calculations work with matrices, which are easier to handle with Numpy and fractions.

We did not make experiments on large scale, but that is a future proposition. This is due to the fact that Python is not an efficient language like C++, so translating the code to C++ would make large scale experiments more viable.

A.2 Functionalities

In this Section we will present a few functionalities of the code.

[1] For access to the code, visit the git repository at https://github.com/tannus/dynamic˙secrets.

A.2.1 Hyper Class

The *Hyper Class* is the most basic class implemented, it served as a basis to the other two important classes we have. This class has the outer and inner as attributes. It also has the get functions for each of the attributes and three print functions for outer, inner and the hyper itself.

A.2.2 Environment Class

The *Environment Class* is an expansion of the Hyper. It is by far the most used class on the code. It has its attributes outer and strat similar to the Hyper Class. It has the same functionalities as the hyper. In addition to that it has a function to return the concise version of the environment, a function to simplify the environment eliminating all its 0 columns, and a function to make guesses relevant to the context.

A.2.3 Aggregation Class

The *Aggregation Class* is simple class that represents the aggregation matrices. It has two attributes for the matrix and its labels, which represent each strategy maps on each column of the aggregation matrix. Most importantly it has a function to make the model matrix, given an environment and its abstraction matrix.

A.2.4 Pushing Functions

There are two classes of *pushing functions*. The first class consists of basic functions that work with the inner steps of the push through operation. We have a function for computing the joint, marginalizing the joint, calculating the outer from a joint, and calculating a inner from a joint. The other class works one step forward, using the previously described operations. We can push a simple distribition through a channel, push a simple distribution through a markov, push a hyper through a channel, push a hyper through a markov, push a column from an environment through a channel, and push an environment through a channel. Additionaly we have a more complex function, called model push that encapsulates all the steps from environment to abstraction matrix to model to posterior hyper and that also calculates vulnerabilities.

A.2.5 Squashing Functions

Squashing Functions are very straightforward, they take a n-hyper as an input and lower it one dimension. There are two types of this function implemented. One for general hypers and one specific for environments.

A.2.6 Vulnerability Functions

Vulnerability and leakage functions are another straightforward concept. We have one function for each case of vulnerability. Prior secret and prior strategy vulnerability behave very similarly and are simple functions following Definitions 5.1 and 5.4. Prior strategy vulnerability adds more complexity to the calculations, since it has to also calculate gains with manhattan distance, it follows Definition 5.2. Posterior strategy vulnerability adds even more complexity to the previous case, but behaves similarly. It follows Definition 5.5.

A.2.7 Guess and Gain Functions

There are two *Guess functions*, one to generate the basic set of guesses based on the environment, and another to make synthetic guesses. *Gain functions* are very simple and follow the Definition 4.6.

A.2.8 Input Functions

Additionally to the functions about QIF, we implemented a few functions to take a file as input and create the relevant data structures for it, that is vectors and stochastic matrices. We also check the validity of each probability distribution used as input.

A.2.9 Print Functions

We have a few functions that print all the matrices and vectors in a human-readable way.