Universidad Autónoma de Madrid | Escuela de Doctorado

Tesis Doctoral

Contributions to cybersecurity risk management for Artificial Intelligence systems

José Manuel Camacho Rodríguez

Programa de Doctorado en Ingeniería Informática y de Telecomunicación

Dirección:
David Ríos Insua
Roi Naveiro

Madrid, 2025

Department: Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid (UAM)
Spain

Title: Contributions to cybersecurity risk management for Artificial Intelligence systems

Author: José Manuel Camacho Rodríguez

Advisor 1: David Ríos Insua

Advisor 2: Roi Naveiro

Tutor: Luis Fernando Lago Fernández

Date: November 2025

Keywords (Palabras claves): Cybersecurity Risk Management (Gestión de riesgos de ciberseguridad); Artificial Intelligence (Inteligencia Artificial); Adversarial Risk Analysis (Análisis de Riesgo Adversario); Adversarial Machine Learning (Aprendizaje Automático Adversario); Disinformation (Desinformación)

Committee:

- President: José Ramón Dorronsoro Ibero
- Secretary: Cristina Conde Vilda
- Vocal: Jesús María Ríos Aliaga
- Substitute 1: Refik Soyer
- Substitute 2: María Concepción Ausín Olivera

Acknowledgements

La realización de esta tesis no habría sido posible sin la ayuda y el apoyo de tantas personas, a quienes me gustaría expresar mi agradecimiento.

Quiero agradecer a mi director, el Prof. David Ríos Insua, por brindarme la oportunidad de realizar el doctorado bajo su tutela, por su apoyo y supervisión durante este periodo, y por su labor formadora. Sus enseñanzas serán, sin duda, de gran valor para mi futura vida profesional. También quiero agradecer al Prof. Roi Naveiro su ayuda, primero como compañero y, más tarde, como codirector. Siempre le estaré agradecido por su ayuda y cercanía, por facilitar mi adaptación al ICMAT y a mi vida en Madrid, así como por su acompañamiento investigador, disponibilidad y paciencia, que me han ayudado a aprender y a sacar el máximo provecho de mi experiencia predoctoral. Quiero mostrar además mi agradecimiento a mi tutor, el Prof. Luis Lago Fernández, por asistirme con los distintos trámites del doctorado.

Asimismo, quiero expresar mi gratitud a los investigadores con los que he tenido la oportunidad de colaborar en los trabajos que conforman esta tesis: Prof. Panizzi, Dr. Arroyo, Prof. Antenore, Luis Pérez-Miguel, Prof. Caballero, Dr. Couce-Vieira y Prof. Ekin. Quiero hacer una mención especial al Prof. Panizzi, por permitirme realizar una estancia en su laboratorio, así como a sus doctorandos, Gabriella, Varma, Stefano y Alba, por su cálida acogida; y al Dr. Arroyo, por su predisposición para ayudarme en todo lo posible.

A mis compañeros durante el doctorado, Pablo G., Daniel C., Carlos G., Mario C., Miguel S., Aida G., Manuel S., Pablo V., Dani G., Simón R., Bruno F., César G., Nicolas F. . . . , por hacer de esta etapa una experiencia más especial y enriquecedora y por contribuir, algunos de ellos, a mejorar mis habilidades de ping-pong.

A mi familia, por ser mi mayor sostén. A mi padre, Francisco, y a mi madre, Mari Paz, por vuestro amor y apoyo incondicional, y por los valores que me habéis transmitido. Sois la guía que siempre me orienta. A mi hermano Jesús, por estar siempre cerca, por tu afecto y tus consejos, así como por ser ese referente que siempre me ha ayudado a ampliar mis horizontes. A mi cuñada, Nathalie, y a mis sobrinos, Emma y Noah, por todo el cariño que me regaláis. A Simo, por acompañarme durante todo el doctorado, por tu amor y respaldo constante, y por creer siempre en mí, dándome la fuerza necesaria para seguir adelante.

A mis amigos Ricky y Belén, por estar siempre presentes y por esas conversaciones en las que las horas pasan volando; y a mis amigos del Erasmus en Madrid, Carmen, David y Sara, por esos planes en los que las preocupaciones se olvidan por completo. Mi agradecimiento también para Alejandro, mi profesor de Matemáticas durante la secundaria y el bachillerato, cuyas habilidades docentes y paciencia despertaron en mí el gusto por las matemáticas.

Por último, quiero agradecer a la Fundación “la Caixa” (ID 100010434), que ha financiado este doctorado mediante la beca LCF/BQ/DI21/11860063; y reconocer el apoyo brindado por la cátedra AXA-ICMAT en Análisis de Riesgo Adversario, el proyecto EOARD-AFOSR (RC2APD 13324227), el programa de investigación e innovación EU’s Horizon 2020 número 101021797 (STARLIGHT) y al ministerio de Ciencia y Tecnología (PID2021-124662OB-IOO).

Abstract

Artificial Intelligence (AI) is becoming a core element in modern digital systems, advancing numerous technologies and disciplines, such as drug discovery, autonomous driving, or cybersecurity in significant ways. However, in relation to cybersecurity, AI is also a double-edged sword. Alongside its benefits, AI introduces novel risks that expand the cybersecurity threat landscape. These include emerging vulnerabilities in AI-based systems and diverse impacts associated with AI. As a result, current cybersecurity risk management (CSRM) methodologies, designed for traditional information systems, are limited in addressing the new reality.

This thesis advances the CSRM domain to address the issues emerging from AI-based systems from a holistic perspective through its first objective (**O1**), while the remaining objectives (**O2** to **O4**) further develop specific components of the field in its intersection with AI. To achieve this, Adversarial Risk Analysis (ARA) is used as a recurring element, as it provides an alternative modeling framework to game theory that enables cybersecurity scenarios to be analyzed more realistically.

The first objective of the thesis (**O1**), covered in Chapter 2, is to develop a cybersecurity risk analysis framework tailored to systems with AI components. The framework addresses the new challenges posed by AI in risk management and supports the selection of an optimal protection portfolio to mitigate risk. It also employs ARA to model targeted threats affecting the system under analysis. The applicability of the framework is demonstrated through a case study on automated driving systems.

The second objective of the thesis (**O2**), accomplished in Chapter 3, is to implement a computational methodology that applies ARA to identify optimal decisions in scenarios involving two adversaries (a defender and an attacker) modeled as security games. By contributing to the analysis and solution of these games, **O2** supports a deeper understanding of attacks that may target AI systems and potentially improves their protection. The methodology handles multi-stage games and accommodates both discrete and continuous decision domains. Using a disinformation war example, the chapter shows how the methodology can provide optimal defensive decisions and forecast attacker behavior in security contexts.

The third (**O3**) and fourth (**O4**) objectives of this thesis examine security controls that rely on AI. The objective **O3**, addressed in Chapter 4, focuses on investigating the vulnerabilities of Hidden Markov Model (HMM) inference when input batch observation data is corrupted. The chapter introduces a set of problems that target filtering, smoothing, and decoding inferences by adopting an ARA approach to model the malicious actor attempting to manipulate the HMM. To tackle the complexity of these problems, the chapter develops and evaluates three algorithmic routines. Extensive experiments, including a part-of-speech tagging case study, show that even small perturbations to batches of data can substantially degrade HMM inference and prediction.

The fourth objective (**O4**), achieved in Chapter 5, is to examine machine learning methods for detecting bots in Twitter. The chapter presents a comparative study of multiple supervised detection models, evaluating data requirements and performance across different bot behaviors.

Building on these results, the chapter applies one of the models to analyze bot presence during the COVID-19 pandemic and develops an application to support the analysis of content spread by bots.

From the contents of this thesis, a series of papers have been published. Those fulfilling the requirements for its submission are:

CAMACHO, J. M., NAVEIRO, R., AND RIOS INSUA, D. (2025). "Algorithmic Decision Analysis for Multi-Stage Games with Incomplete Information". In: *Algorithmic decision theory. ADT 2024. Lecture Notes in Computer Science* 15248, Springer.

CAMACHO, J. M., COUCE-VIEIRA, A., ARROYO, D., AND INSUA, D. R. (2026). "A cybersecurity risk analysis framework for systems with artificial intelligence components". In: *International Transactions in Operational Research* 33.2, pp. 798-825.

The other published papers derived from the thesis are:

INSUA, D. R., AND CAMACHO, J. M. (2022). "Modelos matemáticos para la gestión de la ciberseguridad". In: *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales* 111.1, pp. 38-53.

NAVEIRO, R., INSUA, D. R., AND CAMACHO, J. M. (2022). "Augmented probability simulation for adversarial risk analysis in general security games". In: *Proceedings of 12th International Defence and Homeland Security Simulation Workshop (DHSS 2022)*, 002. **Best Paper DHSS 2022 award**.

ANTENORE, M., CAMACHO RODRIGUEZ, J. M., AND PANIZZI, E. (2023). "A comparative study of bot detection techniques with an application in Twitter Covid-19 discourse". In: *Social Science Computer Review* 41.5, pp. 1520-1545.

CAMACHO, J. M., PEREZ-MIGUEL, L., AND ARROYO, D. (2023). "WIDISBOT: Widget to analyse disinformation and content spread by bots". In: *Proceedings of the 4th Conference on Language, Data and Knowledge (LDK 2023)*, pp. 514-519.

CABALLERO, W. N., CAMACHO, J. M., EKIN, T., AND NAVEIRO, R. (2024). "Manipulating hidden-Markov-model inferences by corrupting batch data". In: *Computers & Operations Research* 162, 106478.

While the next one is under review:

CAMACHO, J. M., NAVEIRO, R., AND RIOS INSUA, D. (2025). "Computational adversarial risk analysis for general security games". In: ArXiv. URL: https://arxiv.org/pdf/2506.02603

In addition, a series of presentations at national and international conferences has also been derived from its content:

"El objetivo 16. Usos y abusos del aprendizaje automático" (January, 2022). Oral presentation at the *Biennial Congress of the Royal Spanish Mathematical Society (RSME22)*, University of Castilla - La Mancha, Ciudad Real, Spain.

"Poisoning Hidden-Markov Model Inferences on Batch Data" (July, 2022). Oral presentation at the *32nd European Conference on Operational Research (EURO 2022)*, Aalto University, Espoo, Finland.

"Augmented probability simulation for adversarial risk analysis in general security games" (September, 2022). Oral presentation at the *12th International Defense and Homeland Security Simulation Workshop (DHSS 2022)*, University of Rome Tor Vergata, Rome, Italy.

"Computational adversarial risk analysis for general security games" (October, 2022). Oral presentation at the *8th Workshop on Risk Management and Insurance Research (RISK 2022)*, University of Barcelona, Barcelona, Spain.

"Manipulation of Hidden-Markov-Model Inferences through Poisoned Batch Data" (May, 2023). Oral presentation at the *13th Bayesian Inferences for Stochastic Processes (BISP13)*, Spanish Royal Academy of Sciences, Madrid, Spain.

"A framework for risk management in systems with AI components" (May, 2023). Oral presentation at the *Games & Decisions in Risk & Reliability 7 (GDRR7)*, Spanish Royal of Sciences, Madrid, Spain.

"Widget to analyze disinformation and content spread by bots" (September, 2023). Oral presentation at the *1st International Workshop on Disinformation and Toxic Content Analysis (DiTox 2023)*, University of Vienna, Vienna, Austria.

"A computational framework to solve general security games through Bayesian approach. (November, 2023). Oral presentation at the *XL National Congress of Statistics and Operations Research (SEIO 2023)*, University Miguel Hernandez of Elche, Elche, Spain.

"Comparing Machine Learning Methods for Bot Detection on Twitter with a Case Study on COVID-19 Discourse and the Implementation of a Bot Content Analysis Tool" (December, 2023). Oral presentation at the *Andaluz.IA 2023*, Pablo de Olavide University, Seville, Spain.

"A Cybersecurity Risk Analysis Framework for Systems with Artificial Intelligence Components" (July, 2024). Oral presentation at the *2024 World Meeting of the Society for Bayesian Analysis (ISBA 2024)*, Ca' Foscari University of Venice, Venice, Italy.

"Algorithmic Decision Analysis for Multi-Stage Games with Incomplete Information" (October, 2024) Oral and poster presentation at the *The 8th International Conference on Algorithmic Decision Theory (ADT24)*, Rutgers University, New Jersey, U.S.A.

"Manipulating Hidden-Markov-Model inferences by corrupting batch data" (December, 2024). Poster presentation at the *Andaluz.AI 2024*, Universidad of Jaen, Jaen,Spain.

"Adversarial Risk Analysis for General Security Games: A Computational Approach" (Mayo, 2025). Oral presentation at the *Games & Decisions in Risk & Reliability 8 (GDRR8)*, Bocconi University, Milan, Italy.

"A Bayesian computational methodology to solve general security games" (June, 2025). Oral presentation at the *XLI National Congress of Statistics and Operations Research (SEIO 2025)*, Universidad de Lleida, España.

Resumen

La inteligencia artificial (IA) se está convirtiendo en elemento fundamental de los sistemas digitales modernos, impulsando tecnologías y disciplinas de manera significativa, como el descubrimiento de fármacos, la conducción autónoma o la ciberseguridad. No obstante, en el ámbito de la ciberseguridad, la IA también representa un arma de doble filo. Junto con sus beneficios, la IA introduce riesgos nuevos que amplían el panorama de amenazas. Entre ellos se incluyen vulnerabilidades emergentes en sistemas basados en IA y una variedad de impactos asociados a su uso. Como consecuencia, las metodologías actuales de gestión de riesgos de ciberseguridad (GRCS), diseñadas para sistemas de información tradicionales, muestran limitaciones a la hora de hacer frente a esta nueva realidad.

Esta tesis impulsa el ámbito de la GRCS para abordar, desde una perspectiva global, los problemas derivados de los sistemas basados en IA a través de su primer objetivo (**O1**), mientras que los objetivos restantes (**O2** a **O4**) desarrollan en mayor profundidad componentes específicos del campo en su intersección con la IA. Para ello, se utiliza el Análisis de Riesgo Adversario (ARA) como elemento recurrente, ya que ofrece un marco de modelización alternativo a la teoría de juegos que permite analizar los escenarios de ciberseguridad bajo hipótesis más realistas.

El primer objetivo de la tesis (**O1**), desarrollado en el Capítulo 2, es elaborar un marco de análisis de riesgos de ciberseguridad adaptado a sistemas con componentes de IA. Dicho marco aborda los nuevos desafíos que plantea la IA en la gestión del riesgo y facilita la selección de una cartera óptima de medidas de protección para mitigar riesgos. Además, emplea ARA para modelizar amenazas dirigidas que afecten al sistema bajo análisis. La aplicabilidad del marco se demuestra mediante un estudio de caso sobre sistemas de conducción autónoma.

El segundo objetivo de la tesis (**O2**), en el Capítulo 3, es implementar una metodología computacional que aplique ARA para identificar decisiones óptimas en escenarios que involucran a dos adversarios (un defensor y un atacante) modelizados como juegos de seguridad. Al contribuir al análisis y la resolución de estos juegos, **O2** aporta una comprensión más profunda de los ataques que pueden dirigirse contra sistemas de IA y, potencialmente, mejorar su protección. La metodología contempla juegos de múltiples etapas y admite tanto dominios de decisión discretos como continuos. Utilizando un ejemplo de guerra de desinformación, el capítulo muestra cómo la metodología puede proporcionar decisiones defensivas óptimas y prever el comportamiento del atacante en contextos de seguridad.

El tercer (**O3**) y cuarto (**O4**) objetivos de esta tesis examinan controles de seguridad que se basan en IA. El objetivo **O3**, tratado en el Capítulo 4, se centra en investigar las vulnerabilidades de la inferencia mediante modelos ocultos de Markov (MOM) cuando se alteran los datos de observación por lotes de entrada. El capítulo introduce un conjunto de problemas que atacan las inferencias de filtrado, suavizado y decodificación, adoptando un enfoque ARA para modelizar al actor malicioso que intenta manipular el MOM. Para abordar la complejidad de estos problemas, el capítulo desarrolla y evalúa tres rutinas algorítmicas. Experimentos extensivos, incluido un caso de estudio de etiquetado de categorías gramaticales, muestran que incluso pequeñas

perturbaciones en los lotes de datos pueden degradar sustancialmente la inferencia en los MOM.

El cuarto objetivo (**O4**), Capítulo 5, es examinar métodos de aprendizaje automático para la detección de bots en Twitter. El capítulo presenta un estudio comparativo de diversos modelos de detección supervisada, evaluando los requisitos de datos y el rendimiento ante distintos comportamientos de bots. A partir de estos resultados, se aplica uno de los modelos para analizar la presencia de bots durante la pandemia de COVID-19 y se desarrolla una aplicación para apoyar el análisis del contenido difundido por bots.

A partir del contenido de esta tesis se han publicado varios artículos. Aquellos que satisfacen los requisitos para su presentación son:

CAMACHO, J. M., NAVEIRO, R., AND RIOS INSUA, D. (2025). "Algorithmic Decision Analysis for Multi-Stage Games with Incomplete Information". In: *Algorithmic decision theory. ADT 2024. Lecture Notes in Computer Science* 15248, Springer.

CAMACHO, J. M., COUCE-VIEIRA, A., ARROYO, D., AND INSUA, D. R. (2026). "A cybersecurity risk analysis framework for systems with artificial intelligence components". In: *International Transactions in Operational Research* 33.2, pp. 798-825.

Las otras publicaciones derivadas de la tesis son:

INSUA, D. R., AND CAMACHO, J. M. (2022). "Modelos matemáticos para la gestión de la ciberseguridad". In: *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales* 111.1, pp. 38-53.

NAVEIRO, R., INSUA, D. R., AND CAMACHO, J. M. (2022). "Augmented probability simulation for adversarial risk analysis in general security games". In: *Proceedings of 12th International Defence and Homeland Security Simulation Workshop (DHSS 2022)*, 002. **Best Paper DHSS 2022**.

ANTENORE, M., CAMACHO RODRIGUEZ, J. M., AND PANIZZI, E. (2023). "A comparative study of bot detection techniques with an application in Twitter Covid-19 discourse". In: *Social Science Computer Review* 41.5, pp. 1520-1545.

CAMACHO, J. M., PEREZ-MIGUEL, L., AND ARROYO, D. (2023). "WIDISBOT: Widget to analyse disinformation and content spread by bots". In: *Proceedings of the 4th Conference on Language, Data and Knowledge (LDK 2023)*, pp. 514-519.

CABALLERO, W. N., CAMACHO, J. M., EKIN, T., AND NAVEIRO, R. (2024). "Manipulating hidden-Markov-model inferences by corrupting batch data". In: *Computers & Operations Research* 162, 106478.

Además el próximo artículo está bajo revisión:

CAMACHO, J. M., NAVEIRO, R., AND RIOS INSUA, D. (2025). "Computational adversarial risk analysis for general security games". In: ArXiv. URL: https://arxiv.org/pdf/2506.02603

Igualmente, una serie de presentaciones en conferencias nacionales e internacionales también se han derivado de su contenido:

"El objetivo 16. Usos y abusos del aprendizaje automático" (Enero, 2022). Presentación oral en *Biennial Congress of the Royal Spanish Mathematical Society (RSME22)*, Universidad de Castilla - La Mancha, Ciudad Real, España.

"Poisoning Hidden-Markov Model Inferences on Batch Data" (Julio, 2022). Presentación oral en *32nd European Conference on Operational Research (EURO 2022)*, Universidad de Aalto, Espoo, Finlandia.

"Augmented probability simulation for adversarial risk analysis in general security games" (Septiembre, 2022). Presentación oral en *12th International Defense and Homeland Security Simulation Workshop (DHSS 2022)*, Universidad Roma Tor Vergata, Roma, Italia.

"Computational adversarial risk analysis for general security games" (Octubre, 2022). Presentación oral en *8th Workshop on Risk Management and Insurance Research (RISK 2022)*, Universidad de Barcelona, Barcelona, España.

"Manipulation of Hidden-Markov-Model Inferences through Poisoned Batch Data" (Mayo, 2023). Presentación oral en *13th Bayesian Inferences for Stochastic Processes (BISP13)*, Real Academia de las Ciencias, Madrid, España.

"A framework for risk management in systems with AI components" (Mayo 2023). Presentación oral en *Games & Decisions in Risk & Reliability 7 (GDRR7)*, Real Academia de las Ciencias, Madrid, España.

"Widget to analyze disinformation and content spread by bots" (Septiembre, 2023). Presentación oral en *1st International Workshop on Disinformation and Toxic Content Analysis (DiTox 2023)*, Universidad de Viena, Viena, Austria.

"A computational framework to solve general security games through Bayesian approach" (Noviembre, 2023). Presentación oral en *XL National Congress of Statistics and Operations Research (SEIO 2023)*, Universidad Miguel Hernandez de Elche, Elche, España.

"Comparing Machine Learning Methods for Bot Detection on Twitter with a Case Study on COVID-19 Discourse and the Implementation of a Bot Content Analysis Tool" (Diciembre, 2023). Presentación oral en *Andaluz.IA 2023*, Universidad Pablo de Olavide, Sevilla, España.

"A Cybersecurity Risk Analysis Framework for Systems with Artificial Intelligence Components" (Julio, 2024). Presentación oral en *2024 World Meeting of the Society for Bayesian Analysis (ISBA 2024)*, Universidad de Venecia Ca' Foscari, Venecia, Italia.

"Algorithmic Decision Analysis for Multi-Stage Games with Incomplete Information" (Octubre, 2024). Presentación oral y póster en *The 8th International Conference on Algorithmic Decision Theory (ADT24)*, Universidad de Rutgers, Nueva Jersey, Estados Unidos.

"Manipulating Hidden-Markov-Model inferences by corrupting batch data" (Diciembre, 2024). Presentación de póster en *Andaluz.AI 2024*, Universidad de Jaén, España.

"Adversarial Risk Analysis for General Security Games: A Computational Approach" (Mayo, 2025). Presentación oral en *Games & Decisions in Risk & Reliability 8 (GDRR8)*, Universidad de Bocconi, Milán, Italia.

"A Bayesian computational methodology to solve general security games" (Junio, 2025). Presentación oral en *XLI National Congress of Statistics and Operations Research (SEIO 2025)*, Universidad de Lleida, España.

Contents

Acknowledgements **v**

Abstract **vii**

Resumen **xi**

Contents **xv**

List of Tables **xviii**

List of Figures **xx**

List of Algorithms **xxi**

Abbreviations **xxiii**

1 Introduction **1**
1.1 Motivation 1
1.2 Cybersecurity risk management: a brief review 3
1.3 Novel AI-related risk analysis issues 5
1.4 Adversarial Risk Analysis 7
1.5 Thesis objectives and structure 9

2 A cybersecurity risk analysis framework for systems with artificial intelligence components **11**
2.1 Introduction 11
2.2 Solutions to novel AI-related risk analysis issues 12
2.3 The framework 19
2.4 Case study 25
2.5 Discussion 30

3 Computational adversarial risk analysis for general security games **31**
3.1 Introduction 31
3.2 Solving simultaneous defend-attack games 32
3.3 General security games 38
3.4 Computational issues 46
3.5 Disinformation campaign case study 48
3.6 Discussion 54

4 Manipulating hidden Markov model inferences by corrupting batch data 57
4.1 Introduction . . . 57
4.2 Background on HMMs . . . 58
4.3 Hidden Markov Model Corruption Problems . . . 60
4.4 Solution Methodologies . . . 66
4.5 Testing, Results, and Analysis . . . 71
4.6 Discussion . . . 84

5 A comparative study of machine learning models for bot detection and their application on Twitter 85
5.1 Introduction . . . 85
5.2 Comparative study of bot detection models . . . 85
5.3 Analyzing bot presence and sentiment differences during the COVID-19 pandemic 95
5.4 Development of an application to analyze the content spread by bots . . . 98
5.5 Discussion . . . 103

6 Conclusions and future work 105
6.1 Conclusions . . . 105
6.2 Future work . . . 107

7 Conclusiones y trabajo futuro 109
7.1 Conclusiones . . . 109
7.2 Trabajos futuros . . . 111

Appendices 113

A Automated Driving System case study parametric setup 115
A.1 Notation and general considerations . . . 115
A.2 Non-targeted attacks . . . 116
A.3 Targeted attacks . . . 118

B Supplementary materials for disinformation campaign case study 125
B.1 Modeling details in case study . . . 125
B.2 Augmentation parameters H used in D's and A's decision stages . . . 129
B.3 Approximating $\psi_D(d_1, a_2, \theta_1)$ from Defender's second stage decision . . . 129
B.4 Approximating $p_D(a_2|d_1, a_1)$ from Attacker's second stage decision . . . 130
B.5 Approximating $\Psi_A(d_1, a_1)$ from Attacker's second stage decision . . . 131
B.6 Approximating $p_D(a_1)$ from Attacker's first stage decision . . . 132

C Implementation Details of the R&S Heuristic 133
C.1 Monte Carlo Tree Search . . . 133
C.2 Simulated Annealing . . . 135
C.3 Hyperparameter Tuning . . . 135

Bibliography 136

List of Tables

2.1 Mapping trustworthiness features into cybersecurity objectives. Indices in terms identify references addressing the principle. Indices in definitions identify the source. References for indices are: (a) NIST AIRMF (NIST, 2023), (b) CRFM's On the Opportunities and Risks of Foundation Models (CRFM, 2021), (c) EU AI Act (European Parliament, 2024), (d) Article 21, EU Charter of Fundamental Rights (European Parliament et al., 2012), (e) ISO/IEC TS 5723:2002 (ISO, 2022), (f) ALTAI list (Ala-Pietilä et al., 2020). 13
2.2 Attacks available to malicious actors. 26
2.3 Impacts deemed relevant for targeted and untargeted threats. 26
2.4 Defense effectiveness. 27
2.5 Impacts mitigated by each cyber insurance product. 27
2.6 Three best portfolios and initial portfolio configuration. 28
2.7 Probability of perpetrating an attack on our company or attacking another one by the cyberterrorists, with initial and optimal configurations. 28
2.8 Γ vector of each targeted attack for initial and optimal portfolio configuration. . 29
2.9 Optimal portfolio for different values of maximum budget. 29
2.10 Optimal portfolio for several price reductions in insurance B. 29

3.1 Nodes, interpretation, and feasible values for BAID in Figure 3.4a. 48

4.1 Summary of Additional Notation for HMM Corruption Problems 61
4.2 2^{3-1}_{III} Design For HMM Structure Testing 77
4.3 Attacker's Objective Function Parameters by Problem and Design Point 77
4.4 Mean Impact and Proportion of Observations Attacked (Structure Testing) . . . 79
4.5 2^2 Full Factorial Design for Uncertainty Testing 80
4.6 Mean Impact and Proportion of Observations Attacked (Uncertainty Testing) . . 81
4.7 Mean Impact, Proportion of Observations Attacked and Expected Utility (NER Experiment) 83

5.1 Name and description of features retrieved directly from User Object. 86
5.2 Name and description of features derived from User Object. 87
5.3 Set of features used for the comparison of bot detection models. 88
5.4 Datasets used for training, from Lee et al. (2011), Cresci et al. (2017b), Varol et al. (2017), Yang et al. (2019), and Yang et al. (2020). 89
5.5 Datasets used for test, taken from Yang et al. (2020), Mazza et al. (2019), Gilani et al. (2017), Rauchfleisch and Kaiser (2020), Cresci et al. (2018), and Cresci et al. (2019). 89
5.6 Number of bots and humans in each training dataset divided depending on whether we use the features from the User Object and the Social Fingerprint. 90

5.7 Number of bots and humans in each test dataset divided depending on whether we use the features from the User Object and the Social Fingerprint. 91
5.8 Best model for each set of features. 93
5.9 Comparison in performance of $BTM3$ and the best models with the set of features *Light* and *D*. 94
5.10 Number of accounts and tweets for each case studied. 96
5.11 Proportion across topics of (Left) bots and human accounts and (Right) tweets by bots and human accounts. 96
5.12 Proportion of sentiment of tweets from human and bot accounts across topics. . . 98
5.13 AUC scores of the models using set *Light* for tweets extracted using Twitter API v2 and v1.1, and of $BTM3$, on the test datasets from Section 5.2. 100

A.1 Prices in euros of insurance on an ADS depending on portfolio. 116
A.2 SCT PNP distribution parameters depending on portfolio. 118
A.3 Impact distribution parameters of SCTs depending on portfolio. 118
A.4 Values of parameter ω for different portfolio configurations and attacks. 120
A.5 Successful attack probability beta parameters of wireless jamming conducted by the cyberterrorist group regarding the targeted block. 121
A.6 Successful attack probability beta parameters of AML attack conducted by criminal gang regarding the targeted block. 121
A.7 Cy AML PNP distribution Beta parameters depending on portfolio. 122
A.8 Cy wjam PNP distribution beta parameters. 122
A.9 Cr AML portfolio dependent PNP beta parameters. 123
A.10 Portfolio dependent impact distribution Gamma parameters of AML attack by cyberterrorist. 123
A.11 Portfolio dependent impact distribution Gamma parameters of wireless jamming conducted by cyberterrorist group. 123
A.12 Portfolio dependent impact distribution Gamma parameters of AML attack by a criminal gang. 123

B.1 Augmentation parameter H used in the Defender's and Attacker's decision stages. 129
B.2 Mean and standard error of 5-fold CV results over 10 iterations for best five MLP architectures. Each element of the architecture represents a hidden layer, with the number indicating the neurons. 130
B.3 Mean and standard error of the 5-fold CV results over 10 iterations for best five MLP architectures (Arch.) and number of MoB components (# Comp.) used to estimate $A_2^*(d_1, a_1)$. 130
B.4 Mean and standard error of the 5-fold CV results over 10 iterations for the top five MLP architectures (Arch.) and number of MoW components (# Comp.) used to estimate $\Psi_A(d_1, a_1)$. 131

C.1 Hyperparameter Combinations for the R&S-A Algorithm. 135
C.2 Hyperparameter Combinations for the R&S-B Algorithm. 135
C.3 Superlative Hyperparameter Combination for Section Structural Testing 136
C.4 Superlative Hyperparameter Combination for HMM Uncertainty Testing 136

List of Figures

1.1 Detailed representation of the components involved in CSRM. Black arrows indicate the components affected by AI-based systems. 4
1.2 BAIDs for (Left) sequential and (Right) simultaneous defend-attack games. . . . 8
1.3 (Left) defender's decision problem; (Right) defender's analysis of attacker's problem. 9

2.1 Cybersecurity objective tree. Red, monetary objectives; blue non-monetary objectives. Adapted from (Couce-Vieira et al., 2020). 12
2.2 (a) ADS, and (b) Threat assessment architectures. 15
2.3 Marketing content creation system architecture. Four blocks in three levels. △ 15
2.4 Security evaluation curve of a deep network under four defense mechanisms against FGSM attack. From Gallego et al. (2024). 19
2.5 Loss curves per ADS on a log scale for initial (blue) and optimal (red) configurations. 28

3.1 (a) Simultaneous defend-attack game BAID, (b) D's problem, (c) D's analysis of A's problem. 33
3.2 Template for basic n-stage simultaneous defend-attack game BAID. 37
3.3 D's and A's influence diagrams for the 2-stage simultaneous defend-attack game. 38
3.4 (a) BAIDS for DC case study, (b) D's influence diagram, (c) A's influence diagram. 39
3.5 ID for problem where all nodes affected by arc inversion are reduced before node D. 41
3.6 ID for problem where not all nodes affected by arc inversion are reduced before node D. 42
3.7 (Left) Mean $\langle d_2^*(d_1, a_2, \theta_1)\rangle_{d_1}$ of $d_2^*(d_1, a_2, \theta_1)$ over d_1. (Right) Expected number $E[\theta_2 \mid d_2^*(d_1, a_2, \theta_1), a_2, \theta_1]$ of individuals to be infected. 50
3.8 Expected random optimal attack $A_2^*(d_1, a_1)$ in second stage. 51
3.9 Density of empirical distribution A_1^*. 52
3.10 Expected degree of recognition $E[\theta_1 \mid d_1^*, a_1, a_2]$ for d_1^* across different (a_1, a_2). . 52
3.11 (Top) Mean $\langle d_2^*(d_1, a_2, \theta_1)\rangle_{d_1}$ of $d_2^*(d_1, a_2, \theta_1)$ over d_1 for different ω_{d_2}. (Bottom) Expected number $E[\theta_2 \mid d_2^*(d_1, a_2, \theta_1), a_2, \theta_1]$ of individuals to be infected for different ω_{d_2}. 53
3.12 $\overline{A}_2^*(d_1, a_1)$ for several (t_d, t_a). 54

4.1 A basic hidden Markov model 58
4.2 HMM Corruption Problem. 61
4.3 Ratio Plots for each Problem-and-Uncertainty Pair 74
4.4 Mean Attack Efficacy by Number of Perturbed Observations 76
4.5 Evolution of Path-Attraction Expected Utilities over time by Algorithm 78
4.6 Temporal Evolution of Distribution-Disruption Expected Utility by Algorithm . . 81
4.7 Expected Utility of Attacks Plus/Minus One Standard Deviation in the POS Case Study 83

5.1 Screenshots of WIDISBOT output. (Left) Proportion of accounts labelled as bots/humans and the fraction of tweets produced by each type. (Right) Sentiment distribution in human/bots tweets. 101
5.2 Ten most shared hashtags by bots and humans. 102
5.3 Screenshot of the most shared website by bots in tweets with hashtag *#deathtoll*, accessed through the Wayback Machine. 103

B.1 ID representing D's problem. 125
B.2 ID representing A's problem. 127
B.3 Empirical distribution of $A_2^*(d_1, a_1)$ (blue) against MoB approximation (red). . . 130
B.4 Density of empirical $\Psi_A(d_1, a_1)$ (blue) and MoW model approximation (red). . . 131
B.5 Comparison between empirical distribution (blue) of A_1^* and its approximation (red). 132

List of Algorithms

1 Compute (random) optimal attacks through Monte Carlo 18
2 Compute distribution of attacker behaviour $\big(\tau(\mathbf{c})\big)$ 18
3 Compute target parameter distribution for attack j $(\Gamma_j^1(\mathbf{c}))$ 18
4 Attack transit simulation in a general facility 20
5 Simulation from impact distribution 21
6 Attack simulation in a general facility 22
7 Compute expected utility 24
8 Global scheme 25
9 MH step (function `MH`) 34
10 MH Defender APS to approximate ARA solution (funtion `DAPS`) 34
11 MH Attacker APS to approximate sample from $p_D(a)$ (function `AAPS`) 35
12 General algorithm to solve BAIDs using ARA and APS 45
13 Forward Algorithm 59
14 Backward Algorithm 59
15 Viterbi Algorithm 60
16 Ranking-and-Selection Heuristic 68
17 APS Approximation 70
18 Complete Monte Carlo Enumeration 71

Abbreviations

To facilitate reading, we provide a glossary with the abbreviations used throughout the thesis.

AAPS: Attacker APS
AAPS_red: Attacker APS reduction
AD: Augmented distribution
ADH: Analysis of discourse around hashtags
ADP: Attacker Decision Path
ADS: Automated driving system
AI: Artificial Intelligence
AIA: EU AI Act
AIRMF: AI Risk Management Framework
ALP: Adversarial logit pairing
ALTAI: Assessment List for Trustworthy AI
AML: Adversarial machine learning
AML_at: Adversarial attacks against ML algorithms
ARA: Adversarial risk analysis
APS: Augmented probability simulation
ASS: Analysis of spread sources
AT: Adversarial training
BAID: Bi-agent influence diagram
BTM3: Botometer v3
CIA: Confidentiality, Integrity, Availability
CME: Complete MC Enumeration
Cr: Criminal gang
CRFM: Center for Research on Foundation Models
CSRM: Cybersecurity risk management
CV: Cross-validation
CV20: Couce-Vieira et al. (2020)
Cy: Cyberterrorist group
DAPS: Defender APS
DAPS_red: Defender APS Reduction
DDP: Defender Decision Path
DE: Data Extraction
DC: Disinformation campaign
DHS: Defense and Homeland security
dp: direct predecessors
DoS: Denial of service
EP: Entrance probability
F: Forensics
FGSM: Fast gradient sign method
FwGw: Firewall and internet gateway
HA: Hashtag analysis
HMM: Hidden Markov Model
ID: Influence diagram
IDS: Intrusion detection system
ISO: International Organization for Standardization
KNN: K Nearest Neighbors
LLM: Large Language Model
MAE: Mean absolute error
MAID: Multi-Agent influence diagram
MC: Monte Carlo
MCMC: Markov Chain Monte Carlo
MCTS: Monte Carlo Tree Search
MH: Metropolis–Hastings
ML: Machine Learning
MLP: Multilayer perceptron
M: Monitoring
MoB: Mixture of Beta Distributions

MoW: Mixture of Weibull Distributions

NER: Named-Entity-Recognition
NLP: Natural language processing
PNP: Probability of not protecting

POS: Part-of-Speech
RAD: Random augmented distribution
RME: Random-Greedy MC enumeration
RMSE: Root mean square error
SCT: Supply chain threat
TAID: Tri-Agent influence diagram
V2V: Vehicle-to-Vehicle
WEF: World Economic Forum
wir_jam: Wireless jamming on the control plane

NIST: National Institute of Standards and Technology
NLL: Negative log-likelihood
NN: Neural networks
PmVs: Patch management, IDS and vulnerability scanner
R&S: Ranking-and-Selection
RAPS: Random APS
RM: Risk management
SA: Simulated annealing
SEA: Sentiment analysis
V2I: Vehicle-to-Infrastructure
VSL: Value of statistical life
W: Wordcloud

Chapter 1

Introduction

1.1 Motivation

In recent years, the digital transformation of society, accelerated by the rapid evolution of Artificial Intelligence (AI), has expanded our capabilities but also increased our vulnerabilities to cyberattacks. These attacks threaten not only our information-related assets but also our lives, our assets and our physical infrastructures. Reflecting this growing concern, cyber insecurity, misinformation and the negative consequences of advances in AI were recently ranked as top risks in the Global Risk Maps of the World Economic Forum (WEF, 2024; WEF, 2025b). Furthermore, this trend is not expected to decline. Rather, it is likely to intensify as governments, companies and individuals become increasingly dependent on information technology, thereby expanding the attack surface and amplifying both the number and impact of cyberattacks on critical societal systems and infrastructure.

A case that vividly illustrates the magnitude of this challenge is Sunburst (Tidy, 2020). Likely orchestrated by a national cyber-military group, the hackers accessed the computer networks of major US government agencies, and other organizations worldwide, by compromising a software update from SolarWinds, a supplier. The attack granted them access to sensitive and confidential information, highlighting how cyberattacks can be executed through a target organization's suppliers. The scale of this problem is exacerbated when considering that some companies and public administrations may rely on hundreds, or even thousands, of third parties, which cybercriminals could exploit to infiltrate their systems.

In this context, apart from driving groundbreaking advancements in sectors such as drug discovery (García et al., 2025), healthcare (Ordovás et al., 2023; Corrales et al., 2024), transportation (Caballero et al., 2023; Gu et al., 2021), finance (Doumpos et al., 2023), and education (Akpan et al., 2025), AI is also playing a crucial role in cybersecurity. In particular, it helps protect against vulnerabilities introduced by digitalisation; for example, Machine Learning (ML) models are increasingly being deployed to detect vulnerabilities, identify threats in information systems, and mitigate them before they can be exploited by attackers. Notable examples include detecting spam (Naveiro et al., 2019), and malware (Redondo and Insua, 2020).

Moreover, AI is also helping to mitigate disinformation (Saeidnia et al., 2025), a highlighted cyber threat in the ENISA Report (ENISA, 2022) and the last WEF Global Risks Map (WEF, 2025b). AI models enable the monitoring of vast amounts of data processed on social media, making possible early detection of disinformation crucial to maintain the trustworthiness and credibility of the content in these platforms. For instance, on Twitter (also known as X), false news stories spread significantly further, faster, deeper, and more broadly than truthful information across all types of content, including topics such as terrorism, natural disasters,

urban legends, and political news (Ardia et al., 2020). This is especially relevant as an increasing number of people get their news from Twitter (Walker and Matsa, 2021), creating an exploitable opportunity for malicious actors to manipulate public opinion or deceive unsuspecting users through disinformation. For example, this can be used as an element of foreign information manipulation and interference, posing a threat to the 16th Sustainable Development Goal, which claims for inclusive and peaceful societies (Bontcheva et al., 2020).

One type of these malicious agents is bots, which are accounts that share content and engage with other users through programmed or automated actions (Yang et al., 2022). Specifically on Twitter, bots have played a significant role in spreading low-credibility content (Shao et al., 2018) and have been crucial in the propagation of fake news (Ferrara, 2017). They have been present in the online discourse of several democratic processes (Pastor-Galindo et al., 2020), health crises (Unlu et al., 2025), and war conflicts (Breve et al., 2024). To address the potential risks posed by these agents, several ML-based bot detection techniques have been developed (Javed et al., 2024).

However, AI acts as a double-edged sword in cybersecurity: besides allowing for the automatic detection of cyberthreats, it also makes the cyberthreat landscape more complex (WEF, 2025a). Specifically, AI can be used to create sophisticated cyberattacks (Schroer et al., 2025), including advanced forms of disinformation (Mazurczyk et al., 2024). Moreover, these risks are exacerbated by the increasing speed of AI progress, as emphasized in the Center for Research on Foundation Models (CRFM) report (CRFM, 2021) on large language models (LLMs) and the International AI Safety Report (Bengio et al., 2025), which highlights emerging AI risks related to safety (ensuring that AI systems behave as expected) and security (protection against threats). Many real cases of safety-related risks can be found in the AI Incident Database (McGregor, 2021). One notable example is hallucinations in LLMs, that is, the generation of untruthful information that either conflicts with or cannot be verified by the available sources (Zhao et al., 2023). Regarding security, it is important to consider that although ML algorithms perform extraordinarily well with standard data, they are vulnerable to the so-called adversarial examples (Goodfellow et al., 2014), which are generated by applying small but carefully chosen perturbations to inputs, causing the models to confidently produce incorrect predictions. In fact, adversarial examples are among the main challenges faced in industrial ML deployment (Cinà et al., 2023): the implications of adversarial examples for domains such as autonomous driving systems, defense systems, law enforcement, and healthcare, to name just a few, could be dramatic.

Effectively responding to this increasingly complex cyber threat environment requires the implementation of a strong cybersecurity risk management (CSRM) framework. This framework should be grounded in risk analysis (Aven, 2019), enabling organizations to evaluate potential threats to their assets and identify appropriate strategies to mitigate both the likelihood and possible consequences of such threats. Nevertheless, many existing CSRM approaches have methodological limitations in risk assessment and management, particularly when addressing the new elements and issues that AI introduces into the CSRM scope, like the novel impacts that need to be considered.

In this PhD dissertation, we work at the intersection of CSRM and AI, providing contributions at both a broad and a specific level to that intersection. At the broader level, we aim to advance CSRM by offering methodological improvements to better handle systems that include AI-based systems. At a more specific level, we seek to contribute to certain components within CSRM by researching the new elements introduced by AI.

In the remainder of this chapter, before specifying the research objectives of the thesis and its structure (Section 1.5), we provide a review of CSRM (Section 1.2), in which we describe the building blocks of a CSRM scenario and highlight those that are affected by the introduction of

AI. We then discuss the novel issues brought by AI-based systems for CSRM (Section 1.3), which will serve to introduce Adversarial Risk Analysis (Section 1.4), as a fundamental pillar in this thesis. The topics covered in sections 1.2, 1.3 and 1.4 motivate and suggest the contributions of this thesis.

1.2 Cybersecurity risk management: a brief review

A general scenario that captures all key aspects of CSRM can be outlined as follows: an organization is exposed to potential cyber threats that could cause significant harm. To mitigate these risks, such organization must select a cybersecurity portfolio, which is a strategic mix of tools, including security and recovery controls as well as cyberinsurance products, all aimed at effectively managing cybersecurity risk. Figure 1.1 provides a representation of the general scenario through a diagram that highlights four key components: the *organization*, the *threats*, the *impacts*, and the *cybersecurity portfolio*. It specifically shows that:

- The *organization* is characterized by its *profile* and *assets*, along with a set of additional *organizational characteristics*.

- The *threats* component includes *non-targeted cyber*, *accidental*, and *environmental threats*. In addition, and key to this thesis, there are *targeted cyber threats*, where *attackers* intentionally focus their efforts on a specific organization.

- The *impacts* of a cyberattack can be categorized into *insurable impacts*, which are partially mitigated through insurance coverage, and *non-insurable impacts*, outside the scope of insurance protection.

- The *cybersecurity portfolio* comprises several elements, including *security controls*, or measures put in place by the organization to prevent, detect, and respond to cyberattacks. These controls help reduce the likelihood and/or the impact of the attacks and are typically classified into *procedural*, *technical*, and *physical*. The portfolio may also contain *recovery controls*, used to manage and recover from cyber incident impacts, thereby minimizing their consequences. Additionally, it includes *insurance controls* that enable the transfer of risk and help mitigate the financial impact of an attack.

A variety of frameworks have been created to aid in CSRM, such as ISO 27005 (International Organization for Standardization., 2022); CRAMM (CCTA, 2005), MAGERIT (Amutio et al., 2012), EBIOS (Agence Nationale de la Sécurité des Systèmes d'Information, 2024) at the European level; and the Risk Management Framework of the U.S. National Institute of Standards and Technology (NIST, 2018) to name but a few. Besides, several compliance and control assessment frameworks offer guidance on adopting cybersecurity best practices, such as the ISO 27001 (International Organization for Standardization, 2022) and the Cloud Controls Matrix (Cloud Security Alliance, 2024). These frameworks typically include comprehensive lists of threats, assets, and controls, along with practical instructions for safeguarding digital assets. Nevertheless, these previous frameworks remain limited for the CSRM in systems with AI components.

To shed light on AI-based systems, particularly regarding their safety and security, the EU recently approved its AI Act (European Parliament, 2024) (hereafter referred to as the AIA), the first law globally regulating AI. It classifies systems according to four risk categories (prohibited, high, limited, or minimal), with corresponding remedial actions (Madiega, 2021). However, implementing the AIA, especially its risk management (RM) activities, poses notable challenges,

as demonstrated when handling LLMs (Helberger and Diakopoulos, 2023). These challenges are related to the lack of a predefined purpose for some of these systems, combined with their scale of use, which makes it difficult to foresee the potential risks and allocate them into the predefined AIA risk categories. Besides, although risk analysis practices consider potentially emergent features of systems, the appearance of unexpected capabilities in AI systems and services introduce an additional layer of complexity. In this context, the NIST has also released an initial version of an AI Risk Management Framework (NIST, 2023) (hereafter referred to as AIRMF). From a policy perspective, it is evident that AI RM is attracting significant international attention (Agarwala et al., 2020), with a broad collection of new research directions on AI governance also being proposed (von Knebel and Anderljung, 2024).

However, most current approaches to RM in AI systems remain at a qualitative level, frequently adopting risk matrices to guide resource allocations and set priorities despite their well-known shortcomings (Cox, 2008). Additionally, they share similar limitations in adequately handling the new elements introduced to CSRM by AI systems, components, and services: *AI-related impacts*, *AI-based assets*, *AI-based targeted attacks*, and *AI-based security and recovery controls*. The black arrows in Figure 1.1 show the CSRM components that these elements belong to. These components are consequently influenced by the novel AI-related risk analysis issues introduced by the new elements. On the one hand, this thesis focuses on addressing these issues, which will be described in the next section, and implement a CSRM approach that integrates the proposed solutions to handle them, thereby providing a contribution to CSRM from a holistic perspective. On the other hand, we advance the understanding of the new elements introduced by AI in CSRM, since by doing so, we also contribute to the specific components of Figure 1.1 in which they are embedded.

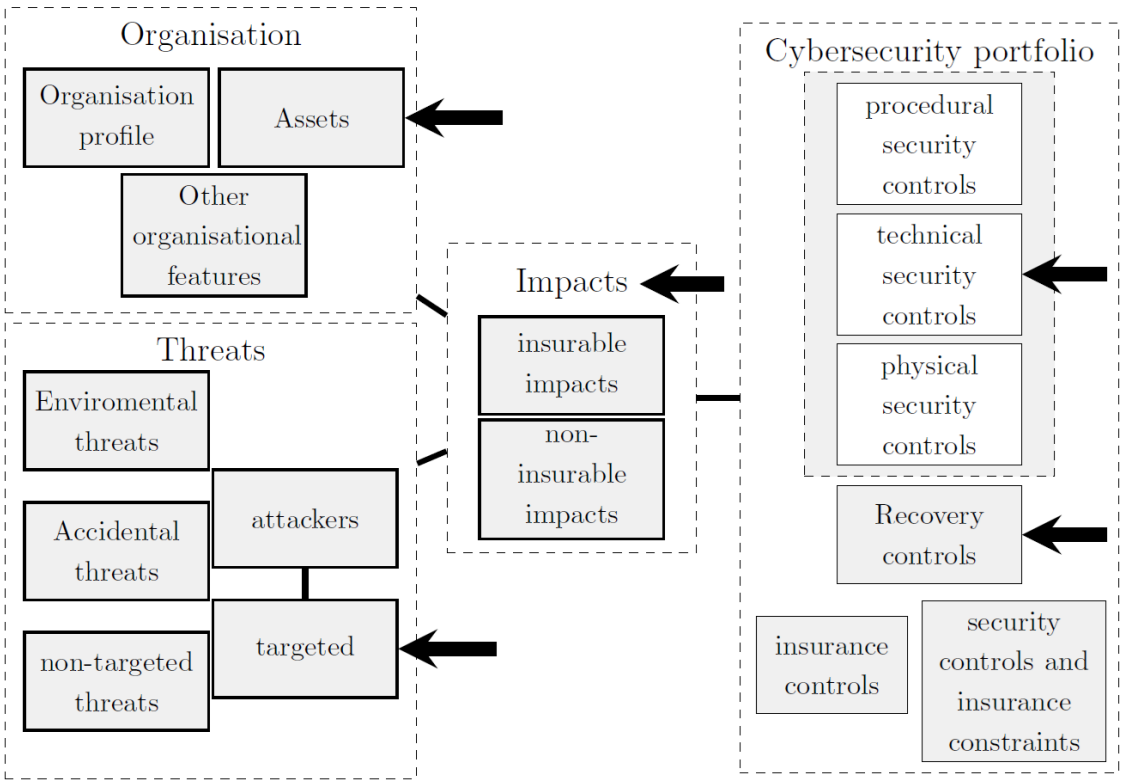


Figure 1.1: Detailed representation of the components involved in CSRM. Black arrows indicate the components affected by AI-based systems.

1.3 Novel AI-related risk analysis issues

As motivated, this section describes the risk analysis issues and challenges that arise in handling each of the new elements introduced into CSRM by AI, as reflected in Figure 1.1.

AI-related impacts

Given the potentially severe consequences of cyberattacks, including financial loss, reputational damage, and even human casualties, it is essential to clearly define the potential impacts of AI systems, as these will constitute the objectives or performance measures to be optimized to mitigate the risk. These objectives have been typically characterized through the CIA (Confidentiality, Integrity, Availability) triad (Ham, 2021). Although useful from a technical standpoint, this approach has become increasingly limited due to the growing diversity of cyberthreats. Notably, questions about safety, security, bias, privacy, and fairness have become a top concern in the digital world, as reflected e.g. in the Assessment List for Trustworthy AI (ALTAI) (Ala-Pietilä et al., 2020). Thus, there is a need to adjust, match, and, possibly, update earlier cybersecurity objectives to cover such new impacts.

The AIRMF, AIA, the OECD AI Recommendation (2019), the ALTAI list, and the CRFM report contribute in this regard. We can identify equivalent impacts among them, even though they use different names and structures to describe them. For instance, the AIRMF focuses on trustworthy AI, which should be *valid*, *reliable*, *safe*, *secure*, *resilient*, *accountable*, *transparent*, *explainable*, *interpretable*, *privacy-enhancing*, and *fair*, with each feature comprising additional subfeatures. Beyond these, the AIRMF identifies potential impacts associated with AI risks, such as *harm to personal rights*. By contrast, the AIA is more explicit in identifying prohibited practices and high-risk AI systems. For example, one forbidden functionality is to *employ subliminal techniques beyond a person's consciousness to materially distort their behavior*. These differences in names and structures do not undermine the substance of the frameworks, but they can make direct comparison and alignment across them more challenging. Moreover, these frameworks remain at a qualitative level and fail to provide a structured approach for decision support purposes. Thus, it becomes imperative to adopt a comprehensive approach that integrates the existing frameworks and provides a quantitative indicator to support decisions aimed at mitigating these impacts.

AI-based assets

A growing number of systems include AI-based components susceptible to specialized attacks. Automated Driving Systems (ADSs) constitute a prime example; they have perception systems as core elements, typically based on convolutional neural networks (Gallego and Insua, 2022), which have been the target of malicious attacks intended to deceive them into making incorrect predictions inducing in turn misguided decisions, potentially resulting in catastrophic consequences. Modeling approaches for these systems such as the one in Insua et al. (2021a), which uses a single block to model a cyber organization, are clearly limited. Therefore, it seems necessary to develop a flexible modeling approach that can individually represent AI assets, capture their dependencies on other system components, and account for vulnerabilities to external attacks, in order to effectively manage and mitigate the risks associated with these systems.

AI-based targeted attacks

As in other cyber domains, attacks can be generic or targeted. In this context, we focus on targeted attacks that leverage AI techniques, such as when attackers employ tainted images to circumvent an image recognition system (Rios Insua et al., 2023). These activities are becoming increasingly relevant internationally given the emergent AI-based Crime-As-A-Service platforms (Matejic and Wilson, 2024). However, note that whether targeted attacks are automated through AI systems or manually crafted make little difference in modeling terms and we would just need to assess if the attackers' capabilities enable them to implement more sophisticated AI attacks and/or have sufficient resources to hire an AI-based attacking platform.

Security games serve as a valuable tool for analyzing these cybersecurity scenarios involving attacks on AI systems, as they facilitate the modeling of strategic and operational defense and homeland security (DHS) problems involving multiple agents, as cogently argued by Brown et al. (2006), Zhuang and Bier (2007), and Hausken (2024). These authors illustrate their analyses mainly from a standard game-theoretic perspective, which is based computationally on approximating Nash equilibria and related refinements. However, these approaches have limitations, as they rely on a *common knowledge assumption*, critically reviewed in e.g. Raiffa et al. (2002) and Hargreaves-Heap and Varoufakis (2004), which frequently assumes that agents know not only their own payoffs, preferences, beliefs, and feasible actions, but also those of their opponents. The *common knowledge assumption* is not realistic in the DHS realm, given that agents will generally lack such knowledge, as adversaries often hide information and even deliberately disinform opponents. Adversarial Risk Analysis (ARA) (Banks et al., 2022) offers an alternative approach to solving security games that partly mitigates this assumption, making it more suitable for use within the CSRM domain than Game Theory. Therefore, advances in ARA methodology will contribute to a more appropriate modeling of AI-based targeted attacks and support improved decision-making to counter these threats.

AI-based security and recovery controls

As the AIA stresses, many systems include AI components as part of their security and safety controls, susceptible of being outsmarted with malicious purposes. For instance, malware detectors (Redondo and Insua, 2020) are plagued by obfuscated malware. AI-based recovery controls may be affected as well by attacks; as an example, an AI-based backup system that depends on an intrusion detection system (IDS) may be deceived by adversarial attacks during operations (Rios Insua et al., 2023), leaving it vulnerable to ransomware designed to encrypt the backup database.

Adversarial Machine Learning (AML) is an emerging field that studies the vulnerabilities and robustness of ML models against adversarial attacks. AML focuses on modeling threats to ML algorithms, understanding their potential impacts, and developing defenses against the associated negative effects. This area of study is incredibly relevant given that AI is already used by millions of people and its adoption continues to increase (Maslej et al., 2025). Although AML is a relatively nascent field, much progress has been made in the last twenty years since Dalvi et al. (2004) published the discipline's seminal paper on adversarial classification. In particular, the tools and breakthroughs of AML could be integrated into CSRM approaches to better determine which security controls are most appropriate to implement within a cybersecurity portfolio.

AML scenarios, which can be framed as security games, have typically been solved using game-theoretic approaches. However, as mentioned previously, these methods rely on *common knowledge assumptions*, which is not appropriate in the AML domain. As before, ARA offers an alternative, helping to weaken this assumption and thereby address AML problems more

adequately. Given the potential of ARA to contribute both in the *technical security controls* and *targeted threats* components at their intersection with AI, it will play a central role in this thesis. We consequently provide a brief overview of ARA.

1.4 Adversarial Risk Analysis

ARA (Banks et al., 2022) is a decision-theoretic framework that combines risk analysis and game theory to support a single decision-maker, known as the defender, in adversarial or competitive settings. It is designed for scenarios in which intelligent opponents, called attackers, may act strategically against the defender.

ARA has the following conceptual differences with Game Theory. Firstly, it partly mitigates the *common knowledge assumptions* regarding payoffs, preferences, beliefs, and feasible actions, by adopting a Bayesian approach: modelling the attacker's problem from the defender's perspective and using probabilistic distributions to capture uncertainty about the attacker's unknown elements. ARA thus operationalizes a Bayesian approach to games, as sketched by Kadane and Larkey (1982) and Raiffa et al. (2002). Secondly, instead of seeking Nash equilibria among all agents or related refinements, ARA aims to identify a decision that maximizes the defender's expected utility. Operationally, ARA transforms games into a simulation-optimization process that typically follows these steps:

1. *Formulating the defender's problem*, identifying all its relevant elements.

2. *Formulating the attacker's decision problem* by gathering partial information about the attacker's beliefs and preferences, leading to random utilities and probabilities (from the defender's perspective).

3. *Simulating the attacker's problem* by sampling from the random utilities and probabilities, assessing the attacker's optimal actions, thereby obtaining a probabilistic forecast of the attacker's behavior.

4. *Incorporating these forecasts into the defender's problem* (step 1) to obtain the decision that maximizes the defender's expected utility.

Influence diagrams (IDs) are used to visually represent the decision-making problems analyzed in ARA (Banks et al., 2015). In particular, we shall use the general term Multi-Agent Influence Diagram (MAID) for an ID that describes a decision situation involving multiple agents. More specifically, we refer to Bi-Agent Influence Diagrams (BAID) for problems involving two agents, and Tri-Agent Influence Diagrams (TAID) for those involving three agents. These diagrams include four types of nodes:

- *Chance or Uncertainty Nodes*: Represent uncertainties that need to be captured in the model. They are typically shown as *circles* or *ovals*.

- *Deterministic Nodes*: A form of uncertainty node in which outcomes are deterministically determined by input values. They are represented through *double ovals*.

- *Decision Nodes*: Represent decisions that the agent must make from a set of alternatives. They are depicted as *squares* or *rectangles*.

- *Utility, Value or Preference Nodes*: Capture agents' preferences, modelled through a utility function. These nodes are shown as *hexagons*.

Nodes associated with a single agent will be displayed in solid colours, with each agent assigned a distinct colour. In contrast, nodes relevant to multiple agents will be depicted using striped patterns. The diagrams also feature two types of arcs:

- *Dashed arcs*, pointing to decision nodes, indicate that the agent makes the corresponding decision knowing the values of all preceding nodes.

- *Solid arcs*, pointing to chance or preference nodes, indicate that the values of these nodes (such as probabilities or utilities) depend on the values of the preceding ones.

In this thesis, we primarily focus on bi-agent problems; hence, we will use BAIDs to represent the corresponding decision-making scenarios. We illustrate their logic with the following examples that will be used and analyzed in later chapters. Figure 1.2 displays respectively the BAIDs for 1-stage sequential (Left) and simultaneous (Right) defend-attack games. In the sequential case (Figure 1.2 Left), the defender selects a defense $d \in \mathcal{D}$, where $\mathcal{D}$ is the set of feasible alternatives. After observing d, the attacker chooses an action $a \in \mathcal{A}$, with $\mathcal{A}$ representing the set of available alternatives. The arc $D \to A$ indicates that the attacker observes the defender's decision. Both d and a influence a random outcome $\theta \in \Theta$, where Θ is the space of possible outcomes. The defender's utility function u_D depends on her chosen defense and the outcome, while the attacker's utility (u_A) depends on a and θ. In contrast, the simultaneous case (Figure 1.2 Right) follows the same semantics, except that the arc $D \to A$ is absent, reflecting that both agents act knowing each other's decision.



Figure 1.2: BAIDs for (Left) sequential and (Right) simultaneous defend-attack games.

From a BAID, we can study the decision problem from each agent's perspective. Let us consider the sequential defend-attack game in Figure 1.2 (Left). When examining the defender's problem (Figure 1.3 Left), the attacker's decision node A appears as an uncertainty from the defender's viewpoint. To address her problem, the defender analyzes the problem from the attacker's standpoint (Figure 1.3 Right). In this case, node D in Figure 1.2 (Left) becomes a chance node for the attacker, modeling his uncertainty about the defender's intentions.

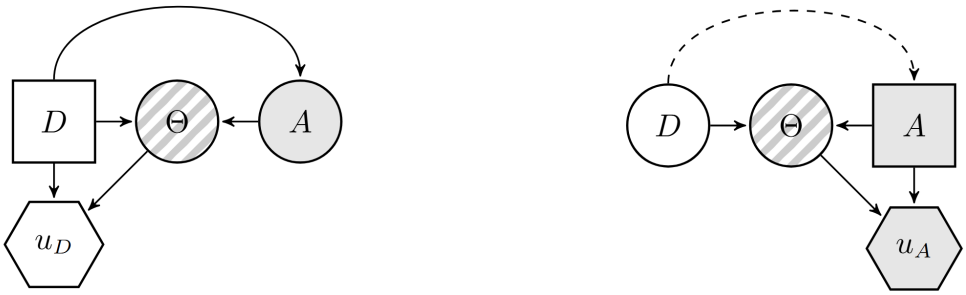


Figure 1.3: (Left) defender's decision problem; (Right) defender's analysis of attacker's problem.

1.5 Thesis objectives and structure

As previously discussed, this thesis contributes to CSRM from both a broad and a specific perspective by concretely addressing certain components detailed in Figure 1.1. First of all, from a broader perspective, we intend to implement a general CSRM framework for systems with AI components and services that handles the issues described in Section 1.3. At a more specific level, this thesis advances selected topics concerning the *technical security controls* and *targeted threats* components at their intersection with AI.

In particular, we aim to address the *technical security controls* component from two different perspectives. First, we study the vulnerabilities of Hidden Markov Model (HMM), a ML model that has been largely unexplored from an AML standpoint using an ARA approach. Research on this understudied method is essential for understanding its weaknesses and improving its robustness. Traditional AML relies on game-theoretic analysis and its common-knowledge assumption; adopting an ARA approach relaxes this assumption, offering significant potential for AML, as recent research shows (Naveiro et al., 2019; Rios Insua et al., 2023). Second, we aim to advance disinformation detection on social networks by focusing on ML models for Twitter bot detection. Given the evolution of bots (Cresci, 2020) and their proliferation, it remains crucial to refine detection methods and address the limitations of current models. Moreover, in addressing *targeted threats*, we focus on enhancing ARA solution methods. Since ARA provides a promising framework for managing uncertainty in security domains, developing algorithmic approaches to solve security games through ARA is especially relevant. This has potential applications in AML, as it allows for a deeper analysis of ML vulnerabilities. Thus, this line of research contributes not only to the *targeted threats* component, but also potentially strengthens *technical security controls*.

The main contributions of this thesis are summarized in the objectives below:

O1. A cybersecurity risk analysis framework for systems with artificial intelligence components. As described previously, AI introduces new challenges regarding CSRM, and both the AIA and NIST AIRMF call for a better understanding and implementation of novel risk analysis approaches when handling systems with AI components. Chapter 2 develops a flexible, broad CSRM framework for systems including AI services and components dealing with the issues in Section 1.3. An example concerning automated driving systems illustrates and conceptually validates the framework. This objective represents the broad contribution to CSRM outlined earlier.

O2. A computational adversarial risk analysis methodology for general security games with two agents. We provide a unified approach capable of solving security games of two agents using ARA, irrespective of the number of decisions per agent, and applicable

across both discrete and continuous decisions domains. The proposed framework can be applied to security scenarios where common knowledge assumptions are unrealistic, as in AML. The methodology is exemplified through a disinformation war example. It is covered in Chapter 3, advancing the *targeted threats* component, with potential applications in the *technical security controls* component.

O3. Development of algorithmic methodologies to attack HMM inference by corrupting input data. HMMs represent an understudied area within the AML literature. To address this, we study how their inferences can be manipulated by leveraging an ARA approach. The development of such attacks is paramount to ensure the security of canonical HMM algorithms when the veracity of the input data is threatened; their weaknesses must first be identified before they can be strengthened. This is the subject of Chapter 4, contributing to the *technical security controls* component.

O4. A comparison on how different ML models perform in detecting bots with diverse behaviors and domains on Twitter. In Chapter 5, we implement several supervised detection models and compare their characteristics, including their data requirements and their ability to detect bots with different behaviors. One of the implemented models is used in two case studies: first, to study bot presence during the COVID-19 pandemic, and second, to design an application prototype that analyzes the content spread by bots for users with non-technical expertise, providing these users with a tool to study this cyber threat. This objective expands the *technical security controls* component.

Finally, Chapter 6 summarizes the main contributions of the thesis and suggests future research directions to build upon this work.

Chapter 2

A cybersecurity risk analysis framework for systems with artificial intelligence components

2.1 Introduction

In line with objective **O1**, this chapter introduces a cybersecurity risk analysis framework for systems with AI components. The proposed framework, which builds on Insua et al. (2021a) and Insua et al. (2021b), constitutes the thesis's contribution to CSRM methodology from a broad perspective.

In particular, the framework presented is designed to support the selection of optimal protection portfolios for AI-based systems. It adopts a quantitative risk analysis approach, aiming to provide a more rigorous methodology than prevailing CSRM practices that rely on qualitative techniques such as risk matrices. The framework can be applied to a wide range of systems due to its flexible and granular modelling of cyber-architectures. Moreover, it integrates ARA elements to address uncertainties associated with potential targeted attacks. The framework aligns with the requirements of AIA, AIRMF, and related regulatory documents, thereby strengthening RM practices for systems incorporating AI components. Its applicability is demonstrated through a case study on ADSs, which illustrates its role in guiding risk analysis and decision-making, while also serving as a template for the assessment of other AI-based systems.

The chapter is organized as follows. Section 2.2 presents constructive solutions to the issues of standard risk analysis introduced by AI systems, components, and services as outlined in Section 1.3. Then, Section 2.3 integrates these solutions into a comprehensive CSRM framework for determining the optimal portfolio to protect AI-based systems. Section 2.4 illustrates the proposed methodology through an ADS case study. Finally, Section 2.5 concludes with a brief discussion.

2.2 Solutions to novel AI-related risk analysis issues

This section suggests methodological approaches to address the risk analysis issues outlined in Section 1.3, so they can be incorporated into a methodology to better protect AI systems. Each subsection details how to handle the complexities that each specific novel AI-related element brings in, providing the pillars needed to integrate them into a CSRM framework, as will be done in Section 2.3. Where relevant literature is available, we primarily provide summaries along with appropriate pointers.

2.2.1 AI-related impacts

As Section 1.3 highlighted, it is necessary to revise, align, and possibly expand existing catalogs of cybersecurity objectives to better reflect the impacts of cyberattacks, as well as to use a quantitative approach to measure alignment with these impacts and support decision-making.

To address this, we stem from Couce-Vieira et al. (2020) (CV20), who proposed a comprehensive cybersecurity perspective that incorporates both business and societal considerations. Their approach is structured as a cybersecurity objectives tree (Figure 2.1), developed to support decision-making by integrating objectives using a multi-attribute utility function tailored to decision-support needs.

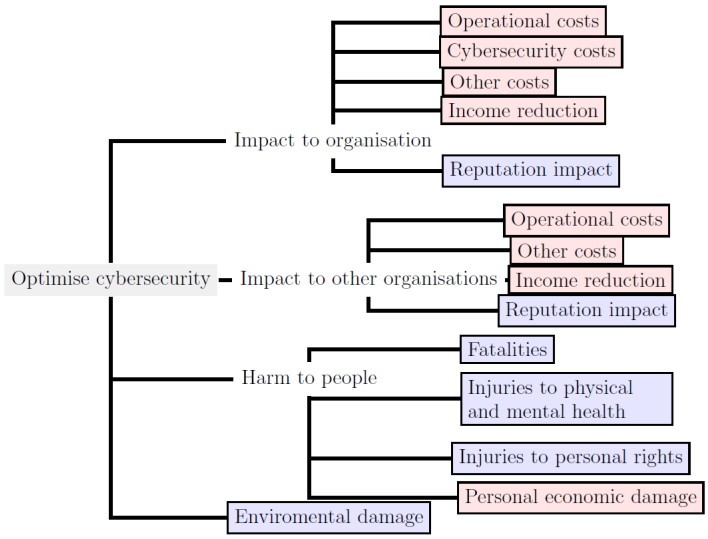


Figure 2.1: Cybersecurity objective tree. Red, monetary objectives; blue non-monetary objectives. Adapted from (Couce-Vieira et al., 2020).

The tree includes objectives that can be measured in monetary terms, such as operational costs, as well as others that are not directly measurable in such terms, like reputation or fatalities. Objectives that are not directly measurable in monetary terms must still be quantified for use in RM models. To achieve this, the main scenarios that could lead to different types of cyber incidents are identified. These hypothetical scenarios should be exhaustive, meaning they encompass all feasible types of impacts related to the corresponding objective, considering the stakeholders, assets and organizational activities potentially affected by an attack. Once these scenarios are identified, they must be quantified. This is done using a natural attribute when one is available and sufficiently straightforward to assess. When a natural attribute is not suitable, a constructed, indirect, or proxy attribute should be used instead (Keeney, 1992).

Interestingly, with appropriate modifications, the CV20 list is capable of covering the novel impacts identified in the AIRMF, AIA, ALTAI, and CRFM reports emphasizing people's rights and physical risks, in contrast to cybersecurity frameworks from previous decades. Indeed, we can map these impacts to the CV20 objectives and leverage the fact that the CV20 tree was designed for decision-support purposes to construct a multi-attribute utility function that integrates the relevant objectives, providing a quantitative way to measure how effectively a system meets the corresponding cybersecurity goals.

Table 2.1 presents AI-related impacts from several lists, organized according to the AI-trustworthiness principles explicitly or implicitly expressed in these documents and their corresponding mappings to the objectives in the CV20 tree. To address specific AI-related impacts on a system, we can trace the mapping to the CV20's objectives and use that information to build a multi-attribute utility function for selecting the optimal portfolio to protect the system.

Table 2.1: Mapping trustworthiness features into cybersecurity objectives. Indices in terms identify references addressing the principle. Indices in definitions identify the source. References for indices are: (a) NIST AIRMF (NIST, 2023), (b) CRFM's On the Opportunities and Risks of Foundation Models (CRFM, 2021), (c) EU AI Act (European Parliament, 2024), (d) Article 21, EU Charter of Fundamental Rights (European Parliament et al., 2012), (e) ISO/IEC TS 5723:2002 (ISO, 2022), (f) ALTAI list (Ala-Pietilä et al., 2020).

Trustworthiness principles and subprinciples		Map to CV20 tree
Principle	Subprinciple	
Validity[a]	**Accuracy**[a,f]: "closeness of results of observations, computations or estimates to true values or values accepted as being true." [e]	Failing to achieve these characteristics involves a degradation and maybe a malfunction or unavailability of the AI system. These impacts may affect goals such as operational costs, income reduction, other costs (including non-compliance costs), reputational impact or AI RM costs.
	Reliability[a,f]: "ability of an item to perform as required, without failure, for a given time interval, under given conditions." [e] ALTAI mentions reproducibility together with reliability [f]	
	Robustness[a,b]: "ability of a system to maintain its level of performance under a variety of circumstances." [e]	
Safety[a,b,f]: "property of a system such that it does not, under defined conditions, lead to a state in which human life, health, property or the environment is endangered." [e]		These impacts affect goals related to harm to people: fatalities or injuries to physical and mental health. Indirectly, it may also create environmental, social [rights] or economic damage [f] to persons, the organization or third parties.
Fairness[a,b]	**Bias is managed**[a,b,f]: "systemic, computational, and human [bias], all of which can occur in the absence of prejudice, partiality, or discriminatory intent." [a]	Impact on this characteristic may affect the reputation impact goal (ethical degradation) and other costs (non-compliance with AI regulation).
	Diversity[a,b]: "reflect demographic diversity and broad domain and user experience expertise." [a]	Impact on these characteristics may affect goals related to injuries to personal rights, personal economic damage and even injuries to physical and mental health. These include the impact of AI on work and skills and on society at large or democracy [f]. These consequences may also impact the organization in terms of reputation (ethical degradation) and costs.
	Risk of discrimination is minimized[b,c]: "based on any ground such as sex, race, color, ethnic or social origin, genetic features, language, religion or belief, political or any other opinion, membership of a national minority, property, birth, disability, age or sexual orientation." [d]	
	Accesibility[f]: "AI systems should be user-centric and designed in a way that allows all people to use AI products or services, regardless of their age, gender, abilities or characteristics. Accessibility [...] for persons with disabilities [...] is of particular importance." [f]	
Human agency[f]: "effect of AI systems that are aimed at guiding, influencing or supporting humans in decision making processes. [...] It also deals with the effect on human perception and expectation when confronted with AI systems that 'act' like humans. Finally, it deals with the effect of AI systems on human affection, trust and (in)dependence." [f]		
Security[a,b,f]	**Protection**[a,b,f]: protocols to avoid, protect against, respond to, or recover from attacks." [a]	These characteristics are represented by the RM controls, affecting also the RM costs goal.
	Resilience[a,f]: "ability to return to normal function after an unexpected adverse event." [a]	

Table 2.1 (cont)

Trustworthiness principles and subprinciples		Map to CV20 Tree
Principle	Subprinciple	
AI Governance	**Transparency**[a,b,f]: "the extent to which information is available to individuals about an AI system, if they are interacting – or even aware that they are interacting – with such a system." [a]	Failing to achieve these characteristics involves a reputational impact (ethical degradation) and may include other costs for non-compliance.
	Traceability[f]: "data and processes that yield the AI system's decisions, is properly documented to allow for traceability, increase transparency and, ultimately, build trust in AI in society." [f]	
	Accountability[a,b,c,f]: "expectations of the responsible party in the event that a risky outcome is realized." [a]	
	Human oversight[c,f]: "natural persons can effectively oversee the AI system during the period in which the AI system is in use." [c]	
	Stakeholder participation[f]: "it is advisable to consult stakeholders who may directly or indirectly be affected by the AI system throughout its life cycle." [f]	
	Auditability[f]: "the existing or necessary level that would be required for an evaluation of the AI system by internal and external auditors. The possibility to conduct evaluations as well as to access records on said evaluations can contribute to Trustworthy AI." [f]	
Understandability	**Explainability**[a,b,f]: "representation of the mechanisms underlying an algorithm's operation." [a]	Failing to achieve these characteristics involves a degradation and maybe a malfunction or unavailability of the function performed by the AI system. Impacts on these may affect goals of the organization such as operational costs, income reduction, other costs (including non-compliance costs), reputational impact or RM costs.
	Communication[f]: "the AI system's capabilities and limitations have been communicated to the users in a manner appropriate to the use case at hand" [f]	
	Interpretability[a,b]: "the meaning of AI systems' output in the context of its designed functional purpose." [a]	
Data governance[b,f]	**Data management**[c]: "data collection, data analysis, data labeling, data storage, data filtration, data mining, data aggregation, data retention and any other operation regarding the data." [c]	
	Privacy[a,b,f]: "safeguard human autonomy, identity and dignity." [a]	Privacy affects the goal of injuries to personal rights and may include other costs for non-compliance.

2.2.2 AI-based assets

An increasing number of systems incorporate AI components that are vulnerable to targeted attacks, as exemplified by ADSs. This highlights the need for a modelling methodology capable of appropriately capturing such complexity, see Section 1.3.

Given the increasing relevance of AI components, we propose to structure organizations with a finer granularity through blocks or components. These could be hardware, software, or hardware-software elements, some of them of AI nature. Consider thus a cyber organization structured according to blocks in levels depending on access, security, and criticality. For instance, when arranging a system according to criticality, those blocks in the first level (level-1) are considered less critical than those in the second one (level-2), which in turn are less critical than those in the third one (level-3), and so on. The links between blocks indicate potential channels through which an attack could enter the system or propagate. Three examples illustrate how cyber organizations are structured, with the first providing the basis for the case in Section 2.4.

Example 1. Fig. 2.2a presents a simplified version of an ADS architecture composed of two level-1 blocks, Perception (P) and Location (L), and one level-2 block, Decision Making (D). The P block covers sensors (LIDAR, Vehicle-to-Vehicle (V2V),...) and processing subsystems allowing the ADS to determine its relative location within the traffic scene; the L block covers systems (GPS, RFID,...) that estimate the vehicle's global position; D contains applications processing data to predict the behavior of nearby obstacles and decide the ADS direction and acceleration. Only attacks through level-1 blocks are possible for the specified attack type a in this case.

Fig. 2.2b depicts a threat assessment system (Theodosiadou et al., 2023), viewable as well as a decomposition of the P block in Fig. 2.2a. The video recorded by a block with cameras (level-1) is processed by a block with computer vision algorithms (level-2) leading to a risk estimate using a threat assessment block (level-3). Again, the attacks may access the system only through the level-1 block.

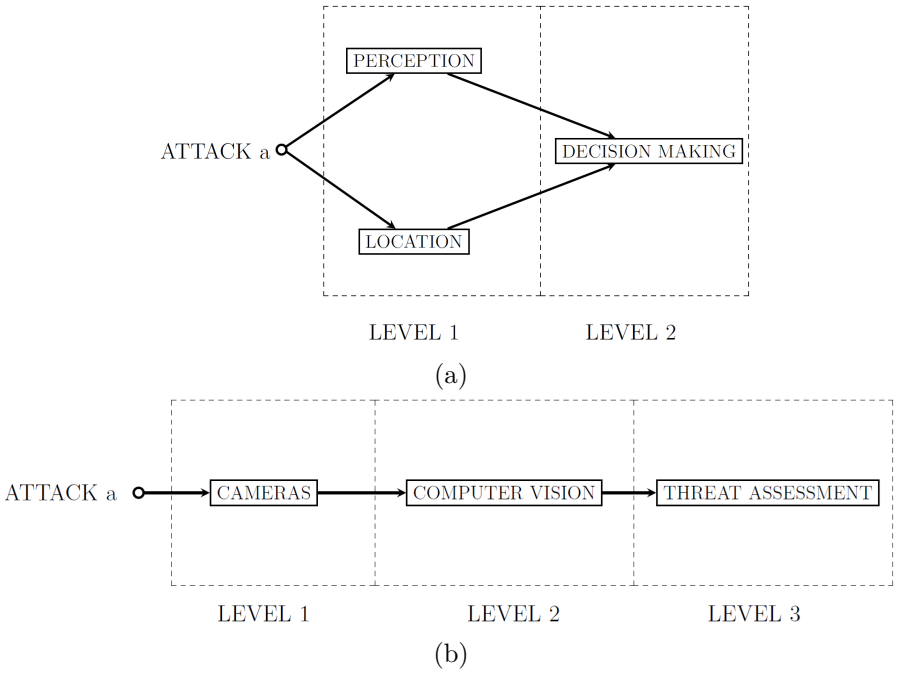


Figure 2.2: (a) ADS, and (b) Threat assessment architectures.

Finally, Fig. 2.3 displays a marketing content application architecture. Its interface (level-1) obtains features from a customer. Then, a *lookalike* block (Ma et al., 2016) using a database, determines users similar to the input one. Both level-2 blocks feed an LLM (level-3) to write personalized messages to users retrieved through the lookalike model. Observe that in this architecture, the attack might penetrate through the interface or the database and there is a cycle in its graph.

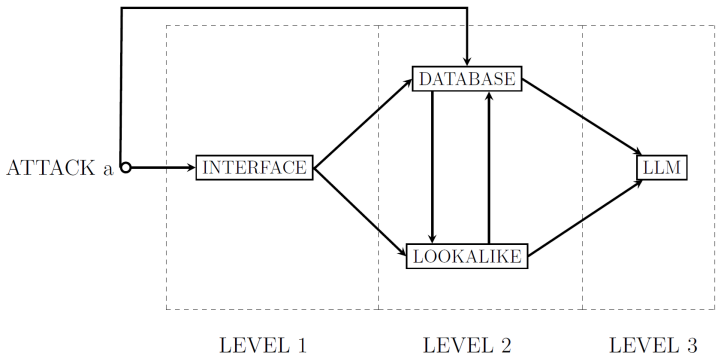


Figure 2.3: Marketing content creation system architecture. Four blocks in three levels. △

A major issue when modeling the architecture in blocks and levels will be to decide upon its

granularity: more detail entails better system understanding yet more intense modeling and computational effort. Besides, sometimes a block decomposition will not be possible when associated with a non-open source module and, hence, has to be treated as a black-box, as with the LLM block in Fig. 2.3 (assuming it corresponds to a non-open source LLM).

Formally, let G be the underlying graph representing the system, where nodes correspond to block components and edges represent possible paths through which an attack could enter or be transferred. The graph may include cycles within a same level. The parameters describing the system are defined as follows, where, for the moment, assume there is only one type of attack. First, the system is divided into k hierarchical levels, with a certain number of blocks assigned to each level. Blocks are indexed in ascending order, and if one block logically precedes another within the same level, it is assigned a smaller index. Specifically, level 1 contains i_1 blocks labeled $(B_1, \ldots, B_{i_1})$; level 2 contains the next $i_2 - i_1$ blocks, $(B_{i_1+1}, \ldots, B_{i_2})$; and so on, until the k-th level, which includes $i_k - i_{k-1}$ blocks labeled $(B_{i_{k-1}+1}, \ldots, B_{i_k})$. Some of these blocks may correspond to AI-based assets, as our examples reflect.

Next, the *entrance probabilities* (EP), that is the probabilities with which an attack will attempt to enter through various blocks, need to be assessed (some could be 0): $P_1 = (p_1, p_2, ..., p_{i_k})$ of the attack entering just through one block; $P_2 = [p_{12}, p_{13}, ..., p_{1i_k}, p_{23}, ..., p_{2i_k}, ..., p_{ij}, ..., p_{i_k-1,i_k}]$ just through two blocks simultaneously; ...; $P_k = (p_{12\cdots i_k})$ of entering simultaneously through all the blocks, with $\sum_i p_i + \sum_{i<j} p_{ij} + ... + p_{12\cdots i_k} = 1$. P designates the union of $P_1, P_2, \ldots, P_{i_k}$. Section 2.2.3 discusses issues concerning the assessment of attack probabilities, emphasizing targeted AI attacks.

The last relevant parameters describing the system are the *probabilities of not protecting* (PNPs), which represent the likelihood that an attack will successfully affect a given block. We differentiate between the probabilities $Q^0 = [q_i]$, which denote not protecting block B_i from an external attack, and $Q^s = [q_{ij}]$ for level s ($s = 1, 2, \ldots, k$), which contain the PNPs of block B_i against attacks originating from B_j, with j at level s or $s - 1$. Consequently, the probabilities of protecting B_i from an external attack or from an attack originating from block B_j are $1 - q_i$ and $1 - q_{ij}$, respectively. The PNPs depend on the type of defense considered, such as the AI-based defense referred to as *safety components* within the AIA (see Section 2.2.4 for details). We use Q to denote the complete set of PNPs.

Example 1 (cont). Focusing solely on the ADS architecture in Fig. 2.2a, we have: the EPs p_P, p_L just through blocks P and L ($p_D = 0$); $p_{P,L}$ the EP through both blocks (and, $p_{L,D} = p_{P,D} = 0$) (hence, $p_L, p_P, p_{P,L} \geqslant 0$ and $p_L + p_P + p_{P,L} = 1$); q_P, q_L are the block PNPs if they are attacked externally. $q_{D,P}$ ($q_{D,L}$), represents the PNP of D from P (L). △

The adopted structure in levels and blocks resembles that of the classical Purdue representation (Williams, 1994), as well as, with the probabilistic parameters included, Bayesian attack graphs (Liu and Man, 2005). However our representation is closer to the actual cybersecurity architecture, tends to be more parsimonious, and allows for cycles, providing a more flexible method to model cyber systems including AI components.

2.2.3 AI-based targeted attacks

As discussed in Section 1.3, attacks can be either targeted or generic (i.e., untargeted), and it makes little difference in modeling terms whether the attack is automated by AI or manually crafted. The generation process of each attack, whether targeted or untargeted, will be modeled by a stochastic process λ describing the number of attacks arriving during the required risk analysis planning period. We can use data and/or expert judgment to estimate and update the

parameters of this arrival stochastic process; see e.g. Hanea et al. (2021).

Moreover, as previously outlined in Section 2.2.2, the key architectural inputs are the EPs p_i of blocks B_i for a given attack. In the case of untargeted attacks, these can be estimated and updated using data and expert judgment, as with the stochastic process λ. However, targeted attacks require specific attention since they entail strategic calculations complicating the assessment of EPs. In our case, we appeal to ARA models, overviewed in Section 1.4, in which we aim to select the cybersecurity portfolio to minimize the potential impacts of attacker's actions. The uncertainties include whether our system will be targeted, and if so the potential entry points, or if, alternatively, another system will be the focus, taking into account the implemented portfolio. ARA can address this issue by considering that a system is targeted when an attacker obtains the highest expected utility from perpetrating such an attack. An important byproduct of this analysis is the determination of the EPs.

To wit, the scenario is modeled as a sequential defend-attack game (Ríos and Insua, 2012), with attackers operating as expected utility maximizers possessing knowledge of the implemented protective portfolio before initiating action. Assume that the adversary can perpetrate the following attacks compiled in set $\mathcal{A}$: 1) Attack of j-th type targeting the system's k-th entry, $a^1_{(j,k)}$, $j \in \{1,\ldots,J\}, k \in \{1,\ldots,K\}$ with J attack types and K entry combinations; 2) Attack of j-th type targeting another system i, $a^i_j, j \in \{1,\ldots,J\}, i \in \{2,\ldots,n\}$, with this set encompassing all other targets that the attacker may choose. We assume that the attacker will select only one type of attack at any given moment. Then, the attacker's expected utility ψ_A when he opts to target our system via $a^1_{(j,k)}$, given the cybersecurity portfolio $\mathbf{c}$, is

$$\psi_A(a^1_{(j,k)},\mathbf{c}) = p_A(Y=1|a^1_{(j,k)},\mathbf{c})u_A(Y=1,a^1_{(j,k)},\mathbf{c}) + p_A(Y=0|a^1_{(j,k)},\mathbf{c})u_A(Y=0,a^1_{(j,k)},\mathbf{c}),$$

where $p_A(Y=1|a^1_{(j,k)},\mathbf{c})$ denotes the adversary's probability that the attack successfully penetrates the system, and $u_A(Y=1,a^1_{(j,k)},\mathbf{c})$ is the adversary's utility of a successful attack of that type. Due to limited knowledge of the adversary's utilities u_A and probabilities p_A, we opt for a Bayesian approach modeling such uncertainty through random utilities U_A and probabilities P_A, leading to the adversary's random expected utility

$$\Psi_A(a^1_{(j,k)},\mathbf{c}) = P_A(Y=1|a^1_{(j,k)},\mathbf{c})U_A(Y=1,a^1_{(j,k)},\mathbf{c}) + P_A(Y=0|a^1_{(j,k)},\mathbf{c})U_A(Y=0,a^1_{(j,k)},\mathbf{c}).$$

Similarly, the random expected utility when the attacker targets the i-th system with attack a_j is $\Psi_A(a^i_j) = P_A(Y=1|a^i_j)U_A(Y=1,a^i_j) + P_A(Y=0|a^i_j)U_A(Y=0,a^i_j)$. Note that, in this, we do not account for changes in defenses of other systems, recognizing a lack of detailed knowledge about the security planning of the other systems.

Then, given $\mathbf{c}$, and using Algorithms 1–3 below, we can compute the following elements, which will allow us to simulate the targeted attacks:

- Distribution of attacker behaviour, $\tau(\mathbf{c}) = (\tau^1_1(\mathbf{c}),\ldots,\tau^1_J(\mathbf{c}),\, \tau^i(\mathbf{c}))$. $\tau^1_j(\mathbf{c})$ represents the probability that the attacker targets our system with attack j given $\mathbf{c}$. Similarly, $\tau^i(\mathbf{c})$ represents the probability of the attacker targeting another system. $\tau(\mathbf{c})$ will serve as the input for sampling from a multinomial distribution to determine whether the attacker targets our system and, if so, which attack is perpetrated.

- Target block parameter distribution for attack j, $\Gamma^1_j(\mathbf{c}) = \left(\gamma^1_{j,1},\ldots,\gamma^1_{j,K}\right)$. $\Gamma^1_j(\mathbf{c})$ is used as input to draw samples from a Dirichlet distribution to obtain the probabilities that attack j targets entry k, being these equivalent to the EPs. Note that Laplace smoothing (Manning et al., 2008) is applied to each $\gamma^1_{j,k}$ to address the potential issue of null components when sampling from the Dirichlet distribution.

Algorithm 1 Compute (random) optimal attacks through Monte Carlo

Input: $\mathbf{c}$, P_A, U_A, number of samples V

1: **for** $v \in \{1, \ldots, V\}$ **do**
2: sample u_A, $p_A \sim U_A$, P_A
3: $\zeta_v(\mathbf{c}) = \arg\max_{x \in \mathcal{A}}(\Psi_A^v(x))$
4: **Return:** $\{\zeta_v(\mathbf{c})\}_{v=1}^V$

Algorithm 2 Compute distribution of attacker behaviour $\big(\tau(\mathbf{c})\big)$

Input: $\{\zeta_v(\mathbf{c})\}_{v=1}^V$ (Output Algorithm 1)

1: **for** $j \in \{1, \ldots, J\}$ **do**
2: $\tau_j^1(\mathbf{c}) = \big|\{\zeta_v(\mathbf{c}) = a_{j,\cdot}^1\}_{v=1}^V\big|/V$
3: $\tau^i(\mathbf{c}) = \big|\{\zeta_v(\mathbf{c}) = a^{i \neq 1}\}_{v=1}^V\big|/V$
4: **Return:** $(\tau_1^1(\mathbf{c}), \ldots, \tau_J^1(\mathbf{c}), \tau^i(\mathbf{c}))$

Algorithm 3 Compute target parameter distribution for attack j $(\Gamma_j^1(\mathbf{c}))$

Input:$\{\zeta_v(\mathbf{c})\}_{v=1}^V$ (Output Algorithm 1), attack j

1: **for** $k \in \{1, \ldots, K\}$ **do**
2: $\gamma_{j,k}^1(\mathbf{c}) = \big|\{\zeta_v(\mathbf{c}) = a_{j,k}^1\}_{v=1}^V\big| + 1$ ▷ Laplace smoothing
3: **Return:** $\big(\gamma_{j,1}^1, \ldots, \gamma_{j,K}^1\big)$

2.2.4 AI-based security and recovery controls

Many systems have components incorporating ML into their security and safety controls that may be vulnerable to being exploited for malicious purposes. To evaluate these vulnerabilities, the PNP q parameters introduced in Section 2.2.2 are crucial, since they account for how effectively a given defense protects against a specific attack type in a particular environment. As with the arrival stochastic process and the EPs for untargeted attacks in Section 2.2.3, the parameters q can be inferred from data or, when data are limited, from expert judgment and updated as new evidence emerges (Hanea et al., 2021). However, in the specific case of dealing with AI components that serve as safety controls of products, like as content filters, it is relevant to consider the *security evaluation curves* (Gallego et al., 2024), which are part of the AML domain, as they are a major source of information for assessing these parameters. These curves depict the probability $(1 - q)$ of succeeding in protecting an ML algorithm from an attack of a certain type and intensity, given the defense implemented. Hence, given a type of attack, its intensity, and the chosen defense, the parameter $1 - q$ would be estimated with the corresponding curve, from which the PNP q would be deduced.

Example 2. Fig. 2.4 illustrates such curves for four AML defenses (none, adversarial training (AT), adversarial logit pairing (ALP), ARA) (Gallego et al., 2024) over a classifier used in a computer vision task, employed in an ADS against a fast gradient sign method (FGSM) attack (Szegedy et al., 2014) of increasing intensity. For a 0.06-intensity attack, the estimated expected accuracy in the task would be around 0.93 (with ARA, AT, and ALP defenses) and about 0.87 with no defense and $\hat{q}$ would be, respectively, 0.07 and 0.13. To complete the assessment, we could account for the uncertainty in the estimate $\hat{q}$ modeling it with a Beta distribution in which, for instance, the mode corresponds to $\hat{q}$. △

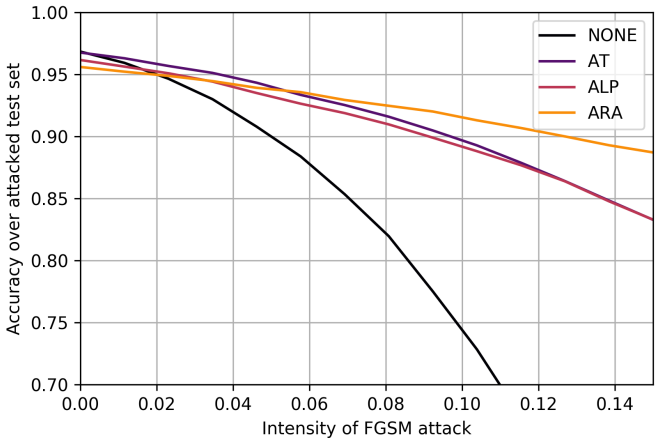


Figure 2.4: Security evaluation curve of a deep network under four defense mechanisms against FGSM attack. From Gallego et al. (2024).

2.3 The framework

We integrate the above solutions to provide a five-stage risk analysis pipeline for systems with AI-based components.

2.3.1 Attack transit simulation

Using the block-level structure in Section 2.2.2, we first introduce a scheme (Algorithm 4) to simulate the transit of an attack within a cyber organization. It outputs an indicator I_i for each block B_i with $I_i = 1$ (0) if the block has (not) been successfully attacked. $N^{'}$ designates a sufficiently big number, interpretable as the expected number of attack transits before detected or achieving its malicious objective, see Miehling et al. (2018) for pointers about time until vulnerability discovery. If necessary, it may be replaced by parameters $N^{'}_{jh}$ referring to transits between blocks j and h. $N^{'}$ or the $N^{'}_{jh}$ would be generated from properly selected distributions. Should the corresponding graph be acyclic, we would make $N^{'} = 1$, there being at most one attack transition through each block.

Let us comment key points in Algorithm 4. Step 2 simulates attack entry points: for a given facility with EPs P, we generate one sample from a multinomial with such probabilities. A Dirichlet distribution is used to generate P, given the uncertainty about it, with modeling approaches in Section 2.2.3. In principle, this requires up to $2^n - 1$ probabilities, with n the number of single entry points, a shortcoming from a storage point of view. A way to mitigate this is to assume that, with a certain probability (say, generated from a beta distribution) the attack is generic affecting all entry point blocks, whereas the remaining probability would be equally allocated to targeted attacks on the identified single entry points, recall Section 2.2.3 on targeted attacks. Steps 3 and 15 simulate whether an attack on a block is successful, based on PNP q, typically drawn from a Beta distribution to account for uncertainty. Specifically, Step 3 applies to attacks on the system's entry blocks, while Step 15 refers to downstream blocks as the attack propagates through the system. Section 2.2.4 discusses how to assess q, with particular emphasis on AI-based blocks.

Algorithm 4 Attack transit simulation in a general facility

Input: G, P, Q, Q^s

$I_j = 0$, $\forall$ block B_j
Given P, simulate the attacked entries and assign $I_j = 1$ to them.
Given Q, assign $I_j = 0$ to non-successfully attacked entries.
if all $I_j = 0$ **then**
  Stop
else
  for $h \in \{1, \ldots, k\}$ **do** ▷ Loop in level-depth
    for $u \in \{1, \ldots, N^{'}\}$ **do**
      for $j \in \{i_h + 1, \ldots, i_{h+1}\}$ **do**
        if $I_j = 0$ **then**
          Next j
        else
          for each successor h of block j **do**
            if $I_h = 0$ **then**
              if successor h successfully attacked from block j **then**
                $I_h = 1$
Return: I_j, $\forall$ block j

2.3.2 Simulation and aggregation of impacts

Algorithm 4 outputs a single sample $(I_1, \ldots, I_{i_k})$ consisting of the blocks affected by the attack. Consider now simulating the associated impacts (Section 2.2.1): some impacts will be *global* (one impact for the whole system), whereas others will be *separable* (one impact per attacked block). Separable impacts are aggregated with a rule b. Finally, multiple impacts are aggregated with a rule g.

Example 1 (cont). Recall Figs. 2.2a and 2.1. The impacts deemed relevant in this case are: *financial* (l_{FI}) encompassing all costs related to loss of sensitive information, including those deriving from the legal ecosystem associated with ADS liability (UNECE, 2022); *equipment damage* (l_{ED}) referring to harm inflicted on ADS components, including those associated with firmware (Halder et al., 2020) and software management (ISO, 2023b); and *downtime* (l_{DT}), the time the ADS is unavailable due to the attack. l_{FI} is global; l_{ED} and l_{DT} are separable. Typical aggregation rules for separable impacts l_{ED} and l_{DT} are: $l_{ED} = \sum_j l_{EDj}$, where $l_{ED_j}, j \in \{P, L, D\}$ is the equipment replacement costs for block j (0 if no replacement required); $l_{DT} = \max(l_{DT_P}, l_{DT_L}, l_{DT_D}) \times d$, where l_{DT_j} is the downtime of block j for $j \in \{P, L, D\}$ and d is the unit downtime cost. Then, aggregate them through a rule $l = g(l_{FI}, l_{ED}, l_{DT})$, e.g. using a multi-attribute weighted value function $g(l_{FI}, l_{ED}, l_{DT}) = \sum_i w_i \times l_i, i \in \{FI, ED, DT\}$, where w_i weighs the importance of the i-th impact, and even transform it through e.g. a risk-averse utility function (González-Ortega et al., 2018). △

At a schematic level, simulating from the impact distribution based on $(I_1, \ldots, I_{i_k})$ runs as in Algorithm 5, with the first L impacts are assumed to be local and the remaining $R - L$, global. The j-th impact, $j = 1, ..., L$ has its corresponding aggregation rule b_j. This procedure generates an impact sample, used to build the loss curves for RM, like those employed in the case study in Section 2.4.

Algorithm 5 Simulation from impact distribution

Input: $(I_1, \ldots, I_{i_k})$

1: $l_1, l_2, ..., l_R = 0$
2: **if** $\max(I_1, \ldots, I_{i_k}) = 0$ **then**
3: $l = 0$
4: **else**
5: **for** $r \in \{L+1, \ldots, R\}$ **do**
6: Generate l_r
7: **for** $h \in \{1, \ldots, i_k\}$ **do**
8: **if** $I_h = 1$ **then**
9: **for** $j \in \{1, \ldots, L\}$ **do**
10: Generate l_j^h
11: **for** $j \in \{1, \ldots, L\}$ **do**
12: Aggregate l_j^h's with rule b_j obtaining $l_j^{'}$
13: Aggregate the $l_j^{'}$'s and l_r's with rule g obtaining l
14: **Return:** l

2.3.3 A general scheme for attack simulation

Algorithm 6 incorporates the ARA-based method in Section 2.2.3 to simulate the impacts of attacks of a given type. There would be one stochastic process Λ for each attack type with its own parameters, with the corresponding generating routine defining its arrival process. It first obtains τ and Γ_j^1 to compute the attack probabilities on the system, if the attack is targeted; otherwise, it would use the EPs P estimated from data and/or expert judgement. The stochastic process generating the attacks in the risk analysis planning period would be invoked. Then, for each targeted attack it is predicted whether or not the attacker selects our system. If this is the case, Γ_j^1 is used to select which block will be targeted within it. If the attack is untargeted, we just use the EPs. Next, for each entering attack, its routine would be used to determine its impacts using Algorithms 4 and 5. Observe that commands 3-12 in Algorithm 6 would be akin to a simulated pentesting procedure (Sarraute et al., 2013). M represents the sample size required to estimate the quantities used below (expected costs and utilities) to the desired precision.

Algorithm 6 Attack simulation in a general facility

Input: G, Λ, distributions for Q,Q^s

Estimate attacker behaviour (τ) and target block (Γ_j^1) parameter distributions ▷ Replace by P if untargeted attack
for $j \in \{1, \ldots, M\}$ **do**
  Generate number N of attacks using Λ
  for $i \in \{1, \ldots, N\}$ **do**
    Choose target using τ
    if our system selected **then**
      Generate Γ_j^1, Q, Q^s
      Simulate attack transit using Γ_j^1, Q, Q^s ▷ Algorithm 4
      Simulate impact l_i from attack ▷ Algorithm 5
    else
      Assign $l_i = 0$
  Compute $l^j = \sum_{i=1}^{N} l_i$
Return: $\{l^j\}_{j=1}^{M}$

2.3.4 Risk assessment

Once with M impact samples, we can estimate *system losses*: the output $\{l^j\}_{j=1}^{M}$ is fit to a mixed-type distribution, with a point mass at 0 and a density for the positive impact observations. The density is typically modeled as a mixture of gamma distributions, since any distribution on the positive real line can be parsimoniously approximated by such mixtures, which allow as well for modeling features such as skewness, as outlined in Wiper et al. (2001). This mixed-typed distribution can be summarized through the probability of the zero part, the moments of the positive part, and quantities like the VaR, the CVaR or the loss exceedance curve (Silva, 2018). For representation, plotting methods as in Weld and Leemis (2019) are relevant. If considered sufficiently risky, we would proceed with RM (Section 2.3.5).

A number of additional relevant pieces of information can also be estimated, like the *block attack probabilities:* the I_j's serve to estimate the attack probabilities on various blocks counting the number of 1's for each block divided by the number M of simulations, which assess the attack susceptibility of the blocks for the corresponding attack type.

2.3.5 Risk management

We present an approach to RM covering the following issues: includes the portfolio costs; handles cyber insurance as part of the portfolio; takes into account the organization's risk attitude and portfolio constraints; acknowledges uncertainty in portfolio evaluation due to sampling; and is based on the proposed granular structure including AI components.

RM formulation

Cyber mitigation portfolios are characterized as vectors $\mathbf{c} = (c_1, ..., c_m)$ with $c_i \in \{0, 1, 2\}$ indicating whether the i-th control is (not) included in the portfolio when $c_i = 1$ ($c_i = 0$) or already implemented ($c_i = 2$). Some controls might correspond to AI-based safety components; one of them, to a cyber insurance product. An organization will typically have a few mitigations already implemented, say the first r, that is $c_1 = \ldots = c_r = 2$. Besides, for compliance reasons

e.g. with the AIA, some of the mitigations would be enforced, say from $r+1$ to s, so that the initial configuration is

$$\mathbf{c}^0 = \big(\underbrace{2,\dots,2}_{r},\underbrace{1,\dots,1}_{s-r},\underbrace{0,\dots,0}_{m-s}\big).$$

The aim is to decide which of the controls $(c_{s+1},...,c_m)$ should be implemented additionally. For this, given the proposed configuration $\mathbf{c}$, suppose we have available the EPs $P(\mathbf{c})$ and PNPs $Q(\mathbf{c})$. A simulation from Algorithm 6 would provide a sample $\{l(\mathbf{c})^j\}_{j=1}^M$ of the losses. Thus, we would solve

$$\max_{\mathbf{c}} \; E\big(u(\mathbf{c})\big)$$

$$\text{s.t. } c_i \in \{0,1,2\};\; c_i = 2 \;(i \leqslant r),\; c_i = 1 \;(r+1 \leqslant i \leqslant s),$$

$$h(\mathbf{c}) \leqslant 0$$

where u is the utility function modeling preferences and risk attitudes, and $h(\mathbf{c}) \leqslant 0$ designates relevant constraints.

Concerning the objective function, we first integrate the portfolio $cost(\mathbf{c})$ in the loss, which adopts the form $l(\mathbf{c}) + cost(\mathbf{c})$, and, next, adopt a constant absolute risk averse (CARA) utility function (González-Ortega et al., 2018) whose form is $u(\mathbf{c}) = 1 - \exp(\rho(l(\mathbf{c}) + cost(\mathbf{c})))$, with ρ the risk aversion coefficient (note that we aim at minimizing costs, therefore maximizing -costs).

In turn, two typical constraints would be: a) *Compliance* with laws like AIA, standards like (ISO, 2023a), and frameworks like AIRMF enforcing certain mitigations, making their $c_i = 1$ and entailing a reduction in the security budget; b) *Budget*, usually there being a constraint indicating the maximum budget C available for cyber mitigation, $\sum_{i=1}^m \text{sign}(c_i) \times cost(i) \leqslant C$, with $cost(i)$ the i-th mitigation cost. Splits between maintenance and implementation costs could be introduced. Then, taking into account the definition of the c_i variables, if $icost(i)$ represents the i-th implementation cost and ic represents the maximum implementation budget, the constraint would be $\sum_{i=r+1}^m \text{sign}(c_i) \times icost(i) \leqslant ic$. Similarly, if $mcost(i)$ represents the i-th maintenance cost and mc the maximum maintenance budget, the constraint would be $\sum_{i=1}^m \text{sign}(c_i) \times mcost(i) \leqslant mc$.

RM implementation

A first strategy, adopted in Section 2.4, would be to search the space of portfolios, find out their corresponding probabilities $P(\mathbf{c})$ and $Q(\mathbf{c})$, and obtain a sample from the loss $l(\mathbf{c})$ to estimate the expected utility of the portfolios, and optimize it, with the aid of a discrete stochastic optimization method, see e.g. Powell (2022). This may become too cumbersome computationally when the set of feasible portfolios is large and/or the cyber system architecture is complex.

Alternatively, based on the typical form of the loss curves, see Fig. 2.5 in Section 2.4, we model the loss as a mixture

$$\mathcal{L}(\mathbf{c}) \sim s(\mathbf{c}) \cdot I_0 + \big(1 - s(\mathbf{c})\big) \cdot Gamma\big(a, t(\mathbf{c})\big) \tag{2.3.1}$$

where $s(\mathbf{c})$ is the probability of no loss if $\mathbf{c}$ is implemented and $t(\mathbf{c})$ adopts a specific parametric form to adapt to the shape of positive losses, using the parameterization $Gamma(a,p)$ with mean $a \times p$. A similar procedure would be followed if the gamma distribution in (2.3.1) has to be replaced by a mixture of gamma distributions.

For the i-th mitigation, assume we have its implementation $icost(i)$ and maintenance $mcost(i)$ costs and parameters (α_i, β_i) describing its effectiveness as characterized below. The corresponding costs will be $cost(\mathbf{c}) = \sum_{i=r+1}^{m} \text{sign}(c_i) \times icost(i) + \sum_{i=1}^{m} \text{sign}(c_i) \times mcost(i)$. In turn, the parametric forms adopted for $s(\mathbf{c})$ and $t(\mathbf{c})$ are

$$s(\mathbf{c}) = 1 - s_0 \exp\Big(- \Big(\sum_{i=1}^{m} \alpha_i \times \text{sign}(c_i) \Big) \Big), \;\; t(\mathbf{c}) = t_0 - \sum_{i=1}^{m} \beta_i \times \text{sign}(c_i), \tag{2.3.2}$$

suggesting diminishing returns in cybersecurity investments, with s_0 and t_0 the values of $s(\mathbf{c})$ and $t(\mathbf{c})$ when no countermeasure is implemented.

Let us briefly discuss how to estimate the loss curves (2.3.1). It is sufficient to consider one term in the functions in (2.3.2). Consider first assessing $s(\mathbf{c})$ with $\mathbf{c} = (1, 0, \ldots, 0)$, which requires estimating s_0 and α_1. If we do not implement the first countermeasure, we have $s(\mathbf{c}) = 1 - s_0$. Hence, s_0 is interpreted as the probability that the system receives a successful attack when no additional countermeasures are implemented and can be elicited with standard expert judgment techniques (Hanea et al., 2021). Similarly, $s_1 = s_0 \exp(-\alpha_1)$ is the probability that the system receives a successful attack when the countermeasure is implemented. Based on expert judgment we may assess s_1 and obtain $\alpha_1 = \log s_0 - \log s_1$. If required, we may assume beta distributions around the values s_0 and s_1 assessed and, eventually, learn about them with beta-binomial models, e.g. French and Insua (2000). The ideas extend immediately to the case when there are several countermeasures, and similarly when there is interaction between the countermeasures.

Moving on to $t(\mathbf{c})$, we follow a similar procedure based on the properties of the gamma distribution. First, when no countermeasure is introduced the impact of successful attacks is assessed with a $Gamma(a, t_0)$ distribution and we may use expert judgment based on two quantiles or two moments, plus consistency checks, to assess a and t_0. When the countermeasure is implemented, the distribution is a $Gamma(a, t_0 - \beta_1)$. Since a is assessed, we use one quantile or one moment to assess through expert judgment $t_0 - \beta_1$ and, consequently, β_1. Again, in presence of data, we can use learning processes based on gamma distributions on a, t_0 and $t_0 - \beta_1$ to improve the assessments.

Algorithm 7 implements the previous methods to estimate, for a given portfolio $\mathbf{c}$, its expected utility, updating parameters given the implemented controls, like its implementation ($ticost$) and maintenance ($tmcost$) costs.

Algorithm 7 Compute expected utility

Input: $\mathbf{c}$, s_0, t_0, α, β, a, M, ρ, $mcost$, $icost$

Compute $tmcost$ and $ticost$; $ancost = tmcost + ticost$
for $i \in \{1, \ldots, m\}$ **do**
  if $(c(i) = 1)$ OR $(c(i) = 2)$ **then**
    $s_0 = s_0 \times \exp\big(-\alpha_i\big)$; $t_0 = t_0 - \beta_i$
$s = 1 - s_0$; $t = t_0$; $util = 0$
for $i = 1, \ldots, M$ **do**
  $cost(i) = 0$
  Generate $u \sim U(0, 1)$
  if $u > s$ **then**
    $cost(i) \sim Gamma(a, t)$
  $cost(i) = cost(i) + ancost$
  $util = util + (1 - \exp\big(\rho \times cost(i)\big))$
Return: $util/M$

Finally, Algorithm 8 integrates the previous pieces. Given a portfolio **c**, its feasibility is first assessed. If feasible but not optimal, the portfolio is updated, where Algorithm * designates a generic routine proposing a portfolio update with optimization purposes, see Powell (2022) for pointers including simulated annealing, or tabu search.

Algorithm 8 Global scheme

Input: c

Compute portfolio costs
if "Infeasible Portfolio" **then**
  Stop
while not convergent **do**
  Compute portfolio costs
  if "Infeasible Portfolio" **then**
    Stop
  else
    Estimate expected utility of portfolio ▷ Algorithm 7
    Update **c** ▷ Algorithm *
Return: Optimal **c**

2.4 Case study

This section illustrates and validates conceptually the framework with a simplified case study. The case is complex enough to reflect its required modeling steps; its advantages, because of providing improved cybersecurity; and its shortcomings, mainly the additional modeling and computational effort entailed. It also serves as a procedural template for other case studies. We consider an ADS fleet owner who uses them for rental purposes and wishes to improve their cybersecurity; two additional companies compete in the same market. Caballero et al. (2023) describe the key role of AI components in ADS. Parameters for this one-year horizon risk analysis problem are available in Appendix A.

2.4.1 Model setup

Let us describe the relevant modeling elements and solving steps. We first characterize the ADS architecture through blocks in levels and specify the relevant risk analysis impacts. Additionally, we identify potential (non-targeted and targeted) threats, as well as the prospective perpetrators, and conceivable controls not yet implemented. Finally, we undertake risk assessment and management stages.

ADS architecture. We use the architecture in Fig. 2.2a, with two level-1 blocks (P, L) susceptible to external attacks, and one level-2 block (D).

Impacts. Following Example 1 (Section 2.3.2), the owner considers three relevant impacts: *financial* and *equipment damage*, both assessed in thousands of euros; and, *downtime*, measured in hours.

Non-targeted threats. There are numerous potential non-targeted threats against ADS (Taslimasa et al., 2023). To simplify, we only consider: 1) *Denial of Service (DoS):* traffic

infrastructure is infected to gain access to vehicles, e.g. to prevent users from entering the vehicle by intervening in the door-locking system demanding a ransom to regain access; and 2) *Software/hardware supply chain threats (SCTs):* rogue updates may make an ADS non-operational, e.g., malicious changes may alter the parameterization of ML algorithms, making them vulnerable to attacks, as highlighted by the recent XZ vulnerability (Vijayan, 2024) showcasing how integration and development of AI products is exposed to critical backdoor risks (Clifford et al., 2022).

Targeted threats. The identified potential attackers are: 1) *Cyberterrorist group (Cy)*: they could modify traffic signals or interfere with Vehicle-to-Infrastructure (V2I) communications to achieve political notoriety; 2) *Criminal gang (Cr)*: they aim at economic gains through stolen sensitive information. These groups may target the ADS through different attack types. We consider: 1) *Adversarial attacks against ML algorithms (AML_at)*: a perpetrator decides to alter the data plane of vehicular communications to outsmart the ADS ML algorithms, see e.g. Wei et al. (2022); 2) *Wireless jamming on the control plane (wir_jam)*: a malicious actor could send signals to interfere with the functioning of devices such as the GPS system, potentially causing accidents. This might also affect the system ML algorithms since their inputs may be altered.

Table 2.2[i] summarizes attacks and actors capable of carrying them out and we assume that attackers can execute only one type of attack at any given moment. Table 2.3 shows the impacts of different threats.

Table 2.2: Attacks available to malicious actors.

Attacks/Attacker	Cy	Cr
AML_at	X	X
wir_jam	X	

Table 2.3: Impacts deemed relevant for targeted and untargeted threats.

Attacks/Impacts	Financial	Equipment damage	Downtime
AML_at (Cy)	X	X	X
AML_at (Cr)	X	X	
Wireless jam.	X	X	X
DoS	X		X
SCTs	X	X	X

Potential defenses. The contemplated defenses not yet implemented in the ADS are: a firewall and internet gateway *(FwGw)*; a robust AML module *(AML)*; and a patch management, IDS and vulnerability scanner (PmVs) module *(PmVs)*. Table 2.4 describes the defense effectiveness against attacks, i.e. making them less likely and/or impactful, if successful.

[i] In this table, and in the subsequent ones, a mark X indicates that the element in that given row applies to the element in the corresponding column.

Table 2.4: Defense effectiveness.

Attacks/Defenses	FwGw	AML	PmVs
Adversarial attack		X	
Wireless jamming	X		X
DoS	X		X
SCTs	X	X	X

Two cyber insurance options are considered: a basic one (A) covering equipment damage occurring to the vehicle and an advanced one (B) additionally covering downtime impacts as displayed in Table 2.5.

Table 2.5: Impacts mitigated by each cyber insurance product.

	Finan.	Equip. damage	Downt.
A		X	
B		X	X

Constraints. The maximum protection budget is 3400 euros. Legislation requires cyber insurance product A to be included at the minimum.

2.4.2 Risk analysis

This section focuses on analyzing the risk associated with the system given its *initial configuration* (compulsory minimum insurance product A and absence of new protections). We apply the proposed framework utilizing the parametric setup in Appendix A to identify the portfolio most efficient in mitigating system risks. As we shall see, a very important risk reduction for the proposed ADS system is obtained showcasing the framework's potential.

Risk assessment. Fig. 2.5 (blue) illustrates the loss curve in log€ for the initial configuration, using a Monte-Carlo (MC) sample size of 10,000. The no-loss probability is 0; positive losses are fit with a Gamma distribution with $a = 5871.48225$ (shape)[ii] and $p = 0.00235$ (scale). The 95% VaR is 1.32 M€. The 95% CVaR is 1.498 M€. The risk is deemed too high, thus demanding a RM stage.

Risk management. Using the parameters in Appendix A, only 12 portfolios out of 16 are feasible, being reasonable to evaluate all of them to find the best one since assessing the 12 portfolios took about 10 minutes on a standard laptop. Their expected utilities are assessed by MC as in Algorithm 7 with MC sample size of 10000. Table 2.6 presents the three best portfolios and the initial portfolio configuration, their expected loss, implementation cost, and expected utility. The optimal portfolio is (A, AML, PmVs), that is, consisting of cyber insurance A, an AML module and a PmVs module; the second best is (B, FwGw, AML), whereas (A, FwGw,

[ii]Decimals are kept to show that the mean $a \times p$ matches the curve in Fig. 2.5.

AML) is the third one. Importantly, the optimal portfolio, is not the most expensive. Fig. 2.5 (red) displays the loss curve for the optimal portfolio. The probability of zero loss rises to 0.174 and the positive part is fit with one gamma component with parameters $a = 192.76327$ and $p = 0.05189$. The 95% Var and 95% CVaR are respectively 59520 and 72756 €, a significant risk reduction, confirming the potential of the framework to reduce system risks.

Table 2.6: Three best portfolios and initial portfolio configuration.

Portfolio	Exp. loss	Cost	Exp. Utility
$A, AML, PmVs$	22834.59	2300	-0.002
$B, FwGw, AML$	43918.99	2950	-0.004
$A, FwGw, AML$	48185.75	1800	-0.005
A	1017207.08	600	-0.107

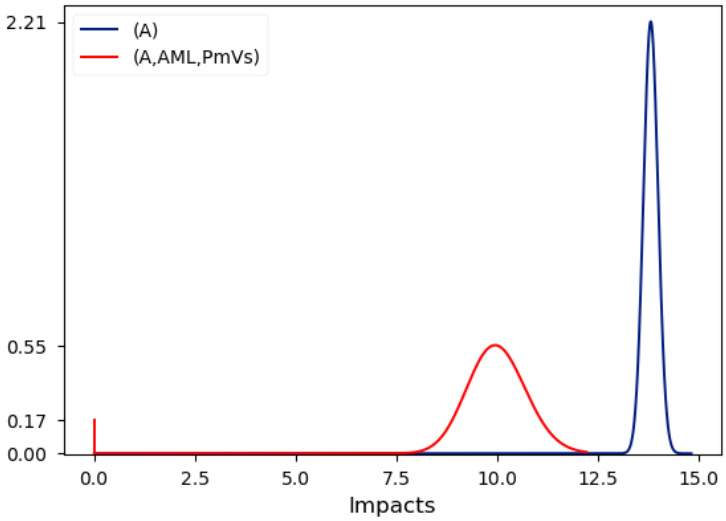


Figure 2.5: Loss curves per ADS on a log scale for initial (blue) and optimal (red) configurations.

The framework provides further contextual security information. Using Section 2.2.3, Table 2.7 provides the probabilities of various attacks from the cyberterrorist group with the initial and optimal portfolios. Observe the deterrent effect of the optimal one, with attack probability greatly transferred to competitors and, in particular, becoming almost fully protected from AML attacks.

Table 2.7: Probability of perpetrating an attack on our company or attacking another one by the cyberterrorists, with initial and optimal configurations.

	AML_at	wir_jam	Other
Initial	0.63	0.16	0.21
Optimal	0.01	0.42	0.57

Tables 2.8 a-b provide further details on the attack entry points when facing the cybercriminal group. In this case, there is virtually little difference between the initial and optimal portfolios as all entry points seem equally susceptible to both types of attacks.

Table 2.8: Γ vector of each targeted attack for initial and optimal portfolio configuration.

	P	L	P, L
Initial	0.33	0.34	0.33
Optimal	0.35	0.30	0.35

(a) AML_at (Cy)

	P	L	P, L
Initial	0.32	0.34	0.34
Optimal	0.34	0.33	0.33

(b) Wir_jam (Cy)

Sensitivity analysis. We performed extensive sensitivity analysis for key parameters. First, we explored the budget impact, with budget negotiation purposes. Table 2.9 displays optimal portfolios as budget ranges from 600 to 4500. The expected loss reduces considerably at each budget step, except for the last one, entailing switching to insurance B.

Table 2.9: Optimal portfolio for different values of maximum budget.

Portfolio	Exp. loss	Exp. Utility	Budget
A	1016077	-0.107	[600,700]
A, AML	814828	-0.085	[800,1700]
A, FwGw, AML	48185	-0.005	[1800,2200]
A, AML, PmVs	22834	-0.002	[2300,4200]
A, FwGw, AML, PmVs	13337	-0.0017	[3500,4200]
B, FwGw, AML, PmVs	11743	-0.0016	[4300,4500]

As B appears only with the biggest budgets, let us explore the impact of price reductions of up to 20%, preserving its coverage. Table 2.10 shows the optimal portfolio for various reductions. Notice that with a price reduction from 20% until 4% there is a switch from insurance A to insurance B. These calculations support better pricing of the product e.g. to increase its sales.

Table 2.10: Optimal portfolio for several price reductions in insurance B.

Price reduction	Portfolio	Expected loss	Cost	Expected utility
20%	$B, AML, PmVs$	20338.85	3170	-0.0024
10%	$B, AML, PmVs$	20441.08	3310	-0.0024
4%	$B, AML, PmVs$	20508.96	3394	-0.0024
3%	$A, AML, PmVs$	22491.14	2300	-0.0025

Besides, we considered sensitivity to the risk aversion coefficient ρ, varying it within a grid from 10^{-7} to 10^{-3}; the same optimal portfolio emerged with only a switch between the second and third portfolios, as ρ gets bigger than 10^{-4}. We also varied the parameter (λ) of the arrival process of untargeted attacks: for DoS attacks, we considered $\lambda \in \{8, 16, \mathbf{32}, 48, 56\}$, while for SCT, $\lambda \in \{1, 2, \mathbf{2.75}, 4, 5\}$, with original values in boldface. We explored as well various coverages (65%, **70**%, 75%, 80%, 85%, 90%) for insurance B. In all cases, the same top three portfolios emerged, underscoring the robustness of the results.

2.5 Discussion

The RM of AI-based systems poses challenges related to their components, the impacts they may incur, the targeted attacks to which they may be subjected, and the security and recovery controls that protect them. To address these challenges, we propose solutions that are integrated into a comprehensive five-stage RM framework. The implemented approach leverages a quantitative risk methodology that uses simulation and optimization to select the most effective protective measures for the systems, thereby advancing beyond prevailing qualitative CSRM techniques. In this way, the framework contributes to objective **O1** by advancing CSRM methodology from a broad perspective and providing a holistic approach to managing risks in systems with AI components.

The proposed solutions include employing an ARA approach to model targeted attacks and identify the system components through which an attack is most likely to gain access, using security curves to assess how vulnerable a given component is to adversarial attacks, and adopting a granular cyber-architecture modeling strategy based on directed graphs that illustrate dependencies between components. The flexibility of the framework makes it suitable for protecting a wide range of AI systems and facilitating compliance with AI-related regulations, such as the AIA and NIST AIRMF, while also addressing their limitations.

Additionally, the framework has been conceptually validated through an ADS case study, which showed a considerable reduction in risk with the portfolio selected by the framework. Beyond identifying the cybersecurity portfolio that most effectively reduces risk, the case study also demonstrates the framework's ability to provide contextual security information for each portfolio, such as the probability that an attacker will target the system under a given protection. This capability further supports budget negotiations by showing the level of protection achieved under different budget constraints, thereby illustrating a practical use case in industry. At the same time, the case study reveals that the framework is more computationally intensive and requires greater modeling effort than simpler approaches. Nevertheless, this additional effort is justified when considering the potential consequences of a successful cyberattack.

Chapter 3

Computational adversarial risk analysis for general security games

3.1 Introduction

The management of cybersecurity issues in systems with intelligent components and services, both at the strategic level (as in Chapter 2) and at the operational level (as in Chapter 4) may be modelled as security games (Brown et al., 2006; Zhuang and Bier, 2007; Hausken, 2024). Analyzing these games helps in understanding the characteristics of various attacks, including those targeting ML algorithms, and represents a crucial first step toward the development of more effective defenses against potential threats.

ARA serves as a solution framework for security games, providing an alternative to game theory that partially mitigates common-knowledge assumptions (Banks et al., 2022). Its potential to address security problems is demonstrated through several applications, which, apart from cybersecurity (Insua et al., 2021a), include anti-IED warfare (Wang and Banks, 2011), international piracy (Sevillano et al., 2012), counter-terrorism online surveillance (Gil and Parra-Arnau, 2019), and combat modeling enhancement (Roponen and Salo, 2015).

This chapter address Objective **O2**, contributing to the *targeted threats* and *technical security controls* components of the CSRM representation in Figure 1.1. Traditionally, ARA has been applied to scenarios with discrete decision spaces and low cardinality. However, in the new paradigms emerging within CSRM that incorporate AI, there is a growing need to advance on computational methods for solving security games using ARA. This is because scenarios such as those found in AML require methods capable of identifying optimal decisions in discrete decision spaces with high cardinality, or even in continuous spaces. This chapter introduces an efficient computational ARA approach for addressing general security games, possibly with continuous decision domains.

From a methodological point of view, in his celebrated paper, Shachter (1986) proposed as a major open problem devising methods to solve IDs with multiple decision-makers. Koller and Milch (2003) provided an algorithm to compute Nash equilibria in problems with discrete decision domains under common knowledge assumptions. Virtanen et al. (2006) extended the ideas to sequential parallel games computing Nash equilibria in air combat models. González-Ortega et al. (2019) provided an ARA solution in discrete problems. In turn, (Ekin et al., 2023) explored augmented probability simulation (APS, Bielza et al., 1999) as a solution method to handle simple sequential single-stage defend-attack games from an ARA perspective. This chapter shows how APS may be used to solve general security games described through BAIDs from an ARA perspective, therefore providing the first general solution to Shachter's query.

Our focus will be on the scenario of a Defender (D, she) facing a single Attacker (A, he). Additionally, we constrain the Defender to act as a level-2 thinker in the sense of Stahl and Wilson (1995): the Attacker is assumed to maximize expected utility while modeling the Defender as a non-strategic player. However, the Defender's uncertainty regarding the Attacker's beliefs and preferences propagates to his optimal decision, which becomes random and results in a probabilistic prediction of the attack.

We begin by describing how to deal with single-stage simultaneous defend-attack games and their multi-stage versions using APS (Section 3.2). Such solutions will serve us to motivate an efficient computational method to handle general security games detailed in Section 3.3. Section 3.4 presents key numerical strategies to support the implementation of our methods. A disinformation war example then illustrates the developments in Section 3.5. We end up with a brief discussion in Section 3.6. Appendix B provides the supplementary materials for the case study.

3.2 Solving simultaneous defend-attack games

This section presents APS and sketches how APS-based methods find optimal defenses in the important class of multi-stage simultaneous defend-attack games under incomplete information, allowing us to introduce key notation. It draws on Ekin et al. (2023) who handled two-stage sequential defend-attack games.

3.2.1 Augmented probability simulation

In decision analysis scenarios, the objective is typically to identify an action or a set of actions that yield the highest expected utility for the decision maker. In general, expected utility cannot be determined analytically and must be approximated. Most solution methods involve first approximating the expected utility for each possible action and then optimizing the resulting estimates. Instead, APS (Bielza et al., 1999) achieves expected utility maximization by transforming the estimation and optimization tasks into a simulation over the augmented distribution defined on the joint space of decisions and outcomes.

Suppose we seek to maximize the expected utility

$$\psi(d) = \int u(d, \theta) \cdot p(\theta|d) d\theta,$$

where θ denotes a random outcome distributed according to $p(\theta|d)$, and $u(d, \theta)$ represents the utility associated with choosing decision d and observing outcome θ. APS stems from the realization that if the utility is positive and integrable, we can define the augmented distribution

$$\pi(d, \theta) \propto u(d, \theta) \cdot p(\theta|d)$$

on the augmented space of decisions and outcomes. Then, under mild conditions, the mode of the marginal augmented distribution with respect to d corresponds to the solution that maximizes the expected utility. This leads to a strategy for determining the optimal decision by simulating from the augmented probability distribution, where Markov Chain Monte Carlo (MCMC) methods (French and Insua, 2000) are crucial, and subsequently identifying the marginal mode through mode estimation techniques (Chacon, 2020).

3.2.2 Single-stage simultaneous defend-attack games

Assume D has to choose her defense d from $\mathcal{D}$, her set of feasible alternatives. In parallel, A chooses his attack a from his feasible set $\mathcal{A}$. The result of the interaction between both agents depends on a random variable $\theta \in \Theta$. The agents have their own assessment of the probability distribution of the random outcome, respectively modeled through $p_D(\theta|d,a)$ and $p_A(\theta|d,a)$. D's utility function $u_D(d,\theta)$ depends on her chosen defense and the outcome; similarly, A's utility function has the form $u_A(a,\theta)$. Using the notation in Section 1.4, we depict such a simultaneous defend-attack game as a BAID in Figure 3.1a, as well as the Defender's problem (Figure 3.1b) and the Attacker's problem from D's perspective (Figure 3.1c).

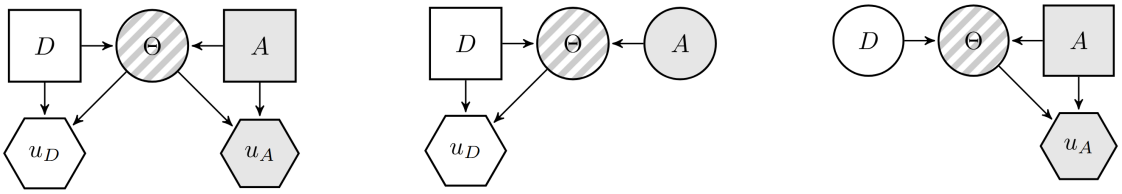


Figure 3.1: (a) Simultaneous defend-attack game BAID, (b) D's problem, (c) D's analysis of A's problem.

As discussed, the complete information assumption, that is, that agents know both their own and their opponents' payoffs, preferences, beliefs, and feasible actions, will not hold in many security scenarios which we shall handle through a decision-analytic approach. Let D's expected utility be

$$\psi_D(d) = \int \psi_D(d,a) \cdot p_D(a)\,\mathrm{d}a, \tag{3.2.1}$$

where $\psi_D(d,a) = \int u_D(d,\theta)\, p_D(\theta|d,a)\,\mathrm{d}\theta$.[i] Her optimal decision will be $d^*_{\mathrm{ARA}} = \mathrm{argmax}_{d\in\mathcal{D}}\ \psi_D(d)$.

Finding it requires estimating $p_D(a)$, D's assessment of A choosing attack a. For this, consider A's problem who aimed to find

$$a^* = \underset{a\in\mathcal{A}}{\mathrm{argmax}} \int \psi_A(d,a) \cdot p_A(d)\,\mathrm{d}d,$$

with $\psi_A(d,a) = \int u_A(a,\theta) p_A(\theta|d,a)\,\mathrm{d}\theta$. However, as Keeney (2007) argues, D has no access to A to assess u_A and p_A, and, therefore, lacks complete knowledge about A's utilities and probabilities. Adopting a Bayesian approach, let us model such uncertainty through random utilities $U_A(a,\theta)$ and random probabilities $P_A(\theta|d,a)$ and $P_A(d)$, defined over a common probability space $(\Omega,\mathcal{F},\mathcal{P})$ with atomic elements $\omega \in \Omega$ (Chung, 2001). Then, the random optimal attack is

$$A^* = \underset{x\in\mathcal{A}}{\mathrm{argmax}} \int \Psi_A(d,x) \cdot P_A(d)\,\mathrm{d}d,$$

with $\Psi_A(d,x) = \int U_A(x,\theta) P_A(\theta|d,x)\,\mathrm{d}\theta$, and we make

$$p_D(A \leqslant a) = Pr\left[A^* \leqslant a\right] = \mathcal{P}\{\omega : A^{*\omega} \leqslant a\}.$$

This defines $p_D(a)$, the missing ingredient when searching for d^*_{ARA} in 3.2.1. Implementing this requires the specification of $U_A(a,\theta)$, $P_A(\theta|d,a)$, and $P_A(d)$. Of these, eliciting $P_A(d)$ has a

[i] We assume here that A, D and Θ are all continuous. Discrete cases are handled in a similar manner.

recursive element as we need to think about how the Attacker reasons about the Defender which, in turn, begs for thinking about how the Defender thinks about how the Attacker thinks about the Defender's, and so on. This leads to a recursion similar to the level-k scheme described in Ríos and Insua (2012). As mentioned, we only consider level-2 defenders, although the scheme described extends to higher levels in the thinking hierarchy.

As seen, ARA models entail integration and optimization procedures that may be computationally challenging. This chapter proposes efficient APS-based methods to handle them. First, for the Defender problem, assuming with no loss of generality that u_D is positive and integrable, we introduce an augmented distribution (AD) over (d, a, θ) for D's problem defined through

$$\pi_D(d, a, \theta) \propto u_D(d, \theta) \cdot p_D(\theta|d, a) \cdot p_D(a).$$

Observe that its marginal $\pi_D(d)$ in d is

$$\iint u_D(d, \theta) \cdot p_D(\theta|d, a) \cdot p_D(a) \, \mathrm{d}\theta \, \mathrm{d}a$$

which is proportional to $\psi_D(d)$. Consequently, $d^*_{\mathrm{ARA}} = \mathrm{mode}\,(\pi_D(d))$. In such a way, one just needs to sample $(d, a, \theta) \sim \pi_D(d, a, \theta)$ and estimate its mode in d to approximate d^*_{ARA}. Sampling may be performed using MCMC methods. In particular, Algorithm 10 (`DAPS` function) presents a Metropolis-Hastings (MH) (French and Insua, 2000) version. Besides the Defender's utility u_D and probabilities p_D, `DAPS` takes as input the number N of MCMC samples and the proposal g_D within the function `MH`.

Algorithm 9 MH step (function `MH`)

Input: g, u, x, x', y, y' θ, θ'

1: Compute acceptance probability $\alpha = \min\left\{1, \frac{u(x,\theta)\cdot g(x'|x)}{u(x',\theta')\cdot g(x|x')}\right\}$

2: Set $z = \begin{cases} (x, y, \theta) & \text{with probability } \alpha \\ (x', y', \theta') & \text{with probability } 1 - \alpha \end{cases}$

3: **Return:** z

Algorithm 10 MH Defender APS to approximate ARA solution (funtion `DAPS`)

Input: N, u_D, p_D, g_D

1: Initialize $d^{(0)}$

2: Draw $a^{(0)} \sim p_D(a)$

3: Draw $\theta^{(0)} \sim p_D(\theta|d^{(0)}, a^{(0)})$

4: **for** $i = 1$ to N **do**

5: Propose new defense $\tilde{d} \sim g_D(\tilde{d}|d^{(i-1)})$

6: Draw $\tilde{a} \sim p_D(a)$

7: Draw $\tilde{\theta} \sim p_D(\theta|\tilde{d}, \tilde{a})$

8: $(d^{(i)}, a^{(i)}, \theta^{(i)}) = \mathtt{MH}(g_D, u_D, \tilde{d}, d^{(i-1)}, \tilde{a}, a^{(i-1)}, \tilde{\theta}, \theta^{(i-1)})$

9: Discard burn-in samples and construct $\widehat{d}^*_{\mathrm{ARA}}$, a consistent estimator of the mode of the marginal augmented distribution, using the remaining d samples

10: **Return:** $\widehat{d}^*_{\mathrm{ARA}}$

`DAPS` requires the ability to sample from $p_D(a)$. To produce such samples, assuming that U^{ω}_A is a.s. positive and integrable, consider the Attacker random AD (RAD) model, defined for

each $\omega \in \Omega$ through,

$$\Pi_A^\omega(d, \theta, a) \propto U_A^\omega(a, \theta) \cdot P_A^\omega(\theta|d, a) \cdot P_A^\omega(d).$$

Then, as before, a.s. the mode of the marginal of $\Pi^\omega(d, \theta, a)$ in a coincides with $A^{*\omega}$. Consequently, by sampling $u_A(a, \theta) \sim U_A(a, \theta)$, $p_A(\theta|d, a) \sim P_A(\theta|d, a)$, and $p_A(d) \sim P_A(d)$, one can build

$$\pi_A(d, \theta, a) \propto u_A(a, \theta) \cdot p_A(\theta|d, a) \cdot p_A(d)$$

which conforms a sample from $\Pi_A(d, \theta, a)$ (Algorithm 11). Therefore, mode$(\pi_A(a))$ is a sample from A^*, thus providing a mechanism (`AAPS`) to sample from the target distribution. A repeated application of `AAPS` (Algorithm 11) would provide a sample of the required size from A^*. Input to `AAPS` includes the number M of samples, the proposal g_A for the algorithm, and the Attacker's random utility U_A and probabilities P_A.

Algorithm 11 MH Attacker APS to approximate sample from $p_D(a)$ (function `AAPS`)

Input: M, U_A, P_A, g_A

Initialize $a^{(0)}$
Draw $u_A(a, \theta) \sim U_A(a, \theta)$
Draw $p_A(\theta|d, a) \sim P_A(\theta|d, a)$
Draw $p_A(d) \sim P_A(d)$
Draw $d^{(0)} \sim p_A(d)$
Draw $\theta^{(0)} \sim p_A(\theta|d^{(0)}, a^{(0)})$
for $i = 1$ to M **do**
  Propose new attack $\tilde{a} \sim g_A(\tilde{a}|a^{(i-1)})$
  Draw $\tilde{d} \sim p_A(d)$
  Draw $\tilde{\theta} \sim p_A(\theta|\tilde{d}, \tilde{a})$
  $(a^{(i)}, d^{(i)}, \theta^{(i)}) = \texttt{MH}(g_D, u_D, \tilde{a}, a^{(i-1)}, \tilde{d}, d^{(i-1)}, \tilde{\theta}, \theta^{(i-1)})$
Discard burn-in samples and construct a^*, a consistent estimator of the mode of the marginal augmented distribution, using the remaining a samples
Return: a^*

Under mild conditions, the convergence of Algorithm 10 to d^*_{ARA} (supported by that of Algorithm 11) follows when using consistent mode estimators.

Proposition 3.1. *Suppose that*

1. *u_D and, a.s., the utilities in the support of U_A are positive and, respectively, continuous in (d, θ) and (a, θ).*

2. *The Defender's and Attacker's decision sets are compact.*

3. *$p_D(\theta|d, a)$ and, a.s., the distributions in the support of $P_A(\theta|d, a)$ are continuous in a and d, respectively, and positive in (d, a).*

4. *The proposal generating distributions g_D and g_A have, respectively, support $\mathcal{D}$ and $\mathcal{A}$.*

Then, Algorithm 10 defines a Markov chain with stationary distribution $\pi_D(d, \theta, a)$. Moreover, a consistent mode estimator based on the marginal samples of d from this Markov chain a.s. approximates d^*_{ARA}.

Proof. For each d and $\omega \in \Omega$, given the assumptions about $U_A(a,\theta)$, $P_A(\theta|d,a)$ and $\mathcal{A}$, we have: 1)

$$\Psi_A^\omega(d,a) = \int U_A^\omega(a,\theta) \cdot P_A^\omega(\theta|d,a)$$

is (a.s) continuous in a; therefore, 2) there exists a.s.

$$A^* = \underset{x\in\mathcal{A}}{\operatorname{argmax}} \int \Psi_A^\omega(d,a) \cdot P_A^w(d)\,\mathrm{d}d,$$

as we are maximising an a.s. continuous function in a compact domain; 3) the distribution

$$\Pi_A^\omega(a,\theta,d) \;\propto\; U_A^\omega(a,\theta) \cdot P_A^\omega(\theta|d,a) \cdot P_A^\omega(d)$$

is well-defined being $U_A^w(a,\theta)$ a.s. positive. By construction, through sampling $u_A(a,\theta) \sim U_A(a,\theta)$, $p_A(\theta|d,a) \sim P_A(\theta|d,a)$ and $p_A(d) \sim P_A(d)$, one builds

$$\pi_A(a,\theta,d) \;\propto\; u_A(a,\theta) \cdot p_A(\theta|d,a) \cdot p_A(d),$$

which is a sample from $\Pi_A(a,\theta,d)$. Using Smith and Roberts (1993) convergence results for MH algorithms, the samples generated in Algorithm 11 define a Markov chain with $\pi_A(a,\theta,d)$ as stationary distribution. Once convergence is detected, a consistent estimator (Chacon, 2020) of the modes of the a samples can be built which converges a.s. to mode$(\pi_A(a))$, providing a sample from A^* whose distribution is $\mathbb{P}_F[A^* \leqslant a] = p_D(A \leqslant a)$. Then, the samples generated through the `AAPS` function are distributed a.s. according to $p_D(a)$.

As $\mathcal{D}$ is compact and $\psi_D(d)$ is continuous in d, d^*_{ARA} exists. Since u_D is positive and integrable,

$$\pi_D(d,a,\theta) \;\propto\; u_D(d,\theta) \cdot p_D(\theta|d,a) \cdot p_D(a)$$

is well-defined, and is the stationary distribution of the MH Markov chain of the `DAPS` routine in Algorithm 10, using arguments in Smith and Roberts (1993). Once convergence is detected, the marginal samples in d are approximately distributed as $\pi_D(d)$, from which we approximate a.s. d^*_{ARA} through a consistent mode estimator (Chacon, 2020). □

Observe that Algorithm 10 (function `DAPS`) constructs an APS for the Defender problem, while Algorithm 11 (function `AAPS`) constructs an APS for the Attacker, which is iteratively called.

3.2.3 n-stage simultaneous defend-attack games

Figure 3.2 depicts the case of n stages with simultaneous decisions of both agents. Numerous DHS scenarios, such as the air-combat model presented in Virtanen et al. (2006), can be readily adapted to this framework. Likewise, reinforcement learning problems under threats, introduced in Gallego et al. (2019), can be reformulated to fit within this framework.

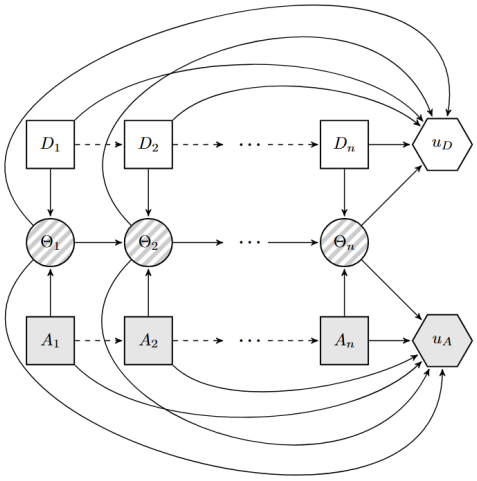


Figure 3.2: Template for basic n-stage simultaneous defend-attack game BAID.

Let us briefly discuss how the single-stage approach in Section 3.2.2 extends to a 2-stage case. Figure 3.3 presents the (a) defender's and (b) attacker's problems in this case. Assuming that the utility function $u_D(d_1, d_2, \theta_1, \theta_2)$, and the distributions $p_D(\theta_2|d_2, \theta_1, a_2)$, $p_D(a_2|a_1)$, $p_D(a_1)$, and $p_D(\theta_1|d_1, a_1)$ are available, we proceed as follows, using Shachter (1986)'s algorithm to solve the Defender's problem.

1. Remove chance node θ_2, computing $\psi_D(d_1, d_2, \theta_1, a_2) = \int u_D(d_1, d_2, \theta_1, \theta_2) p_D(\theta_2|d_2, \theta_1, a_2) \mathrm{d}\theta_2$.
2. Remove chance node A_2, computing $\psi_D(d_1, d_2, \theta_1, a_1) = \int \psi_D(d_1, d_2, \theta_1, a_2) p_D(a_2|a_1)\, \mathrm{d}a_2$.
3. Remove chance node θ_1, computing $\psi_D(d_1, d_2, a_1) = \int \psi_D(d_1, d_2, \theta_1, a_1) p_D(\theta_1|a_1, d_1)\, \mathrm{d}\theta_1$.
4. Remove chance node A_1, computing $\psi_D(d_1, d_2) = \int \psi_D(d_1, d_2, a_1) p_D(a_1)\, \mathrm{d}a_1$.
5. Remove decision node D_2, computing $\psi_D(d_1) = \max_{d_2} \psi_D(d_1, d_2)$ and storing $d_2^*(d_1) = \operatorname{argmax}_{d_2} \psi_D(d_1, d_2)$.
6. Remove decision node D_1, computing $d_1^* = \operatorname{argmax}_{d_1} \psi_D(d_1)$.

Remark 1. *We could combine steps 5 and 6, jointly removing nodes D_1 and D_2, to solve for $(d_1^*, d_2^*) = \operatorname{argmax}_{(d_1,d_2)} \psi_D(d_1, d_2)$. However in some contexts it is of interest to provide a full second-stage policy $d_2^*(d_1)$. In particular, this will be relevant when addressing A's problem.* △

Interestingly, we may implement the above six steps through two DAPS reductions:

DAPS1. Since

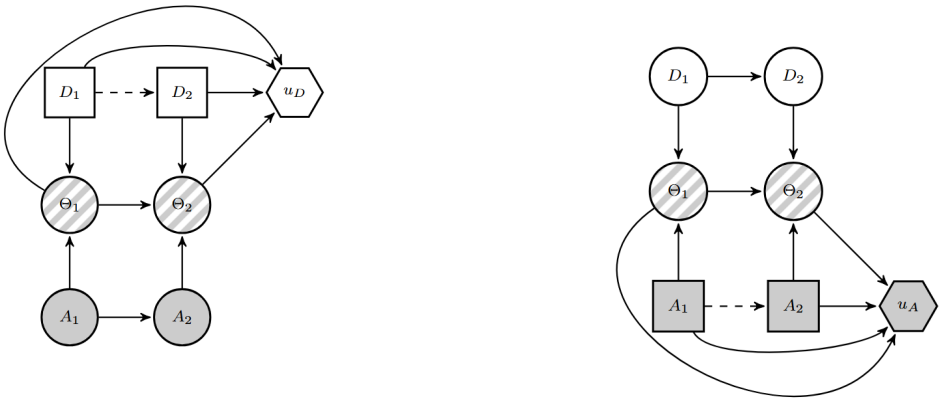
$$d_2^*(d_1) = \operatorname*{argmax}_{d_2} \iiiint u_D(d_1, d_2, \theta_1, \theta_2) \cdot p_D(\theta_2|d_2, \theta_1, a_2) \cdot p_D(a_2|a_1) \cdot p_D(\theta_1|d_1, a_1) \cdot p_D(a_1)\, \mathrm{d}a_1\, \mathrm{d}\theta_1\, \mathrm{d}a_2\, \mathrm{d}\theta_2,$$

we aggregate steps 1-5 by defining the AD

$$\pi(d_2, \theta_1, \theta_2, a_1, a_2|d_1) \propto u_D(d_1, d_2, \theta_2) \cdot p_D(\theta_2|d_2, \theta_1, a_2) \cdot p_D(a_2|a_1) \cdot p_D(\theta_1|d_1, a_1) \cdot p_D(a_1),$$

so that $\operatorname{mode}(\pi(d_2|d_1)) = d_2^*(d_1)$. Next, proceed by simulating from $\pi(d_2, \theta_1, \theta_2, a_1, a_2|d_1)$ and, based on a sample from it, obtain the corresponding sample from $\pi(d_2|d_1)$, using it to get a consistent estimate of its mode.

DAPS2. This handles step 6 and uses $d_1^* = \text{argmax}_{d_1}\, \psi_D(d_1)$. Although obvious,[ii] we view it through an AD with $\pi(d_1) \propto \psi_D(d_1)$ from which we sample to estimate its mode $\widehat{d_1^*}$.



(a) D's influence diagram. (b) A's influence diagram.

Figure 3.3: D's and A's influence diagrams for the 2-stage simultaneous defend-attack game.

As in Section 3.2.2, whereas $u_D(d_1, d_2, \theta_1, \theta_2)$, $p_D(\theta_2|d_2, \theta_1, a_2)$, and $p_D(\theta_1|d_1, a_1)$ are standard to elicit from a decision analytic point of view, $p_D(a_2|a_1)$ and $p_D(a_1)$ include strategic elements that need to be somehow reflected. For this, let us deal with the Attacker problem from the Defender point of view (Figure 3.3b), which is symmetric to her problem. If, as in Section 3.2.2, we model our uncertainty about its ingredients with a random utility function U_A and random probability distributions P_A, we would solve the Attacker problem through two AAPS reductions:

AAPS1. Define the RAD

$$\Pi(a_2, \theta_1, \theta_2, d_1, d_2|a_1) \;\propto\; U_A(a_1, a_2, \theta_1, \theta_2) \cdot P_A(\theta_2|a_2, \theta_1, d_2) \cdot P_A(d_2|d_1) \cdot P_A(\theta_1|a_1, d_1) \cdot P_A(d_1),$$

to estimate the random optimal decision $\widehat{A_2^*}(a_1)$. From it, estimate $\widehat{p_D}(a_2|a_1)$.

AAPS2. Define the RAD $\Pi(a_1) \propto \Psi(a_1)$. From it sample the random optimal attack $\widehat{A_1^*}$, and deduce $\widehat{p_D}(a_1)$.[iii]

The approach is actually general as Proposition 3.2 states. We omit its proof since it is a particular case of the, more general, Proposition 3.5 below.

Proposition 3.2. *An n-stage simultaneous game can be solved from an ARA perspective by using n DAPS reductions and n AAPS reductions.*

3.3 General security games

The solution process sketched in Section 3.2 for specific types of security games will be adapted, formalized, and expanded to solve general ones, providing a broad methodology to handle them based on ARA and APS. We present the complexities of solving such security games through a disinformation war case, later utilized to motivate and illustrate the methodology.

[ii] It is operationally obvious but computationally convenient to provide, later on, a unified approach, when facing general problems.

[iii] For completeness, we describe estimations of $\widehat{p_D}(a_2|a_1)$ and $\widehat{p_D}(a_1)$ but, effectively, we only require samples from them.

Case. According to the Global Risks Map report (WEF, 2024; WEF, 2025b), misinformation and disinformation will pose severe global risks in the short term, a situation further exacerbated by recent technological advances in e.g. LLMs (Barman et al., 2024). As the reports highlight, this threat potentially entail multiple impacts affecting DHS, the environment, or public health, just to quote a few.

Our case refers to foreign disinformation interference. We simplify assumptions to better illustrate the proposed concepts although the case is sufficiently complex to reflect the required modeling steps. Figure 3.4a refers to a scenario with two countries: an attacking one (A) that may attempt to interfere with a defending nation (D) through a disinformation campaign (DC) during a sensitive moment such as a pandemic. A seeks to disrupt D through a campaign encouraging D's citizens to adopt behaviors leading to infections, increasing D's hospital visits and healthcare costs. Even more, if a significant portion of D's population becomes infected, this could overwhelm its health system, potentially requiring the transfer of infected citizens to private hospitals, incurring in additional costs. D aims to allocate resources $d_1 \in D_1$ to proactively mitigate attacks by A, say by supporting disinformation debunking research projects. At the same time, A decides its investment level $a_1 \in A_1$ to enhance its disinformation capabilities, such as creating more sophisticated bots to disseminate it through social networks (Cresci, 2020). Subsequently, A determines the attack intensity $a_2 \in A_2$, for instance, by selecting the channels to propagate the disinformation and by targeting users more likely to spread it across these channels. Suppose A obtains information about D's investment d_1 through its intelligence services. A's investment a_1 and attack intensity a_2, and D's preparedness d_1 collectively determine the extent to which D recognizes that it is being attacked ($\theta_1 \in \Theta_1$). This understanding enables D to implement better reactive measures $d_2 \in D_2$, such as launching advertising campaigns to counter the disinformation spread. Actions d_2 and a_2, taken respectively by D and A together with θ_1, influence the number of individuals $\theta_2 \in \Theta_2$ affected by the campaign who, as a result, are more likely to become infected. This number, combined with the resources allocated at d_1 and d_2, defines D's utility, $u_D(d_1, d_2, \theta_2)$. Similarly, A's utility is $u_A(a_1, a_2, \theta_2)$. △

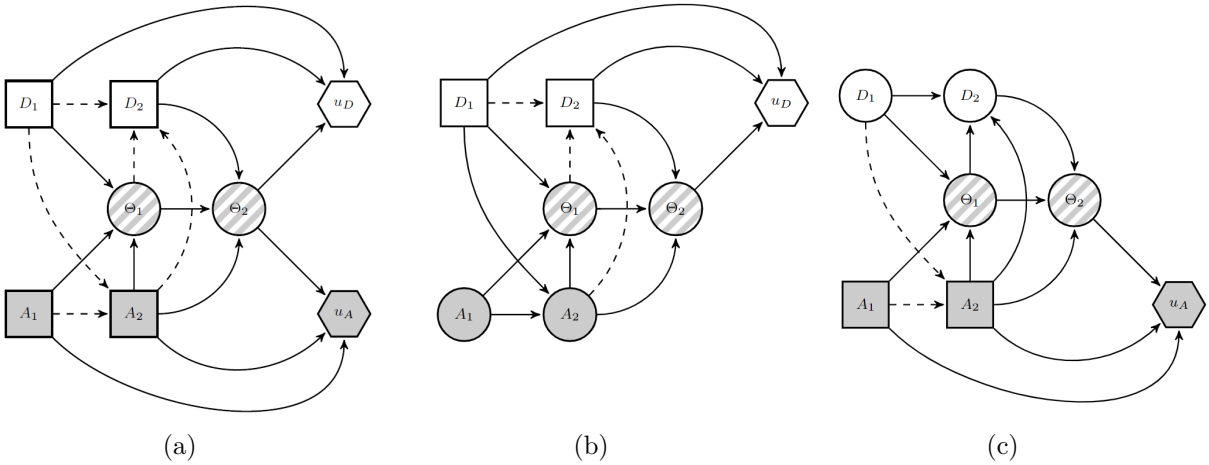


Figure 3.4: (a) BAIDS for DC case study, (b) D's influence diagram, (c) A's influence diagram.

Figures 3.4b and 3.4c respectively reflect IDs for the Defender and Attacker problems.

3.3.1 Methodology motivation

Let us first solve D's problem conceptually. Assume for now that we have available the required utilities and probabilities. Then, we treat D's ID as follows, using Shachter (1986) reduction operations.

$\mathcal{D}_1$: Remove chance node Θ_2, computing $\psi_D(d_1, d_2, \theta_1, a_2) = \int u_D(d_1, d_2, \theta_2) p_D(\theta_2 | d_2, a_2, \theta_1) d\theta_2$.
$\mathcal{D}_2$: Remove decision node D_2, computing the optimal $d_2^*(d_1, a_2, \theta_1) = \text{argmax}_{d_2} \psi_D(d_1, d_2, \theta_1, a_2)$ and storing the optimal expected utilities $\psi_D(d_1, a_2, \theta_1) = \max_{d_2} \psi_D(d_1, d_2, \theta_1, a_2)$.
$\mathcal{D}_3$: Remove chance node Θ_1, computing $\psi_D(d_1, a_1, a_2) = \int \psi_D(d_1, a_2, \theta_1) p_D(\theta_1 | d_1, a_1, a_2) \, \mathrm{d}\theta_1$.
$\mathcal{D}_4$: Remove chance node A_2, computing $\psi_D(d_1, a_1) = \int \psi_D(d_1, a_1, a_2) p_D(a_2 | d_1, a_1) \, \mathrm{d}a_2$.
$\mathcal{D}_5$: Remove chance node A_1, computing $\psi_D(d_1) = \int \psi_D(d_1, a_1) p_D(a_1) \, \mathrm{d}a_1$.
$\mathcal{D}_6$: Remove decision node D_1, obtaining the optimal first stage decision $d_1^* = \text{argmax}_{d_1} \psi_D(d_1)$.

Then, the optimal strategy for D would consist of selecting d_1^* at node D_1, and $d_2^*(d_1^*, a_2, \theta_1)$ at node D_2, given the a_2 and θ_1 observed. Note that, in principle, to implement steps $\mathcal{D}_3$-$\mathcal{D}_6$, it is not actually necessary to store the optimal action $d_2^*(d_1, a_2, \theta_1)$, but just the optimal value $\psi_D(d_1, a_2, \theta_1)$, from which we can subsequently reconstruct $d_2^*(d_1^*, a_2, \theta_1)$. This point is particularly relevant in continuous domains, as it allows for a reduction in storage requirements. Importantly, assuming with no loss of generality, that the utility function $u_D(d_1, d_2, \theta_2)$ is positive and integrable, the previous problem may be solved by applying two APS steps:

DAPS1. Aggregate steps $\mathcal{D}_1$-$\mathcal{D}_2$, corresponding to

$$d_2^*(d_1, a_2, \theta_1) = \underset{d_2}{\text{argmax}} \int u_D(d_1, d_2, \theta_2) \cdot p_D(\theta_2 | d_2, a_2, \theta_1) \, \mathrm{d}\theta_2.$$

For this, define the AD

$$\pi(d_1, d_2, \theta_2 | a_2, \theta_1) \propto u_D(d_1, d_2, \theta_2) \cdot p_D(\theta_2 | d_2, a_2, \theta_1).$$

By sampling from this distribution and consistently estimating the $\text{mode}\big(\pi(d_1 | a_2, \theta_1)\big)$, we approximate $d_2^*(d_1, a_2, \theta_1)$ and store $\widehat{\psi_D}(d_1, a_2, \theta_1) = \pi(d_1, d_2^*(d_1, a_2, \theta_1), \theta_2 | a_2, \theta_1)$, which is proportional to $\psi_D(d_1, a_2, \theta_1)$.
DAPS2. Aggregate steps $\mathcal{D}_3$-$\mathcal{D}_6$ corresponding to

$$d_1^* = \underset{d_1}{\text{argmax}} \iiint \widehat{\psi_D}(d_1, a_2, \theta_1) \cdot p_D(\theta_1 | d_1, a_1, a_2) \cdot p_D(a_2 | d_1, a_1) \cdot p_D(a_1) \, \mathrm{d}\theta_1 \, \mathrm{d}a_2 \, \mathrm{d}a_1.$$

For this, define the AD

$$\pi(d_1, a_2, \theta_1, a_1) \propto \widehat{\psi_D}(d_1, a_2, \theta_1) \cdot p_D(\theta_1 | d_1, a_1, a_2) \cdot p_D(a_2 | d_1, a_1) \cdot p_D(a_1). \tag{3.3.1}$$

By sampling from it and estimating $\text{mode}(\pi(d_1))$ consistently, we approximate d_1^*. We explain in Section 3.4 how to approximate $\widehat{\psi_D}(d_1, a_2, \theta_1)$ for outcome and decision spaces that are either discrete with high cardinality or continuous.

Of D's required utilities and probabilities, $p_D(a_2 | d_1, a_1)$ and $p_D(a_1)$ demand strategic thinking about A's problem. We could solve this should we have the Attacker's utility function, and probabilities. However, as argued, typically D will not be able to obtain such elements precisely. Suppose we represent the corresponding uncertainty with a distribution of random utilities and probabilities $F \sim (U_A(a_1, a_2, \theta_2), P_A(\theta_2 | \theta_1, d_2, a_2), P_A(d_2 | d_1, \theta_1, a_2), P_A(\theta_1 | d_1, a_1, a_2), P_A(d_1))$ to solve A's problem as follows, where random expected utilities and random optimal alternatives are introduced.

$\mathcal{A}_1$: Remove Θ_2, computing $\Psi_A(a_1, a_2, \theta_1, d_2) = \int U_A(a_1, a_2, \theta_2) P_A(\theta_2|\theta_1, d_2, a_2)\,\mathrm{d}\theta_2$.
$\mathcal{A}_2$: Remove D_2, computing $\Psi_A(d_1, \theta_1, a_1, a_2) = \int \Psi_A(a_1, a_2, \theta_1, d_2) P_A(d_2|d_1, \theta_1, a_2)\,\mathrm{d}d_2$.
$\mathcal{A}_3$: Remove Θ_1, computing $\Psi_A(d_1, a_1, a_2) = \int \Psi_A(d_1, \theta_1, a_1, a_2) P_A(\theta_1|d_1, a_1, a_2)\,\mathrm{d}\theta_1$.
$\mathcal{A}_4$: Remove A_2, computing $A_2^*(d_1, a_1) = \mathrm{argmax}_{a_2}\, \Psi_A(d_1, a_1, a_2)$ and storing $\Psi_A(d_1, a_1) = \Psi_A(d_1, a_1, A_2^*(d_1, a_1))$.
$\mathcal{A}_5$: Remove D_1, computing $\Psi(a_1) = \int \Psi_A(d_1, a_1) P_A(d_1)\,\mathrm{d}d_1$.
$\mathcal{A}_6$: Remove A_1, finding $A_1^* = \mathrm{argmax}_{a_1}\, \Psi_A(a_1)$.

Through $\mathcal{A}_1$-$\mathcal{A}_6$, we are able to obtain samples from $A_2^*(d_1, a_1)$ and A_1^* that, respectively, correspond to samples from $p_D(a_2|d_1, a_1)$ and $p_D(a_1)$, the missing ingredients to implement DAPS1-DAPS2. As in D's problem, we only require the random optimal value $\Psi_A(d_1, a_1) = \Psi_A(d_1, a_1, A_2^*(d_1, a_1))$ to implement $\mathcal{A}_5$-$\mathcal{A}_6$, which will be relevant when addressing nodes with continuous domains. However, in contrast to the Defender's problem, it is necessary to compute $A_2^*(d_1, a_1)$ (or at least approximate it in continuous problems) as it will be used in DAPS2. Again, APS allows us to implement the $\mathcal{A}_1$-$\mathcal{A}_6$ operations in a principled manner, through

AAPS1. Aggregates steps $\mathcal{A}_1 - \mathcal{A}_4$ defining

$$A_2^*(d_1, a_1) = \underset{a_2}{\mathrm{argmax}} \iiint U_A(a_1, a_2, \theta_2) \cdot P_A(\theta_2|\theta_1, d_2, a_2) \cdot P_A(d_2|d_1, \theta_1, a_2) \cdot P_A(\theta_1|d_1, a_1, a_2) \mathrm{d}\theta_2 \mathrm{d}d_2 \mathrm{d}\theta_1.$$

Consider the RAD

$$\Pi(a_1, a_2, \theta_2, d_2, \theta_1|d_1, a_1) \propto U_A(a_1, a_2, \theta_2) \cdot P_A(\theta_2|\theta_1, d_2, a_2) \cdot P_A(d_2|d_1, \theta_1, a_2) \cdot P_A(\theta_1|d_1, a_1, a_2).$$

By sampling from it, which entails previously sampling from each of the aforementioned distributions, we can estimate $\mathrm{mode}(\Pi(a_2|d_1, a_1))$ consistently, therefore estimating $A_2^*(d_1, a_1)$.

AAPS2. Aggregates steps $\mathcal{A}_5 - \mathcal{A}_6$. We solve for

$$A_1^* = \underset{a_1}{\mathrm{argmax}} \int \Psi_A(d_1, a_1) \cdot P_A(d_1)\,\mathrm{d}d_1.$$

Consider the RAD $\Pi(a_1, d_1) \propto \Psi_A(d_1, a_1) P_A(d_1)$. By sampling from it, which entails sampling from $\Psi_A(d_1, a_1)$ and $P_A(d_1)$, we consistently estimate $\mathrm{mode}(\Pi(a_1))$ to approximate A_1^*.

In conclusion, by solving two APS for the Defender (DAPS1 and DAPS2) and two random APS (RAPS) for the Attacker (AAPS1 and AAPS2) we are conceptually capable of solving the problem.

Importantly, the approach is actually general. Before discussing this, let us analyse two final examples that introduce additional computational issues not covered in the case.

Example 1. Consider D's ID in Figure 3.5. Assume we have $u_D(x_3, d)$, $p_D(x_1|d, x_2)$, $p_D(x_2|x_3)$ and $p_D(x_3)$ available.

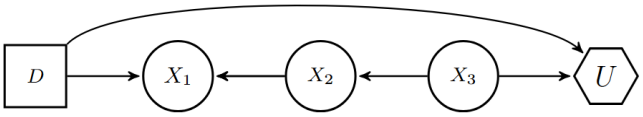


Figure 3.5: ID for problem where all nodes affected by arc inversion are reduced before node D.

Appealing to Shachter (1986) algorithm, we would go through the steps

- Invert arc $X_2 - X_3$, with updated probabilities $p_D(x_3|x_2) = p_D(x_2|x_3)p(x_3)/p_D(x_2)$ and $p_D(x_2) = \int p_D(x_2|x_3)p_D(x_3)\,\mathrm{d}x_3$.
- Eliminate X_3, obtaining the expected utility $\psi_D(x_2, d) = \int u_D(x_3, d)p_D(x_3|x_2)\,\mathrm{d}x_3$.
- Invert arc X_1-X_2 with updated probabilities $p_D(x_2|x_1, d) = p_D(x_2)p_D(x_1|x_2, d)/p_D(x_1|d)$ and $p_D(x_1|d) = \int p_D(x_1|x_2, d)p_D(x_2)\,\mathrm{d}x_2$.
- Eliminate X_2, obtaining the expected utility $\psi_D(x_1, d) = \int \psi_D(x_2, d)p_D(x_2|x_1, d)\,\mathrm{d}x_2$.
- Eliminate X_1, obtaining the expected utility $\psi_D(d) = \int \psi(x_1, d)p(x_1|d)\,\mathrm{d}x_1$.
- Eliminate D, obtaining the optimal decision $d^* = \operatorname{argmax} \psi_D(d)$.

Observe now that, assuming that $u_D(x_3, d)$ is positive, this can be solved in a single step by considering the AD

$$\begin{aligned}\pi(d, x_1, x_2, x_3) &\propto u_D(x_3, d) \cdot p_D(x_1|d) \cdot p_D(x_2|x_1, d) \cdot p_D(x_3|x_2) \\ &\propto u_D(x_3, d) \cdot p_D(x_1|d, x_2) \cdot p_D(x_2|x_3) \cdot p_D(x_3),\end{aligned}$$

defined using only the terms originally available.

Something similar would occur in an Attacker's ID. Assume that A's problem is also described by an ID as that in Figure 3.5. If D has access to the random probabilities and utilities $U_A(x_3, d)$, $P_A(x_1|d, x_2)$, $P_A(x_2|x_3)$, $P_A(x_3)$, the problem may be solved with the RAD

$$\Pi(d, x_1, x_2, x_3) \propto U_A(x_3, d) \cdot P_A(x_1|d, x_2) \cdot P_A(x_2|x_3) \cdot P_A(x_3),$$

again defined using only the original terms. △

Example 2. Consider the Defender problem in Figure 3.6. We have available $u_D(d, x_1)$, $p_D(x_1)$ and $p_D(x_2|x_1)$.

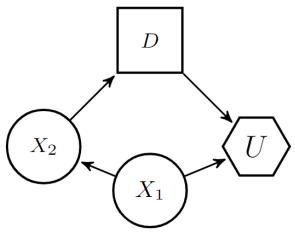


Figure 3.6: ID for problem where not all nodes affected by arc inversion are reduced before node D.

The steps followed to solve the problem are

- Invert arc $X_1 \to X_2$, computing $p_D(x_2) = \int p_D(x_2|x_1)p_D(x_1)\,\mathrm{d}x_1$ and $p_D(x_1|x_2) = p_D(x_2|x_1)$ $p_D(x_1)/p_D(x_2)$.
- Eliminate X_1, computing $\psi_D(x_2, d) = \int u_D(d, x_1)p_D(x_1|x_2)\,\mathrm{d}x_1$.
- Eliminate D, computing $d^*(x_2) = \operatorname{argmax}_d \psi_D(x_2, d)$.

Observe that, assuming $p_D(x_2) > 0$,

$$\psi_D(x_2, d) = \int u_D(x_1, d) \cdot \frac{p_D(x_2|x_1) \cdot p_D(x_1)}{p_D(x_2)} \, \mathrm{d}x_1 \propto \int u_D(x_1, d) \cdot p_D(x_2|x_1) \cdot p_D(x_1) \, \mathrm{d}x_1.$$

Define the AD $\pi_D(x_1, d|x_2) \propto u_D(x_1, d) p_D(x_2|x_1) p_D(x_1)$ and use the fact that $d^*(x_2) = \mathrm{mode}_d(\pi_D(x_1, d|x_2))$, to solve the problem with one APS, again based only on the original terms in the ID. △

Observe that in both examples we were able to define the AD and RAD models in terms of ingredients available in the original influence diagrams. In the first one, all nodes affected by arc inversions were reduced before node D and, therefore, the marginal terms cancelled. In turn, in the second one, some nodes affected by arc inversions were not reduced before node D but we could ignore the corresponding marginal in the denominator, being a positive constant multiplying the objective function of a maximization problem, therefore not affecting the maximum to be found.

3.3.2 General computational scheme

González-Ortega et al. (2019) introduced a scheme to deal with discrete BAIDs adapting ARA methods (Banks et al., 2015) and classic ID reductions from Shachter (1986). However, such reductions were discrete and solvable analytically. Inspired by the reasoning in Sections 3.2.3 and 3.3.1 we replace here such reductions in batches with APS schemes and take advantage of the fact that, for expected utility maximization, we do not need to take into account the denominators in Bayes' formula, as they are positive constants. In this way, we are able to efficiently handle BAIDs with continuous decision and chance nodes providing the first general solution to Shachter (1986)'s query.

Assume that the security game is represented by a proper BAID (Banks et al., 2015), i.e., an acyclic directed graph over decision, chance and utility nodes, where some chance nodes can be shared by both agents, such that from each decision maker's perspective, the corresponding IDs are proper (Shachter, 1986). Among other things, this implies that each agent has a directed path connecting their decision nodes, culminating in their respective utility node. We refer to these paths as the *Defender Decision Path*, $DDP = \{D_1, D_2, \ldots, D_n\}$, and the *Attacker Decision Path*, $ADP = \{A_1, A_2, \ldots, A_m\}$, where the Defender has n decisions to make and the Attacker, m decisions. Besides, the proper BAID definition implies that if two decisions are simultaneous, in the sense of Section 3.2, there is no directed path between them.

Drawing on the case and Section 3.2, let us introduce two core BAID operations that serve to solve general security problems. As proved afterwards, they require the Defender's utility function u_D and, a.s., the random utility function U_A to be positive and integrable. The first one, DAPS reduction, eliminates the last decision node in the DDP, together with the chance nodes to be eliminated before it, through an APS and serves to find the corresponding Defender's optimal decision, given the information available when making such decision.

Proposition 3.3. (**Defender APS reduction**, `DAPS_red`). *Consider the Defender's ID associated to a proper BAID. Let D be the last node in its DDP. Let $\mathcal{X}_c(D) = \{X_1, \ldots, X_{n_D}\}$ be the chance nodes to be eliminated before D (possibly including several arc inversions). Then, we can reduce the nodes $(D, \mathcal{X}_c(D))$ using APS with the value node inheriting the direct predecessors (dp) of $\mathcal{X}_c(D)$ not eliminated in this process (and storing the optimal value function).*

Proof. Let D be the last Defender's decision node and $\psi_D(y)$ be the current utility where y represents the instantiations of the dp of the value node. By assumption, the initial $\psi_D(y)$

is positive (we prove below that, in the other stages, it is positive by construction). Let $\mathcal{X}_c(D) = \{X_1, \ldots, X_{n_D}\}$ be the chance nodes to be eliminated before D, detectable applying Shachter (1986)'s algorithm qualitatively, possibly including several arc inversions, and $pr[X_i]$ designate the dp of node X_i. Then, by iteratively applying Shachter's operations, we eliminate the nodes in $(D, \mathcal{X}_c(D))$ by maximising expected utility, with the value node inheriting the dp of $\mathcal{X}_c(D)$ not eliminated in the process, that is $y' = \Big(y \backslash \big(\mathcal{X}_c(D) \cup D\big)\Big) \cup \big(pr[\mathcal{X}_c(D)] \backslash \mathcal{X}_c(D)\big)$ (and the optimal conditional decision to be stored $d^*(y')$).

Consider the AD

$$\pi(d, x_1, \ldots, x_{n_D} | pr[\mathcal{X}_c(D)]) \propto \psi_D(y) \prod_{X_i \in \mathcal{X}_c(D)} p_D(x_i | pr[X_i]),$$

which is well-defined as $\psi_D(y)$ is positive. If all nodes participating in arc inversions are eliminated, then $d^*(y') = \text{mode}(\pi(d|pr[\mathcal{X}_c(D)]))$, since the conditional marginal is proportional to the expected utility. Otherwise, the AD is proportional to the expected utility, except for a positive constant, and, again, $d^*(y') = \text{mode}(\pi(d|pr[X_c(D)]))$. The new evaluation is

$$\psi_D(y') = \max_d \int \cdots \int \psi_D(y) \prod_{X_i \in \mathcal{X}_c(D)} p_D(x_i | pr[X_i]) \, \mathrm{d}x_1 \ldots \mathrm{d}x_{n_D},$$

which is positive by construction. □

The second one, AAPS reduction, operates in the Attacker ID. It eliminates the last decision node in the *ADP*, together with some chance nodes, and serves to find D's distribution over the corresponding Attacker's decisions.

Proposition 3.4. (**Attacker APS reduction,** `AAPS_red`). *Consider the Attacker's ID associated to a proper BAID. Let A be the last node in its ADP. Let $\mathcal{X}_c(A) = \{X_1, \ldots, X_{n_A}\}$ be the chance nodes to be eliminated before A (possibly including several arc inversions). Then, we can reduce the nodes $(A, \mathcal{X}_c(A))$ using RAPS with the value node inheriting the dp of $\mathcal{X}_c(A)$ not eliminated in this process. The random optimal conditional decisions are recorded to predict the Attacker's decisions (and the random optimal value function is stored).*

Proof. Let A be the last Attacker's decision node and $\Psi_A(y)$ be the current random utility where y represents the instantiations of the dp of the value node, which includes D. By assumption, the initial $\Psi_A(y)$ is a.s. positive (we prove below by construction that it will also be a.s. positive for earlier nodes). Let $\mathcal{X}_c(A) = \{X_1, \ldots, X_{n_A}\}$ be the chance nodes to be eliminated before A using Shachter, 1986's algorithm, possibly including several arc inversions, and $pr[X_i]$ the dp of node X_i. By applying iteratively Shachter, 1986's operations, we eliminate the nodes in $(A, \mathcal{X}_c(A))$ by maximizing the random expected utility with the value node inheriting the dp of $\mathcal{X}_c(A)$ not eliminated in the process, that is $y' = \Big(y \backslash \big(\mathcal{X}_c(A) \cup A\big)\Big) \cup \big(pr[\mathcal{X}_c(A)] \backslash \mathcal{X}_c(A)\big)$ and the random optimal decisions to be stored are $A^*(y')$.

Consider the RAD

$$\Pi(a, x_1, \ldots, x_{n_A} | pr[\mathcal{X}_c(A)]) \propto \Psi_A(y) \prod_{X_i \in \mathcal{X}_c(A)} P_A(x_i | pr[X_i]),$$

which is a.s. well-defined as $\Psi_A(y)$ is positive a.s. If all nodes participating in arc inversions are eliminated, then $A^*(y') = \text{mode}(\Pi(a|pr[\mathcal{X}_c(A)]))$, since the random conditional marginal

is a.s. proportional to the random expected utility. Otherwise, the RAD is a.s. proportional to the random expected utility, except for a positive constant and, therefore, again $A^*(y') = \text{mode}(\Pi(a|pr[\mathcal{X}_c(A)]))$. The new random evaluation is defined as

$$\Psi_A(y') = \max_a \int ... \int \Psi_A(y) \prod_{X_i \in \mathcal{X}_c(A)} P_A(x_i|pr[X_i])\, \mathrm{d}x_1 \ldots \mathrm{d}x_{n_A},$$

which is a.s. positive. □

Note that DAPS_red needs the probability distributions originally available to D and D's distribution forecasts over A's decisions, allowing efficient sampling from these distributions, avoiding costly arc inversions, and greatly facilitating the computation of the optimal decision. Similarly, AAPS_red requires the random probability distribution initially assigned by D to A, thus allowing for efficient execution of the operation and, actually, not requiring explicit arc inversions.

Based on both results we provide Algorithm 12 to solve general proper BAIDs regardless of the number of Defender's and Attacker's decision nodes and their temporal interrelations. We use the notation $d^*(\cdot)$ ($\psi_D(\cdot)$) to indicate the arguments $(\cdot)$ upon which the optimal decision d^* (value function $\psi(\cdot)$) at a node depends; similarly, $A^*(\cdot)$ refers to the variables upon which the random optimal decision A^* (random value function $\Psi(\cdot)$) at a given node depends. Algorithm 12 outputs the optimal decision $d_i^*(\cdot)$ for D at each of her decision nodes D_i and, as useful complementary information, the forecast $A_j^*(\cdot)$ of A's actions at each of his decision nodes A_j, taking as input the BAID B with D's distributions and A's random distributions. Note that in steps 1 and 2, we include the "no forgetting" arcs (Shachter, 1986) in the Defender (Attacker) ID. These are the edges connecting the dp of decision node D_i (A_i) with decision node D_j (A_j), where D_i (A_i) precedes D_j (A_j) in the DDP (ADP).

Algorithm 12 General algorithm to solve BAIDs using ARA and APS

Input: B

1: Define Attacker ID, identify ADP and add "no forgetting" arcs
2: Define Defender ID, identify DDP and add "no forgetting" arcs
3: **while** $DDP \neq \varnothing$ **do**
4: Find last decision node D_i in DDP
5: **if** all p_D for DAPS_red of D_i available **then**
6: Apply DAPS_red, storing optimal decision $d_i^*(\cdot)$ and value function $\psi_D(\cdot)$
7: Update Attacker's beliefs
8: Eliminate D_i from DDP
9: **else**
10: Find last decision node A_j in ADP
11: Apply AAPS_red, obtaining optimal random decision $A_j^*(\cdot)$ and value function $\Psi_A(\cdot)$
12: Update Defender's beliefs
13: Eliminate A_j from ADP
14: **Return:** $\{d_1^*(\cdot), \ldots, d_n^*(\cdot), A_1^*(\cdot), \ldots, A_m^*(\cdot)\}$

Algorithm 12 implements n DAPS reductions and m AAPS reductions to provide the optimal policy, as proved below. When applying a DAPS (AAPS) reduction, it may happen that the following D_i (A_i) in DDP (ADP) becomes a barren node, that is, it has no successors (Shachter, 1986). In this case, the solution for D_i (A_i) will be a degenerate case, since any alternative would be optimal (Shachter, 1986).

Proposition 3.5. *In a proper BAID, Algorithm 12 provides the optimal policy for the Defender.*

Proof. If the BAID is proper, we can define the corresponding ADP and DDP. Each time we find the corresponding last D_i in the DDP.

- If all p_D's are available for reduction, we reduce node D_i and eliminate it from the DDP using Proposition 3.3, with the utility model being proportional to the expected utility model, therefore leading to the same optimal decision.
- If not, we find the required p_D's through `AAPS_red` operations, eliminating the corresponding nodes from the ADP, using Proposition 3.4, possibly several times. Then, reduce the corresponding D_i from the DDP, with the same optimality preserving features.

As both the DDP and the ADP are finite, the algorithm necessarily terminates providing the required output as deduced from Propositions 3.3 and 3.4. □

3.4 Computational issues

Algorithm 12 utilizes backward dynamic programming and, implicitly, probabilistic inversion. It is directly applicable only in security games with discrete and low-cardinality decision and uncertainty spaces. To illustrate this, consider the DAPS1 step when solving the problem in Figure 3.4b. To find the optimal d_1^* using APS, we sample from the AD in (3.3.1). This is typically performed with MH, creating a Markov chain on $(d_1, a_2, \theta_1, a_1)$ by proposing candidates for these variables and updating them based on an acceptance probability that depends, among other factors, on $\psi_D(d_1, a_2, \theta_1)$ evaluated at both the old and new candidate values. If the decision and outcome spaces have sufficiently low cardinality, it is practical to solve DAPS2 to compute ψ_D at every feasible (d_1, a_2, θ_1). Similarly, empirical estimates of $p_D(a_2|d_1, a_1)$ can be built for all feasible (d_1, a_1). However, this approach is not applicable in continuous decision spaces and, in the discrete case, can lead to a combinatorial explosion as the cardinality of decision spaces and the number of stages increase. In such scenarios, it is necessary to develop models that can effectively approximate future optimal decisions and predict attack distributions based on a few cases. This section outlines computational details necessary to efficiently apply Algorithm 12 under these circumstances, illustrated in Section 3.5.

3.4.1 Approximating Defender's optimal value functions

In continuous problems or problems with high cardinality decision and/or uncertainty spaces, computing optimal values for the Defender, such as $\psi_D(d_1, a_2, \theta_1) = \max_{d_2} \psi_D(d_1, d_2, a_2, \theta_1)$ for every (d_1, a_2, θ_1) becomes infeasible. However, we can approximate optimal value functions with a statistical metamodel (Law et al., 2007) to be propagated backwards to earlier decision stages. To illustrate this, let us discuss DAPS1 in the disinformation war case, though the arguments are general.

To develop the metamodel, collect first data from simulations that represent various scenarios within the (d_1, a_2, θ_1) spaces, covering a broad range of values through a grid $(d_1, a_2, \theta_1)_{j=1}^J$ to ensure metamodel robustness. For each j, the optimal d_2 is computed by invoking DAPS1, and $\psi_D(d_1, a_2, \theta_1)_j$ is estimated via MC. Denote the data by $\mathcal{D} = \{(d_1, a_2, \theta_1)_j, \psi_D(d_1, a_2, \theta_1)_j\}_{j=1}^J$. Next, a regression metamodel $\widehat{\psi}_{D,\gamma}(d_1, a_2, \theta_1)$ with covariates (d_1, a_2, θ_1), and parameterized by γ, is trained to predict $\psi_D(d_1, a_2, \theta_1)$ using $\mathcal{D}$. The selection of an appropriate statistical modeling technique is crucial, with neural networks (NN) (Gallego and Insua, 2022) as prime example. The metamodel's performance is assessed using metrics such as mean squared error, and evaluated

on a hold-out dataset. Model parameters are tuned mainly through cross-validation (CV). This process of model development and refinement iterates to achieve better performance. Once a sufficiently good metamodel is available, it is integrated within the backward induction process. In our particular case, when simulating from the AD in (3.3.1) using MCMC to solve DAPS2, the metamodel $\widehat{\psi}_{D,\gamma}(d_1, a_2, \theta_1)$ will be used instead of the original optimal value function, allowing us to evaluate it for every possible candidate value. However, we still need to be able to sample from $p_D(a_2|d_1, a_1)$ for every possible d_1, a_1. We next explain how to build an approximate sampler.

3.4.2 Approximately sampling from random optimal attacks

Again, in continuous problems or problems with high cardinality decision and/or uncertainty spaces, sampling from attack distributions such as $p_D(a_2|d_1, a_1)$ for every possible (d_1, a_1) or, equivalently, computing random optimal values for the Attacker, such as $\Psi_A(d_1, a_1) = \max_{a_2} \Psi(d_1, a_1, a_2)$ is infeasible. Alternatively, we can approximate the Attack distribution and random optimal value function with a statistical metamodel, to be propagated backward to earlier stages. To illustrate this, let us discuss AAPS1 in the case though, again, the arguments are general.

As before, we collect data from simulations for a broad range of values for d_1, a_1 to ensure robustness, through a grid $(d_1, a_1)_{j=1}^{J}$, where J represents the number of data points. For each value j, we produce a sample from the conditional probability $p_D(a_2|d_1, a_1)$ using AAPS1. Let $\mathcal{D} = \left\{(d_1, a_1)_j, \{a_2\}_{k=1}^{K}\right\}_{j=1}^{J}$ denote the data, with K the number of samples for each point in the grid. Once $\mathcal{D}$ has been obtained, $\{\Psi_A(d_1, a_1)\}_{k=1}^{K}$ is computed using MC simulation, resulting in the dataset $\mathcal{D}' = \{(d_1, a_1)_j, \left(\{\Psi_A(d_1, a_1)\}_{k=1}^{K}\right)_j\}_{j=1}^{J}$. Next, we choose appropriate statistical models $\widehat{p_D}(a_2|d_1, a_1)$ and $\widehat{\Psi}_A(d_1, a_1)$ to approximate the Attack distribution and the random optimal value function, respectively. Methods such as kernel density estimation, Gaussian processes or NN (Bishop, 2006) can be used as basic models. Note that if the Attack decision space or random value function are bounded, a transformation may be necessary before modeling. Model performance is evaluated using likelihood-based measures, assessed on a hold-out dataset. Parameters are tuned through CV to improve accuracy, with possibly several iterations of model development and refinement as necessary to achieve better performance. Once the models are developed, they are integrated within the backward induction process. Specifically, $\widehat{p_D}(a_2|d_1, a_1)$ is used in place of the original distribution to sample efficiently from the approximated attack distribution for every (d_1, a_1), which will be used to solve DAPS2. Similarly, $\widehat{\Psi}_A(d_1, a_1)$ will be used instead of the original random optimal value function when using MCMC to solve AAPS2.

3.4.3 Availability of probabilities

Algorithm 12 includes two *update beliefs* statements (7 and 12). First (7), the **if** condition demands whether all p_D are available to implement `DAPS_red` for the last decision node D_i in DPP. To check this, we qualitatively apply Shachter's algorithm to find $\mathcal{X}_c(D_i)$. If none of the nodes in this set corresponds to Attacker's uncertainties that have not been treated before, then declare all p_D as available. Otherwise, we move to the **else** block and reduce the required A_j nodes in the ADP inverse sequence order until deducing the corresponding attack distributions based on Section 4.2 and updating the corresponding $p_D(a_j|.)$ distributions.

The second beliefs update (12) refers to assessing the Attacker's beliefs for the purpose of later iterations. Originally we set up such distributions as non-informative ones, given the lack of knowledge. At later stages, as we find the optimal decisions $d_i^*(\cdot)$, we can reassess the $p_A(d_i|.)$

distributions as centered around $d_i^*(\cdot)$, with some uncertainty around them.

3.4.4 Enhancing the efficiency of APS

In all D and A stages, we increase the efficiency of APS by sampling from a power transformation of the corresponding marginal AD or RAD, see Müller et al. (2004): we substitute the marginal augmented distribution $\pi(\cdot)$ by a power transformation $\pi^H(\cdot)$ where $H \geqslant 1, H \in \mathbb{N}$ is an augmentation parameter. This transformation is more peaked around the mode, facilitating mode identification.

Let us focus on the Defender's problem in Algorithm 10 (`DAPS`) to illustrate how to sample from $\pi^H(\cdot)$. Specifically, in that algorithm, sampling from $\pi^H(d)$ can be performed by drawing H copies of $\tilde{a}$ and $\tilde{\theta}$, instead of just one, and modifying the acceptance probability at the i-th iteration by $\min\left\{1, \frac{g(d^{(i-1)}|\tilde{d})}{g(\tilde{d}|d^{(i-i)})} \prod_{t=1}^{H} \frac{u_D(\tilde{d},\tilde{\theta}_t)}{u_D(d^{(i-1)},\theta_t^{(i-1)})}\right\}$. For more details on the computations used to obtain this acceptance probability, we refer the reader to the supplementary materials in Ekin et al. (2023).

3.5 Disinformation campaign case study

3.5.1 Basic modeling assumptions

Let us illustrate the methodology with a numerical instance of the disinformation war case in Figure 3.4. Appendix B details D's parametric assessments concerning both her decision-making problem and her perspective on how A would address his. Table 3.1 summarises the nodes in the BAID and their ranges. Feasible values for D_1 (A_1) are viewed as the proportion of the available budget that D (A) aims to allocate. For example, $d_1 = 0$ indicates a minimum investment in D's defense program. Similarly, A_2 corresponds to the intensity with which A aims the attack to occur, with $a_2 = 0$ meaning that he decides not to attack: higher values increase the number of individuals exposed to the campaign and enhance its effectiveness. d_2 represents the proportion of available budget for D_2 utilised by D for reactive countermeasures. θ_1 represents the degree with which the Defender recognizes the disinformation campaign, allowing resources invested in D_2 to be more effective: higher θ_1 suggests a better detected attack. $\Theta_2 \in \{0, 1, \ldots, n\}$ represents the number of citizens susceptible of becoming affected by the campaign and, thus, will be likely to be infected. D's utility u_D and A's utility u_A respectively range in $(0, y_D]$ and $(0, y_A]$.

Table 3.1: Nodes, interpretation, and feasible values for BAID in Figure 3.4a.

Node	Concept	Range
D_1	Resources to proactively defend against disinfo	[0,1]
D_2	Resources to reactively protect against disinfo	[0,1]
A_1	Resources for more efficient disinfo campaign	[0,1]
A_2	Intensity of disinfo campaign	[0,1]
Θ_1	Recognition degree of disinfo campaign	[0,1]
Θ_2	Individuals affected by disinfo campaign	$\{0, 1, \ldots, n\}$
u_D	D's utility	$(0,y_D]$
u_A	A's utility	$(0,y_A]$

3.5.2 Results

This section presents the results of the proposed analysis providing D's optimal decisions at nodes D_1 and D_2. Additionally, our approach provides probabilistic predictions of A's actions at nodes A_1 and A_2, serving to compute D's optimal decisions but also to raise awareness about likely attacks. In all decision stages, we use the method in Section 3.4.4 to enhance APS efficiency, with augmentation parameters H provided in Appendix B.

Given the continuous nature of the decision and uncertainty spaces, we draw on the approaches in Section 3.4. We approximate $\psi_D(d_1, a_2, \theta_1)$ (Section 3.4.1), $p_D(a_2|d_1, a_1)$ and $\Psi_A(d_1, a_1)$ (Section 3.4.2) with multilayer perceptrons (MLPs) (Gallego and Insua, 2022). Regarding $\widehat{\psi}_D(d_1, a_2, \theta_1)$, the MLP is trained to approximate a scalar function by minimizing mean squared error (Bishop, 2006) using the dataset $\mathcal{D}_{\psi_{D_2}} = \{(d_1, a_2, \theta_1)_j, \psi_D(d_1, a_2, \theta_1)_j\}_{j=1}^J$. For $\widehat{p}_D(a_2|d_1, a_1)\big(\widehat{\Psi}_A(d_1, a_1)\big)$, the MLP is trained to approximate density functions by minimizing the negative log-likelihood (NLL) of the distribution it parametrizes, which we specify below, evaluated at the samples in the dataset $\mathcal{D}_{A_2} = \{(d_1, a_1)_j, \{a_2^*(d_1, a_1)\}_{k=1}^K\}_{j=1}^J$ $\big(\mathcal{D}_{\Psi_{A_2}} = \{(d_1, a_1)_j, \big(\{\Psi_A(d_1, a_1)\}_{k=1}^K\big)_j\}_{j=1}^J\big)$.

To select the best MLP architecture for each case, and evaluate the performance of the models, we proceed as follows: we split the corresponding dataset $\mathcal{D}_i, i \in \{\psi_{D_2}, A_2, \Psi_{A_2}\}$, into 80% for training and 20% for testing. The training set is used to select the MLP architecture via 5-fold CV with early stopping, repeating the process 10 times. Using the selected architecture, we train an MLP on the full training set and evaluate it on the test set to assess performance. Finally, we retrain the MLP with the selected hyperparameters on the entire dataset $\mathcal{D}_i$ to obtain the model to be propagated backwards to earlier decision stages. For each stage, we present only the finally selected architecture for the corresponding NN. Details regarding their selection and performance are in Appendix B.

DAPS1. Defender's second stage decision

For a grid of (d_1, a_2, θ_1) values we: (a) compute the optimal decision $d_2^*(d_1, a_2, \theta_1)$; and (b) approximate $\psi_D(d_1, a_2, \theta_1) = \max_{d_2} \psi_D(d_1, d_2, a_2, \theta_1)$, as in Section 3.4.1, based on a grid of $J = 9621 = 21^3$ (d_1, a_2, θ_1) values, obtained by uniformly dividing the sets D_1, A_2, and Θ_1 with splits of 0.05.

(a) The optimal $d_2^*(d_1, a_2, \theta_1)$ is obtained by invoking DAPS1 for each (d_1, a_2, θ_1), obtaining $\mathcal{D}_{D_2} = \{(d_1, a_2, \theta_1)_j, d_2^*(d_1, a_2, \theta_1)_j\}_{j=1}^J$. Figure 3.7 (Left) displays the mean of $d_2^*(d_1, a_2, \theta_1)$ over the values of d_1; Figure 3.7 (Right) shows the expected number $E[\theta_2|d_2^*(d_1, a_2, \theta_1), a_2, \theta_1]$ of infected individuals, obtained by determining $d_2^*(d_1, a_2, \theta_1)$ and performing a MC simulation for each $\big((d_2^*(d_1, a_2, \theta_1))_j, (a_2, \theta_1)_j\big)$. Colors in this figure indicate whether the infected individuals can be accommodated within D's public healthcare system (green) or if excess cases must be transferred to the private sector (red), showcasing the protection level that d_2^* provides to D's health system. Observe that $E[\theta_2 \mid d_2^*(d_1, a_2, \theta_1), a_2, \theta_1]$ could have been computed analytically. However, we chose to use MC simulation to approximate this expectation in order to illustrate how to conduct the analysis in general, regardless of whether the analytical form of the chance node is known.

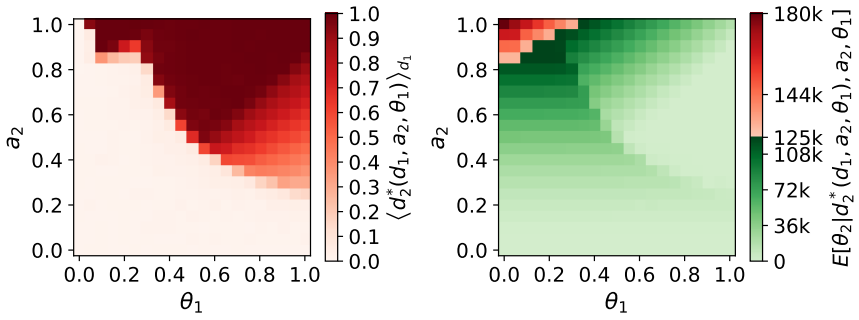


Figure 3.7: (Left) Mean $\langle d_2^*(d_1, a_2, \theta_1)\rangle_{d_1}$ of $d_2^*(d_1, a_2, \theta_1)$ over d_1. (Right) Expected number $E[\theta_2 \mid d_2^*(d_1, a_2, \theta_1), a_2, \theta_1]$ of individuals to be infected.

Figure 3.7 (Left) reveals three regions for $d_2^*(d_1, a_2, \theta_1)$. First, practically no defense ($d_2^*(d_1, a_2, \theta_1) \approx 0$, white region) is deployed when recognition (θ_1) is low, except for high intensity attacks ($a_2 \gtrsim 0.9$), as d_2 would be ineffective in reducing infections. Additionally, no defense is deployed when healthcare costs are lower than the required d_2 investment, even at high θ_1, since D's health system can manage the outbreak. Second, an intermediate defense ($0 < d_2^*(d_1, a_2, \theta_1) < 1$, pale to medium red) occurs when an increase in recognition (θ_1) enhances the efficiency of d_2, reducing infections with a lower investment, thus preventing both D's under- and overspending. Finally, full deployment of defenses ($d_2^*(d_1, a_2, \theta_1) \approx 1$, dark red) takes place when recognition (θ_1) is low under a high-intensity attack ($a_2 \gtrsim 0.9$), in an effort to mitigate numerous infections that may have severe consequences, even if the approach is inefficient. This option is also preferred when the investment prevents infected individuals from being transferred to the private sector, as the number of infections remains within the healthcare system's capacity (red transitioning to green in Figure 3.7 (Right)).

(b) For each $\big((d_1, a_2, \theta_1)_j, d_2^*(d_1, a_2, \theta_1)_j)\big)$ in $\mathcal{D}_{D_2}$, we compute $\psi_D(d_1, a_2, \theta_1)$ via MC obtaining $\mathcal{D}_{\psi_{D_2}}$. We approximate $\psi_D(d_1, a_2, \theta_1)$ by fitting a MLP with three hidden layers containing 32, 64, and 16 neurons, respectively, and ReLU activation functions. This approximation is used when computing the Attacker's first-stage decision.

AAPS1. Prediction of Attacker's second stage decision

For a grid of values (a_1, d_1), we approximate (a) the random optimal decisions $A_2^*(d_1, a_1)$ with density $p_D(a_2|d_1, a_1)$, and (b) the random expected utility $\Psi_A(d_1, a_1) = \max_{a_2} \Psi_A(d_1, a_1, a_2)$. The grid used has $J = 41^2 = 1681$ combinations of (d_1, a_1) generated by uniformly splitting D_1 and A_1 with splits of 0.025.

(a) Compute $\{a_2^*(d_1, a_1)_k\}_{k=1}^{100}$ for each (d_1, a_1) invoking AAPS1, thereby simulating the behavior of 100 attackers at each point, leading to the dataset $\mathcal{D}_{A_2}$. Figure 3.8 displays the estimated expected random optimal attack ($\overline{A}_2^*(d_1, a_1)$), the mean of $\{a_2^*(d_1, a_1)_k\}_{k=1}^{100}$ for each (d_1, a_1).

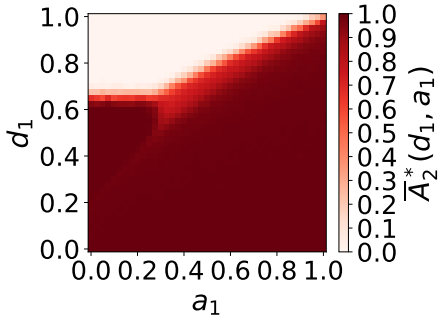


Figure 3.8: Expected random optimal attack $A_2^*(d_1, a_1)$ in second stage.

Figure 3.8 suggests three behaviors for $\overline{A}_2^*(d_1, a_1)$. (1) $\overline{A}_2^*(d_1, a_1) \approx 0$ (white region), occurring when d_1 is much higher than a_1; A observes that D invested significantly more in d_1 than he has in a_1, deciding not to attack, as success is unlikely. (2) $\overline{A}_2^*(d_1, a_1) \approx 1$ (dark red): when a_1 is low, A launches a high-intensity attack as long as D's proactive investment remains moderate ($d_1 \lesssim 0.6$). As a_1 increases, A continues to launch high-intensity attacks even at higher d_1, as greater investment in a_1 enables implementing more impactful attacks (a_2) that remain effective despite stronger defensive measures by D. Finally, (3) when $0 < \overline{A}_2^*(d_1, a_1) < 1$ (moderate red): A launches an intermediate-intensity attack that is sufficiently effective to harm D, but refrains from a high-intensity attack, as the cost of a stronger a_2 is not justified by its expected impact given the investments (d_1, a_1).

We approximate $A_2^*(d_1, a_1)$ using the dataset $\mathcal{D}_{A_2}$ with an MLP that outputs parameters $\{(w_i, \alpha_i, \beta_i)\}_{i=1}^2$ defining a mixture of Beta distributions for each input (d_1, a_1). The MLP consists of three hidden layers with 64 neurons each, using ReLU activations in the intermediate layers, softmax activation for the output neurons corresponding to $\{w_i\}_{i=1}^2$, and a softplus activation for those corresponding to $\{(\alpha_i, \beta_i)\}_{i=1}^2$. The approximation is integrated into the Defender's first-stage decision process.

(b) For each (d_1, a_1) in $\mathcal{D}_{A_2}$, we compute $\Psi_A(d_1, a_1)$ via MC simulation, resulting in $\mathcal{D}_{\Psi_{A_2}}$. To approximate $\Psi_A(d_1, a_1)$, we fit to $\mathcal{D}_{\Psi_{A_2}}$ an MLP that outputs the parameters $\{(w_i, \lambda_i, \alpha_i)\}_{i=1}^2$ characterizing a mixture of Weibull distributions for each (d_1, a_1). Again, the MLP comprises three hidden layers with 64 neurons each, employing ReLU activations in the hidden layers, softmax activation for the output neurons associated with $\{w_i\}_{i=1}^2$, and softplus activation for those corresponding to $\{(\lambda_i, \alpha_i)\}_{i=1}^2$. $\widehat{\Psi}_A(d_1, a_1)$ is used in the Attacker's first stage.

AAPS2. Prediction of Attacker's first stage decision

Using $\widehat{\Psi}_A(d_1, a_1)$, we invoke AAPS2 to generate 10000 samples of A_1^* and display their density $(p_D(a_1))$ in Figure 3.9. Observe that, from D's perspective, A adopts two strategies: most likely choosing not to allocate resources to produce a more effective disinformation campaign ($a_1 \approx 0$), or, though much less likely, investing all resources ($a_1 \approx 1$). A mixture of Beta distributions is employed to approximate A_1^*, and used in the Defender's first stage.

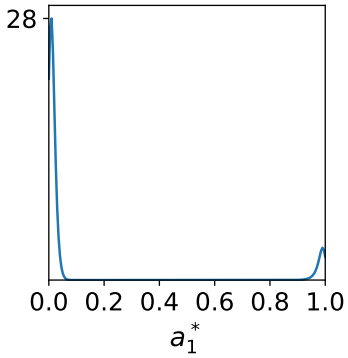


Figure 3.9: Density of empirical distribution A_1^*.

DAPS2. Defender's first stage decision

With the approximations $\widehat{\psi}_D(d_1, a_2, \theta_1)$, $\widehat{p}_D(a_2|d_1, a_1)$, and $\widehat{p}_D(a_1)$, finding the Defender's first stage decision involves applying APS by invoking DAPS2. The optimal decision is $d_1^* = 0.7$, suggesting that D's best strategy involves allocating a moderate proportion of her budget to proactive measures.

Let us analyze the implications of d_1^* through Figure 3.7 (Right), Figure 3.8 and Figure 3.10, displaying $E[\theta_1|d_1^*, a_1, a_2]$. Notice two effects. First, d_1^* acts as a deterrent, leading A to not attack on average when it has previously invested at least a moderate to low amount in A_1 ($a_1^* \approx 0.3$), as observed in Figure 3.8. Second, if A decides to attack, assuming $a_1 > 0.3$, either with a moderate or high-intensity attack (Figure 3.8), d_1^* enables a moderately high degree of recognition θ_1 (Figure 3.10), helping to mitigate a_2, as countermeasures d_2 will be more efficient, contributing to reducing the potential harm to D's health system (Figure 3.7(Right)).

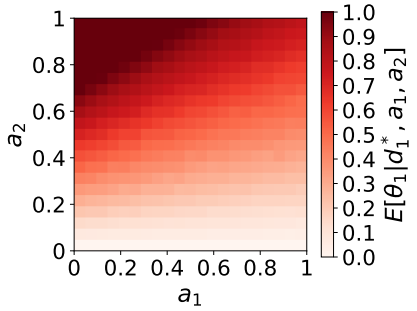


Figure 3.10: Expected degree of recognition $E[\theta_1 \mid d_1^*, a_1, a_2]$ for d_1^* across different (a_1, a_2).

Summary of Defender's optimal policy and Attacker's random optimal policy

Defender's optimal policy. In the first stage, the Defender invests moderately in proactive defense against disinformation ($d_1^* = 0.7$). Given the Attacker's random optimal policies for both decisions (A_1^*, $A_2^*(d_1, a_1)$), the Defender faces the following scenarios according to her beliefs: either $a_2 \approx 0$ or $a_2 \approx 1$. Specifically, if $a_2 \approx 0$, the Defender should not implement reactive defenses ($d_2^* \approx 0$); otherwise, if $a_2 \approx 1$, the Defender should use all its resources to protect against the attack ($d_2^* \approx 1$).

Attacker's random optimal policy. In the first stage, the Attacker, most likely according to her beliefs, does not invest in a more effective disinformation campaign ($a_1 \approx 0$). However, in rare cases, the Attacker may commit all available resources to improve its capabilities ($a_1 \approx 1$). In the

second stage, if $a_1 \approx 0$, the Attacker will refrain from launching an attack ($a_2 \approx 0$). Alternatively, if $a_1 \approx 1$, the Attacker will carry out a maximum-strength attack ($a_2 \approx 1$).

3.5.3 Sensitivity analysis

We perform sensitivity analysis on two key parameters specified in Appendix B.

Effect of ω_{d_2} on the Defender's second stage decision

This parameter determines the effectiveness of the defensive measures d_2 and the recognition level θ_1 in countering attack a_2. When $\omega_{d_2} > 1$, d_2 is more effective than a_2, neutralizing it even without full recognition ($\theta \neq 1$) or maximum investment in d_2 ($\neq 1$). When $\omega_{d_2} = 1$, full recognition ($\theta = 1$) and maximal allocation to d_2 ($= 1$) are required to entirely stop a_2. Finally, when $\omega_{d_2} < 1$, even with full resource allocation and recognition, it may not be feasible to halt the attack spread. In the original setting, $\omega_{d_2} = 0.9$, suggesting that a_2 is more effective than d_2 and θ_1.

We consider scenarios where d_2 and θ_1 have the same effect than a_2 ($\omega_{d_2} = 1$), as well as scenarios where d_2 and θ_1 are more ($\omega_{d_2} \in \{1.3, 1.7\}$) or less ($\omega_{d_2} \in \{0.4, 0.7\}$) influential. Figure 3.11 displays the mean of $d_2^*(d_1, a_2, \theta_1)$ over the values of d_1 and $E[\theta_2 \mid d_2^*(d_1, a_2, \theta_1), a_2, \theta_1]$ across such ω_{d_2} values.

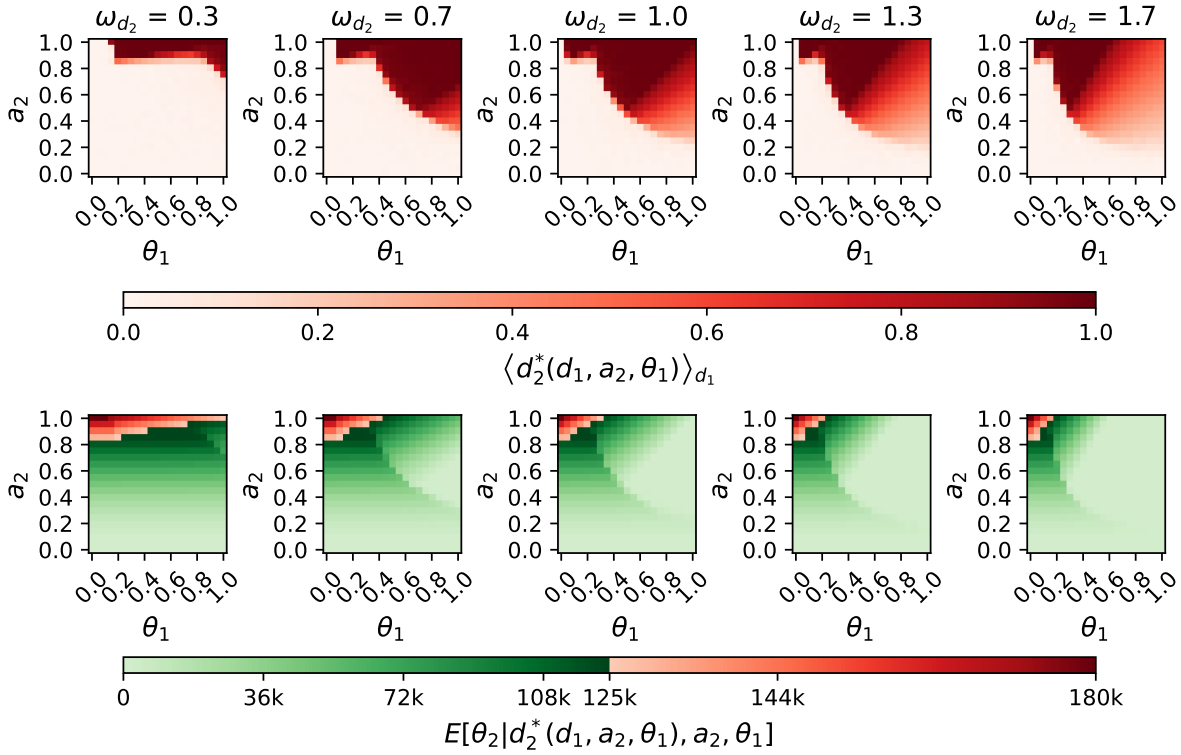


Figure 3.11: (Top) Mean $\langle d_2^*(d_1, a_2, \theta_1)\rangle_{d_1}$ of $d_2^*(d_1, a_2, \theta_1)$ over d_1 for different ω_{d_2}. (Bottom) Expected number $E[\theta_2 \mid d_2^*(d_1, a_2, \theta_1), a_2, \theta_1]$ of individuals to be infected for different ω_{d_2}.

Observe the differences across different values of ω_{d_2}. In the top row, as ω_{d_2} increases, there are more (θ_1, a_2) values where $d_2^*(d_1, a_2, \theta_1) \neq 0$ (non-white regions), meaning that D should implement reactive measures to counteract a_2 in more instances. As d_2 becomes more effective,

D is more inclined to invest, being more likely to successfully stop the disinformation spread. Conversely, when ω_{d_2} is lower, $d_2^*(d_1, a_2, \theta_1) \approx 1$ (dark red) is applied for higher values of θ_1, as maximum reactive measures are used primarily in cases where the consequences of a_2 may be severe, despite d_2 being less effective. Additionally, the regions where $0 < d_2^*(d_1, a_2, \theta_1) < 1$ (pale to medium red region) expand as ω_{d_2} increases, since greater effectiveness allows D to better calibrate its d_2 investment, avoiding both over- and underspending. In the bottom row, observe that, as ω_{d_2} increases, the number of cases requiring patient transfer to the private system (red region) decreases. Moreover, the expected number of infected people for each (θ_1, a_2) also decreases, given that light green regions expand. This occurs because, as ω_{d_2} increases, d_2 becomes more effective, leading to better outcomes for a same investment level.

Effect of t_d/t_a on the Attacker's second stage decision

This ratio determines the effectiveness of d_1 compared to a_1 on detecting a_2. For fixed d_1, a_1, and a_2, if $t_d > t_a$, the detection degree θ_1 on average will be higher than if $t_d = t_a$ or $t_d < t_a$, since the investment in d_1 is more efficient than the investment in a_1. Similarly, if $t_d = t_a$, the expected value of θ_1 ($E[\theta_1|d_1, a_1, a_2]$) will be higher than when $t_a > t_d$. In the original setting, $t_a = 1.2$ and $t_d = 1$, suggesting that A's investment is more effective than D's.

Let us test scenarios where the investments are equally effective ($t_d = t_a = 1$) and D's investment is more effective ($t_d = 1.2$, $t_a = 1$). Figure 3.12 displays estimates of $\overline{A}_2^*(d_1, a_1)$ for the different (t_d, t_a) pairs. Although the three plots display similar patterns, we observe some differences between them. First, when t_a/t_d decreases, the region $\overline{A}_2^*(d_1, a_1) \approx 0$ (white) expands, suggesting that A perceives a lower probability of successful outcome, thereby choosing not to attack. Besides, the region where $0 < \overline{A}_2^*(d_1, a_1) < 1$ (moderate red) shrinks as t_a/t_d decreases: this occurs because, as A recognizes that the attack is less effective, the only way to compensate it is by launching a maximum attack. Additionally, $\overline{A}_2^*(d_1, a_1) \approx 1$ (dark red) for more points (d_1, a_1) as the ratio t_a/t_d increases: A is more likely to implement a higher-intensity attack when it becomes more effective, as his expected outcome is greater.

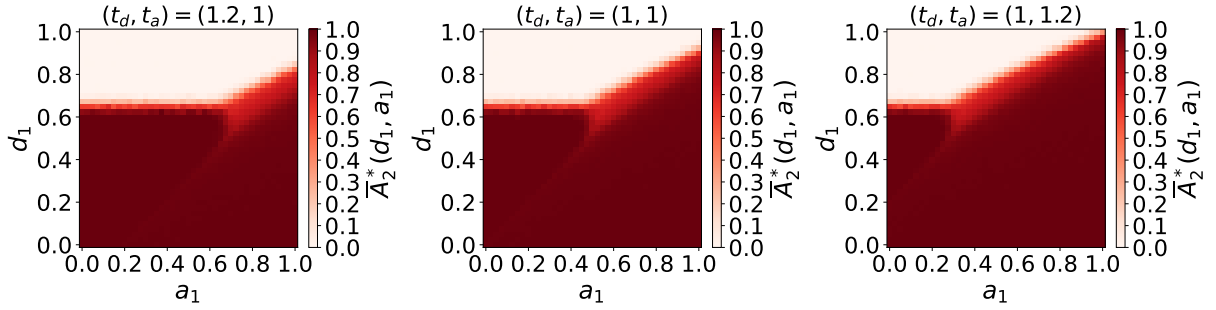


Figure 3.12: $\overline{A}_2^*(d_1, a_1)$ for several (t_d, t_a).

3.6 Discussion

This chapter has addressed objective **O2** by presenting an ARA methodology that applies APS to handle bi-agent general security games with incomplete information, contributing to the *targeted threats* and (potentially) to the *technical security controls* components of a CSRM scenario. The proposed approach offers a unified framework capable of solving any game modelled as a proper BAID, regardless of the number of decisions per agent or whether the decision domain is discrete

or continuous. In doing so, it provides the first comprehensive solution to the problem raised by Shachter (1986). We thoroughly discussed theoretical convergence, numerical modelling, and implementation issues of the methodology, and illustrated the framework with a disinformation warfare example. The methodology is particularly valuable for DHS scenarios, such as the increasingly important case of AML, where mitigating common-knowledge assumptions of game theory enables better decision-making.

However, while the methodology resolves general games across continuous domains, it may become computationally burdensome when applied in scenarios with a large number of decision stages per agent, as it will require approximating the (random) optimal value function of the Defender (Attacker) multiple times. This computational load could be partly reduced, as several stages of the algorithm are amenable to parallelization. In particular, for each value of the grid used in Sections 3.4.1, both the Defender's optimal values and Attacker's distribution can be approximated in parallel. Similarly, computing the MC approximation of $\widehat{\psi_D}$ ($\widehat{\Psi_A}$) for each value in the grid in each Defender's (Attacker's) stage can be performed in parallel. Additionally, there may be a trade-off between the explainability and performance of the models used to approximate agents' value functions. Some models, such as NN, may better estimate the value functions compared to less complex models like linear regression; however, a linear regressor allows for a better understanding of variable importance in decision support as it is interpretable, in contrast to a NN.

Chapter 4

Manipulating hidden Markov model inferences by corrupting batch data

4.1 Introduction

In accordance with objective **O3**, this chapter makes two primary contributions, advancing the *technical security controls* component of the CSRM scenario in Figure 1.1. First, we present a comprehensive range of HMM corruption problems that address filtering, smoothing, and decoding inferences under realistic uncertainty conditions leveraging an ARA perspective. Second, we design and benchmark a suite of attack algorithmic routines that can be tailored to specific scenarios. Through extensive empirical testing, we demonstrate the effectiveness of these attacks and highlight the trade-off between solution quality and computational effort when corrupting larger-scale HMMs, exploring the limits of the developed methods and providing insight into their capabilities.

Time-series models typically assume untainted and legitimate data streams. However, an adversary may have an incentive to corrupt the input data, thereby altering downstream inferences. Within the broader field of AML, this chapter provides a novel, probabilistic perspective on the manipulation of HMM inferences via corrupted batch data using an ARA approach, an unexplored topic in AML literature. Specifically, this chapter exposes the vulnerabilities of HMMs to adversarial attacks, emphasizing the importance of strengthening them to maintain security.

The research in this chapter is motivated by the utility and importance of HMM predictions and inferences, as demonstrated by their many applications across diverse disciplines. These include vehicle counting in computer vision (Miller et al., 2015), chromatin state learning in computational biology (Ernst and Kellis, 2012), stochastic thermodynamics in physics (Bechhoefer, 2015), signal processing in electrical engineering (Crouse et al., 1998), and speech recognition (Gales and Young, 2008). In addition, HMMs have been applied in security contexts such as network intrusion detection (Scott, 2002), spam detection (Gordillo and Conde, 2007), SMS spam misclassification (Xia and Chen, 2020), and crisis prediction in international politics (O'Brien, 2010).

The organization of this chapter is as follows. We begin in Section 4.2 by providing the requisite background on HMMs. Section 4.3 formally defines a collection of HMM corruption problems targeting filtering, smoothing, and decoding inferences. The resulting mathematical programs are stochastic, combinatorial and non linear. Section 4.4 sets forth three heuristic methods that enable the attacker to tractably identify high-quality solutions, with additional implementation details of the developed computational algorithms in Appendix C. These solution

techniques are tested in Section 4.5. The practical relevance of our methods is illustrated via a case study whereby an HMM used for part-of-speech tagging is attacked. Finally, we conclude the chapter with a brief discussion.

4.2 Background on HMMs

HMMs are stochastic processes over some time interval $\mathcal{T}$ which assume that an observed sequence of outputs, $\{x_t\}_{t\in\mathcal{T}}$, is a noisy observation from an underlying unobservable (hidden) sequence of states, $\{q_t\}_{t\in\mathcal{T}}$, that is, in turn, generated by a Markov chain, $\{Q_t\}_{t\in\mathcal{T}}$. In canonical HMMs, the state evolution model is a discrete-time, discrete-state, first-order Markov chain such that $\mathcal{T} = \{1, 2, \ldots, |\mathcal{T}|\}$ and $Q_t \in \mathcal{Q} = \{1, \ldots, |\mathcal{Q}|\}$. Transitions between Q_t and Q_{t+1} are governed by the state-transition probability matrix A having elements $a_{ij} = P(Q_{t+1} = j|Q_t = i), \forall (i, j) \in \mathcal{Q} \times \mathcal{Q}$, and initial-state probability vector π given by $\pi_i = P(Q_1 = i), \forall i \in \mathcal{Q}$. Furthermore, each observation $X_t \in \mathcal{X} = \{1, \ldots |\mathcal{X}|\}$ is determined by the observation probability matrix B[i] having elements $b_{ik} = P(X_t = k|Q_t = i),\ \forall (i, k) \in \mathcal{Q} \times \mathcal{X}$. Figure 4.1 depicts this standard model; the nodes and arcs represent random variables and conditional dependencies, respectively.

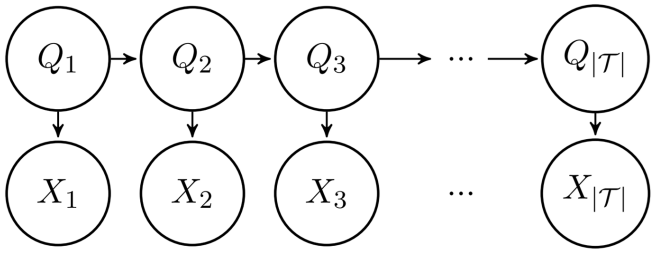


Figure 4.1: A basic hidden Markov model

Despite their relative simplicity, HMMs can be leveraged to perform a wide array of inferences and predictions. Rabiner (1989) discusses three commonly considered problems: (1) identifying the likelihood $P(\{X_t\}_{t\in\mathcal{T}})$ for a given HMM parameterization, (2) selecting the most likely sequence of latent states given an HMM model and a sequence of observations, i.e., decoding, and (3) learning to optimally parameterize, through maximum likelihood estimation, an HMM given a sequence of observations (e.g., estimating A and B). In addition to these standard problems, researchers and practitioners are also often interested in the probability of a specific hidden state at some time t. That is, given a sequence of observations, one may wish to infer $P(Q_{|\mathcal{T}|} = i|\{X_\tau\}_{\tau\in\mathcal{T}} = \{x_\tau\}_{\tau\in\mathcal{T}})$ and $P(Q_t = i|\{X_\tau\}_{\tau\in\mathcal{T}} = \{x_\tau\}_{\tau\in\mathcal{T}})$ for $t < |\mathcal{T}|$. These inferences are also known as filtering and smoothing, respectively. Moreover, whereas inference may be performed sequentially, HMMs often consider inference on batch data, i.e., using complete sequences of observations. We focus on filtering, smoothing, and decoding inferences using batch data.

Specifically, given a batch of observations $\{x_t\}_{t\in\mathcal{T}}$, the filtering and smoothing problems can be solved respectively using the Forward and Backward algorithms summarized as Algorithms 13 and 14.

[i] In this chapter, matrix B and its elements b_{ik} will also be referred to as the emission matrix and the emission probabilities, respectively.

Algorithm 13 Forward Algorithm

Input: A, B, π, $\{x_t\}_{t\in\mathcal{T}}$, time t

1: **for** $i \in \mathcal{Q}$ **do**
2: $\quad \alpha_{1,i} = \pi_i b_{i,x_1}$
3: **for** $\tau = 2, \ldots, t$ and $i \in \mathcal{Q}$ **do**
4: $\quad \alpha_{\tau,i} = \sum_{j\in\mathcal{Q}} \alpha_{\tau-1,j} a_{j,i} b_{i,x_\tau}$
5: **Return:** $\{\alpha_{\tau,i}\}_{\tau\in\{1,\ldots,t\},i\in\mathcal{Q}}$

Algorithm 14 Backward Algorithm

Input: A, B, π, $\{x_t\}_{t\in\mathcal{T}}$, time t

1: **for** $i \in \mathcal{Q}$ **do**
2: $\quad \beta_{|\mathcal{T}|,i} = 1$
3: **for** $\tau = |\mathcal{T}| - 1, \ldots, t$ and $i \in \mathcal{Q}$ **do**
4: $\quad \beta_{\tau,i} = \sum_{j\in\mathcal{Q}} \beta_{t+1,j} a_{i,j} b_{j,x_{\tau+1}}$
5: **Return:** $\{\beta_{\tau,i}\}_{\tau\in\{t,\ldots,|\mathcal{T}|\},i\in\mathcal{Q}}$

In the case of filtering, only Algorithm 13 must be utilized to identify the forward probabilities,

$$\alpha_{t,i} = P\left(Q_t = i, \{X_\tau\}_{\tau=1}^t = \{x_\tau\}_{\tau=1}^t\right).$$

Once identified, it can be shown that

$$P(Q_{|\mathcal{T}|} = i | \{X_\tau\}_{\tau\in\mathcal{T}} = \{x_\tau\}_{\tau\in\mathcal{T}}) = \frac{\alpha_{|\mathcal{T}|,i}}{\sum_{j\in\mathcal{Q}} \alpha_{|\mathcal{T}|,j}}.$$

Alternatively, for smoothing problems, the time of interest is some $t < |\mathcal{T}|$, implying that both the forward and backward algorithms must be leveraged. Once the backward probabilities,

$$\beta_{t,i} = P(\{X_\tau\}_{\tau=t+1}^{|\mathcal{T}|} = \{x_\tau\}_{\tau=t+1}^{|\mathcal{T}|} | Q_t = i).$$

are identified via Algorithm 14, it is known that

$$P\left(Q_t = i | \{X_\tau\}_{\tau\in\mathcal{T}} = \{x_\tau\}_{\tau\in\mathcal{T}}\right) = \frac{\alpha_{t,i}\beta_{t,i}}{\sum_{j\in\mathcal{Q}} \alpha_{t,j}\beta_{t,j}}. \tag{4.2.1}$$

Decoding problems, in turn, are typically addressed via the Viterbi algorithm summarized in Algorithm 15. Given some $\{x_t\}_{t\in\mathcal{T}}$, this algorithm functions by successively identifying the probability of the most likely latent-state path to $Q_t = i$, $\forall t \in \mathcal{T}, i \in \mathcal{Q}$. These are referred to as Viterbi probabilities and denoted by $\delta_{t,i}$. By storing the backtraces, i.e., $\psi_{t,i}$, the most likely sequences of states can be reconstructed by identifying $q^*_{|\mathcal{T}|} = \text{argmax}_{i\in\mathcal{Q}}\, \delta_{|\mathcal{T}|,i}$ and tracing the corresponding chain of $\psi_{t,i}$-values back to $t = 1$.

Algorithm 15 Viterbi Algorithm

Input: A, B, π, $\{x_t\}_{t\in\mathcal{T}}$

1: **for** $i \in \mathcal{Q}$ **do**
2: Set $\delta_{1,i} = \pi_i b_{i,x_1}$ and $\psi_{1,i} = 0$
3: **for** $t = 2, \ldots, |\mathcal{T}|$ and $i \in \mathcal{Q}$ **do**
4: $\delta_{t,i} = \max_{j\in\mathcal{Q}} \delta_{t-1,j} a_{j,i} b_{i,x_t}$
5: $\psi_{t,i} = \text{argmax}_{j\in\mathcal{Q}} \delta_{t-1,j} a_{j,i}$
6: Set $q^*_{|\mathcal{T}|} = \text{argmax}_{i\in\mathcal{Q}} \delta_{|\mathcal{T}|,i}$
7: Set $q^*_t = \psi_{t+1,q^*_{t+1}}$ for $t = |\mathcal{T}| - 1, ..., 1$
8: **Return:** $\{\delta_{t,i}\}_{t\in\mathcal{T},i\in\mathcal{Q}}$, $\{\psi_{t,i}\}_{t\in\mathcal{T},i\in\mathcal{Q}}$,$\{q^*_t\}_{t\in\mathcal{T}}$

4.3 Hidden Markov Model Corruption Problems

In this section, we introduce corruption problems aimed at altering the filtering, smoothing, and decoding inferences of HMMs from an ARA perspective. We consider herein an attacker, i.e., Z[ii], attempting to thwart inference conducted on a HMM by a decision maker, i.e., D. Given a full observation sequence $\{x_t\}_{t\in\mathcal{T}}$, we assume the attacker seeks to modify this information so that, when utilized by D, the resulting inference somehow benefits Z. Whereas the attacker knows D is utilizing an HMM, they are uncertain of its exact parameterization (i.e., in the terms of Biggio and Roli (2018), we consider a grey-box setting) and describe their beliefs probabilistically in a Bayesian manner. Moreover, the attacker's attempts at data corruption are subject to error and are not guaranteed to be successful. Corruption attacks are also accompanied by an associated risk of discovery, implying that Z must balance the risks and rewards of their attacks. For this initial research, D is assumed to utilize the standard HMM inference procedures in Section 4.2; that is, as with many HMM applications currently in use, D has not hardened their algorithms against potential attacks.

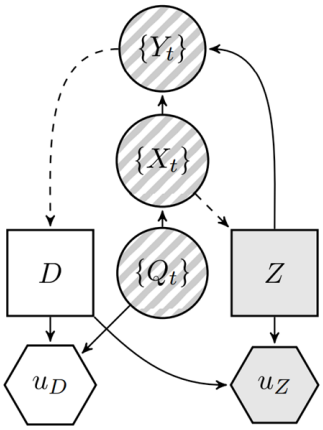
Figure 4.2 (a) provides a BAID which graphically depicts this interaction[iii]. Akin to the scenarios discussed by Krasser (2023), the attacker is assumed to have either infiltrated D's information system or controls the input data. They desire to use this access, along with intelligence about the form of $\mathcal{Q}$ and $\mathcal{X}$, to manipulate inference while maintaining data plausibility. The series of true latent states, $\{Q_t\}_{t\in\mathcal{T}}$, is unknown to each player; however, it affects the observations, $\{x_t\}_{t\in\mathcal{T}}$. Z is able to first view this information and, in accordance with their own self interest, attempt to corrupt it by altering observations. That is, if it will maximize their expected utility, Z would attempt to change a set of x_t-variables but, because these attacks are subject to error, the attack might not be entirely successful. Via Z's interaction with $\{x_t\}_{t\in\mathcal{T}}$, a new, potentially perturbed series of observations, $\{y_t\}_{t\in\mathcal{T}}$ is utilized by D as the basis of their inference.

Should Z have perfect knowledge about the HMM parameterization, then the resulting HMM corruption problem would be deterministic from the attacker's perspective. Herein, we adopt a more realistic alternative approach. Namely, an attacker can quite readily ascertain via nefarious means the class of ML algorithm utilized in application by D, but discovering its exact parameterization is a more difficult task. According to such uncertainties, Figure 4.2 (b) presents an ARA reduction of the BAID in Figure 4.2 (a) from the attacker's perspective. Z's beliefs

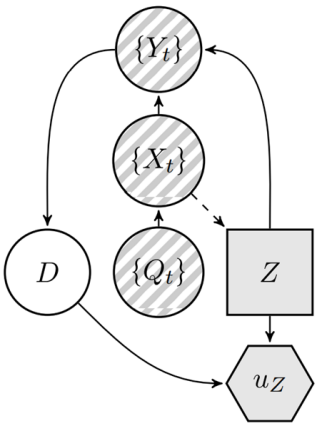
[ii] In this chapter, for notational tradition, we denote the attacker by Z and its attacks by z, while we preserve A and a_{ij} to represent the state–transition matrix and its elements, respectively.

[iii] In select problem variants (e.g., distribution disruption), an arc between $\{X_t\}$ and u_Z may need to be considered. We discuss such problems subsequently.

about the D's parameterization (i.e., $a_{i,j}$, $b_{i,k}$, and π_i) are encapsulated by the uncertainty node on D's decision.



(a) BAID for the HMM Corruption Problem



(b) Attacker Reduction of the HMM Corruption Problem

Figure 4.2: HMM Corruption Problem.

Although Figure 4.2 (b) provides an intuitive graphical means of depicting the attacker's problem, further notation is required to fix its formulation. Table 4.1 details the additional notation leveraged within this section to do so. Therein, variable indices are represented at their finest level of granularity; however, hereafter, arrays of decision variables are denoted via the absence of a subscript (e.g., z_t contains $z_{t,k}$ over every $k \in \mathcal{X}$). The same convention is used for the probability matrices, e.g., row i of A is denoted as $a_i = (a_{i,1}, \ldots, a_{i,|\mathcal{Q}|})$.

Table 4.1: Summary of Additional Notation for HMM Corruption Problems

Notation	Definition
$u_Z(\cdot)$	Z's utility for a given set of decision variables and realized uncertainties
$z_{t,k}$	Binary decision variable equal to 1 if Z inserts k at t, and 0 otherwise
w_1	Z's objective-function weight on D's inference
w_2	Z's objective-function weight on data corruption costs
$\rho_{t,k}$	Random variable equaling 1 if insertion of k at t is successful, and 0 otherwise
y_t	Perturbed observation at t viewed by D
$\alpha_{t,i}$	Standard forward probabilities calculated with $\{y_t\}_{t\in\mathcal{T}}$
$\beta_{t,i}$	Standard backward probabilities calculated with $\{y_t\}_{t\in\mathcal{T}}$
$\delta_{t,i}$	Standard Viterbi probabilities calculated with $\{y_t\}_{t\in\mathcal{T}}$
P_ρ	Joint probability mass function of ρ-variables having support $\mathcal{P}$
g_A	Joint density over entries in A having support $\mathcal{A}$
g_B	Joint density over entries in B having support $\mathcal{B}$
g_π	Dirichlet density over entries in π having support Π
g_ω	Joint density over all random variables having support Ω

4.3.1 Corrupting Filtering and Smoothing Inference

The attacker's problem described previously is simultaneously stochastic, combinatorial, and nonlinear, as reflected in Problem **P1** which represents an attacker with general, multi-objective utility u_z over the corruption of D's filtering or smoothing distributions at time t'. When encountered with such a problem, the attacker should solve

$$
\begin{aligned}
\mathbf{P1}: \max_{z} \quad & u_Z(z, \alpha_{t'}, \beta_{t'}) = \mathbb{E}\left[w_1 f_1(\alpha_{t'}, \beta_{t'}) - w_2 f_2(z)\right] \\
\text{s.t.} \quad & \sum_{k \in \mathcal{X}} z_{t,k} = 1, \ \forall t \in \mathcal{T}, && \text{(4.3.1a)} \\
& \alpha_{1,i} = \pi_i \left(\sum_{k \in \mathcal{X}} z_{1,k}(b_{i,k}\rho_{1,k} + b_{i,x_1}(1-\rho_{1,k})) \right), \ \forall i \in \mathcal{Q}, && \text{(4.3.1b)} \\
& \alpha_{t,i} = \sum_{j \in \mathcal{Q}} \alpha_{t-1,j} a_{j,i} \left(\sum_{k \in \mathcal{X}} z_{t,k}(b_{i,k}\rho_{t,k} + b_{i,x_t}(1-\rho_{t,k})) \right), \ \forall t \in \mathcal{T}\backslash\{1\}, \ i \in \mathcal{Q}, && \text{(4.3.1c)} \\
& \beta_{|\mathcal{T}|,i} = 1, \ \forall i \in \mathcal{Q}, && \text{(4.3.1d)} \\
& \beta_{t-1,i} = \sum_{j \in \mathcal{Q}} \beta_{t,j} a_{i,j} \left(\sum_{k \in \mathcal{X}} z_{t,k}(b_{j,k}\rho_{t,k} + b_{j,x_t}(1-\rho_{t,k})) \right), \ \forall t \in \mathcal{T}\backslash\{1\}, \ i \in \mathcal{Q}, && \text{(4.3.1e)}
\end{aligned}
$$

with

$$
\begin{aligned}
\rho_{t,k} &\sim Ber(\lambda_k), \forall t \in \mathcal{T}, \ k \in \mathcal{X}, \\
a_i &\sim Dir(\xi_i), \forall i \in \mathcal{Q}, \\
b_i &\sim Dir(\zeta_i), \forall i \in \mathcal{Q}, \\
\pi &\sim Dir(\upsilon),
\end{aligned}
$$

where Ber and Dir are shorthand for the Bernoulli and Dirichlet distributions characterized by the provided parameters. We note that attack success, i.e., $\rho_{t,k}$, is stationary across t but may vary in k. Similarly, uncertainty over the transition and emission probabilities may vary in i. For notational convenience, the joint probability mass function of all ρ-variables is denoted by P_ρ over support $\mathcal{P}$, the joint density of the transition matrix rows is g_A with support $\mathcal{A}$, the joint density of the emission matrix rows is g_B with support $\mathcal{B}$, and the Dirichlet distribution over π is denoted by g_π with support Π. The joint density of all random variables is denoted by g_ω with support $\Omega = \mathcal{P} \times \mathcal{A} \times \mathcal{B} \times \Pi$.

The objective function of Problem **P1** is the expected value of a linear combination of $f_1(\alpha_{t'}, \beta_{t'})$ and $f_2(z)$ whereby the former captures the attacker's utility from D's inference and the latter represents the utility associated with their corruption decisions. That is, a weighted-sum method from multicriteria optimization is leveraged (Ehrgott, 2005). Constraint (4.3.1a) ensures that the attacker only selects a single $k \in \mathcal{X}$ to insert into the perturbed observation vector at each time. Note that, if $x_t = k$ and $z_{t,k} = 1$, the attacker is not altering x_t and $y_t = k$ with certainty, i.e., this observation is not being changed. Alternatively, Constraints (4.3.1b)–(4.3.1e) ensure the correct calculation of both the forward and backward probability recursions. Although visually distinct, these constraints are conceptually similar to the equations provided in Section 4.2. The summand in (4.3.1b)–(4.3.1e) over $k \in \mathcal{X}$, ensures that the forward (backward) probabilities are calculated correctly with the realized $\{y_t\}_{t \in \mathcal{T}}$. Moreover, provided a realized set of attack-success outcomes (i.e., $\rho_{t,k}$), this sequence of perturbed observations can readily be calculated via

$$y_t = \sum_{k' \in \mathcal{X}} z_{t,k'} \left(\rho_{t,k'} k' + (1 - \rho_{t,k'}) x_t \right), \ \forall t \in \mathcal{T}.$$

From inspecting Problem **P1**, its difficulty becomes immediately apparent. By expanding the recursions in (4.3.1c) and (4.3.1e), it can be observed that the resulting functions are nonlinear for any non-trivial $\mathcal{T}$; both the forward and backward probabilities expand to, potentially, high-order polynomial functions. Adding further complexity is the fact that all of the parameters in (4.3.1b)–(4.3.1e) are unknown and subject to probabilistic uncertainty. Furthermore, whereas (4.3.1a) is deterministic and linear, it highlights the combinatorial nature of the attacker's problem. This complexity may be further exacerbated by the specific functional form of the attacker's multi-objective utility function.

Herein, we explore a subset of the most interesting utility functions. Numerous options exist to parameterize the component utility function $f_2(z)$. One straightforward and flexible option is

$$f_2(z) = \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{X} : x_t \neq k} z_{t,k}. \tag{4.3.2}$$

This simple alternative penalizes any corruption conducted by the attacker, and ensures this risk is appropriately represented in the attacker's utility function by its product with w_2. Higher-order polynomials in the $z_{i,k}$-variables could also be leveraged if the Z's attack costs are not constant. A similar diversity of functional representation is associated with $f_1(\alpha_{t'}, \beta_{t'})$. Subsequent sections explore this dynamic in greater detail whereby we provide alternative structures according to varying attacker intentions.

Given that Problem **P1** contains unknown parameters described probabilistically, it is clearly a stochastic programming problem. The recursions within (4.3.1b) – (4.3.1e) are defined to ensure clarity of communication and highlight the problem's relationship to canonical HMM algorithms but are not strictly necessary. That is, the recursion may be written explicitly in the objective function.

As mentioned before, the form of $f_1(\alpha_{t'}, \beta_{t'})$ will define the inference that the attacker intends to manipulate. In the following subsections, we present some cases that, in our opinion, are relevant.

State-Attraction and State-Repulsion Problems

In some settings, the attacker may only be interested in D's conditional distribution over a latent state at a particular time t'. For example, if the latent state represented the attacker's position and they wish to evade detection by D, it is in the attacker's best interest to minimize D's belief about this true position. Similarly, it is conceivable that an attacker may wish to lure D into believing a specific latent state i' occurred at time t'. In either setting, these objectives can be modeled using

$$f_1(\alpha_{t'}, \beta_{t'}) = c \left(\frac{\alpha_{t',i'} \beta_{t',i'}}{\sum_{j \in \mathcal{Q}} \alpha_{t',j} \beta_{t',j}} \right)$$

where $c = 1$ in a state-attraction problem and $c = -1$ in a state-repulsion problem. Therefore, the attacker will maximize or minimize, respectively, the smoothing probability provided in Equation (4.2.1).

This component utility function is associated with an attacker who is concerned with a single latent state. While useful, this may not always properly characterize the attacker's intent. Some

attackers may wish to alter D's inference over multiple latent states, e.g., potentially the entirety of $\mathcal{Q}$ as presented subsequently.

Distribution-Disruption Problems

The attacker may be interested in disrupting D's beliefs at time t' across $\mathcal{Q}$ to the maximum extent possible. More specifically, the attacker may wish to maximize the distance between the filtering (or smoothing) distributions under $\{x_t\}_{t\in\mathcal{T}}$ and $\{y_t\}_{t\in\mathcal{T}}$. Numerous options exist to characterize the distance between the uncorrupted and corrupted distributions, e.g., the Kullback-Leibler divergence and the Hellinger distance (Cha, 2007). Generally speaking, a distribution-disruption problem can be formulated by setting $f_1(\cdot)$ equal to the desired distance measure.

Let $\hat{\gamma}_{t,i}$ represent probability of state i at some time t given the uncorrupted data, $\{x_t\}_{t\in\mathcal{T}}$. Using this notation, the Kullback-Leibler divergence between the distributions induced by $\{x_t\}_{t\in\mathcal{T}}$ and $\{y_t\}_{t\in\mathcal{T}}$ is

$$f_1(\alpha_{t'},\beta_{t'}) = \sum_{i\in\mathcal{Q}} \hat{\gamma}_{t',i}\left(\log(\hat{\gamma}_{t',i}) - \log\left(\frac{\alpha_{t',i}\beta_{t',i}}{\sum_{j\in\mathcal{Q}}\alpha_{t',j}\beta_{t',j}}\right)\right).$$

The above can be used to identify a maximally perturbed distribution akin to an ill-performing M-projection, that is, the distribution that minimizes the Kullback–Leibler divergence between the distributions induced by $\{x_t\}_{t\in\mathcal{T}}$ and $\{y_t\}_{t\in\mathcal{T}}$ (Koller and Friedman, 2009). However, the order of the distributions in the Kullback–Leibler divergence may be reversed to identify a modified I-projection as well, that is, the distribution that minimizes the Kullback–Leibler divergence between the distributions induced by $\{y_t\}_{t\in\mathcal{T}}$ and $\{x_t\}_{t\in\mathcal{T}}$. Alternatively, if the Hellinger distance is utilized, then one may set

$$f_1(\alpha_{t'},\beta_{t'}) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i\in\mathcal{Q}}\left((\hat{\gamma}_{t',i})^{1/2} - \left(\frac{\alpha_{t',i}\beta_{t',i}}{\sum_{j\in\mathcal{Q}}\alpha_{t',j}\beta_{t',j}}\right)^{1/2}\right)^2}.$$

Other statistical distances calculated via an optimization step may also be used (e.g., the Wasserstein distance or the Kolmogorov–Smirnov statistic); however, when adopting such an approach, Problem **P1** may need to be augmented with distance-specific constraints. Henceforth, we utilize the Kullback-Leibler divergence in our exploration, but such a selection is merely a matter of attacker preference.

4.3.2 Corrupting Decoding Predictions

The probabilistic dynamics of the underlying HMM suggests that, when corrupting decoding predictions, the attacker is once more confronted with a non-linear, combinatorial, stochastic problem. Problem **P2** presents a mathematical programming model for the attacker trying to disrupt D inferring the most likely sequence of latent states given $\{y_t\}_{t\in\mathcal{T}}$.

$$\mathbf{P2}: \max_{z} \quad u_Z(z,\delta) = \mathbb{E}\left[w_1 f_1(\delta) - w_2 f_2(z)\right] \tag{4.3.3a}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{X}} z_{t,k} = 1,\ \forall t \in \mathcal{T}, \tag{4.3.3b}$$

$$\delta_{1,i} = \pi_i \left(\sum_{k\in\mathcal{X}} z_{1,k}(b_{i,k}\rho_{1,k} + b_{i,x_1}(1-\rho_{1,k})) \right),\ \forall i \in \mathcal{Q}, \tag{4.3.3c}$$

$$\delta_{t,i} = \max_{j\in\mathcal{Q}} \delta_{t-1,j} a_{j,i} \left(\sum_{k\in\mathcal{X}} z_{t,k}(b_{i,k}\rho_{t,k} + b_{i,x_t}(1-\rho_{t,k})) \right),\ \forall t \in \mathcal{T}\backslash\{1\},\ i \in \mathcal{Q}. \tag{4.3.3d}$$

where

$$\begin{aligned} \rho_{t,k} &\sim Ber(\lambda_k), \forall t \in \mathcal{T},\ k \in \mathcal{X}, \\ a_i &\sim Dir(\xi_i), \forall i \in \mathcal{Q}, \\ b_i &\sim Dir(\zeta_i), \forall i \in \mathcal{Q}, \\ \pi &\sim Dir(\upsilon). \end{aligned}$$

As with Problem **P1**, Problem **P2**'s objective function is the expected value of a linear combination of two component utility functions. The cost of an attack vector, $f_2(z)$, may be of a similar form as those discussed for Problem **P1**. However, since D is assumed to be solving a decoding problem, the attacker's component utility on their inference, $f_1(\delta)$, differs from Problem **P1**. The storage of forward and backward probabilities is no longer strictly required, and the Viterbi probabilities, $\delta_{t,i}$, are computed instead. However, in doing so, the constraints of Problem **P2** are further complicated via the inclusion of a maximum operator, i.e., Constraint (4.3.3d).

Given a realized set of uncertainties (i.e., ρ, A, B, π), the most-likely state sequence from D's perspective can be calculated via backtracking. However, for notational simplicity, the requisite back pointers that store the states maximizing Constraint (4.3.3d) are excluded from Problem **P2**; they are not necessary to identify the attacker's optimal solution for the following problems.

Path-Attraction and Path-Repulsion Problems

As with Problem **P1**, there exist multiple component utility functions that may be used for $f_1(\delta)$. However, we are primarily concerned herein with path-attraction and path-repulsion problems. In a path-attraction problem, the attacker wishes to encourage D to believe in a particular sequence of latent states. This is accomplished by setting

$$f_1(\delta) = \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{Q}} c_{t,i}\delta_{t,i}$$

such that $c_{t,i} > 0$ for states the attacker would like to encourage at each $t \in \mathcal{T}$. Alternatively, a path-repulsion problem is characterized by $c_{t,i} < 0$. The attacker may identify these sequences of latent states exogenously or base them off their expectation of $\{q_t\}_{t\in\mathcal{T}}$ under the true data $\{x_t\}_{t\in\mathcal{T}}$. Likewise, it is straightforward to combine these two frameworks to simultaneously attract and repulse a subset of latent state sequences. By their nature, the $\delta_{t,i}$-variables decrease in t, implying that, unless offset by increasing $c_{t,i}$-values, latent states visited later in the sequence would become relatively less important. If such an effect is not desirable, an effective method is to set $c_{t,i} = (\sum_{j\in\mathcal{Q}} \delta_{t,j})^{-1}$ for any state i the attacker wishes to encourage and set

$c_{t,i} = -(\sum_{j \in \mathcal{Q}} \delta_{t,j})^{-1}$ for any state i the attacker wishes to discourage. Notably, by working as such with the $\delta_{t,i}$-probabilities, the attacker does not necessarily induce the Viterbi algorithm to output the desired path; however, it is an approximation that is clearly associated with such induction.

4.4 Solution Methodologies

The HMM structure utilized by D presents multiple difficulties. In the filtering, smoothing, and decoding settings, the attacker is presented with a non-linear objective function. Moreover, this difficulty is compounded by a set of decision variables of combinatorial cardinality, as well as a lack of knowledge that induces a stochastic programming problem. Of the commercial optimization solvers available, only global solvers are capable of accepting such problems as input; however, as shown by Caballero et al. (2018), even these methods are not foolproof in complex situations such as the one described in this paper. Finally, as a stochastic programming problem, solving **P1** and **P2** coincides with maximizing an expectation, but its non-linearity limits the usefulness of canonical methods (e.g., sample-average approximation).

It is therefore apparent that customized solution techniques are required for **P1** and **P2**. Given the success of standard HMM algorithms, their modification to our perturbed setting would be ideal. Unfortunately, the relative efficiencies of the forward, backward, and Viterbi algorithms draw from their dynamic-programming structure, and this structure is dependent upon having observed a sequence of observations, e.g., $\{x_t\}_{t \in \mathcal{T}}$, upon which to base calculations. While desirable, it is unclear how (or if), the dynamic-programming structure of the aforementioned algorithms can be adapted to simultaneously optimize the attacker's actions and calculate the HMM's joint probabilities. The tractable identification of an optimal attack vector is, at this juncture, a dubious prospect. Therefore, we set forth alternative means the attacker may leverage to maximize their expected utility based on their subjective beliefs.

Within this section, we provide customized algorithms for solving variants of **P1** and **P2**, with minor modifications. The first solution method is a heuristic inspired by the *ranking-and-selection* problem (Powell, 2019). The second is based on the APS technique used in Chapter 3 to solve general security games. The third is an MC enumeration technique having complete and random-greedy variants. When allocated enough computational resources, the complete variant will converge to the optimal solution of **P1** or **P2**. Based on this property, this algorithm serves as the benchmark solution method. Notably, these attacks may be used for each of the problems from Section 4.3 with minor modifications. Likewise, they are applicable to alternative prior distributions to those assumed in the previous section; an analytic prior is not even necessary, as long as the attacker can sample from the associated distribution.

4.4.1 Ranking-and-Selection Heuristic

Once the attacker has probabilistically codified their beliefs about D's problem, **P1** and **P2** can be recast via the universal canonical model set forth by Powell (2019). More specifically, the HMM corruption problem is conceptualized as a variant of the ranking-and-selection model whereby the attacker is asked to estimate the optimal $z^* \in \mathcal{Z} = \{z : z_{t,j} \in \{0,1\}, \forall (t,j) \in \mathcal{T} \times \mathcal{X}; \sum_{j \in \mathcal{Q}} z_{t,j} = 1, \forall t \in \mathcal{T}\}$ after having run N experiments according to some policy η. These experiments are simulations of potential realities based upon the attacker's beliefs about attack success and the HMM parameterization. Traditional ranking-and-selection (R&S) problems consider a finite action space of relatively small cardinality, thereby enabling estimates of this expectation to be stored tabularly after each experiment n. In juxtaposition, the action-space

cardinality may be exceedingly large in our HMM corruption problems. Therefore, in what follows, we utilize a function-approximation approach to estimate an action's expected value. This method is highly flexible, allowing for use of various combinations of function-approximation methods and combinatorial-optimization routines.

Powell's universal canonical model requires the following elements be defined. Sequences of experimental values and estimates are denoted by n superscripts to distinguish them from the temporal, HMM sequences denoted by t subscripts.

- *State Variables*: The system state prior to conducting experiment n is denoted θ^n, and determines the attacker's beliefs about each action's expected value. These are codified via a functional approximation $\hat{\mu}(z^n|\theta^n)$, i.e., θ^n are the parameters of the attacker's functional approximation prior to experiment n. The exact structure of θ^n depends upon the approximation utilized; however, under any setting, the set of all possible state variables is denoted by Θ.
- *Decision Variables*: For each experiment $n = 1, ..., N$, the attacker selects $z^n \in \mathcal{Z}$.
- *Exogenous Information*: Given some attack vector z^n, the attacker's payoff is determined by D's inference on $\{y_t\}_{t\in\mathcal{T}}$ which is in turn affected by the realized exogenous information, i.e., the random ρ-, A-, B-, and π-variables. An instance of this exogenous information during experiment n is referred to as $\omega^n \in \Omega$.
- *Objective Function*: For **P1** and **P2**, the attacker's objective is to maximize their expected reward by selecting an experimental policy η that respectively solves

$$\max_{\eta} \mathbb{E}\left[u(Z^{\eta}(\theta^{N+1}), \alpha_{t'}, \beta_{t'})\right]$$

  and

$$\max_{\eta} \mathbb{E}\left[u(Z^{\eta}(\theta^{N+1}), \delta)\right]$$

  where $Z^{\eta}(\theta^{N+1})$ represents the specific attack decision that the policy η prescribes when it uses $\hat{u}(z^{N+1}|\theta^{N+1})$.
- *System Model*: Given that the system state θ^n denotes the parameters underpinning $\hat{\mu}(z^n|\theta^n)$, the system model, $S^M(\theta^n, z^n, \omega^n)$, corresponds to the sequential update of θ^n after each experiment, i.e., $S^M : \Theta \times \mathcal{Z} \times \Omega \to \Theta$. The system model reflects the sequential optimization of the θ^n parameters to θ^{n+1} after each experiment n. Example system models θ^n-updates include stochastic gradient descent and recursive least squares, among others.

Given this formulation, Algorithm 16 provides an overarching R&S heuristic that is applicable to either problem under any of the attacker utility functions presented in Section 4.3. The attacker must input their beliefs about the associated uncertainities (i.e., g_ω), their initial functional-approximation parameterization (i.e., θ^1), and the number of experiments they wish to conduct (i.e., N). The R&S heuristic concludes by outputting a recommended attack, $\hat{z}^*$. Note that Step 4 may be omitted for most problems; it is only required in select settings, e.g., in distribution-disruption problems.

Algorithm 16 Ranking-and-Selection Heuristic

Input: g_ω, θ^1, N

1: **for** $n = 1, \ldots, N$ **do** ▷ May alternatively use clock-time limit
2: Select $z^n = Z^\eta(\theta^n)$
3: Sample $\omega^n \sim g_\omega$
4: Solve D's HMM problem with $\{x_t\}_{t\in\mathcal{T}}$ ▷ If required
5: Use ω^n and z^n to determine $\{y_t\}_{t\in\mathcal{T}}$
6: Solve D's HMM problem with $\{y_t\}_{t\in\mathcal{T}}$ and ω^n, and compute $u_Z(\cdot)$
7: Update $\theta^{n+1} = S^M(\theta^n, z^n, \omega^n)$
8: Select $\hat{z}^* = Z^\eta(\theta^{N+1})$
9: **return** $\hat{z}^*$

The flexibility of the formulation allows for the construction of several R&S-heuristic variants. The policy η may adopt multiple structural forms, see Powell (2019). Greedy or ε-greedy policies on $\hat{\mu}(z^n|\theta^n)$ are sensible alternatives prevalent in the R&S literature. However, since $|\mathcal{Z}|$ can inhibit the use of complete enumeration techniques, it may be necessary to identify the greedy policy via integer programming or analogous heuristic methods (e.g., genetic algorithms). Moreover, as with canonical reinforcement learning methods, the function approximation, $\hat{\mu}(z^n|\theta^n)$, may be based upon any statistical, regression model (e.g., regression trees, NN, etc.) with a correspondingly tailored system model. For example, if $\hat{\mu}(z^n|\theta^n)$ is a linear regression model then, akin to Powell (2007), it is sensible to utilize recursive least squares as a system model; however, if $\hat{\mu}(z^n|\theta^n)$ is a NN then system model updates via stochastic gradient descent are more applicable[iv]. Ultimately, the attacker is likely best served by tuning these qualitative hyperparameters via some sort of experimental design, e.g., see Jenkins et al. (2021).

4.4.2 APS

The problems described in Section 4.3 can each be recast as indentifying a solution to

$$\max_z \sum_\rho \int_{\mathcal{A}} \int_{\mathcal{B}} \int_{\Pi} u_Z(z,\phi) g_A(A) g_B(B) g_\pi(\pi) P_\rho(\rho)\, \mathrm{d}\pi\, \mathrm{d}B\, \mathrm{d}A \tag{4.4.1}$$

where $\phi = \Phi(z, A, B, \pi, \rho)$ is determined by z, A, B, π, and ρ, and represents the relevant set of parameters and decision variables for the associated problem (e.g., $\phi = (\alpha_{t'}, \beta_{t'})$ for Problem **P1**).

One possible means to solve this problem consists of leveraging simulation-based techniques, such as the APS method already used in Chapter 3. This requires defining an AD on the product space of attacks and uncertainties of the form

$$\breve{g}(z, A, B, \pi, \rho) \propto u_Z(z,\phi) g_A(A) g_B(B) g_\pi(\pi) P_\rho(\rho). \tag{4.4.2}$$

This distribution is well-defined provided that the utility is positive and bounded for all z and ϕ, which can be achieved by scaling the utility function appropriately.

It is straightforward to see that the global mode of the marginal of $\breve{g}(z, A, B, \pi, \rho)$ in z, coincides with the solution of Equation (4.4.1). Therefore, if we sample $(z, A, B, \pi, \rho) \sim$

[iv] In Section 4.5, we illustrate two R&S variants based on NN, that differ in identification of the greedy action, i.e. Monte Carlo tree search and simulated annealing.

$\breve{g}(z, A, B, \pi, \rho)$, the sample mode of z approximates the optimal solution. However, as acknowledged by Müller et al. (2004), identifying the mode of the z samples may be challenging, especially when z is high dimensional. This difficulty is most apparent when the marginal AD is characterized by flat regions around its global mode. Both of these conditions are likely to emerge in our setting, suggesting an alternative to standard APS methods may be required.

Following the same procedures as in Chapter 3, we enhance the efficiency of APS by defining an alternative distribution to facilitate mode identification. More specifically, an AD must be defined such that its marginal in z is proportional to $\left[\sum_{\rho}\int_{\mathcal{A}}\int_{\mathcal{B}}\int_{\Pi} u_Z(z,\phi)g_A(A)g_B(B)g_\pi(\pi)\right.$ $\left.P_\rho(\rho)\,\mathrm{d}\pi\,\mathrm{d}B\,\mathrm{d}A\right]^H$ where $H \geqslant 1$, $H \in \mathbb{N}$. Since the associated, marginal AD is more peaked, its mode is more easily identified. Therefore, we define an AD by creating H copies of our unknown parameters. This set of copies is denoted by $\{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}$ such that $\mathcal{H} = \{1, 2, ..., H\}$. The AD is given by

$$\breve{g}_H(z, \{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}) \propto \prod_{h\in\mathcal{H}} u_Z\left(z, \phi^h\right) g_A(A^h) g_B(B^h) g_\pi(\pi^h) P_\rho(\rho^h). \tag{4.4.3}$$

where $\phi^h = \Phi(z, A^h, B^h, \pi^h, \rho^h)$. It is straightforward to prove that the marginal of $\breve{g}_H(z, \{A^h, B^h,$ $\pi^h, \rho^h\}_{h\in\mathcal{H}})$ in z is proportional to $\left[\sum_{\rho}\int_{\mathcal{A}}\int_{\mathcal{B}}\int_{\Pi} u_Z(z,\phi)g_A(A)g_B(B)g_\pi(\pi)\ P_\rho(\rho)\,\mathrm{d}\pi\,\mathrm{d}B\,\mathrm{d}A\right]^H$.

Direct sampling from the distribution in (4.4.3) is not feasible, since the exact specification of $\breve{g}_H$ is costly. Instead, samples can be obtained using MCMC methods (Tierney, 1994). In particular, we design a Metropolis-within-Gibbs sampling approach for use herein.

Implementing the Gibbs step requires generating samples from the full conditionals $\breve{g}_H(z_t|z_{-t}\{A,$ $B, \pi, \rho\}_{h\in\mathcal{H}})$ where $z_{-t} = \{z_{t'}\}_{t'\in\mathcal{T}\setminus\{t\}}$, and to sample from the full conditionals for A^h, B^h, π^h and ρ^h for $h = 1, \ldots, H$. For communicative clarity, let us rewrite (4.4.3) as

$$\breve{g}_H(z, \{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}) \propto \exp\left\{ \sum_{h\in\mathcal{H}} \log[u_Z\left(z, \phi^h\right)] + \log[g_A(A^h)] + \log[g_B(B^h)] \right.$$
$$\left. + \log[g_\pi(\pi^h)] + \log[P_\rho(\rho^h)] \right\}.$$

In so doing, it can be readily observed that

$$\breve{g}_H(z_t|z_{-t}, \{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}) \propto \exp\left(\sum_{h\in\mathcal{H}} \log\left[u_Z\left(z_t \cup z_{-t}, \phi^h\right)\right]\right).$$

This is simply the *softmax* distribution over $\sum_{h\in\mathcal{H}} \log[u_Z\left(z_t \cup z_{-t}, \phi^h\right)]$ for every possible value of z_t. Therefore, given z_{-t} and $\{A^h, B^h, \pi^h, \rho^h\}_{h\in\mathcal{H}}$, sampling from the full conditional of z_t is straightforward.

Conversely, to sample from the full conditionals for A^h, B^h, π^h and ρ^h for $h = 1, \ldots, H$, we need to utilize the Metropolis algorithm. For example, noticing that the full conditional for A^h is

$$\breve{g}_H\left(A^h|z, \{A^{h'}\}_{h'\in\mathcal{H}\setminus\{h\}}, \{B^{h'}, \pi^{h'}, \rho^{h'}\}_{h'\in\mathcal{H}}]\right) \propto \exp\left(\log[g_A(A^{h'})] + \log\left[u_Z(z, \phi^h)\right]\right),$$

samples of $A^{h'}$ can be obtained via a Metropolis approach as follows. Assume the current state of the Markov chain is $z, A^h, B^h, \pi^h, \rho^h$ and $\phi^h = \Phi\left(z, A^h, B^h, \pi^h, \rho^h\right)$, then

1. Sample $\tilde{A}^h \sim g_A$.
2. Compute $\tilde{\phi}^h = \Phi\left(z, \tilde{A}^h, B^h, \pi^h, \rho^h\right)$

3. Accept $\tilde{A}^h$ with probability

$$\min\left\{1, \frac{u_Z(z, \tilde{\phi^h})}{u_Z(z, \phi^h)}\right\}.$$

Samples from the conditionals in B^h, π^h, ρ^h can be obtained sequentially in a similar way. Moreover, samples from the full conditionals for A^h, B^h, π^h, ρ^h having different h-values can also be obtained in parallel. Sampling sequentially from these conditional distributions will asymptotically produce samples from $\breve{g}_H(z, \{A, B, \pi, \rho\}_{h\in\mathcal{H}})$.

However, as suggested by Müller et al. (2004) this sampling framework may result in the algorithm getting stuck in local modes, thereby inhibiting the identification of the global mode. This issue can be mitigated by combining this sampling scheme with an annealing schedule that iteratively increases H. This produces an inhomogeneous Markov chain whose limiting distribution is, under certain conditions, uniform over the set of optimal attacks. Algorithm 17 outlines our APS approximation method which utilizes the AD $\breve{g}_H$, the aforementioned Metropolis-within-Gibbs sampling approach, and an annealing schedule $\{\mathcal{H}_n\}_{n=1}^{\infty}$ of variable length; note that e_k denotes the standard basis vector in $\mathbb{R}^{|\mathcal{X}|}$ having a one in the position k and zero elsewhere. At each iteration n, a complete vector z^n is stored. The optimal attack vector $\hat{z}^*$ is estimated as the mode of these samples, excluding the burn-in period $n = 1, ..., n' - 1$. Since z is a discrete random variable, it is often useful to approximate this mode for large $\mathcal{T}$, e.g., by estimating the mode of each z_t independently, to avoid excessively large N-values.

Algorithm 17 APS Approximation

Input: g_A, g_B, g_π, P_ρ and $\{\mathcal{H}_n\}_{n=1}^{\infty}$, N

Initialize: z_t for $t = 1, \ldots, \mathcal{T}$
Set $n = 1$
while $n < N$ **do** ▷ May alternatively use clock-time limit
  Sample A^h, B^h, π^h, ρ^h using the Metropolis step $\forall h \in \mathcal{H}_n$
  for $t = 1, 2, \ldots, \mathcal{T}$ **do**
    For $h \in H_n$ and $k \in \mathcal{X}$, determine $\phi^h := \Phi\left(e_k \cup z_{-t}), A^h, B^h, \pi^h, \rho^h\right)$
    Sample $\tilde{z}_t$ from *softmax* distribution described in the Gibbs step
    Update $z_t = \tilde{z}_t$.
  Set $z^n = \{z_t\}_{t\in\mathcal{T}}$
  Set $n = n + 1$
Estimate $\hat{z}^*$ as the sample mode of $\{z^n : n \geqslant n'\}$
return $\hat{z}^*$

To guarantee convergence to the optimal attack(s), the inhomogeneous MCMC simulation needs to be designed so that the stationary distribution of the Markov chain for a fixed H is precisely $\breve{g}_H$. Following well-established literature of MCMC (Tierney, 1994), it is straightforward to see that the limiting distribution of the Metropolis-within-Gibbs Markov chain defined in Algorithm 17 for each $h \in H_n$ is precisely $\breve{g}_h$. For further information regarding APS, we refer the interested reader to Müller et al. (2004) for theoretical proofs of the algorithm's convergence, and to Bielza et al. (1999) and Ekin et al. (2023) for APS applications in decision-theoretic settings.

4.4.3 MC Enumeration

MC sampling techniques are popular solution approaches for standard ARA problems; they serve to numerically approximate otherwise intractable expectation functions. Akin to the proposed R&S heuristic, our complete MC enumeration (CME) method leverages samples from g_ω to inform better estimates of the utility of taking each $z \in \mathcal{Z}$; however, the sampling methods utilized are distinct. Both solution approaches randomly sample the attack effects and D's HMM parameterization (i.e., $\omega^n \sim g_\omega$), but they differ in their evaluation of corruption attacks. Whereas the R&S heuristic selects a single $z \in \mathcal{Z}$ to evaluate on ω^n according to η, the CME technique evaluates every $z \in \mathcal{Z}$ on each ω^n. In so doing, the latter method is able to garner more information from each ω^n, but at the expense of greater computational effort. Psuedocode for this solution method is provided in Algorithm 18. The algorithm is applicable to any of the problems presented in Section 4.3, but Step 3 is only strictly required in a distribution-disruption problem. This approach serves as a baseline for the other algorithms discussed herein.

Algorithm 18 Complete Monte Carlo Enumeration

Input: g_ω, N

1: **for** $n = 1, \ldots, N$ **do**
2: Sample $\omega^n \sim g_\omega$
3: Solve D's HMM problem with $\{x_t\}_{t\in\mathcal{T}}$ ▷ If required
4: **for** $z \in \mathcal{Z}$ **do**
5: Use ω^n and z to determine $\{y_t\}_{t\in\mathcal{T}}$
6: Solve D's HMM problem with $\{y_t\}_{t\in\mathcal{T}}$
7: Set $\hat{u}^{n,z} = u_Z^n(\cdot)$
8: Set $\bar{u}^z$ to the sample average of $\hat{u}^{n,z}$, $\forall z \in \mathcal{Z}$
9: Identify $\hat{z}^* = \text{argmax}_z\, \bar{u}^z$
10: **Return:** $\hat{z}^*$

To ensure termination on sufficiently large instances, Algorithm 18 can be modified to form a random-greedy variant. The random-greedy MC enumeration (RME) approach selects an incumbent attack at random and calculates its expected utility. The expected utility of another random attack is then calculated and, if improved, this new attack becomes the incumbent solution. The algorithm terminates when no improvement is identified. While CME may be effective for solving small-size instances, RME is more scalable to larger instances.

Finally, we note that the CME approach is a simpler alternative to the APS scheme. However, several issues are anticipated to arise for larger-sized instances. The most significant concern arises when the maximization of the expected utility, approximated through MC sampling, becomes challenging due to the flatness of the approximation with respect to z. In such cases, numerical errors inherent in the MC approximation of the expected utility may overshadow a relatively small difference in expected utility between the optimal and inferior attacks. In juxtaposition, the APS scheme transforms the simulation-optimization problem into a grand simulation problem using a series of augmented probability models that become more peaked around the optimal attack.

4.5 Testing, Results, and Analysis

This section analyzes the effects of HMM corruption on D's inference and examines the efficacy of the developed heuristics to conduct such attacks. This is accomplished via four blocks of

experiments. Our focus in Section 4.5.1 is to determine the degree to which data corruption can negatively affect a D's inference. Therein, we explore the effect of varying objective-function weights on the attacker's behavior and illustrate the devastating effects that relatively low-perturbation data-corruption attacks may generate. Analysis within this section also reveals that, although the MC enumeration algorithm can identify high-quality solutions, it is computationally infeasible for larger-sized HMMs. Therefore, in Sections 4.5.2 and 4.5.3 we provide detailed analyses of the use of our R&S heuristic and the APS approximation routine. Section 4.5.2 analyzes the effect of the HMM structure on each algorithm's performance, whereas Section 4.5.3 focuses on the impact of uncertainty on solution quality. Section 4.5.4 provides a case study illustrating the practical relevance of our attacks by attacking an HMM used for part-of-speech tagging.

While the case study leverages a standard laptop for computation to demonstrate real world application feasibility, the remainder of testing was performed using relatively powerful machines. This enabled parallelization and multiple runs of each attack for improved statistical analysis.

4.5.1 Efficacy of HMM Corruption

The purpose of this section is to illustrate the adverse effects of data corruption on HMM inference and showcase how this relates to the attacker's objectives and knowledge. More specifically, we demonstrate that, even under substantial uncertainty about D's HMM, the methods developed herein can devastate inference quality. Moreover, we also explore the effectiveness of the CME attack and its limitations with respect to instance size. We examine each problem-and-objective-function combination with this attack by utilizing a modestly sized HMM for tractable illustration. Whereas in-depth testing of the R&S and APS solution approaches are performed in subsequent sections, this section also demonstrates the marginal effect of allocating additional computational resources for each attack. In so doing, we highlight that our attacks can expeditiously identify quality solutions.

Assume that D's true HMM is defined by

$$A_D = \begin{bmatrix} 0.85 & 0.05 & 0.1 \\ 0.05 & 0.9 & 0.05 \\ 0.5 & 0.25 & 0.25 \end{bmatrix},$$

$$B_D = \begin{bmatrix} 0.699 & 0.05 & 0.1 & 0.05 & 0.1 & 0.001 \\ 0.001 & 0.1 & 0.1 & 0.299 & 0.3 & 0.2 \\ 0.1 & 0.2 & 0.1 & 0.2 & 0.1 & 0.3 \end{bmatrix},$$

$$\pi_D = \begin{bmatrix} 0.5 & 0.3 & 0.2 \end{bmatrix},$$

such that there are three possible latent states and six possible emissions at each state. This precise parameterization is unknown to Z; however, we assume that the attacker's beliefs are correct in expectation (i.e., $\mathbb{E}[A] = A_D$). The attacker is also assumed to believe that the probability of a successful attack on an observation is constant. In this manner, we define two distinct uncertainty levels upon which to test our attacks. The lower uncertainty condition is characterized by a Dirichlet precision (i.e., κ)[v] of 10,000 and a constant attack-success probability of 0.95. Conversely, the high uncertainty condition corresponds to a Dirichlet precision of 100 and an attack-success probability of 0.75. Assuming $\mathcal{T} = \{1, ..., 5\}$, Z observes the true emissions $X = \{5, 4, 6, 4, 5\}$ and wishes to thwart D's inference in accordance with their own self-interest.

[v] Recall that the precision of a Dirichlet distribution equals the sum of its parameters. These values are varied under the constraint that, in expectation, Z's beliefs coincide with D's true parameterization.

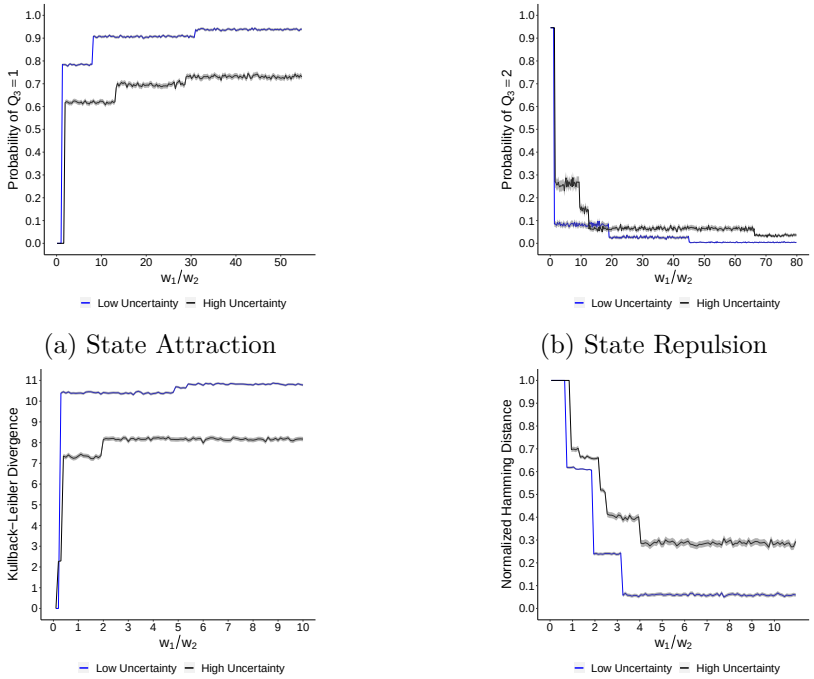
We examine Z's behavior in the state-attraction, state-repulsion[vi], distribution-disruption, and path-attraction problems under the two aforementioned uncertainty levels and varying ratios of w_1/w_2. For the state-attraction problem, Z desires D to believe $Q_3 = 1$, whereas in the state-repulsion problem, Z desires D not to believe $Q_3 = 2$. In the distribution-disruption problem, Z wants to maximally perturb D's smoothing distribution at $t' = 3$ while, in the path-attraction problem, Z wishes D to infer $Q = \{3, 1, 1, 1, 3\}$ as the most probable sequence of hidden states. The attacker aims to maximize their expected utility in these settings; however, to highlight the effect of HMM corruption for each problem, distinct performance measures are reported. More specifically, performance measures for the state-attraction, state-repulsion, distribution-disruption, and path-attraction problems are respectively set as D's subjective belief of $Q_3 = 1$, D's subjective belief of $Q_3 = 2$, the Kullback Leibler divergence between the true and corrupted smoothing distributions, and the normalized Hamming distance[vii].

We approximate the effect of $\hat{z}^*$ on these metrics for each problem-and-uncertainty combination as a function of w_1/w_2. An N-value of 10,000 was utilized to estimate $f_1(\cdot)$ for each attack alternative and problem type. These estimates were subsequently used to estimate the expected utility of every attack for each w_1/w_2. The attack alternatives that maximize the expected utility were selected as $\hat{z}^*$-variables. Once identified, the attack $\hat{z}^*$ was simulated M times to generate an approximate distribution over the performance metrics. The value of M was selected to ensure an appropriate level of precision for the mean performance in each problem. This required $M = 5,000$ simulations for the state-attraction and distribution-disruption problems, but $M = 1,000$ sufficed for the state-repulsion and path-attraction problems. Within each problem, the associated M-value is kept constant across uncertainty levels to ensure proper comparisons.

Figure 4.3 summarizes the results of this testing for each problem-and-uncertainty pair. Within each plot, the mean performance measures over the M samples are represented by a blue and black line for the low- and high-uncertainty levels, respectively; $\pm 2\,(s/\sqrt{M})$ confidence regions are shaded grey. It can be observed that, in any problem when $w_1/w_2 \to 0$, the attacker determines the benefits of corrupting the data are outweighed by its costs. The attacker chooses to keep $\{x_t\}_{t\in\mathcal{T}}$ unchanged and, in so doing, concedes undesirable inference behavior by D. For example, in the state-repulsion problem, when $w_1/w_2 \to 0$, the probability of $Q_3 = 2$ is approximately 0.95, i.e., the probability inferred by D using the true observations. However, as w_1/w_2 increases and relative corruption costs decrease, this balance shifts and Z begins to corrupt the data. Notably, across all plots, it is interesting to note that step-function-like behavior is induced. This behavior can also be observed by inspecting $\hat{z}^*$ at each w_1/w_2 for every problem. That is, $\hat{z}^*$ tends to stay constant across an interval of ratios until the reward of attacking some x_t is worth the penalty, at which point a new optimal attack is determined. The result of this behavior is that the expected value of the performance measures stays constant over the domain wherein $\hat{z}^*$ is constant.

[vi]The state-attraction and -repulsion problems are, structurally speaking, the same problem. However, we solve both herein to explore how varying objective-function parameterizations may affect the attacker's solution. For brevity, we exclude such exploration of the remaining problems.

[vii]The normalized Hamming distance between two strings with the same length is the number of positions at which the corresponding symbols are different, divided by the strings' length.



(a) State Attraction

(b) State Repulsion

(c) Distribution Disruption

(d) Path Attraction

Figure 4.3: Ratio Plots for each Problem-and-Uncertainty Pair

Inspection of Figure 4.3 reveals that distinct values of w_1/w_2 are necessary to induce modified attacker behavior. For example, in the state-attraction problems, Z is incentivized to maximally perturb the observation vector at $w_1/w_2 \approx 30$, but this behavior is not induced in the path-attraction problems until $w_1/w_2 > 3$. By inspecting each subfigure in Figure 4.3, once can discern how the uncertainty levels affect the attacker's behavior. Notably, within each problem, the high-uncertainty level alters the range of w_1/w_2-values in which a $\hat{z}^*$ is estimated to be optimal and, in expectation, tends to induce a less-preferable, maximally perturbed outcome. Likewise, tighter confidence regions about the attack's expected value can be derived under low uncertainty than high uncertainty.

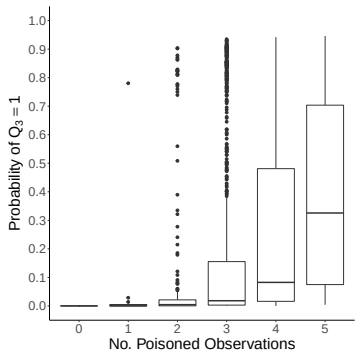
The plots in Figure 4.3 are also interesting in that, once w_1/w_2 is sufficiently far from zero, a dramatic increase in the performance measure of each problem is observed. The degree of improvement is problem dependent (e.g., compare the behavior of Subfigures 4.3c and 4.3d near $w_1/w_2 = 0$), but the pattern is consistent. Nevertheless, this behavior naturally leads one to inquire about the nature of the requisite attacks, i.e., the degree to which they perturb $\{x_t\}_{t\in\mathcal{T}}$. Fortunately, the size of the HMM examined in this section allows us to readily examine this in greater detail.

We perform additional analysis on the attacker's problems by enumerating the effects of every $z \in \mathcal{Z}$; M simulations of each attack z are performed to estimate its mean performance for the aforementioned problem types. The box plots in Figure 4.4 examine to what degree attacks of differing perturbation levels (i.e., the number of corrupted observations) affect D's inference. Examination of these plots yields noteworthy insights. The attacker could considerably disrupt D's inference by corrupting only a single observation in some cases, assuming they have the means to readily identify it. For example, in the case of low uncertainty, corrupting one observation could change D's subjective probability from ≈ 0 to 0.8 in the state-attraction

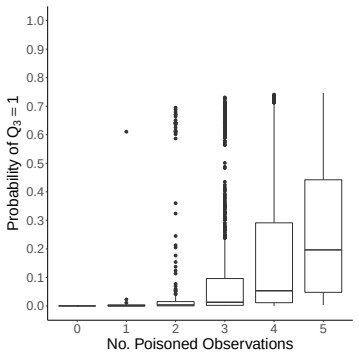
problem while a similar change from ≈ 0.95 to 0.1 is observed in the state-repulsion problem. Such modest attacks are not quite as successful in the distribution-disruption or path-attraction problems, but high-quality results can still be achieved without corrupting the entirety of $\{x_t\}_{t\in\mathcal{T}}$. Moreover, it can also be observed in the variability of each subfigure that high-perturbation attacks are not necessarily effective; in fact, the variability in an attack's success increases with the number of corrupted observations. Although these results collectively highlight the disruption that HMM corruption can induce on D's inference, they also emphasize the difficulty of the attacker's problems. This is most apparent by considering the medians and inter-quartile ranges in Figure 4.4. Effective attacks that limit the number of corrupted observation are the exception, not the rule. Conversely, blindly corrupting a large number of observations has no guarantee of success; such attacks have better median performance but are highly variable.

Finally, perhaps the most noteworthy result in this section pertains to the performance of the CME algorithm. Despite the difficulty of the examined problems, this solution methodology was readily able to identify high-quality solutions. Furthermore, it is also worth emphasizing that, whereas the attacks are built under both aleatoric and epistemic uncertainty (i.e., the unknown attack success and HMM parameters, respectively), the evaluations provided herein are performed under aleatoric uncertainty only. This fact further emphasizes the utility of our methodology. Despite not knowing the outcome of an attack or D's true HMM at design time, our attacks were able to substantially thwart D's inferences.

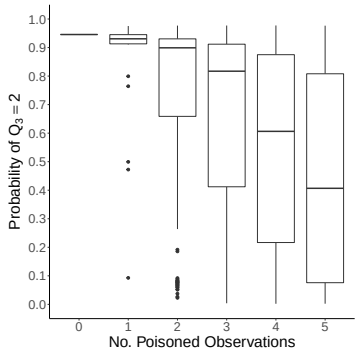
Such a result bodes well for the real-world applicability of such corruption attacks. The attacker is unlikely to know the true HMM, but the methods presented herein allow to identify high-quality attacks despite this limitation. Unfortunately, although the CME method is effective, additional experimentation found it too computationally burdensome for larger instances. Given some w_1/w_2, the problems examined within this section can generally be solved within 30-55 minutes using the aforementioned hardware but, for larger-sized HMMs having a $\mathcal{T}$ of greater cardinality, the required computational effort rapidly increases. This motivates the utilization of other methods, e.g., the R&S and APS methods discussed in the next two subsections.



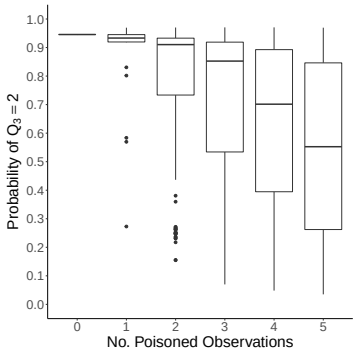
(a) State-Attraction, Low Uncertainty



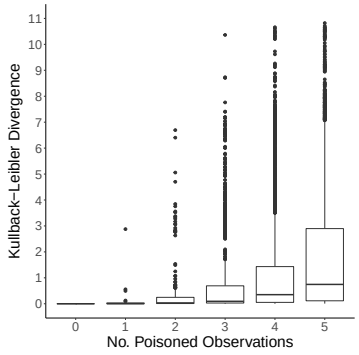
(b) State-Attraction, High Uncertainty



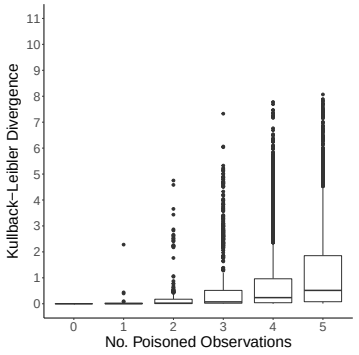
(c) State-Repulsion, Low Uncertainty



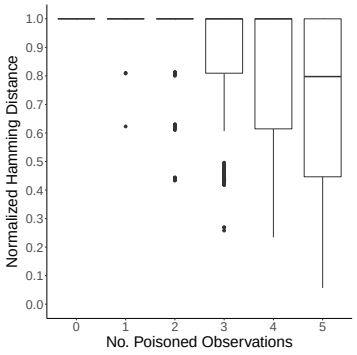
(d) State-Repulsion, High Uncertainty



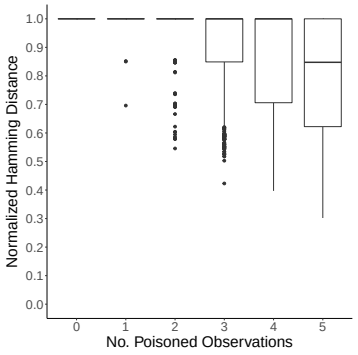
(e) Distribution-Disruption, Low Uncertainty



(f) Distribution-Disruption, High Uncertainty



(g) Path-Attraction, Low Uncertainty



(h) Path-Attraction, High Uncertainty

Figure 4.4: Mean Attack Efficacy by Number of Perturbed Observations

4.5.2 Effect of HMM Structure on Solution Method Performance

This section focuses on the effect of the HMM structure on the R&S and APS methods. Following the best practices set forth by Coffin and Saltzman (2000), we consider the 2^{3-1}_{III} design presented in Table 4.2 upon which to test algorithm performance. The distributions defining D's true HMM parameters (e.g., A_D) are generated randomly via a Dirichlet distribution with all concentration parameters set to one. The true observations are generated randomly from a discrete uniform distribution with support $\{1, 2, ...|X|\}$, whereas $\lambda = 0.95$ (attack-success probability) and $\kappa = 10,000$ (Dirichlet precision) for each. Table 4.3 depicts the attacker's goals for each problem, and the objective function weight ratios, w_1/w_2, are selected to correspond with a conservative attacker who is unwilling to corrupt all observations. The complete parameterizations are available online in the code repository for this chapter: https://github.com/roinaveiro/corrupting_hmms.

Table 4.2: 2^{3-1}_{III} Design For HMM Structure Testing

Design Point	$\|Q\|$	$\|X\|$	$\|T\|$
1	30	30	30
2	10	10	30
3	10	30	10
4	30	10	10

Table 4.3: Attacker's Objective Function Parameters by Problem and Design Point

Problem	w_1/w_2	Design Point			
		1	2	3	4
State-Attraction	20	$\{t' = 25, i' = 4\}$	$\{t' = 21, i' = 9\}$	$\{t' = 9, i' = 7\}$	$\{t' = 9, i' = 22\}$
State-Repulsion	15	$\{t' = 23, i' = 9\}$	$\{t' = 29, i' = 4\}$	$\{t' = 5, i' = 6\}$	$\{t' = 9, i' = 12\}$
Distribution-Disruption	3	$t' = 23$	$t' = 29$	$t' = 5$	$t' = 9$
Path-Attraction	2.55	$\{0\}_{\forall t \in \mathcal{T}}$			

Two variants of each of the R&S and APS methods are tested against the RME algorithms; the CME approach is ignored because empirical testing found it to be overly encumbered for instances of this size. The two R&S methods are henceforth referred to as R&S-A and R&S-B. Both utilize an MLP NN as $\hat{\mu}(\cdot)$ to regress the attack's expected utility, but differ in their use of Monte-Carlo tree search and simulated annealing to identify a greedy action, respectively. Further details on the implementation of these methods, as well the hyperparameter tuning that informed them, is provided in Appendix C. Likewise, the APS techniques are referenced as APS-A and APS-B; the approaches have annealing schedule of $\{\mathcal{H}_n\}^{\infty}_{n=1}$ and $\{\mathcal{H}_n\}^{\infty}_{n=500}$, respectively. These schedules result in APS-A exploring the solution space more thoroughly than APS-B which instead favors exploitation.

We explore the efficacy of these attacks both in terms of solution quality and computational effort. This is accomplished by allowing each algorithm to operate for $\{15, 30, 45, ..., 1200\}$ seconds on every problem instance; the expected utility of the output attack at each time limit is calculated. Due to the stochastic nature of the attacks and the solution approaches, this procedure is replicated 10 times for each algorithm; mean expected utilities across these repetitions plus/minus two standard deviations are reported.

Several trends emerged from this analysis across each problem type. This behavior is typified within the path-attraction instances as provided in Figure 4.5. Notably, APS-B and R&S-B converge toward attacks having comparable expected utilities across all instances and, although not a firm rule, APS-B tends to do so quicker than R&S-B. Alternatively, APS-A and R&S-A are more variable. For instances defined by smaller $|\mathcal{T}|$ (i.e., Design Points 3 and 4) they also converge toward high-quality solutions but, for larger $|\mathcal{T}|$-values, they do not identify valuable attacks. The computational time required to identify such solutions is variable as well. In juxtaposition, RME consistently identifies lower-quality attacks in every instance. Although not depicted herein, similar trends are also apparent across the other three problem types.

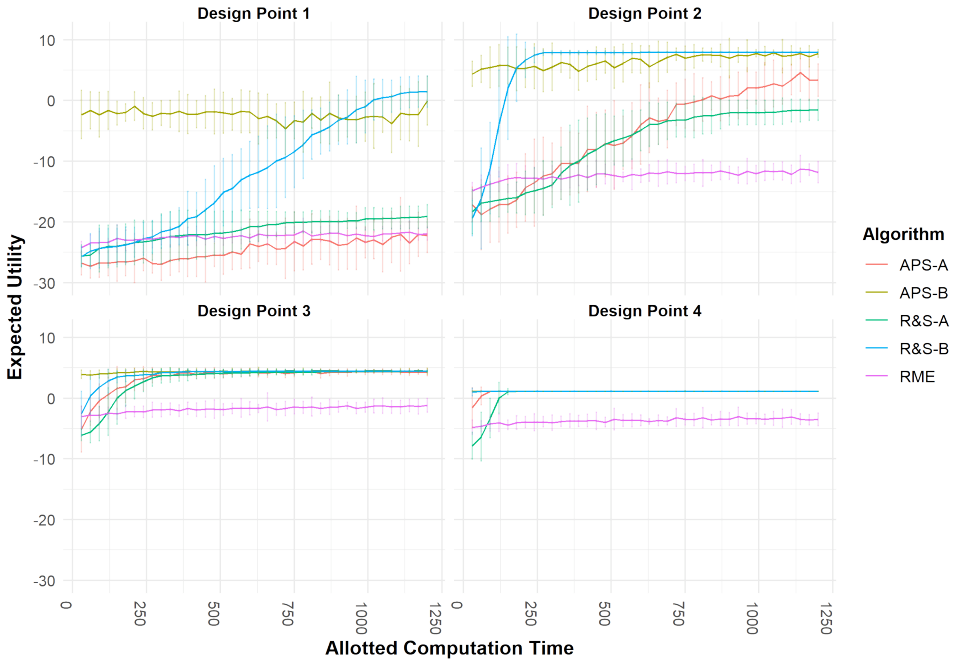


Figure 4.5: Evolution of Path-Attraction Expected Utilities over time by Algorithm

Despite their global similarities, it is interesting to note that the algorithms vary systematically in the type of attacks they identify. Table 4.4 compares and contrasts each algorithm with respect to the performance of their final attacks, i.e., those identified at 1200 seconds, across all problem-and-design-point pairs. Performance is based on the respective objective function of the problem of interest. To account for the stochasticity in attack success, solution quality was estimated by simulating their effects 500 times for the state-attraction and distribution-disruption instances, and 100 times for the state-repulsion and path-attraction instances; averages are reported across all repetitions. The impact measure is problem-specific and refers to damage caused to D's inference, whereas Δ refers to the total number of observations attacked. The greatest impact and the least $\Delta/|\mathcal{T}|$ achieved by the algorithms are bolded for each problem-and-design-point pair.

Table 4.4: Mean Impact and Proportion of Observations Attacked (Structure Testing)

Problem	Algorithm	Design Pt. 1		Design Pt. 2		Design Pt. 3		Design Pt. 4	
		Impact	$\Delta/\lvert\mathcal{T}\rvert$	Impact	$\Delta/\lvert\mathcal{T}\rvert$	Impact	$\Delta/\lvert\mathcal{T}\rvert$	Impact	$\Delta/\lvert\mathcal{T}\rvert$
State Att.[1]	APS-A	0.144	0.583	0.040	0.030	0.635	0.310	0.124	0.130
	APS-B	0.136	**0.043**	0.009	**0.007**	0.636	**0.300**	0.120	**0.120**
	R&S-A	0.169	0.773	0.038	0.517	**0.637**	0.420	**0.128**	0.280
	R&S-B	**0.179**	0.327	0.012	0.010	0.627	0.340	0.125	0.130
	RME	0.125	0.830	**0.074**	0.687	0.605	0.960	0.127	0.580
State Rep.[2]	APS-A	0.173	0.540	0.502	**0.033**	0.568	**0.100**	0.168	**0.100**
	APS-B	0.172	**0.037**	0.509	**0.033**	0.563	**0.100**	**0.170**	**0.100**
	R&S-A	0.174	0.740	0.500	0.490	0.566	**0.100**	0.166	0.180
	R&S-B	**0.177**	0.163	**0.515**	0.047	0.566	0.140	0.169	**0.100**
	RME	0.167	0.810	0.493	0.680	**0.577**	0.710	0.167	0.520
Dist. Disrupt.[3]	APS-A	2.221	0.653	2.202	**0.033**	**1.445**	0.210	**1.520**	**0.100**
	APS-B	2.140	**0.033**	2.213	**0.033**	1.435	**0.200**	1.518	**0.100**
	R&S-A	**2.283**	0.747	**2.297**	0.510	1.436	0.370	1.459	0.200
	R&S-B	2.206	0.453	2.279	0.327	1.426	0.290	1.511	**0.100**
	RME	2.203	0.877	2.192	0.687	1.270	0.830	1.517	0.550
Path Att.[4]	APS-A	0.980	0.817	**0.851**	0.190	**0.612**	0.340	**0.900**	**0.000**
	APS-B	0.940	0.080	0.860	**0.010**	0.707	**0.280**	**0.900**	**0.000**
	R&S-A	0.962	0.740	0.857	0.483	0.648	0.520	0.921	0.150
	R&S-B	**0.930**	**0.027**	0.867	0.017	0.620	0.360	**0.900**	**0.000**
	RME	0.972	0.813	0.897	0.690	0.718	0.700	0.923	0.460

[1] Impact is the difference between perturbed and unperturbed data of the probability of i' at t'
[2] Impact is the difference between unperturbed and perturbed data of the probability of i' at t'
[3] Impact is the KL divergence between state distributions under unperturbed and perturbed data
[4] Impact is the Normalized-Hamming distance of most-likely state sequences under perturbed data and attacker's goal

Inspection of this table reveals additional patterns obfuscated by the expected-utility calculations. APS-B and R&S-B are better able to balance the attacker's multi-objective utility function; however, in so doing, they often find less damaging attacks to D's inference than their competitors. In particular, of the 16 problem-and-design-point pairs, the APS-B and R&S-B algorithms identified the most impactful attack two and four times, respectively. The expected impact varied substantially about problem types as well. R&S-B never identified the most impactful distribution-disruption attack; APS-B never did so for the state-attraction problem. However, the APS-B attacks appear to perturb relatively few perturbations in comparison to their impact. For specific problem-and-design-point pairs, the other three methods identified more impactful attacks. This success, however, was generally counterbalanced by higher rates of data perturbation. The APS-A approach is an exception to this rule in that it at times acts too conservatively by perturbing relatively few observations.

Collectively, these results illustrate that the APS and R&S algorithms can effectively thwart larger-scale HMM instances. The APS-B and R&S-B configurations were the most effective for the instances explored herein, but their performance varied. This variation was especially apparent across $|\mathcal{T}|$-values, a fact deriving from the exponential effect $|\mathcal{T}|$ has on the attack space's cardinality. Nevertheless, many of the results discussed herein are specific to the parameterization utilized within this section. Alternative uncertainty structures about the D's HMM may also play a significant role in the algorithm's efficacy. Exploring such dynamics is the focus of the next subsection.

4.5.3 Effect of Uncertainty on Solution Method Performance

The same algorithm variants studied in the previous section are utilized herein to explore the effect of uncertainty on their performances. The HMM size is fixed at $|\mathcal{Q}|$, $|\mathcal{X}|$, and $|\mathcal{T}| = 20$; however, the attacker's beliefs and the attack's success probability is varied via the 2^2 factorial design provided in Table 4.5. The D's true HMM parameters are built in a similar manner to Section 4.5.2 sampling from Dirichlet distributions; the true observations are likewise built by sampling from discrete uniform distributions. The w_1/w_2-values for each problem are maintained constant from Section 4.5.2, as is the attacker's goal in the path-attraction problem. However, their goals in the state-attraction, state-repulsion and distribution-disruption instances correspond to $\{t' = 17, i' = 4\}$, $\{t' = 16, i' = 7\}$, and $t' = 16$, respectively. The complete parameterization of these instances is available online in the code repository for this chapter.

Table 4.5: 2^2 Full Factorial Design for Uncertainty Testing

Parameter	Design Points			
	1	2	3	4
λ	0.95	0.95	0.75	0.75
κ	10,000	100	10,000	100

A similar testing to the previous subsection was conducted on each algorithm for every problem-and-design-point pair. Figure 4.6 provides the algorithms' performance in relation to the allocated computation time for the distribution-disruption problem. These results bear resemblance to those of the previous subsection in the APS-B and R&S-B tend to find comparable solutions, but with APS-B doing so more expeditiously. The RME method again underperforms with respect to its peers. Nevertheless, the performance of APS-A is distinct in these instances from the previous section. It tends to converge toward comparable solutions to that found by APS-B and R&S-B; however, the variability about its expected utility is great, indicating highly variable performance. As with the previous subsection, similar trends are also apparent across the other three problem types.

Table 4.6 depicts the mean performance of each algorithm for the final attacks across the four design points. These values are calculated by averaging the attacks' performances over quantities referenced in Section 4.5.2. Despite the varied uncertainty, APS-B continues to effectively balance perturbed observations with impact, and is able to identify high-quality but low-cardinality attacks. Likewise, as in the structural testing, the RME algorithm finds high-perturbation attacks of varied impact. In juxtaposition, the two R&S approaches only identify the most impactful attack once out of the sixteen combinations. The APS-A algorithm performs much better across the uncertainty levels than the structural level; its impact is often among the top three of the examined attacks. Finally, no attack generates substantial impact on the path-attraction problem since the w_1/w_2-value examined makes the cost of the attack less than its reward.

The overall performance of the attacks appears to be influenced most by λ, i.e., the probability of an attack changing the true observation, in the state-attraction and -repulsion problems. However, this pattern does not necessarily hold in the other problems. Within the distribution-disruption instances, κ has the most influence, whereas both λ and κ have a marginal to nil effect on the path-attraction instances as a consequence of the high w_1/w_2-values. The APS-A, APS-B and R&S-B algorithms identify a high-quality attack in the state-attraction, state-repulsion, and distribution-disruption problems. However, the identified attacks balance impact and data perturbation differently between algorithms. The totality of these results illustrate that, in general, the trends identified in the previous subsection with regard to algorithmic performance

are valid with varied uncertainty levels while the improved performance of APS-A is an exception.

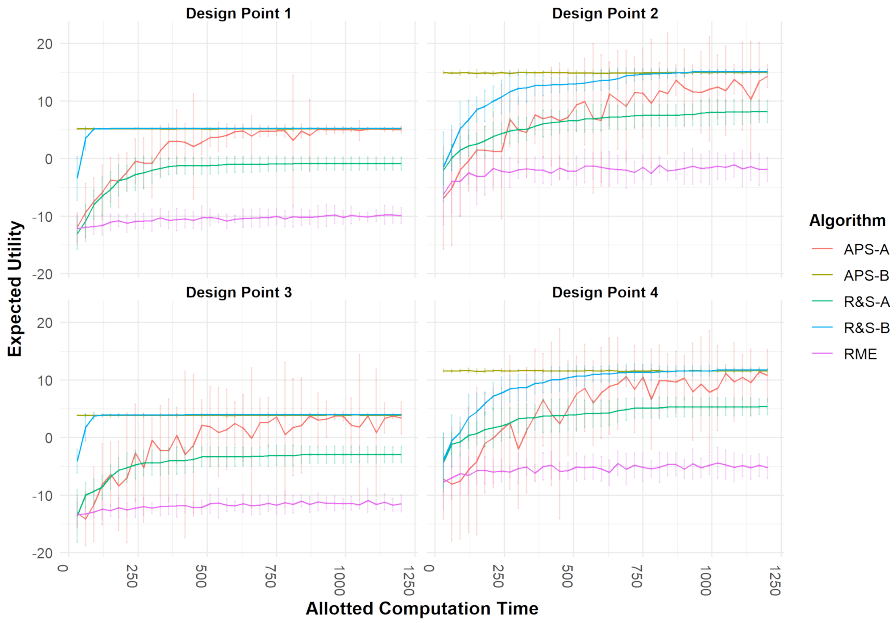


Figure 4.6: Temporal Evolution of Distribution-Disruption Expected Utility by Algorithm

Table 4.6: Mean Impact and Proportion of Observations Attacked (Uncertainty Testing)

Problem	Algorithm	Design Pt. 1		Design Pt. 2		Design Pt. 3		Design Pt. 4	
		Impact	$\Delta/\lvert\mathcal{T}\rvert$	Impact	$\Delta/\lvert\mathcal{T}\rvert$	Impact	$\Delta/\lvert\mathcal{T}\rvert$	Impact	$\Delta/\lvert\mathcal{T}\rvert$
State Att.[1]	APS-A	0.255	**0.145**	0.254	0.140	0.121	**0.085**	0.121	0.080
	APS-B	**0.266**	0.150	0.247	0.135	**0.133**	0.090	0.127	**0.085**
	R&S-A	0.171	0.625	0.162	0.550	0.108	0.570	0.116	0.625
	R&S-B	0.216	0.275	**0.259**	0.145	0.122	0.345	**0.133**	0.090
	RME	0.167	0.830	0.156	0.810	0.079	0.755	0.077	0.765
State Rep.[2]	APS-A	0.328	**0.050**	0.319	**0.050**	0.261	**0.050**	0.252	**0.050**
	APS-B	**0.329**	**0.050**	**0.330**	**0.050**	**0.266**	**0.050**	0.259	**0.050**
	R&S-A	0.318	0.470	0.322	0.570	0.252	0.505	0.249	0.560
	R&S-B	0.324	0.260	0.327	0.220	0.258	0.265	**0.266**	0.230
	RME	0.313	0.755	0.308	0.755	0.246	0.725	0.235	0.750
Dist Disrupt.[3]	APS-A	**2.171**	0.075	**1.394**	0.090	1.629	0.075	**1.156**	0.070
	APS-B	2.057	**0.050**	1.364	**0.050**	1.603	**0.050**	1.082	**0.050**
	R&S-A	2.153	0.525	1.364	0.595	1.622	0.595	1.071	0.615
	R&S-B	2.104	0.225	1.352	0.185	**1.677**	0.295	1.089	0.185
	RME	2.154	0.820	1.387	0.900	1.478	0.795	1.104	0.900
Path Att.[4]	APS-A	**0.941**	0.045	**0.940**	0.105	**0.942**	0.065	**0.945**	0.060
	APS-B	0.950	**0.000**	0.950	**0.000**	0.950	**0.000**	0.950	**0.000**
	R&S-A	0.942	0.365	0.954	0.325	0.955	0.090	0.952	0.490
	R&S-B	0.951	0.190	0.964	0.295	0.954	0.045	0.955	0.145
	RME	**0.941**	0.750	0.953	0.750	0.947	0.770	**0.945**	0.750

[1] Impact is the difference between perturbed and unperturbed data of the probability of i' at t'
[2] Impact is the difference between unperturbed and perturbed data of the probability of i' at t'
[3] Impact is the KL divergence between state distributions under unperturbed and perturbed data
[4] Impact is the Normalized-Hamming distance of most-likely state sequences perturbed data and attacker's goal

4.5.4 Case Study: Attacking an HMM for Part-of-Speech Tagging

To highlight the practical relevance of our methods, we consider attacks against a larger-scale HMM used for natural-language processing (NLP). In particular, we are motivated by part-of-speech (POS) tagging, a widely used approach in the lexical analysis of text data. POS tagging improves accuracy of text analysis by reducing the computational effort required for data processing and revealing the syntactic structure of sentences. Herein, hidden states correspond to a POS, and the observations are words. Emission probabilities refer to the probability of a word given a POS, whereas transition probabilities capture POS sequencing. Attacks in this section are accomplished using an Apple M2 workstation with 8 GB of RAM and 8 CPU cores in order to demonstrate algorithm efficacy even by using a standard laptop.

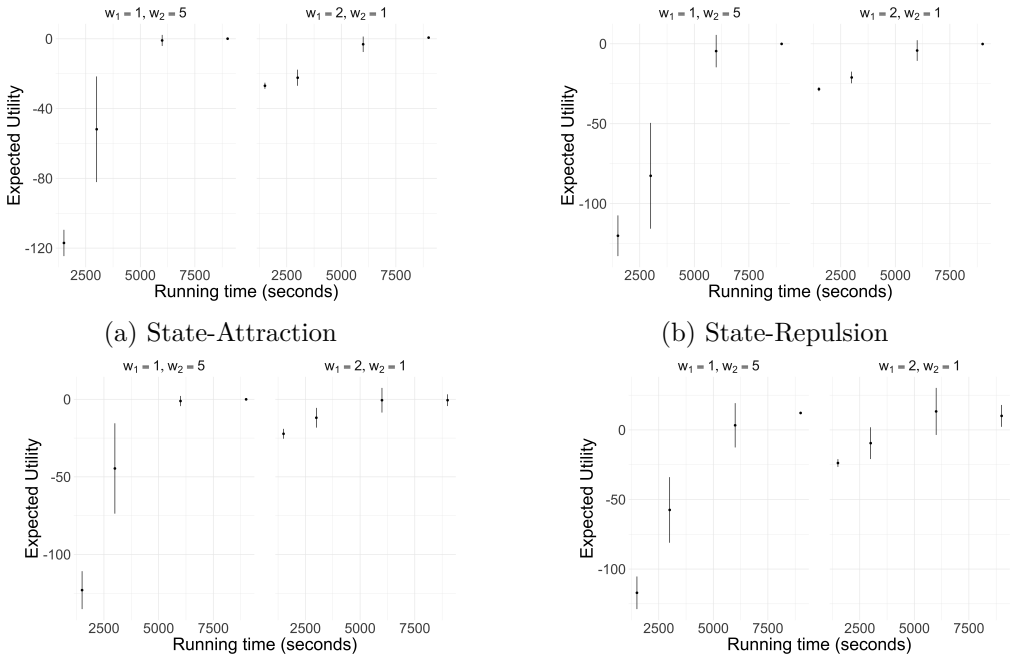
Concretely, we train an HMM on the Named-Entity-Recognition (NER) dataset (Kaggle, 2023). The top-300 words in the database are considered, along with all 29 POSs, in the examination of 30-word text strings. The trained HMM is taken as the decision-maker's model having $|\mathcal{Q}| = 29$, $|\mathcal{X}| = 300$, $|\mathcal{T}| = 30$. As in previous sections, the attacker's prior beliefs are centered about the D's true model. We work in a low uncertainty scenario. We implement R&S-B method for this case-study due to its better scalability with respect to $|\mathcal{X}|$.

Let the following pre-processed phrase from the NER data be the uncorrupted text:

> *mr. said among countries six party talks (among china, south korea, japan, russia, united states north korea) north korea's nuclear program*

Figure 4.7 examines the utility of attacking this phrase via the R&S-B method across the state-attraction, state-repulsion, distribution-disruption and path-attraction problems. Specific details regarding the associated POS information for the state-attraction and -repulsion problems, as well as the desired POS sequences for the path-attraction problem are available online in the code repository for this chapter. Attack success probabilities are similar to the previous sections and are also detailed in the code repository. For each of the problems, we again vary the objective weights. We alternatively set (w_1, w_2) to $(1, 5)$ and $(2, 1)$ to explore its effect. These weights are selected to ensure problem difficulty; namely, we do not want the attacker to be incentivized to make drastic changes to the uncorrupted text. For each of the problem-and-weight combinations, the attack was allowed to search the solution space for 1500, 3000, 6000 and 9000 seconds. Each experiment was performed 10 times to enable estimation of the mean expected utility as well as its standard deviation.

Examination of Figure 4.7 reveals several noteworthy insights. For each of the ten R&S-B attacks performed on each problem-and-weight pair, a different $\hat{z}^*$ may be identified based on the stochasticity inherent in the simulated-annealing optimization. Inspection of Figure 4.7 confirms intuition that lower computation times result in attacks of lesser quality. As additional computation time is allocated, it is apparent that not only does the solution quality improve, but the reduction in variability suggest that the algorithms converge toward an attack across the ten runs. Alternatively, variability in the expected utility measurements is reduced when $(w_1, w_2) = (2, 1)$, implying that, even when allocated less computation time, the runs converge toward comparable solutions.



(a) State-Attraction

(b) State-Repulsion

(c) Distribution Disruption

(d) Path Attraction

Figure 4.7: Expected Utility of Attacks Plus/Minus One Standard Deviation in the POS Case Study

Additional information regarding the performance of the attacks generated using 9000 seconds of computational effort is detailed in Table 4.7. The same performance metrics are tallied as in Table 4.6. Notably, whenever $w_2 > w_1$, it can be observed that the attack is minimally perturbing the true data with limited impact due to the higher weighted cost. More aggressive and impactful attacks are identified when $w_1 > w_2$. The totality of these results highlight that a conventional machine can effectively corrupt HMMs in a grey-box setting, even when the optimal attack minimally perturbs the uncorrupted data.

Table 4.7: Mean Impact, Proportion of Observations Attacked and Expected Utility (NER Experiment)

Problem	w_1	w_2	Impact	$\Delta/\lvert T\rvert$	Expected Utility
State Attraction[1]	1	5	0.00	0.00	0.07
	2	1	0.77	0.13	0.68
State Repulsion[2]	1	5	0.00	0.00	-0.07
	2	1	0.00	1.0	-0.14
Distribution Disruption[3]	1	5	1.47	0.01	-0.03
	2	1	11.21	0.77	-0.59
Path Attraction[4]	1	5	0.53	0.00	12.17
	2	1	0.71	0.74	10.08

[1] Impact is the difference between perturbed and unperturbed data of the probability of i' at t'

[2] Impact is the difference between unperturbed and perturbed data of the probability of i' at t'

[3] Impact is the KL divergence between state distributions under unperturbed and perturbed data

[4] Impact is the Normalized-Hamming distance of most-likely state sequences perturbed data and attacker's goal

This feature is exemplified in the state-attraction case having $(w_1, w_2) = (2, 1)$ wherein the attacker wishes to encourage the classification of the fifth word as a singular proper noun. The best attack identified across all ten runs is listed below:

> *mr. said among countries u.s. party talks (among china, south korea, japan, russia, united states north korea) north korea's nuclear program*

Notably, this text string is quite similar to the uncorrupted data. Their variations could easily be attributed to a typographical error or a bug in the text's pre-processing routine. However, this small change greatly increases the probability that the attacker achieves his goal; the attack increases the posterior probability of a singular-proper-noun by 0.77. The ability of this attack to thwart D's HMM with minimal changes to the true data is reminiscent of adversarial examples in other applications, e.g., computer vision (Goodfellow et al., 2014).

4.6 Discussion

HMMs are probabilistic graphical models with broad applications, ranging from NLP and thermodynamics to security and cybersecurity. In this chapter, in line with objective **O3**, we demonstrate that like other ML techniques, HMMs are vulnerable to adversarial attacks. Using an ARA perspective, we formulate stochastic programming problems designed to disrupt the HMM inference tasks of filtering, smoothing, and decoding. Because these problems are nonlinear, combinatorial, and subject to parameter uncertainty, we develop three tailored algorithms to obtain tractable solutions: a R&S heuristic, an APS approach, and MC enumeration.

To assess the practicality and robustness of these methods, we evaluate their performance through extensive experiments. First, we measure how effectively data corruption alters inference on a modestly sized HMM using MC enumeration, providing a benchmark for solution quality. We then analyze the R&S and APS approximation routines on larger HMMs, highlighting how parameter uncertainty and model size influence their computational efficiency and accuracy. Finally, we display the real-world practicality of the implemented attacks by applying them to an HMM model used for POS tagging.

The experimental results confirm that the proposed methodologies can successfully disrupt the studied HMM inferences. These findings underscore the need for robustification strategies to secure HMM-based systems and suggest promising directions for future work, including extensions to real-time inference and applications to other probabilistic graphical models beyond HMMs.

Chapter 5

A comparative study of machine learning models for bot detection and their application on Twitter

5.1 Introduction

AI has become increasingly integrated into cybersecurity controls, enabling the processing and analysis of vast amounts of data for the automatic detection of cyber threats. Among these threats, bots, which are automated accounts controlled by software, have emerged as a particularly significant concern on social media platforms such as Twitter. These automated accounts can manipulate online discourse by disseminating low-credibility information or fake news, thereby influencing public opinion. Such effects are especially troubling when discussions involve sensitive or high-impact contexts, including health crises or armed conflicts. This chapter focuses on the detection of bots, which we consider an ML binary classification problem, and illustrates through two case studies how it can be effectively applied in the context of Twitter.

This work contributes to the *technical security controls* component of CSRM by addressing **O4**. Specifically, we advance the study of bot detection by comparing several supervised ML models that differ in their data requirements and input features, and assessing their effectiveness in identifying bots with diverse behaviors. Building on this comparison, we further demonstrate two practical applications of bot detection on Twitter. First, we analyze the presence of bots across a range of topics related to the COVID-19 pandemic and compare the sentiment (i.e., the emotional tone or attitude) of their tweets with those of human accounts. Second, we introduce WIDISBOT, a software tool designed to help non-technical users compare content shared by bots to that shared by humans, and demonstrate how it can be used to explore bot activity around specific topics.

The remainder of this chapter is structured as follows. Section 5.2 presents the comparative study of the ML-based bot detection models. Then, Section 5.3 examines the differences in bot presence across topics related to the COVID-19 pandemic, and Section 5.4 describes WIDISBOT. Finally, we provide a brief discussion of the results presented in this chapter.

5.2 Comparative study of bot detection models

This section presents a comparison of bot detection models. We consider the bot detection task an ML classification problem, where we aim to determine whether an account is a bot or a

human. First, we describe the process of obtaining the features used in the detection techniques and introduce the datasets employed for training and testing. Then, we evaluate the performance of the different classification models implemented using data selection (Yang et al., 2020), a method that aims to identify a subset of training data that optimizes CV performance on the training set as well as cross-domain generalization, that is, the ability to classify accounts from domains different from the training data. Finally, we compare the best implemented models with Botometer v3 (Yang et al., 2019), one of the most cited tools in the field of bot detection (Rauchfleisch and Kaiser, 2020).

5.2.1 Feature engineering

The features used as input to the classification algorithms for bot detection can be divided into two groups: (a) those obtained from each account's metadata and (b) those acquired using compression statistics in each account's *social fingerprinting*.

The first group uses the account metadata and new variables derived from them. The metadata is retrieved from the User Object, a data structure that contains information about a user's account and is obtained using the Twitter API v1.1 (Twitter Developer, 2020). The features retrieved are listed in Table 5.1.

Table 5.1: Name and description of features retrieved directly from User Object.

Feature name	Description
statuses_count	Number of tweets posted, including retweets.
followers_count	Number of followers.
friends_count	Number of accounts followed.
favourites_count	Number of tweets liked by the account.
listed_count	Number of public lists in which the account is involved.
default_profile	Boolean indicating if the profile's theme or background has been altered.
verified	Boolean indicating that the user has a verified account.

The features derived from the metadata of the User Object are listed in Table 5.2. The variable *user_age* is used to compute some derived features from the metadata, where *user_age* corresponds to the difference in hours between the creation time of the last accessible tweet and the creation time of the user.

Table 5.2: Name and description of features derived from User Object.

Feature name	Description
screen_name_length	Length of screen name string.
num_digits_in_screen_name	Number of digits in screen name string.
name_length	Length of name string.
num_digits_in_name	Number of digits in name string.
description_length	Length of description string.
friend_growth_rate	*friends_count/ user_age.*
listed_growth_rate	*listed_count / user_age.*
favourites_growth_rate	*favourites_count / user_age.*
tweet_freq	*statuses_count / user_age.*
followers_growth_rate	*followers_count / user_age.*
followers_friend_ratio	*followers_count / friends_count.*
screen_name_likelihood	Geometric mean of likelihood of all bigrams in a screen name.

The intuition behind *screen_name_likelihood* (Table 5.2) is that the screen name (username used to identify accounts on the platform, preceded by "@") of bot-operated accounts sometimes is constituted by a random string (Yang et al., 2020), being this a distinctive characteristic from humans. We retrieved more than 2 million unique screen names from Twitter random accounts to compute the likelihood of each of the 3969 bigrams that can be created using the characters allowed in the screen name (upper- and lower-case letters, digits, and underscore).

The second group of features uses *social fingerprinting*, a technique that involves modeling the behavior of an account through its *digital DNA*, which is a string of characters representing the sequence of actions performed by a Twitter account. The behavior is encoded by mapping the actions to characters, or bases, forming a set of unique characters called the alphabet, which is used to generate a sequence represented by a row vector or string which encodes user behavior (Cresci et al., 2017a). More formally, an alphabet $\mathbb{B}$ is defined as (Pasricha and Hayes, 2019)

$$\mathbb{B} = \{B_1, B_2, \ldots, B_N\} \quad B_i \neq B_j \quad \forall i, j = 1, \ldots, N \wedge i \neq j$$

where $\{B_1, B_2, \ldots, B_N\}$ denotes the set of bases representing the distinct types of actions performed by an account. Then, $\mathbb{B}$ is used to generate the sequence

$$s = (b_1, b_2, \ldots, b_n) = b_1 b_2 \ldots b_n \quad b_i \in \mathbb{B} \quad \forall i = 1, \ldots, n.$$

In our experiment, we use the following alphabet to encode a Twitter user behaviour:

$$\mathbb{B}^3_{type} = \left\{ \begin{array}{ll} A \leftarrow & \text{tweet} \\ \text{C} \leftarrow & \text{reply} \\ \text{T} \leftarrow & \text{retweet} \end{array} \right\} = \{\text{A}, \text{C}, \text{T}\}$$

A Twitter account's behavior is captured through its timeline and is utilized to generate the DNA sequence. For instance, if an account first did a retweet, then two tweets, and finally, a retweet, its $\mathbb{B}^3_{type}$ sequence is $TAAT$. Hence, the sequence's length depends on the number of actions considered. We retrieved the maximum number of tweets (including retweets and replies) for each account, with 3,200 being the limit imposed by the Twitter API for tweet retrieval (Twitter Developer, 2020), which thus defines the maximum possible length of an analyzed sequence. Accounts which were protected or did not possess any timeline could not be analyzed with this methodology.

The DNA sequences generated from the timelines are compressed using a lossless compression algorithm from the Python library, `zlib`. In particular, as in Pasricha and Hayes (2019), we use `zlib.compress()` with default parameters to compress the DNA strings and then the function `getsizeof()` from the Python module `sys` to obtain the size in bytes of the compressed and uncompressed DNA strings. Hereafter, we represent DNA strings through the following features: *original size of DNA string*, *compressed size of DNA string* and *compression ratio* (*original DNA size*/*compressed DNA size*).

For our experiment, we use all the sets of features shown in Table 5.3, and compare the performance of the ML models implemented with each one. *Light* corresponds to the features used for bot detection by Yang et al. (2020), except for the feature *profile_use_background_image* since it has been deprecated from the Twitter API (Twitter Developer, 2020). *Light* allows implementing a scalable bot detection technique since each tweet retrieved with the Twitter API versions 1.1 and Gnip 2.0 (Twitter Developer, 2020) contains the User Object of the corresponding account, with no need of obtaining extra data. Methodologies using sets *A*, *B*, *C*, and *D*, which are based on the research by Pasricha and Hayes (2019), scale worse.

Table 5.3: Set of features used for the comparison of bot detection models.

Set name	Features included
Light	Features extracted and derived from the User Object.
A	Original size and compressed size of DNA string.
B	Original size and compression ratio of DNA string.
C	Compressed size and compression ratio of DNA string.
D	Original size, compressed size and compression ratio of DNA string.

5.2.2 Datasets

This section presents the datasets used to compare bot detection models. As Yang et al. (2020), we used some datasets for training, and other datasets were set aside for testing. In this way, we aim to build a bot detection model that performs well in CV on the training data and generalizes to accounts with new behaviors, achieving cross-domain generalization. Datasets were obtained from https://botometer.iuni.iu.edu/bot-repository, with Tables 5.4 and 5.5 displaying those used for training and testing, respectively, together with a brief description.

Table 5.4: Datasets used for training, from Lee et al. (2011), Cresci et al. (2017b), Varol et al. (2017), Yang et al. (2019), and Yang et al. (2020).

Dataset	Description
caverlee	Created using honeypot accounts to attract bot-operated accounts. Mainly spammers, malicious promoters, and friend infiltrators.
cresci-17	Manually annotated accounts, labelled accounts from other datasets, and bots purchased in online markets. Bots include retweet spammers for political campaigns, hashtag spammers, URL spammers, job-promoting bots, fake followers, and URL scammers.
varol	Manually annotated accounts from different deciles of Botometer scores.
pronbots	Shared on GitHub. The bots are Twitter advertising scam sites.
political	Politics-oriented bots shared by Twitter user @john_emerson.
botometer-feed	Manually annotated accounts reported by Botometer users.
vendor-purchased	Uniquely composed of fake follower bots.
celebrity	Composed uniquely of human accounts, in particular, celebrity accounts.

Table 5.5: Datasets used for test, taken from Yang et al. (2020), Mazza et al. (2019), Gilani et al. (2017), Rauchfleisch and Kaiser (2020), Cresci et al. (2018), and Cresci et al. (2019).

Dataset	Description
botwiki	Bots extracted from botwiki.org.
verified	Human-verified user accounts extracted through the Twitter Streaming API.
rtbust	Created by manually annotating the retweets retrieved.
stock	Bots detected through similarities in timelines of accounts containing tweets with specific cashtags over five months. Bots present a coordinated behaviour.
gilani	The dataset was formed by retrieving accounts with the Twitter Streaming API and splitting them into four groups regarding their followers. Then, accounts from each group were extracted and annotated manually.
midterm	Accounts that interacted during the 2018 U.S. Midterm elections, manually annotated using the correlation between tweeting and creation timestamps.
kaiser	American and German politician accounts are labelled as humans. Bots are manually annotated for German accounts and extracted from botwiki.org for English accounts.

The *botwiki* and *verified* datasets are considered together during the test as *botwiki-verified*. It is worth mentioning that the datasets used for training are the same as in Yang et al. (2020), while for testing, we added the datasets *stock*, which contain bots with coordinated nature (Yang et al., 2020), and *kaiser*. With this, we wanted to test the models with other bots of different nature. Tables 5.6 and 5.7 display the number of bots and humans which constitute each dataset for training and test, dividing them between User Object and Social Fingerprint. The differences between the columns are due to the fact that, as mentioned in Section 5.2.1, it is not possible to apply DNA methods to accounts that are protected or lack a timeline. These differences are not considered big enough to be misleading when comparing the User Object and Social Fingerprint approaches.

Table 5.6: Number of bots and humans in each training dataset divided depending on whether we use the features from the User Object and the Social Fingerprint.

Training datasets	User Object		Social Fingerprint	
	Human	Bot	Human	Bot
botometer-feed	347	108	337	107
varol	1525	690	1331	668
political	0	13	0	13
cresci-17	2907	5925	2440	5607
celebrity	5814	0	5763	0
vendor	0	731	0	718
pronbots	0	1899	0	1723
caverlee	15211	14619	12824	14156

Table 5.7: Number of bots and humans in each test dataset divided depending on whether we use the features from the User Object and the Social Fingerprint.

Test Dataset	User Object		Social Fingerprint	
	Human	Bot	Human	Bot
rtbust	332	321	317	314
gilani	1418	1043	1293	997
kaiser	1007	290	959	232
botwiki-verified	1985	685	1974	610
midterm	7416	37	7290	32
stock	6132	6964	5333	6246

5.2.3 Comparison between different bot detection models using data selection

This section compares different bot detection models that use the features and datasets described above. In implementing these models, we apply *data selection* and, using a heuristic approach, select those that achieve the optimal balance between CV performance and cross-domain generalization. We then compare these models with Botometer v3 ($BTM3$). This version of Botometer has been used in several studies in the literature and has even been considered a state-of-the-art tool for bot detection on Twitter (Sayyadiharikandeh et al., 2020). It is based on a Random Forest classifier that uses more than 1,000 features from each account, related to various aspects such as tweet content, sentiment, account network, and user metadata. The model was trained on the following datasets: *caverlee*, *varol*, *cresci-17*, *pronbots*, *vendor*, *botometer-feed*, *celebrity*, and *political* (Varol et al., 2017).

Specifically, we can obtain the best models using all training data when meeting the following conditions (Yang et al., 2020):

- The labels of the train data are correct.
- The data are independent and identically distributed in the feature space.

We can employ a data selection method if these conditions are not met, which is the scenario that we assume. We perform data selection on the training data by identifying the optimal set of at least two training datasets from the 247 possible combinations for implementing a classification model.

For each one of the 247 combinations of training datasets, each feature set (*Light*, *A*, *B*, *C*, *D*) is used with the following classification algorithms: *Logistic Regression*, *AdaBoost*, *Support Vector Machine with Linear Kernel*, *Random Forest*, *Gradient Boosting*, *K Nearest Neighbors (KNN)*, *Naive Bayes*, *MLP* (Sarker, 2021). Each possible tuple formed by a combination of training datasets, a feature set, and a classification algorithm is evaluated using the AUC score on all test datasets. The *MLP* had one hidden layer for sets *A*, *B*, *C* (120 neurons), and *D* (150

neurons), and two hidden layers in the case of *Light* (300 and 200 neurons). We use the default hyperparameters of the library `sklearn` (Pedregosa et al., 2011) for the other algorithms.

We created 9880 different models based on the 247 combinations of training datasets, eight algorithms, and five sets of features. We selected 5 of them through a heuristic process, with which we aim to identify a model that may not perform best in every single test but performs consistently well across all test datasets, ensuring stability in applications. The process is the following:

1. We group the models by feature set (obtaining five groups), and within each group, we evaluate each of the 247×8=1976 models against all the 6 test sets.
2. We create a ranking for each test set in each group (6 rankings per group; in each one, every model gets a value in the range 1 to 1976 based on its AUC score), and then we compute the sum of the six ranking values obtained by each model (1976 sums per group).
3. For each algorithm in each group, we take the model with the lowest ranking sum (8 models per group).
4. In each group, we manually select the best performing model out of the eight lowest-sum-models based on the AUC scores on the test sets and its 5-fold CV value. We based our selection primarily on the test results, always checking that the model performs well overall. We prioritize the CV if we get similar results on the test datasets.

Table 5.8 shows the best models for each set of features according to our heuristic, along with the AUC score of the models in each test dataset and in a 5-fold CV. We observe that the models with the features obtained through social fingerprint outperform or obtain similar results to the model with the set *Light* in all cases. The *stock* dataset is where the models whose features are extracted from Digital DNA outperform more evidently the model with *Light* feature set, with the model with the set of features *D* obtaining the best result. This is because the bots in the *stock* dataset showed a coordinated behavior that makes a model dependent on metadata features as *Light* not convenient for detection. In contrast, as evidence shows, the social fingerprints and compression statistics effectively detect bots with coordinated behavior. Besides, we observe that the data selection is effective since none of the best models for each set of features used all the training datasets.

Table 5.8: Best model for each set of features.

Feature set	Model	Training datasets	AUC Scores						
			rtbust	*gilani*	*kaiser*	*botwiki -verified*	*midterm*	*stock*	*5-fold*
Light	*Gradient Boosting*	*botometer-feed*, *varol*, *cresci-17*, *celebrity*	0.613	0.631	**0.944**	**0.991**	**0.964**	0.631	0.961
A	*Logistic Regression*	*political*, *cresci-17*	0.700	0.719	0.943	**0.991**	0.962	0.922	0.957
B	*Gradient Boosting*	*botometer-feed*, *political*, *cresci-17*	**0.720**	**0.726**	0.938	**0.991**	0.961	0.858	**0.968**
C	*Random Forest*	*political*, *cresci-17*	0.660	0.691	0.927	0.980	0.944	0.863	**0.968**
D	*Logistic Regression*	*botometer-feed*, *cresci-17*	0.699	0.719	**0.944**	**0.991**	0.962	**0.924**	0.945

Comparison with $BTM3$

We compared the performance of the best models using the feature sets *Light* and D with $BTM3$. The three models present significant differences. Both $BTM3$ and the model using *Light* use features extracted from the account metadata, whereas the model using D constructs the Digital DNA from the account's timeline for prediction. Another difference is the number of features used by each model to classify accounts: while $BTM3$ uses more than 1,000 features, the model using *Light* utilizes 19 features, and the model using D, only 3. However, the main difference among all the models lies in scalability. The model that uses only *Light* features requires just one tweet for detection, which allows detection to occur at the same time as tweets are retrieved, since the User Object is embedded within each tweet. In contrast, the other model requires access to each account's timeline for detection, which limits scalability due to Twitter API rate limits.

In this section, in addition to the AUC score, the following metrics were used to evaluate each model's performance: F1, Accuracy, Recall, Precision, and Specificity. To compute these metrics, we use a classification threshold of 0.3 for *BTM3* (Luceri et al., 2019), while for the models using sets D and *Light*, the threshold corresponds to the value that maximizes the F1 metric through CV (Yang et al., 2020).

Table 5.9 shows the performance of the three models. We observe that the model using D performs consistently well overall, outperforming or achieving results comparable to the other two

models. It is worth highlighting its strong performance on the *stock* dataset, where it achieves the best results. This provides evidence that the compression statistics extracted from the Digital DNA can effectively detect bots exhibiting coordinated behavior. Moreover, it is possible to build a classifier that generalizes correctly across different domains by combining set D with data selection. Alternatively, the model using *Light* produces similar results to those of the other models, except for the *stock* dataset, where it does not perform well due to the coordinated nature of the accounts. This model also demonstrates the best specificity in all cases, in addition to the advantage of being scalable. Finally, $BTM3$ appears to be more robust against bots in *stock* than the model using *Light*, probably because its features cover additional aspects beyond user metadata. These results confirm that competitive performance can be achieved using only a small set of features, as in *Light* and D, rather than relying on a larger feature set as in $BTM3$.

Table 5.9: Comparison in performance of $BTM3$ and the best models with the set of features *Light* and D.

Test Dataset	Model	Evaluation metrics					
		AUC	F1	Accuracy	Recall	Precision	Specificity
botwiki-verified	*Light*	0.990	0.916	**0.960**	0.857	**0.985**	**0.995**
	D	**0.991**	**0.917**	**0.960**	**0.936**	0.899	0.968
	$BTM3$	0.922	0.785	0.905	0.685	0.920	0.980
gilani	*Light*	0.631	0.274	0.615	0.172	0.681	**0.941**
	D	**0.718**	**0.508**	**0.670**	**0.390**	**0.726**	0.886
	$BTM3$	0.689	0.456	0.644	0.341	0.687	0.880
kaiser	*Light*	**0.944**	**0.817**	**0.919**	0.807	**0.827**	**0.951**
	D	**0.944**	0.683	0.837	**0.901**	0.550	0.822
	$BTM3$	0.829	0.594	0.827	0.572	0.617	0.899
midterm	*Light*	**0.964**	**0.176**	**0.964**	0.784	**0.099**	**0.964**
	D	0.962	0.051	0.859	0.875	0.027	0.859
	$BTM3$	0.958	0.101	0.912	**0.905**	0.054	0.912
rtbust	*Light*	0.613	0.217	0.536	0.131	0.636	**0.928**
	D	**0.699**	0.451	0.567	0.357	0.612	0.776
	$BTM3$	0.625	**0.473**	**0.584**	**0.377**	**0.636**	0.788
stock	*Light*	0.631	0.375	0.495	0.285	0.548	**0.732**
	D	**0.924**	**0.819**	**0.771**	**0.960**	**0.714**	0.549
	$BTM3$	0.756	0.780	0.719	0.927	0.673	0.480

Based on the insights gained from the experiments in this section, we use the model implemented with the *Light* feature set for the two practical applications of bot detection on Twitter

presented in Sections 5.3 and 5.4, as it demonstrated good performance and scalability.

5.3 Analyzing bot presence and sentiment differences during the COVID-19 pandemic

Bots may distort public debate, for instance, by producing content that differs from that of humans or by disseminating disinformation. Such behavior can be particularly hazardous during a global health emergency like COVID-19, when much of the discussion takes place on social media platforms such as Twitter.

This section, as an application of the studied bot detection models to Twitter, uses the best model with set *Light* implemented in the previous section, to examine the prevalence of bots and their posting volume compared to human accounts across different topics during the first months of the COVID-19 pandemic. We then analyze the sentiment of tweets produced by bots and compare it with that of human users to assess how it differs between the two groups. To conduct the experiment, we used hashtags to identify the tweets related to the same topic. We considered a tweet to belong to a certain topic if it contains hashtags with the topic name or a subvariant. For instance, if we consider COVID as a topic, tweets with hashtags such as #COVID19, #covid, and #CovidPandemic would belong to it.

Tweets used in the experiment were extracted from public datasets in Chen et al. (2020), Lamsal (2020), Banda et al. (2020) and Kaggle. The tweets include specific hashtags or keywords related to COVID-19, or they come from particular accounts such as the World Health Organization. Although most datasets contain tweets in multiple languages, the majority are in English because the hashtags and keywords used to collect them are primarily English terms. These datasets, due to Twitter regulations, contain only the tweets' IDs. Consequently, it was necessary to retrieve the full tweet objects corresponding to those IDs by using the twarc library (Documenting the Now, 2020). Once we retrieved the tweets, we filtered them to include only those written in English for the experiment.

Table 5.10 displays the topics and periods that we analyze in this section, as well as the number of unique tweets and accounts considered after retrieving the tweets. We studied these topics and periods because we considered them likely to exhibit bot presence, as they involved controversial issues. In the following subsections, we refer to each case as OUTBREAK, WUHAN, COVID, LOCKDOWN, and TRUMP. We briefly describe each topic and the relevant events[i] related to each one:

- WUHAN and OUTBREAK refer to the pandemic beginning where the virus had rapidly spread in China and received names such as "Wuhan virus" or "Wuhan coronavirus". In this context, authorities canceled large-scale events such as the Spring Festival, and there were traveling restrictions for more than 30 million people. Moreover, 15 Chinese cities suffered partial or complete lockdowns.

- The COVID topic on March 28th and 29th coincides with Trump considering quarantining New York, the shortage of equipment for health workers, and hospitals overload. Moreover, the United States surpassed 2000 deaths.

- In the scope of LOCKDOWN on May 10th, a high criticism was raised from the first steps out of the lockdown proposed by U.K. Prime Minister Boris Johnson.

[i]The mentioned events are extracted from reliable media sources according to https://mediabiasfactcheck.com.

- The TRUMP case refers to President Trump's management of the early stages of the pandemic, which was widely criticized. In this period, there were problems with COVID testing in the U.S., making it difficult to stop the spread of the virus. Besides, little attention was given to the coronavirus in the State of the Union on 4th February. Moreover, during February, the U.S. government had to manage the critics concerning the Americans' conditions in the Diamond Princess cruise ship.

Table 5.10: Number of accounts and tweets for each case studied.

Topic	Date	Accounts	Tweets
OUTBREAK	25/01/2020–26/01/2020	64602	82030
WUHAN	25/01/2020–26/01/2020	103916	163723
COVID	28/03/2020–29/03/2020	312034	414097
LOCKDOWN	10/05/2020	26813	31052
TRUMP	04/02/2020–21/02/2020	10144	26865

5.3.1 Bot analysis: presence of bots across topics

We classify each account as a bot or human using the best model with feature set *Light* and compute the number of tweets produced by bots and humans in each case. Table 5.11 (Left) displays the proportion of accounts classified as bots and humans in each case. On the other hand, Table 5.11 (Right) shows the proportion of total tweets produced by each type of account across topics.

Table 5.11: Proportion across topics of (Left) bots and human accounts and (Right) tweets by bots and human accounts.

Topic	Human accounts (%)	Bot accounts (%)
OUTBREAK	**93.85**	6.15
WUHAN	**92.88**	7.12
COVID	**89.95**	10.05
LOCKDOWN	**88.69**	11.31
TRUMP	**81.74**	18.26

Topic	Human tweets (%)	Bot tweets (%)
OUTBREAK	**93.35**	6.65
WUHAN	**90.71**	9.29
COVID	**88.79**	11.21
LOCKDOWN	**87.12**	12.88
TRUMP	44.27	**55.73**

From these two tables, we obtain the following conclusions:

- OUTBREAK and WUHAN have the smallest amount of bots, while TRUMP has the largest.

- In all cases, except for TRUMP, the proportions of each type of account and tweets made by those accounts are analogous. This fact indicates that bots and humans present a similar activity rate in these cases.

- Conversely to the other cases, bots are more active than humans in TRUMP. The bots, only 18.26% of the accounts, produce 55.73% of total tweets.

5.3.2 Differences in the sentiment between humans and bots

We analyze the sentiment of tweets posted by bots and humans across the topics, allowing us to observe whether bots produce content with different sentiment than humans. We use VADER (Hutto and Gilbert, 2014) to compute the sentiment for all the tweets. VADER is a computationally efficient and scalable model that performs well across various domains, particularly in analyzing microblogging content. In fact, according to Ribeiro et al. (2016), it is an effective method for predicting three-class sentiment (positive, neutral, negative) in social network messages. It employs a lexicon to assign scores to each word, subsequently combined using five rules that consider grammatical and syntactical aspects. The output is a unidimensional continuous metric (y) ranging from -1 (most negative) to 1 (most positive). To categorize y discretely, we use the thresholds provided by the authors: positive if $y \geqslant 0.05$, negative if $y \leqslant -0.05$, and neutral if $-0.05 < y < 0.05$.

Table 5.12 displays the proportions of positive, neutral, and negative tweets for each case after using the sentiment thresholds above. We obtain the following insights:

- In contrast to OUTBREAK, WUHAN, and COVID, the cases LOCKDOWN and TRUMP exhibit the most significant differences in proportions of positive, neutral, and negative tweets when comparing the sentiment of bots and humans. It might indicate that bots have attempted to alter the natural course of conversation around these two themes, albeit in two different ways.

- In LOCKDOWN, humans mainly display a negative tendency compared to bots. It might be interpreted as a public disagreement with the first steps out of the lockdown proposed by the UK Prime minister.

- In TRUMP, the tweets generated by the bots have a predominantly negative attitude, while humans show a slighter dissent with President Trump's political performance.

Table 5.12: Proportion of sentiment of tweets from human and bot accounts across topics.

Topic	Human accounts (%)			Bot accounts (%)		
	Positive	Neutral	Negative	Positive	Neutral	Negative
OUTBREAK	**44.96**	41.01	14.03	36.18	**40.91**	22.90
WUHAN	17.72	**41.38**	40.89	18.53	37.23	**44.24**
COVID	33.37	30.04	**36.59**	**37.24**	30.49	32.27
LOCKDOWN	25.77	23.49	**50.74**	**37.22**	30.59	32.19
TRUMP	30.68	27.01	**42.30**	11.46	11.53	**77.01**

The experiments in this section have some limitations. First, VADER is not attuned to tweets about politics, reducing its performance on occasions. Besides, using hashtags to extract tweets with the same topic might be spam-sensitive. Twitter users can use hashtags to gain popularity or attention, though unrelated to the tweet content. Moreover, our extraction method can retrieve some tweets which are not entirely related to the topic we are studying. That said, the limitations are not significant enough to prevent us from gaining valuable insights into the presence of bots in certain topics related to the COVID-19 pandemic, or from analyzing differences in sentiment between humans and bots.

5.4 Development of an application to analyze the content spread by bots

As a second use case of bot detection on Twitter, we developed a tool to scrutinize and compare content disseminated by bots and human users. While it can be used to study a wide range of online content, disinformation campaigns represent a specific and particularly relevant case, as they highlight how bots can influence online public conversation.

The tool, named WIDISBOT (standing for Widget to Analyze Disinformation and Content Spread by Bots), was built using Streamlit[ii]. Its primary goal is to provide professionals without technical expertise with an effective means of examining how bots propagate content, including disinformation. It empowers them to contribute to research on these agents and enrich the field with insights from diverse disciplines. The tool enables the comparison of discourse between bots and humans within specific Twitter conversations. Users can input either a Twitter query or tweet IDs to conduct further analyses of tweet datasets. WIDISBOT provides an intuitive interface for visualizing differences in discourse between automated and human users by applying several analyses, such as sentiment and word frequency. Additionally, it supports the in-depth examination of fabricated or misleading content propagated by these entities.

The following subsections describe WIDISBOT and present an illustrative example of how to use it to analyze the content produced by bots.

[ii]https://streamlit.io/

5.4.1 Application description

This section presents an overview of the application's functionalities. When analyzing tweets through the various functionalities, the input format requires Tweet Objects obtained via the Twitter API, either v1.1 or v2, and the related User Object representing the tweet author. The functionalities of WIDISBOT are:

- *Data extraction (DE).* It enables the retrieval of tweets by connecting to the API; therefore, valid credentials are required. In particular, the user may extract tweets by ID, or via search containing a certain keyword, hashtag or URL on a specific date. This functionality is limited by Twitter API restrictions and rate limits. The generated dataset can then be used as an entry to any other WIDISBOT functionality.

- *Monitoring (M).* It identifies which of the input tweets were generated by bots or humans. Additionally, it displays a density plot showing the likelihood that the analyzed accounts are bots, together with the proportion of accounts classified as bots or humans and the number of tweets produced by each account type.

- *Forensics (F).* Given the accounts' usernames, it computes the likelihood of them being bots, allowing the results to be presented in an aggregated manner.

- *Sentiment analysis (SEA).* It computes the sentiment of the input tweets, displaying the human/bot sentiment distribution in both a discrete (positive-negative-neutral) and continuous fashion. The sentiment analysis is computed using VADER, where the discrete labels are determined based on the thresholds used in the experiment described in Section 5.3. For the continuous distribution, the unidimensional continuous metric y output by VADER is used.

- *Hashtag analysis (HA).* It allows the visualization of the most frequently used hashtags by both humans and bots within the input tweets. This functionality is not case-sensitive, as bots may utilize variations of the same hashtag to promote diverse content.

- *Wordcloud (W).* It provides a visualization of the 25 most frequent words on the tweets shared by bots and humans.

- *Analysis of spread sources (ASS).* It displays the most shared URLs by bots and humans. It is connected to the Wayback Machine[iii] to retrieve the content from deleted websites, as content spread by bots is often removed after a certain time. The app also provides access to Media Bias Fact Check[iv] to determine the bias of a media and whether it is a non-reliable source.

- *Analysis of discourse around hashtags (ADH).* It enables the utilization of the functionalities *M*, *SEA*, *HA*, *ASS* on tweets that contain a specific hashtag, allowing for the analysis of how the discourse surrounding the given hashtag is influenced by both bots and humans.

The application uses the best model with feature set *Light* to determine whether an account is a bot or a human. This model was implemented for tweets extracted by Twitter API v1.1. format. However, if the input tweets are in API v2 format, we employ an adapted version of the model that does not consider features inaccessible in API v2 but available in v1.1. Table

[iii] https://archive.org/web/
[iv] https://mediabiasfactcheck.com/

5.13 shows that the adapted version achieves results comparable to those of the original model when using the *Light* features extracted using the Twitter API v1.1 and $BTM3$ across the test datasets described in Section 5.2.

Table 5.13: AUC scores of the models using set *Light* for tweets extracted using Twitter API v2 and v1.1, and of $BTM3$, on the test datasets from Section 5.2.

	Test datasets					
	Botwiki-verified	Rtbust	Gilani	Kaiser	Stock	Midterm
Light (v2)	.975	.518	.580	.936	.653	.947
Light (v1.1)	**.990**	.613	.631	**.944**	.631	**.964**
$BTM3$	.922	**.625**	**.689**	.829	**.756**	.958

Besides, the classification of input accounts as bots or humans by WIDISBOT is conducted using a threshold specified by the user. A higher threshold leads to a more cautious approach by the model in determining which accounts are classified as bots. Additionally, the application enables users to download files with the results of the various functionalities for further analysis, either manually or in another application.

5.4.2 Use Case Demonstration

This section displays how the software could be used to study the role of bots in a potential disinformation campaign. For illustrative purposes, we selected a set of 527 tweets from the TRUMP topic in Section 5.3 dated February 7th, which contain the words "Trump" and "deathtoll" in any uppercase or lowercase variant. We aim to display how to use WIDISBOT to study whether bots intended to promote certain content by taking advantage of the crisis situation. We follow the steps below to carry out the tweets' analysis:

1. *Analysis of bot presence.* Utilising the *M* functionality, we examined the proportion of tweets produced by bots compared to humans. In Figure 5.1 (left) we observe that a smaller number of bots produced a larger proportion of the total tweets than humans, an indication that bots are being used to promote content in this conversation.

2. *Checking differences in sentiment.* Another indication of bot activity aimed at promoting content different from that of humans may be found in differences in sentiment distribution between bots and humans. We used the *SEA* functionality to determine if any differences were present. Specifically, as depicted in Figure 5.1 (right), we observed substantial discrepancies, evidence about the different content that bots and humans are sharing.

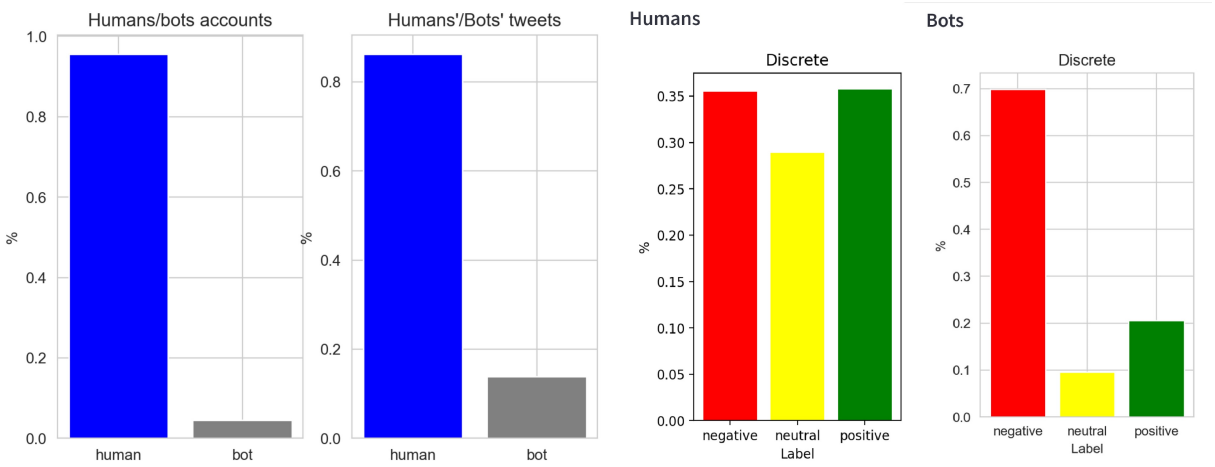


Figure 5.1: Screenshots of WIDISBOT output. (Left) Proportion of accounts labelled as bots/humans and the fraction of tweets produced by each type. (Right) Sentiment distribution in human/bots tweets.

3. *Checking differences between hashtags.* Through the *HA* functionality, we examined how hashtags were used by both groups of accounts. The results for the ten most used hashtags by bots and humans are depicted in Figure 5.2. We observed that bots used more hashtags and, while there was a stair-like shape in the human case, the bots had several hashtags with the same number of occurrences. This may be an indication that bots are promoting their content using multiple hashtags in the same tweets.

4. *Studying tweets with a certain hashtag.* We studied the hashtag *#deathtoll* as it was highly shared by bots, but not at the same rate as the first six hashtags, and it was not among the most frequently used hashtags by humans. We utilized the *ASH* functionality and discovered that only one human and one bot posted tweets with the hashtag. However, the bot produced 44 tweets while the human produced only one. Furthermore, we examined the URLs shared by the bot on these tweets, observing that it shared 34 times the same URL.

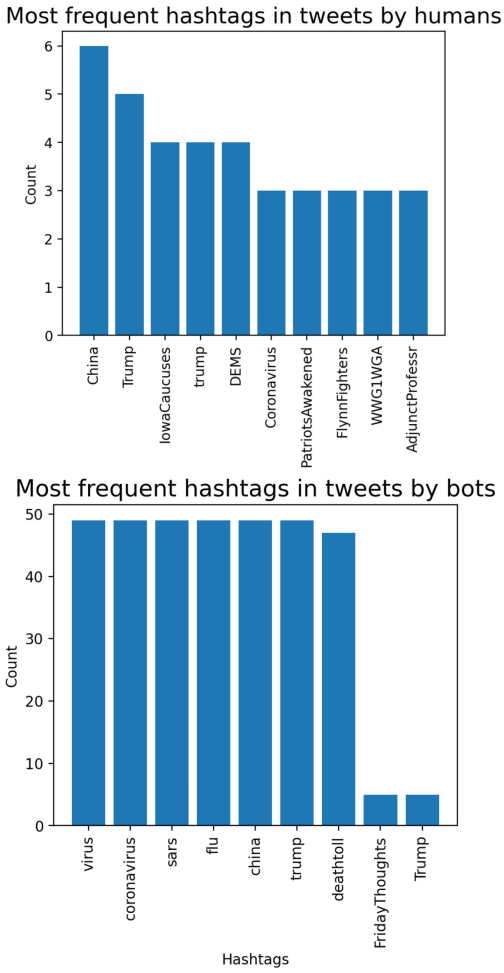


Figure 5.2: Ten most shared hashtags by bots and humans.

5. *Analysis of the most shared URLs.* We browsed the most shared URL by the bot, finding out that it is no longer available. We used the *ASS* functionality to retrieve the version of the website archived by the Wayback Machine at the time the tweet was posted. Figure 5.3 displays the website. It can be observed that some content is advertised, such as how to survive without medication or publicity about masks. Hence, we have uncovered that the identified bot was disseminating content that could potentially contribute to disinformation during the COVID-19 pandemic.

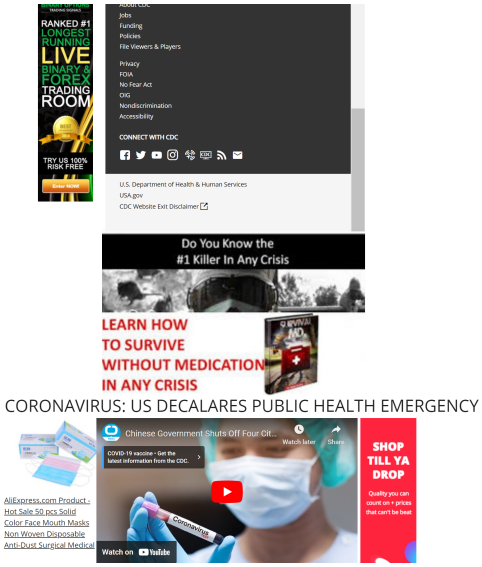


Figure 5.3: Screenshot of the most shared website by bots in tweets with hashtag *#deathtoll*, accessed through the Wayback Machine.

5.5 Discussion

This chapter has addressed Objective **O4** by comparing several supervised ML models for bot detection on Twitter, complemented by two practical demonstrations of their application. Together, these contributions strengthen the *technical security controls* component of the CSRM scenario.

In the comparative study, we analyzed different models using a data selection approach based on features extracted from account metadata, as well as compression statistics obtained from Social Fingerprint and $BTM3$. The goal was to identify a consistent model that achieved the best balance between CV performance and cross-domain generalization. Our results show that models leveraging compression statistics achieved the highest performance, effectively detecting bots even when they exhibited coordinated behavior. Although the most scalable model produced slightly worse results, its performance remained competitive, making it suitable for subsequent experiments.

In the first case study, we examined various topics discussed during the COVID-19 pandemic and analyzed differences in sentiment between human users and bots using the VADER sentiment analysis tool. We found that discussions related to Donald Trump's management of the pandemic exhibited the highest volume of bot activity, with bot-generated content expressing more negative sentiment.

In the second case study, we introduced WIDISBOT, a Streamlit-based widget specifically developed to identify bots on Twitter and analyze the content they disseminate compared to that of human users. This tool provides a comprehensive perspective of the information shared by both groups. Additionally, we presented an example demonstrating how WIDISBOT can be used to uncover bots potentially spreading misleading content.

Chapter 6

Conclusions and future work

This chapter outlines the thesis main conclusions and discusses some future research lines.

6.1 Conclusions

We provide a summary of the main insights obtained from the work presented in Chapters 2, 3, 4 and 5. Along these chapters, we not only contributed to CSRM and AI in a broader sense, but also focused specifically on certain components of the CSRM landscape in relation to AI systems and services, addressing the objectives presented in Section 1.5.

Chapter 2 fulfils objective **O1** by advancing on CSRM methodologies from a general standpoint. It proposes a comprehensive cybersecurity risk analysis framework tailored to systems that incorporate AI components and services. This framework addresses new challenges introduced by AI, to wit: managing AI-specific impacts, integrating AI-based assets within the cyber architecture, considering AI-based security and recovery controls within the cybersecurity portfolio and addressing novel AI-targeted attacks.

The framework expands the CIA triad to address AI-related impacts and maps them onto the CV20 objective tree, enabling their systematic integration into risk analysis and decision-support. To handle AI-based assets, it models the cyber architecture of systems using a granular, block-based approach, employing directed graphs divided into hierarchical levels. This allows for the explicit representation of these assets, their identification as either critical components or vulnerable entry points, and their dependencies on other elements, which leads to more realistic risk analysis.

Moreover, it enables the inclusion of AI-based security controls into portfolio optimization, allowing decision-makers to assess their impact on risk reduction alongside costs. To address targeted attacks, including those involving AI, the framework incorporates an ARA approach that allows handling the uncertainty related to the adversarial aspects of the threats. This facilitates more realistic defense planning and portfolio optimization, as it supports the estimation of attack probabilities and entry points through simulations that account for attacker incentives and capabilities.

A major advantage of the implemented framework is that it provides a quantitative indicator of system risk, overcoming the limitations of qualitative approaches. The framework is also highly flexible, as it can adapt to different system architectures, types of AI components, attack scenarios, and defense strategies. Its modular block-based modeling and probabilistic nature allow organizations to customize the analysis to their specific assets, risks, and regulatory requirements, making it applicable across diverse AI-enabled systems.

The framework has the potential to support compliance with the AIA, the NIST AIRMF, and similar regulatory frameworks. By explicitly addressing AI-specific risks (such as safety, fairness, and transparency) and integrating them into a rigorous risk analysis and management process, it may help organizations demonstrate proactive risk assessment and adherence to AI governance standards.

The chapter uses a case study on ADS to illustrate and validate the framework. By modeling the system using the granular block representation, evaluating multiple attack scenarios, and comparing different security portfolios, it shows substantial risk reduction and improved resilience. This practical example demonstrates how the proposed approach may effectively support detailed risk analysis and informed decision-making in real-world AI-integrated systems. While the approach is technical and demands intensive modeling work, the values at stake are so significant that the extra effort is justified.

Chapter 3 delivers on objective **O2**, providing a contribution to the *targeted threats component* of CSRM, as well as to the *technical security control* component, as it could be applied in the AML context. The chapter introduces a methodology that uses APS to address general security games modeled as proper BAIDs with incomplete information from an ARA perspective. This approach provides a unified framework capable of approximating the solution of any game represented as a proper BAID, regardless of the number of decisions each agent makes or whether the decision spaces are discrete or continuous. In doing so, it offers the first general solution to a problem posed by Shachter (1986). In addition, the chapter discusses computational issues that may arise when the methodology is applied to continuous decision spaces and offers practical solutions, including approximating the (random) value function of the (attacker) defender and improving the efficiency of APS.

The methodology is exemplified through a disinformation war example, illustrating how it helps a decision maker plan effective and proactive strategies to defend against an attacker, while showing how the solutions to the computational issues are implemented in practice. It demonstrates that the approach can handle realistic, complex decisions to support defensive planning in adversarial settings, as well as obtain probabilistic forecasts about the attacker's behavior.

Chapter 4 then meets objective **O3** and strengthens the *technical security controls* component of CSRM by examining the vulnerabilities of HMM inference to data corruption. It highlights the critical need to understand and mitigate the risks of adversarial manipulations in HMM and AI-based systems that integrate them, an area largely unexplored in the AML literature, and represents a first step toward making them more robust. To explore these vulnerabilities, the attacker's problem of corrupting the input batch data is modeled, resulting in a suite of mathematical programming formulations developed to manipulate HMM inferences in filtering, smoothing, and decoding contexts. This modeling leverages an ARA approach to address the adversarial setting's uncertainty. To solve these formulations, three algorithmic methods are introduced: an R&S heuristic, an approach based on the APS method used in Chapter 3, and an MC enumeration strategy. These algorithms are extensively tested empirically with respect to the structure of, and the uncertainty in, the target HMM. The results illustrate the significant impact that data perturbations can have on HMM inferences and show that high-quality attacks can be obtained within reasonable computational times, even for complex problem instances under partial knowledge. Lastly, a case study on POS tagging demonstrates the practical relevance of these attacks, showing that such vulnerabilities are not purely theoretical but can also affect AI systems relying on HMM inference.

Finally, Chapter 5 fulfils objective **O4**, contributing to the *technical security controls* component of CSRM. The chapter presents a comparison of several supervised ML methods for bot

detection using data selection, analyze the presence of bots during the COVID-19 pandemic and develop WIDISBOT, a prototype application with a user-friendly interface to help non-technical users analyze content disseminated by bots.

By comparing several ML models, we aimed to identify the one that achieves the best balance between CV performance and cross-domain generalization. In our evaluation, we used data selection to compare models that use features extracted from account metadata with those that employ compression statistics of the social fingerprint. Our work extends that from Yang et al. (2020) by using additional test datasets, the metadata available from the Twitter API at the time of our research, and several classification algorithms. The experiments demonstrated that combining compression statistics with data selection yielded excellent results, not only outperforming the models that use features derived from metadata in certain scenarios, but also proving more effective in detecting bots with coordinated behavior. Alternatively, the models using metadata, after testing different classification algorithms, also produced competitive results. We selected the best of the models using metadata to analyze the presence of bots during COVID-19 and develop WIDISBOT due to its performance and scalability.

6.2 Future work

This section proposes possible directions for future research from the developments of this thesis:

The CSRM framework presented in Chapter 2 can be enhanced through the development of a user-friendly software that integrates it, possibly adopting the terminology of well-known organizations like ENISA (Papadatos et al., 2023). This software could include reusable templates for similar systems, covering blocks, security components, impacts, preference models and distributions, which would simplify elicitation tasks and make system evaluations more scalable. By introducing this software as an extension, the framework would become more widely adopted and applicable across different scenarios.

Additionally, beyond serving as a risk analysis tool, the framework's risk assessment stage could help determine a system's position within the EU AI Act's four-tier risk ladder, alleviating the challenges in implementing the AI Act and guiding the implementation of necessary security controls. This mapping could also be incorporated into the proposed application to further support legal compliance and certification programs as demanded in the EU Cybersecurity and Cyber-relilience Acts (Cihon, 2019). Besides, the framework could be used to address the design of AI systems, promoting a security-by-design approach.

Future work for the ARA methodology proposed in Chapter 3 includes extending the methodology from bi-agent to multi-agent settings, addressing both the computational and conceptual challenges that may arise.

Another possible direction is to implement the methodology in an open-source Python library, making it accessible to researchers and practitioners applying ARA to their own problems. Given a BAID representation of their problem and the relevant parameters, users could then obtain the optimal decisions along with forecasts of the attackers' behaviour.

In addition, exploring the use of Hamiltonian MC methods for APS represents a promising avenue for future research, as it may accelerate convergence and reduce computational costs. This will facilitate the application of the proposed approach to real-world scenarios in the emerging field of AML, where the methodology will be particularly relevant, as algorithmic solutions are imperative to address the inherent challenges in this domain. The deployment of ARA solutions

in such contexts is essential, as *common knowledge* assumptions typically used in AML are often invalid, thereby requiring a framework that more accurately handles uncertainty.

Moreover, in the disinformation case study, we approximate functions and conditional distributions using NN trained on samples obtained over fixed grids in the input space. While this approach ensures broad coverage, it can be computationally inefficient. A possible direction for future work is the use of adaptive or optimal experimental design to select input points more strategically to approximate the Defender's optimal values and the Attacker's distributions. By focusing on sampling from regions of high variation or model uncertainty, such methods could improve approximation quality while reducing the number of required evaluations.

The algorithmic approaches developed in Chapter 4 suggest several possible directions for future inquiry. Given the vulnerability of HMM to data perturbation, there is a clear need for more robust HMM inference algorithms. Therefore, one potential research line should consist of implementing methods to protect these algorithms. Specifically, a new ARA-based methodology could be developed to help HMMs make accurate inferences during operations in the presence of adversaries, similar to the approach in Gallego et al. (2024) for protecting classifiers.

Another research direction consists of relaxing the various assumptions developed to suit distinct real-world settings, such as real-time inference. Moreover, given the generality of the HMM corruption problems, the variety of prior distributions available to the attacker, and the flexible hyperparameterization of the R&S heuristic, further experimentation on these algorithms is a worthwhile endeavor, as is the development of other attacks.

Alternatively, the efficacy of the developed attacks on the inference of trained HMMs also suggests that traditional approaches to HMM learning may be vulnerable, indicating that future research should focus both on corrupting the Baum-Welch algorithm and on developing modifications to make it more robust.

Additionally, similar questions to those posed for standard HMMs can be formulated with respect to other probabilistic graphical models. Notably, the problems and attacks presented here can be extended to inference over alternative dynamic Bayesian networks, including Kalman filters and autoregressive HMMs, which can serve as a first step toward developing defenses to make them more robust.

A possible future direction for the work in Chapter 5 is to extend the comparative study by including additional bot detection methods from the literature and incorporating new datasets published since then, to assess whether other techniques achieve better performance in terms of CV and cross-domain generalization. Furthermore, analyzing the specific bot behaviors that the most effective detection methods fail to address is important in order to develop defenses with increased robustness.

In addition, future extensions of WIDISBOT could involve incorporating more ML models to analyze other aspects of bot discourse, such as identifying content that constitutes hate speech, and integrating these approaches with applications focused on detecting specific forms of misinformation e.g., Arroyo et al. (2023), thereby enhancing the versatility of WIDISBOT in specific contexts.

Chapter 7

Conclusiones y trabajo futuro

Este capítulo expone las principales conclusiones de la tesis y discute algunas líneas de investigación futura[i].

7.1 Conclusiones

Ofrecemos un resumen de las principales aportaciones obtenidas a partir del trabajo presentado en los Capítulos 2, 3, 4 y 5. A lo largo de estos capítulos, no solo hemos contribuido a la GRCS y a la IA en un sentido amplio, sino que también nos hemos centrado específicamente en ciertos componentes del panorama de la GRCS en relación con sistemas y servicios basados en IA, abordando los objetivos presentados en la Sección 1.5.

El Capítulo 2 cumple el objetivo **O1** al avanzar en las metodologías de GRCS desde una perspectiva general. Propone un marco integral de análisis de riesgos de ciberseguridad adaptado a sistemas que incorporan componentes y servicios de IA. Este marco aborda los nuevos desafíos introducidos por esta disciplina: la gestión de impactos específicos de la IA, la integración de activos basados en IA dentro de la arquitectura de ciberseguridad, la consideración de controles de seguridad y recuperación basados en IA dentro de la cartera de ciberseguridad, y la atención a nuevos ataques dirigidos usando IA.

El marco amplía la tríada CID para abordar impactos relacionados con la IA y los asigna al árbol de objetivos CV20, lo que permite su integración sistemática en el análisis de riesgos y en la toma de decisiones. Para gestionar los activos basados en IA, el marco modeliza la arquitectura de ciberseguridad de los sistemas mediante un enfoque granular basado en bloques, utilizando grafos dirigidos divididos en niveles jerárquicos. Esto permite la representación explícita de dichos activos, su identificación como componentes críticas o puntos de entrada vulnerables, así como sus dependencias con otros elementos, lo que conduce a un análisis de riesgos más realista.

Además, permite la inclusión de controles de seguridad basados en IA dentro de la optimización de la cartera de ciberseguridad, lo que facilita que los responsables de la toma de decisiones evalúen su impacto en la reducción del riesgo junto con sus costes asociados. Para abordar ataques dirigidos, incluidos aquellos que involucran IA, el marco incorpora un enfoque ARA que permite gestionar la incertidumbre relacionada con las amenazas adversarias. Esto facilita una planificación defensiva y una optimización de la cartera más realistas, ya que permite estimar las probabilidades de ataque y los puntos de entrada mediante simulaciones que tienen en cuenta los incentivos y capacidades del atacante.

[i]Como es preceptivo, se incluye esta traducción al castellano del último capítulo de la tesis.

Una de las principales ventajas del marco propuesto es que proporciona un indicador cuantitativo del riesgo del sistema, superando las limitaciones de los enfoques cualitativos. El marco también es altamente flexible, ya que puede adaptarse a diferentes arquitecturas de sistemas, tipos de componentes de IA, escenarios de ataque y estrategias de defensa. Su modelización modular basada en bloques y su naturaleza probabilística permiten a las organizaciones personalizar el análisis según sus activos, riesgos y requisitos normativos específicos, lo que lo hace aplicable a una amplia variedad de sistemas basados en IA.

El marco tiene el potencial de dar apoyo al cumplimiento de la ley de IA de la UE, del marco de gestión de riesgos de la IA del NIST y de otros marcos regulatorios similares. Al abordar explícitamente los riesgos específicos de la IA e integrarlos en un proceso riguroso de análisis y gestión de riesgos, puede ayudar a las organizaciones a mostrar una evaluación proactiva del riesgo y el cumplimiento de estándares de gobernanza de la IA.

El capítulo utiliza un caso de estudio sobre sistemas de conducción automatizada para ilustrar y validar el marco. Al modelizar el sistema, evaluar múltiples escenarios de ataque y comparar distintas carteras de seguridad, se demuestra una reducción sustancial del riesgo y una mejora de la resiliencia. Este ejemplo práctico muestra cómo el enfoque propuesto puede apoyar de manera efectiva un análisis de riesgos detallado y una toma de decisiones informada en sistemas reales integrados con IA. Aunque el enfoque es técnico y requiere un trabajo intenso de modelización, los valores en juego son tan significativos que el esfuerzo adicional está justificado.

El Capítulo 3 persigue el objetivo **O2**, aportando una contribución a la componente de amenazas dirigidas de la GRCS, así como al componente de controles técnicos de seguridad, dado que puede aplicarse en el contexto del Aprendizaje Automático Adversario (AAA). El capítulo introduce una metodología que utiliza Simulación de Probabilidades Aumentadas (SPA) para abordar juegos de seguridad generales modelizados como diagramas de influencia bi-agente (DIBA) propios con información incompleta desde una perspectiva ARA. Este enfoque proporciona un marco unificado capaz de aproximar la solución de cualquier juego representado como un DIBA propio, independientemente del número de decisiones que tome cada agente o de si los espacios de decisión son discretos o continuos. Asi, se aporta la primera solución general a un problema planteado por Shachter (1986). Además, el capítulo analiza los problemas computacionales que pueden surgir cuando la metodología se aplica con espacios de decisión continuos y propone soluciones prácticas, como la aproximación de la función de valor (aleatoria) del defensor (o atacante) y la mejora de la eficiencia de la SPA.

La metodología se ilustra mediante un caso referido a la guerra de desinformación, mostrando cómo ayuda a un decisor a planificar estrategias eficaces y proactivas para defenderse de un atacante, y presenta de forma práctica las soluciones propuestas a los desafíos computacionales de la metodología. Se demuestra así que el enfoque puede gestionar decisiones complejas y realistas de apoyo a la planificación defensiva en entornos adversarios, así como obtener previsiones probabilísticas sobre el comportamiento del atacante.

El Capítulo 4 se requiere al objetivo **O3** y refuerza el componente de controles técnicos de seguridad de la GRCS al examinar las vulnerabilidades de la inferencia en modelos ocultos de Markov (MOM) frente a datos corruptos. Destaca la necesidad crítica de comprender y mitigar los riesgos derivados de manipulaciones adversarias en los MOM y en los sistemas basados en IA que los integran, un ámbito en gran medida inexplorado dentro de la literatura AAA, y representa un primer paso hacia su mayor robustez. Para explorar estas vulnerabilidades, se modeliza el problema del atacante al corromper los datos de observación por lotes, lo que da lugar a un conjunto de formulaciones de programación matemática desarrolladas para manipular las inferencias de los MOM en contextos de filtrado, suavizado y decodificación. Esta modelización emplea un enfoque ARA para abordar la incertidumbre inherente en escenarios adversarios.

Para resolver estas formulaciones, se introducen tres métodos algorítmicos: una heurística de ordenación y selección, un enfoque basado en el método SPA utilizado en el Capítulo 3, y una estrategia de enumeración Montecarlo. Los algoritmos se evalúan empíricamente en profundidad, teniendo en cuenta tanto la estructura como la incertidumbre del MOM objetivo. Los resultados ponen de manifiesto el impacto significativo que las perturbaciones en los datos pueden tener sobre la inferencia en MOM y muestran que es posible obtener ataques de alta calidad en tiempos computacionales razonables, incluso para instancias complejas del problema y bajo conocimiento parcial. Por último, un caso de estudio sobre etiquetado de categorías gramaticales demuestra la relevancia práctica de estos ataques, evidenciando que tales vulnerabilidades no son meramente teóricas, sino que pueden afectar también a sistemas de IA que dependen de la inferencia mediante MOM.

Por último, el Capítulo 5 cumple el objetivo **O4**, contribuyendo al componente de controles técnicos de seguridad de la GRCS. El capítulo presenta una comparación de varios métodos de aprendizaje supervisado usando selección de datos para la detección de bots, analiza la presencia de bots durante la pandemia de la COVID-19 y desarrolla WIDISBOT, una aplicación prototipo con una interfaz de fácil uso para ayudar a usuarios sin conocimientos técnicos a analizar el contenido difundido mediante bots.

Al comparar varios modelos de aprendizaje automático, nuestro objetivo fue identificar aquel que lograra el mejor equilibrio entre el rendimiento en validación cruzada y la capacidad de generalización entre dominios. En nuestra evaluación, utilizamos selección de datos para comparar modelos que emplean características extraídas de los metadatos de las cuentas con otros que utilizan estadísticas de compresión de la huella social. Nuestro trabajo amplía el desarrollos anteriores al incorporar conjuntos de datos de prueba adicionales, los metadatos disponibles en la API de Twitter en el momento de la investigación y varios algoritmos de clasificación. Los experimentos demostraron que combinar estadísticas de compresión con selección de datos produce resultados excelentes, superando en ciertos escenarios a los modelos basados en metadatos y resultando más eficaz para detectar bots con comportamientos coordinados. Por otro lado, los modelos que utilizan metadatos, tras evaluar distintos algoritmos de clasificación, también ofrecieron resultados competitivos. Debidos a estos buenos resultados, así como su escalabilidad, seleccionamos el mejor de los modelos basados en metadatos para analizar la presencia de bots durante el COVID-19 y desarrollar WIDISBOT.

7.2 Trabajos futuros

Esta sección propone posibles líneas de investigación futura derivadas de los desarrollos presentados en esta tesis:

El marco de CSRM presentado en el Capítulo 2 puede ampliarse mediante el desarrollo de un software de fácil uso que lo integre, adoptando posiblemente la terminología de organizaciones ampliamente reconocidas como ENISA (Papadatos et al., 2023). Este software podría incluir plantillas reutilizables para sistemas similares, cubriendo bloques, componentes de seguridad, impactos, modelos de preferencias y distribuciones, lo que simplificaría las tareas de elicitación y haría que las evaluaciones de sistemas fueran más escalables. Mediante este software, el marco podría ser ampliamente adoptado y aplicable en distintos escenarios.

Además, más allá de servir como herramienta de análisis de riesgos, la fase de evaluación de riesgos del marco podría ayudar a determinar la posición de un sistema dentro de la escala de cuatro niveles de riesgo de la ley de la IA de la UE, aliviando los desafíos asociados a su implementación y guiando la incorporación de los controles de seguridad necesarios. Esta asignación también

podría integrarse en el software propuesto para apoyar aún más el cumplimiento legal y los programas de certificación exigidos en las leyes de ciberseguridad y ciberresilencia de la UE (Cihon, 2019). Asimismo, el marco podría emplearse para abordar el diseño de sistemas de IA, promoviendo un enfoque de seguridad desde el diseño.

El trabajo futuro para la metodología ARA propuesta en el Capítulo 3 incluye extender la metodología de escenarios con dos agentes a escenarios multiagente, abordando tanto los desafíos computacionales como los conceptuales que puedan surgir.

Otra posible línea de trabajo consiste en implementar la metodología en una biblioteca de Python de código abierto, haciéndola accesible para investigadores y profesionales que apliquen ARA a sus propios problemas. Dada una representación en forma de BAID de su problema y los parámetros relevantes, los usuarios podrían obtener las decisiones óptimas junto con las previsiones sobre el comportamiento de los atacantes.

Además, explorar el uso de métodos Montecarlo hamiltonianos para SPA representa una vía prometedora para investigaciones futuras, ya que puede acelerar la convergencia y reducir los costes computacionales. Esto facilitará la aplicación del enfoque propuesto a escenarios reales en el campo emergente del AAA, donde la metodología será particularmente relevante, dado que las soluciones algorítmicas son imprescindibles para abordar los desafíos inherentes a este ámbito. El despliegue de soluciones basadas en ARA en tales contextos es esencial, puesto que las hipótesis de *conocimiento común* habitualmente empleadas en AAA suelen ser inválidas, lo que requiere un marco que gestione la incertidumbre de forma más precisa.

Asimismo, en el estudio de caso sobre desinformación, aproximamos funciones y distribuciones condicionales utilizando redes neuronales entrenadas con muestras obtenidas sobre mallas fijas en el espacio de entrada. Si bien este enfoque asegura una cobertura amplia, puede resultar computacionalmente ineficiente. Una posible línea de trabajo futuro es el uso de diseños experimentales adaptativos u óptimos para seleccionar los puntos de entrada de manera más estratégica, con el fin de aproximar los valores óptimos del Defensor y las distribuciones del Atacante. Al centrarse en el muestreo de regiones con alta variación o incertidumbre del modelo, dichos métodos podrían mejorar la calidad de la aproximación a la vez que reducir el número de evaluaciones empleadas.

Los enfoques algorítmicos desarrollados en el Capítulo 4 sugieren varias posibles líneas de investigación futura. Dada la vulnerabilidad de los MOM frente a perturbaciones en los datos, existe una clara necesidad de contar con algoritmos de inferencia en MOM más robustos. Por lo tanto, una posible línea de investigación debería consistir en implementar métodos para proteger estos algoritmos. En concreto, podría desarrollarse una nueva metodología basada en ARA que permita a los MOM realizar inferencias precisas durante operación en presencia de adversarios, de manera similar al enfoque de Gallego et al. (2024) para proteger clasificadores.

Otra línea de investigación consiste en relajar las diversas suposiciones realizadas para adaptarse a distintos entornos del mundo real, como la inferencia en tiempo real. Además, dada la generalidad de los problemas de corrupción en HMM, la variedad de distribuciones a priori disponibles para el atacante y la parametrización flexible de la heurística de ordenación y selección, resulta valioso llevar a cabo más experimentación sobre estos algoritmos, así como desarrollar otros tipos de ataques.

Por otro lado, la eficacia de los ataques desarrollados sobre la inferencia de MOM entrenados también sugiere que los enfoques tradicionales para el aprendizaje de MOM pueden ser vulnerables, lo que indica que investigaciones futuras deberían centrarse tanto en corromper el algoritmo de Baum-Welch como en desarrollar modificaciones que lo hagan más robusto.

Además, pueden formularse cuestiones similares a las planteadas para los MOM estándar en relación con otros modelos gráficos probabilísticos. En particular, los problemas y ataques aquí presentados pueden extenderse a la inferencia en otras redes bayesianas dinámicas, incluidos los filtros de Kalman y los MOM autorregresivos, lo que puede constituir un primer paso hacia el desarrollo de defensas que los hagan más robustos.

Finalmente, una posible línea de investigación futura para el trabajo del Capítulo 5 es ampliar el estudio comparativo incluyendo métodos adicionales de detección de bots presentes en la literatura e incorporando nuevos conjuntos de datos publicados desde entonces, con el fin de evaluar si otras técnicas logran un mejor rendimiento en términos de validación cruzada y generalización entre dominios. Además, analizar los comportamientos específicos de los bots que los métodos de detección más eficaces no logran abordar resulta fundamental para desarrollar defensas con una mayor robustez.

Asimismo, futuras extensiones de WIDISBOT podrían consistir en incorporar más modelos de aprendizaje automático para analizar otros aspectos del discurso de los bots, como la identificación de contenido que constituya discursos de odio, e integrar estos enfoques con aplicaciones centradas en la detección de formas específicas de desinformación, aumentando así la versatilidad de WIDISBOT en contextos particulares.

Appendix A

Automated Driving System case study parametric setup

This appendix outlines the models and parameters adopted in the proposed ADS scenario in Chapter 2. It serves as well as a template for the process to be followed with other AI-based systems. The costs and other parameters are derived from public sources or extracted from one expert in cybersecurity and one expert in financial planning in the team, using standard expert judgment elicitation techniques as in Hanea et al. (2021). For a few representative parameters, we comment on some of their implications. Several of the parameters are subject to sensitivity analysis in Section 2.4.2.

A.1 Notation and general considerations

- *Portfolio.* To simplify the notation, we do not include the insurance products in $\mathbf{c}$. Thus, portfolios $\mathbf{c}$ will have the structure $(FwGw, AML, PmVs)$, with $FwGw$, AML, and $PmVs$ being 1 if the corresponding control is implemented and 0, otherwise. The initial portfolio will be $(0, 0, 0)$. We include only portfolios relevant when handling the corresponding threat; for example, an AML module is irrelevant against DoS attacks and would not be included in portfolios against such attacks. The implementation costs of the controls are: 1250 €(FwGw), 300 € (AML) and 1750 € (PmVs).

- *Access probabilities.* The access probabilities were defined as p_P, p_L and $p_{P,L}$.

- *Non-protection probabilities.* The *PNPs* were defined as q_P, q_L, $q_{D,P}$, and $q_{D,L}$.

- *Impacts.* Recalling the discussion in Section 2.3.2, denote by l_{FI} the financial impact of an attack on the system; l_{ED}, the equipment damage impact; and l_{DT}, the downtime impact. Assume that the rental car company experiences a cost of 100 € for each downtime hour of a single ADS. $l(\mathbf{c})$ indicates the loss when portfolio $\mathbf{c}$ is implemented. l_{FI} is a *global* impact, whereas l_{DT} and l_{ED} are *separable*: we denote l_{i_j} with $i \in \{ED, DT\}$ and $j \in \{L, P, D\}$ as the i-th impact on the j-th block. l_{FI}, l_{DT}, and l_{ED} will be sampled from Gamma distributions. Their parameters are detailed in subsequent sections, and we apply aggregation rules in Section 2.3.2. For the value function, we assume that each type of impact is equally important ($w_i = 1, i \in \{FI, ED, DT\}$), and we apply a risk-averse coefficient $\rho = 10^{-7}$ to the utility function.

- *Insurance products.* Product A reduces the economic impact over equipment damage by 65%. If product B is purchased, 70% of the total expenses resulting from equipment damage and downtime will be covered. Table A.1 displays the costs of the insurance products on an ADS depending on the adopted portfolio. Since these products have no effect on access and non-protection probabilities, when referring to these probabilities, we do not differentiate depending on the product considered, as, regardless of the insurance adopted, they will remain the same.

Table A.1: Prices in euros of insurance on an ADS depending on portfolio.

	(0,0,0)	(1,0,0)	(0,1,0)	(0,0,1)	(1,1,0)	(1,0,1)	(0,1,1)	(1,1,1)
A	600	500	500	500	250	250	250	150
B	1800	1600	1600	1600	1400	1400	1400	1000

A.2 Non-targeted attacks

This section discusses the distribution and parameters used to model non-targeted attacks that potentially threaten the ADS system. We model the arrival process, access and protection probabilities, and the impact of various attacks, given the portfolio.

A.2.1 Denial of Service attack features

Let us specify the distributions modeling the relevant parameters when the ADS is affected by a DoS attack.

- *Arrival process.* The number of potential DoS attacks within a one-year horizon is modeled through a Poisson distribution with mean 32, is based on the number of organizations in the transportation sector reported to have fallen victim to DoS attacks in the USA during 2022 (FBI, 2022).

- *Access probabilities.* Regardless of portfolio $\mathbf{c}$, we assume the same distribution for ($p_P(\mathbf{c})$, $p_L(\mathbf{c})$, $p_{P,L}(\mathbf{c})$). These probabilities are sampled from a Dirichlet $Dir(4, 8, 1)$ distribution, as the location system is more prone to being attacked through a DoS attack than the perception system. Such an attack is more likely to occur through the V2I system via an infected fixed element, say a traffic light, than through a component of the perception system. We posit an attack occurring through both blocks as less likely, since it would entail a too elaborate attack.

- *Non-protection probabilities.* Here are the *PNPs* given the portfolios.

  - *None.* We assume that $q_P(0, 0, 0) \sim Beta(27, 3)$. Among other things, this entails that the expected probability of not protecting the perception system when no additional measures are introduced is $27/(27 + 3)$. Similarly, assume $q_L(0, 0, 0) \sim Beta(26, 3)$, $q_{D,P}(0, 0, 0) \sim Beta(25, 3)$ and $q_{D,L}(0, 0, 0) \sim Beta(24, 3)$.
  - $FwGw$. When $FwGw$ is implemented, PNPs are distributed as $q_P(1, 0, 0) \sim Beta(5, 65)$, $q_L(1, 0, 0) \sim Beta(4, 65)$, $q_{D,P}(1, 0, 0) \sim Beta(3, 65)$ and $q_{D,L}(1, 0, 0) \sim Beta(2, 65)$. This entails e.g. that the expected PNP the perception system when FwGW is implemented is $5/(5 + 65)$, thus improving security against such threats.

- *PmVs.* The PNPs when a *PmV* is used are assessed as $q_P(0,0,1) \sim Beta(5,95)$, $q_L(0,0,1) \sim Beta(4,95)$, $q_{D,P}(0,0,1) \sim Beta(3,95)$ and $q_{D,L}(0,0,1) \sim Beta(2,95)$.
- *FwGw, PmVs.* If both protection measures are implemented, PNPs are modeled as $q_P(1,0,1) \sim Beta(5,125)$, $q_L(1,0,1) \sim Beta(4,125)$, $q_{D,P}(1,0,1) \sim Beta(3,125)$ and $q_{D,L}(1,0,1) \sim Beta(2,125)$.

- *Impacts.* The impact of a DoS attack depends on the implemented defense.
  - *Financial.* All financial impacts are modeled using Gamma distributions in K euros.
    - *None.* The impacts when no additional countermeasure is introduced would be modeled as $l_{FI}(0,0,0) \sim Gamma(7,3)$ K euros. This implies, for example, that the expected output is 21 K euros.
    - *FwGw.* The impacts would follow $l_{FI}(1,0,0) \sim Gamma(6,3)$K euros.
    - *PmVs.* The impacts are modeled as $l_{FI}(0,0,1) \sim Gamma(4,3)$K euros.
    - *FwGw, PmVs.* If both defenses are implemented, impacts are modeled as $l_{FI}(1,0,1) \sim Gamma(3,3)$K euros.
  - *Downtime.* The distributions that model the impact on each component are
    - *None.* Assume $l_{DT_P}(0,0,0) \sim Gamma(14,2)$, $l_{DT_L}(0,0,0) \sim Gamma(15,2)$ and $l_{DT_D}(0,0,0) \sim Gamma(18,3)$ hours. For instance, for the perception block, this entails that the expected downtime is 28 hours.
    - *FwGw.* The distributions are $l_{DT_P}(1,0,0) \sim Gamma(12,2)$, $l_{DT_L}(1,0,0) \sim Gamma(13,2)$ and $l_{DT_D}(1,0,0) \sim Gamma(16,3)$ hours.
    - *PmVs.* In this case, impacts are distributed as $l_{DT_P}(0,0,1) \sim Gamma(9,2)$, $l_{DT_L}(0,0,1) \sim Gamma(10,2)$ and $l_{DT_D}(0,0,1) \sim Gamma(13,3)$ hours.
    - *FwGw,PmVs.* The impacts would be distributed as $l_{DT_P}(1,0,1) \sim Gamma(8,2)$, $l_{DT_L}(1,0,1) \sim Gamma(9,2)$ and $l_{DT_D}(1,0,1) \sim Gamma(12,3)$ hours.

A.2.2 Software/hardware supply chain threat features

This section presents the models associated with a software/hardware SCT attack.

- *Arrival process.* The annual number of potential attacks resulting from SCT follows a Poisson distribution with parameter 2.75, derived from reported number of accidents associated with models from a specific brand by the National Highway Traffic Administration when employing an advanced driver assistance system (US. Dep. Trans., 2021), presuming that an attacker could exploit the same vulnerability in the vehicle software at a similar rate.

- *Access probabilities.* Consider equally likely that a perpetrator will target either of the two entry points and less likely that it will attack both simultaneously, precisely ($p_P(\mathbf{c})$, $p_L(\mathbf{c})$, $p_{P,L}(\mathbf{c})$) are modeled as $Dir(5,5,1)$. Assume the attacker faces the same difficulty level in accessing the system through a component regularly updated in the perception system, such as the V2V, or the location block, such as the GPS. Otherwise, we assume it is less likely an access attempt through an update affecting components in both entry blocks, as it would require a much more elaborate attack.

- *Non-protection probabilities.* Table A.2 displays the parameters of the *Beta* distributions employed to model PNPs given the implemented portfolio.

Table A.2: SCT PNP distribution parameters depending on portfolio.

Portfolio	q_P	q_L	$q_{D,P}$	$q_{D,L}$
(0,0,0)	(35,3)	(33,3)	(32,3)	(31,3)
(1,0,0)	(5,10)	(4,10)	(3,10)	(2,10)
(0,1,0)	(5,45)	(4,45)	(3,45)	(2,45)
(0,0,1)	(5,30)	(5,30)	(3,30)	(2,30)
(1,1,0)	(5,75)	(4,75)	(3,75)	(2,75)
(0,1,1)	(5,105)	(4,105)	(3,105)	(2,105)
(1,0,1)	(5,90)	(4,90)	(3,90)	(2,90)
(1,1,1)	(5,125)	(4,125)	(3,125)	(2,125)

- *Impacts.* Table A.3 quantifies the financial impacts (l_{FI}, l_{ED}, l_D) of an SCT through the specified Gamma distributions.

Table A.3: Impact distribution parameters of SCTs depending on portfolio.

Portfolio	l_{FI}	l_{ED}			l_D		
		P	L	D	P	L	D
(0,0,0)	(10,2)	(9,2)	(8,2)	(10,2)	(17,2)	(14,2)	(16,2)
(1,0,0)	(8,2)	(8,2)	(7,2)	(9,2)	(12,2)	(11,2)	(13,2)
(0,1,0)	(6,2)	(6,2)	(5,2)	(7,2)	(10,2)	(9,2)	(11,2)
(0,0,1)	(7,2)	(7,2)	(6,2)	(8,2)	(11,2)	(10,2)	(12,2)
(1,1,0)	(3,1)	(4,2)	(3,2)	(5,2)	(4,2)	(5,2)	(6,2)
(0,1,1)	(2,1)	(3,2)	(2,2)	(4,2)	(3,2)	(4,2)	(5,2)
(1,0,1)	(4,1)	(5,2)	(4,2)	(6,2)	(5,2)	(6,2)	(7,2)
(1,1,1)	(1,1)	(2,1)	(1,1)	(3,1)	(1,2)	(2,2)	(3,2)

A.3 Targeted attacks

This section presents modeling specifics of targeted attacks, requiring the assessment of attackers' motivations to predict if they find advantageous to target the system under study (Section 2.2.3). Details cover the attackers' utility functions taking into account the corresponding uncertainties, the attack arrival processes, and the success probabilities. Subsequently, we discuss distributions for PNPs and attack impacts. The distributions will reflect that the cyberterrorist group is better skilled than the criminal gang.

A.3.1 Attacker utility functions

Let us construct the attackers' utility functions, discussing their generic objectives and the parametric form adopted, finally delving into the uncertainties about their preferences.

- *Attackers objectives.* Attackers may pursue these objectives: *Maximizing notoriety nt*, an attacker striving to gain geopolitical influence; *Minimizing detection costs* c_d, associated

to the attacker being identified, including economic sanctions and/or legal condemnation, which could even lead to their disappearance; *Maximizing sensitive information s obtained,* the quantity of relevant data (valuable business information, customers' personal data,...) that an attacker illicitly obtains to sell for an economic gain. The *cyberterrorist group* wants to maximize its notoriety; the *criminal gang* aims to steal as much sensitive information as possible. Both aim to minimize detection costs. The costs of implementing wireless jamming or AML attacks are negligible; thus, attackers will not consider implementation expenses involved in carrying out such cyberattacks.

- *Utility parametric form.* Assume the attackers' preferences are modeled with the following piecewise risk-prone utility function

$$u_A(y, a, \mathbf{c}) = \exp(h_A \times (y \times \pi - c_d)),$$

where y represents the attack success (1, if successful; 0, otherwise), π denotes the variable that each attacker aims to maximize ($\pi_{Cy} = nt$, $\pi_{Cr} = s$), and h_A the risk proneness parameter. The defender's lack of knowledge about the attacker's preferences leads to the random utility model

$$U_A(y, a, \mathbf{c}) = \exp(H_A \times (y \times \Pi - C_d)),$$

where Π and C_d designate the random variables incorporating the uncertainties over the attacker's objectives. Assume $H_A \sim \mathcal{U}(1 \times 10^{-6}, 2 \times 10^{-6})$ models the uncertainty about the risk proneness (for both adversaries).

- *Specificities for cyberterrorist group.*

  - *Notoriety modeling.* As a proxy for notoriety, we aggregate the financial impact and number of deaths that the attacker is predicted to cause with an attack through $nt = l^A_{FI} + l^A_{FA} \times VSL$, where l^A_{FI} represents the financial impacts that the attacker anticipates causing to the targeted ADS and l^A_{FA} is the number of deaths expected by the attacker, and VSL is the value of statistical life (VSL), equivalent to 6 M €, based on the VSL for Spain (Viscusi, 2020) and an exchange rate euro-dollar of 0.9. VSL estimates changes in mortality risk in monetary terms and is easily misinterpreted as the financial value of a person's life (Cameron, 2010); however, there is still no consensus on rewording (Freeman III et al., 2014).

    We assume $l^A_{FI} \sim Gamma(\omega \times 8, \omega \times 3)$ and $l^A_{FA} \sim Poisson(\omega \times 25)$. Table A.4 displays the value of ω depending on whether the cyberterrorist group makes a wireless jamming or AML attack and the implemented portfolio. For instance, when no defenses are implemented ($\mathbf{c}$=(0,0,0)), the expected financial impacts for wireless jamming are $0.8 \times 8 \times 3 = 19200$ euros, and the expected number of deaths is $0.8 \times 25 = 20$. Acknowledging the lack of information about the other companies, we assume that $l^A_{FI} \sim Gamma(7.2, 2.7)$ and $l^A_{FA} \sim Poisson(22.5)$ for an AML attack when targeting the other companies. Similarly, in the case of a wireless jamming attack to other companies, we assume $l^A_{FI} \sim Gamma(5.6, 2.1)$ and $l^A_{FA} \sim Poisson(17.5)$.

Table A.4: Values of parameter ω for different portfolio configurations and attacks.

Portfolio	AML_at	wir_jam
(0,0,0)	1	0.8
(1,0,0)	1	0.6
(0,1,0)	0.2	0.8
(0,0,1)	1	0.4
(1,1,0)	0.2	0.6
(0,1,1)	0.2	0.4
(1,0,1)	1	0.2
(1,1,1)	0.2	0.2

  - *Detection costs and probability.* Denote by $p^{det}_{(1,j,k)}$ the detection probability of attack j targeting block k of our ADS, and $p^{det}_{(i,j)}$, the detection probability of attack j targeting company i's ADS, $i = 2, 3$. We assume for the AML attack that $p^{det}_{(1,AML_at,k)} \sim Beta(2, 998), k \in \{L, P\}$ and $p^{det}_{(1,AML_at,(L,P))} \sim Beta(1, 99)$. Regarding the wireless jamming attack, $p^{det}_{(1,wir_jam,k)} \sim Beta(1, 999), k \in \{L, P\}$ and $p^{det}_{(wir_jam,(L,P))} \sim Beta(5, 995)$. Acknowledging the lack of information about the other companies, we assume that $p^{det}_{(i,j)} \sim Beta(6, 994)$, $i \in \{2, 3\}$, $j \in \{AML_at, wir_jam\}$. If either of the two attacks is detected, $cost_{det}$ will be modeled as $\mathcal{U}(100000, 130000)$. We then compute the expected detection costs c_d as $p^{det}_{(1,j,k)} \times cost_{det}$ or $p^{det}_{(i,j)} \times cost_{det}$ depending on the targeted system.

- *Specificities for criminal gang.*

  - *Sensitive information modeling.* Let v_s be the value of a data record with sensitive information and N_s, the count of stolen data records; sales of stolen sensitive information produce $N_s \times v_s$. To account for uncertainty, a data record value is modeled as $v_s \sim \mathcal{U}(0.8 \times 100, 1.2 \times 100)$. To estimate the number of data records stolen in a successful attack, consider $N_s \sim \mathcal{U}(0, ft_{records})$, being $ft_{records} = 660$ with the default portfolio and $ft_{records} = 360$ when the AML is implemented. Recognizing a lack of information on the other companies, assume that $ft_{records} = 600$ when the other companies' vehicles are attacked.
  - *Detection costs and probability.* It is considered that $p^{det}_{(1,AML_at,k)} \sim Beta(3, 997), k \in \{L, P\}$, $p^{det}_{(1,AML_at,(L,P))} \sim Beta(5, 995)$ and $p^{det}_{(i,AML_at)} \sim Beta(4, 996)$. Detection costs are $costs_{det} \sim \mathcal{U}(100000, 130000)$. As with the cyberterrorist group, these values are employed to calculate the expected costs c_d as $p^{det}_{(AML_at,k)} \times costs_{det}$ or $p^{det}_{(i,AML_at)} \times costs_{det}$, $i = 2, 3$.

A.3.2 Attack arrival process and success probability

- *Arrival process.* We model the attack arrival process from potential attackers through Poisson processes with parameters determined using expert knowledge. In particular, for the cyberterrorist group, the number of potential attacks within a one-year period is modeled using a Poisson process with rate 4; for the criminal gang, the Poisson rate is 3.

- *Successful attack probabilities.* We assess the attackers' beliefs that an attack is successful given a certain portfolio. When the cyberterrorist group conducts an AML attack:

  - When no defense is applied $P_A(Y = 1|a^1_{(AML_at,P)}, (0,0,0)) \sim Beta(75,25)$,being $P_A(Y = 1|a^1_{(AML_at,P),}(0,0,0))$ the probability that the attack is successful when the attacker launches an AML attack to block P of the system, given that the portfolio (0,0,0) has been implemented. Similarly, $P_A(Y = 1|a^1_{(AML_at,L)}, (0,0,0)) \sim Beta(75,25)$, $P_A(Y = 1|a^1_{(AML_at,(P,L))}, (0,0,0)) \sim Beta(90,10)$. We assume the probability of a successful attack is higher when both entries are attacked, despite being more easily detected.
  - When the AML protection is implemented $P_A(Y = 1|a^1_{(AML_at,P)}, (0,1,0))$, $P_A(Y = 1|a^1_{(AML_at,L)}, (0,1,0)) \sim Beta(15,85)$,$P_A(Y = 1|a^1_{(AML_at,(P,L))}, (0,1,0)) \sim Beta(10, 90)$.

  As the distributions suggest, implementing a defensive measure not only reduces the PNPs, but would also act as a deterrent, since the attacker's beliefs over a successful attack decreases. Security curves, Fig. 2.4 (Section 2.2.4), are used to determine the distribution parameters. In turn, Table A.5 displays the parameterization when the attacker carries out a wireless jamming learning attack.

Table A.5: Successful attack probability beta parameters of wireless jamming conducted by the cyberterrorist group regarding the targeted block.

Portfolio	P	L	P,L
(0,0,0)	(60,40)	(60,40)	(70,30)
(1,0,0)	(25,75)	(25,75)	(35,65)
(0,0,1)	(25,75)	(25,75)	(35,65)
(1,0,1)	(5,95)	(5,95)	(15,85)

Recognizing a lack of information, we model the attack probability of the cyberterrorist group on a company i's vehicle as $P_A(Y = 1|a^i_j) \sim Beta(1,1), i \in \{2,3\}$, $j \in \{AML_at, wir_jam\}$, under the current status.

Similarly, Table A.6 shows the parameters of the beta distributions modeling the probabilities of a successful AML attack carried out by the criminal gang, given the portfolio implemented regarding the targeted block. As before, we model the probabilities of a successful AML attack on vehicles from other companies as $P_A(Y = 1|a^i_{AML_at}) \sim Beta(1,1), i = 2,3$.

Table A.6: Successful attack probability beta parameters of AML attack conducted by criminal gang regarding the targeted block.

Portfolio	P	L	P,L
(0,0,0)	(65,35)	(65,35)	(80,20)
(0,1,0)	(15,85)	(15,85)	(25,75)

A.3.3 Computation of attack probabilities

To estimate the attack probabilities for each attacker, we conducted 10000 simulations using the previously outlined modeling details to derive vectors $\tau(\mathbf{c})$ and $\Gamma_j^1(\mathbf{c})$, as explained in Section 2.2.3. In particular, for the cyberterrorist group case, we assess the probability of implementing one of the following actions:

- [1-3] AML attack targeting the location block, perception block, or both ($a^1_{AML_at,k}$).
- [4-7]Wireless jamming targeting the location block, perception block, or both ($a^1_{wir_jam,k}$).
- [8-9] AML attack to vehicles of other company ($a^i_{AML_at}, i = 2, 3$).
- [9-10] Wireless jamming targeting vehicles of other company ($a^i_{wir_jam}, i = 2, 3$).

A similar analysis for the criminal gang was performed.

A.3.4 Non-protection probabilities

This section describes the modeling details for parameters related to PNPs when the system is subject to a targeted attack. Note that the selection of PNPs for AML defenses can be accomplished through an analysis using security curves similar to those depicted in Fig. 2.4.

- *Cyberterrorist group.* Table A.7 shows the *Beta* distribution parameters modeling the PNP for each defense when an AML attack is perpetrated. For instance, the parameters for q_p associated with the portfolio $(0, 0, 0)$ indicate that the expected PNP is 72/(72+3) for an AML attack conducted by the cyberterrorist group targeting the perception block. Table A.8 displays the parameters in the event of an attack through wireless jamming.

Table A.7: Cy AML PNP distribution Beta parameters depending on portfolio.

Portfolio	q_P	q_L	$q_{D,P}$	$q_{D,L}$
(0,0,0)	(72,3)	(73,3)	(74,3)	(75,3)
(0,1,0)	(5,195)	(4,195)	(3,195)	(2,195)

Table A.8: Cy wjam PNP distribution beta parameters.

Portfolio	q_P	q_L	$q_{D,P}$	$q_{D,L}$
(0,0,0)	(8,3)	(9,4)	(10,3)	(11,3)
(1,0,0)	(5,60)	(4,60)	(3,60)	(3,60)
(0,0,1)	(5,95)	(4,95)	(4,90)	(3,90)
(1,0,1)	(5,115)	(4,115)	(3,115)	(2,115)

- *Criminal gang.* Table A.9 shows the Beta distribution parameters modeling the PNP for each defense against an AML attack.

Table A.9: Cr AML portfolio dependent PNP beta parameters.

Portfolio	q_P	q_L	$q_{D,P}$	$q_{D,L}$
(0,0,0)	(16,5)	(15,5)	(22,5)	(20,5)
(0,1,0)	(5,245)	(4,245)	(3,245)	(2,245)

A.3.5 Attack impacts

Concerning targeted attacks, we discern when the system is targeted between attacks originating from cyberterrorists or a criminal gang, assuming the first ones to be more skilled.

- *Cyberterrorist group.* Table A.10 displays the parameters of the Gamma and Poisson distributions modeling the impacts of adversarial attacks perpetrated by a cyberterrorist group, both with and without AML protection in place.

Table A.10: Portfolio dependent impact distribution Gamma parameters of AML attack by cyberterrorist.

Portfolio	l_{FI}	l_{ED}			l_D		
		P	L	D	P	L	D
(0,0,0)	(12,2)	(6,2)	(7,2)	(6,3)	(30,2)	(36,2)	(45,2)
(0,1,0)	(2,2)	(1,2)	(1,2)	(1,3)	(2,2)	(3,1)	(4,2)

Similarly, Table A.11 shows the parameters of the distributions modeling the impacts caused by a wireless jamming attack.

Table A.11: Portfolio dependent impact distribution Gamma parameters of wireless jamming conducted by cyberterrorist group.

Portfolio	l_{FI}	l_{ED}			l_D		
		P	L	D	P	L	D
(0,0,0)	(13,2)	(4,2)	(4,2)	(5,2)	(22,2)	(21,2)	(23,2)
(1,0,0)	(6,2)	(3,2)	(3,2)	(4,2)	(16,2)	(15,2)	(17,2)
(0,0,1)	(3,2)	(2,2)	(2,2)	(2,2)	(9,2)	(8,2)	(10,2)
(1,0,1)	(1,1)	(1,2)	(1,2)	(1,2)	(2,2)	(2,2)	(3,2)

- *Criminal gang.* Table A.12 displays the distribution parameters of the impacts of an AML attack carried out by a criminal gang.

Table A.12: Portfolio dependent impact distribution Gamma parameters of AML attack by a criminal gang.

Portfolio	l_{FI}	l_{ED}		
		P	L	D
(0,0,0)	(10,2)	(5,2)	(5,2)	(4,2)
(0,1,0)	(2,2)	(1,2)	(1,2)	(1,3)

Appendix B

Supplementary materials for disinformation campaign case study

This appendix include the parameters describing the disinformation war case study (Section B.1), the chosen augmentation parameters h to increase APS's efficiency (Section B.2), details to select the models estimating $\psi_A(d_1, a_2, \theta_1)$ (Section B.3), $p_D(a_2|d_1, a_1)$ (Section B.4), and $\Psi_A(d_1, a_1)$ (Section B.5), as well as the specification of the distribution approximating $p_D(a_1)$ (Section B.6).

B.1 Modeling details in case study

This section describes the modeling assumptions made by the Defender to numerically solve the disinformation war case study in Section 3.5.

B.1.1 D's problem

Recall D's problem, depicted in the ID in Figure B.1.

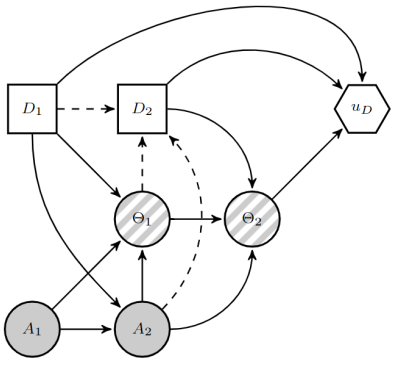


Figure B.1: ID representing D's problem.

Initially, D must determine the proportion of resources $d_1 \in [0, 1]$ available to invest in proactive protection against disinformation. Then, after observing $\theta_1 \in [0, 1]$, D allocates reactive resources d_2. The maximum budgets that D can allocate to D_1 and D_2 are $b_{d_1} = \$400\text{M}$ and $b_{d_2} = \$200\text{M}$, respectively. A allocates resources $a_1 \in [0, 1]$ from its budget to improve its capabilities prior to the attack, assuming a maximum budget of $b_{a_1} = \$380M$. Then, in the second stage, $a_2 \in [0, 1]$ represents the intensity of the attack; the higher the value of a_2, the greater the intensity.

The interaction between D and A's decisions gives rise to random events Θ_1 and Θ_2. The first one is interpreted as the degree of recognition of the attack by D, influencing the effectiveness of d_2. We assume that

$$\Theta_1|d_1, a_1, a_2 \sim Be(\tau_1, \tau_2); \quad \tau_1 = \alpha_{\theta_1}\mu_{\theta_1}(\phi_{\theta_1} + \epsilon), \quad \tau_2 = \alpha_{\theta_1}(1 - \mu_{\theta_1})(\phi_{\theta_1} + \epsilon), \tag{B.1.1}$$

where μ_{θ_1} denotes the mean of Θ_1, $\phi_{\theta_1} = \max\left(\frac{1}{\mu_{\theta_1}}, \frac{1}{1-\mu_{\theta_1}}\right)$, $\alpha_{\theta_1} \geqslant 1$ is a coefficient that helps to control the variance and $\epsilon > 0$ forces the probability density function to be concave. In our problem, we consider $\alpha_{\theta_1} = 2$ and $\mu_{\theta_1} = \min\left(a_2 \times \frac{d_1+t_d}{a_1+t_a}, 1 - \delta\right)$ with $t_d, t_a \geqslant 1$ and sufficiently small $\delta > 0$. The ratio t_d/t_a represents the relative effectiveness of A's investment compared to D's. It ensures that, even when $d_1 = 0$ and $a_1 = 0$, the recognition degree (θ_1) may still be high due to a_2. This follows from our assumption that for large values of a_2, θ_1 will remain significant. As a_2 reaches a substantial portion of the population, D's recognition will also increase. The extent of this effect depends on t_d and t_a. In our case, we set $t_d = 1$ and $t_a = 1.2$. Since $t_a > t_d$, A's investment is more effective than D's when $d_1 \approx a_1$, meaning that D must invest more to counteract A's efforts. Observe that $\mu_{\theta_1} = 0.83$ when $d_1 = 0$, $a_1 = 0$, and $a_2 = 1$. Furthermore, if $a_2 = 0$, we assume that $\theta_1 = 0$, as no attack has been executed. On the other hand, δ is used to avoid $\mu_{\theta_1} = 1$, as this value is not within the support of the Beta distribution.

Θ_2 designates the number of individuals affected by the DC, modelled as

$$\Theta_2|d_2, a_2, \theta_1 \sim Bin(\phi_1, \phi_2); \quad \phi_1 = \lfloor a_2 \times n \rfloor, \quad \phi_2 = \max(0, a_2 - w_{d_2} \times \theta_1 \times d_2) \tag{B.1.2}$$

where $\lfloor x \rfloor$ denotes the floor function applied to x, n are the number of D's citizens and $\omega_{d_2} > 0$ regulates how effective d_2 and θ_1 are regarding A's attack. When $\omega_{d_2} > 1$, reactive measures are more effective: the attack may be stopped and have no consequences even when there is no full recognition ($\theta_1 \neq 1$) and maximum resources are not allocated ($d_2 \neq 1$). Instead, when $\omega_{d_2} < 1$, it means that it may be possible that using all resources for reactive measures ($d_2 = 1$) and full recognition ($\theta_1 = 1$), it would be impossible to stop the attack spread. In the example, $n = 180000$ and $\omega_{d_2} = 0.9$, implying that when $a_2 = 1$, the defenses would be only able to reduce the probability ϕ up to 0.1.

Finally, consider the utility function $u_D(d_1, d_2, \theta_2)$. Define first a value function v_D aggregating monetarily the outcomes

- The minimal security program against disinformation ($d_1 = 0$) entails a cost of $d_1^0 = 15M$. Therefore, the cost of D_1 will be $m_1^D(d_1) = d_1 \times b_{d_1} + d_1^0$.
- The cost of D_2 corresponds directly to the investment, that is $m_2^D(d_2) = d_2 \times b_{d_2}$.
- Let r denote the cost of having an individual in the hospital until its recovery, and c the maximum capacity of D's health system. Once capacity is reached, the health system is unable to accommodate more patients, and infected citizens will need to be transferred to private hospitals for treatment, incurring in additional costs. Thus,

$$m_s^D(\theta_2) = \begin{cases} r \times \theta_2 & \text{if } \theta_2 \leqslant c \\ r \times \theta_2 + (\theta_2 - c) \times l & \text{if } \theta_2 > c, \end{cases}$$

  where l corresponds to the additional costs incurred when a citizen is treated in the private health system. In the example, $r =$ \$0.005 M, $c = 125000$ and $l =$ \$0.02 M.

Then, the value function v_D is

$$v_D(d_1, d_2, \theta_2) = -m_1^D(d_1) - m_2^D(d_2) - m_s^D(\theta_2).$$

For simplicity, assume that, being risk neutral, D's utility is linear in costs,

$$u_D(d_1, d_2, \theta_2) = v_D(d_1, d_2, \theta_2) + \gamma_D,$$

where γ_D is a constant such that $u_D > 0$, to ensure APS requirements. We used $\gamma_D (= 2616)$.

B.1.2 A's problem

Recall the problem faced by A, displayed in the ID in Figure B.2.

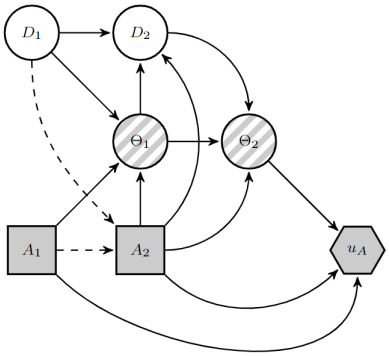


Figure B.2: ID representing A's problem.

To evaluate A's estimates on random events, D assumes that A's assessment will correspond to hers with some additional uncertainty. As relevant data about the scenario becomes available, D could update her estimates in a Bayesian manner.

Considering that Θ_1 is distributed as in (B.1.1), we incorporate uncertainty regarding τ_1 and τ_2 through

$$\Theta_1 | D_1, A_1, A_2 \sim Be(\mathcal{T}_1, \mathcal{T}_2); \quad \mathcal{T}_1 = \kappa \times \tau_1, \mathcal{T}_2 = \kappa \times \tau_2$$

where $\kappa \sim \mathcal{U}\left(\frac{3}{5}, \frac{3}{4}\right)$. As a result, the variance of $P_A(\Theta_1 | D_1, A_1, A_2)$ is bigger than that of $p_D(\theta_1 | d_1, a_1, a_2)$, while μ_{θ_1} remains the same.

Based on (B.1.2), Θ_2 is modelled as

$$\Theta_2 | \Theta_1, D_2, A_2 \sim Bin\big(\phi_1, \Phi_2\big),$$

where we do not consider uncertainty regarding ϕ_1, as it is reasonable to assume that both D and A have a similar estimate of the number of individuals susceptible of being affected by the disinformation. Otherwise, $\Phi_2 \sim \max\Big(0, \min\big(1, \mathcal{U}\big((1 - \delta_{\phi_2}) \times \phi_2, (1 + \delta_{\phi_2}) \times \phi_2\big)\big)\Big)$. Thus, Φ_2 is selected uniformly at random within an interval of width $2 \times \delta_{\phi_2} \times \phi_2$ centered around ϕ_2: D assumes that A's estimates may differ by up to $\delta_{\phi_2}\%$ from its own, with higher δ_{ϕ_2} indicating greater uncertainty. We set $\delta_{\phi_2} = 0.05$, allowing for up to 5% difference from the real value.

We next assess $P_A(D_1 = d_1)$ and $P_A(D_2|D_1, A_1, \Theta_1)$. To prevent recursive loops, we regard both probabilities as based on expert knowledge, allowing for a certain level of uncertainty. We consider that

- For D_1, the defender assumes that A estimates she will spend approximately 70% of b_{d_1} (budget for D_1) on proactive defense. We model $P_A(D_1) \sim Be\big(\kappa \times \mu_{d_1}, \kappa \times (1 - \mu_{d_1})\big)$ where $\mu_{d_1} = 0.7$ and $\kappa \sim \mathcal{U}(7.5, 8)$.

- D supposes that A models her second decision as

$$p_A(d_2|d_1, \theta_1, a_2) \sim Be(v_1, v_2); \quad v_1 = \alpha_{d_2}\mu_{d_2}(\phi_{d_2} + \epsilon), \quad v_2 = \alpha_{d_2}(1 - \mu_{d_2})(\phi_{d_2} + \epsilon)$$

  where $\phi_{d_2} = \max(\frac{1}{\mu_{d_2}}, \frac{1}{1-\mu_{d_2}})$, $\alpha_{d_2} = 2$ is a coefficient to regulate the variance, and $\epsilon > 0$, as before, to force the distribution to be concave. In our problem, $\mu_{d_2} = \min(\frac{\theta_1 a_2}{a_2+(1-d_1)}, 1 - \delta)$, with sufficiently small $\delta > 0$. As before, δ is used to avoid $\mu_{d_2} = 1$, as it is not within the Beta distribution support. With this parametrization, it is assumed that if there is no prior preparation ($d_1 = 0$), but maximum recognition ($\theta_1 = 1$) and maximum attack ($a_2 = 1$), the estimated mean of D_2 would be 0.5. Additionally, for fixed a_2 and d_1, a higher recognition level (θ_1) will result in a higher mean of μ_{d_2}. Similarly, for fixed a_2 and θ_1, greater proactive research (d_1) leads to a higher u_{d_2}. Moreover, for fixed values of d_1 and θ_1, an increase in a_2 also increases μ_{d_2}. Furthermore, assume that $d_2 = 0$ when (a) $a_2 = 0$, as no attack has been produced and, therefore, no defense is necessary; (b) $\theta_1 = 0$, given that D does not perceive any attack and thus does not execute any countermeasure.

  Then, uncertainty is included in v_1 and v_2 through

$$P_A(D_2|D_1, \Theta_1, A_2) \sim Be(\Upsilon_1, \Upsilon_2); \quad \Upsilon_1 = \kappa \times v_1, \quad \Upsilon_2 = \kappa \times v_2$$

  where $\kappa \sim \mathcal{U}\left(\frac{3}{5}, \frac{3}{4}\right)$. As a result, the variance of $P_A(D_2|D_1, \Theta_1, A_2)$ increases compared to $p_A(d_2|d_1, \theta_1, a_2)$, while μ_{d_2} remains the same.

Finally, D assesses A's random utility $U_A(a_1, a_2, \theta_2)$. Consider a scenario where the Defender, through expert judgment, has developed an understanding of the attacker's value function. In particular, assume that

- For A_1, the cost will correspond to his investment in improving its attack without accounting for uncertainty, i.e, $m_1^A(a_1) = a_1 \times b_{a_1}$.

- Without loss of generality, consider that the intensity of an attack on A_2 is linear in costs. The costs to account correspond to the expenses to be paid to disseminate the disinformation, such as paying social media platforms to propagate the content through ads. Therefore, we model the cost of the intensity as: $M_2^A(a_2) = Y_2^A \times a_2$, where $Y_2^A \sim \mathcal{U}\big((1 - \delta_{y_2^A}) \times y_2^A, (1 + \delta_{y_2^A}) \times y_2^A\big)$, with $y_2^A = \$300M$ and $\delta_{y_2^A} = 0.05$.

- Suppose the assessment of A on θ_2 would be similar to D's, having A uncertainty about D on the cost of treating a patient, r_1^A, and the capacity of D's health system, c^A. D assumes that A considers treating a person costs $r_1^A = \$0.005M$; to account for uncertainty, it will be modeled as $R_1 \sim \mathcal{U}\big((1 - \delta_{r_1^A}) \times r_1^A, (1 + \delta_{r_1^A}) \times r_1^A\big)$. On the other hand, D estimates that A believes that the health system has the capacity to treat up to $c^A = 125000$ patients, and the extra costs per patient to be transferred to the private system to be $l^A = \$0.02M$.

Similarly, to account for uncertainty, the capacity of the health system and the extra costs of transferring a patient will be modeled as $C^A \sim \mathcal{U}\big((1-\delta_{c^A})\times c^A, (1+\delta_{c^A})\times c^A\big)$ and $L \sim \mathcal{U}\big((1-\delta_l^A)\times l^A, (1+\delta_l^A)\times l^A\big)$, respectively. Therefore, $M_s^A(\theta_2)$ is modelled as

$$M_s^A(\theta_2) = \begin{cases} R_1 \times \theta_2 & \text{if } \theta_2 \leqslant C^A \\ R_1 \times \theta_2 + (\theta_2 - C^A) \times L & \text{if } \theta_2 > C^A \end{cases}$$

In the case study, $\delta_{r_1^A} = 0.05$, $\delta_{c^A} = 0.05$ and $\delta_{l^A} = 0.01$, assuming that A may have accurate estimates based from its intelligence services. Combining the previous assumptions,

$$V_A(a_1, a_2, \theta_2) = -m_1^A(a_1) - M_2^A(a_2) + M_s^A(\theta_2)$$

Finally, for simplicity, assume that A's utility is linear in costs

$$U_A(a_1, a_2, \theta_2) = V_A(a_1, a_2, \theta_2) + \gamma_A,$$

where γ_A is a constant so that $U_A > 0$, to guarantee applicability of APS. We used $\gamma_A (= 1280)$.

B.2 Augmentation parameters H used in D's and A's decision stages

To enhance the efficiency of APS in all decision stages, for both the Defender and the Attacker, we used the approach described in Section 3.4.4, which involves an augmentation parameter H. Table B.1 displays the parameter H used at each decision stage.

Table B.1: Augmentation parameter H used in the Defender's and Attacker's decision stages.

Decision stage	D's second	A's second	A's first	D's first
H	40	80	120	20

To determine H for each stage, several values were tested heuristically until convergence was achieved. The objective was to identify a value of H that ensured convergence while avoiding unnecessarily high computational cost to obtain the solution.

B.3 Approximating $\psi_D(d_1, a_2, \theta_1)$ from Defender's second stage decision

Following the process described in Section 3.5.2, we select the best MLP from among those with $n \in \{1, 2, 3\}$ hidden layers, $t \in \{16, 32, 64\}$ neurons per hidden layer, and ReLU activation functions, using $\mathcal{D}_{\psi_{D_2}}$. We evaluate the performance of each MLP configuration using the mean absolute error (MAE) and root mean square error (RMSE). Table B.2 presents the mean and standard error of the CV results for the five best models over 10 iterations. Based on these results, we select the model with three hidden layers containing 32, 64, and 16 neurons, respectively.

After selecting the architecture, we train an MLP model on the full training dataset and evaluate it on the test dataset, obtaining an MAE of 9.3 and an RMSE of 12.9. Given the scale of $\psi_D(d_1, a_2, \theta_1)$, with a mean of 2143.6 and a median of 2206.0 in $\{\psi_D(d_1, a_2, \theta_1)\}_{j=1}^J$, these metrics indicate that the proposed MLP effectively approximates the target expected utility. Finally, we train the MLP with the selected hyperparameter configuration over the entire dataset $\mathcal{D}_{\psi_{D_2}}$ to obtain $\hat{\psi}_D(d_1, a_2, \theta_1)$, used for the attacker's first-stage decision.

Table B.2: Mean and standard error of 5-fold CV results over 10 iterations for best five MLP architectures. Each element of the architecture represents a hidden layer, with the number indicating the neurons.

Architecture	**[32,64,16]**	[32,64,32]	[32,64,64]	[16,64,64]	[64,64,32]
MAE	**10.2±0.7**	11.3±0.4	11.7±0.8	11.6±0.6	12.8±1.4
RSME	**12.5±0.8**	13.2±0.6	13.3±0.8	14.1±0.7	14.5±1.4

B.4 Approximating $p_D(a_2|d_1, a_1)$ from Attacker's second stage decision

We approximate $A_2^*(d_1, a_1)$ for each (d_1, a_1) with a mixture of Beta distributions (MoB), with at most two components ($N \in \{1, 2\}$) described by the parameters $\{(w_i, \alpha_i, \beta_i)\}_{i=1}^N$. As Section 3.5.2 describes, we use an MLP that outputs $\{(w_i, \alpha_i, \beta_i)\}_{i=1}^N$ for each (d_1, a_1), with a softmax activation applied to the neurons producing $\{w_i\}_{i=2}^N$ and a softplus activation applied to those producing $\{\alpha_i, \beta_i\}_{i=1}^N$. The MLP configuration is selected following the procedure outlined in Section 3.5.2, from among models with $N \in \{1, 2\}$ mixture components, $n \in \{1, 2, 3\}$ hidden layers, $t \in \{16, 32, 64\}$ neurons per layer, and ReLU activations in the hidden layers. Table B.3 reports the mean and standard error of the CV results for the five best-performing MLP configurations across 10 iterations. Based on these results, we select the model with three hidden layers of 64 neurons each, whose outputs define a MoB with $N = 2$ components.

Table B.3: Mean and standard error of the 5-fold CV results over 10 iterations for best five MLP architectures (Arch.) and number of MoB components (# Comp.) used to estimate $A_2^*(d_1, a_1)$.

Arch.;# Comp.	**[64,64,64];2**	[32,64,64];2	[64,64,32];2	[64,32,64];2	[32,64,32];2
NLL	**-31.8±0.1**	-31.7±0.1	-31.7±0.2	-31.5±0.2	-31.4±0.2

Once the configuration is selected, the MLP is trained on the complete training dataset and evaluated on the test set, yielding an NLL of -39.9. Figure B.3 (blue) displays $A_2^*(d_1, a_1)$ for some (d_1, a_1) in the test set, while Figure B.3 (red) shows the MoB approximation for the same (d_1, a_1). The final MLP, trained on the full dataset $\mathcal{D}_{A_2}$, is used in the Defender's first-stage decision process.

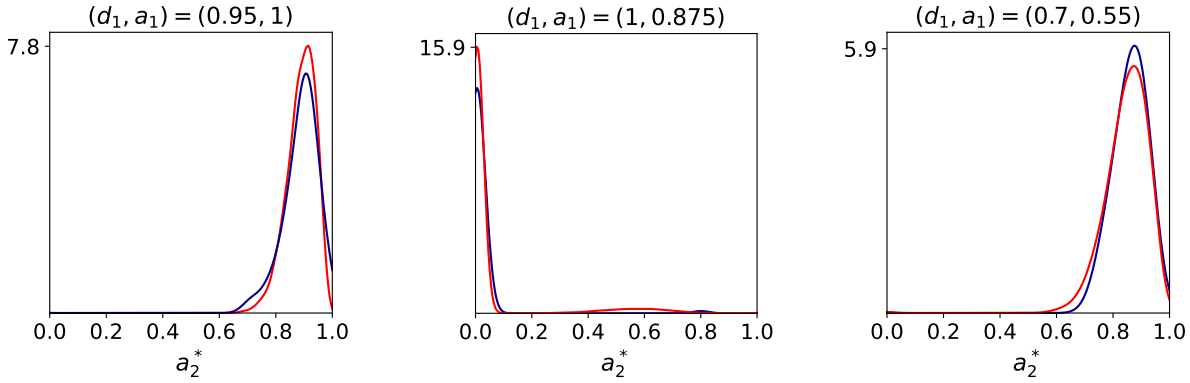


Figure B.3: Empirical distribution of $A_2^*(d_1, a_1)$ (blue) against MoB approximation (red).

B.5 Approximating $\Psi_A(d_1, a_1)$ from Attacker's second stage decision

To approximate $\Psi_A(d_1, a_1)$, we use an MLP that defines a mixture of Weibull distributions (MoW) with at most $N = 2$ components. Specifically, the MLP output consists of the parameters $\{w_i, \lambda_i, \alpha_i\}_{i=1}^N$ for an input (d_1, a_1), characterizing the MoW. We follow the procedure described in Section 3.5.2 to select the best MLP architecture from among the models with $N \in \{1, 2\}$ components, $n \in \{1, 2, 3\}$ hidden layers, and $t \in \{16, 32, 64\}$ neurons per layer. Additionally, all considered architectures use ReLU activations in the hidden layers, a softmax activation for $\{w_i\}_{i=1}^N$, and a softplus activation for $\{\lambda_i, \alpha_i\}_{i=1}^N$. Table B.4 reports the mean and standard error of the CV results for the five best-performing MLP models over 10 iterations. In light of the results, we select the MLP with $N = 2$ and three hidden layers of 64 neurons each, obtaining a NLL of -31.7 on the test set.

Table B.4: Mean and standard error of the 5-fold CV results over 10 iterations for the top five MLP architectures (Arch.) and number of MoW components (# Comp.) used to estimate $\Psi_A(d_1, a_1)$.

Arch.;# Comp.	**[64,64,64];2**	[64,32,64];2	[64,64,64];1	[64,64,32];2	[64,64];2
NLL	**-25.7±0.1**	-25.1±0.1	-24.8±0.0	-24.7±0.1	-24.7±0.1

Figure B.4 shows the MoW approximation (in red) against the probability density function density of the empirical $\Psi_A(d_1, a_1)$ (in blue) for some (d_1, a_1). Finally, we train the MLP with the selected hyperparameters on the full dataset $\mathcal{D}_{\Psi_{A_2}}$ for use in the Attacker's first-stage.

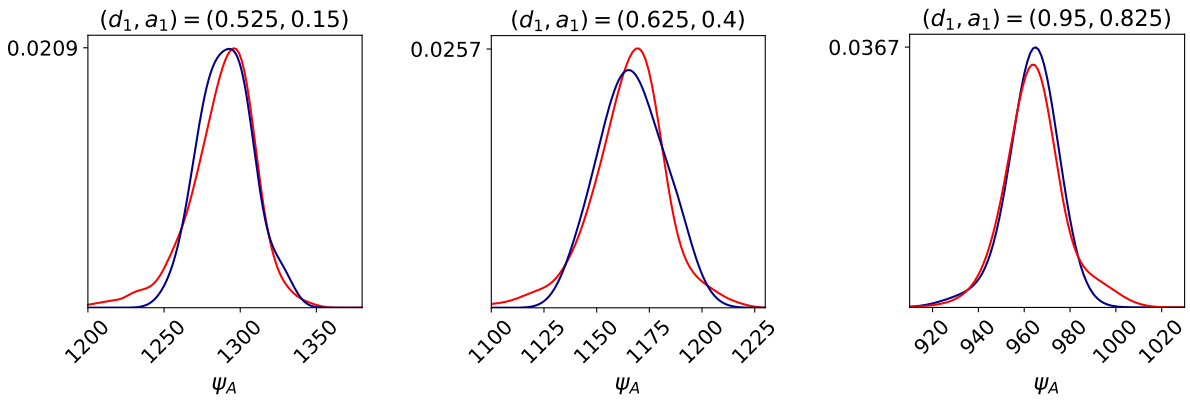


Figure B.4: Density of empirical $\Psi_A(d_1, a_1)$ (blue) and MoW model approximation (red).

B.6 Approximating $p_D(a_1)$ from Attacker's first stage decision

Given the shape of the distribution A_1^* (Figure B.5 blue), and the fact that it is bounded on $[0, 1]$, we use Maximum Likelihood Estimation (Hastie et al., 2017) to fit a two-component Beta mixture model. The parameters of the mixture are $\big(\{\alpha_i\}, \{\beta_i\}, \{w_i\}\big)_{i=1}^{2} = \big((100.9, 2.3), (1.6, 213.4), (0.1, 0.9)\big)$. Figure B.5 (red) displays 10000 samples of the fitted distribution compared to the empirical distribution of A_1^* (blue), showing a good approximation.

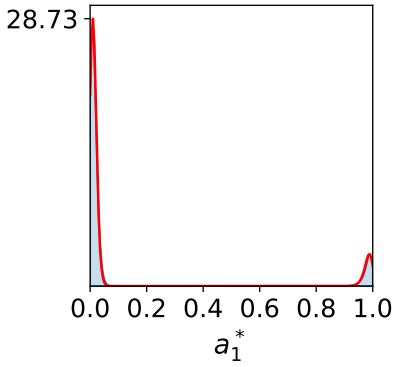


Figure B.5: Comparison between empirical distribution (blue) of A_1^* and its approximation (red).

Appendix C

Implementation Details of the R&S Heuristic

This appendix provides implementation details of the R&S heuristics utilized in Sections 4.5.2 and 4.5.3. We discuss the functional approximation $\hat{\mu}(z^n|\theta^n)$, the associated state variables (i.e., θ^n), the system model (i.e., S^M), and the optimization routines used to select z^n given some parameterization of $\hat{\mu}(z^n|\theta^n)$. Associated code implementing the procedures discussed herein is available at the code repository for the chapter.

As a functional approximation of an attack's value, we use a fully connected NN. Different architectures were tested and the superlative was selected, i.e., see Appendix C.3. The system state at each iteration n captures the current parameterization of this NN, e.g., the arc weights. The updated state variables (i.e., θ^{n+1}) are identified using a stochastic-gradient-descent algorithm as the system model. In particular, the Adam optimizer is utilized. The performance of different learning rates is tested as well; see Appendix C.3 for additional details.

To iteratively select z^n, the R&S heuristic uses an ε-greedy policy to encourage exploration. Within each iteration of R&S, the greedy action that maximizes the objective function under our current belief state is of foremost interest. This requires solving a non-linear, integer optimization problem. Unfortunately, given the high cardinality of the attack space (i.e., $\mathcal{Z}$), solving this optimization via complete ennumeration is infeasible; other canonical branch-and-bound techniques are likely to struggle as well. Therefore, we compare the performance of two meta-heuristics to approximate this solution: Monte Carlo Tree Search (MCTS) and Simulated Annealing (SA). Implementation details for both algorithms are provided subsequently in Appendices C.1 and C.2. Hyperparameter tuning is discussed in Appendix C.3.

C.1 Monte Carlo Tree Search

To find the greedy action using MCTS, we first reformulate this optimization problem as an undiscounted, sequential-decision problem wherein the greedy attack is constructed iteratively by time period. With slight recycling of notation, this problem is formally defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{E}, \mathcal{R})$ wherein $\mathcal{S}$ is the set of all partially and fully filled attack vectors, $\mathcal{A}$ is the set of feasible actions at each $s \in \mathcal{S}$, $\mathcal{E}$ is a map from $\mathcal{S} \to \mathcal{S}$, and $\mathcal{R}$ is the set of instantaneous rewards at each $s \in \mathcal{S}$. At the beginning of each episode, we start with an empty attack vector (i.e., s_0). An episode concludes after the attack vector is fully specified. This is accomplished by selecting an emission for each time t from $\mathcal{A}(s_t)$ whereby s_t captures all emissions inserted into the attack up to this time. The transition function $\mathcal{E}$ deterministically maps a state-action pair (s_t, a_t) into

the next state s_{t+1} by inserting the selected emission into the attack vector. The set of state rewards, $\mathcal{R}$, varies depending upon whether the state is complete or incomplete, i.e., whether emissions have or have not been selected for every $t \in \mathcal{T}$, respectively. The reward associated with any incomplete state is zero. However, the reward for complete states (i.e., fully specified attack vectors) corresponds to the evaluation of $\hat{\mu}(s|\theta^n)$.

Using this foundation, our MCTS algorithm constructs a tree comprised of all possible attack vectors. Its nodes correspond to some $\mathcal{S}' \subseteq \mathcal{S}$. At each iteration of the algorithm, the tree is traversed by selecting actions via a tree policy until a node with a child outside of the partially constructed tree is reached. If such a node is reached, actions are henceforth selected at random until a complete state is achieved. After reaching a complete state, its reward is computed and used to update the value of nodes that have been traversed in the tree such that better nodes are more likely to be selected in the future.

More specifically, given a partially explored tree $\mathcal{G}$, each MCTS iteration consists of four distinct steps:

1. *Selection.* Starting from the root node, (i.e., the empty attack vector, s_0), actions are selected using the tree policy until a node with some child outside the tree is reached. In our implementation, we use the UCT selection criterion as the tree policy's basis. That is, at state s_t, action a_t (leading to node $\mathcal{E}(s_t, a_t)$) is identified using

$$\pi^{\mathcal{G}}(s_t) = \underset{a \in \mathcal{A}(s_t)}{\operatorname{argmax}} Q(s_t, a) + c \cdot \sqrt{\frac{2 \log N(s_t)}{N(s_t, a)}}$$

   where $Q(s_t, a)$ is a MC estimate of the state-action value, $N(s_t)$ is the number of visits to the parent node, $N(s_t, a)$ is the number of times action a has been taken at node s_t and c is an exploration parameter. In our implementation, we fix $c = 0.5$.

2. *Expansion.* If a leaf node is reached in the selection step, it is appended to the tree and the simulation step begins.

3. *Simulation.* Actions outside the tree are selected uniformly at random from the available actions until a complete state has been reached.

4. *Backpropagation.* Upon reaching a complete state, its reward R is evaluated by invoking $\hat{\mu}(s|\theta^n)$. This is used to update the value of the leaf node reached in the selection step as well as each of its parents. For every (s', a) along the traversed path in the tree, the following updates are performed:

$$\begin{aligned} N(s', a) &\leftarrow N(s', a) + 1 \\ Q(s', a) &\leftarrow Q(s', a) + \frac{R - Q(s', a)}{N(s', a)} \end{aligned}$$

In our implementation of the R&S heuristic, MCTS is used to approximate the greedy policy in each iteration of R&S (i.e., each time Step 2 is reached in Algorithm 16). In particular, we perform $1,000$ MCTS iterations and record the best complete state reached as the approximate greedy action.

C.2 Simulated Annealing

The second meta-heuristic used to find the action maximizing $\hat{\mu}(z^n|\theta^n)$ is SA (Kirkpatrick et al., 1983). SA generates a Markov chain in the space of possible attacks, whose stationary distribution is proportional to $\exp(\hat{\mu}(z^n|\theta^n)/T_e)$, where T_e is gradually decreased to 0 according to some pre-specified annealing schedule. We use Gibbs sampling to generate such a Markov chain, sequentially sampling from full conditional of the stationary distribution for each z_t^n conditioned on z_{-t}^n. T_e is reduced according to the exponential decay annealing schedule suggested by Spears, 1993. Therefore, at the j^{th} iteration of SA, we have $T_e = \exp(-l \cdot j/\mathcal{T})$, where l is set to 5.

C.3 Hyperparameter Tuning

This appendix explains how hyperparameter tuning was performed for the two R&S variants. Specifically, we varied the following hyperparameters: the architecture of the MLP NN (i.e., the number of neurons in the two layers), the number of SA or MCTS iterations used to find the greedy action, the learning rate, and ε. Tables C.1 and C.2 lists the tested hyperparameter combinations.

Table C.1: Hyperparameter Combinations for the R&S-A Algorithm.

Combination	MLP Architecture	Iterations	Learning rate	ε
1	{16; 8}	100	0.005	0.05
2	{32; 16}	10	0.005	0.05
3	{16; 8}	100	0.1	0.005
4	{32; 16}	10	0.1	0.005
5	{64; 64}	100	0.005	0.05
6	{64; 64}	100	0.1	0.05

Table C.2: Hyperparameter Combinations for the R&S-B Algorithm.

Combination	MLP Architecture	Iterations	Learning rate	ε
1	{16; 8}	50	0.005	0.05
2	{32; 16}	10	0.005	0.05
3	{16; 8}	50	0.1	0.005
4	{32; 16}	10	0.1	0.005
5	{64; 64}	100	0.005	0.05
6	{64; 64}	100	0.1	0.05

An extensive grid search on the hyperparameters was not performed; instead, a simple search procedure was conducted to (1) examine how the algorithms behave under different hyperparameters and (2) identify settings that perform well on the examined instances. To do so, we replicated the testing procedure described in Sections 4.5.2 and 4.5.3 for each combination. Each algorithm-and-hyperparameter combination was tested 10 times for run times of $\{15, 30, 45, ..., 1200\}$ seconds on each problem instance. Mean expected utility values plus/minus two standard deviations were recorded for each run time across all 10 repetitions. Algorithm

performance was judged by their convergence time, solution quality, and relative balance of these characteristics. A qualitative assessment of these measures was utilized for model selection.

Tables C.3 and C.4 indicate the superlative hyperparameter combinations for each problem-and-design-point pair. The indicated hyperparameter setting is leveraged within Sections 4.5.2 and 4.5.3, respectively. Notably, no single combination is dominant across all instances, but some combinations were always dominated by another. For example, combination four and three of the R&S-A and -B algorithms, respectively, are dominated in the Section 4.5.2 instances. Likewise, combination six is dominated across all problem-and-design-point pairs. Analogous patterns are apparent in the Section 4.5.3 testing as well.

Table C.3: Superlative Hyperparameter Combination for Section Structural Testing

Problem	Algorithm	Design Point			
		1	2	3	4
State-Attraction	R&S-A	2	3	2	2
	R&S-B	2	1	2	2
State-Repulsion	R&S-A	2	3	2	2
	R&S-B	2	1	2	5
Distribution-Disruption	R&S-A	2	5	2	1
	R&S-B	2	5	2	5
Path-Attraction	R&S-A	2	1	1	1
	R&S-B	4	1	2	2

Table C.4: Superlative Hyperparameter Combination for HMM Uncertainty Testing

Problem	Algorithm	Design Point			
		1	2	3	4
State-Attraction	R&S-A	3	3	5	5
	R&S-B	5	1	2	1
State-Repulsion	R&S-A	5	5	5	5
	R&S-B	2	2	2	2
Distribution-Disruption	R&S-A	5	2	5	2
	R&S-B	2	2	2	2
Path-Attraction	R&S-A	2	2	1	1
	R&S-B	4	4	4	4

Bibliography

Agarwala, G, Latorre, A, Raffel, S, Mehta, R, Zhao, J, Nurullayev, A, Clark, B, and Tang, R (2020). *Supervisory expectations and sound model risk management practices for AI and ML*. Ernst & Young. URL: https://www.ey.com/content/dam/ey-unified-site/ey-com/en-us/insights/banking-capital-markets/documents/ey-mrm-ai-ml.pdf.

Agence Nationale de la Sécurité des Systèmes d'Information (2024). *Expression des Besoins et Identification des Objectifs de Sécurité (EBIOS) Risk Manager*. Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI). URL: https://cyber.gouv.fr/la-methode-ebios-risk-manager.

Akpan, I. J., Kobara, Y. M., Owolabi, J., Akpan, A. A., and Offodile, O. F. (2025). "Conversational and generative artificial intelligence and human–chatbot interaction in education and research". In: *International Transactions in Operational Research* 32.3, pp. 1251–1281.

Ala-Pietilä, P., Bonnet, Y., Bergmann, U., Bielikova, M., Bonefeld-Dahl, C., Bauer, W., Bouarfa, L., Chatila, R., Coeckelbergh, M., Dignum, V., et al. (2020). *The assessment list for trustworthy artificial intelligence (ALTAI)*. European Commission. URL: https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment.

Amutio, M., Candau, J., and Mañas, J. (2012). *MAGERIT – Versión 3.0.Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información. Libro I-Método*. URL: https://pilar.ccn-cert.cni.es/docman/documentos/1-magerit-v3-libro-i-metodo/file.

Ardia, D., Ringel, E., Ekstrand, V. S., and Fox, A. (2020). "Addressing the Decline of Local News, Rise of Platforms, and Spread of Mis- and Disinformation Online: A Summary of Current Research and Policy Proposals". In: *UNC Legal Studies Research Paper*.

Arroyo, D., Degli-Esposti, S., Gómez-Espés, A., Palmero-Muñoz, S., and Pérez-Miguel, L. (2023). "On the Design of a Misinformation Widget (MsW) Against Cloaked Science". In: *Network and System Security. NSS 2023. Lecture Notes in Computer Science, Springer*.

Aven, T. (2019). *The science of risk analysis: Foundation and practice*. Routledge.

Banda, J. M., Tekumalla, R., Wang, G., Yu, J., Liu, T., Ding, Y., and Chowell, G. (2020). *A Twitter Dataset of 150+ million tweets related to COVID-19 for open research*. Zenodo. URL: https://zenodo.org/record/3738018#.YCpnvmhKg2w(Accessed:18-10-2020).

Banks, D. L., Ríos, J., and Insua, D. R. (2015). *Adversarial Risk Analysis*. Francis & Taylor.

Banks, D., Gallego, V., Naveiro, R., and Ríos Insua, D. (2022). "Adversarial risk analysis: An overview". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 14.1, e1530.

Barman, D., Guo, Z., and Conlan, O. (2024). "The Dark Side of Language Models: Exploring the Potential of LLMs in Multimedia Disinformation Generation and Dissemination". In: *Machine Learning with Applications* 16, 100545.

Bechhoefer, J. (2015). "Hidden Markov models for stochastic thermodynamics". In: *New Journal of Physics* 17.7, 075003.

Bengio, Y., Mindermann, S., Privitera, D., Besiroglu, T., Bommasani, R., Casper, S., Choi, Y., Fox, P., Garfinkel, B., Goldfarb, D., et al. (2025). *International AI Safety Report*. URL: https://arxiv.org/pdf/2501.17805.

Bielza, C., Müller, P., and Insua, D. R. (1999). "Decision analysis by augmented probability simulation". In: *Management Science* 45.7, pp. 995–1007.

Biggio, B. and Roli, F. (2018). "Wild patterns: Ten years after the rise of adversarial machine learning". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2154–2156.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bontcheva, K., Posetti, J., Teyssou, D., Meyer, T., Gregory, S., Hanot, C., and Maynard, D. (2020). *Balancing act: countering digital disinformation while respecting freedom of expression: Broadband Commission research report on 'Freedom of Expression and Addressing Disinformation on the Internet'*. UNESCO. URL: https://unesdoc.unesco.org/ark:/48223/pf0000379015.

Breve, B., Caruccio, L., Cirillo, S., Deufemia, V., and Polese, G. (2024). "Analyzing the worldwide perception of the Russia-Ukraine conflict through Twitter". In: *Journal of Big Data* 11.1,76.

Brown, G., Carlyle, M., Salmerón, J., and Wood, K. (2006). "Defending critical infrastructure". In: *Interfaces* 36.6, pp. 530–544.

CCTA (2005). *CCTA Risk Analysis and Management Method (CRAMM) Version 5.0/5.1*. Central Computer and Telecommunications Agency (CCTA).

CRFM (2021). *On the Opportunities and Risks of Foundation Models*. Stanford University. URL: https://crfm.stanford.edu/report.html.

Caballero, W. N., Kline, A. G., and Lunday, B. J. (2018). "Challenges and Solutions with Exponentiation Constraints using Decision Variables via the BARON Commercial Solver". In: *2018 IISE Annual Conference Proceedings*, pp. 1331–1336.

Caballero, W. N., Rios Insua, D., and Banks, D. (2023). "Decision support issues in automated driving systems". In: *International Transactions in Operational Research* 30.3, pp. 1216–1244.

Cameron, T. A. (2010). "Euthanizing the value of a statistical life". In: *Review of Environmental Economics and Policy* 4.2, pp. 161–178.

Cha, S. (2007). "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions". In: *International Journal of Mathematical Models and Methods in Applied Sciences* 1.4, pp. 300–307.

Chacon, J. (2020). "The Modal Age of Statistics". In: *International Statistical Review* 88.1, pp. 122–141.

Chen, E., Lerman, K., and Ferrara, E. (2020). "Tracking social media discourse about the COVID-19 pandemic: Development of a public coronavirus Twitter data set". In: *JMIR Public Health and Surveillance* 6.2, e19273.

Chung, K. L. (2001). *A Course in Probability Theory*. Academic Press.

Cihon, P. (2019). *Standards for AI Governance: International Standards to Enable Global Coordination in AI Research & Development*. Future of Humanity Institute, University of Oxford. URL: https://cdn.governance.ai/Standards_-FHI-Technical-Report.pdf.

Cinà, A. E., Grosse, K., Demontis, A., Vascon, S., Zellinger, W., Moser, B. A., Oprea, A., Biggio, B., Pelillo, M., and Roli, F. (2023). "Wild patterns reloaded: A survey of machine learning security against training data poisoning". In: *ACM Computing Surveys* 55.13s, pp. 1–39.

Clifford, T., Shumailov, I., Zhao, Y., Anderson, R., and Mullins, R. (2022). "*ImpNet: Imperceptible and blackbox-undetectable backdoors in compiled neural networks*". In: ArXiv. URL: https://arxiv.org/pdf/2210.00108.

Cloud Security Alliance (2024). *Cloud Controls Matrix v4.0*. Cloud Security Alliance (CSA). URL: https://cloudsecurityalliance.org/artifacts/cloud-controls-matrix-v4.

Coffin, M. and Saltzman, M. J. (2000). "Statistical analysis of computational tests of algorithms and heuristics". In: *INFORMS Journal on Computing* 12.1, pp. 24–44.

Corrales, D., Santos-Lozano, A., López-Ortiz, S., Lucia, A., and Insua, D. R. (2024). "Colorectal cancer risk mapping through Bayesian networks". In: *Computer Methods and Programs in Biomedicine* 257,108407.

Couce-Vieira, A., Insua, D. R., and Kosgodagan, A. (2020). "Assessing and forecasting cybersecurity impacts". In: *Decision Analysis* 17.4, pp. 356–374.

Cox, L. A. (2008). "What's Wrong with Risk Matrices?" In: *Risk Analysis* 28.2, pp. 497–512.

Cresci, S. (2020). "A decade of social bot detection". In: *Communications of the ACM* 63.10, pp. 72–83.

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. (2017a). "Social fingerprinting: Detection of spambot groups through DNA-inspired behavioral modeling". In: *IEEE Transactions on Dependable and Secure Computing* 15.4, pp. 561–576.

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M. (2017b). "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race". In: *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 963–972.

Cresci, S., Lillo, F., Regoli, D., Tardelli, S., and Tesconi, M. (2018). "$FAKE: Evidence of spam and bot activity in stock microblogs on Twitter". In: *Proceedings of the International AAAI Conference on Web and Social Media* 12.1.

Cresci, S., Lillo, F., Regoli, D., Tardelli, S., and Tesconi, M. (2019). "Cashtag piggybacking: Uncovering spam and bot activity in stock microblogs on Twitter". In: *ACM Transactions on the Web (TWEB)* 13.2, pp. 1–27.

Crouse, M. S., Nowak, R. D., and Baraniuk, R. G. (1998). "Wavelet-based statistical signal processing using hidden Markov models". In: *IEEE Transactions on signal processing* 46.4, pp. 886–902.

Dalvi, N., Domingos, P., Sanghai, S., and Verma, D. (2004). "Adversarial classification". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 99–108.

Documenting the Now (2020). *Twarc*. URL: https://github.com/DocNow/twarc(Accessed:18-10-2020).

Doumpos, M., Zopounidis, C., Gounopoulos, D., Platanakis, E., and Zhang, W. (2023). "Operational research and artificial intelligence methods in banking". In: *European Journal of Operational Research* 306.1, pp. 1–16.

ENISA (2022). *ENISA Threat Landscape 2022*. URL: https://www.enisa.europa.eu/sites/default/files/publications/ENISA%20Threat%20Landscape%202022.pdf.

Ehrgott, M. (2005). *Multicriteria optimization*. Springer.

Ekin, T., Naveiro, R., Insua, D. R., and Torres-Barrán, A. (2023). "Augmented probability simulation methods for sequential games". In: *European Journal of Operational Research* 306.1, pp. 418–430.

Ernst, J. and Kellis, M. (2012). "ChromHMM: Automating chromatin-state discovery and characterization". In: *Nature methods* 9.3, pp. 215–216.

European Parliament (2024). "*Artificial Intelligence Act.*" In: *Official Journal of the European Union*. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L_202401689.

European Parliament, Council, and Commission (2012). "*Charter of Fundamental Rights of the EU. OJ C 326, 26.10.2012* ". In: *Official Journal of the European Journal*. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:C2012/326/02.

FBI (2022). *Internet crime report 2022. U.S. Department of Justice*. URL: https://www.ic3.gov/Media/PDF/AnnualReport/2022_IC3Report.pdf.

Ferrara, E. (2017). "Disinformation and social bot operations in the run up to the 2017 French presidential election". In: *First Monday* 22.8.

Freeman III, A. M., Herriges, J. A., and Kling, C. L. (2014). *The Measurement of Environmental and Resource Values*. Routledge.

French, S. and Insua, D. R. (2000). *Statistical Decision Theory*. Wiley.

Gales, M. and Young, S. (2008). "The application of hidden Markov models in speech recognition". In: *Foundations and Trends in Signal Processing* 1.3, pp. 195–304.

Gallego, V. and Insua, D. R. (2022). "Current advances in neural networks". In: *Annual Review of Statistics and Its Application* 9, pp. 197–222.

Gallego, V., Naveiro, R., and Insua, D. R. (2019). "Reinforcement learning under threats". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01, pp. 9939–9940.

Gallego, V., Naveiro, R., Redondo, A., Insua, D. R., and Ruggeri, F. (2024). "Protecting Classifiers From Attacks. A Bayesian Approach". In: *Statistical Science* 39.3, pp. 449–468.

García, E. G., Varas, P., González-Naranjo, P., Ulzurrun, E., Marcos-Ayuso, G., Pérez, C., Páez, J. A., Insua, D. R., Santana, S. R., and Campillo, N. E. (2025). "AI-Driven De Novo Design and Development of Nontoxic DYRK1A Inhibitors". In: *Journal of Medicinal Chemistry* 68.10, pp. 10346–10364.

Gil, C. and Parra-Arnau, J. (2019). "An Adversarial-Risk-Analysis Approach to Counterterrorist Online Surveillance". In: *Sensors* 19.3 , 480.

Gilani, Z., Farahbakhsh, R., Tyson, G., Wang, L., and Crowcroft, J. (2017). "Of bots and humans (on Twitter)". In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 349–354.

González-Ortega, J., Insua, D. R., and Cano, J. (2019). "Adversarial risk analysis for bi-agent influence diagrams: An algorithmic approach". In: *European Journal of Operational Research* 273.3, pp. 1085–1096.

González-Ortega, J., Radovic, V., and Insua, D. R. (2018). "Utility elicitation". In: *The Science and art of structuring judgment*, pp. 241–264.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). "Explaining and harnessing adversarial examples". In: ArXiv. URL: https://arxiv.org/pdf/1412.6572.

Gordillo, J. and Conde, E. (2007). "An HMM for detecting spam mail". In: *Expert Systems with Applications* 33, pp. 667–682.

Gu, Y., Goez, J. C., Guajardo, M., and Wallace, S. W. (2021). "Autonomous vessels: state of the art and potential opportunities in logistics". In: *International Transactions in Operational Research* 28.4, pp. 1706–1739.

Halder, S., Ghosal, A., and Conti, M. (2020). "Secure over-the-air software updates in connected vehicles: A survey". In: *Computer Networks* 178, 107343.

Ham, J. V. D. (2021). "Toward a better understanding of "cybersecurity"". In: *Digital Threats: Research and Practice* 2.3, pp. 1–3.

Hanea, A. M., Nane, G. F., Bedford, T., and French, S., eds. (2021). *Expert Judgement in Risk and Decision Analysis*. Springer.

Hargreaves-Heap, S. and Varoufakis, Y. (2004). *Game theory: A critical introduction*. Routledge.

Hastie, T., Tibshirani, R., and Friedman, J. (2017). *The elements of statistical learning: data mining, inference, and prediction*. Springer.

Hausken, K. (2024). "Fifty years of operations research in defense". In: *European Journal of Operational Research* 318.2, pp. 355–368.

Helberger, N. and Diakopoulos, N. (2023). "ChatGPT and the AI Act". In: *Internet Policy Review* 12.1.

Hutto, C. and Gilbert, E. (2014). "VADER: A parsimonious rule-based model for sentiment analysis of social media text". In: *Proceedings of the international AAAI conference on web and social media* 8.1, pp. 216–225.

ISO (2022). *Trustworthiness – Vocabulary (ISO Stand. N. TS 5723:2022)*. URL: https://www.iso.org/standard/81608.html.

ISO (2023a). *Information technology — AI — Guidance on risk management (ISO Stand. N. TR 23894:2023)*. URL: https://www.iso.org/standard/77304.html.

ISO (2023b). *Road vehicles — Software update engineering (ISO Stand. N. TR 24089:2023)*. URL: https://www.iso.org/standard/77796.html.

Insua, D. R., Couce-Vieira, A., Rubio, J. A., Pieters, W., Labunets, K., and G. Rasines, D. (2021a). "An adversarial risk analysis framework for cybersecurity". In: *Risk Analysis* 41.1, pp. 16–36.

Insua, D. R., Baylon, C., and Vila, J. (2021b). *Security Risk Models for Cyber Insurance*. CRC Press.

International Organization for Standardization (2022). *ISO/IEC 27001 – Information Security, Cybersecurity, and Privacy Protection – Information Security Management Systems - Requirements*. URL: https://www.iso.org/standard/27001.

International Organization for Standardization. (2022). *ISO/IEC 27005 – Information security, cybersecurity and privacy protection - Guidance on managing information security risks*. URL: https://www.iso.org/standard/80585.html.

Javed, D., Jhanjhi, N., Khan, N. A., Ray, S. K., Al Mazroa, A., Ashfaq, F., and Das, S. R. (2024). "Towards the future of bot detection: A comprehensive taxonomical review and challenges on Twitter/X". In: *Computer Networks* 254, 110808.

Jenkins, P. R., Robbins, M. J., and Lunday, B. J. (2021). "Approximate dynamic programming for military medical evacuation dispatching policies". In: *INFORMS Journal on Computing* 33.1, pp. 2–26.

Kadane, J. B. and Larkey, P. D. (1982). "Subjective probability and the theory of games". In: *Management Science* 28.2, pp. 113–120.

Kaggle (2023). *Named Entity Recognition Dataset*. URL: https://www.kaggle.com/datasets/debasisdotcom/name-entity-recognition-ner-dataset(Accessed:18-09-2023).

Keeney, R. (1992). *Value Focused Thinking*. Harvard Univerity Press.

Keeney, R. (2007). "Modeling values for anti-terrorism analysis". In: *Risk Analysis* 27.3, pp. 585–596.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). "Optimization by simulated annealing". In: *Science* 220.4598, pp. 671–680.

Koller, D. and Milch, B. (2003). "Multi-agent influence diagrams for representing and solving games". In: *Games and Economic Behavior* 45.1, pp. 181–221.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Krasser, S. (2023). *#InfosecurityEurope: Preparing for Adversarial Machine Learning Attacks*. URL: https://www.infosecurity-magazine.com/opinions/adversarial-machine-learning/ (Accessed:05-07-2023).

Lamsal, R. (2020). "Design and analysis of a large-scale COVID-19 tweets dataset". In: *Applied Intelligence* 51, 2790–2804.

Law, A. M., Kelton, W. D., and Kelton, W. D. (2007). *Simulation modeling and analysis. 3rd Ed.* McGraw-Hill.

Lee, K., Eoff, B., and Caverlee, J. (2011). "Seven months with the devils: A long-term study of content polluters on Twitter". In: *Proceedings of the International AAAI Conference on Web and Social Media* 5.1, pp. 185–192.

Liu, Y. and Man, H. (2005). "Network vulnerability assessment using Bayesian networks". In: Proceedings of the SPIE 5812, pp. 61–71.

Luceri, L., Deb, A., Badawy, A., and Ferrara, E. (2019). "Red bots do it better: Comparative analysis of social bot partisan behavior". In: *Companion Proceedings of the 2019 World Wide Web Conference*, pp. 1007–1012.

Ma, Q., Wen, M., Xia, Z., and Chen, D. (2016). "A sub-linear, massive-scale look-alike audience extension system a massive-scale look-alike audience extension". In: *JMLR: Workshop and Conference Proceedings* 53.1, pp. 1–16.

Madiega, T. (2021). "*AI act. Briefing of EU Legislation in Progress, Document PE 698.792.*" In: European Parliamentary Research Service. URL: https://www.europarl.europa.eu/RegData/etudes/BRIE/2021/698792/EPRS_BRI(2021)698792_EN.pdf.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *An Introduction to Information Retrieval*. Cambridge University Press.

Maslej, N. et al. (2025). "Artificial Intelligence Index Report 2025". In: ArXiv. URL: https://arxiv.org/pdf/2504.07139.

Matejic, N. and Wilson, C. (2024). "Crimes of Influence: Generative Artificial Intelligence-led Crime as a Service". In: *The Commonwealth Cyber Journal*, pp. 75–95.

Mazurczyk, W., Lee, D., and Vlachos, A. (2024). "Disinformation 2.0 in the age of AI: A cybersecurity perspective". In: *Communications of the ACM* 67.3, pp. 36–39.

Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., and Tesconi, M. (2019). "Rtbust: Exploiting temporal patterns for botnet detection on Twitter". In: *Proceedings of the 10th ACM Conference on Web Science*, pp. 183–192.

McGregor, S. (2021). "Preventing repeated real world AI failures by cataloging incidents: The AI incident database". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.17, pp. 15458–15463.

Miehling, E., Rasouli, M., and Teneketzis, D. (2018). "A POMDP Approach to the Dynamic Defense of Large-Scale Cyber Networks". In: *IEEE Transactions on Information Forensics and Security* 13.10, pp. 2490–2505.

Miller, N., Thomas, M. A., Eichel, J. A., and Mishra, A. (2015). "A hidden Markov model for vehicle detection and counting". In: *Proceedings of 2015 12th Conference on Computer and Robot Vision*, pp. 269–276.

Müller, P., Sansó, B., and De Iorio, M. (2004). "Optimal Bayesian design by inhomogeneous Markov chain simulation". In: *Journal of the American Statistical Association* 99.467, pp. 788–798.

NIST (2018). *Special Publication (SP) 800-37, Revision 2. Risk Management Framework for Information Systems and Organizations*. National Institute of Standards and Technology (NIST). URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf.

NIST (2023). *NIST AI 100-1*. National Institute of Standards and Technology (NIST). URL: https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf.

Naveiro, R., Redondo, A., Insua, D. R., and Ruggeri, F. (2019). "Adversarial classification: An adversarial risk analysis approach". In: *International Journal of Approximate Reasoning* 113, pp. 133–148.

O'Brien, S. P. (2010). "Crisis early warning and decision support: Contemporary approaches and thoughts on future research". In: *International Studies Review* 12.1, pp. 87–104.

OECD (2019). *Recommendation of the Council on AI. Document OECD/LEGAL/0449*. The Organization for Economic Co-operation and Development (OECD). URL: https://legalinstruments.oecd.org/en/instruments/oecd-legal-0449.

Ordovás, J. M., Rios-Insua, D, Santos-Lozano, A., Lucia, A, Torres, A, Kosgodagan, A., and Camacho, J. M. (2023). "A Bayesian network model for predicting cardiovascular risk". In: *Computer Methods and Programs in Biomedicine* 231,107405.

Papadatos, K., Rantos, K., Markrygergeou, A., Koulouris, K., ..., and Zacharis, A. (2023). *Interoperable EU Risk Management Toolbox*. European Union Agency for Cybersecurity (ENISA). URL: https://www.enisa.europa.eu/sites/default/files/publications/Interoperable%20EU%20RM%20Toolbox.pdf.

Pasricha, N. and Hayes, C. (2019). "Detecting bot behaviour in social media using digital DNA compression". In: *Proceedings of 27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 376–387.

Pastor-Galindo, J., Zago, M., Nespoli, P., Bernal, S. L., Celdrán, A. H., Pérez, M. G., Ruipérez-Valiente, J. A., Pérez, G. M., and Mármol, F. G. (2020). "Spotting political social bots in Twitter: A use case of the 2019 Spanish general election". In: *IEEE Transactions on Network and Service Management* 17.4, pp. 2156–2170.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*. Wiley.

Powell, W. B. (2019). "A unified framework for stochastic optimization". In: *European Journal of Operational Research* 275.3, pp. 795–821.

Powell, W. B. (2022). *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*. Wiley.

Rabiner, L. R. (1989). "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2, pp. 257–286.

Raiffa, H, Richardson, J, and Metcalfe, D (2002). *Negotiation analysis: the science and art of collaborative decision making*. Harvard University Press.

Rauchfleisch, A. and Kaiser, J. (2020). "The False positive problem of automatic bot detection in social science research". In: *PloS ONE* 15.10, e0241045.

Redondo, A. and Insua, D. R. (2020). "Protecting from malware obfuscation attacks through adversarial risk analysis". In: *Risk Analysis* 40.12, pp. 2598–2609.

Ribeiro, F. N., Araújo, M., Gonçalves, P., Gonçalves, M. A., and Benevenuto, F. (2016). "Sentibench-a benchmark comparison of state-of-the-practice sentiment analysis methods". In: *EPJ Data Science* 5.1, pp. 1–29.

Rios Insua, D., Naveiro, R., Gallego, V., and Poulos, J. (2023). "Adversarial machine learning: Bayesian perspectives". In: *Journal of the American Statistical Association* 118.543, pp. 2195–2206.

Ríos, J. and Insua, D. R. (2012). "Adversarial risk analysis for counterterrorism modeling". In: *Risk Analysis* 32.5, pp. 894–915.

Roponen, J. and Salo, A. (2015). "Adversarial risk analysis for enhancing combat simulation models". In: *Journal of Military Studies* 6.2, pp. 82–103.

Saeidnia, H. R., Hosseini, E., Lund, B., Tehrani, M. A., Zaker, S., and Molaei, S. (2025). "Artificial intelligence in the battle against disinformation and misinformation: a systematic review of challenges and approaches". In: *Knowledge and Information Systems* 67, pp. 3139–3158.

Sarker, I. H. (2021). "Machine learning: Algorithms, real-world applications and research directions". In: *SN Computer Science* 2, 160.

Sarraute, C., Buffet, O., and Hoffmann, J. (2013). "Penetration Testing == POMDP Solving?" In: ArXiv. URL: https://arxiv.org/pdf/1306.4714.

Sayyadiharikandeh, M., Varol, O., Yang, K. C., Flammini, A., and Menczer, F. (2020). "Detection of novel social bots by ensembles of specialized classifiers". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 2725–2732.

Schroer, S. L. et al. (2025). " SoK: On the Offensive Potential of AI ". In: *Proceedings of 2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 247–280.

Scott, S. (2002). "Bayesian Methods for Hidden Markov Models". In: *Journal American Statistical Association* 97.457, pp. 337–351.

Sevillano, J. C., Insua, D., and Rios, J. (2012). "Adversarial Risk Analysis: The Somali Pirates Case". In: *Decision Analysis* 9.2, pp. 86–95.

Shachter, R. D. (1986). "Evaluating influence diagrams". In: *Operations Research* 34.6, pp. 871–882.

Shao, C., Ciampaglia, G. L., Varol, O., Yang, K.-C., Flammini, A., and Menczer, F. (2018). "The spread of low-credibility content by social bots". In: *Nature communications* 9.1, pp. 1–9.

Silva, V. (2018). "Critical Issues on Probabilistic Earthquake Loss Assessment". In: *Journal of Earthquake Engineering* 22.9, pp. 1683–1709.

Smith, A. F. and Roberts, G. O. (1993). "Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 55.1, pp. 3–23.

Spears, W. M. (1993). "Simulated annealing for hard satisfiability problems." In: *Cliques, Coloring, and Satisfiability* 26, pp. 533–558.

Stahl, D. O. and Wilson, P. W. (1995). "On players' models of other players: Theory and experimental evidence". In: *Games and Economic Behavior* 10.1, pp. 218–254.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). "Intriguing properties of neural networks". In: *Proceedings of 2nd International Conference on Learning Representations (ICLR)*.

Taslimasa, H., Dadkhah, S., Neto, E., Xiong, P., Ray, S., and Ghorbani, A. (2023). "Security issues in IoV: A comprehensive survey". In: *Internet of Things* 22, 100809.

Theodosiadou, O., Chatzakou, D., Tsikrika, T., Vrochidis, S., and Kompatsiaris, I. (2023). "Real-time threat assessment based on hidden Markov models". In: *Risk Analysis* 43.10, pp. 2069–2081.

Tidy, J. (2020). *SolarWinds: Why the Sunburst hack is so serious*. URL: https://www.bbc.com/news/technology-55321643.

Tierney, L. (1994). "Markov chains for exploring posterior distributions". In: *The Annals of Statistics* 22.4, pp. 1701–1728.

Twitter Developer (2020). *Twitter API Documentation*. URL: https://developer.twitter.com/en/docs/twitter-api(Accessed:18-10-2020).

UNECE (2022). *WP.29 - Introduction*. URL: https://unece.org/wp29-introduction(Accessed:03-10-2024).

US. Dep. Trans. (2021). *ODI Resume*. URL: https://static.nhtsa.gov/odi/inv/2021/INOA-PE21020-1893.PDF(Accessed:03-10-2024).

Unlu, A., Truong, S., Sawhney, N., and Tammi, T. (2025). "Unveiling the veiled threat: The impact of bots on COVID-19 health communication". In: *Social Science Computer Review* 43.4, pp. 675–704.

Varol, O., Ferrara, E., Davis, C., Menczer, F., and Flammini, A. (2017). "Online human-bot interactions: Detection, estimation, and characterization". In: *Proceedings of the International AAAI Conference on Web and Social Media* 11.1, pp. 280–289.

Vijayan, J. (2024). *XZ Utils Backdoor Implanted in Carefully Executed, Multiyear Supply Chain Attack*. URL: https://www.darkreading.com/cyber-risk/xz-utils-backdoor-implanted-in-intricate-multi-year-supply-chain-attack(Accessed:03-10-2024).

Virtanen, K., Karelahti, J., and Raivio, T. (2006). "Modeling air combat by a moving horizon influence diagram game". In: *Journal of Guidance, Control and Dynamics* 29.5, pp. 1080–1091.

Viscusi, W. K. (2020). "Pricing the global health risks of the COVID-19 pandemic". In: *Journal of Risk and Uncertainty* 61, pp. 101–128.

WEF (2024). *The Global Risks Report 2024*. World Economic Forum (WEF). URL: https://www3.weforum.org/docs/WEF_The_Global_Risks_Report_2024.pdf.

WEF (2025a). *Global Security Outlook 2025*. World Economic Forum (WEF). URL: https://reports.weforum.org/docs/WEF_Global_Cybersecurity_Outlook_2025.pdf.

WEF (2025b). *The Global Risks Report 2025*. World Economic Forum (WEF). URL: https://reports.weforum.org/docs/WEF_Global_Risks_Report_2025.pdf.

Walker, M. and Matsa, K. E. (2021). *News Consumption across social media in 2021*. Pew Research Center. URL: https://www.pewresearch.org/wp-content/uploads/sites/20/2021/09/PJ_2021.09.20_News-and-Social-Media_FINAL.pdf.

Wang, S. and Banks, D. (2011). "Network routing for insurgency: An adversarial risk analysis framework". In: *Naval Research Logistics* 58.6, pp. 595–607.

Wei, X., Guo, Y., and Yu, J. (2022). "Adversarial sticker: A stealthy attack method in the physical world". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3, pp. 2711–2725.

Weld, C. and Leemis, L. (2019). "Mixed-type distribution plots". In: *Information Visualization* 18.3, pp. 311–317.

Williams, T. J. (1994). "The Purdue enterprise reference architecture". In: *Computers in Industry* 24, pp. 141–158.

Wiper, M., Insua, D. R., and Ruggeri, F. (2001). "Mixtures of gamma distributions with applications". In: *Journal of Computational and Graphical Statistics* 10.3, pp. 440–454.

Xia, T. and Chen, X. (2020). "A discrete hidden Markov model for SMS spam detection". In: *Applied Sciences* 10.14, 5011.

Yang, K. C., Varol, O., Davis, C. A., Ferrara, E., Flammini, A., and Menczer, F. (2019). "Arming the public with artificial intelligence to counter social bots". In: *Human Behavior and Emerging Technologies* 1, pp. 48–61.

Yang, K. C., Varol, O., Hui, P. M., and Menczer, F. (2020). "Scalable and generalizable social bot detection through data selection". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.01, pp. 1096–1103.

Yang, K.-C., Ferrara, E., and Menczer, F. (2022). "Botometer 101: Social bot practicum for computational social scientists". In: *Journal of Computational Social Science* 5, pp. 1511–1528.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., ..., and Wen, J. (2023). "A Survey of LLMs". In: ArXiv. URL: https://arxiv.org/pdf/2303.18223.

Zhuang, J. and Bier, V. M. (2007). "Balancing terrorism and natural disasters—Defensive strategy with endogenous attacker effort". In: *Operations Research* 55.5, pp. 976–991.

von Knebel, M. and Anderljung, M. (2024). *A Collection of AI Governance Research Ideas (2024).* URL: https://www.markusanderljung.com/blog/a-collection-of-ai-governance-research-ideas-2024(Accessed:30-03-2025).