Authentication and Secure Communication in Satellite Systems

Joshua Smailes

St Anne's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2025

Acknowledgements

I would like to thank my supervisor, Prof. Ivan Martinovic, for his guidance and continued support over the years. Since our initial collaboration for my Bachelor's and Master's projects he has always been a strong believer in my abilities, without which this DPhil would not have been possible.

My thanks also go to the members of the Systems Security Lab for providing a fun and supportive working environment, good discussions, and just enough distraction from work. Particular thanks to Sebastian and Simon for their scientific expertise, support in iterating on ideas and writing, and general enthusiasm for the work I've been doing – I would have had a much harder time without them.

I appreciate the support of the EPSRC, the Department of Computer Science, and St Anne's College in providing funding for my research. Thanks also to the Cyber-Defence Campus of armasuisse S+T for supporting many of my experiments, and for providing an enjoyable working environment during the 6 months I spent with them. I am also grateful to Dr Martin Strohmeier for making it happen, and for working tirelessly to make sure my project, and countless others, run smoothly.

I would also like to express appreciation to my assessors, Prof. Christina Pöpper and Prof. Kasper Rasmussen. Kasper's early feedback (alongside that of Prof. Andrew Markham) was instrumental in directing my research, and their discussions and input helped refine the final document.

I am immensely blessed to have the support of so many wonderful friends, who provided some much-needed respite from work, and helped me get through this period with my sanity mostly intact. There are too many to name here, but particular thanks to Edd, a great colleague and even better friend.

Special thanks to my parents and wider family, for believing in me, praying for me, and supporting me through the highs and lows of this journey.

Finally, thanks to Abbie for putting up with my long hours and endless deadlines, for trying her best to understand what I've been working on and being enthusiastic about it regardless, and for providing rest and encouragement to switch off work when I badly needed it – I am endlessly grateful to have you.

...and thanks to you for reading this thesis! I hope you find it interesting.

Abstract

In recent years, satellite systems have become increasingly critical to modern global infrastructure, with applications ranging from communication and scientific observation through to banking and defence. As a result, they have naturally become a lucrative target to attacks – made easier by the wide availability of off-the-shelf hardware – leaving many legacy systems vulnerable and in need of protection. However, there is also great diversity in space systems: legacy systems coexist alongside modern megaconstellations and interplanetary networks, with new security challenges stemming from their scale and complexity. As a result, a multi-faceted approach to security is required, combining multiple techniques which build upon one another to provide effective protection as a whole.

One key challenge in securing both legacy and modern space systems is ensuring robust authentication mechanisms – legacy systems often lack sufficient cryptographic protection, while modern systems face challenges in managing the complexity of their networks and scaling key distribution. In this thesis we address these gaps by investigating authentication techniques that can effectively secure both legacy and modern space systems. We identify and tackle two crucial challenges in this area: the need for effective transmitter authentication, and the requirement of scalable and interoperable key management. First, by exploring physical layer fingerprinting as a means to authenticate transmitters and detect attacks, even in the face of optimised attackers, and enabling fingerprinting to be more easily applied even in single-transmitter systems, we can ensure effective authentication driven by ground systems. This is particularly useful in legacy systems, but can also provide benefits to modern systems, providing authentication mechanisms orthogonal to encryption to grant additional insight into attacks. Second, by adapting terrestrial Public Key Infrastructure (PKI) approaches so they can be used in interplanetary networks, and providing new satellite network simulation tools, we can facilitate seamless interoperability with terrestrial networks and accelerate future protocol research and development.

By providing authentication mechanisms that cover satellite systems in all their many forms, this research enhances the security and resilience of space infrastructure across both legacy and modern systems. Current satellite networks are small in scale, but as the space industry looks to establish more complex and interconnected networks, including a full solar system internet, the security and authentication mechanisms developed in this research will play an increasingly critical role in enabling reliable and secure communication across interplanetary distances. The impact of this work reaches across many aspects of the space industry, enabling the protection of critical infrastructure, supporting the development of more complex and interconnected space missions, and facilitating future research.

Contents

List of Figures viii

List of Abbreviations xii

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Contributions . . . 4
- 1.3 Outline . . . 8

2 Background 11
- 2.1 Emerging Trends in Space . . . 12
  - 2.1.1 Historical Context . . . 12
  - 2.1.2 New Space . . . 13
  - 2.1.3 Old Space . . . 19
- 2.2 Radio and the Physical Layer . . . 19
  - 2.2.1 Signal Sampling . . . 20
  - 2.2.2 Signal Modulation . . . 21
  - 2.2.3 Signal Processing . . . 23
  - 2.2.4 Software Defined Radios . . . 23
- 2.3 Attacks on Satellite Systems . . . 24
  - 2.3.1 Eavesdropping . . . 25
  - 2.3.2 Jamming . . . 25
  - 2.3.3 Spoofing . . . 26
  - 2.3.4 Time Spoofing . . . 28
  - 2.3.5 Broadcast Signal Intrusion . . . 29
  - 2.3.6 Supply Chain . . . 29
  - 2.3.7 Attacker Model . . . 31
- 2.4 Countermeasures to Attacks . . . 34
  - 2.4.1 Cryptography . . . 34
  - 2.4.2 Physical Layer Fingerprinting . . . 38

3 Authenticating Satellite Downlinks using RF Fingerprinting 43
- 3.1 Motivation . . . 44
  - 3.1.1 Contributions . . . 46
- 3.2 Related Work . . . 47
  - 3.2.1 Satellite Fingerprinting . . . 47
  - 3.2.2 Security . . . 49

3.2.3 Our Approach . . . 49
3.3 Threat Model . . . 50
3.3.1 Goals . . . 50
3.3.2 Capabilities . . . 51
3.4 System Design . . . 52
3.4.1 Design Decisions . . . 52
3.4.2 Fingerprinting Model . . . 55
3.5 Data Collection . . . 57
3.5.1 Data Preprocessing . . . 62
3.5.2 Dataset Construction . . . 62
3.6 Model Training . . . 63
3.6.1 Initial Results . . . 64
3.7 Evaluation . . . 68
3.7.1 Time Stability . . . 68
3.7.2 Correlations . . . 72
3.7.3 Subsampling . . . 74
3.7.4 Extensibility . . . 75
3.7.5 Transferability . . . 76
3.7.6 Security . . . 79
3.7.7 Comparison with Existing Systems . . . 82
3.7.8 Deployment Considerations . . . 86
3.8 Future Work . . . 88
3.9 Conclusion . . . 90
3.9.1 Availability . . . 90

4 Investigating Attacks on Physical Layer Satellite Authentication Systems 91
4.1 Motivation . . . 92
4.1.1 Contributions . . . 93
4.2 Related Work . . . 94
4.3 Threat Model . . . 96
4.3.1 Goals . . . 96
4.3.2 Capabilities . . . 98
4.3.3 Effective Attack Range . . . 101
4.4 Experiment Design . . . 104
4.4.1 Experimental Foundations . . . 104
4.4.2 Simple Jamming . . . 106
4.4.3 Optimised Jamming . . . 106
4.4.4 Identity Shift . . . 107

4.4.5 Fingerprint Masking 108
4.4.6 Data Poisoning 111
4.5 Data Collection 113
4.5.1 Jamming 113
4.5.2 Transmit-Receive Loop 117
4.6 Results 119
4.6.1 Simple Jamming 119
4.6.2 Optimised Jamming 122
4.6.3 Identity Shift 125
4.6.4 Fingerprint Masking 126
4.6.5 Poisoning 128
4.6.6 Single-Transmitter Fingerprinting 131
4.7 Discussion 134
4.7.1 Countermeasures 135
4.7.2 Future Work 136
4.8 Conclusion 137

5 Effective Authentication and Key Revocation in Interplanetary Networks 139
5.1 Motivation 140
5.1.1 Contributions 142
5.2 Related Work 143
5.2.1 Internet Drafts 143
5.2.2 Academic Works 145
5.2.3 Simulators 146
5.3 Threat Model 147
5.4 Framework Design 148
5.4.1 Scenarios 150
5.5 Experiment Design 151
5.5.1 Key Management Systems 152
5.5.2 Network Topologies 154
5.5.3 Routing 158
5.5.4 Traffic Modelling 159
5.6 The Deep Space Network Simulator (DSNS) 159
5.6.1 Architecture 160
5.6.2 PKI Functionality 163
5.6.3 Extending DSNS 164
5.7 Results 166
5.7.1 Connectivity 166

5.7.2 Connection Establishment 169
5.7.3 Key Revocation 173
5.8 Discussion 177
5.8.1 Limitations and Future Work 178
5.9 Conclusion 182
5.9.1 Availability 183

6 Conclusion **184**
6.1 Summary of Results 185
6.2 Future Work 186
6.3 Final Remarks 187

Appendices

A Signal Processing **190**
A.1 Frequency Bands 190
A.2 Antennas 190
A.3 Filters 192
A.4 Amplifiers 193
A.5 Attenuators 193
A.6 Digital Signal Processing 193

B Extended Attack Results **195**
B.1 Jamming: Phase Synchronisation 195
B.2 Identity Shift 195
B.3 Fingerprint Masking 196

C PKI Protocol Sequence Diagrams **199**
C.1 OCSP 200
C.2 OCSP Stapling 200
C.3 OCSP with Validators 201
C.4 CRL 201
C.5 CRL Broadcast 202

References **203**

List of Figures

1.1 Number of satellites currently and no longer in orbit, and total number of objects launched into space. 2

2.1 Comparison between direct sampling and quadrature (IQ) sampling, and the range of frequencies covered by each. 20
2.2 Illustration of the distribution of points under different digital modulation schemes. 21
2.3 Diagram of an RF signal modulation pipeline when using IQ modulation, and the hardware components involved. 38

3.1 An overview of the end-to-end fingerprinting process used by SatIQ. 52
3.2 An overview of the Siamese autoencoder architecture used in SatIQ's fingerprinting model. 53
3.3 The layout of an autoencoder. 53
3.4 An illustration of the triplet loss function. 56
3.5 The layers of the Siamese neural network used in SatIQ. 56
3.6 An overview of the data collection system. 58
3.7 Timeline of data collection in each location. 60
3.8 The distribution of the number of Iridium messages received per transmitter. 61
3.9 A message header received from an Iridium satellite. 61
3.10 Training and validation loss curves for one training run of SatIQ. . 63
3.11 The performance of the base SatIQ model. 64
3.12 ROC curves comparing each incoming message to different numbers of anchors. 66
3.13 Graph showing the performance of SatIQ for different training dataset sizes. 69
3.14 Graph showing the performance of SatIQ when anchors are collected at a time offset from the testing messages. 70
3.15 Heatmaps showing the performance of SatIQ for each dataset size and anchor time offset. 70
3.16 Heatmap showing the number of pairs of messages with the same transmitter ID in each day of the testing dataset. 71
3.17 Performance of the SatIQ model on slices of the dataset, sorted by various properties. 72
3.18 Performance of the SatIQ model on slices of the dataset, plotted against physical conditions during data collection. 73

3.19 Performance of the SatIQ model trained on subsampled data. . . . 75
3.20 ROC curve showing the system's performance on transmitters it has never seen before. . . . 75
3.21 ROC curves for models trained on datasets from each location, tested against different locations. . . . 77
3.22 Hardware setup for the replay attacks. . . . 79
3.23 ROC curves showing the system's performance when detecting replayed messages. . . . 80
3.24 Distance in fingerprint space between attacker-replayed messages and legitimate messages over time. . . . 82
3.25 Evaluation of the distribution of bits in the PAST-AI dataset, and how this may skew results. . . . 84
3.26 Heatmaps showing the predicted vs actual class under each type of attack, using the PAST-AI model. . . . 86

4.1 Illustration of each of the attacks described in the threat model. . . 97
4.2 The proportion of Iridium messages which fail to decode as the jammer power increases, with and without the use of Iridium's built-in error correcting code. . . . 102
4.3 The received power of a noise jammer as distance to the victim antenna varies, under free space path loss. . . . 103
4.4 Architecture of the "Siamese GAN" model used for the fingerprint masking experiments. . . . 109
4.5 Layers of the "Siamese GAN" model used for the fingerprint masking experiments. . . . 110
4.6 Overview of the hardware used to collect Iridium signals with additional noise. . . . 113
4.7 Level of noise added to the collected data over time. . . . 114
4.8 Number of messages collected for each noise level. . . . 115
4.9 Two Iridium message headers received during data collection. . . . 116
4.10 Examples of each of the techniques used in simple software jamming. 117
4.11 Illustration of the transmit-receive loop used for optimised spoofing attacks. . . . 118
4.12 Distance in fingerprint space caused by each simple jamming technique, as attacker power increases. . . . 120
4.13 False Rejection Rate (FRR) caused by each simple jamming technique, as attacker power increases. . . . 120
4.14 Results from the jamming experiments, when the attacker does not have phase synchronisation. . . . 122
4.15 Examples of some of the signals produced by the jamming attack. . 123

4.16 Initial results from the identity shift attack, on messages from the legitimate dataset. 124
4.17 Examples of some of the signals produced by the identity shift attack. 125
4.18 Results from the identity shift attack on messages that have been transmitted and received using an SDR. 126
4.19 Distance in fingerprint space between legitimate transmitters and those generated by the GAN. 127
4.20 A selection of the steps involved in poisoning by interpolating between the fingerprints of two legitimate transmitters. 128
4.21 Number of steps required to poison the reference fingerprints to accept one transmitter instead of another. 129
4.22 The distance in fingerprint space from each step in the poisoning process to the initial/target/previous step. 130
4.23 A selection of the steps involved in interpolating between a legitimate transmitter's fingerprint and a random fingerprint. 131
4.24 Distance in fingerprint space for the replay dataset, comparing SatIQ to the GAN's discriminator. 132
4.25 ROC curves comparing the GAN against SatIQ for distinguishing legitimate messages from attacker-replayed communication. 133

5.1 Example of an interplanetary network of satellites and the communication between them. 141
5.2 256 ground stations placed at random points on the Earth's land surface to produce a simulated network with global communication. 155
5.3 Overall structure of DSNS and its high-level operation. 160
5.4 An example visualisation of an interplanetary network in DSNS. . . 162
5.5 State of the relays from Earth to the Moon and Mars over time. . . 166
5.6 Latency over time for an Earth/Moon/Mars interplanetary network, passing messages between nodes in each segment. 167
5.7 Latency of messages between ground stations on opposite sides of the globe. 167
5.8 Distribution of latencies when establishing a new connection under each constellation topology. 169
5.9 Establishment overhead of initiating a connection in the Earth/-Moon/Mars network under the distributed CA configuration. 170
5.10 Overall message hop count to initiate a connection in the Earth/-Moon/Mars network under the distributed CA configuration. 171
5.11 Coverage over time for a revocation, relative to the time at which the revocation was issued. 172

5.12 Example of the race between attacker and CA, when an attack and revocation originate in different network segments. 172

B.1 Results from the jamming experiments, when the attacker can achieve phase synchronisation with the victim. 196
B.2 False Acceptance Rate for the identity shift experiments under each configuration. 197
B.3 False Acceptance Rate for the fingerprint masking experiments when a GAN is not used. 198

List of Abbreviations

ADC . . . . . Analogue to Digital Converter

AM . . . . . Amplitude Modulation

ASK . . . . . Amplitude Shift Keying

AUC . . . . . Area Under Curve

AWG . . . . . Arbitrary Waveform Generator

BGP . . . . . Border Gateway Protocol

BP . . . . . Bundle Protocol

BPSec . . . . . Bundle Protocol Security

BPSK . . . . . Binary Phase Shift Keying

CA . . . . . Certificate Authority

CCSDS . . . . Consultative Committee for Space Data Systems

cFS . . . . . core Flight System

COTS . . . . . Commercial Off-The-Shelf

CRL . . . . . Certificate Revocation List

DAC . . . . . Digital to Analogue Converter

DSN . . . . . Deep Space Network

DSNS . . . . . Deep Space Network Simulator

DSP . . . . . Digital Signal Processing

DTN . . . . . Delay Tolerant Network

EDRS . . . . . European Data Relay System

EER . . . . . Equal Error Rate

ESA . . . . . European Space Agency

FAR . . . . . False Acceptance Rate

FIR . . . . . Finite Impulse Response

FM . . . . . Frequency Modulation

FNR . . . . . False Negative Rate

FPR . . . . . False Positive Rate

FRR . . . . . False Rejection Rate

FSK . . . . . Frequency Shift Keying

FSS . . . . . Federated Satellite System
GAN . . . . . Generative Adversarial Network
GEO . . . . . Geosynchronous Equatorial Orbit
GNSS . . . . . Global Navigation Satellite System
GPS . . . . . Global Positioning System
GSaaS . . . . . Ground Station as a Service
HF . . . . . High Frequency
HIBC . . . . . Hierarchical Identity Based Cryptography
ILL . . . . . Inter-Layer Link
IP . . . . . Internet Protocol
IQ . . . . . In-phase / Quadrature
IRA . . . . . Iridium Ring Alert
ISL . . . . . Inter-Satellite Link
LEO . . . . . Low Earth Orbit
LNA . . . . . Low Noise Amplifier
LTP . . . . . Licklider Transmission Protocol
MEO . . . . . Medium Earth Orbit
ML . . . . . Machine Learning
NASA . . . . . National Aeronautics and Space Administration
OCSP . . . . . Online Certificate Status Protocol
OSDMA . . . Opportunistic Space-Division Multiple Access
PKI . . . . . Public Key Infrastructure
PM . . . . . Phase Modulation
PSK . . . . . Phase Shift Keying
QAM . . . . . Quadrature Amplitude Modulation
QPSK . . . . . Quadrature Phase Shift Keying
RF . . . . . Radio Frequency
ROC . . . . . Receiver Operating Characteristic
SDR . . . . . Software Defined Radio
SNR . . . . . Signal to Noise Ratio
SSA . . . . . Space Situational Awareness

SSB . . . . . Single-Sideband

TCP . . . . . Transmission Control Protocol

TDOA . . . . . Time Difference of Arrival

TDRSS . . . . Tracking and Data Relay Satellite System

TLE . . . . . Two Line Element

TPR . . . . . True Positive Rate

TT&C . . . . . Telemetry, Tracking, and Control

UDP . . . . . User Datagram Protocol

UHF . . . . . Ultra High Frequency

VHF . . . . . Very High Frequency

1 Introduction

Contents

1.1 Motivation **2**
1.2 Contributions **4**
1.3 Outline **8**

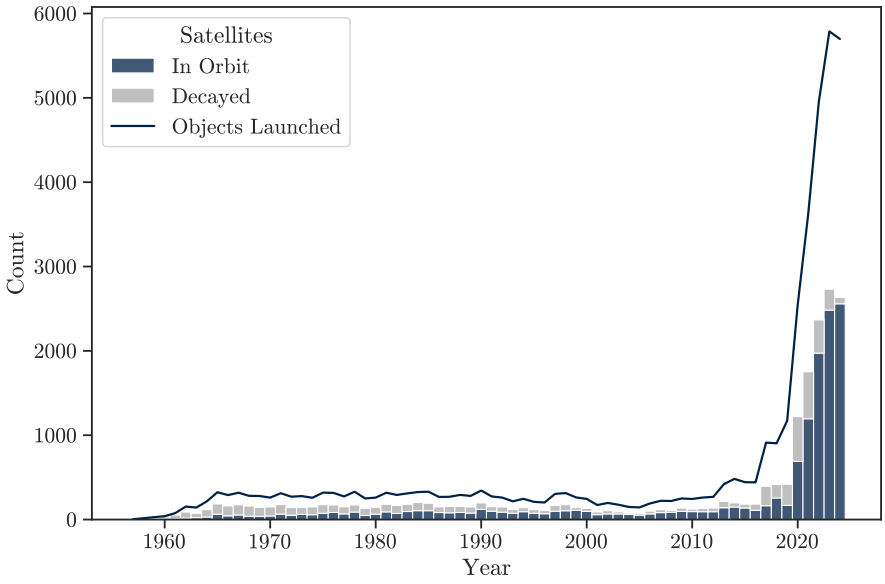


Figure 1.1: Number of satellites currently and no longer in orbit (data from [1]), and the total number of objects launched into space (data from [2]), by launch year.[1]

1.1 Motivation

The rapid growth of the space industry, driven by technological advancements in recent decades, has resulted in a significant increase in the number of satellites launched into orbit, as illustrated in Figure 1.1. This expansion has led to a growing reliance on satellite systems to provide critical infrastructure and services, including communication, navigation, and remote sensing, with the global space economy valued at approximately 600 billion USD, and estimated to reach over 900 billion by 2033 [3]. However, this increased reliance on space-based systems has also opened up new threats, as satellites have become attractive targets for malicious actors seeking to disrupt or exploit these systems – a global space blackout would have wide-reaching effects on transport (by land, air, and sea), the financial sector, power grids, and more [4].

Attacks on space systems can take many forms, including jamming, spoofing and replay attacks, which can compromise the integrity and availability of satellite communication. Recent notable examples of attacks include GPS and Starlink jamming over conflict areas in Ukraine and Israel [5, 6], TV satellite hijacking to

[1]Data from November 1998 contained reporting errors, which have been removed.

broadcast propaganda [7], and hijacking of US weather satellites [8]. We can see from these that satellites are critical to modern international infrastructure, and that existing protections are not enough to prevent attacks.

It is clear that the criticality of satellite systems to modern infrastructure is not matched by a corresponding focus on security, with many systems lacking robust cryptographic protection [9], or with outdated and compromised cryptography or leaked keys [10, 11]. Many of these systems cannot support modern cryptographic standards, particularly legacy satellites which lack sufficient computational power and storage, due to the great deal of radiation shielding present in their hardware. Furthermore, a good number of systems support rekeying but not full reconfiguration, due to a focus on availability over other security properties, and operators are hesitant to add complicating factors which risk interrupting service. To counter these issues, there is a growing need for techniques which can detect and prevent attacks even in the absence of cryptography, to secure legacy satellites and bolster existing cryptographic systems.

In addition to growing threats at the physical layer, the space industry is also expressing increased interest in new paradigms of satellite communication, with a particular interest in interplanetary networks. These involve communication over long distances between different planets, with initiatives such as NASA's LunaNet and ESA's Moonlight providing an initial exploration of this concept by establishing relay networks around the moon [12–14]. As these networks grow, they are requiring new protocols to be built: traditional terrestrial protocols are not well-suited to handle intermittently and sparsely connected networks, and those optimised for wireless sensor networks and other ad-hoc networks largely ignore the predictability of movement in space, making significant efficiency losses in the process. Standards bodies are making progress in designing protocols suited for space systems at all layers of the network stack [15], but there are still a large number of unsolved problems – in particular, the architecture of certificate-based authentication and Public Key Infrastructure (PKI), which require low-latency

queries to trusted authorities from anywhere in the network.[2] This also raises some interesting governance issues, since operators may not fully trust one another but might nevertheless use shared infrastructure, thus requiring trusted mediators or mutually agreed certificate authorities.

At both the physical layer and in higher-level protocol design, satellites present a particularly interesting security challenge: at face value they appear to be similar to many terrestrial systems, but their remote location, long lifespans, and high cost of replacement make them more difficult to secure and maintain. The long-distance links used by satellites add difficulty to both physical layer techniques and the building of new protocols, creating distortions which obfuscate useful information present at the physical layer, and producing a unique network topology in which low-latency centralised queries are impossible, but the state of the network can still be predicted ahead of time.

In this thesis we address both of these problems to strengthen security across the network stack. We first evaluate physical layer fingerprinting as a countermeasure to attacks, taking advantage of the unique hardware characteristics present in transmitters, differentiating satellites not only from one another but also from attackers. We then look at attacks targeting these systems directly, such as jamming, spoofing, and replay attacks, which can compromise the integrity and availability of satellite communication. Finally, we look at authentication within the larger context of future space networks, demonstrating that the predictable movement and connectivity of these networks enables the use of terrestrial Public Key Infrastructure (PKI) in interplanetary networks, instead of requiring probabilistic or gossip-based approaches. Our results provide a foundation for more secure and interconnected satellite systems, thus supporting the growing demands of the global space economy.

1.2 Contributions

In this section I describe the ways in which the work that comprises this thesis has contributed to the field of systems security. The majority of the content in this thesis

[2] A summary of existing research and open questions in this area can be found in Section 2.4.1.

has been produced in collaboration with others, but I have only included projects for which I was both the primary contributor and lead author. In these works, my co-authors provided assistance in gathering data, carrying out experiments, and preparing manuscripts for submission.

- We explore the application of radio transmitter fingerprinting as a method by which the satellite downlink can be secured, proposing novel techniques which look in particular at high sample rate features of the waveform, making fingerprints harder for an attacker to forge. We demonstrate the effectiveness of this technique on the Iridium satellite constellation, both in distinguishing legitimate transmitters from one another and detecting illegitimate (i.e. attacker-controlled) hardware. We also provide a dataset of over 1.7 million high sample rate Iridium messages, the first of its kind. This work has been published at the 2023 ACM Conference on Computer and Communications Security (CCS) [16].

- We demonstrate that the above techniques are stable over time, assess the impact of weather and signal quality factors on performance, and demonstrate that new transmitters can be introduced to the system without retraining and with no impact on performance. In achieving this, we also extended the collected data, providing a further 10 million Iridium messages, spread across 3 different receiver configurations in different locations. This work has been published in ACM Transactions on Privacy and Security (TOPS) [17], as an extension of [16].

- We evaluate the robustness of high sample rate fingerprinting against adversarial interference and jamming attacks, assessing the difficulty of specific disruption to a fingerprint-based authentication system, and comparing the results to traditional jamming techniques against the underlying message decoder. During the completion of this work, we produced a dataset of 500 000 messages collected under varying levels of Gaussian noise. This work

has been published at the 2024 Workshop on Security of Space and Satellite Systems (SpaceSec) [18].

- We evaluate targeted attacks on high sample rate fingerprinting in a more realistic setting, looking in particular at jamming and spoofing attacks. We show that if an attacker has access to the underlying model weights, it is easy to craft a jamming signal that provides generalised fingerprint disruption at significantly lower amplitudes than traditional jamming techniques. We also evaluate a number of techniques for spoofing or altering the identity of a transmitter, and make use of a hardware RF loop to enable the training of a Generative Adversarial Network (GAN) to counteract the hardware fingerprint of the transmitter. Finally, we show how this technique also enables fingerprinting to be used in authenticating single transmitters, without requiring a dataset comprising a large number of different devices. This is particularly useful in securing legacy satellite systems, which are unlikely to be part of a larger constellation. This work is being prepared for submission to the 2026 ACM Conference on Computer and Communications Security (CCS) [19].

- We study authentication in the context of emerging network topologies, shifting from legacy authentication schemes to modern PKI approaches suitable for megaconstellations and interplanetary communication. We define a suite of standardised experiments and introduce the Deep Space Network Simulator (DSNS) to assess connection establishment, key revocation, and the resilience of various PKI configurations under realistic attack scenarios. Our work demonstrates that terrestrial PKI techniques can be effectively adapted to work in the challenging environment of highly distributed space networks, and we further propose two configurations – OCSP hybrid, and relay node firewalls – to constrain the reach of compromised keys and alleviate the load on interplanetary relay links. This work is under review for the 2026 ACM Asia Conference on Computer and Communications Security (AsiaCCS) [20]. A

tool release paper for the simulator arising from this work has been published at the 2025 ESA conference on Security for Space Systems (3S) [21].

The following works are also referenced in this thesis. These are shorter or smaller works, or papers to which I contributed but was not lead author. Any content which was not primarily or entirely produced by me is clearly marked in the thesis.

- We provide a proof of concept implementation of the QUIC transport layer protocol for NASA's cFS satellite operating system. This was presented as a poster at the 2023 ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec) [22]. These results laid the groundwork for another student project, in which QUIC was tested on the "OPS-SAT" satellite, demonstrating the protocol's features and the impact they can have on performance.

- We demonstrate a denial of service attack on the Starlink user terminal. This was presented as a poster at the 2023 ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec) [23].

- We evaluate the real-world requirements of spoofing attacks on satellite systems, linking the feasibility and impact of an attack to hardware requirements and budget. We use real-world experiments and RF simulations to show that many satellite systems can be attacked at significant distances using off-the-shelf hardware. This work has been published at the 2023 ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec) [24].

- We demonstrate that the aforementioned high sample rate fingerprinting techniques are adaptable to different satellite constellations, by gathering a new dataset of 8.9 million messages from the ORBCOMM satellite formation and training a new model to identify satellites and attacker-controlled transmitters. We also assess a new technique to anonymise messages, removing identifying information to gain access to more data for training purposes. This work

has been published at the 2025 Workshop on Security of Space and Satellite Systems (SpaceSec) [25].

- Finally, we show that high sample rate fingerprinting techniques are also useful in the context of aviation, demonstrating effectiveness on the ADS-B protocol used by many aircraft. We also show that similar techniques can be used to identify wired devices communicating over a shared bus. This work has been published at the 2023 IEEE Digital Avionics Systems Conference (DASC) [26].

1.3 Outline

The remainder of this thesis is structured as follows:

- Chapter 2 provides an overview of the background surrounding this thesis. We explore the evolution of the space industry, looking in particular at the emergence of modern "New Space" systems. We contrast these with the legacy "Old Space" systems, highlighting their differences and the unique security challenges associated with each. We also introduce key concepts including physical layer radio systems, digital signal processing, and software defined radios. Finally, we examine existing threats to space systems, including attacks targeting the physical layer and supply chain vulnerabilities, providing a necessary understanding for the remainder of the thesis.

- Chapter 3 introduces the topic of high sample rate satellite fingerprinting, exploring its use as an authentication mechanism to secure legacy satellite systems lacking cryptography. We use the Iridium satellite constellation as a case study, collecting over 10 million messages at a high sample rate (25 MS/s). Using this dataset, we train and evaluate a model composed of a Siamese network and autoencoder, and demonstrate its effectiveness at both distinguishing legitimate transmitters from one another and detecting attacker-controlled transmitters. We also demonstrate that new transmitters

can be added to the system during operation with no measurable impact on performance, and establish experimentally the rate at which reference messages must be refreshed. Finally, we discuss the possibility of transferring the techniques and results to new satellite systems.

- Chapter 4 builds upon this, evaluating the resilience of high sample rate satellite fingerprinting techniques under a range of attacks. We begin by establishing in detail the threat model under which the attacker is operating, including an evaluation of the distance at which an attacker should be able to cause disruption or inject messages on the base message decoder. This is backed up by real-world measurements and experimentation. Next we look at standard RF jamming techniques in the context of fingerprinting systems – in particular, Gaussian noise and tone jamming – to see how effective these are in disrupting transmitter fingerprints. This continues on to an evaluation of targeted jamming and spoofing attacks on the fingerprinting system, in which the trained fingerprinter is used by an attacker to optimise waveforms to disrupt the fingerprinter. We demonstrate that this enables jamming at a significantly lower amplitude, even when the attacker cannot achieve perfect phase synchronisation. Finally, we build a GAN to spoof the identity of transmitters by altering the fingerprint, and show that the resulting model can also be used as an effective method of authentication against spoofing attacks on single transmitters, thus eliminating the need for a full constellation (or even multiple transmitters) during training.

- Chapter 5 shifts focus from protecting legacy systems, looking instead at authentication challenges in emerging satellite networks. In particular, we assess the challenges involved in connection establishment, authentication, and revocation in interplanetary networks with huge numbers of nodes and long-distance links between them which are not always available. Through the development of a new network simulator optimised for interplanetary networks, we show that the predictable movement and connectivity of these networks

enables terrestrial protocols to be adapted to work in this new context. We also propose and evaluate two new configurations of these standard protocols to provide improved performance in the interplanetary setting: a hybrid protocol to enable more efficient communication across relay links, and a firewall or filter to reduce the reach of compromised keys, and minimise the attacker's load on the relay link. We conclude by exploring the possibility of integration between terrestrial and interplanetary networks to form a unified "interplanetary internet", and the standardisation efforts required to achieve this goal.

- Chapter 6 provides a summary of all the work in this thesis, and concludes with a discussion of open questions in the field and interesting areas of future work.

2
Background

Contents

2.1 Emerging Trends in Space 12
2.1.1 Historical Context 12
2.1.2 New Space 13
2.1.3 Old Space 19
2.2 Radio and the Physical Layer 19
2.2.1 Signal Sampling 20
2.2.2 Signal Modulation 21
2.2.3 Signal Processing 23
2.2.4 Software Defined Radios 23
2.3 Attacks on Satellite Systems 24
2.3.1 Eavesdropping 25
2.3.2 Jamming 25
2.3.3 Spoofing 26
2.3.4 Time Spoofing 28
2.3.5 Broadcast Signal Intrusion 29
2.3.6 Supply Chain 29
2.3.7 Attacker Model 31
2.4 Countermeasures to Attacks 34
2.4.1 Cryptography 34
2.4.2 Physical Layer Fingerprinting 38

This chapter discusses the required background topics related to this thesis, providing the foundation of understanding required for the remaining chapters. We start by examining emerging trends in space, in which we compare the evolving paradigms of Old Space and New Space, focusing on key differences in budget, operational approaches, and security challenges. We then explore radio and physical layer communication in terrestrial and satellite systems, moving from the physical layer through to software defined radios and digital signal processing. We then look at the threat profile of satellite systems, outlining common types of attacks and their impact. Finally, we outline some commonly used countermeasures to attacks, with a focus on cryptographic protocols and physical layer fingerprinting.

2.1 Emerging Trends in Space

This section addresses the changing landscape of the space industry, and the ways it has affected the domain of security.

2.1.1 Historical Context

In all but the most recent decades, space has been dominated by nation-state actors, militaries, and a small number of large companies. Satellites were built with high budgets and long lifespans, and were primarily for scientific measurement, Earth observation, or military applications. In these systems, security (where present) tended to operate on a "security by obscurity" model with highly restricted information flows to prevent espionage, and with a heavy focus on ensuring availability – this often came at the expense of other security properties, due to the immense financial and operational consequences of losing a satellite. This paradigm is commonly referred to as "Old Space".

As technology advanced, Commercial Off-The-Shelf (COTS) components became more readily available and satellite launches became more affordable. These changes paved the way for the "New Space" era, characterised by lower costs, the emergence of small satellites and CubeSats, and ridesharing opportunities which allow many satellites to be launched on a single rocket. Consequently, the number of satellites

in orbit has grown exponentially, as we have already seen in Figure 1.1, and the range of applications has expanded from its traditional use cases to include a variety of commercial and academic endeavours. While Old Space systems were built to last with a focus on reliability and availability, the advance to New Space introduces different considerations, including the need for updated security practices to match the increased complexity and diversity of modern space operations.

2.1.2 New Space

The advent of New Space represents a significant shift in satellite design and operation. Enabled by a number of recent technological advances, New Space has lowered the barrier for entry to space activities, attracting private industry, academic institutions, and startups to participate in space missions thanks to a reduced budget and accelerated development timeline. However, this increased activity and diversity also raises a number of novel security challenges.

Technological Advances

Recent improvements in miniaturisation, standardisation, and launch methodologies have led to dramatic changes in the way satellites are designed, built, and deployed. These advances both reduce cost and foster innovation, enabling a wider range of organisations to participate in space development and driving the growth of the industry as a whole.

Commoditisation The commoditisation of space technology and the increased availability of COTS components has been a significant factor in making space more accessible to a broader range of organisations and individuals, enabling smaller companies, universities, and even hobbyist groups to participate in space development. Instead of developing components from scratch, standard components can now be purchased significantly more cheaply, including onboard computers, power modules, antennas, transponders, and even payloads.

This increase in standard components provides both opportunities and risks from a security perspective. Reducing the number of bespoke parts can often improve

security, particularly when it prevents recreation or reimplementation of protocols and security mechanisms, but it also runs the risk of opening up many systems to attack at the same time if a wide-reaching vulnerability is discovered. Standards documents and standard implementations of protocols or hardware can mitigate this, by ensuring all deployed systems meet a minimum set of security requirements.

CubeSats The CubeSat standard provides a a fixed unit dimension of $10 \times 10 \times 10\,\text{cm}$ and maximum mass of $2.0\,\text{kg}$ for a "1U" unit, and supports larger satellites comprising multiple units – 1.5U, 2U, 3U, 6U, and 12U are commonly used [27]. This standard has had a number of parallel benefits: it enables the development of components with widely compatible dimensions and ensures the use of consistent deployers with standardised release mechanisms, simplifying launch operations.

Ridesharing Assisted by the CubeSat standard, ridesharing has emerged as a cost-effective method for launching secondary payloads. By accommodating multiple small satellites or CubeSats on a single rocket (or including them alongside a larger mission), ridesharing not only optimises the use of available launch capacity but also significantly reduces the launch cost of individual missions. This practice has proven popular in recent years, with approximately 250 secondary payloads launched in 2021, almost 200 of which were CubeSats [28].

Clusters and Constellations A satellite cluster is a group of satellites that works together to achieve a common goal, by either providing increased coverage or mutually beneficial services or measurement. Constellations expand on this concept, using a larger number of satellites moving in a coordinated pattern to provide global coverage. Many constellations use a variant of the Walker constellation, in which all satellites are spread out in planes of the same altitude and inclination, with the positions of satellites in each plane defined by three numbers $T/P/F$: T satellites total, spread across P different orbital planes, with a relative phase of F between each adjacent plane [29]. A constellation can place satellites in higher orbits to provide coverage with fewer satellites, or place a larger number of satellites in

Low Earth Orbit (LEO) to achieve the same coverage with lower latency. Clusters and constellations have become increasingly popular in recent years, thanks to advances in miniaturisation and cheaper launches. Examples of constellations include the Global Positioning System (GPS), composed of 31 satellites [30], the Iridium communication constellation, composed of 66 satellites [31], and the Starlink internet constellation, composed of 7135 satellites [32].[1]

Federated Satellite Systems Federated satellite systems (FSS) take this concept a step further, constructing a network in which satellites and systems communicate and work together, even if owned and operated by different organisations. This allows for greater interoperability and cooperation between different companies and space agencies, enabling significantly improved communication coverage and resource sharing that would be difficult or impossible for a single organisation to achieve on its own.

Interplanetary Networks As satellite networks continue to grow, there has been an increased interest in building satellites further afield, expanding to the Moon, Mars, or deeper in space [12, 33]. Communicating with these satellites results in an interplanetary network topology, with low-latency planetary segments partitioned by high-latency long-distance links between planets; for example, the latency between the Earth and Mars can vary between 3 and 22 minutes. This type of network is considered to be a Delay Tolerant Network (DTN) [34], and raises a number of novel challenges, particularly in routing and key management; we discuss this further in Chapter 5.

Ground Segment The space segment is not the only area seeing significant change, however – the ground segment has also improved considerably. In addition to a wider range of off-the-shelf hardware, ground stations are increasingly adopting a cloud-based approach referred to as Ground Station as a Service (GSaaS). Under this model, satellite operators rent access to a global network of ground stations,

[1] All numbers correct as of 2025.

which they use to communicate remotely with the satellite. Not only does this reduce up-front hardware costs, enabling smaller organisations to access modern ground system hardware, it also provides on-demand use with global coverage, scalability to match current operational requirements, and lower latency by using ground stations closer to the satellite. Currently deployed GSaaS providers include Amazon's AWS Ground Station, and Viasat's Real-Time Earth [35, 36].

Relay Networks Alongside GSaaS, relay networks provide a backbone for inter-satellite communication and information transfer, allowing satellites to transmit data to ground stations (or further afield) even when there is not a direct line of sight. Examples of such networks include NASA's Tracking and Data Relay Satellite System (TDRSS) [37], and ESA's European Data Relay System (EDRS) [38], which provide critical communication infrastructure to their respective space agencies. Proposals like NASA's LunaNet and ESA's Moonlight take this further with plans to provide additional relays around the Moon for lunar communication, taking a first step towards interplanetary networks [12–14].

Use Cases

These technical advances have led to a significant increase in the range of organisations launching satellites, as well as the purposes for which they are used. Alongside governments and large aerospace companies, increasing numbers of satellites are now being launched by other groups, including startups, universities, and non-profit groups.

One of the primary drivers of this trend is a growing demand for Earth observation data, which can be used for a variety of applications including weather monitoring, mapping, and managing natural resources. Companies like Planet Labs and Maxar Technologies have launched constellations of small satellites to provide high resolution imagery and data to customers in a wide range of industries, including agriculture, mining, energy, and telecommunications [39, 40]. Non-profit organisations are also using this data to support their goals, with companies like

EOS Data Analytics providing their satellite data to non-profits and NGOs for crop monitoring, agriculture development, and environmental protection [41].

The use of satellites by governments is expanding beyond traditional applications like weather forecasting and navigation. Space agencies such as ESA and NASA are launching increasing numbers of satellites to perform a variety of functions including Earth observation, scientific measurement, and the study of other planets. Satellites are also being increasingly used for military purposes, including navigation, surveillance, and communication, as countries invest in satellite systems to bolster their security and defence capabilities [42]. We can see the critical position these systems hold with regard to national security by observing Russian jamming of GPS, Starlink, and Iridium satellites during its invasion of Ukraine since 2022 [6, 43].

In addition to these commercial applications, satellites are also being used for a range of scientific and educational purposes. Universities and research institutions are launching satellites to conduct experiments and gather data in areas ranging from wildfire monitoring [44] and atmospheric studies [45] through to astrobiology [46] and even art [47, 48]. Satellites are also being launched specifically for space research: ESA's "OPS-SAT" (deorbited in 2024) was a satellite platform for testing mission control capabilities [49], and the upcoming "CyberCUBE" satellite seeks to enable security-specific onboard experiments [50]. This is not limited to the academic world either: in the 2023 "Hack-A-Sat" competition, offensive security experts were encouraged to directly attack a satellite, competing to gain control of the "Moonlighter" satellite created for this event [51]. Each of these missions and projects provides valuable opportunities for experimentation, and for students and researchers to gain hands-on experience with space technology.

As the number of satellites launched continues to grow, so too does the interest in tracking and monitoring their activities. Combined with tracking the movement of space debris, this is commonly referred to as Space Situational Awareness (SSA). Alongside space agency monitoring efforts, community projects like "SatNOGS" provide open-source monitoring of satellite communication through a network of ground stations [52].

Challenges

Although recent technical advances have opened up new opportunities for organisations to launch and operate satellites, there are still a number of challenges involved in operating satellites. As the scale and complexity of satellite networks grows, standardisation becomes more critical than ever. Although interoperability is a highly beneficial outcome from this, the more important goal is to ensure best practices are implemented and followed across the board, which may not occur if organisations are left to develop and implement their own protocols. As a result, common vulnerabilities may find their way into protocols, software, and hardware, opening systems up to unnecessary exploits.

The Consultative Committee for Space Data Systems (CCSDS) is a cross-agency collaboration positioned to solve this issue, providing a huge range of standards, recommended practices, information reports, and other useful documents for spacecraft operators. They publish standard protocols and delivery mechanisms for both payload data and Telemetry, Tracking, and Control (TT&C) messages, enabling standardisation across the board [15]. They also release documents covering many different aspects of space missions, including physical layer communication, onboard interfaces, and mission management. Finally, they provide useful standards for authentication and encryption in space missions. NASA have also released the source code for their onboard flight software [53], and their implementation of the "SDLS" protocol to secure communication between ground and space [54]. Alongside enabling more efficient development and potential interoperability, these standards and reference implementations also improve the security of space systems, combining the knowledge of multiple agencies and reducing the security risks involved with building new software and protocols from scratch.

As the space industry continues to grow with the emergence of New Space, the increasing scale of satellite networks has introduced further security challenges that must be addressed. This includes ensuring the security of communication at the physical layer, as well as protecting against attacks targeting the underlying communication protocols and vulnerabilities in the supply chain. In Section 2.3,

we evaluate the specific threats posed to satellite systems, and in Section 2.4 we introduce some existing countermeasures which can be used to enhance the security of space systems.

2.1.3 Old Space

Although recent research focuses on New Space systems, Old Space satellites still provide a wealth of useful scientific insight, Earth observation data, connectivity and more. For example, the US Department of Agriculture combines satellite data from multiple sources including NASA, ESA, NOAA, and Spot Image, to monitor crop health and predict yields [55]. These systems were designed with decades-long lifespans, and as a result, many of them are still operational today, providing valuable data and services despite being launched many years ago.

However, keeping these systems operational raises new challenges, particularly as the capability of attackers increases. Lacking modern cryptographic security, many Old Space satellites are vulnerable to attacks such as eavesdropping, jamming, and spoofing, which can compromise the integrity and confidentiality of the payload data. Furthermore, these attacks can potentially be carried out on the TT&C link itself, risking permanent damage to the satellite.

The outdated hardware and software used in these systems can make it difficult to implement modern security practices, leaving them exposed to known vulnerabilities. Additionally, the high development budget of these satellites means it is usually not financially viable to simply replace them if security properties are compromised – operators must instead focus on techniques to improve the security of the system without physical modification to the satellite. We discuss this further in Section 2.4.

2.2 Radio and the Physical Layer

Much of the work in this thesis relies upon Digital Signal Processing (DSP): the capture, manipulation, and/or reproduction of radio signals at the physical layer. This section introduces these concepts and outlines the mathematics and electronics underpinning them.

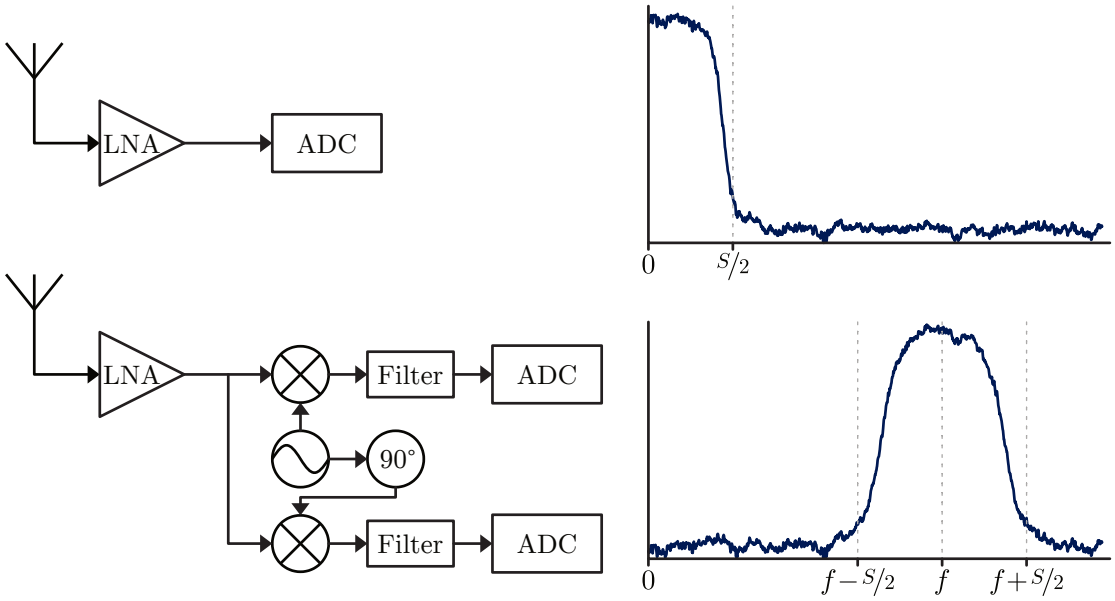


Figure 2.1: Comparison between direct sampling and quadrature (IQ) sampling, and the range of frequencies covered by each when sampling at a rate of S samples per second.

2.2.1 Signal Sampling

In order to process signals, they must first be sampled – that is, converted from a continuous signal to a discrete signal by measuring it at fixed points. This occurs in both time (discretisation; dividing the signal into equal intervals) and amplitude (quantisation; digitally measuring the amplitude for the given point in time, resulting in a loss of precision). The Nyquist-Shannon sampling theorem states that a signal with maximum frequency f Hz can be perfectly reconstructed from discrete samples, provided the rate of sampling is at least $2f$ samples per second [56]. This limits the bandwidth of the signal, and is often complemented by using analogue filters to remove any higher-frequency components to eliminate artefacts in the signal.

One concept at the core of digital signal processing is IQ sampling, or quadrature sampling. By taking a carrier signal at a given frequency and sampling the components of the incoming signal that are in phase (I) and 90° out of phase (quadrature, Q) with the carrier before sampling, the incoming signal is downsampled, shifting the carrier frequency to 0 Hz [57]. This results in a captured bandwidth of S Hz around the carrier (for a sample rate of S samples per second), rather than 0 Hz, as demonstrated in Figure 2.1. This significantly reduces the sample

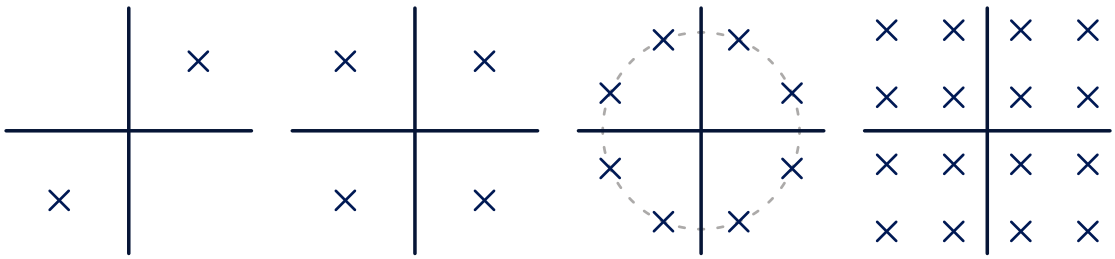


Figure 2.2: Illustration of the distribution of points under different digital modulation schemes. From left to right: BPSK, QPSK, 8PSK, 16QAM.

rate required compared to direct sampling, enabling significantly more effective data collection. Most commercially available software-defined radio hardware uses this technique, including the hardware used for our data collection and attack simulation in later chapters.

One added benefit of sampling in this way is that samples can be represented as complex numbers, and plotted on the complex plane. In this representation, distance from the origin represents amplitude and angle from the horizontal axis represents phase relative to the carrier signal. This makes it a particularly useful representation of certain modulation schemes like Phase Shift Keying (PSK), discussed below. In this case, symbols appear as distinct points on the complex plane, producing a constellation diagram, illustrated in Figure 2.2. We use these constellations to extract transmitter characteristics in Chapter 3.

2.2.2 Signal Modulation

In order to transmit or receive communication using radio signals, the data must be converted into a signal of the desired frequency, using a chosen carrier frequency as the base. Usually the data to be transmitted is a digital signal, or is at a significantly lower frequency than the carrier (e.g. sound waves). In either case, the data must be modulated onto the carrier frequency to enable it to be transmitted. This can be done with both analogue and digital signals, using different methods of modulation in each case.

This thesis primarily focuses on digital signals, for which there exist a range of modulation schemes, including:

- Amplitude Shift Keying (ASK): the amplitude of the carrier is adjusted to one of a finite number of amplitudes;
- Frequency Shift Keying (FSK): the frequency relative to the carrier is shifted to one of a finite number of frequencies;
- Phase Shift Keying (PSK): the phase of the carrier is shifted to one of a finite number of values;
- Quadrature Amplitude Modulation (QAM): the amplitude of the in-phase and quadrature components of the carrier are varied independently – or equivalently, the amplitude and phase of the carrier is varied – once again with a finite number of values.

Each of the above digital modulation schemes encodes a fixed number of symbols, which have a mapping to bits in the data stream. A larger symbol alphabet allows each individual symbol to encode more information – for instance, 4-PSK (or QPSK) has 4 symbols and can therefore encode 2 bits per symbol, but 2-PSK (or BPSK) only has 2 symbols and encodes only 1 bit per symbol. This increase in data density comes at the cost of decreased resilience against disruption; in denser constellations, it takes less energy or noise to switch between symbols in the constellation. QAM is commonly used with dense constellations for higher throughput, with some systems capable of adjusting constellation density to adapt to changes in channel conditions [58]. The majority of signals evaluated in this thesis are PSK modulated, but the techniques are largely transferable between modulation schemes.

If we achieve phase synchronisation with the carrier and sample at the exact symbol rate of the modulation scheme, we can see distinct points on the constellation diagram (see Figure 2.2). If the signal is captured at a higher sample rate (i.e. oversampled), we start to see points between the symbols, as the transmitter hardware modulates between them. The appearance of this interpolation between points is affected by a number of factors including atmospheric noise and multipath distortion, but also by small variations in the transmitter hardware – we use features present in the modulation to identify the transmitter in Chapter 3.

Analogue modulation schemes are commonly used to transmit audio and other analogue data, and are therefore not used as much in satellite communication. The commonly used analogue modulation schemes largely match their digital counterparts, including the following:

- Amplitude Modulation (AM): the amplitude of the carrier is varied to match the amplitude of the information;
- Frequency Modulation (FM): the information signal is encoded by varying the frequency of the transmitted signal relative to the carrier;
- Phase Modulation (PM): the phase of the carrier (relative to its initial phase) is varied to encode the signal;
- Single-Sideband modulation (SSB): the amplitude is modulated, but one of the sidebands is eliminated to reduce power requirements.

2.2.3 Signal Processing

Signal processing refers to a broad range of both hardware and software techniques. Appendix A provides an overview of these concepts and their usage in satellite systems, looking at frequency band allocation, antenna design, frequency filtering, amplifiers, attenuators, and digital signal processing.

2.2.4 Software Defined Radios

A Software Defined Radio (SDR) takes a signal processing pipeline, traditionally built using analogue hardware, and implements the majority of the pipeline in software instead, providing a more versatile platform for signal processing. A typical SDR is composed of an input to be connected to an antenna and suitable filters and amplifiers, followed by an LNA, tuner, and Analogue-to-Digital Converter (ADC), as illustrated in Figure 2.1. Equivalent hardware is used to convert the signal back to analogue form and remodulate it onto the tuned carrier frequency. Everything following the analogue-to-digital conversion is performed in software, including further conversion and filtering, and protocol decoding or encoding. The

result is a radio that can receive or transmit a wide range of different protocols depending on the software implementation, and can usually be tuned to a wide range of frequencies. This provides significant versatility to signal processing over traditional radio hardware, at the cost of additional processing power.

In recent years off-the-shelf SDRs have become significantly cheaper and more widely available, with dramatically increased capabilities, with devices like the HackRF One able to transmit and receive in frequencies ranging from 1 MHz to 6 GHz [59], and costing only 229 GBP.[2] As a result, the ability to carry out physical layer attacks (particularly spoofing, jamming, and replay), once exclusive to large-budget organisations and nation-state actors, is now within reach for even motivated hobbyists.

2.3 Attacks on Satellite Systems

As satellite systems become more ubiquitous in sensing, communication, scientific experimentation, and more, they are becoming increasingly critical components of national and international infrastructure. As a result, they have become a highly lucrative target for attacks. This development has been driven particularly by the aforementioned increase in availability of off-the-shelf SDR hardware, enabling motivated hobbyists to carry out attacks on radio systems that were previously limited to nation-state level actors.

The widespread accessibility of SDRs has led to the proliferation of a range of attacks on satellite systems, including eavesdropping, jamming, and spoofing. Additionally, more sophisticated attacks have emerged, such as message delay attacks on positioning systems, and broadcast intrusion attacks on television systems. Finally, there are are some attacks which do not rely upon the physical layer – in particular, attacks on the satellite supply chain.

[2] Price from Martin Lynch & Sons, accurate as of 2025.

2.3.1 Eavesdropping

Eavesdropping is a significant threat in satellite communications, where all information is transmitted over the air and can be intercepted by any attacker present in the beam. The risk is particularly pronounced in systems that lack robust encryption, enabling unauthorised parties to listen in on all communication.

Eavesdropping is most common on the downlink, due to the wide beam footprint of most satellites. Uplink eavesdropping is less common but not impossible, as it requires the attacker to place a device in a sidelobe of the transmitting antenna or within the main beam by use of a drone or other technology.

Limited value can be gained from eavesdropping if communication is properly encrypted, but the legacy nature of many satellite systems mean this is often not the case. For example, the "VSAT" system popular in maritime operations sends huge amounts of sensitive information without encryption, including personal details and credit card information [60]. The wide footprint of the satellites enables attackers to eavesdrop from hundreds of miles away. Even in systems that have cryptographic security in some form, keys can be leaked, cracked, or reverse engineered, opening them up to attack [10, 11, 61]. Finally, some systems like NASA's Earth Observing System are open by design, deliberately lacking cryptography [62].

Beamforming techniques can be used to mitigate eavesdropping attacks by reducing the width of the beam, making it much harder for an attacker to place their hardware in the receiving path. Some newer systems have also been testing optical communication, further increasing the difficulty of eavesdropping – however, this technology is still in its infancy and has only been tested in a handful of systems [63]. Optical communication is particularly useful for the inter-satellite links used in some satellite constellations, since the position of other satellites can be predicted with high precision and there is little to no atmospheric attenuation or optical distortion.

2.3.2 Jamming

Jamming attacks involve deliberate disruption of satellite communication, by overwhelming the victim signal with noise or other interference. This can be

performed on the uplink or the downlink and prevents the communication channel from being used. This may be done continuously for complete denial of service, or focus on specific messages or users to provide more targeted disruption. Some jamming techniques disrupt only small portions of the message, in order to cause the decoder to fail while using less power. Finally, reactive jamming techniques can be used for real-time disruption of communication, by partially decoding a message before deciding whether or not to transmit the jamming signal [64, 65].

Commonly utilised jamming waveforms include Gaussian noise (sometimes called barrage jamming), a single frequency (tone jamming), sweeping over a range of frequencies, or transmitting a modulated waveform that resembles the victim signal [66]. There has also been some work looking at protocol-aware jamming, which takes the victim modulation into account to maximise impact while remaining difficult to detect [67]. Jamming signals may be deployed constantly or pulsed, and may cover a narrow range of frequencies or a wider spectrum depending on the desired outcome.

To counteract jamming, satellite systems can implement frequency hopping or spread spectrum techniques, increasing the required bandwidth for the attacker to jam communication. Some systems also implement Adaptive Coding and Modulation (ACM), in which the modulation scheme and coding rate are adjusted based on link conditions [68]. When jamming or noise is detected, the system switches to more robust modulation techniques to maintain link integrity. This allows performance to adapt to changing channel conditions, but does risk opening the system up to further attacks – by manipulating feedback mechanisms, the attacker can cause the wrong modulation or coding scheme to be selected, reducing data rates or enabling easier jamming and spoofing attacks [69].

2.3.3 Spoofing

In spoofing attacks, an attacker impersonates a legitimate satellite or ground station with the goal of deceiving the target system. By transmitting false messages which

mimic those of a genuine transmitter, the attacker can introduce inaccurate data, manipulate system operations, or even inject unauthorised commands.

Spoofing relies on the attacker's ability to accurately recreate the physical and protocol characteristics of the legitimate signal, and sometimes to exploit vulnerabilities in the authentication and encryption schemes utilised by the satellite system. Weak or absent cryptographic protection can also allow the attacker to record and replay messages multiple times, causing originally legitimate actions to be repeated on demand.

The impact of a spoofing attack varies depending on its scope and the system targeted. Spoofing can lead to errors or misdirection in navigation systems like GPS [70], injection of falsified data into datasets used for scientific monitoring and Earth observation [9], triggering of further exploits in the message decoder [9], or execution of malicious control commands. This can endanger both the satellite and the ground system, in addition to any downstream systems which rely upon them.

With modern off-the-shelf SDR hardware capable of spoofing satellite downlink messages from a distance of multiple kilometres [24], detection and prevention of spoofing is critical. When properly implemented, cryptographic authentication can prevent the vast majority of spoofing attacks by ensuring only valid messages are accepted by the receiver. To supplement this, or in cases where cryptography is not possible, physical layer techniques can be used. These include:

- **Data Inspection:** The integrity of data can be verified by receiving the same transmission at multiple locations, and comparing the data or its hash between receivers. This requires an attacker to be physically present at each location in order to successfully carry out spoofing attacks, where they would have previously only needed to be at a single location. This is particularly useful in cases where there are already large numbers of community-operated ground stations, such as with NASA's Direct Readout Laboratory (168 operated worldwide at the time of writing) [71]. However, it is likely to be infeasible for smaller organisations to set up multiple ground stations.

- **Timing Analysis:** These techniques involve looking at the timing of signals in order to verify their legitimacy. At the simplest level, this could be ensuring the signal is received at a time the satellite was known to be transmitting, but this does not provide much real security. More advanced techniques in this area include Time Difference of Arrival (TDOA) analysis, looking at the time difference between multiple receivers to verify the transmitter's location against where the satellite is known to be [72]. This is an effective technique, forcing any potential attackers to be physically present at every ground station, but is once again only feasible for larger organisations capable of operating many ground stations. Distance bounding techniques may also be adapted to work in the context of satellites, although these are typically more effective over short distances [73].

- **Signal Analysis:** Alongside data and timing analysis, the waveform itself can be inspected to detect attacks. A spoofed signal is likely to have different properties from the legitimate signal, particularly when spoofing requires overshadowing an ongoing transmission. These properties include amplitude, phase shift, SNR, Doppler shift, and signal distortion [74, 75]. With appropriate radio hardware it is possible to verify these parameters, making attacks more difficult to execute – the adversary must replicate the measured properties in order to successfully spoof messages. Physical layer fingerprinting also falls into this category; we discuss this further in Section 2.4.2 and again in Chapter 3.

Each of these techniques raises the bar for the attacker by making it harder to perfectly spoof messages, and can present a strong defence against attacks when properly deployed and combined.

2.3.4 Time Spoofing

Time spoofing attacks cause disruption via the manipulation of message delivery times, by either delaying or advancing transmissions. By jamming a message

and replaying it at a different time, it is possible to alter behaviour without violating any cryptographic properties of the system. For example, by precisely timing message replays it is possible to cause Global Navigation Satellite Systems (GNSS) to misreport the location of the receiver, giving instead an attacker-specified location [76, 77]. Since these attacks are carried out by simply introducing delay to messages rather than altering message contents, conventional cryptography does not protect against them. As a result, countermeasures must rely primarily upon the physical layer.

2.3.5 Broadcast Signal Intrusion

Satellite broadcast signal intrusion attacks involve the unauthorised interference or hijacking of a broadcast signal (often video) transmitted from a satellite. By overpowering the legitimate signal, the attacker is able to insert their own data stream, overlaying counterfeit programming, inserting false information, or even replacing the entire stream. Television networks are a particularly high-value target for these types of attack due to their wide reach and significant impact.

These attacks are typically carried out by using a directional antenna to overshadow the uplink signal from a broadcast station, or by gaining access to the transmitter or station and replacing the stream directly. Notable incidents include the 1986 “Captain Midnight” and 1987 “Max Headroom” attacks, as well as the 2024 Russian hijacking of the “BabyTV” channel to broadcast propaganda [7, 78, 79].

2.3.6 Supply Chain

Satellite launches involve a large number of organisations and contractors due to the complex nature of their deployment. Many people are involved end-to-end in developing software and hardware, testing, regulation, launch, ground station construction, and ongoing operation. This is particularly true in multi-tenant satellite systems, in which multiple payloads from different organisations are launched on a single satellite, or ride-shared launches, in which multiple satellites share the same launch vehicle. Given the challenges of managing such a diverse

supply chain, it is highly likely that vulnerabilities are present (or can be introduced) in some stage of the process.

Attackers may seek to corrupt or intercept access credentials, overwrite software or firmware, or manipulate data transmissions to disrupt operations. By interfacing with onboard hardware, the attacker may also damage payloads, exceed communication restrictions, or alter the satellite's orbit to cause permanent disruption or even destroy the satellite.

These threats can arise through both software- and hardware-based vulnerabilities. Software attacks may take advantage of exploits or bugs in the software or firmware, or extract (or insert) keys and other important data prior to launch. These attacks pose a particular threat to low-budget systems which tend to rely more heavily upon off-the-shelf components, thereby exposing them to known exploits affecting these platforms.

On the hardware side, attackers with physical access to the satellite during testing or pre-launch phases can introduce malicious components, or swap out genuine parts for ones of their own design. These can be activated after launch to compromise system functionality, target other benign satellites or payloads in rideshared or multi-tenant systems, or exceed regulatory limits [80]. In many cases, these vulnerabilities are very difficult to rectify or even detect post-launch, since operators no longer have physical access to the satellite to verify its integrity.

Software issues may be mitigated through hardware attestation [81] and robust update mechanisms, provided these systems have not also been compromised. However, addressing hardware faults remains far more challenging, especially if malicious hardware has already been integrated into the satellite. Physical layer monitoring might help in detecting the presence of unwanted hardware, and implementing redundant backup systems – especially for ground systems – can ensure continued service even if parts of the system are compromised.

A number of these attacks have already been seen in the wild. For example, the Russian "Kosmos-2558" satellite launched in 2022 has been accused of spying on and potentially shooting down other countries' satellites [82]. In another incident in 2022,

the "AcidRain" malware exploited known vulnerabilities in VPN software to disable Viasat's KA-SAT ground modems, coinciding with the start of the Russian invasion of Ukraine [83]. These events emphasise not only the technical risks associated with the supply chain, but also the geopolitical impact of disruption to the space segment.

2.3.7 Attacker Model

Finally, we define a threat model, focusing in particular on the capabilities of an attacker who wishes to carry out a range of attacks on satellite systems, using off-the-shelf hardware. This will be used as a baseline and expanded upon in later chapters.

Motivation and Goals

Attackers generally fall into one of the following categories:

- **Hobbyist:** These attackers seek to eavesdrop on or disrupt systems out of personal interest, rather than a greater vendetta. These attackers are characterised by a significantly limited budget, but often a greater willingness to build their own hardware rather than relying upon off-the-shelf solutions. Hobbyists are often interested in eavesdropping on communication, but may also wish to disrupt communication or gain access to systems.

- **Nation-state actor:** Backed by a government, non-government organisation, or terror group, these attackers are tasked with gathering intelligence on rival actors, or disrupting service via jamming communication or compromising hardware and datasets. These are the most well-equipped attackers, with a much greater budget and access to hardware.

Despite their varied motivations, goals, and budgets, these attackers are surprisingly similar in the mechanisms in which they carry out attacks: by eavesdropping on communication, jamming messages to disrupt service, spoofing telecommand data to compromise satellite hardware, or spoofing telemetry and payload data to falsify information.

Table 2.1: Three example hardware configurations which could be used to carry out attacks on satellite communication systems.[3]

Component	Model number	Cost (GBP)	Power/Gain
SDR	USRP N210, UBX 40 daughterboard	4783	
Antenna/Amplifier	Beam RST740[A]	1160	12.3 dBW
Total	**High-budget integrated**	5943	12.3 dBW
SDR	USRP N210, UBX 40 daughterboard	4783	
Amplifier	Mini Circuits ZHL-10W-2G+	1374	11 dBW
Antenna	Log-Periodic Ultra Wideband	10	5 dB
Total	**High-budget custom**	6167	16 dBW
SDR	HackRF One	229	
Amplifier	Mini Circuits ZFL-11AD+	96	−33.5 dBW
Antenna	Log-Periodic Ultra Wideband	10	5 dB
Total	**Low-budget**	335	−28.5 dBW

A: This antenna is purpose-built for Iridium systems; an attacker wishing to target other systems could purchase a different antenna with similar specifications, or use the custom configuration.

Capabilities

We therefore assume a general attacker with access to off-the-shelf SDR hardware within a given budget, enabling eavesdropping, jamming, and spoofing over a wide range of frequencies. Such hardware is widely available at low cost, although their particular capabilities vary by cost. We separate potential attackers into two budget classes: high-budget, with access to a more expensive SDR and amplifier, and low-budget, using cheaper alternatives. Three example hardware configurations for these attackers are given in Table 2.1: two high-budget (one using an active antenna with integrated amplifier, and the other with separate antenna/amplifier), and one low-budget.

Both hardware classes are capable of transmitting and receiving messages, with two important differences: the high-budget configuration transmits at a higher power and covers a wider bandwidth, enabling an attacker to carry out attacks from a greater distance, or cover a greater range of frequencies via jamming. Physical layer attacks will still be possible using either hardware configuration, and the increased distance granted by the use of a higher power amplifier can be calculated

[3]Prices are accurate as of 2025.

by comparing the free-space path loss of the two configurations:

$$\text{FSPL} = \left(\frac{4\pi df}{c}\right)^2$$

where d is the distance between transmitter and receiver (m), f is the frequency (Hz), and c is the speed of light (m/s). This can also be expressed in decibels:

$$\text{FSPL}_{\text{dB}} = 20\log_{10}(d_{\text{m}}) + 20\log_{10}(f_{\text{Hz}}) - 147.55$$
$$\text{FSPL}_{\text{dB}} = 20\log_{10}(d_{\text{m}}) + 20\log_{10}(f_{\text{MHz}}) - 27.55$$

At Iridium's frequency (approximately 1626 MHz [31]) and using the configurations above, a high-budget attacker transmitting from a distance of 1000 m would be received with the same signal strength as a low-budget attacker from a distance of 6 m. Of course, other factors will also affect signal quality (e.g. receiver antenna, multipathing, background noise), but this provides a good approximation of attacker capabilities.

For even higher budgets (e.g. nation-state actors) there are a number of possibilities:

- Utilise multiple copies of the same hardware, allowing for multiple receivers to be targeted at once or for a wider area to be covered.
- Obtain amplifiers with even higher transmit power, enabling a single transmitter to cover a wider area, or to target a particular receiver from a greater distance.
- Purchase an Arbitrary Waveform Generator (AWG), which can recreate signals at significantly higher sample rates and with much greater fidelity.

Since the primary effect of the first two options is simply to increase the scale of attacks, they are not considered further in this thesis. We briefly address the possibility of AWGs in Chapter 3, but consider them largely out of scope due to their dramatically increased cost; simply protecting against SDR-equipped attackers mitigates the vast majority of threats.

2.4 Countermeasures to Attacks

Given the critical position of satellite systems within modern global infrastructure and the widespread threat of attacks, robust security measures are essential. In this section we build upon discussion of countermeasures in the previous section, exploring in more detail the implementation and challenges of authentication and encryption as satellite networks grow in scale and complexity. We also introduce physical layer fingerprinting as a method by which transmitters can be authenticated in the absence of cryptography.

2.4.1 Cryptography

Cryptography is a crucial component of satellite communication security, providing mechanisms by which commands can be authenticated and communication can be encrypted.

There are two primary approaches to cryptography: symmetric and asymmetric [84].

Symmetric cryptography uses the same key at both the sender and receiver, for both encryption and decryption. The resulting operation is fast and efficient, but requires the secret to be shared between both (or all) communicating parties. This can be a challenge in satellite networks where keys may need to be updated or rotated frequently, and does not scale well to large numbers of devices.

On the other hand, asymmetric cryptography uses a pair of keys: a public key for encryption, and a private key for decryption.[4] This enables secure communication without the need for a shared key, making it ideal for large-scale networks where key management can be complex. However, this usually comes at the cost of increased computational cost on cryptographic operations, so asymmetric keys are often used to establish a symmetric "session key" which is used for the remainder of a session.

[4]The private key may also be used to "sign" data, authenticating the origin of the message.

Public Key Infrastructure

A Public Key Infrastructure (PKI) attaches identities to asymmetric keypairs in order to secure communication in larger networks. This is achieved through certificates provided by Certificate Authorities (CAs) which verify a keypair's identity. A network participant can check a keypair's legitimacy by verifying its certificate, signed by a trusted CA which is part of a signature chain established by the root CA. This system is near-universal in the terrestrial internet, using the X.509 standard [85].

On receipt of a signed message, the recipient must check the certificate has not been revoked. This can be achieved using Certificate Revocation Lists (CRLs), listing all revocations, or using the Online Certificate Status Protocol (OCSP), which allows devices to query the CA about the status of certificates [86]. The "OCSP Stapling" variant shifts this burden to the sender, who queries the CA and attaches the response to the message, reducing the risk of denial-of-service attacks and the number of handshakes required [87].

As discussed in Section 2.1.2, satellite networks (and particularly interplanetary networks) are broadly considered to be Delay-Tolerant Networks (DTNs), due to their sporadic high-latency links between segments. Key management in these networks is considered to be an unsolved problem [88], with the majority of research focusing on the unpredictable movement and connectivity of the network, making use of techniques like gossip protocols and web of trust [89–94]. Traditional methods like CRLs and OCSP are often dismissed as infeasible due to bandwidth and latency constraints, leading to the use of probabilistic techniques for key distribution. However, in Chapter 5 we show that these assumptions do not hold for all DTNs, and that the predictable links and network topology of interplanetary networks enables many terrestrial PKI concepts to be effective.

Current Systems

Many current satellite systems opt to use pre-shared encryption keys. This is significantly simpler to deploy in small systems, as keys can be loaded onto the satellite before launch, but does not scale well to systems with more satellites or

Table 2.2: Comparison of the different network simulators currently in use.

Simulator						
Name	Origin	Language	License	Summary	Features	Limitations
ns-3 [97]	NSF	C++	GPLv2	Discrete event simulator	Extensible, open, wide range of protocols	Does not scale well, simulates full network stack
SNS3 [98]	Magister/ESA	C++	GPLv3	Extension to ns-3	DVB-S2 and spot beam simulation	Narrow scope
Hypatia [99]	ETH Zürich	C++/Python	GPLv2/MIT	Extension to ns-3	LEO constellation mobility	Only supports certain topologies, lacks dynamic ISLs
ONE Simulator [100]	TKK	Java	GPLv3	DTN network simulator	Lightweight	Random mobility not suited to satellites, no longer maintained
OMNeT++ [101]	OpenSim Ltd.	C++	Academic Public License	Discrete event simulation framework	Large existing body of research/tools/libraries	Lacks native satellite support, hard to isolate components of the network stack, limited scalability
NOS3 [102]	NASA	C	NASA Open Source Agreement	Small satellite operational simulator	Simulates flight and ground software with high fidelity	Single instance, lacks complex networking support
SpaceSecLab/ NSE2 [103]	ESA	—	Not yet open source	Integrated simulator, containerised satellite simulator	Realistic simulation, configurable, possible integration with real hardware	Containers do not scale well, not yet open source

shared operation. It also presents additional risks in cases where keys are leaked, broken, or reverse engineered (see the GK-2A and COMS-1 satellites [10, 11]). Some systems support over-the-air rekeying in this case, using tightly controlled keys to authenticate the operation of replacing session keys [95, 96].

The majority of current satellite constellations are privately owned (e.g. Starlink), so less is known about their internal operation. The Iridium constellation uses pre-shared keys stored in the SIM card, with no mechanism for updating keys short of rewriting or replacing the SIM [31]. The key management of the satellites themselves is not known.

Network Simulation

In evaluating PKI approaches and other protocols, network simulators are commonly used. These enable protocols to be tested in networks much larger than would otherwise be possible, under a wide range of configurations, without risking any damage to real-world systems. Depending on the use case, simulations may be left purely in the software domain, paired with hardware simulation, or connected

to real-world hardware. In Table 2.2 we outline the current state of the art in network simulators, both general and satellite-specific.

A number of network simulators are already in use: in particular, ns-3 [97], the ONE Simulator [100], and OMNeT++ [101]. However, each of these come with their downsides – ns-3 and OMNeT++ do not scale well to large numbers of nodes with lots of connections, and simulate the entire network stack, which is not necessary in many cases. The ONE Simulator is designed with DTNs in mind and is significantly more lightweight, but is no longer maintained and does not natively support simulating satellite mobility. There also exists an extension to ns-3 for simulating satellite networks [99], but due to the limitations of ns-3 it does not support dynamic links based on proximity or other factors, which are crucial for simulating federated networks and other novel topologies.

More recently, NASA have released their "NOS3" simulator, designed to be a satellite digital twin for developing and testing onboard software [102]. Although highly useful for this purpose, it is not possible to run at scale to test network protocols between many nodes. ESA are also planning to release their "Network Simulation and Emulation Environment (NSE2)" as part of their "SpaceSecLab" [103], providing a Docker-based network simulation that can support highly realistic emulation of the network stack and connectivity for small numbers of nodes. This will be useful for testing implementation-specific details and small-scale mission control, but is not as useful for large satellite networks.

In Chapter 5 we design our own simulation system to address some of the limitations in existing offerings: the Deep Space Network Simulator (DSNS). This simulator can scale to arbitrary numbers of nodes, with highly realistic simulation of network topology and connectivity. It is also highly extensible due to its pattern matching approach to modelling behaviour – in order to add functionality, developers simply add rules matching the desired events or messages and define the system's response. In Section 5.6 we describe the design and features of DSNS in greater detail, and in Section 5.7 we test a number of terrestrial PKI

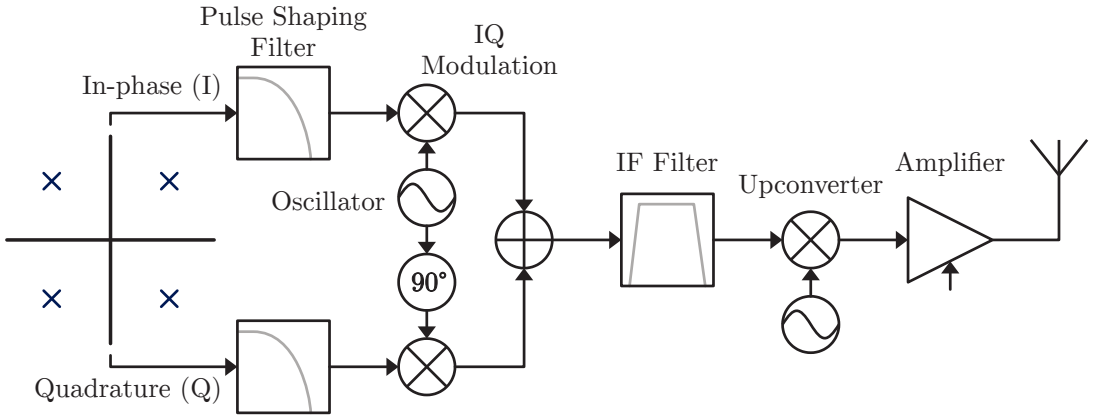


Figure 2.3: Diagram of an RF signal modulation pipeline when using IQ modulation, and the hardware components involved.

approaches, demonstrating good performance in interplanetary networks when properly configured.

Of course, network simulation is not only useful in the realm of PKI; it also facilitates interesting research in a number of related areas, by enabling operators and standards bodies to easily test the performance of different protocols under different configurations and network topologies. One particularly useful area is routing: the complexity of future satellite networks means this task is not trivial. Although connections are sporadic, they can be predicted ahead of time, so discovery-based mechanisms are not required except in the case of unexpected link loss. In Section 5.8 we discuss the usefulness of DSNS for this and other open research topics.

2.4.2 Physical Layer Fingerprinting

In the absence of cryptographic authentication, physical layer techniques can be used to provide some measure of authenticity by verifying the origin of communication. One such technique is fingerprinting; that is, extracting characteristics of the transmitter as expressed in the physical layer signal. These can be compared to previous messages to verify that they match, and that the message was therefore likely sent by the legitimate transmitter and not an attacker with an SDR.

Fingerprinting techniques take advantage of the fact that RF signal modulation involves a large number of physical components; this can be seen in Figure 2.3.

Each of these can cause distortions on the signal unique to the hardware, due to small imperfections in the manufacturing process. Fingerprinting techniques can identify transmitters based on a number of features, including [104]:

- Carrier frequency offset between the transmitter and receiver, resulting in drift of the symbols;
- Nonlinearities and distortion in the amplifier;
- Noise in the phase domain introduced by the oscillator;
- Imbalance between the in-phase (I) and quadrature (Q) portions of the signal, skewing the constellation.

Radio fingerprinting is a mature field, with a large base of research looking at a wide range of techniques on many different systems – [105] provides a good overview of existing research. It has already been applied to a wide range of terrestrial systems including RFID [106], the ADS-B air traffic control system [26], Bluetooth [107], WiFi [108], and satellites – we will discuss fingerprinting in a satellite-specific context in Section 3.2. Fingerprinting techniques can be partitioned into two key areas: transient fingerprinting and steady-state fingerprinting.

Transient Fingerprinting

The transient of a radio signal occurs when the transmitter first powers on, or changes power levels following a signal lock. Various properties of the transient, such as its duration or the number of peaks which occur in the carrier signal, are characteristic to the transmitter and can be used to identify it, if properly extracted. The majority of historical fingerprinting research makes use of transient analysis, since almost all radio transmission involves a transient, and the transient typically exhibits the same characteristics every time the device powers up. Much of the work in transient fingerprinting revolves around novel techniques for precisely identifying the start and end of the transient [109, 110], or processing the transient to extract useful identifying features [111].

Transient fingerprinting has seen some use in security contexts – in [112], it is used to identify devices even when all other identifying information has been removed, and to detect wormhole and device cloning attacks.

Steady-state Fingerprinting

In contrast to transient fingerprinting looking at a very brief portion of the signal, steady-state fingerprinting instead looks at the modulated portion of the signal. The features in the steady-state portion of the signal are different from the transient, often looking at how the IQ constellation is affected by hardware impairments. These impairments include quadrature errors, self-interference, amplitude clipping, and frequency offsets in the various stages of signal modulation (Digital-to-Analogue Converter (DAC), mixer, filter, upconverter, amplifier) [113]. Additionally, the signal is affected by properties of the wireless channel, including background noise, free space path loss, and multipath distortion, which are likely to change over time as the environment changes. These wireless channel properties are significantly more prominent in satellite communication, due to the long-distance radio links involved. Very good performance has been achieved using fingerprinting techniques to authenticate devices using over-the-wire communication [26], but these have much lower levels of noise on the channel, making the problem significantly easier.

There is some variety in the techniques used, including averaging multiple messages to eliminate or reduce noise [114, 115], analysing features in the frequency domain [116], and looking at high sample rate signals to observe high frequency impairments [117]. Machine learning techniques are more commonly used to aid steady-state fingerprinting, particularly in approaches working at a high sample rate, in order to pull out features which may not be immediately obvious. Manual feature engineering is also used, but is less common than in transient analysis.

The majority of steady-state fingerprinting looks at a single protocol or class of devices at a time, but with the recent rapid increase in machine learning capabilities this is no longer a requirement, and there has been some work into

generalisable fingerprinting techniques which do not require retraining to apply to new contexts [118].

There are also some interesting techniques extending the concept of fingerprinting – the authors of [119] train a convolutional neural network to identify SDRs at 5 MS/s, then intentionally introduce signal impairments at the transmitter in order to further increase classification accuracy. This achieves incredible accuracy (greater than 0.995), but fingerprint forgery is not considered in this context.

Implementation Considerations

When implementing fingerprinting in satellite systems, the primary use case is in protection against spoofing and replay attacks, particularly from SDR-equipped adversaries. Adding fingerprint-based authentication to a system's downlink can verify that messages have come from the expected satellite, rather than from an SDR or other unauthorised transmitter, thus providing additional protection – particularly in the absence of cryptographic authentication. Although prior work has shown that an arbitrary waveform generator with a sufficiently high sample rate can successfully impersonate device fingerprints [120], this hardware is prohibitively expensive for the majority of adversaries. We discuss our own approach to satellite fingerprinting in Chapter 3 and demonstrate its effectiveness at high sample rates, thus excluding the vast majority of lower-budget attackers.

Although more difficult to implement, it is also possible (and potentially useful) to use fingerprinting on the uplink as well. Systems with suitable onboard SDR and computational hardware are likely capable of also executing cryptographic operations, so it is unlikely that fingerprinting would be used by itself in this context for authentication; however, it may still be useful to protect against timing attacks discussed in Section 2.3.4, which do not require the attacker to break the underlying cryptography of the message. Fingerprinting may also be useful for device segmentation purposes: in a system with many users such as Iridium, the satellite operator may wish to identify the user's ground hardware, both for diagnostic and monitoring purposes, as well as to flag any unexpected hardware

changes. Fingerprints can be used in this context to identify or cluster transmitter hardware, rather than to identify the individual transmitter.

Fingerprinting may also be deployed alongside other physical layer authentication techniques, such as looking at SNR or the distribution of symbols in a message, to provide a holistic view of the system's status and a wider range of attack detection capabilities. This can be deployed in systems making use of cryptographic authentication, since jamming and timing-based attacks can still be carried out, and in broadcast systems (e.g. television), where cryptography is often infeasible to implement. Once an attack has been identified, appropriate countermeasures can be deployed to ensure the impact is minimised. We discuss this further in Section 3.8.

3

Authenticating Satellite Downlinks using RF Fingerprinting

Contents

3.1 Motivation 44
3.1.1 Contributions 46
3.2 Related Work 47
3.2.1 Satellite Fingerprinting 47
3.2.2 Security 49
3.2.3 Our Approach 49
3.3 Threat Model 50
3.3.1 Goals 50
3.3.2 Capabilities 51
3.4 System Design 52
3.4.1 Design Decisions 52
3.4.2 Fingerprinting Model 55
3.5 Data Collection 57
3.5.1 Data Preprocessing 62
3.5.2 Dataset Construction 62
3.6 Model Training 63
3.6.1 Initial Results 64
3.7 Evaluation 68
3.7.1 Time Stability 68
3.7.2 Correlations 72
3.7.3 Subsampling 74
3.7.4 Extensibility 75
3.7.5 Transferability 76
3.7.6 Security 79
3.7.7 Comparison with Existing Systems 82
3.7.8 Deployment Considerations 86
3.8 Future Work 88
3.9 Conclusion 90
3.9.1 Availability 90

As satellite systems become a greater part of critical infrastructure, they have become a significantly more appealing target for attacks. With the availability of cheap off-the-shelf radio hardware making signal spoofing and physical layer attacks increasingly accessible, it is essential that effective authentication methods for satellite communication are developed and deployed. This chapter explores the use of radio transmitter fingerprinting as a means to enhance the security of satellite systems. We propose a novel approach for authentication of satellite transmitters using high sample rate characteristics of the transmitter hardware, and evaluate its effectiveness in detecting simple spoofing attacks.

3.1 Motivation

Recent years have seen a dramatic rise in the availability of cheap radio hardware, particularly Software Defined Radios (SDRs). Not only have these devices become more widely available, but their capabilities have increased, with the HackRF One costing 229 GBP and working with frequencies up to 6 GHz [59].[1] This has brought physical layer attacks into the reach of a much wider range of attackers and poses a particular threat to satellite systems, many of which were built under the assumption that tampering with signals would be prohibitively expensive for the vast majority of attackers.

Physical layer attacks have been widely explored in wireless systems – attackers equipped with an SDR can overshadow legitimate communications or spoof messages outside normal communication. Many widely used systems are vulnerable to these attacks, including the ADS-B avionics protocol [121], the LTE telephony system [122, 123], and satellite systems including GPS [124]. Due to the critical nature of satellite systems, it is vital that operators are able to prevent or detect spoofing attacks, in order to protect the systems and applications that rely on them.

There are a wide range of techniques for detection and prevention of spoofing attacks, the foremost of which is cryptography – a properly implemented cryptosystem with associated key management provides robust authentication, making

[1] Price accurate as of 2025.

spoofing attacks near-impossible. However, there are a number of reasons why cryptography may not be desirable (or possible) in the context of satellite systems. Firstly, there are a huge number of legacy satellites currently in orbit. Many of these do not implement cryptography, and cannot be retrofitted to do so due to their limited onboard processing power. However, the data collected by these satellites is immensely useful in both scientific and private use cases – they are used for monitoring forest fires, land usage, population density, flooding, and more [125–128]. These satellites are often bespoke designs which would be prohibitively expensive to replace; it is important to ensure systems like these can be used for their entire projected lifespan (and potentially beyond).

There are also a number of satellite systems which were initially built with cryptography, but which have become insecure post-launch due to leaked keys [11] or outdated cryptosystems [10]. Some of these cannot be patched due to a lack of over-the-air update capabilities, so other methods must be used to authenticate their telemetry data.

Finally, some attacks can be carried out without violating any cryptographic properties of the system. The authors of [77] show that precisely timed message replays can cause Global Navigation Satellite Systems (GNSS) to misreport the location of the receiver to an attacker-specified location. Since these attacks are carried out by simply introducing delay to messages rather than altering message contents, conventional cryptography does not protect against them. We still want to be able to detect and prevent these attacks, so we must turn to non-cryptographic techniques for message authentication. In particular, we investigate radio transmitter fingerprinting, in which radio signal characteristics are used to identify transmitters. This is achieved by identifying impairments on the signal which are created by small differences in the radio transmitter hardware. These impairments are unique to transmitters and consistent over time, as we will show for the satellite case.

3.1.1 Contributions

In this chapter we present SatIQ, a novel approach to fingerprinting satellite signals. We work with signals at a high sample rate to counteract problems with spoofing at lower sample rates. This makes the techniques more useful in a security context, requiring attackers to use more expensive radio hardware that works at high sample rates in order to successfully impersonate a device – thus excluding a large number of low-budget adversaries. In doing this we can provide an additional level of confidence in the authenticity of the origin of satellite signals, particularly in systems where cryptography is either unavailable or ineffective. We verify that the system can detect replay attacks by replaying captured messages using an SDR.

To build SatIQ we use a Siamese model – unlike conventional classifiers, these compare two signals and produce a distance metric representing the likelihood of two messages having been sent from the same transmitter. This technique enables one-shot learning: new transmitters can be introduced without requiring the system to be retrained, and can be used immediately with only a small number of examples. This is particularly useful in the context of satellites in Low Earth Orbit (LEO), which must be replaced more frequently.

In building SatIQ we collect a dataset of 10 290 000 messages from Iridium satellites across 3 different locations using different hardware configurations. Using this data we perform a wide range of analyses, demonstrating that SatIQ is stable over time, extensible to new transmitters and receiver configurations, and performs well irrespective of weather and other conditions. We also look at the performance of SatIQ under a replay attack, showing that it can achieve very good performance in this context, and that fingerprinting is an effective technique for use in legacy satellite systems to detect and prevent spoofing and replay attacks. Finally, we compare SatIQ to the previous state of the art under 3 different attack configurations, showing that SatIQ performs better in all cases.

Table 3.1: Summary of related work in satellite fingerprinting, and fingerprinting for security. Ticks (✓) denote support, dashes (—) indicate that the feature was not discussed or tested.

	Fingerprinting Technique	Satellites	High Sample Rate	Stable over Time	Extensible to New Transmitters	Transferable to New Receivers	No Manual Features	Single Message
Rasmussen et al. [112]	Transient		✓	—	✓	—		✓
Tekbaş et al. [129]	Transient			—	—	—		✓
Kennedy et al. [116]	Steady State			—	—	—		✓
Bassey et al. [117]	Steady State		✓	—	(✓)[A]	—	✓	✓
DeepRadioID [118]	Steady State			—	(✓)[B]	—		✓
ORACLE [119]	Steady State		✓	—		—	(✓)[C]	✓
Wang et al. [114]	Steady State			—		—	(✓)[D]	✓
PAST-AI [115]	Steady State	✓		—		—	✓	
Spotr [113]	Steady State	✓	✓	✓	—	✓		✓
SatIQ	Steady State	✓	✓	✓	✓	✓	✓	✓

A: Claimed to be extensible but not experimentally verified
B: Requires transmitter modifications
C: Some manual features, optional transmitter modifications
D: Manual filtering/smoothing, neural network classifier

3.2 Related Work

We have already explored physical layer fingerprinting in the general case in Section 2.4.2; in this section we focus instead on research that targets satellites specifically, and works that use fingerprinting in a security context. These are summarised in Table 3.1.

3.2.1 Satellite Fingerprinting

Unlike signals from terrestrial devices, satellite communication must travel hundreds of kilometres through the atmosphere, causing significant signal attenuation and channel noise. This adds additional challenge to fingerprinting in this context, particularly since many techniques rely on the minimal presence of background noise. There are some works looking at fingerprinting in the presence of noise, either by adding noise to clean signals during model training (effectively training models to remove/ignore the noise) [129], or by smoothing out long signals at low sample rates to obtain average symbol positions [114]. In the context of satellites it is difficult to obtain signals without noise, and smoothing does not work with high sample rate signals, since important detail is lost – we must find other methods of reducing or ignoring noise. We discuss this further in Section 3.4.

The authors of [115] design "PAST-AI", analysing heatmaps of low sample rate transmissions from the Iridium constellation to classify satellites. This technique achieves an accuracy of approximately 0.85, increasing to nearly 1.00 for small subsets of the constellation. Although this technique achieves high accuracy, it is not as useful from a security context – the classifier works by processing large batches of consecutive messages, making it more difficult to detect individual message spoofing. Furthermore, fingerprinting at a very low sample rate (1 sample per symbol) makes fingerprint forgery significantly easier, since the attacker does not need to replicate as many features of the waveform. The resulting bandwidth of 25 kHz is significantly less than the Iridium channel spacing of 41.667 kHz [31], so transmitter characteristics at higher frequencies will be discarded. The work claims to be able to discriminate SDR-equipped attackers from legitimate satellites by solving the harder problem of discriminating between satellites with the same hardware. This is true, however, the assumption only holds at the sample rate used by the fingerprinting system. As PAST-AI operates at 1 sample per symbol, it cannot protect against SDR-based attacks at higher sample rates. We demonstrate this experimentally in Section 3.7.7.

There has also been some fingerprinting work looking at other satellite systems – the authors of [113] make use of manual feature extraction to identify spoofing of GPS localisation satellites. This technique is effective, but manual feature extraction is less likely to transfer easily to other satellites or constellations.

Alongside this increased interest from an academic perspective, satellite fingerprinting has also gained the attention of the commercial and public sectors: the European Space Agency (ESA) has recently allocated funding to explore applications for fingerprinting in the satellite context [130], and in 2021 Expedition Technology was awarded a contract by DARPA (USA) to expand their RF fingerprinting and spectrum characterisation capabilities [131].

3.2.2 Security

Most fingerprinting is done for the purpose of security, preventing spoofing and replay attacks, but some techniques have been specifically proposed to provide better security properties. For instance, the authors of [117] address the problem of identifying devices which have not been seen in the training dataset by separating the feature extraction and classification components of the model, using clustering techniques on the extracted features to identify transmitters. This allows new transmitters to be introduced without retraining the feature extraction component. We take a similar approach in our work, using an autoencoder to produce the fingerprints and a Siamese model in the place of clustering. The SatIQ system is fully described in Section 3.4.

There has also been work assessing attacks on fingerprinting systems – it has been shown that an arbitrary waveform generator with a sufficiently high sample rate can be used to impersonate devices, fooling fingerprinting systems [120]. Hardware that can achieve these sample rates is prohibitively expensive for the vast majority of adversaries (we discuss budget further in Section 3.3), so we do not consider this to be a major issue – although our techniques are likely vulnerable to impersonation at a very high sample rate, preventing spoofing below a certain transmitter sample rate is sufficient to exclude the vast majority of lower-budget attackers. In Chapter 4 we look at attacks on the SatIQ fingerprinting system itself, testing its resilience against optimised jamming and spoofing and exploring methods by which its resilience can be improved.

3.2.3 Our Approach

Table 3.1 summarises the features supported in each of the related works discussed. We see that our approach provides many desirable features, not all of which are present in other related works. SatIQ is designed to work in the high-noise environment of satellite communication, and works at a high sample rate to prevent spoofing and replay attacks using cheap SDRs from forging the fingerprint. Performance is stable over time, and the system can be trivially extended to

support new transmitters – this is critical in satellite systems, where at any time new satellites may be introduced or existing satellites replaced, and gathering enough data on the new transmitters to retrain the model would be time-consuming. Furthermore, it does not rely upon any manual feature extraction, and operates on individual incoming messages, making it easy to deploy and use. In later sections we describe the architecture of SatIQ that makes these features possible. We also demonstrate its features through experimentation, confirming its performance under attack and its ability to function over time and as new transmitters are introduced. Finally, we compare SatIQ to the existing PAST-AI system, showing that our system is more effective at detecting attacks.

3.3 Threat Model

We start by establishing a threat model used for the remainder of this chapter, based on the general attacker model in Section 2.3.7. By keeping attacker capabilities and hardware consistent between experiments, we ensure our results are consistent with one another and with the state of the art.

3.3.1 Goals

In this chapter we concentrate on attacks involving spoofing and message replay; we assess the vulnerability of fingerprinting techniques to jamming attacks in Chapter 4. In the case of spoofing, the adversary's goal is to broadcast messages appearing to come from a satellite such that the ground system processes them alongside legitimate messages. Alternatively, the attacker may delay or advance messages (jamming the original and replaying a recording) to affect timing-based systems such as GPS [77]. Similarly, they could carry out "wormhole attacks", in which messages are captured at one location and tunnelled to another location, from where they are broadcast – this is also effective against GPS and other localisation systems. Unlike spoofing, these attacks can be performed even on signed or encrypted messages, since they do not affect message contents.

We are primarily concerned with attackers overshadowing messages on the downlink – whilst attacks on the uplink are possible, they have not been extensively explored due to the greater hardware cost of a suitable amplifier and directional dish. Furthermore, device fingerprinting on the space segment is currently infeasible, requiring large amounts of computational power, and cannot be carried out aboard the legacy satellites with which we are primarily concerned. We discuss in Section 3.8 how future work might authenticate the uplink of legacy systems by capturing signals in-transit, and explore other uses for uplink fingerprinting.

3.3.2 Capabilities

As established in Section 2.3.7, we can expect that the attacker has access to off-the-shelf SDR equipment, enabling spoofing or replay attacks within the vicinity of a ground station, with budget primarily affecting the distance from which these attacks can be carried out.

It has been demonstrated in [120] that device fingerprinting is vulnerable to signal replay attacks, provided the attacker has access to a high-end arbitrary waveform generator capable of transmitting signals with a sufficiently high sample rate.[2] We do not consider attackers with this capability, since they are always capable of purchasing hardware that can fool a fingerprinting system. In this chapter we are particularly interested in SDR-based attacks; it has been demonstrated that simple spoofing and replay can have devastating effects on improperly secured satellite systems, and this hardware reflects the capabilities of the vast majority of attackers. Through robust high sample rate device fingerprinting, we aim to defeat these attacks by making it impossible to forge the fingerprint on spoofed signals using only off-the-shelf radio hardware. In doing so, we increase the budget of attacks such that they can no longer be carried out without hardware which is significantly more expensive and harder to obtain.

[2]We approached a reputable manufacturer, and received a quote for approximately 125 000 USD at academic institution rates. We therefore conservatively assume that even in the best case this hardware would cost no less than 60 000 USD.

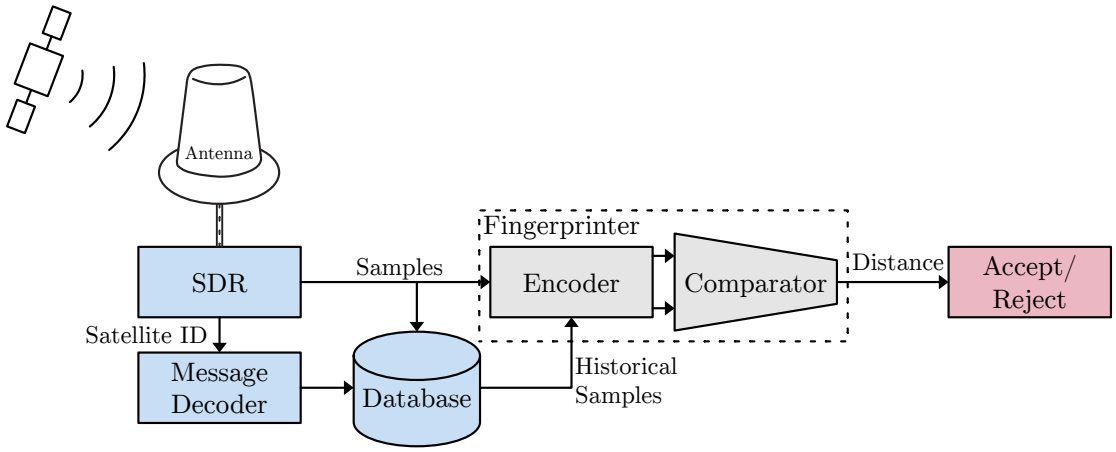


Figure 3.1: An overview of the end-to-end fingerprinting process used by SatIQ. Satellite signals are received, decoded, processed into fingerprints, and compared to historical fingerprints to determine a distance metric, which is used to accept or reject the message.

3.4 System Design

Our system comprises two primary components: data collection, and the SatIQ machine learning fingerprinting system. A representation of the end-to-end system can be seen in Figure 3.1 – messages are collected by an SDR and decoded into message contents, and the raw samples are ran through an encoder network (explained below), and compared against known example messages from the same transmitter. This produces a distance metric – low distances indicate the message is likely to be legitimate. This is used to accept or reject the message.

3.4.1 Design Decisions

Our fingerprinting model, illustrated in Figure 3.2, uses an autoencoder combined with a Siamese model to compare two input waveforms and produce a distance metric between the two inputs. In the following, we describe autoencoders and Siamese networks in general, and explain why we chose them as the basis for our fingerprinting system.

Autoencoders

An autoencoder is a type of neural network which is used to learn an efficient encoding of data. This is achieved by simultaneously training an encoder and

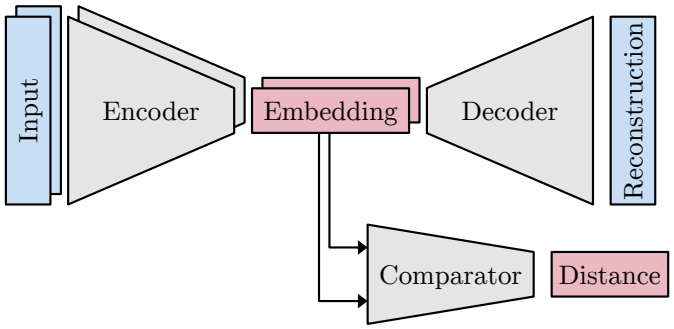


Figure 3.2: An overview of the Siamese autoencoder architecture used in SatIQ's fingerprinting model. Two inputs are passed into the encoder with identical weights, and the encodings are compared using the comparator (angular distance) to generate a distance metric.

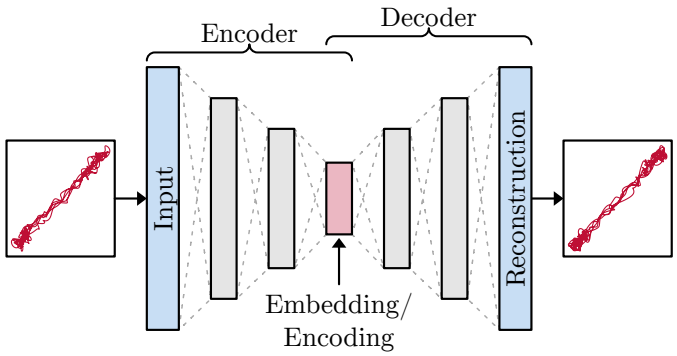


Figure 3.3: The layout of an autoencoder. Input is passed through an encoder to produce a low-dimensional encoding, and then through a decoder to produce output of the same dimension as the input. The model is trained to reconstruct the input as best as possible.

decoder, validating the accuracy of the encoding by comparing the output of the decoder to the input (reconstruction accuracy). The output of the encoder is restricted in size, thus forcing data to pass through a bottleneck at this portion of the network. This forces the model to discard less important information, producing an efficient encoding. This technique is particularly useful for dimensionality reduction, since the encoder produces an embedding of the input in a significantly lower-dimensional space. The layout of an autoencoder can be seen in Figure 3.3. Prior works have shown autoencoders to be effective in fingerprinting contexts, both in terrestrial systems [132, 133] and on satellites specifically [115, 134].

Siamese Neural Networks

Siamese neural networks are designed to be effective for one-shot classification. To this end, they generate a similarity score between two inputs [135]. This is achieved by passing each input through the same "encoder" network to generate an embedding of the inputs in some feature space, followed by a comparison function to generate a distance metric in the feature space – the lower the distance, the more similar the samples are. The weights of the encoder network are shared between the two inputs.

We chose this approach due to its advantages over a simple classifier, particularly in the context of fingerprinting:

- The number of classes is not fixed – new classes can be introduced after training by comparing inputs to examples from the new class.
- The one-shot (or in some cases few-shot) nature of the model means a new class can be identified using only a very small number of examples.
- The distance threshold can be raised to increase the acceptance rate of legitimate messages at the cost of increased false positives (or vice versa), granting fine-grained control of the level of protection granted by fingerprinting.

Past work has shown these models to be effective in a wide range of use cases, particularly in classification problems with huge numbers of classes such as malware detection and gait recognition [136, 137]. The architecture has also been demonstrated to be successful at detecting spoofing and replay attacks on other systems, such as face recognition and voice biometrics [138, 139]. Siamese networks have also seen use in radio systems, shown to be effective in areas such as automatic modulation classification [140]. Finally, some research has shown promise in using Siamese networks for fingerprinting radio transmitters [141, 142].

Our work builds upon these in a number of key aspects. Firstly, in satellite systems we deal with a more difficult scenario than the majority of terrestrial fingerprinting cases – all signals must travel a great distance (hundreds of kilometres or further), introducing large amounts of atmospheric attenuation and multipath

distortion, which dwarf the hardware impairments on the signal. Secondly, we consider in particular the security implications of radio fingerprinting, assessing the level of protection against spoofing and replay attacks granted by our approach, and the expected budget required to circumvent these techniques. Finally, we use an autoencoder alongside the Siamese network to provide encodings that better capture meaningful features in the input – prior work has shown this to be effective in areas such as signature verification, but to our knowledge we are the first to apply this architecture to radio fingerprinting [143].

In a non-adversarial classification setting, simple classifiers typically exhibit higher performance than Siamese networks, since they partition the entire input space into fixed categories. However, this is less effective in adversarial settings, since malicious transmitters will still be given a legitimate label. Our results in Section 3.7 show that Siamese architectures are particularly effective at detecting replay attacks, even with the levels of noise observed in satellite signals, and perform better than the previous state of the art in a direct comparison.

Siamese models can require more data during training, since they must learn a distance metric that works for all transmitters. Furthermore, independently of the model architecture, greater levels of noise mean more training data is required. Our results show that our dataset was sufficient for this work, but more data may yield further performance improvements.

3.4.2 Fingerprinting Model

The Siamese network of SatIQ uses the encoder portion of the autoencoder to produce embeddings of two inputs, which are then compared to one another using an angular distance function. A triplet loss term encourages the model to produce embeddings that are close to one another for messages from the same transmitter, and different for messages from different transmitters:

$$d(u, v) = 1 - \frac{u \cdot v}{\|u\|\|v\|}$$
$$L(a, p, n) = \max\left(d(a, p) - d(a, n) + \alpha, 0\right)$$

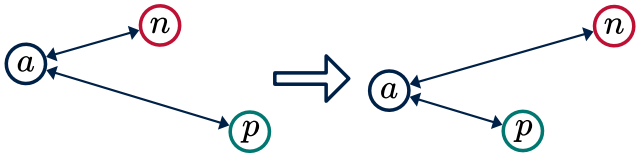


Figure 3.4: An illustration of the triplet loss function. This function takes an anchor a, a positive sample p of the same class as the anchor, and a negative sample n of a different class. Optimising this loss function minimises the distance between the anchor and positive samples, and maximises the distance between the anchor and negative samples.

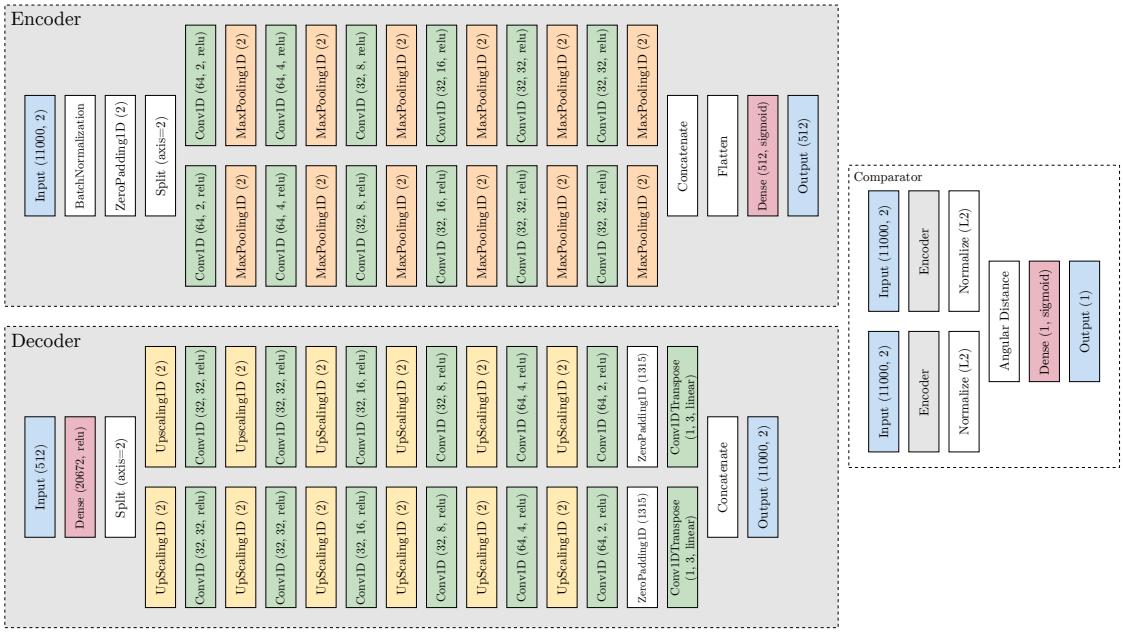


Figure 3.5: The layers of the Siamese neural network used in SatIQ. The encoder uses separate convolutional and max-pooling layers for the I and Q portions of the signal, before producing a final embedding using a dense layer. Similarly, the decoder uses separate convolutional and upsampling layers. The comparator uses two copies of the encoder with identical weights, computing a difference score between the outputs.

This takes an anchor a, a positive example p belonging to the same class as the anchor, and a negative example n of a different class, and encourages the distance from a to p to be less than a to n. A margin α ensures effort is not wasted on optimising triplets for which this is already the case. This is illustrated in Figure 3.4.

Convolutional layers are used in the encoder to enable identification of position-invariant features and reduce the overall size of the model. For the comparator, we compute the angular distance between the embeddings – this tends to work better than L2 distance for high-dimensional data. A diagram of the model architecture can be seen in Figure 3.5.

The inclusion of convolutional layers reduces the model's size, and allows it to extract position-invariant features from the waveform. We also use separate layers for the in-phase (I) and quadrature (Q) portions of the signal – although the components are tightly coupled to one another, we find that they express different features and the model is able to perform better when the two are separated. Following the convolutional layers, we concatenate the outputs and flatten, before using a fully connected layer to reduce the output to the correct size. The decoder uses a very similar architecture to the encoder, composed of alternating upsampling and convolutional layers.

3.5 Data Collection

For a fingerprinting model to be effective, a good dataset is essential. Community projects such as "SatNOGS" (an open-source network of ground stations [52]) provide data from a wide range of satellites. However, for this work we collect our own data for a couple of important reasons: firstly, collecting our own data enables us to capture at a significantly higher sample rate than existing datasets, providing a good foundation for a fingerprinting system. We also capture signals from a specific constellation rather than a collection of individual satellites; by doing this, we ensure that the signal modulation and protocol does not vary between messages. We can therefore guarantee that the message header is always the same between messages, so differences in the captured waveform will be caused only by hardware differences and channel noise – the contents are always identical at the bit level. This gives us a consistent baseline upon which a fingerprinting model can be built.

We focus on the Iridium constellation, used in telecommunications. This constellation has a number of useful properties:

- The constellation contains a large number of satellites (66, each with 48 transmitters [31]), providing sufficient variety within the dataset;

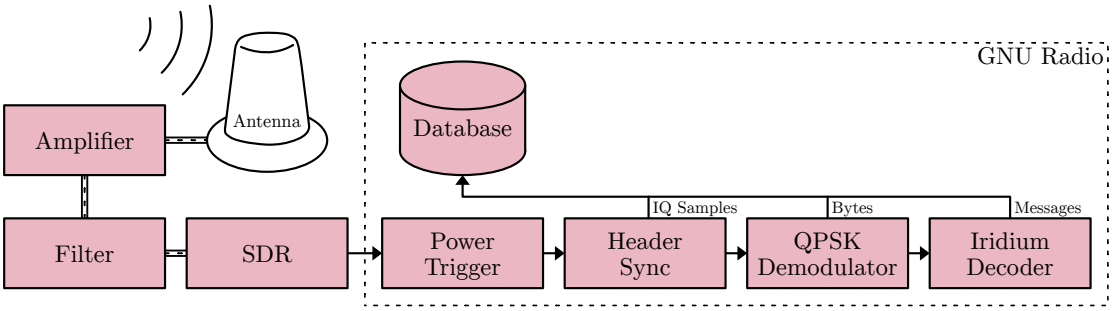


Figure 3.6: An overview of the data collection system. Signals are captured by the SDR and sent to GNURadio for processing. Raw IQ samples, demodulated bytes, and fully decoded message data are all sent to a database.

- The transmitter hardware on each satellite is identical, so a fingerprinter will need to distinguish satellites only through differences introduced at time of manufacture, rather than distinguishing between entirely different components;
- The communication protocols are known and well documented [31], so no reverse engineering is required;
- Downlinked transmissions can be received using cheap and widely available COTS hardware, as it transmits at a frequency below 6 GHz.

Other constellations are available; for instance, the ORBCOMM and Globalstar constellations have open-source decoders available but fewer satellites, and Starlink and Planet's Dove constellations have more satellites but operate at high frequencies and (at the time of writing) have no open-source decoder available. Furthermore, existing research in fingerprinting also uses Iridium satellites [115], providing evidence that some fingerprinting is plausible in this context and providing a baseline to which fingerprinting systems can be compared. We argue that testing on Iridium is sufficient to demonstrate and validate the effectiveness of SatIQ, as the properties of the signal used by SatIQ are not constellation-specific. We discuss in Section 3.8 how the versatility of the SatIQ architecture enables the system to work across different constellations, demonstrating this using the ORBCOMM satellites.

Figure 3.6 illustrates our full data collection and processing pipeline. For the hardware components of this setup, we use the following:[3]

[3]Prices are accurate as of 2025.

- Beam RST740 active antenna (1160 GBP, GTC)
- Mini-Circuits ZKL-33ULN-S+ low-noise amplifier (205 GBP, Mouser)
- NooElec DC block (25 GBP, Amazon)
- Mini-Circuits VBF-1560+, 1500–1620 MHz band-pass filter (43 GBP, Mouser)
- USRP N210 SDR (3154 GBP, DigiKey)
- UBX 40 USRP daughterboard (1629 GBP, DigiKey)

The SDR is connected to a computer running GNU Radio, a software library to aid digital signal processing. We use components from the "gr-iridium" library, created and maintained by the Chaos Computer Club München e.V., to demodulate and decode messages [144].

Iridium downlink messages have a number of different message types, one of which is the Iridium Ring Alert (IRA) message – this is the only type of message we collect. These messages contain diagnostic information about the satellite, including the satellite ID, beam ID (identifier of the current transmitter), position, and altitude. They also contain the anonymous identifiers of subscribers currently receiving incoming paging calls. The messages are openly broadcast and do not contain any personally identifiable information, so we can decode and collect them without raising any ethical concerns.

We save the raw IQ samples from the message headers in a database, alongside demodulated bytes and decoded message contents. This gives us a consistent dataset of raw message headers which can be used for fingerprinting, using the decoded messages to label the data. All data was collected at 25 MS/s, giving us an oversampling rate of 1000 (Iridium messages have a symbol rate of 25 000 symbols per second). This allows us to capture the high-frequency features characteristic to a transmitter for effective fingerprinting.

Our data collection was spread across multiple locations (summarised in Table 3.2), enabling analysis of the effect of receiver hardware and physical location on the performance of a trained fingerprinter. The bulk of the data was collected

Table 3.2: Hardware and dataset information for the data collected at each location.

	Oxford, UK	Thun, Switzerland	Würzburg, Germany
SDR	USRP N210	USRP N210	USRP B205mini-i
Antenna	Beam RST740	Comant CI-490-490	Tallysman TW4600
Amplifier	Mini-Circuits ZKL-33ULN-S+	Qorvo SPF5189Z	Nooelec LaNA
Filter	Mini-Circuits VBF-1560+	Mini-Circuits VBF-1560+	Mini-Circuits VBF-1560+
Start Date	2023-05-11	2024-08-09	2023-03-14
End Date	2024-04-29	2024-09-01	2023-04-13
Days Active	248	20	40
Messages Collected	9 695 000	450 000	145 000

Figure 3.7: Timeline of data collection in each location.

in Oxford, UK, with additional hardware in Würzburg, Germany, and Thun, Switzerland. Each had an antenna located on the roof of a building with a good view of the sky in all directions – however, in Germany a cheaper patch antenna was used, resulting in fewer messages per day and lower signal quality. Figure 3.7 illustrates the timeline over which messages were collected.

An initial dataset of 1 705 202 messages was collected over 40 days and used to train the original SatIQ model [16], using the same data collection setup used in our main UK dataset. Although this data has been made available, we do not use it for our analysis in this chapter due to differences in the data formatting, and the presence of a significant time gap between it and our main dataset.

For each transmitter in the UK dataset, as many as 1588 messages were received, with a mean of 260 messages per transmitter. The distribution of message count per transmitter can be seen in Figure 3.8.

An example of the collected data can be seen in Figure 3.9. We can see the signal encodes 8 QPSK symbols, corresponding to the bit sequence for the Iridium

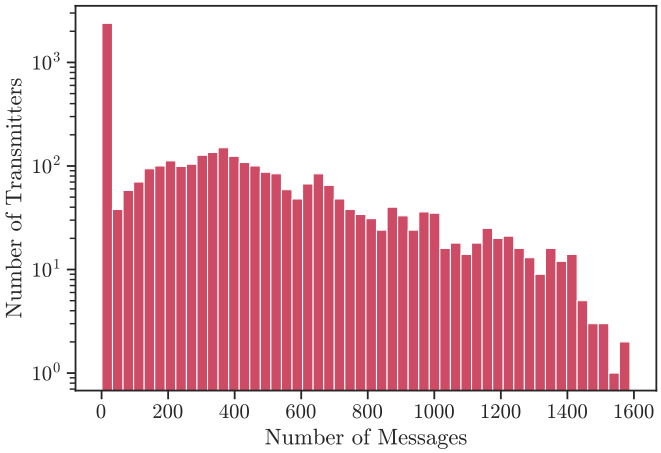


Figure 3.8: The distribution of the number of Iridium messages received per transmitter.

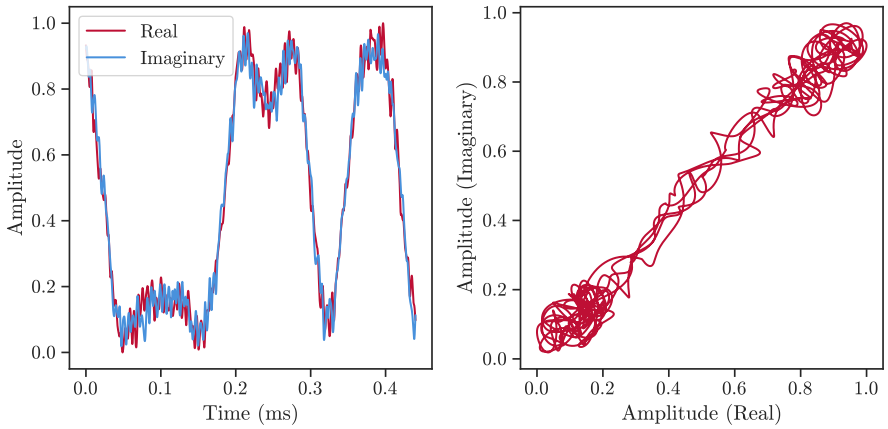


Figure 3.9: A message header received from an Iridium satellite, shown in the time domain (left) and as a constellation plot (right).

message header: `11 00 00 11 11 00 11 00`.[4] However, unlike a constellation plot at 1 sample per symbol, we can see the movement between the two symbols and the impairment on the signal. It is clear that there is significant impairment; this is likely caused by a combination of channel noise, multipath distortion, and hardware characteristics of the transmitter. Our goal is to isolate the last of these, ignoring the channel noise and other characteristics – these will be the same regardless of the transmitter, and not useful in this context.

[4]The message header uses only two of the four QPSK symbols, so the resulting signal resembles BPSK. The remaining two symbols are used in the remainder of the message.

3.5.1 Data Preprocessing

We opt to use a minimal amount of preprocessing to avoid destroying data which might be useful for fingerprinting. On top of the band-pass filtering and phase synchronisation performed by the Iridium decoder, we scale each waveform in the dataset so values range between $-1 - \mathrm{i}$ and $1 + \mathrm{i}$. This removes amplitude as a factor the model needs to learn to adjust for, and makes visualisation easier. We also remove all messages which do not decode as valid IRA messages – although this removes a large number of messages, it ensures that all data is labelled and eliminates the noisiest messages which do not properly decode. This leaves us with messages that are the most likely to contain meaningful identifying factors. In Section 3.7.2 we look into this further, identifying environmental and signal features that affect performance.

Finally, we process the data into "TFRecord" files – this format is optimised for use in TensorFlow, storing data as raw protocol buffers. This enables us to read data directly from disk as needed, minimising RAM usage with only a small buffer to reduce read latency. This is vital for a dataset this large; even on machines with significant amounts of RAM ($\geq$500 GB) the dataset cannot be loaded into RAM in its entirety.

3.5.2 Dataset Construction

After data preprocessing, the entire dataset is split up into files of 5000 entries each, split up by location and organised by date.[5] This is split into training, validation, and testing datasets as follows: for each date, the first file (i.e. the first 5000 entries) is used for testing, the second file for validation, and the remaining files are used for training. This resulted in a training:validation:testing split of 74:13:13. This ensures we have sufficient data for validation and testing without significantly reducing the size of the training dataset, whilst maintaining an even distribution of timestamps across all three datasets.

[5]Previous versions of the dataset shuffled the data at this point, but this proved impossible once it grew sufficiently large. Instead, the data is shuffled by the TensorFlow data pipeline as needed.

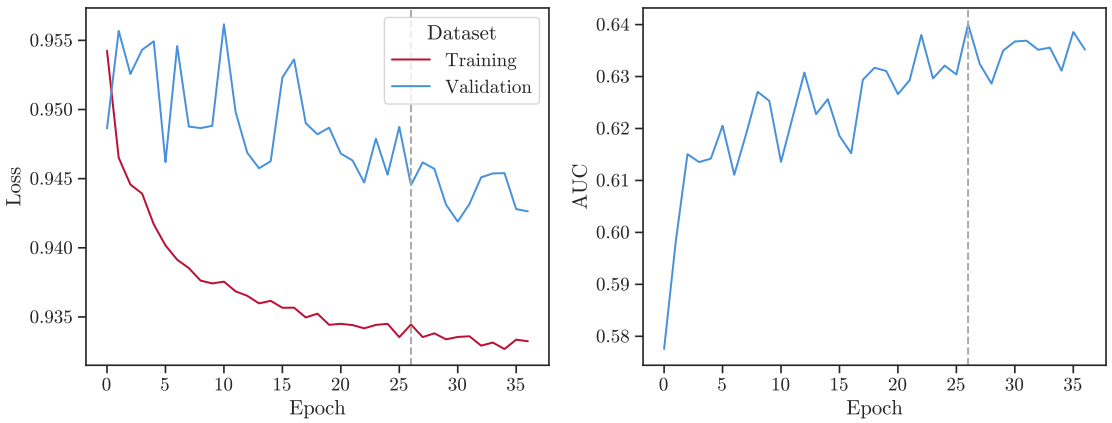


Figure 3.10: Training and validation loss curves for one training run of SatIQ, and the corresponding validation AUC. The dashed line indicates the epoch with the best-performing AUC.

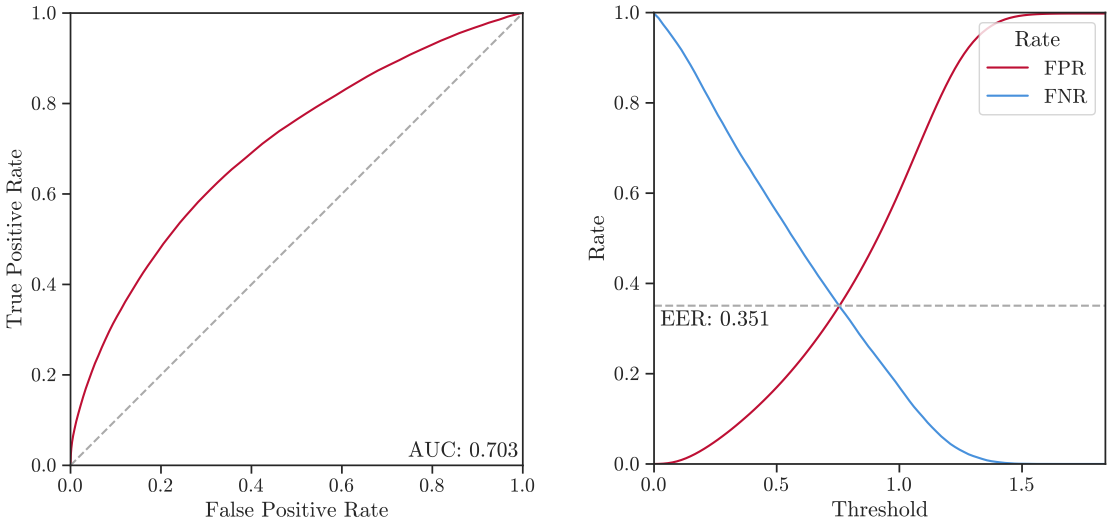
Since the triplet loss function is used, the batch generator is configured to produce batches of inputs with 32 messages (8 batches of 4 messages, each from the same transmitter). This ensures the loss function has a large number of triplets that can be selected from each batch.

3.6 Model Training

Model training was performed on a machine with the following hardware:

- Intel Xeon Gold 6134 CPU (32 threads, 3.20 GHz)
- Nvidia TITAN V GPU (1455 MHz, 12 GB VRAM)
- 512 GB DDR4 RAM

On this hardware, 2 model configurations can be trained simultaneously – the primary constraint is VRAM, with the model requiring approximately 6 GB to train. All models are trained for up to 200 epochs, with early stopping if no improvement is seen within 10 epochs. The training and validation loss curves for one training run can be seen in Figure 3.10, demonstrating how early stopping prevents overfitting: as soon as validation performance stops increasing, training is stopped and the model is rolled back to its best-performing weights. Training on the full dataset took



(a) ROC curve **(b)** FPR against FNR

Figure 3.11: The performance of the base SatIQ model, evaluated by comparing messages from different pairs of satellites without analysing the attack case.

approximately 10 days, with most other training runs taking closer to 24 hours – each epoch runs through the full dataset, so larger datasets result in longer training runs.

Running a trained model takes significantly fewer resources, taking a fraction of a second to verify messages – this makes it possible to run SatIQ in real-time, verifying messages as they arrive. Furthermore, much less RAM and disk space is required, as the training dataset is not needed. Hardware requirements are tested further in Section 3.7.8.

3.6.1 Initial Results

We start by assessing the base performance of the SatIQ model, trained on the full dataset from a single location and tested on the problem of differentiating satellites from different transmitters in the dataset. Note that this differs from the most likely attack case, in which a ground-based attacker transmits messages via an SDR. However, it does enable us to get a good idea of initial performance and fine-tune the system, and enables us to work with a large dataset when performing analysis later on. It is also much harder to distinguish between nearly identical transmitters than to distinguish a ground-based SDR, so we can be confident that good performance in this evaluation will likely translate to good performance in

a more realistic attack case. We back this up in Section 3.7.6 with an evaluation of SatIQ under simple replay attacks.

Figure 3.11 shows the base performance of SatIQ. In this section, models are trained and tested on the problem of differentiating satellites in the dataset, in order to achieve better performance later on when identifying adversaries in an attack scenario. Results in this section are primarily to assess relative performance and fine-tune the system. When assessing performance, we focus on two key metrics:

- **Equal Error Rate (EER):** the error rate when the False Positive Rate (FPR) and False Negative Rate (FNR) are equal.

- **Area Under Curve (AUC):** the area under the Receiver Operating Characteristic (ROC) curve, obtained by plotting the True Positive Rate (TPR) against the FPR. This can be intuitively thought of as the probability that the system can distinguish between two inputs of different classes [145].

To compute each of these, we vary the acceptance threshold: the maximum allowed difference between the fingerprints of two messages, below which we accept them as being from the same transmitter. By raising this threshold, we accept a greater number of legitimate messages, but open the system up to easier spoofing attacks. Conversely, by lowering the threshold the system is made more secure by rejecting more attacker messages, but legitimate messages are more likely to be erroneously rejected.

Our base model has an AUC of 0.703 and EER of 0.351. This is sufficient performance to demonstrate the feasibility of our techniques, particularly in the more difficult case of distinguishing satellite transmitters with identical hardware. In Section 3.7 we go on to demonstrate that performance is significantly better in a replay attack scenario, confirming its usefulness in a security context. For the rest of this section we continue to assess performance on the original (non-attacked) dataset.

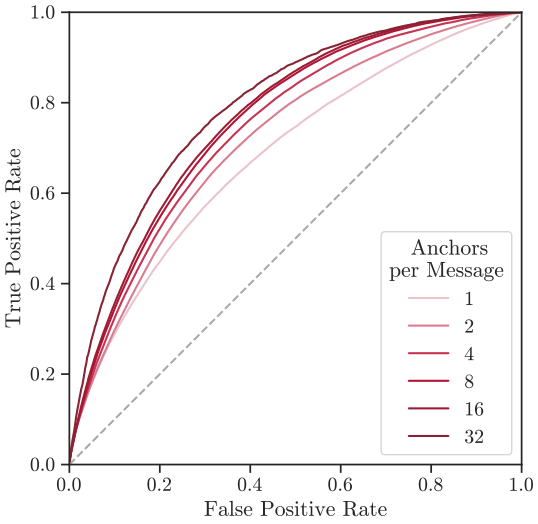


Figure 3.12: ROC curves as we compare each incoming message to a larger number of anchors, taking the mean distance.

Multiple Anchors

To improve performance over the base system, we can compare each incoming message to a larger number of "anchors" (known messages from that transmitter), taking the mean distance between the message and each anchor in the embedding space. The results of this are shown in Figure 3.12. By taking 32 anchors for each incoming message, we can achieve an EER of 0.277 and AUC of 0.795 – a significant improvement! In practice, this can be implemented by saving a larger set of messages from each known transmitter, or by comparing multiple consecutive messages to the same set of anchors. Both of these techniques are practical – our observations suggest that during an Iridium phone call or web connection approximately 11 packets are exchanged per second, so an attacker will need to spoof many packets to have a meaningful impact on the victim. Such an attack would certainly be picked up by SatIQ, even if multiple consecutive messages are compared. We explore attack scenarios further in Section 3.7.

Model Variations

We also tested a number of variations on the model architecture and hyperparameters. To save time and enable further iteration, these were all tested on a reduced portion

Table 3.3: Performance of SatIQ models (ROC AUC, Equal Error Rate) trained on 16 days of data, as we adjust the "margin" term on the triplet loss parameter.

Margin	AUC	EER
2.00	0.674	0.377
1.00	0.674	0.377
0.50	0.677	0.374
0.10	0.678	0.373
0.05	0.669	0.380
0.01	0.665	0.383

of the dataset with only 16 days of data, and compared to an equivalent base model trained on the same dataset. The base model has an AUC of 0.674 on the file used for testing (5000 messages).

We first trained a model with the autoencoder removed. This model achieves an AUC of 0.676 on the same dataset – a very small change, well within random variation. We may therefore be able to save training time by removing the autoencoder, although this would remove the ability to reconstruct message headers from their fingerprints. This capability is not used in this work, but may be useful if adversarial ML techniques are used. We discuss this in Section 3.8.

We also tested the effect of switching the autoencoder for a "variational autoencoder" [146] – in these models, the encoder maps the input to a probabilistic latent space (in this case, a multivariate Gaussian distribution) instead of mapping directly to the latent space. This provides benefits in avoiding overfitting and can sometimes produce more meaningful features in the latent space. However, our trained model on this architecture had an AUC of 0.499, essentially equivalent to random guessing. Variational autoencoders are noted to be quite sensitive to certain hyperparameters, so it may be possible to achieve good performance on this architecture via tuning.

Finally, we adjust the "margin" of the triplet loss parameter used in model training. By increasing this, the separation of the anchor and negative samples during training can be increased further before it stops yielding further gains, and vice versa. Table 3.3 shows the performance of models trained on different margin values – we can see that changing the margin has nearly no effect, so we do not need to worry about fine-tuning this parameter to maximise performance.

Further optimisations to the model are likely possible – there are a wealth of machine learning techniques not explored in this work, and model/hyperparameter optimisation is a known difficult problem. We have demonstrated that SatIQ is effective for fingerprinting satellites, and operators may choose to fine-tune the model further to achieve greater performance, either on the dataset provided or on a new constellation.

3.7 Evaluation

In the previous section we designed and trained the SatIQ system to distinguish between different legitimate satellite transmitters. In this section we examine the key properties that enable SatIQ to be deployed in the real world as a component of a satellite authentication system. We analyse the system's stability over time, look at how performance correlates with weather and signal quality factors, test against subsampled data, and measure extensibility to new satellite transmitters and receiver configurations. We next evaluate the robustness of SatIQ against replay attacks over a wire, demonstrating a dramatic increase in effectiveness at distinguishing an attacker-controlled SDR from legitimate transmitters. We also compare SatIQ against the previous state of the art, showing improved performance at detecting attacks in all cases. Finally, we discuss practical questions surrounding a real-world deployment of SatIQ, including hardware requirements, handling rejected messages, and keeping anchors up to date.

3.7.1 Time Stability

We start by evaluating the stability of SatIQ over time. Existing works in radio fingerprinting observe a decrease in performance when there is a time gap between the training and testing data [147, 148]. This is thought to be caused by changes in the conditions of the wireless channel over time. Our dataset is sufficiently large that we can easily establish the extent of this phenomenon, and explore methods of counteracting it without compromising the security of the system.

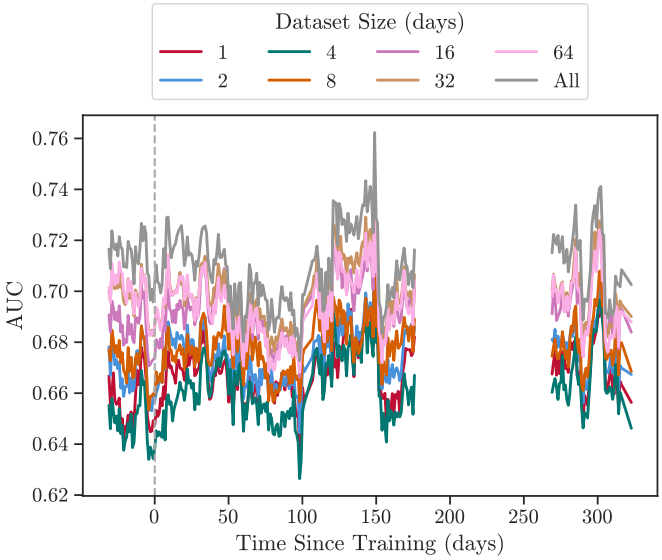


Figure 3.13: Graph showing the performance of SatIQ on incoming messages over time, relative to the start of the training data, for different training dataset sizes. The gap is caused by a break in the training data.

We assess the effect in two ways. Firstly, we train a number of models on increasing portions of the dataset, starting with just 1 day of data and going up in multiples of 2 up to 64 days, followed by the full dataset. We choose the start time for these dataset slices to be 1 month after the start of data collection, so we can look backwards in time as well as forwards. Next, we test each model by computing its AUC and EER on 24-hour slices of the dataset, taking the anchors from the same dataset as the testing samples. This enables us to see how, or indeed if, performance drops off over time as the collected data becomes newer than the training data.

Secondly, we assess the effect of using older anchors in the testing data. We use the same trained models as above, and the same testing methodology, but instead of taking the anchors from the same 24-hour slice as the testing samples, we offset the anchor by a number of days (ranging from 1 to 32 days). This enables us to see how accuracy drops off if anchors are not replaced, and establish how frequently they must be updated.

The results of the first experiment can be seen in Figure 3.13. We can see that overall performance improves with dataset size, and does not drop off with newer testing data when fresh anchors are used. This is a promising result, indicating that it

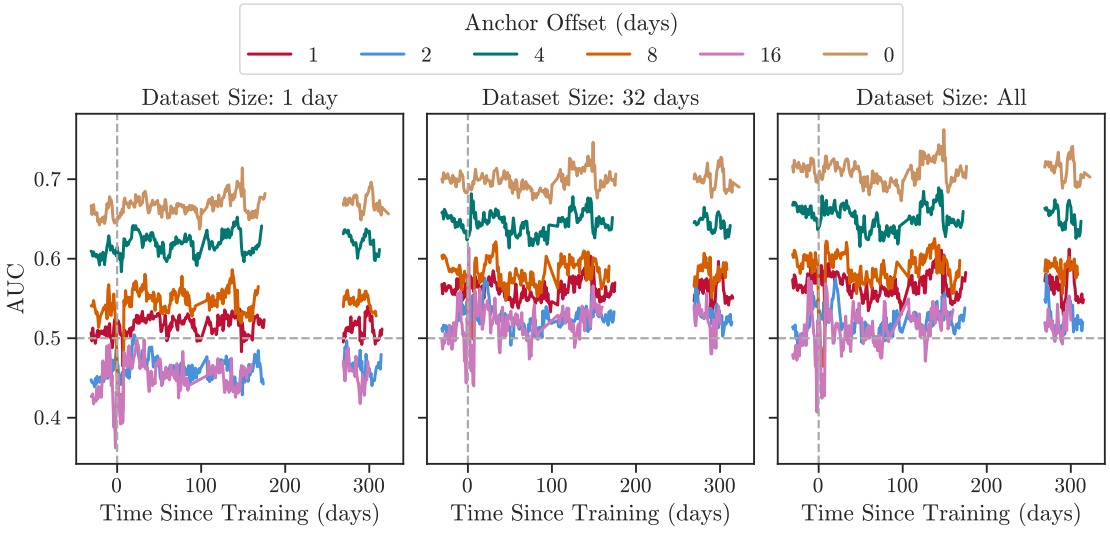
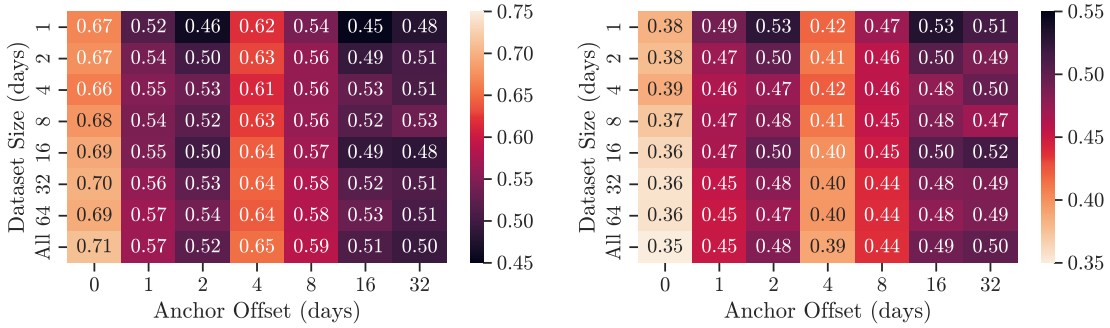


Figure 3.14: Graph showing the performance of SatIQ on incoming messages over time, relative to the start of the training data, with anchors collected at a time offset from the testing messages. The gap is caused by a break in the training data.



(a) ROC AUC **(b)** EER

Figure 3.15: Heatmaps showing the average performance (AUC and EER) of models trained on each dataset size, as the time difference between the anchors and testing samples is increased.

will not be necessary to retrain the model repeatedly over time; it is sufficient to train the model once and then keep updating the anchors. We also see that larger datasets start to yield diminishing performance returns – this is unsurprising, as all systems demonstrate this behaviour, but it is promising that this occurs with only weeks to months of training data. We can therefore be confident that a system trained on this amount of data will perform well and maintain its accuracy as time goes on.

The results of the second experiment are shown in Figure 3.14, for a subset of the training dataset sizes – aggregated metrics for all dataset sizes are shown

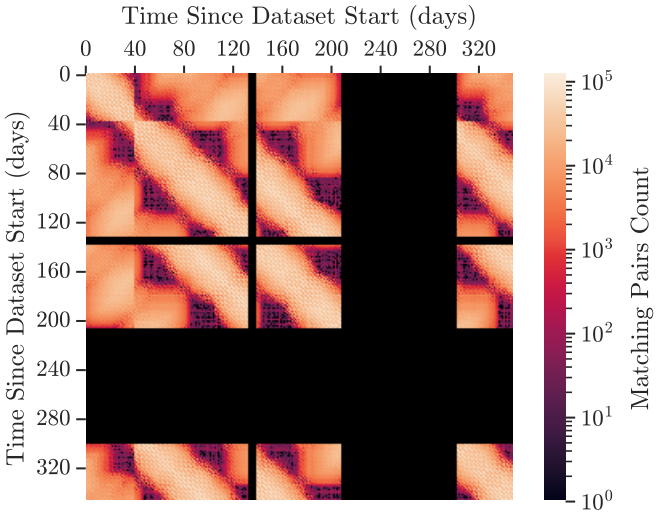


Figure 3.16: Heatmap showing the number of pairs of messages with the same transmitter ID in each day of the testing dataset. Black sections represent gaps in the collected data.

in Figure 3.15. We can see that when SatIQ is trained on only 1 day of data there is a significant drop in performance the moment anchors deviate from the testing data – with just 1 day of difference the average AUC drops dramatically from 0.667 to 0.515. When training on larger datasets the drop in performance is still present, but less pronounced: on 64 days of data the AUC drops from 0.695 to 0.568. We recommend for Iridium that data collection should run for a minimum of 32 days, and that anchors are refreshed every 8 days or fewer to ensure freshness and maximise performance.

Interestingly, there is a spike in performance at an offset of 4 days. This may be caused by periodicity in the dataset, with transmitters reappearing after a certain number of days due to the orbits of the satellites. Figure 3.16 shows a heatmap of how many transmitter IDs overlap for different offset values – we can see there is a non-uniform distribution, with some IDs disappearing for a while before reappearing. It is likely that this is largely due to the sequential nature in which data was collected – each 1-day slice of the testing data represents only a small amount of time, with approximately 24-hour spacing, and the same periodicity may not be present in the full dataset.

Note that refreshing anchors presents some security issues of its own: if an

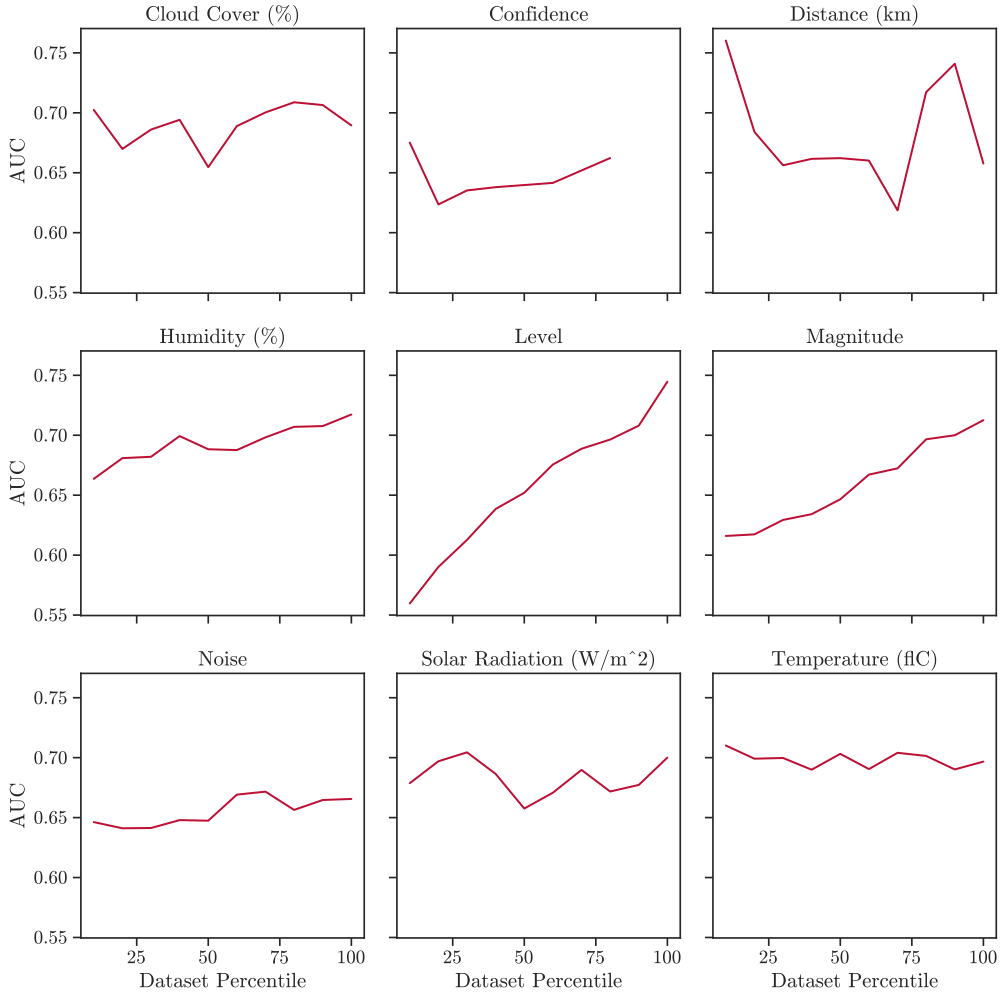


Figure 3.17: Performance of the SatIQ model on slices of 10% of the dataset, sorted by various properties output by the decoder and data collection system.

attacker manages to insert their own messages into the dataset as anchors in place of legitimate messages, then the entire authentication system may be compromised, with legitimate messages rejected and the attacker's messages accepted. Caution must be exercised to ensure anchors are securely refreshed. This could be achieved by periodically verifying messages through other means – for instance, by precisely measuring angle of arrival, or making use of other physical measurements.

3.7.2 Correlations

As satellites pass overhead, their downlinked signals are subject to different amounts of attenuation due to the change in distance to the receiver, and the atmospheric

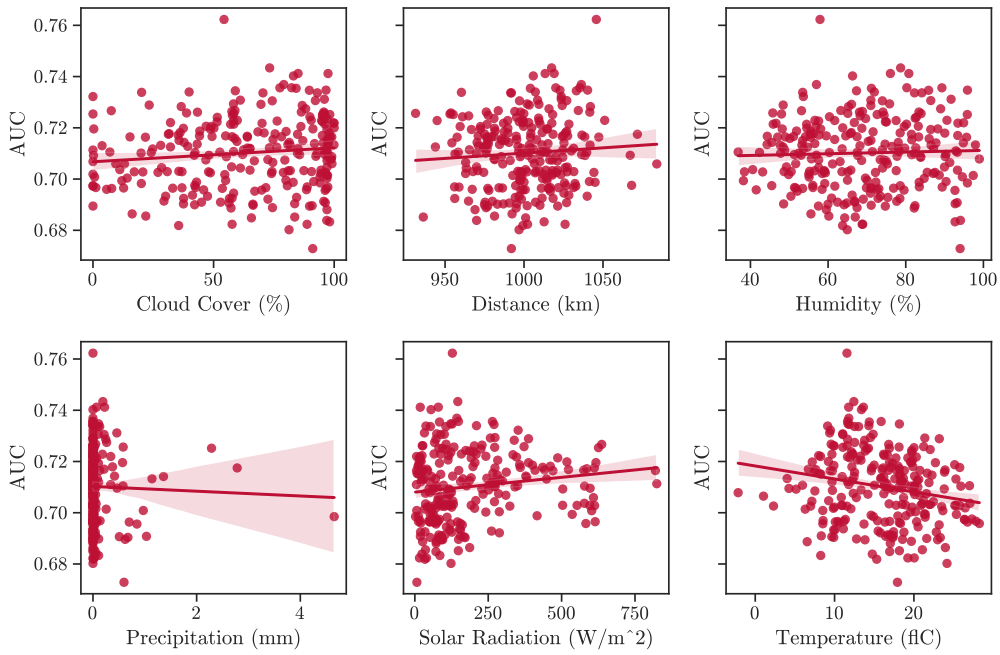


Figure 3.18: Performance of the SatIQ model on 1-day slices of the dataset, plotted against the physical conditions during data collection at that time.

conditions between the satellite and the ground station. As a result, our dataset contains messages of varying amplitudes, levels of noise, and other factors. Although all messages are normalised before use, this is likely to have an effect on performance; for example, it will be more difficult to extract identifying information from signals with high levels of background noise.

To see which factors affect performance, we test a trained SatIQ model on different slices of the dataset. For a given tested variable, we sort the dataset by the variable, then divide it into slices comprising 10% of the original data. Using this data, we compute the model's performance as normal. The tested variables include those reported by the decoder (confidence, level, magnitude, noise), distance from the satellite to the receiver (using the satellite's own reported position), and weather data (cloud cover, humidity, temperature, solar radiation).[6]

Figure 3.17 displays the results of this analysis. We can see that for most of the weather properties there does not appear to be any correlation between the property and the performance of SatIQ, with the exception of humidity, which shows a

[6]Weather data provided by Visual Crossing [149].

very slight positive trend. However, we do see a strong correlation for the "level" and "magnitude" properties output by the decoder, which roughly correspond to signal strength. This is unsurprising – cleaner and louder signals are easier for the fingerprinter to identify, and contain more identifying information. We also see a weak negative correlation between distance and performance – this is also unsurprising, as messages sent from satellites closer to the receiver will have been subject to less atmospheric attenuation.

To see if any weather effects emerged over time, we also compute the average of each weather-related variable for each 24-hour slice of the testing dataset. This can be seen in Figure 3.18 – even looking at the entire dataset, no clear correlations emerge. This indicates that weather does not have a significant effect on the performance of SatIQ, and suggests that weather features are unlikely to be incorporated into the transmitter fingerprint.

Some of the tested properties could be used to filter the dataset to improve performance, but operators must take care when performing this kind of filtering. If only a small fraction of messages are accepted, a denial of service attack can be performed on the system by affecting measured factors – for instance, by adding noise to the channel. Furthermore, if the data is filtered using factors affected by weather, a deployed system might naturally go through an extended period with no suitable messages. Not only would this prevent the satellite from being authenticated, but if anchors cannot be refreshed frequently enough then authentication will be impacted going forward. Nevertheless, these factors may be useful alongside the confidence of the SatIQ fingerprinter to gain a good understanding of the signal state.

3.7.3 Subsampling

We also look at models trained on data that has been subsampled from the original 25 MS/s, to see to what extent it is necessary to operate at such a high sample rate, and to verify that system performance is not held back via the inclusion of unnecessary data. For this evaluation, we trained models on the reduced dataset of 16 days of data, and subsampled to a factor of N by setting all but every Nth sample

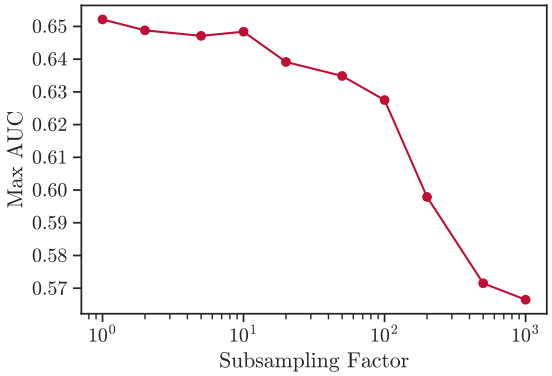


Figure 3.19: Performance of the SatIQ model trained on subsampled data.

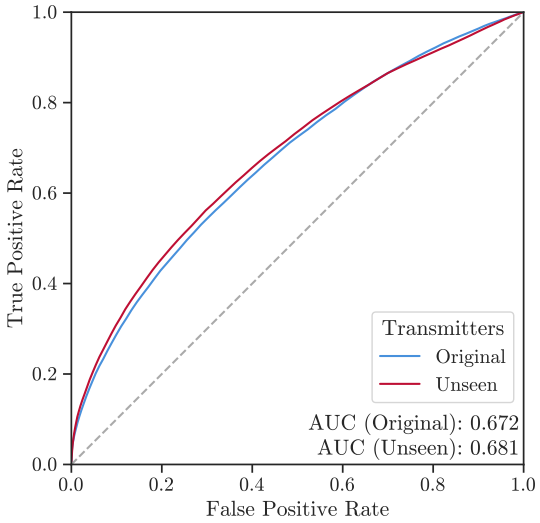


Figure 3.20: ROC curve showing the system's performance on transmitters it has never seen before.

to 0, for values between 1 and 1000. We then take the highest validation AUC reached during training. We can see from the results in Figure 3.19 that performance is highest with no subsampling (i.e. a factor of 1), remains somewhat stable until a subsampling factor of 10, and begins to drop quite quickly above this value. This suggests that SatIQ may be effective when using data collected at a sample rate as low as 2.5 MS/s, without excessively limiting performance of the trained model.

3.7.4 Extensibility

Next we look at how well SatIQ performs on transmitters it has never seen before. This is of particular importance in satellite constellations, where satellites

may need to be replaced at any time, and we want to minimise time and effort spent retraining the model. To test the extensibility of SatIQ, we trained the fingerprinting model on a dataset with some of the transmitters in the testing dataset removed. Specifically, we sorted the transmitter IDs in the testing files by how frequently they appeared, and removed every other ID from training. This resulted in the removal of 422 transmitters. To minimise training time, we once again trained the model on a reduced dataset with only 16 days of data.

The results of this analysis are shown in Figure 3.20. As expected, the base performance roughly matches the testing performance of our original model trained on this dataset, with an AUC of 0.672. When tested on the transmitters removed from the training data, the performance actually increases slightly to an AUC of 0.681. This increase is only slight and may well disappear on subsequent training runs, but from this result we can be confident that SatIQ can be easily extended to new transmitters without retraining. This is far better than existing classifier-based systems, which require retraining each time a satellite is launched or replaced – in modern systems this can be very frequent, with Starlink launching 1984 satellites across 63 launches in 2024 [150]. By using SatIQ instead of a classifier-based fingerprinting system with a closed set of transmitters, we can ensure the fingerprinting system can continue to be used indefinitely as transmitters are added and replaced.

3.7.5 Transferability

We have already shown that SatIQ can authenticate transmitters it has never seen before, but we also need to ensure that it can be used across different receiver configurations. All our training and evaluation so far has looked at data collected from a single source, but this does not provide a useful system if the data collection hardware differs from that of the deployed system – even if the hardware is identical, the receiver may be imparting its own fingerprint, or additional factors like multipath distortion may be affecting the signal. We therefore perform analyses on the

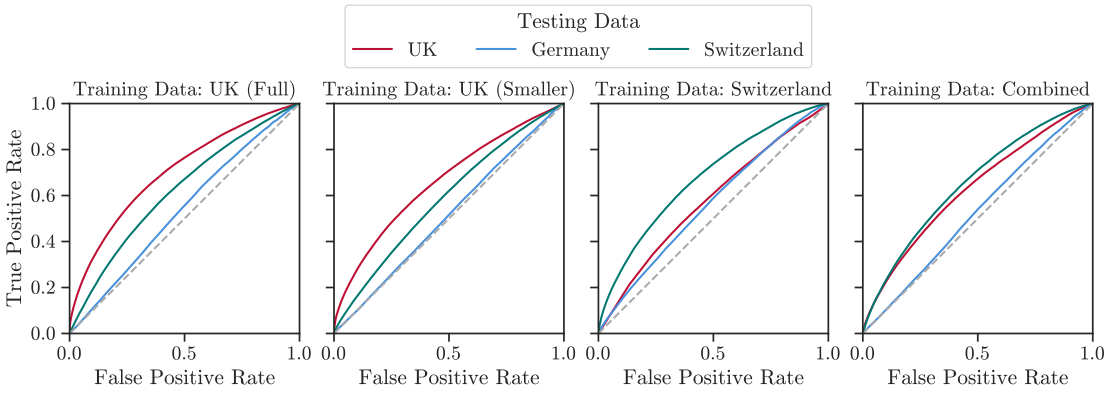


Figure 3.21: ROC curves for models trained on datasets from each location, tested against different locations.

additional data collected in Germany and Switzerland, in addition to the main dataset collected in the UK.

We start by taking our base model (trained on the UK data only) and assess its performance on datasets from the other two locations – we evaluate a model trained on the full dataset, and a model trained on a reduced dataset comparable in size to the data collected in Switzerland.[7] Next, we train models on the Switzerland dataset and test it on the other two datasets, to see if the same effects are present. Finally, we train using a combined dataset from both the UK and Switzerland.[8] This provides potential for interesting analysis – not only can we see whether training on two datasets is a good technique for performance across both datasets, but we can also see how well this model can transfer to the third dataset.

The results of these experiments are shown in Figure 3.21, with statistics for each model on each dataset in Table 3.4. In each case we see a drop in performance when using data from other locations, but not a complete loss of ability to distinguish transmitters. This is an encouraging result, indicating that SatIQ is looking at characteristics inherent to the transmitter even across different locations and receiver configurations, rather than at unrelated properties. We also see that the model trained on data collected in the UK does not show good performance on

[7]The reduced dataset contained only 8 days of collected data.

[8]We do not include the data collected in Germany due to the smaller number of messages and low signal quality.

Table 3.4: Performance of models trained on datasets from each location when tested against the other locations.

Training Data	Testing Data	AUC	EER
UK (Full)	UK	0.703	0.351
	Germany	0.536	0.473
	Switzerland	0.620	0.410
UK (Smaller)	UK	0.660	0.385
	Germany	0.512	0.492
	Switzerland	0.585	0.440
Switzerland	UK	0.577	0.444
	Germany	0.564	0.458
	Switzerland	0.678	0.372
Combined	UK	0.627	0.408
	Germany	0.524	0.481
	Switzerland	0.652	0.392

the data collected in Germany – this data was collected using cheaper hardware, and as a result has lower signal quality, so this is not surprising. Interestingly, the model trained on the Switzerland dataset shows better performance on the Germany dataset. This could be due to the lower signal quality of the training data, resulting in a greater ability to withstand noise. We also note that the smaller UK dataset performs similarly to the Switzerland dataset, suggesting that good performance will be possible with a larger dataset from any location or hardware configuration.

When training a model on the combined dataset, good performance is achieved on both the UK and Switzerland datasets, with the results on the data from Germany somewhere between the two single-dataset models. The training dataset for this analysis was relatively small compared to the full dataset, but it is sufficient to show the transferability of the architecture. From these results we can also be reasonably confident that with a larger dataset that mixes data from more hardware configurations and physical locations, good performance can be achieved even on unseen hardware. This will dramatically increase the deployability of SatIQ, as it can be deployed in new locations on any receiver hardware without requiring a new dataset or model retraining.

It may also be possible to transfer a trained SatIQ model to a completely new satellite constellation, since many of the signal impairments will be common between

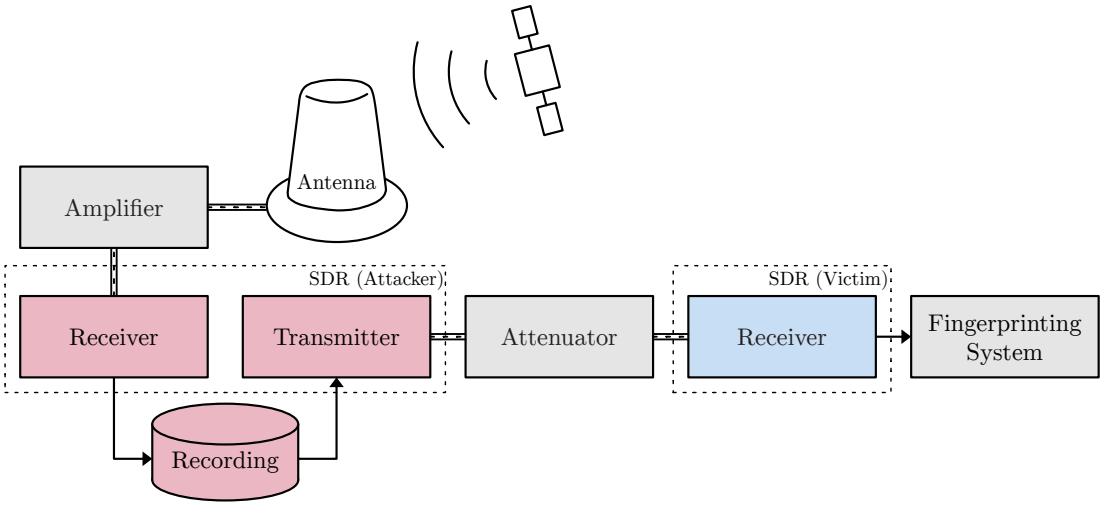


Figure 3.22: Hardware setup for the replay attacks. Raw samples are captured using an SDR, then replayed directly into the fingerprinting system's SDR over a cable.

hardware configurations. With a small amount of retraining, similar performance might be achieved across a wide range of satellite systems. In Section 3.8 we discuss how the SatIQ architecture has been adapted to work with the ORBCOMM constellation of satellites, and how future work might create a fingerprinting model that works across multiple satellite architectures.

3.7.6 Security

We next evaluate the security properties of SatIQ, assessing its performance under an attack scenario. A basic attack involves swapping the identifiers of transmitters in our existing dataset, simulating an internal attack where an attacker has gained control of a currently operational satellite. Although this aligns with our training scenario and the results discussed in Section 3.6.1, it is somewhat unrealistic and does not match our threat model of a ground-based attacker with an SDR.

The more interesting case is to instead evaluate SatIQ's robustness under real-world replay attacks. We can test this in a realistic setting by replaying recorded messages over a wire. By capturing and replaying real-world messages, we ensure signal characteristics like background noise, path loss, and attenuation are as realistic as possible, since they are already a part of the recorded data. Furthermore, by replaying the messages over a wire, the signal is not being further degraded

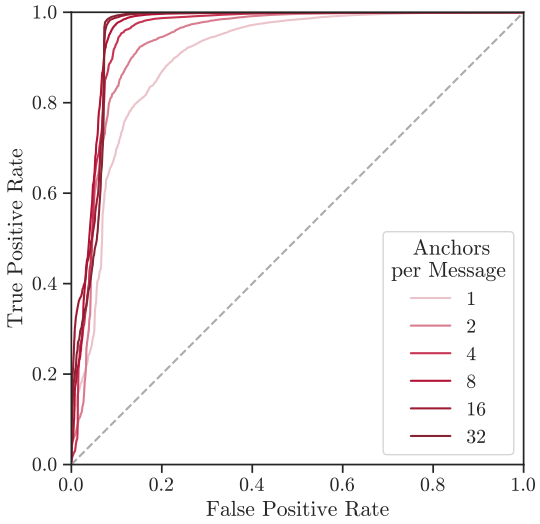


Figure 3.23: ROC curves showing the system's performance when detecting replayed messages. Performance is significantly better than our training results, achieving a maximum AUC of 0.946.

by the inclusion of channel impairments a second time over. However, it is still impacted by the features of the attacker's SDR, so the final signal will include the fingerprint of the original transmitter distorted by the new features of the attacker's transmitter. A similar result could be achieved through the use of an RF-shielded box, but this has the potential to introduce further impairments through reflections and other effects. Our experimental setup is as shown in Figure 3.22. We first capture Iridium messages at 25 MS/s, saving the raw IQ samples to a file – this provides us with a dataset of samples identical to what the SDR would normally receive. We then replay these samples over a wire connected to the "victim" SDR, feeding the captured messages into the fingerprinting system. By this strategy, 253 messages were collected.

For each replayed message, we take a number of "known good" messages from the same transmitter from our testing dataset. We randomly select a number of these messages to be our anchors using a "shuffle split" strategy. We compare the anchors to the replayed messages to obtain the false positive rate, and to the other known good messages to get the true positive rate. The results of this experiment are shown in Figure 3.23. We can see that SatIQ performs significantly better in this scenario, with a base AUC of 0.927. When we compare each message to

Table 3.5: True positive (true accept) rates and false positive (false accept) rates for key threshold values, tested on replayed messages. Messages are tested against 32 anchors, and the mean distance is taken.

TPR	FPR	Threshold
0.999	0.257	1.238
0.990	0.107	1.092
0.950	0.074	0.970
0.900	0.072	0.907
0.989	0.100	1.081
0.524	0.050	0.703
0.327	0.010	0.624
0.148	0.000	0.538

16 anchors this performance increases even further, with an AUC of 0.960 and an EER of 0.072![9] This indicates that the attacker's SDR has introduced its own fingerprint, distorting the message and altering its features.

Furthermore, this performance is good enough to deploy in a real-world system – by adjusting the acceptance threshold we can achieve a high true acceptance rate while minimising the number of spoofed messages that are accepted. These results are summarised in Table 3.5. By setting the threshold such that 99% of legitimate messages are accepted, we accept only 10% of the attacker's messages. This performance is good enough to use in a real-world setting, particularly if we continuously fingerprint messages over the course of a communication session, taking the average acceptance rate over time as an indicator of attack – in order to have a meaningful impact the attacker will need to spoof multiple messages, which significantly raises the likelihood of detection.

Figure 3.24 shows the distribution of fingerprint distances from the replay dataset over time. We can see that there are clusters of consecutive messages all with similar distances in fingerprint space, and further that these all belong to the same transmitter ID. We draw from this result that some transmitters may be easier to spoof than others. The overall false acceptance rate is still low, however, so the risk is limited.

[9]Note that performance starts to drop off slightly for larger numbers of anchors – this is not an indication of the strategy's ineffectiveness, but of the limited size of the replay dataset. If more replayed messages were collected then this artefact would disappear.

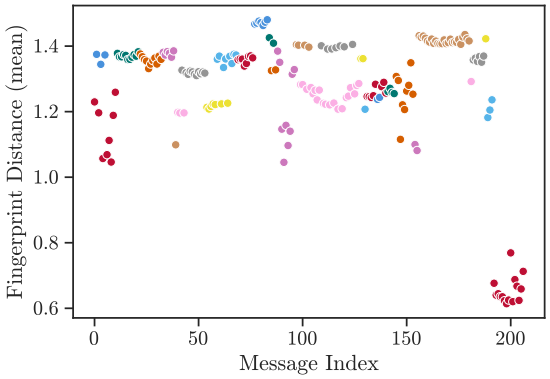


Figure 3.24: Distance in fingerprint space between attacker-replayed messages and legitimate messages over time. Transmitter ID is represented by message colour; repeated clusters of the same colour belong to different transmitters.

This attack scenario assumes a well-equipped adversary with access to a high-end SDR, and eliminates all the difficulties of over-the-air replay attacks. Despite all these concessions, we are still able to detect the attack in the majority of cases. With an even higher budget (to transmit at an even higher sample rate) and careful effort to eliminate noise introduced by the radio, it will certainly be possible to circumvent this system [120], but our results show that it will take a concerted effort to do so – simple message replay is not enough. We can therefore exclude a large proportion of attackers with all but the highest budgets, granting a real-world security benefit to ground systems.

In Chapter 4 we build on these results, moving beyond naïve spoofing attacks to look at optimising attacks targeting the SatIQ system directly. These range from optimised jamming to spoofing the fingerprinter by removing the fingerprint of the attacker's SDR, or poisoning the dataset through the inclusion of reference anchors containing the fingerprint of the attacker's transmitter.

3.7.7 Comparison with Existing Systems

We also compare the performance of SatIQ against the similar "PAST-AI" system discussed in Section 3.2 [115]. Instead of looking at the signal at a high sample rate, this system instead combines a large number of incoming messages from the same transmitter into a single heatmap (with signals captured at 1 sample per symbol),

using an image classifier to identify transmitters. This is a much simpler system, but it comes with a number of caveats: perhaps most importantly, that it requires a large number of messages (approximately 100, following the implementation in the paper) to build each heatmap, so authentication cannot be done on a per-message basis. Attacks that take a small number of messages to execute are therefore unlikely to be detected. Furthermore, the system is a simple classifier with a closed set of identifiers, so it is impossible for the attacker to be identified as anything other than a legitimate transmitter – the only possible outcome is misclassification.

We demonstrate this with a quantitative comparison between PAST-AI and SatIQ. The dataset used in the paper has been made open access [151], and we gained access to the code by contacting the authors directly. The training data and code are therefore identical to those used in their paper.

We compare the two systems under three different types of attack:

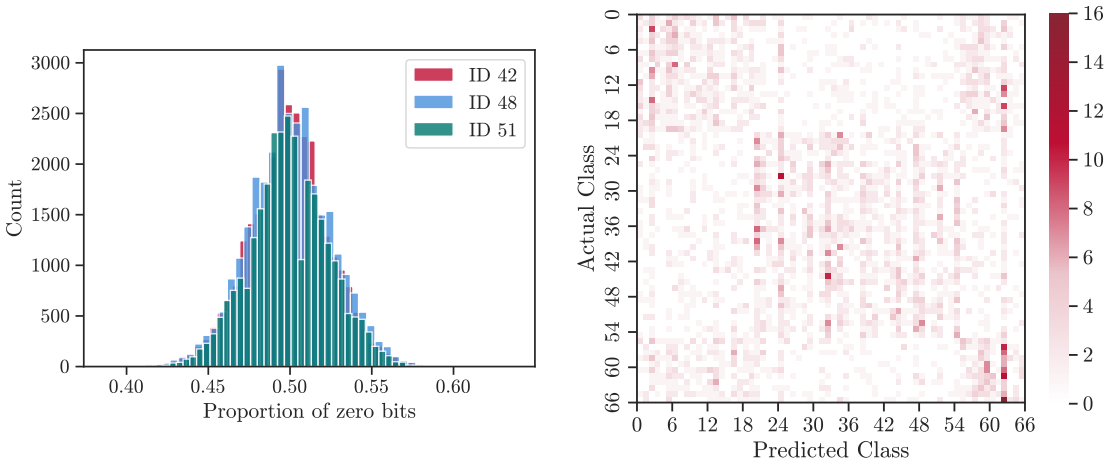
- We first look at the "label swap" attack evaluated in Section 3.6.1, in which messages are sent from a legitimate transmitter but the identifier has been falsified. This is the least realistic attack, as it requires the adversary to take over a satellite from the original set.

- We next look at the "continuous SDR" case, in which an attacker is continuously sending spoofed messages from an SDR.

- Finally, we look at the "single message SDR" case, in which the attacker sends a single spoofed message from the SDR.

When evaluating SatIQ, we use the same results established earlier in this chapter. Note that when using SatIQ the latter two attacks are considered to be the same, since we are primarily fingerprinting individual incoming messages (although performance may be increased by tracking accept/reject rates over time for consecutive messages).[10] On the other hand, PAST-AI requires many incoming

[10]If multiple anchors are used, we assume a single incoming message is checked against multiple stored anchors, rather than averaging across multiple incoming messages.



(a) Example of the different distributions of zero bits between transmitters in the dataset.

(b) Confusion matrix from a Support Vector Machine trained to classify transmitters from the data distribution alone, resulting in an accuracy of 3.70%.

Figure 3.25: Evaluation of the distribution of bits in the PAST-AI dataset, and how this may skew results.

messages to build a single image for classification, so single message spoofing events are much harder to detect.

We also note the differences between PAST-AI and SatIQ for the label swap attack. Firstly, PAST-AI is only capable of classifying entire satellites, and cannot identify individual beams within the satellite as SatIQ can – this is a much harder problem due to the larger number of classes involved, and naturally results in a moderate drop in performance. Despite this, however, SatIQ performs better under the label swap attack compared to PAST-AI. Secondly, we note that PAST-AI does not attempt to mask out identifying information present in messages, creating its input images from the whole message, including transmitter ID. This affects the distribution of symbols present in the message, and opens up the possibility for the model to learn the underlying distribution. Figure 3.25 illustrates the differing distribution of symbols between messages from two different transmitters, and shows the output of a Support Vector Machine trained to classify transmitters based on the distribution of symbols alone. We achieve a maximum validation accuracy of 3.70% (random guessing would yield 1.51%), demonstrating the presence of identifying information in the underlying distribution of symbols. This casts doubt on the

Table 3.6: Success rate of the attacker in three different types of attack, comparing between PAST-AI and SatIQ (lower numbers indicate the system is better at detecting the attack). All results are taken from experiments in this section.[12]

	PAST-AI	SatIQ
Label Swap	0.287	0.277
Continuous SDR	0.278	0.072
Single Message SDR	0.713	0.072

performance figures quoted by the authors of PAST-AI, as it seems likely the model is improving its performance by looking at the distribution of symbols in a group of messages. On the other hand, SatIQ operates on message headers alone, which do not contain any identifying information which could skew the results.

For the SDR-based attacks, we must create a replay dataset in the same format as used by PAST-AI. We achieve this using a hardware configuration similar to the one shown in Figure 3.22, replaying messages from the original PAST-AI dataset through an SDR transmit-receive loop over a wire (this hardware is described further in Section 4.5.2). Using this dataset, we construct images composed solely of replayed messages (for the "continuous SDR" attack), and heatmaps derived primarily from the original dataset, but with a single message replayed (for the "single message SDR" case). We classify these images using the trained PAST-AI model. The results of this are shown in Figure 3.26, with performance figures in Table 3.6. We can see that under the "label swap" attack, performance is very similar between SatIQ and PAST-AI, despite the fact that SatIQ is authenticating a much larger set of transmitters.[11] We also see that under the "continuous SDR" attack, PAST-AI incorrectly classifies the replayed messages as the legitimate transmitter with a rate of 0.278, compared to SatIQ's 0.072. Performance is worse when only a single message is replayed: the message is incorrectly classified as legitimate in 0.713 of cases. We therefore conclude that SatIQ is more suitable for deployment as a countermeasure to spoofing attacks, both continuous and on individual messages.

[11]We note that our experimental reproduction of PAST-AI (using the exact code and dataset used in their paper) yielded worse performance than reported in the paper [115], with a baseline accuracy of 0.82, leading to an attacker success rate of 0.18.

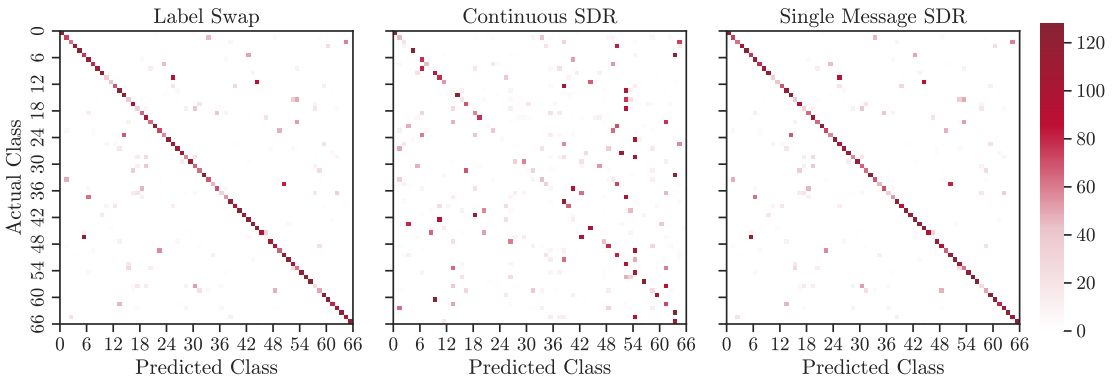


Figure 3.26: Heatmaps showing the predicted vs actual class under each type of attack, using the PAST-AI model. Note that in the first attack, the attacker's goal is for transmitters to be misclassified, whereas in the second and third their goal is for replayed messages to be classified as legitimate.

3.7.8 Deployment Considerations

Finally, we address some of the challenges involved in deploying SatIQ in real-world systems.

Model Performance

First and foremost, although it is highly likely that future optimisation will increase performance, we must consider how a model could be deployed in the real world with a measured EER of 0.072 – in practice, this means as many as 7.2% of malicious messages might be accepted, or legitimate messages rejected. To mitigate false rejections or false acceptances, an operator can adjust the acceptance threshold in the desired direction as established in Table 3.5, but reducing false rejections naturally increases false acceptances and vice versa. One alternative operators might consider is to only rely upon the fingerprint during a connection establishment process, establishing a secure session key which can be used for the remainder of the session. During this process, the threshold can be set with very low tolerance for incorrect fingerprints, restarting as many times as needed in order to ensure an attacker was not able to influence the messages. Even if this takes multiple retries,

[12]In the PAST-AI paper [115], the authors report a baseline accuracy of 0.82 when using all 66 labels, which would result in an attacker success rate of 0.18 in the "label swap" attack. However, running their code on the dataset provided yielded different results, which we report in the table.

connection establishment comprises relatively few messages, so the process will terminate relatively quickly (within a few seconds), and result in a session key which operators can be confident has not been tampered with in-transit. Such a mechanism can be deployed on top of existing legacy connection establishment processes (e.g. by manually restarting each time the fingerprint fails to authenticate), or integrated directly into more modern systems.

Device fingerprints can still be useful during nominal operation, but it is important for operators to decide what action to take when a (potentially legitimate) message is flagged. In high security systems messages may be rejected entirely, with operators accepting the increased losses in exchange for greater security against spoofing and replay attacks. More likely is that operators would log the alert but still accept the message, increasing awareness of potential attacks without risking impact to service through rejecting legitimate communication.

Finally, we must consider the process of refreshing the anchor messages against which incoming messages are compared. As discussed in Section 3.7.1, we propose that SatIQ-based systems periodically refresh anchors to maximise performance. An additional benefit of this approach is that no centralised database of fingerprints is required, significantly reducing the complexity of deploying SatIQ – there is no need to gather or maintain a database of fingerprints, and devices do not need to connect to a server to update their databases or verify incoming messages. SatIQ-based systems can therefore also be developed by either the satellite operator or the receiver manufacturer, or by a third party building devices to integrate with existing ground systems.

Hardware Requirements

As mentioned in Section 3.6, a trained SatIQ model can be run in real-time, authenticating messages as they arrive. We verify this by running the model on lightweight hardware. Our data collection observed roughly 0.5 ring alert messages per second, or 6 to 7 messages per second across all message types. This gives us approximately 150 ms to evaluate each incoming message. On the Intel Xeon CPU

used for training (disabling the GPU), the model takes 4.85 s to process a batch of 543 messages (8.9 ms per message), with a maximum RAM usage of 1.7 GB – this is more than fast enough to handle all incoming messages in real time.

We also ran SatIQ on a Raspberry Pi 4B single-board computer, with 4 GB of RAM and a 1.8 GHz quad-core processor. On this hardware, it took 93.25 s to process the same 543 messages (171 ms per message), using 2.0 GB of RAM. This demonstrates that SatIQ can run on lightweight hardware – although slightly too slow to operate in real time, this can certainly authenticate incoming ring alert messages, and could run even faster with the addition of a dedicated AI accelerator.

We must also consider how deployed devices can capture high sample rate message fingerprints. Our current setup requires a moderately powerful computer to perform the necessary high sample rate message synchronisation and decoding, but this can be reduced significantly with the use of a dedicated FPGA demodulator. Combined with the good performance on low-power CPUs shown above, it will be possible to use SatIQ with even lightweight devices, or to deploy it as a separate device that operates alongside the original ground system.

3.8 Future Work

Our results have revealed several promising avenues of future research. Firstly, it is likely that better performance can be achieved through fine-tuning, using the same base model architecture as SatIQ. We have already seen that by training a model on data collected from multiple receiver configurations, we can produce a system that transfers more readily to different ground systems; with a larger dataset from more locations, this would likely be even more versatile.

It would also be useful to implement some of the other techniques explored in related works, particularly in assessing the extent to which a trained system can be transferred to another constellation, and what degree of retraining is required. Furthermore, it would be particularly beneficial to standardise the analyses used in this chapter – we currently have no method of empirically comparing fingerprinting techniques to each other in terms of their security properties, and a standard suite

of tests would remedy this. This could be adapted from our tests in Section 3.7.7 (comparing SatIQ against the PAST-AI system) with an increased focus on consistent measurement, so the experiments can be reliably applied to different fingerprinting systems. One particular challenge in this area is that each technique relies on a different form of data; for instance, PAST-AI combines multiple messages into a single heatmap, some works look at the transient of messages, and some look at the steady state. A standard set of tests will need to accommodate for differences in data format to ensure results are both consistent and fair, not favouring one technique over another.

Another promising area of research would be to assess the effectiveness of fingerprinting in conjunction with other methods of spoofing detection, such as assessing SNR or distortion. Multiple fingerprinting methods could also be used in concert, providing even greater effectiveness than any model alone. However, some methods are likely to use the same signal properties as each other (providing no mutual benefit), so a full analysis is needed in order to understand which methods are effective together.

It would also be useful to assess the effectiveness of fingerprinting in systems which already have some amount of authentication. For instance, such an analysis could evaluate fingerprinting as a preventative measure against GNSS message delay/advancement attacks [77]. This would demonstrate that fingerprinting is not just effective in protecting legacy systems, but has concrete benefits even in new satellite systems. One area in which this might be particularly valuable is uplink fingerprinting: looking at the characteristics of the ground-based signal as received at the satellite. This has not yet been explored since it would require suitable SDR hardware aboard a satellite, but could yield interesting results, looking not only at authentication but also identifying hardware characteristics of ground users.

Finally, future work might consider adapting SatIQ to work across multiple constellations. This has already been achieved to some extent; "OrbID" adapts the SatIQ architecture to work with the ORBCOMM constellation of satellites, demonstrating the versatility of the techniques [25]. However, no work has yet

looked at fingerprinting models that are effective across multiple satellite systems and modulation schemes without retraining. With a sufficiently large and diverse dataset this should be possible, and would be highly useful as a drop-in security measure across a huge range of satellite systems.

3.9 Conclusion

In this chapter we have demonstrated the effectiveness of high sample rate fingerprinting at securing satellite communication against spoofing and replay attacks, thereby advancing the state of the art. By showing its stability over time, transferability to new transmitters and receiver configurations, and consistent performance in the face of changing signal and weather conditions, we ensure SatIQ is maximally useful as a protective measure against SDR-equipped attackers, without requiring huge amounts of finetuning or maintenance. We have also provided the tools necessary for SatIQ to be adapted to work with different satellite systems, enabling it to be used across a wide range of systems, instead of being limited to only a single satellite constellation, protocol, or signal modulation scheme.

Although we have already shown that SatIQ is effective at detecting naïve spoofing attacks, it remains to be seen how effective the system is against optimised attacks targeting the fingerprinting system itself. We go on to evaluate this in the following chapter.

3.9.1 Availability

To facilitate the deployment of SatIQ on real world systems and enable future research, all code has been made fully available at github.com/ssloxford/SatIQ. Model weights are available at zenodo.org/record/8298532, and the original dataset of 1 706 556 messages can be found at zenodo.org/record/8220494. The full dataset has not yet been uploaded to a repository due to its size, but is available on request.

4

Investigating Attacks on Physical Layer Satellite Authentication Systems

Contents

4.1 Motivation 92
4.1.1 Contributions 93
4.2 Related Work 94
4.3 Threat Model 96
4.3.1 Goals 96
4.3.2 Capabilities 98
4.3.3 Effective Attack Range 101
4.4 Experiment Design 104
4.4.1 Experimental Foundations 104
4.4.2 Simple Jamming 106
4.4.3 Optimised Jamming 106
4.4.4 Identity Shift 107
4.4.5 Fingerprint Masking 108
4.4.6 Data Poisoning 111
4.5 Data Collection 113
4.5.1 Jamming 113
4.5.2 Transmit-Receive Loop 117
4.6 Results 119
4.6.1 Simple Jamming 119
4.6.2 Optimised Jamming 122
4.6.3 Identity Shift 125
4.6.4 Fingerprint Masking 126
4.6.5 Poisoning 128
4.6.6 Single-Transmitter Fingerprinting 131
4.7 Discussion 134
4.7.1 Countermeasures 135
4.7.2 Future Work 136
4.8 Conclusion 137

In the previous chapter we have seen the effectiveness of fingerprinting for authentication of satellite transmitters. But before we can start thinking about deploying this technique more widely, it is essential that we first consider the threats it may face – if implementing this countermeasure creates new vulnerabilities, then the system's security remains compromised. To address this concern, we examine various attacks targeting the SatIQ fingerprinting system directly, evaluate its robustness, and explore potential improvements and countermeasures informed by our findings.

4.1 Motivation

As the rise of off-the-shelf Software Defined Radio (SDR) hardware makes it easier for even hobbyist-level attackers to cause disruption to radio systems, legacy satellite systems have been made particularly vulnerable due to their lack of cryptographic security. In the previous chapter we demonstrated fingerprinting as a countermeasure to this problem: by looking at unique characteristics of the transmitter hardware expressed as impairments on the physical layer radio signal, transmitters can be differentiated from one another, even in the uniquely difficult environment of satellite communication with high levels of atmospheric noise. Crucially, this can also separate legitimate transmitters from attacker-controlled SDRs, enabling robust authentication even in the absence of cryptography.

However, an adversary's goal is not always to spoof communication; sometimes simple denial of service is sufficient. Traditionally this has been achieved through jamming techniques, involving the use of targeted noise or other signals to stop the legitimate signal from being properly decoded. This has been observed widely in the real world, particularly in the recent jamming attacks on Starlink during Russia's invasion of Ukraine [152, 153]. If fingerprinting techniques are used to secure a ground system, incoming messages may also be rejected if the transmitter fingerprint does not match the expected value. Therefore, an attacker may achieve easier denial of service by simply disrupting the fingerprint.

Furthermore, there is a gap of knowledge in attacks on fingerprinting-based authentication systems in a more general sense: the majority of fingerprinting works stop at simple authentication or classification, and do not consider direct attacks on the system. Although some works do look at attacks on the fingerprinting system itself, this is limited to simple replay [154, 155], which we have already shown is easily detected, or perfect signal reproduction using an Arbitrary Waveform Generator (AWG) [120], which is sufficiently expensive as to be out of budget for the vast majority of attackers. It is not known to what extent fingerprinting systems are vulnerable to attacks on the underlying model, carried out using consumer-grade hardware and optimised for maximum disruption.

4.1.1 Contributions

In this chapter we provide the first end-to-end evaluation of wireless fingerprinting under optimised jamming, spoofing, and poisoning attacks, assessing the extent to which these systems are vulnerable to attackers equipped with off-the-shelf hardware. We focus on the satellite use case due to the particular relevance of fingerprinting in this area – legacy satellites are uniquely both vulnerable to attacks on the wireless channel and prohibitively expensive to upgrade or replace, making fingerprinting a particularly appealing countermeasure. In undertaking this work, we can better understand the risks associated with fingerprinting alongside its benefits, enabling operators to make well-informed decisions surrounding its implementation.

We look first at simple jamming attacks, assessing the effect of noise on fingerprinter performance and comparing to its effect on the base message decoder. We then move on to looking at optimised jamming, spoofing, and data poisoning attacks. We demonstrate that optimised jamming signals can cause significant disruption even at a very low amplitude, and that spoofing signals can be constricted to make messages from one transmitter appear to come from another. We also show that attackers can mask out the fingerprint of their own transmitting hardware when replaying messages, and that reference messages can be poisoned over time through incremental updates, allowing an attacker-controlled transmitter to be accepted

as legitimate. The implications of these findings are significant, highlighting the potential for fingerprinting systems to be vulnerable to targeted attacks, and the importance of using fingerprinting alongside other countermeasures.

Finally, we explore the use of Generative Adversarial Networks (GANs) to improve the security of fingerprinting systems, demonstrating that a GAN-trained discriminator can match the performance of our previous model at detecting replay attacks, even from previously unseen transmitters. Alongside detecting attacks, this technique also enables fingerprinting in systems with only a single transmitter – rather than training on a dataset with many different transmitters, operators can instead train a model using our SDR transmit-receive loop to differentiate between legitimate and malicious communication. This will enable operators of small constellations, or even single satellites, to gain additional protection against attacks by deploying fingerprinting as an additional countermeasure, without requiring a huge dataset containing many transmitters.

4.2 Related Work

In this section we look specifically at attacks on satellite fingerprinting and other RF authentication systems. This builds on the general fingerprinting background from Section 2.4.2.

The authors of [156] and [157] provide an overview of adversarial attacks on RF machine learning systems. This includes attacks on signal and modulation classification systems [158–160], jamming attacks [161], and attacks on RF fingerprinting systems [162, 163]. In [162] a generalised approach is given for adversarial jamming and spoofing against a target neural network, and the authors demonstrate its effectiveness against a classifier for the "ADS-B" protocol used in aviation, and a signal modulation classifier. We use a similar approach for our attacks, using gradient descent to find optimised jamming signals. There has also been a small amount of work looking at jamming general fingerprinting systems using BPSK modulated signals [164], but performance is not as good as in our attacks – possibly due to their system working with a smaller number of classes, and thus

requiring more power to disrupt. In [163], the authors instead use reinforcement learning techniques to fool a discriminator classifier into accepting messages from an attacker-controlled transmitter, based on binary classification alone, evaluating the approach using 8 SDRs. The desired outcome in this case is similar to our spoofing experiments, but we examine a much larger set of transmitters and focus on real-world systems. The authors of [165] and [166] also investigate attacks on fingerprinting systems, in which an attacker causes misclassification of malicious messages through the use of adversarial machine learning. Finally, in [167] the authors use SDRs to gather a dataset for fingerprinting, but all imperfections are introduced at the software level, and the impact of the SDRs is not measured.

There has also been some work looking at direct impersonation of device fingerprints: for example, the authors of [120] use an AWG to replay messages with sufficient precision that the fingerprint is duplicated. In [154] a similar approach is attempted using SDR hardware at lower sample rates, with limited success. We further discuss the implications of the attacker's capabilities and hardware in Section 4.3.

The vast majority of the attacks explored in related works are against classifier-based systems, with a particular focus on image classification [168]. In the case of classifiers, attacks naturally target misclassifications within the system. However, when targeting a distance-based system like SatIQ, attacks instead focus on the distance metric produced by the model, aiming to either increase the distance between legitimate samples and their anchors to create jamming behaviour, or decrease the distance to known anchors to spoof messages or alter the fingerprint. This aligns more closely with adversarial research in biometric systems, in which there already exists a good amount of work. For instance, in [169] the authors use gradient descent to produce adversarial perturbations for misclassification in face recognition systems, and in [170] the authors use hill climbing (a similar approach) to find optimised attacks on a human fingerprint recognition system. Finally, in [171] the authors train a GAN to perform spoofing attacks on voice recognition systems,

and discuss its usefulness as a countermeasure to these attacks. We explore similar approaches to each of these in our spoofing experiments later in this chapter.

Finally, in recent years "poisoning" has emerged as a popular attack strategy against machine learning systems – these attacks involve the introduction of malicious data into the model, affecting downstream operation of the model [172–175]. Poisoning attacks can take place during the training phase, adding false data into the training dataset in order to reduce performance, increase misclassifications, or falsely accept specific inputs. Alternatively, they can take place during operation, altering the ground truth over time through the gradual introduction of adversarial examples – this is particularly effective in biometric systems, where inputs are compared against previous data [176]. Poisoning techniques are broadly transferable to the satellite fingerprinting context, although to the best of our knowledge no other works exist that specifically target satellite fingerprinting systems. We explore poisoning as an attack strategy in Section 4.4.6.

4.3 Threat Model

In this chapter we will cover a range of different attacks on satellite fingerprinting systems – it is therefore important that we understand the goals behind each type of attack, and the physical hardware to which an attacker is likely to have access. This threat model expands upon the generalised attacker model given in Section 2.3.7.

4.3.1 Goals

In this chapter we consider a satellite ground system that has been protected by the SatIQ fingerprinting system, to provide authentication of downlinked communication. Therefore, anyone wishing to attack the system must first overcome its fingerprinting-based defences. This may be achieved by denying service through increased false rejections, or disrupting authenticity by altering the identity of messages, causing their own transmitters to be falsely accepted. In particular, we look at the following attacks (illustrated in Figure 4.1):

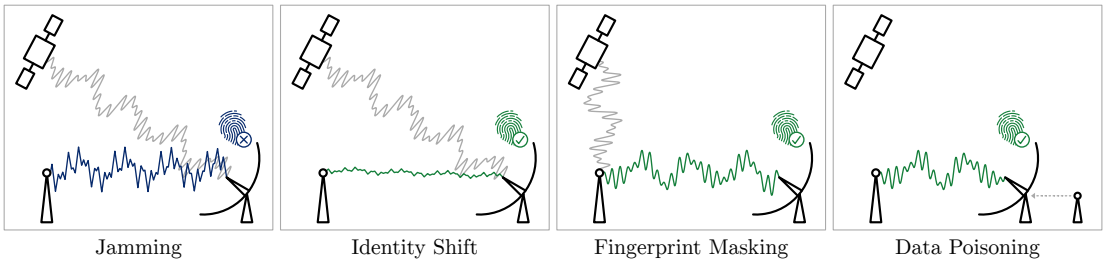


Figure 4.1: Illustration of each of the attacks described in the threat model.
(i) Jamming: the attacker broadcasts a signal to disrupt a legitimate transmitter.
(ii) Identity Shift: the attacker modifies an incoming signal to alter the fingerprint.
(iii) Fingerprint Masking: the attacker transmits a replayed message, undoing the effect of their own transmitter fingerprint.
(iv) Data Poisoning: the attacker adds malicious data to the fingerprinting system, causing their transmitter's fingerprint to be accepted without modification.

Jamming (Simple) The attacker wishes to disrupt the availability of the system by altering the fingerprint of incoming messages. This can be achieved by adding a Gaussian or tone jamming signal to incoming messages, disrupting both the message decoder and the fingerprinting system [177].

Jamming (Optimised) Instead of transmitting a general jamming signal, the attacker can instead craft a jamming signal optimised to disrupt the fingerprint as much as possible within their power limitations. Once again, this denies availability by altering message fingerprints, but has the potential to cause more disruption or to use less power.

Spoofing (Identity Shift) The attacker aims to change the perceived originator of the message, making it appear as though it has been sent from a different transmitter. This may involve overshadowing a message (or part of a message) and changing the transmitter ID, thus requiring the attacker to change the fingerprint to match the new transmitter. Alternatively, this may manifest as an "assisted attack", in which the attacker compromises a satellite that is attempting to impersonate other transmitters in the constellation – in this case, a ground-based transmitter can assist the compromised satellite by shifting the fingerprint to match the spoofed messages.

Spoofing (Fingerprint Masking) The attacker transmits an arbitrary message from a ground-based SDR, preceded by a replayed or generated message header from the satellite they are spoofing, in an attempt to mimic the fingerprint. However, the fingerprint is altered by their transmitter hardware, so they must undo the effect of their transmitter's fingerprint in order for the message to be accepted.

Data Poisoning The attacker replaces the example messages used to identify a given transmitter with their own messages, causing their own transmitter's fingerprint to be accepted as legitimate. Depending on the poisoning process, the result can be inclusive (the victim's original hardware is accepted alongside the attacker's transmitter) or exclusive (the victim hardware is rejected, and only the attacker is accepted).

The impact of these attacks is varied, and can target multiple different aspects of the satellite system. By focusing on payload data, the attacker can deny service to applications making use of the payloads by jamming communication, or inject their own data to affect downstream applications using the data (for example, by causing forest fire detection systems to report fake fires) or even exploit vulnerabilities in the data processing pipeline itself [9]. If the attacker instead targets Telemetry, Tracking, and Control (TT&C) communication, they could block diagnostic messages from being received, or alter the contents of these messages, leading the operator to send corrective commands for problems which do not exist: for example, raising or lowering the temperature, correcting spin, or even altering the satellite's orbit. By poisoning the anchors used for authentication, the attacker can continue to execute these attacks in a much more persistent way, sending false commands without needing to worry about altering the fingerprint of their own transmitter.

4.3.2 Capabilities

As in Section 2.3.7, we assume the attacker has access to commercial off-the-shelf SDR hardware, such that they can transmit signals in the vicinity of the victim ground station and have it picked up by the target receiver. The attacker is

assumed to run their SDR at a sample rate at or below the sample rate used by the fingerprinting system – 25 MS/s in our experiments. It has already been established in related work that an attacker equipped with an AWG can perfectly replicate signals down to the fingerprint in an experimental setting [120], but due to the high cost of this hardware we consider it to be out of scope for this work. We instead primarily focus on SDR hardware, which impairs the signal with its own unique fingerprint that must be counteracted in the case of spoofing attacks.

Since the attacker transmits messages over the air, we assume the attacker can achieve time synchronisation with the victim receiver, enabling them to transmit targeted interference (for example, jamming signals) over the top of legitimate messages, or send their own messages and have them picked up at the victim receiver. However, we do not assume the attacker can achieve perfect synchronisation with the victim at the symbol or phase level – to do so would require a feedback loop with the victim ground system, which is not possible in an attack setting.

As part of the experimental setup in this chapter, we describe a transmit-receive loop with two SDRs connected to each other by a cable. This is described further in Section 4.5.2, and is used by the attacker to learn the fingerprint of their own hardware and attempt to remove it from their replayed message. Of course, it is highly unlikely that an attacker would be able to connect their own SDR hardware to the legitimate ground station in order to directly measure the impact of their transmitter on the fingerprint; furthermore, to do so would negate the need for the attacks described in this chapter in the first place. Instead, the attacker can set up the transmit-receive loop with two SDRs under their control, train the system to carry out the desired attacks, and then deploy their attacks over the air, using the transmitting SDR only. One caveat to this approach is that any trained model will learn the fingerprint of both the transmitting and receiving SDRs; we have already shown in Chapter 3 that changing the receiver has a small but measurable effect on the observed fingerprint of the transmitter. This might be sufficiently small as to be insignificant, but if the attacker wishes to remove the receiving SDR's fingerprint, they might use multiple different SDRs during

training so there is no single consistent fingerprint for the model to learn – this is similar to our approach in Section 3.7.5, in which we train a fingerprinter on multiple receivers to improve performance across the board.

When using the transmit-receive loop, we assume perfect phase synchronisation between the original message and the attacker's additions. This does not break with previous assumptions, since synchronisation is only applied during training – once the system is deployed in the real world, spoofing is carried out by taking previously recorded legitimate messages, adding the attacker's signal on top, and retransmitting the combined signal. Phase synchronisation with the victim receiver is therefore not required. If the attacker wishes instead to spoof identities by modifying the fingerprint of an incoming message in real time, phase synchronisation would be required, possibly in addition to reactive jamming techniques; however, this is out of scope for this work.

During poisoning attacks, we assume the attacker has some mechanism by which they can introduce malicious data into the fingerprinting system. This could be achieved using an AWG, or via an alternative side channel into the system itself (for example, if the machine hosting the fingerprint examples has been briefly compromised). Although this attack is initially more challenging, it has a much higher success rate in the long term: once the data has been altered to match the attacker's hardware, they do not need to worry about masking their own fingerprint, as it is being accepted with the same rate as any other transmitter. If inclusive poisoning is used, the victim's original transmitter will also continue to be accepted alongside the attacker's hardware, increasing the longevity of the attack.

Finally, for most of our experiments we assume the attacker to have access to the underlying weights of the fingerprinting model. This is a reasonable assumption to make in many cases, since for many existing fingerprinting systems the code, dataset, and model weights are openly available. However, even if the attacker does not have access to this data, it has been shown in previous works that the same attacks can be achieved by training a "surrogate model" on a similar dataset, executing the attacks on this model, and transferring the attack to the original

system [175, 178]. We demonstrate a similar outcome through the use of a GAN in the "fingerprint masking" attack.

4.3.3 Effective Attack Range

To better understand the threat posed by an attacker on the wireless channel, we consider the distance over which attacks can take place for a given transmitter configuration. This requires us to understand not only the effect of the transmitter hardware and distance on the signal level at the receiver, but also the resilience of the underlying modulation and error correction scheme. For this analysis we focus on Iridium messages, but the results can easily be repeated for any given system.

Iridium Message Structure

In order to establish the power required to disrupt communication through jamming attacks, we must first understand the structure of the error correcting codes used in Iridium messages [31]. Each message contains a fixed data portion – this part of the message is 93 bits in length, and is protected by a "BCH" error correcting code which performs error detection and correction on the received bits [179]. This code is applied to the message in three interleaved blocks of 21 bits each, with 10 parity bits to form a block of 31 total bits.[1] The encoded message, including parity bits, is given by multiplying the input message by the following polynomial:

$$g(x) = x^{10} + x^7 + x^5 + x^4 + x^2 + x + 1$$

The resulting code has a minimum Hamming distance of 5, enabling a decoder to detect up to 5 bit errors or correct up to 2 per block. Therefore, if there are more than 2 bit errors on a single message block, the block will fail to decode; this could be caused by noise, or by deliberate interference from an attacker. If the probability of an error on any bit in the message is $p = \mathbb{P}(\textit{bit error})$, then the probability of an error on a message block is as follows:

$$\mathbb{P}(\textit{block error}) = 1 - (1-p)^{31} - 31p(1-p)^{30} - \binom{31}{2} p^2 \cdot (1-p)^{29}$$

[1] An implementation of this can be found in the *iridium-toolkit* software decoder, at the following URL: github.com/muccc/iridium-toolkit.

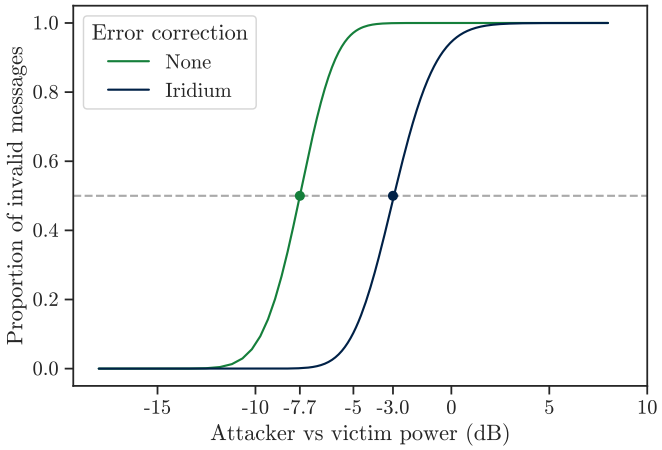


Figure 4.2: The proportion of Iridium messages which fail to decode as the jammer power increases, with and without the use of Iridium's built-in error correcting code. A dashed line represents the point at which half of all messages fail to decode.

If there is a decoder error on any block, the entire message is irrecoverable and is discarded by the receiver. Therefore the probability of a message decode error is given by:

$$\mathbb{P}(\textit{message error}) = 1 - (1 - \mathbb{P}(\textit{block error}))^3$$

Using this relationship, we can find the expected message error rate for any given bit error rate, reaching a 50% message error rate at a bit error rate of $p \approx 0.08$.

Physical Limitations

We look next at the signal modulation scheme, and how this affects its resilience against noise or interference. Iridium messages use Quadrature Phase Shift Keying (QPSK) [31], from which we can derive the relationship between attacker-to-victim ratio and bit error rate, and from there to message error rate [18]. The resulting message error rates, with and without the BCH error correction, is given in Figure 4.2, reaching a 50% error rate at $-2.98\,\text{dB}$.

We can then look at typical attacker hardware to see at what distance jamming becomes possible. Using the "high-budget custom" and "low-budget" attacker capabilities defined in Section 2.3.7, an attacker can achieve a total transmit power of $16\,\text{dBW}$ and $-28.5\,\text{dBW}$ respectively. By factoring in free-space path loss, we

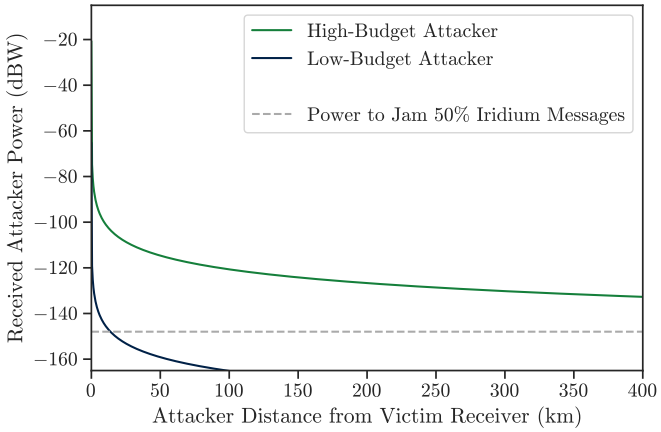


Figure 4.3: The received power of a noise jammer as distance to the victim antenna varies, under free space path loss. The dashed line represents the power required to cause a 50% loss rate of Iridium messages.

can calculate the distance at which attackers using these hardware budgets can achieve the target attacker-to-victim ratio of $-2.98\,\text{dB}$. It has been previously determined in the context of radio overshadowing attacks that the peak received power of the Iridium-NEXT constellation is $-145\,\text{dBW}$ [24]. Service is therefore denied when the attacker achieves a received power of $-145 - 2.98 = -147.98\,\text{dBW}$.

Assuming the attacker has a direct line of sight to the victim, the attenuation of the attacker's signal over distance is given by the free-space path loss formula:

$$fspl = 20\log_{10}(d) + 20\log_{10}(f) - 27.55$$

Where d is the distance in metres, and f the frequency in MHz, with $f = 1626\,\text{MHz}$ for Iridium [31].

For any given distance, we can subtract path loss from the attacker's transmit power to get the received power of the attacker's signal, enabling us to determine the distance from which jamming is effective. If the victim uses a highly directional antenna then its gain pattern must be taken into account, and added or subtracted from the final received power. However, the majority of Iridium antennas are omnidirectional, so this does not factor into our analysis.

The received power for a given distance is summarised in Figure 4.3. We can see from this that a high-budget attacker can achieve the target received power at

Table 4.1: Parameters and variables for each of the experiments.

Variable	Values	Simple Jamming	Optimised Jamming	Identity Shift	Fingerprint Masking	Data Poisoning
Number of messages during training	1, 10, 100		✓	✓[A]		
Phase synchronisation	True, False	False	✓	✓		
Attacker-to-victim power ratio	$-75\,\text{dB}$ to $5\,\text{dB}$	✓[B]	✓	✓	∼[C]	
Filter signal	True, False		✓	✓	∼[D]	
Victim messages all from same transmitter	True, False			✓		
Fingerprinter acceptance threshold	$a \in [0, 1]$					✓
Fingerprinter update threshold	$u \in [0, 1], u < a$					✓
Inclusive poisoning attack	True, False					✓

A: Due to dataset size limitations, 1, 16, 32 are used instead.
B: A different range of attacker-to-victim ratios were used, depending on the effectiveness of the jamming signal, to cover a useful range of results.
C: The attacker's modifications are transmitted alongside the original message, which is sent at a fixed power level.
D: The signal is not filtered in software, but a hardware filter is used during experiments.

a distance of over 400 km, and a low-budget attacker can do the same from over 10 km! It is surprising that communication can be disrupted at such a great distance, but it must be remembered that the attacker's transmitter is competing with a 9.5 dBW satellite transmitter, which is subject to 780 km of path loss. It is likely that in practice the attacker will need to be closer due to line-of-sight restrictions, multipath propagation, and other losses, but this result nonetheless shows that cheaply available hardware is more than sufficient to deny service at long distances.

4.4 Experiment Design

In this section we design experiments to test each of the attacks described in Section 4.3. We start by describing some of the common methodologies used across experiments, before moving to the experiments themselves, and the hardware and software architectures used. The experiments and controlled variables are summarised in Table 4.1.

4.4.1 Experimental Foundations

We first outline the common methodologies and setups shared across the experiments described in this chapter.

Base Fingerprinting Model

For each of our experiments, we use the trained SatIQ model from Chapter 3. Instead of classifying messages as belonging to a specific transmitter, this system compares two message fingerprints to one another using cosine distance, giving a distance metric which is used to authenticate the transmitter, or reject illegitimate communication. Accepting or rejecting a message depends on the distance threshold set by the operator, and can be increased to make the system more strict, rejecting a higher proportion of illegitimate messages at the expense of accepting fewer legitimate messages, or decreased to make it more lenient. For the majority of this work we do not consider a specific threshold, but instead consider the distance as a raw value (or in some cases consider a range of thresholds), to give operators an insight into how setting the threshold affects the security of their system under each of the tested attacks.

Dataset

Alongside the model, we also use the dataset of Iridium messages gathered in Section 3.5, comprising Iridium messages from 66 satellites, each of which have 48 transmitters [31]. In all the experiments in this chapter, all signals are generated at the same sample rate as the original dataset, 25 MS/s. They are then scaled to a consistent overall energy level, and the signal is rotated by a random phase offset between 0 and 2π, to mimic the difficulty of perfect phase synchronisation at such a high sample rate.[2]

Filtering

Some of the experiments in this chapter require signals to be filtered in software, in order to prevent unrealistic wideband interference. However, the SciPy signal processing functions are not differentiable by TensorFlow, so they cannot be incorporated into models or used during gradient descent. We therefore reimplement the filter function directly as a convolution over the signal, enabling differentiation

[2]Note that only the attacker's additions are rotated in this manner – the original dataset corrects for phase offset, so the victim messages are not rotated.

by TensorFlow. Whenever this filter is used in experiments, we use a Finite Impulse Response (FIR) filter with a cutoff of 0.333 MHz and 128 taps.

4.4.2 Simple Jamming

We start with simple jamming attacks, testing the resilience of SatIQ to simple Gaussian and tone jamming attacks. In doing so, we test the hypothesis that satellite receiver systems are more easily disrupted when fingerprinting systems are used to authenticate communication, falling to simple attacks more easily than the message decoder.

We have already established the attacker's constraints and ability to jam communication by disrupting the message decoder in Section 4.3.3; we compare this to the required power for an attacker to disrupt the fingerprint of a legitimate message through various conventional jamming techniques. We assess this by taking a number of different conventional jamming techniques, applying them to legitimate Iridium message headers, and finding the false rejection rate of the resulting messages by the SatIQ system. We set the acceptance threshold (i.e. the distance between two fingerprints below which the message is accepted) such that 95% of legitimate messages are accepted,[3] and look at the fingerprint distance and resulting False Rejection Rate (FRR) as the power of the jamming signal increases.

We use the experimental hardware described in Section 4.5.1 to add varying levels of Gaussian noise to incoming Iridium messages, in addition to collecting clean messages and adding various jamming signals in software – this is described further in Section 4.5.1.

4.4.3 Optimised Jamming

Moving on from simple jamming attacks, we look next at attacks which find optimised signals to disrupt the message fingerprint as much as possible within a given power limitation. Our goal in this case is to find a generalised jamming signal that works across many different messages which, when added to the synchronisation

[3]This threshold can be adjusted to accept more legitimate messages at the cost of easier spoofing (or vice versa). We choose 95% as a good middle ground.

header for a victim message, increases the distance from the message fingerprint to a reference anchor so that the message is rejected. To find this, we perform gradient descent directly on the samples of the jamming signal. At each step of gradient descent, we filter the jamming signal, normalise it so the attacker-to-victim power ratio matches a pre-defined value, and add it to the victim's signal. Finally, we define the loss function to maximise the distance between the fingerprint of the jammed signal and the fingerprint of the original victim signal.

To find a jamming signal that can be generalised across messages, we apply the jamming to a set of multiple different messages during training, taking the mean fingerprint distance of the resulting jammed signals when calculating loss. We also optionally rotate the jamming signal to a random phase offset after each use, forcing the gradient descent to produce a jamming signal that can work at any phase offset, and is therefore effective even when the attacker cannot achieve phase synchronisation. These variables are summarised in Table 4.1.

To assess the performance of the attack, we apply the jamming signal to a new set of clean messages (separate from the training data used during gradient descent), and once again set an acceptance threshold such that 95% of legitimate messages are accepted. We compare this to traditional Gaussian jamming on the Iridium decoder and on the fingerprinting system, by looking at the attacker-to-victim ratio required to cause a 50% error rate in each case.

4.4.4 Identity Shift

Next we look at simple spoofing attacks, in which the attacker is trying to alter the fingerprint of legitimate messages to change their perceived identity. We assess this using similar methods to the jamming attack, performing gradient descent on the samples of a message, but with a modified loss function: instead of maximising the fingerprint distance from the modified signal to the original message, we instead minimise the distance to a given set of target messages. The target messages all belong to the same transmitter, so the resulting modification attempts to both remove the original fingerprint and add the fingerprint of the target signal. We

perform this attack using victim messages from the legitimate Iridium dataset, attempting to shift transmitter IDs from either a set of messages all belonging to the same transmitter, or random messages from any transmitter. These and the other experimental parameters are summarised in Table 4.1.

Similar to the jamming attack, we assess the effectiveness by looking at the change in fingerprint distance on a separate test dataset with previously unseen messages. The target messages are taken from the same class of target transmitter as was used in the training data, and the same is true of the victim messages. The test therefore measures how effective the attack is at modifying the fingerprints of the same transmitters it has seen during training, but using unseen messages to ensure the spoofing signal has not overfitted. Alongside looking at the distance between messages in fingerprint space, we also consider the False Acceptance Rate (FAR) of the spoofed messages, fixing the acceptance threshold such that 95% of illegitimate messages are rejected when no spoofing signal is present.

4.4.5 Fingerprint Masking

We look next at spoofing attacks in which the attacker is directly transmitting their own messages, using synthesised or replayed message headers in order to impersonate a specific transmitter. In order to do this, they must learn to counteract the effect of their own SDR on the fingerprint such that it cannot be detected by the receiver.

In this experiment we start by using the same architecture and parameters as in the identity shift experiments, but first passing all the attacker's messages through an SDR transmit-receive loop, described in Section 4.5.2. This means that in order to be successful the attacker's signal must counteract the fingerprint of the SDRs.

We then build on this by using a full GAN architecture, training a discriminator to distinguish legitimate messages from those that have been replayed, at the same time as training a generator model to create convincing fake messages. These are particularly popular in image generation [180], and can be adapted to work in other contexts. The architecture is characterised by the combination of two components: a *generator*, which creates adversarial examples, and a *discriminator*,

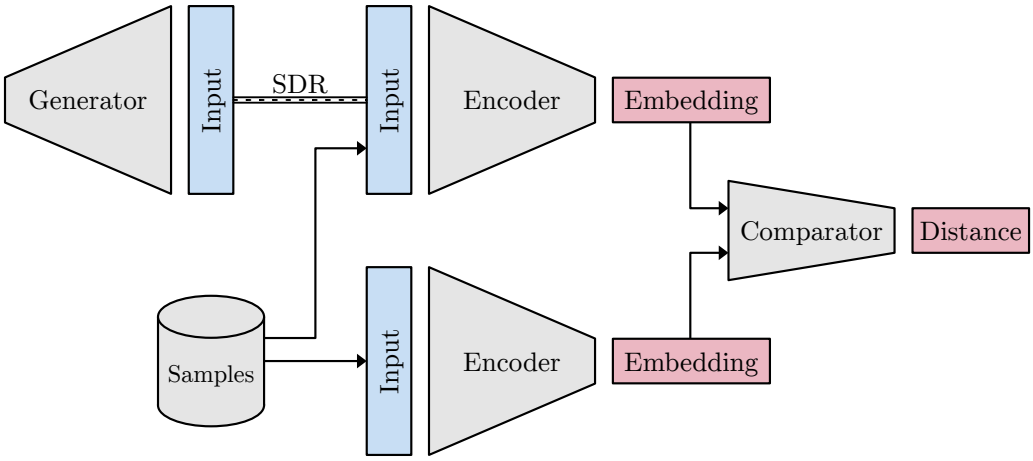


Figure 4.4: Architecture of the "Siamese GAN" model used for the fingerprint masking experiments. "SDR" represents the physical hardware transmit-receive loop described in Section 4.5.2.

which must tell the difference between legitimate and generated data [181]. Both of these are trained at the same time so that, over time, the discriminator gets better at telling real and generated data apart, thus forcing the generator to get better at creating realistic data. We adapt this architecture to work with radio signals, and once again incorporate real-world SDR hardware into the experimental configuration using the transmit-receive loop. This forces the generator to learn how to remove the impairments on the fingerprint added by the SDRs, at the same time that the discriminator is learning to detect them.

The high-level architecture of this approach is illustrated in Figure 4.4, and Figure 4.5 shows the layers of the model. The `Generator` takes message headers and generates modifications designed to remove the fingerprint of the SDR hardware. The `TxRxGenerator` takes these modifications, adds them to the original message, and passes the result through the transmit-receive loop. The `Embedder` takes a message header and reduces it to a lower-dimensional fingerprint. A discriminator is constructed using the embedder, computing the angular distance between a pair of embeddings. Taking the generator g and discriminator d, we then build two loss terms:

- For any given input i, the generator loss minimises $d(i, g(i))$, the distance between the input and the input following the generator.

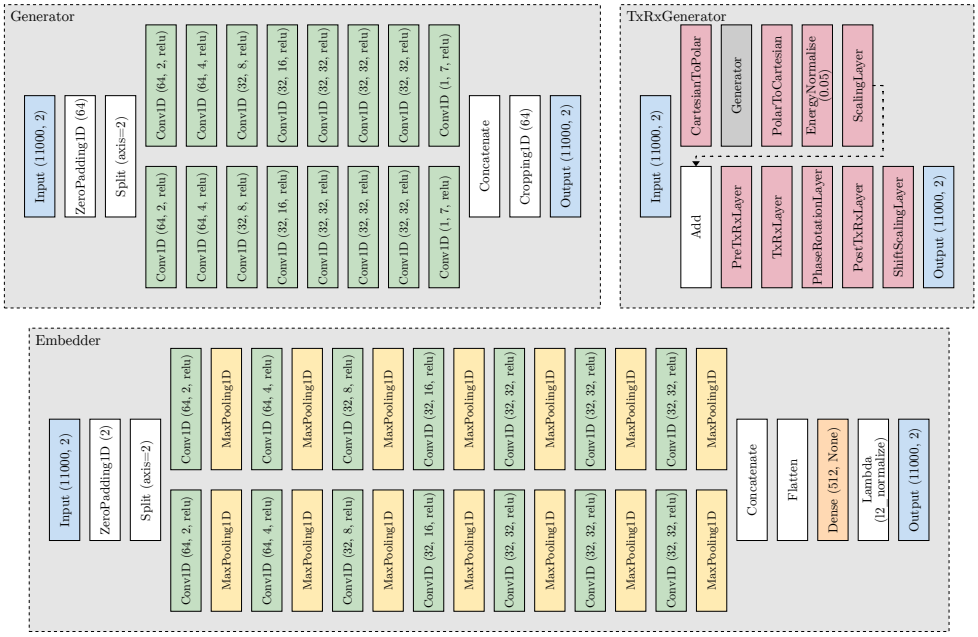


Figure 4.5: Layers of the "Siamese GAN" model used for the fingerprint masking experiments.

- For two inputs i_a and i_b, the discriminator loss encourages $d(i_a, g(i_a))$ to be close to 1 and $d(i_a, i_b)$ to be close to 0.

One beneficial side effect of this architecture is that it does not require access to the weights of the original model (as the discriminator is being trained from scratch), reducing the attacker's requirements. The trained discriminator may also be used as a fingerprinting system itself; we evaluate the efficacy of this fingerprinter in Section 4.6.6.

Once a model has been trained, we measure its effectiveness by taking a group of messages from the testing dataset, passing it through the GAN and the transmit-receive loop, and comparing the distance between the resulting messages in fingerprint space to a separate group of clean messages from the testing data.[4] We also look at the effect of the transmit-receive loop only (without the corrections added by the GAN), and compare the effectiveness of the GAN's discriminator to the original fingerprinting model. We primarily assess the performance of the GAN by looking at the Receiver Operating Characteristic (ROC) curve, which plots

[4]Since our focus is on spoofing attacks, we look at pairs of messages belonging to the same transmitter as each other, one of which has been passed through the GAN.

the true positive rate against the false positive rate as the acceptance threshold changes. The Area Under Curve (AUC) indicates the overall performance across all thresholds, with 0.5 indicating random guessing. We also look at the Equal Error Rate (EER), the point at which the false accept and false reject rates are the same.

4.4.6 Data Poisoning

Finally, we look at poisoning attacks. This technique is commonly used in biometric authentication systems, in which the examples used to authenticate new measurements are gradually replaced by adversarial examples, altering the behaviour of the system so the attacker is accepted as legitimate. In this attack, we assume the attacker has access to the weights of the fingerprinting model (although similar results could likely be achieved by training a surrogate model on the original dataset [178]) and the messages originally used by the fingerprinter. As discussed in Section 4.3, we also assume the attacker has some mechanism by which they can introduce their adversarial examples into the fingerprinting system.

The fingerprinting system at the receiver is configured to accept messages whose fingerprint distance falls below a given threshold $a \geq 0$, and to replace anchor messages when the fingerprint falls between this threshold and a separate update threshold $u \geq 0$ ($u < a$). We attempt to construct a short sequence of messages such that no message is rejected by the fingerprinter, and resulting in acceptance of the attacker's messages by the end of the sequence. This mirrors the behaviour of previous attacks on biometric systems [176], and generating adversarial examples that fall between these two thresholds is a key challenge in poisoning attacks. Other update conditions are not considered in this work, but given a different condition it would be straightforward to update the algorithm to generate messages which satisfy it.

Although autoencoders can permit simple interpolation, the specific architecture of SatIQ raises some issues with this technique: the decoder and encoder are not perfect inverses of one another, so interpolation can result in waveforms whose fingerprints differ quite significantly from one another, particularly at the start

and end of the chain. We instead find that attacks are more successful if only the encoder is used, with a sequence of steps making use of gradient descent on the raw samples in the waveform. At each step i, we perform gradient descent starting from the previous message S_i, looking for a new message header S_{i+1} which satisfies the following conditions:

$$u < \text{dist}(e(S_i), e(S_{i+1})) < a$$
$$\text{dist}(e(S_i), e(S_N)) - \text{dist}(e(S_{i+1}), e(S_N)) > u$$

Where e is the fingerprint encoding function, dist is the distance function, a and u are the acceptance and update thresholds for the fingerprints, and S_N is the target. A message header which satisfies these conditions will, at each step, be accepted by the fingerprinter and trigger an update to the anchor message. They are used as the exit condition for the gradient descent, and the loss function has two corresponding components: one to encourage $\text{dist}(e(S_i), e(S_{i+1}))$ to be between u and a, and another to encourage $\text{dist}(e(S_{i+1}), e(S_N))$ to be as small as possible. This enables us to iteratively move the fingerprint closer to the target, until the final step at which the target's messages are accepted.

The following update condition (with corresponding loss component) can also be added to the process:

$$\text{dist}(e(S_{i+1}), e(S_0)) < a$$

This results in an "inclusive poisoning" attack, in which both the attacker and victim transmitter are accepted as legitimate. Although the attack is harder to execute due to stricter update conditions, the result is more subtle, since the original victim transmitter is no longer rejected by the fingerprinting system, in addition to the attacker's messages getting accepted. The attack is therefore more likely to persist for longer without being corrected.

We assess the effectiveness of poisoning by looking at how many steps it takes to get between two messages. We consider the attack to have failed if it takes more than 50 steps, or if the gradient descent does not find a suitable next step within 1000 iterations.

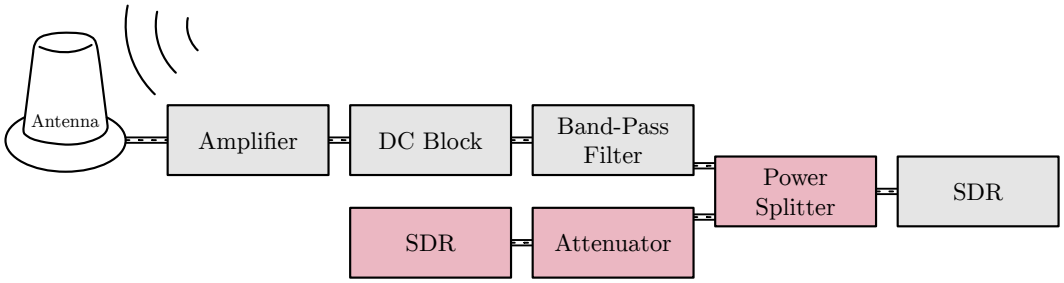


Figure 4.6: Overview of the hardware used to collect Iridium signals with additional noise. The hardware that has been added to enable variable noise injection has been highlighted.

4.5 Data Collection

In order to carry out the experiments in this chapter, we need a number of new datasets. In particular, we need data containing Iridium messages with varying levels and types of noise applied, and we need an SDR transmit-receive loop in order to measure the impact of an attacking SDR on the fingerprint of a message. In this section we describe the hardware setups we use to achieve both of these goals, and explore the resulting data.

4.5.1 Jamming

In order to carry out the simple jamming experiments described in Section 4.4.2, we need a new dataset containing legitimate incoming Iridium messages with varying levels of noise added to the messages. Our data collection setup for this is similar to the one used to originally train SatIQ (see Section 3.5), with one notable difference: an additional transmitting SDR has been connected to the receiver, allowing interference to be added onto the incoming signal. This is illustrated in Figure 4.6, and the hardware used is given in Table 4.2.

In Section 3.7.1 we note that the accuracy of the SatIQ system is impacted by the time difference between the anchor message and the incoming message to be tested. To remove this as a factor from our experiments, we interleave each of the noise levels – this is illustrated in Figure 4.7. Every 15 minutes the data collection pipeline is restarted, and the noise level N is set according to the formula

Table 4.2: Hardware used for data collection.[5]

Component	Model number	Cost (GBP)
Antenna	Beam RST740	1160
Low-noise amplifier	Mini-Circuits ZKL-33ULN-S+	205
DC block	NooElec	25
Band pass filter	Mini-Circuits VBF-1560+	43
SDR (receiving)	USRP N210, UBX 40 daughterboard	4783
SDR (transmitting)	BladeRF 2.0 micro xA4	583
Attenuator	Mini-Circuits BW-S20W20+	180
Power splitter	Mini-Circuits ZFRSC-123-S+	96

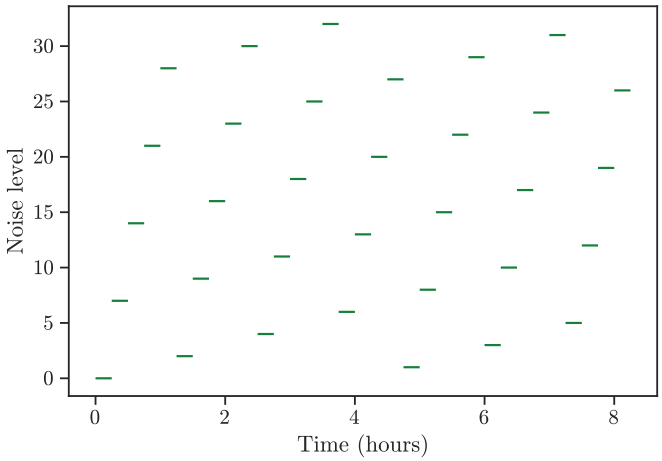


Figure 4.7: Level of noise added to the collected data over time. This pattern loops every 8 hours.

$N = 7 * i \bmod 33$, where i is the number of restarts. This ensures an even spread of noise levels over time, and keeps the time gap between different portions of the dataset consistent. A Gaussian noise function is used, generating signals with amplitude N^2. Note that the amplitude of the noise here is arbitrary and unitless, with a maximum amplitude corresponding to the maximum transmit power of the jamming SDR. In order to convert this to useful values for our analysis, we look instead at the attacker-to-victim power ratio P_a/P_v, measured in decibels and computed from the received data as follows:

$$P_a/P_v = 20 \log_{10} \left(\frac{\sqrt{(rms_a)^2 - (rms_v)^2}}{rms_v} \right)$$

Where rms_a and rms_v are the root-mean-squared amplitude of the attacker and victim signals respectively. This assumes the attacker signal includes the victim signal;

[5]Prices are accurate as of 2025.

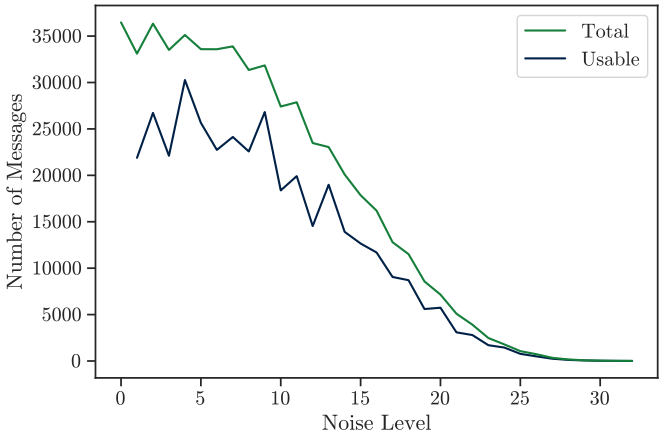


Figure 4.8: Number of messages collected for each noise level, and the number of messages that are useful for analysis (messages from a transmitter also seen in the zero-noise dataset).

if working with a clean attacker signal, the ratio is instead $20 \log_{10}(rms_a/rms_v)$.

Data Analysis

We collected data using the above setup for 33 days, during which time 540 066 messages were received. This dataset has been made openly available to aid future work.[6] As the level of noise added to the signal increases, fewer valid messages are received, an effect that is exacerbated by the lack of error correction in the decoder. This can be seen in Figure 4.8: as noise increases, the number of received messages drops off smoothly, with no valid messages received above a noise level of 32, corresponding to $P_a/P_v = 10.2\,\text{dB}$. Also shown in this figure is the number of "usable" messages: the number of messages for which there exists at least one message from the same transmitter in the zero-noise control dataset. Without such a reference message to use as an anchor, we cannot perform fingerprinting on the transmitter. As data collection runs and more messages are received, the number of usable messages will approach 100% of the dataset.

In Figure 4.9 we see the effect of adding noise onto the message headers: it is still discernible as a PSK modulated signal, but the waveform is (unsurprisingly)

[6]The dataset can be found at zenodo.org/records/10678124, and the code can be found at github.com/ssloxford/SatIQ-noise.

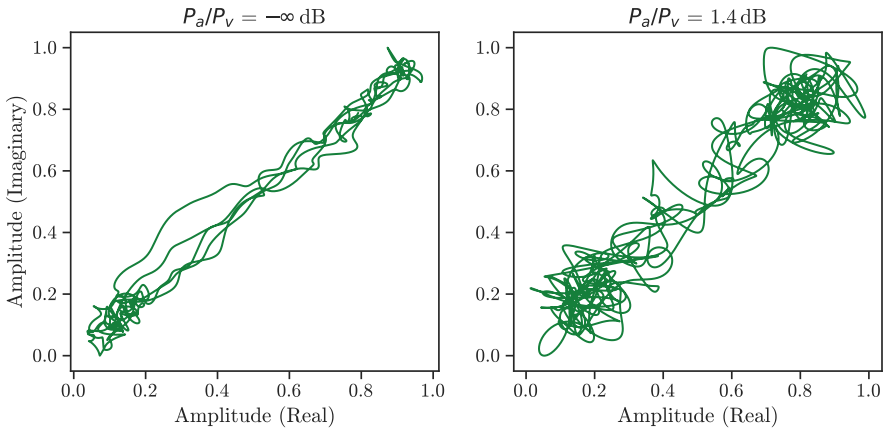


Figure 4.9: Two Iridium message headers received during data collection, depicted as constellation plots. The message on the left has no noise added, and the message on the right has Gaussian noise added, resulting in an attacker-to-victim power ratio of 1.4 dB.

significantly noisier. Any of the original impairments on the signal which could have been used to identify the transmitter have likely disappeared into this noise.

Software Jamming

We also perform a software analysis of jamming techniques, in which we add different noise and jamming signals to clean signals gathered from our data collection pipeline above. This enabled us to evaluate a wider range of jamming techniques, and removed our reliance upon the decoder pipeline – unlike the data collection above, the dataset does not shrink as we add more noise, since messages have already been demodulated and decoded.

We evaluate Gaussian noise jamming, as in the hardware experiments described above, and tone jamming, in which a constant frequency is added to the incoming signal. These are the two main jamming techniques explored in recent works and analyses of space systems [177, 182]. For our analysis, we use the following jamming signals:

- Gaussian noise captured from the data collection hardware, to obtain a signal that matches the hardware jamming experiments as closely as possible;
- Gaussian noise generated in software;

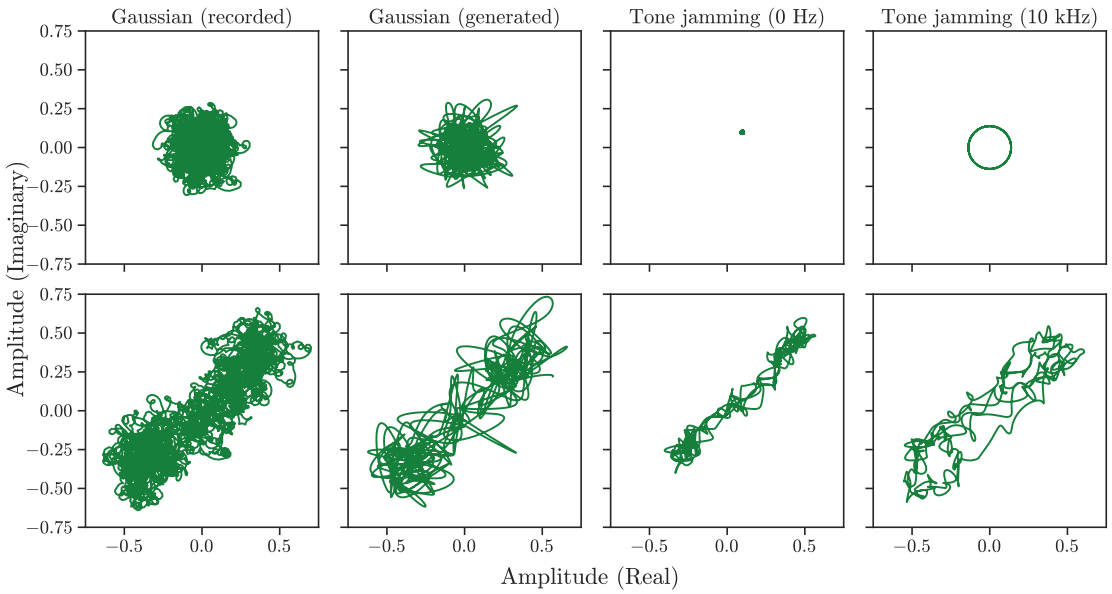


Figure 4.10: Examples of each of the techniques used in the simple jamming software analysis, as constellation diagrams. The top row shows the jamming signal by itself, and the bottom row is the same signal added onto a legitimate Iridium message header.

- Tone jamming at a relative frequency of 0 Hz;
- Tone jamming at a relative frequency of 10 kHz.

We can see each of these jamming techniques illustrated in Figure 4.10, calibrated to an attacker-to-victim ratio of −10 dB. Note that the recorded and generated Gaussian noise look slightly different – it is likely that the recorded Gaussian noise contains some higher frequencies outside the filter used by the software noise generator, resulting in a slightly different looking waveform. We can also see differences between the two different forms of tone jamming: at a relative frequency of 0 Hz, the tone jamming appears as a constant offset, shifting the position of the victim signal by a fixed amount but not changing its shape. At a higher relative frequency (10 kHz) this instead looks like a circle, shifting different parts of the waveform by different amounts.

4.5.2 Transmit-Receive Loop

Finally, in order to execute and test the optimised spoofing attacks described in Section 4.4, we need a mechanism by which we can measure the impact of

Table 4.3: Hardware used for the SDR transmit-receive loop.[7]

Component	Model number	Cost (GBP)
Clock distribution module	OctoClock-G CDA-2990	1870
SDR (transmitting)	USRP X300, SBX-120 daughterboard	9479
SDR (receiving)	USRP X300, SBX-120 daughterboard	9479
Attenuator	Mini-Circuits BW-S20W20+	180
Filter	Mini-Circuits VBF-1560+	43

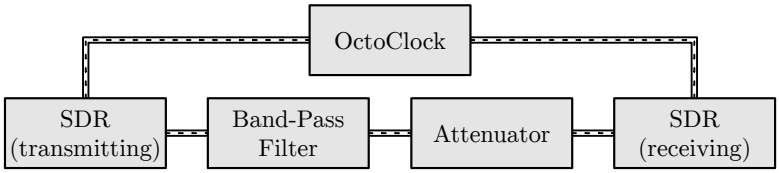


Figure 4.11: Illustration of the transmit-receive loop used for optimised spoofing attacks.

attacker hardware on the fingerprint of a signal, and the attacker's success in adding, removing, or altering it, with real-time feedback. We achieve this by building a transmit-receive loop composed of two SDRs connected to one another by a cable, configured such that arbitrary samples can be sent to the transmitting SDR, sent over the wire, and received at the other end with near-perfect time and phase synchronisation. By transmitting messages over a wire instead of over the air, we control for as many sources of noise and distortion as possible, ensuring the primary source of impairment is the fingerprint of the transmitting and receiving SDRs. Alongside increasing the difficulty of carrying out attacks, this has the added benefit of realistically filtering and attenuating signals.

The hardware used for the transmit-receive loop is given in Table 4.3, and the overall setup is illustrated in Figure 4.11. The two SDRs are connected to each other via SMA cables, with the filter and attenuator between them, and are both connected to an OctoClock to ensure perfect time synchronisation. Messages are preceded by a rising edge which is used to synchronise at the sample level, and a header with a known phase is used to correct fixed phase offsets and drift over time. The attenuator and filter protect the hardware from damage, and provide filtering characteristics matching the real-world receiver hardware.

[7]Prices are accurate as of 2025.

On top of this hardware setup we provide an easy-to-use ZeroMQ interface, enabling software to send samples over a socket and receive those same samples after they have been sent through the transmit-receive loop. This is also incorporated into a TensorFlow layer, allowing the hardware loop to be used for dataset preprocessing or integrated into the model itself, enabling it to learn the characteristics of physical layer distortions and how to counteract them. In the latter case, the gradient for the layer must be implemented separately, as the RF characteristics are not differentiable. This can be achieved by estimating the gradient from multiple measurements, or extrapolating from a single measurement. However, we found that these estimates did not perform as well as "straight through" estimation, in which the layer is assumed to have a constant gradient and causing backpropagation to proceed as though the layer applied the identity function. We use this gradient function for the remainder of our experiments in this chapter.

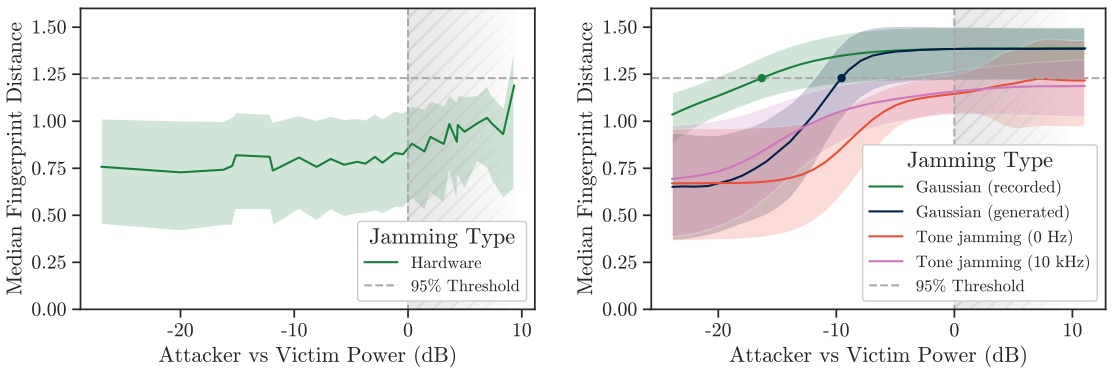
When executing the optimised jamming and spoofing experiments, we use the original dataset used to train SatIQ and, where applicable, the same underlying model. These are described in more detail in Chapter 3, and in Section 4.4.1 we describe the processing and filtering we use for the experiments in this chapter.

4.6 Results

In this section we analyse the results of each of our attack experiments. Although we ran experiments under all the given configurations, due to space constraints we focus on the most interesting results; the remaining results can be found in Appendix B.

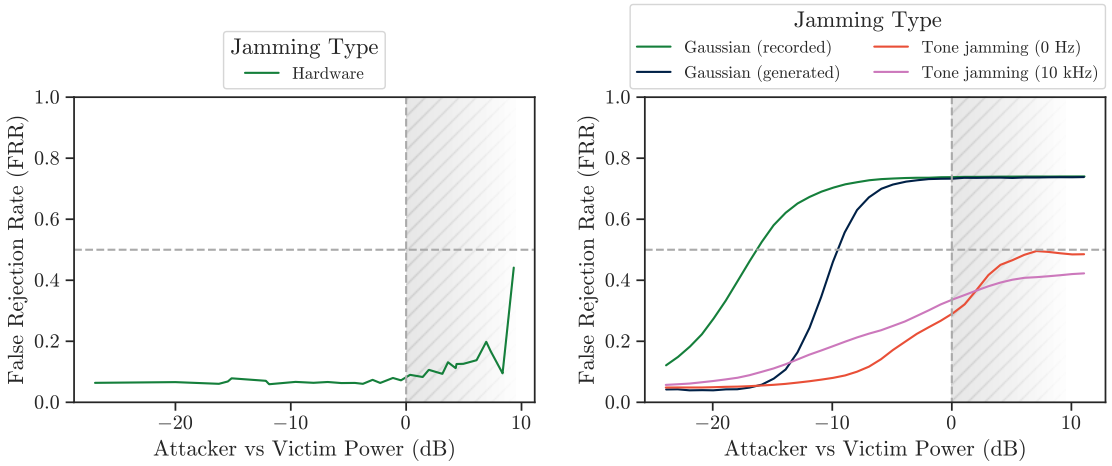
4.6.1 Simple Jamming

We start by looking at the simple jamming attacks, in which we look at the effectiveness of Gaussian jamming and tone jamming on the decoder. The effect of each attack on the fingerprint can be seen in Figure 4.12, with the resulting FRR



(a) Hardware evaluation. **(b)** Software evaluation.

Figure 4.12: Distance in fingerprint space caused by each simple jamming technique, as attacker power increases. Above 0 dB, jamming is likely to disrupt the decoder rather than the fingerprint. This region is marked in grey.



(a) Hardware evaluation. **(b)** Software evaluation.

Figure 4.13: False Rejection Rate (FRR) caused by each simple jamming technique, as attacker power increases. The region above 0 dB, in which jamming is likely to affect the decoder, is marked in grey.

in Figure 4.13.[8] The overall performance of each jamming technique is in Table 4.4. We can see that as expected, adding noise to the signal disrupts the fingerprint, and that as noise increases the variance of the distribution of fingerprints also increases. Tone jamming at both 0 Hz and 10 kHz are much less effective than Gaussian jamming against the fingerprinter, likely due to the fingerprinter filtering

[8]When assessing a given attack, FRR is by far the more useful metric, but looking at the distribution and median of fingerprint distances gives us a better insight into the attack's effect on the fingerprinter. For example, we can see whether the attack increases the overall distance, or simply increases uncertainty or variance.

Table 4.4: Attacker-to-victim power ratio (P_a/P_v) required to disrupt 50% of messages under each jamming technique, and the maximum false reject rate reached at any P_a/P_v.

Target	Jamming type	P_a/P_v (dB)	Maximum FRR (%)
Decoder	Gaussian	− 2.98	100.0
Fingerprinter	Gaussian (hardware)	—	44.1
Fingerprinter	Gaussian (recorded)	−16.3	75.2
Fingerprinter	Gaussian (generated)	− 9.55	75.0
Fingerprinter	Tone (0 Hz)	15.3	51.0
Fingerprinter	Tone (10 kHz)	—	44.8
Fingerprinter	Optimised	−20.3 [A]	67.2

A: The error rate at −34.3 dB is 47.7%, and the error rate the next step up at −20.3 dB is 66.5%. It is likely the error rate crosses the 50% threshold somewhere between these two points.

out individual frequencies – the effect of jamming at a singular frequency is limited. The hardware-introduced Gaussian noise also demonstrates limited effectiveness, getting close to a 50% FRR but requiring almost 10 dB to do so – this could be explained by the additional frequencies present in the real-world Gaussian noise data, outside the range of the filter used by the decoder.[9] On the other hand, Gaussian jamming in the software analysis is moderately effective, exceeding the performance of jamming on the Iridium decoder to achieve a 50% FRR at an attacker-to-victim ratio of −16.3 dB. This is quite a bit more effective than jamming against the decoder, raising concerns that using fingerprinting as a drop-in security measure might present a new vulnerability in the form of easier jamming attacks.

Interestingly, this differs from the results we found in [18], in which we tested the same jamming attacks against the original SatIQ model, trained on a smaller dataset. In this analysis, we found that the model withstood jamming significantly better, reaching a 50% error rate at −3.76 dB, roughly matching the resilience of the Iridium decoder. There are a number of reasons why this might be the case. First and foremost, fingerprinting in the context of satellite systems is much more difficult than terrestrial systems, due to the high levels of free space path loss and atmospheric distortion; any identifiable information present in the signal will be

[9] Of course, when the jamming signal exceeds the victim by this great an amount, the message will fail to decode long before it reaches the fingerprinting stage. This result, and others above 0 dB, are included purely to demonstrate the ineffectiveness of this technique.

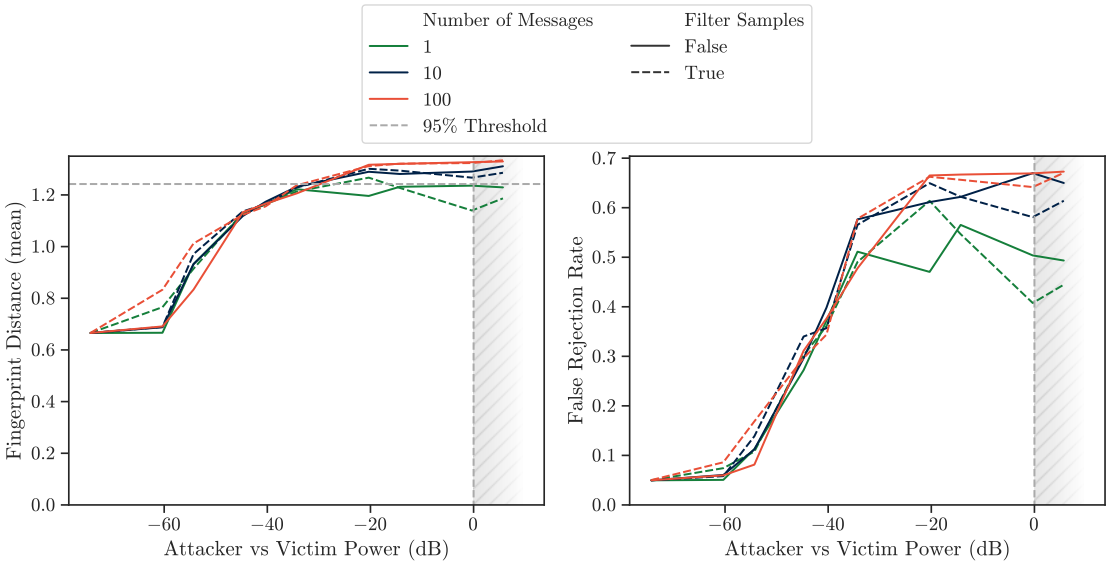


Figure 4.14: Results (mean fingerprint distance and false rejection rate) from the jamming experiments, when the attacker does not have phase synchronisation. False rejection rate is given relative to an initial 95% acceptance threshold. Jamming is effective long before the decoder is affected (indicated in grey, above 0 dB).

subject to significant atmospheric attenuation. In order to achieve good performance, the fingerprinting model must be able to overcome this, and it is likely that this sometimes results in the model becoming good at ignoring Gaussian noise. However, the fact that this does not occur in all cases suggests the property is not inherent to the model architecture. In Section 4.7 we explore the possibility of training a denoising autoencoder to make this property a core component of the system.

4.6.2 Optimised Jamming

We next look at the optimised jamming attacks; these results are summarised in Figure 4.14.[10] We can see that even at very low amplitudes it is easy to generate adversarial perturbations that disrupt the victim's fingerprint, with or without a filter applied to the signal (although the inclusion of a filter does impede performance slightly). We also see that by training the jamming signal on a larger number of example messages, performance is improved on unseen messages not present in the training data. In all cases, optimised jamming significantly exceeds the performance

[10]Results when the attacker can achieve phase synchronisation with the victim can be found in Appendix B.1, but are not discussed here as the performance is very similar.

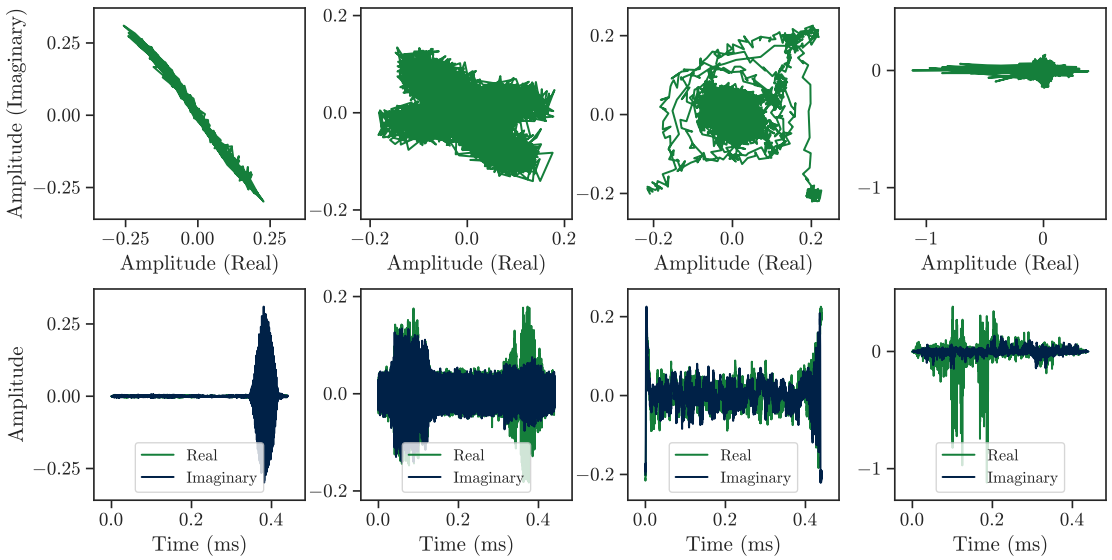


Figure 4.15: Examples of some of the signals produced by the jamming attack.

of traditional jamming techniques, achieving a 50% error rate on the fingerprinting system at an attacker-to-victim ratio below −20 dB – this is well below the noise floor, making the attack very difficult to detect. In practice, the attacker may wish to increase the amplitude of their jamming signal higher than strictly necessary in order to ensure effectiveness in a noisy environment with high levels of attenuation, particularly when they lack any feedback regarding the success of the attack.

Alongside its effectiveness this technique is also versatile, requiring no real-time input of the victim signal or generative capabilities to jam communication. Real-time reactive jamming has been shown to be technically feasible [64], but it is difficult and requires high-end hardware; it is much easier to prepare a jamming signal in advance and broadcast it in time with the message header. We can see from the results that the characteristics of optimised jamming signals are largely message- and transmitter-agnostic, so the attacker does not need to capture the message in real time, and can instead cause significant disruption by using a generalised jamming signal.

Looking at the actual jamming signals produced by this technique, they appear to converge upon a number of different jamming modes, illustrated in Figure 4.15:

- In the first mode, the jamming signal produces a high-amplitude burst over a short period of time. The time of the burst can vary, but appears to be

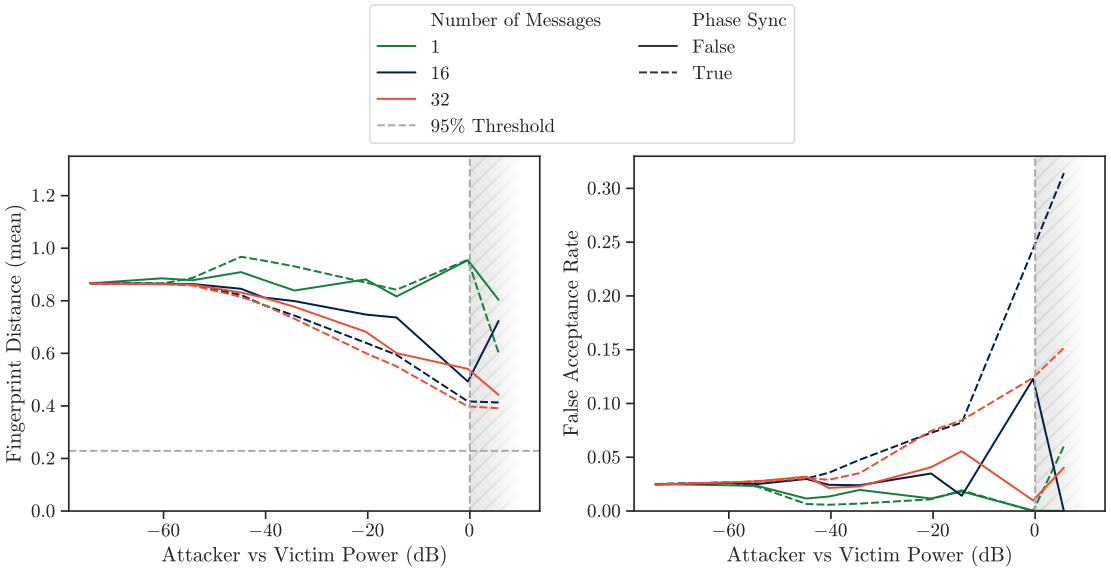


Figure 4.16: Initial results from the identity shift attack, on messages from the legitimate dataset. False acceptance rates are given relative to an initial acceptance threshold, set such that 95% of illegitimate messages are rejected. Pure spoofing dominates identity shifting above 0 dB, in the region marked in grey.

aligned with symbol transitions.

- In the second mode, the waveform has more consistent noise overall, punctuated by some bursts of noise which appear to resemble a QPSK-modulated constellation, matching the modulation scheme used by Iridium.

- In the third mode, noise is even more consistent across the message, with "swirl" patterns that vaguely resemble tone jamming.

- In the final mode (which only emerges when phase alignment is enabled), sharp bursts are aligned with the I and Q axes to produce maximal offsets.

We also note that the short impulses used in the final jamming mode may be ineffective in practice, due to the presence of filters on the receiver. They do, however, help confirm the theory in [17] that fingerprint information is derived from the transitions between symbols, as the jamming impulses are aligned with these sections.

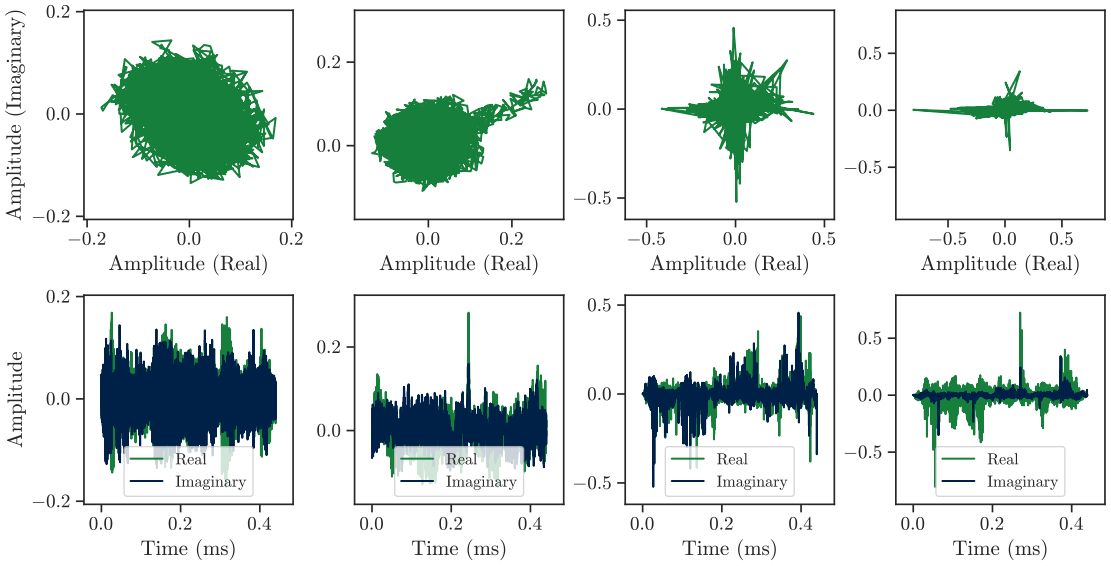


Figure 4.17: Examples of some of the signals produced by the identity shift attack.

4.6.3 Identity Shift

Next we look at the identity shift attack described in Section 4.4.4. We can see the initial results of these experiments in Figure 4.16, with a single spoofing signal optimised to modify messages from one transmitter so their fingerprint looks like another.[11] We can see that as the attacker power increases, there is a measurable decrease in the distance to the target fingerprint, showing that identity spoofing is possible. This is not the case when only a single training message is used; multiple messages are required in order to correctly learn the fingerprint characteristics. In the best case, without phase synchronisation, a False Acceptance Rate (FAR) of 12.3% is achieved. When phase synchronisation is permitted, this improves to 31.4%, suggesting the fingerprint of messages is not phase invariant; however, this does not match our threat model, since it is unlikely for the attacker to be able to synchronise perfectly with the victim message. We also see that although optimal performance is achieved at the highest power levels, good performance can still be achieved at lower amplitudes, so the original message does not need to be overshadowed entirely. Further results can be found in Appendix B.2, particularly for experiments in which the messages are not filtered, and those in which the starting

[11] Similar to the jamming experiments, testing data is entirely separate from the training data.

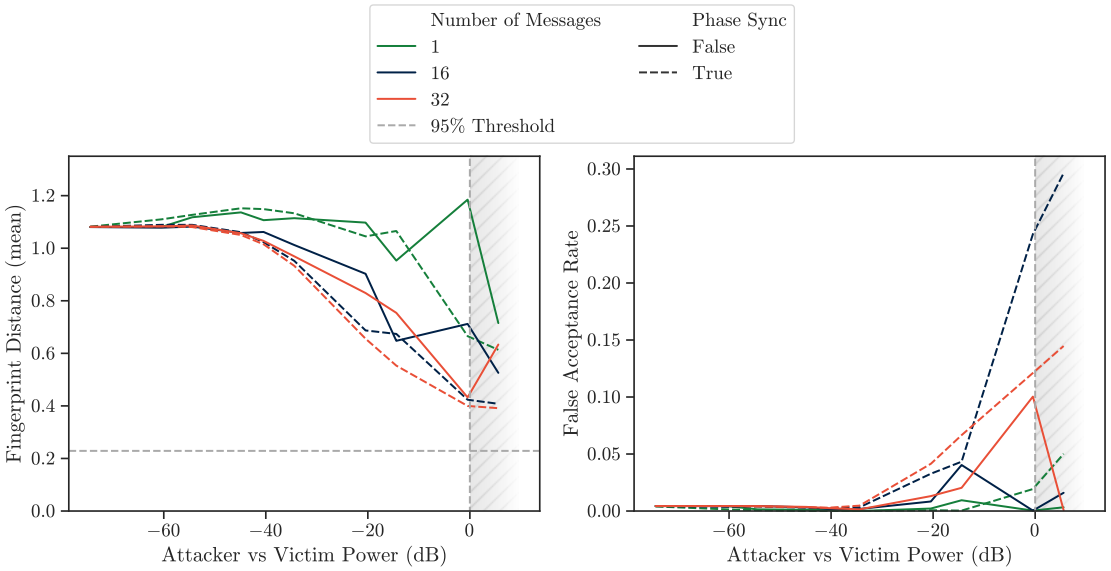


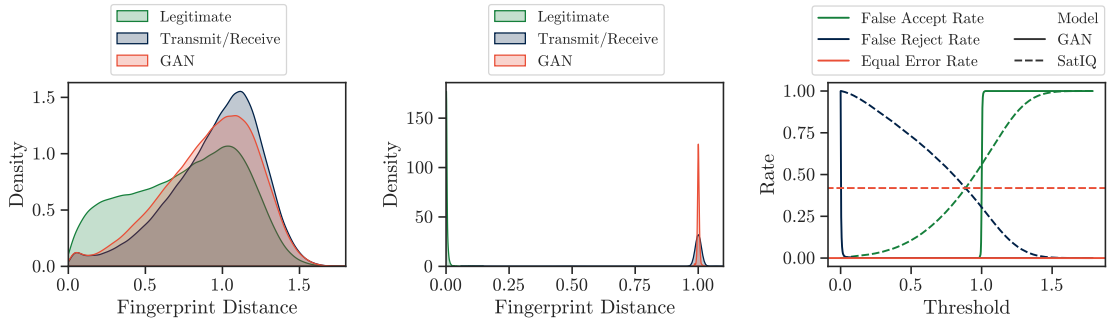
Figure 4.18: Results from the identity shift attack on messages that have been transmitted and received using an SDR. False acceptance rates are given relative to an initial acceptance threshold, set such that 95% of illegitimate messages are rejected. Pure spoofing dominates the identity shift above 0 dB, in the region marked in grey.

samples are random (i.e. from different transmitters) rather than all belonging to the same transmitter – in both these cases, attack performance is worse due to the increased difficulty of the attack.

Examples of the signals generated by this technique can be seen in Figure 4.17. These once again make use of higher-amplitude bursts, albeit with more activity elsewhere in the waveform in order to affect the fingerprint as a whole.

4.6.4 Fingerprint Masking

Next we look at fingerprint masking attacks, starting with the simple case: using the same architecture as in the identity shift attacks, but with SDR-replayed messages. This creates a more realistic attack model, in which the attacker is not only trying to shift the identity of a specific transmitter, but also to undo the effect of their own SDR. The results for this attack are in Figure 4.18, with full results in Appendix B.3. In these results, the starting fingerprint distance is measurably higher than in the identity shift attack, since the SDR hardware in the loop has imparted its own additional fingerprint upon the signal, which must be removed. For this reason, the



(a) SatIQ discriminator. **(b)** GAN discriminator. **(c)** Error curves.

Figure 4.19: Distance in fingerprint space for messages from legitimate transmitters, compared to messages sent through the SDR transmit-receive loop, and messages generated by the GAN. Tested on the original SatIQ discriminator (left) and the GAN's discriminator (middle), in addition to error curves for each model on the generated data (right).

initial FAR with no corrections applied is close to 0%. Despite this, the generated spoofing signal is able to counteract this, with a peak FAR of 10.0%, or 29.6% when phase synchronisation is permitted.

Following this, we move on to looking at the GAN architecture, in which the generator is learning to counteract the SDR fingerprints at the same time as a generator is being trained to distinguish the attacker from legitimate data. These results can be seen in Figure 4.19, in which we take a set of base messages from legitimate transmitters, and compare their distance in fingerprint space to legitimate messages, those that have been passed through the transmit-receive loop, and those created by the GAN's generator. We can see from these that the SDRs alter the fingerprint to some extent, which the GAN is able to counteract, ultimately resulting in a ROC AUC of 0.6553 and EER of 0.3909.

We can also see that it is very easy for the discriminator from our newly trained GAN to separate the legitimate messages from those passed through the SDR loop, and the effect of the generator on its effectiveness is near-negligible – it can identify the attack with an EER of less than 0.00001. This suggests there is a clear fingerprint attached to the SDR which can be detected with explicit training, but that the SatIQ model struggles to detect, due to having only been trained on legitimate Iridium transmitters.

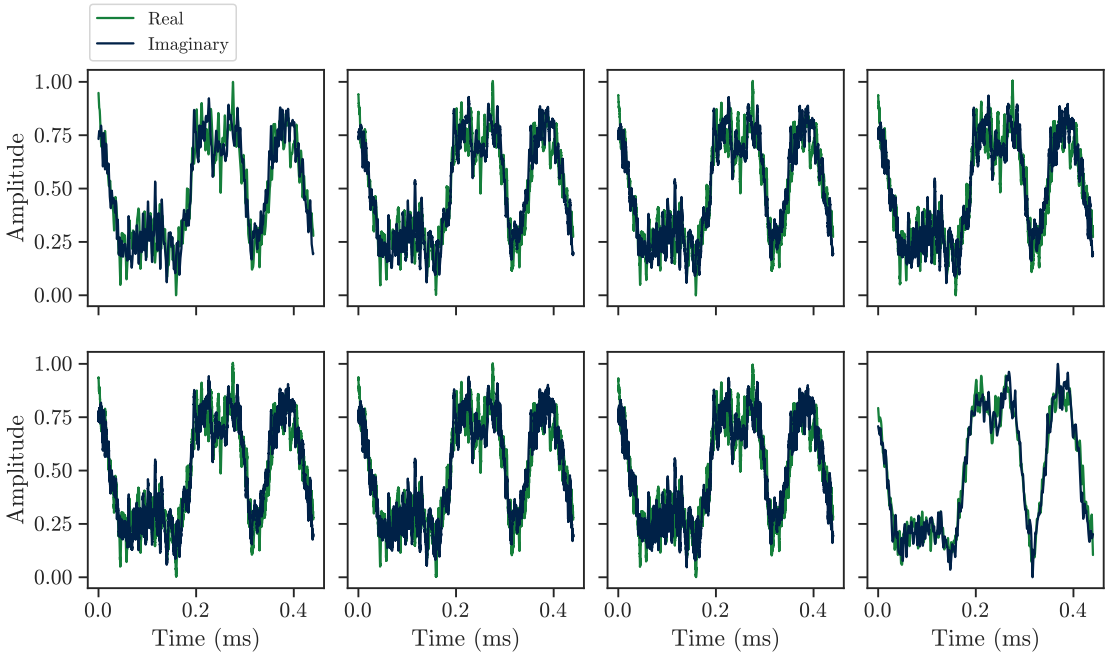


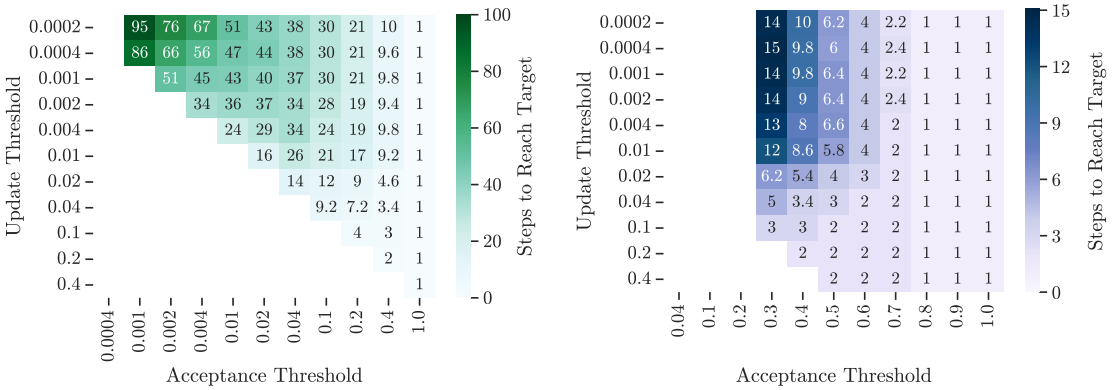
Figure 4.20: A selection of the steps involved in poisoning by interpolating between the fingerprints of two legitimate transmitters. To save space, not all steps have been shown.

Training a GAN with transmit-receive hardware in the loop opens up some interesting new possibilities in the realm of single-transmitter fingerprinting. Since the GAN's discriminator has been trained directly to detect spoofing attacks, rather than gaining this ability as an incidental side effect of training to distinguish between legitimate transmitters, it demonstrates much better performance when detecting attacks. We evaluate this further in Section 4.6.6, and discuss how it can be used in practice to secure single-satellite systems, enabling fingerprinting techniques to be used outside the realm of large satellite constellations.

4.6.5 Poisoning

Finally, we look at the poisoning attacks described in Section 4.4.6, in which the anchors are gradually updated to include the fingerprint of an attacker-controlled SDR.

We start by demonstrating the technique on legitimate messages from the training dataset, poisoning the fingerprinter such that it recognises transmitter B as transmitter A. An example of the steps in this process can be seen in Figure 4.20. In this figure we can see the noise-removing effect of the fingerprinter's autoencoder:



(a) "Exclusive" poisoning: original anchor not included in the target.

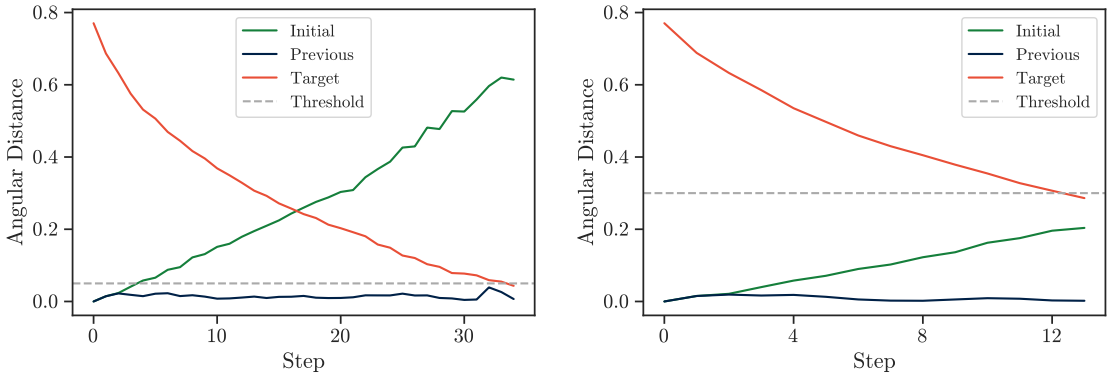
(b) "Inclusive" poisoning: original anchor included in the target.

Figure 4.21: Number of steps required to poison the reference fingerprints to accept one transmitter instead of another, given the acceptance threshold and update threshold.

even though the final waveform generated by this process has more noise than the target waveform, the fingerprints are still highly similar to one another. We also see that, similar to the jamming and spoofing experiments explored earlier, altering the fingerprint does not require a signal with a huge amplitude, and the use of a low-pass filter does not impede performance (although it does produce substantially different-looking adversarial signals). Interestingly, the signal does not need to be higher amplitude in the filtered case. In order to produce properly optimised signals, the attacker will need to obtain the approximate parameters of the receiver so the generated messages properly match.

The poisoning technique can be repeated for any acceptance threshold and target threshold; Figure 4.21 illustrates the number of steps required for a range of thresholds.[12] Note that the process is more difficult (either taking more steps or failing to converge entirely) if the thresholds are very close to one another, or if the acceptance threshold is sufficiently low. Similarly, if the acceptance threshold is below approximately 0.3 and inclusive poisoning is used, the technique fails to find a next step and fails. If the acceptance threshold is too low, then the fingerprints of two different transmitters are sufficiently far apart that it is not possible to find an anchor that includes both the victim's original transmitter and

[12]To ensure computation finished in a timely manner, gradient descent was aborted after 1000 iterations if a suitable next step was not found.



(a) "Exclusive" poisoning: original anchor not included in the target.

(b) "Inclusive" poisoning: original anchor included in the target.

Figure 4.22: The distance, in fingerprint space, from each step in the poisoning process to the initial/target/previous step, for both exclusive and inclusive fingerprinting.

the attacker's transmitter, making inclusive poisoning impossible. We can also see this in Figure 4.22 – the distance to the initial and target fingerprint are very similar for both exclusive and inclusive poisoning, but the inclusive strategy is forced to terminate earlier as it fails to find a suitable next step. Despite this, an attacker may be able to use inclusive poisoning when the acceptance threshold is sufficiently high, and in other cases may take advantage of inclusive poisoning to evade detection for longer – the initial stages of poisoning can be completed inclusively to evade detection, switching only to exclusive poisoning when forced to do so.

Finally, we demonstrate that this poisoning attack can work on arbitrary data and fingerprints, even those that do not resemble any legitimate transmitters seen by the model during training. We show this by generating a random fingerprint and poisoning the reference fingerprints so the model recognises this false transmitter as legitimate. An example of this can be seen in Figure 4.23. Note once again that the final message differs quite noticeably from the generated poisoning samples, likely due to the denoising effect of the fingerprinter. This works in the attacker's favour, as they do not necessarily need to send messages which resemble the target fingerprint at first glance.

Through these results we have shown that the fingerprinter can be poisoned to accept any target transmitter as legitimate, by injecting a short sequence of

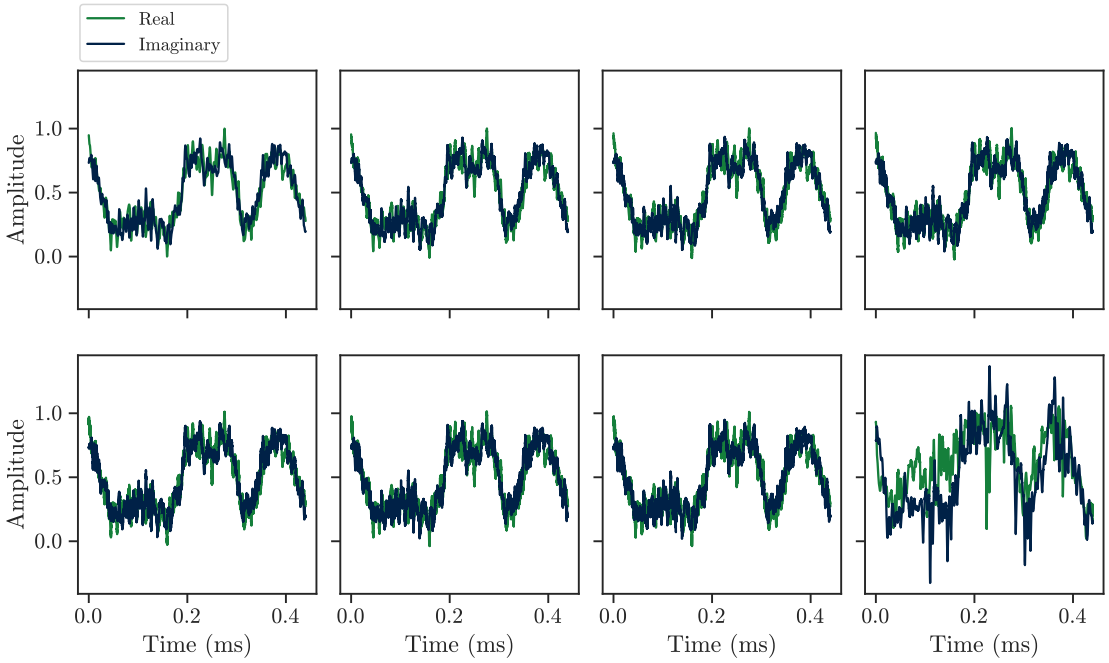


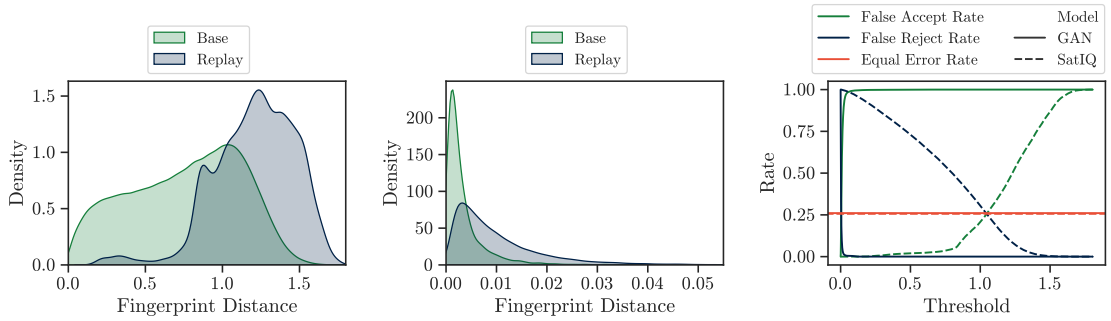
Figure 4.23: A selection of the steps involved in interpolating between a legitimate transmitter's fingerprint and a random fingerprint. To save space, not all steps have been shown.

controlled messages to guide the reference anchors away from the original transmitter. This can certainly be achieved using an AWG, but it remains to be seen whether off-the-shelf SDR hardware could deliver the same result. In Section 4.7 we discuss improvements to the fingerprint update function which might mitigate this attack.

4.6.6 Single-Transmitter Fingerprinting

Alongside optimised spoofing attacks, the architecture of the GAN-based attack in Section 4.6.4 also opens up new opportunities in the area of single-transmitter fingerprinting. Unlike the original SatIQ model, which has learned how to detect spoofing attacks as a side effect of learning to distinguish between legitimate transmitters, the GAN has been trained directly to detect spoofing attacks, so its performance is significantly better. We validate this by testing the model on the message replay dataset used to test SatIQ in Chapter 3, which was created using completely different hardware from the experiments in this chapter.

The results of this analysis can be seen in Figure 4.24, with ROC curves in



(a) SatIQ discriminator. **(b)** GAN discriminator. **(c)** Error curves.

Figure 4.24: Distance in fingerprint space for messages from legitimate transmitters, compared to messages from the replay dataset provided in [16]. Tested against the original SatIQ discriminator (left) and the GAN's discriminator (middle), in addition to error curves for each model on the generated data (right).

Figure 4.25. Results are summarised in Table 4.5. From these we see that in addition to detecting the spoofed messages from its own training loop with near-perfect accuracy, the GAN also displays comparable performance to SatIQ on the replay dataset, from transmitters it has never seen before. On this data, it achieves an AUC of 0.8094 and EER of 0.2605 – only a slight performance drop from SatIQ's EER of 0.2564, despite having only been trained on a single attacker transmitter.[13] Importantly, this is achieved without ever having seen replayed messages from the attacker SDR during training, giving us confidence that the model will protect against a wider range of attackers than just those seen in the training data.

This result opens the door for fingerprinting on individual satellites: instead of gathering a dataset from a whole constellation of satellites, operators can instead collect messages from a single transmitter, and train a model to distinguish between these messages and an attacking SDR. This could be done in the lab and even updated on the fly to incorporate different SDRs or to add new transmitters to the system; we discuss this further in Section 4.7. This lies in contrast to all previous works, which focus on larger constellations with many transmitters, and cannot be used to secure individual satellites or transmitters. A "conditional GAN"

[13]Note that the results here compare against only a single anchor; performance can be improved further by using multiple anchors.

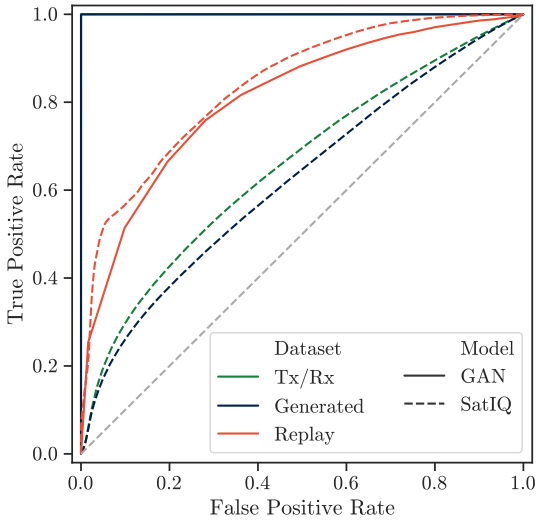


Figure 4.25: ROC curves showing performance of the GAN and SatIQ model at distinguishing legitimate communication from attacker-replayed communication in each dataset.

Table 4.5: Performance of the newly trained GAN discriminator compared to the original SatIQ model, for each of the tested datasets.

Discriminator	Attack	AUC	EER
GAN	Transmit/Receive	0.9999	0.0007
	GAN	0.9999	0.0001
	Replay	0.8094	0.2605
SatIQ	Transmit/Receive	0.6553	0.3909
	GAN	0.6224	0.4191
	Replay	0.8399	0.2564

architecture [181] could also be used to bridge the gap to providing transmitter-specific authentication in a small constellation, given a sufficiently large dataset of legitimate messages from each transmitter. With recent increases in attacks on satellite systems, even those equipped with cryptographic security, any additional authenticity and signal intelligence that can be applied on top of existing systems is invaluable. By expanding the scope of previous works, our results enable the detection and mitigation of attacks even on individual satellites, thereby providing more robust and resilient communication and enhancing the overall security of satellite-dependent infrastructure.

4.7 Discussion

The experiments presented in this chapter have demonstrated the vulnerability of satellite fingerprinting to various attacks, including jamming, spoofing, and dataset poisoning. While these attacks can be effective, there are some limitations and potential mitigations that can be explored. For instance, comparing incoming messages against multiple reference anchors or taking a rolling average from multiple messages can reduce the false rejection rate and make attacks more difficult to execute, requiring the attacker to synchronise their signal to multiple messages in a short time period in order to have a significant effect. The presence of filters in standard receiver hardware may also affect attacks, although our results show that including these filters in the adversarial training pipeline enables attacks to remain effective with a minimal performance drop.

When considering the jamming results in particular, it is important to note that we are only considering a single instance of a trained model – it is possible that random differences in training might lead to different results, particularly as jamming is not at all considered during the training process. We have seen that random variation during the training process can lead to models that have different levels of resilience to jamming attacks, with the results in [18] showing increased robustness against Gaussian jamming, despite using the same model architecture on a very similar dataset. Randomness is not a particularly desirable outcome, so operators may instead decide to incorporate jamming attacks into the training process directly, through the use of "denoising autoencoders" – these train a model on data with additional noise present, thus requiring the model to remove noise in addition to recreating the input [183]. These have already been adapted to work in a wide range of applications [184, 185], including RF fingerprint identification [132], improving identification accuracy under higher levels of Gaussian noise. This could be applied to the SatIQ architecture to provide additional resilience against jamming attacks.

4.7.1 Countermeasures

There are a number of other countermeasures which may be deployed to protect against the attacks we have explored. One approach to protect against dataset poisoning is to prevent attackers from controlling when anchors are updated, which could be achieved by triggering updates manually within a given time window, or by randomly choosing anchors from the most recent N messages sent from a given transmitter. This can make poisoning attacks substantially more difficult to execute, requiring persistent effort from the attacker over an extended period of time.

Another approach to countering attacks is to combine multiple sources of signal intelligence to provide improved coverage against attacks. Although the optimised attacks in this chapter are effective against the fingerprinting system tested, they may be detected by a different mechanism – for example, the high-amplitude bursts exhibited by some of the jamming signals in Section 4.6.2 could be identified by monitoring the signal-to-noise ratio over time. By combining multiple methods, satellite communication systems can be designed to avoid having a single point of failure.

The single-transmitter fingerprinting techniques discovered in this chapter offer another promising countermeasure against spoofing and replay attacks. By training a model directly on detecting spoofing attacks, it can be more effective in this area, even when the specific attacker SDR was not present in the training dataset. Alongside enabling fingerprinting in smaller satellite systems, this technique also opens up the opportunity for centralised training: a single SDR loop system can be used to train many models at once, targeting different satellite systems. This could even result in system-agnostic fingerprinting, with a model trained to detect attacks regardless of the underlying signal modulation scheme.

Finally, we can draw inspiration from existing systems in their approach to countering attacks [156, 168]. These include training classifiers on an augmented dataset containing adversarial examples ("adversarial training") [186, 187], or using techniques like randomised smoothing and defensive distillation to smooth gradients, thus reducing the attacker's ability to find adversarial perturbations [159, 188, 189].

Preprocessing techniques are also proposed, such as training an auxiliary detection model to remove adversarial perturbations prior to the classifier [160, 190–192] (optionally also taking into account characteristics of the physical RF channel [193]) or using an ensemble of classifiers to counteract one another's weaknesses [194]. Finally, "certified defence" attempts to verify that a model cannot be attacked by adversarial examples, providing a certificate that proves that no adversarial perturbation below a given amplitude can result in more than a given level of misclassification on the test data [195] – this has been demonstrated on wireless signal classifiers [159]. Many of these techniques work with distance-based systems, or could easily be adapted to do so, and could therefore result in significantly more robust satellite fingerprinting systems.

By learning from these systems and combining multiple countermeasures, satellite communication systems can be made more resilient to attacks.

4.7.2 Future Work

Ultimately, our results show that no single security measure can be immune to attacks, and highlight the need for continued research and multi-faceted countermeasures. Future work should therefore focus on developing and evaluating combined countermeasures to provide broad coverage against various types of attacks. Additionally, the development of standardised testing frameworks would enable operators to assess their systems' resilience against different types of attacks in a consistent and reliable manner, and deploy new countermeasures.

Future work might also expand the scope of our experimental setup. For example, when carrying out our hardware experiments, we do not employ a channel model between the transmitting and receiving SDRs. This is not a major limitation, as the messages sent from the original satellite have already been impaired by the wireless channel. The attacker's signal is added on top of this, followed by some small distortion from the wired channel. It is important to distinguish between the impairments from the original satellite channel and those from the attacker's channel, as they have distinct characteristics – the attacker's channel is likely to have a much

smaller degree of attenuation, particularly if the attacker is transmitting close to the victim receiver. Although the absence of a channel model does not significantly impact results, it may be interesting for future work to explore how a channel model might impede performance, and to see if a GAN can learn to counteract the SDR's fingerprint in the presence of greater levels of channel distortion. This can be achieved using specialised hardware, such as Keysight's "F8820A PROPSIM FS16 Channel Emulator", costing approximately 150 000 USD.[14] By attaching this hardware inline with the rest of the experimental hardware, varying levels of attenuation and fading can be added to the signal in real time.

4.8 Conclusion

In this chapter we have explored a wide range of attacks against satellite fingerprinting systems, providing a better understanding of the potential vulnerabilities and threats faced by these systems.

Our results demonstrate that, like all machine learning systems, satellite fingerprinting systems are inherently vulnerable to adversarial perturbations and other targeted attacks, with varying degrees of success. It is always important to consider attacks when designing, building, and deploying systems such as these, to ensure the result is robust and suitable for deployment in security-critical contexts. For instance, the vulnerability of SatIQ to targeted jamming means it would be unwise to deploy it in such a way that all messages which fail the fingerprint check are immediately discarded – instead, operators could use strict fingerprint authentication only during a connection establishment phase, reverting to a less strict version once session keys have been derived. This provides many of the benefits of fingerprint-based authentication, and allows them to still be used for attack detection without substantially affecting the attack surface.

We also find that the incorporation of an SDR transmit-receive loop into training enables the creation of a fingerprinting system that can identify attacks, even when the training dataset contains only a single transmitter. Alongside

[14] Price estimate provided by Keysight; accurate as of 2025.

mitigating the attacks explored in this chapter, this improvement also makes fingerprinting significantly more useful for legacy systems: previous techniques required a dataset comprising messages from many different transmitters, in order to learn characteristics of each. Our single-transmitter technique does not need this type of data, since it is trained directly on simulated attacks; a much smaller dataset of messages from one transmitter is all that is required. It can therefore be trained and deployed to protect legacy systems, many of which comprise only a single satellite, providing security where it is most urgently needed.

5

Effective Authentication and Key Revocation in Interplanetary Networks

Contents

5.1 Motivation 140
5.1.1 Contributions 142
5.2 Related Work 143
5.2.1 Internet Drafts 143
5.2.2 Academic Works 145
5.2.3 Simulators 146
5.3 Threat Model 147
5.4 Framework Design 148
5.4.1 Scenarios 150
5.5 Experiment Design 151
5.5.1 Key Management Systems 152
5.5.2 Network Topologies 154
5.5.3 Routing 158
5.5.4 Traffic Modelling 159
5.6 The Deep Space Network Simulator (DSNS) 159
5.6.1 Architecture 160
5.6.2 PKI Functionality 163
5.6.3 Extending DSNS 164
5.7 Results 166
5.7.1 Connectivity 166
5.7.2 Connection Establishment 169
5.7.3 Key Revocation 173
5.8 Discussion 177
5.8.1 Limitations and Future Work 178
5.9 Conclusion 182
5.9.1 Availability 183

In previous chapters, we focus on physical layer authentication of satellite communication through transmitter fingerprinting. Although we discuss its usefulness across the board, its primary use case lies in protecting Old Space systems which lack suitable cryptographic security. However, as New Space continues to grow and become more relevant, it raises a number of interesting security questions of its own. One particularly pressing need in this area is the need for effective Public Key Infrastructure (PKI) in larger satellite networks.

In this chapter, we develop a standardised framework for comparing PKI systems across various network topologies. To test various protocols, we build the Deep Space Network Simulator (DSNS): a new network simulator optimised for large space networks. We use this simulator to demonstrate that terrestrial PKI protocols can be adapted for effective use in interplanetary networks, and propose some novel optimisations to improve performance and reduce the impact of attacks.

5.1 Motivation

Due to the decreasing cost of launching and operating satellites, the number of satellites in orbit has grown dramatically in recent years, and is only set to increase with time. In particular, the availability of commodity components and rideshared launches have made it especially easy to launch CubeSats – their small form factor and well-defined dimensions enable many satellites to fit on a single launch vehicle. As this trend continues, there will be a huge number of satellites around Earth in a wide range of orbits, as well as around other bodies in our solar system like the Moon and Mars, or in deeper space [12, 33]. This raises some interesting questions surrounding communication: how do we ensure communication with these satellites is possible (to relay data for telemetry, control, and normal operation), and how do we ensure it is secured via authentication or authenticated encryption?

As discussed in Section 2.1, new architectures are emerging to support large-scale satellite communication – in particular, relay networks and federation. The first of these enables communication across a highly distributed network by relaying traffic over a small number of high-bandwidth links. This has already seen significant use by

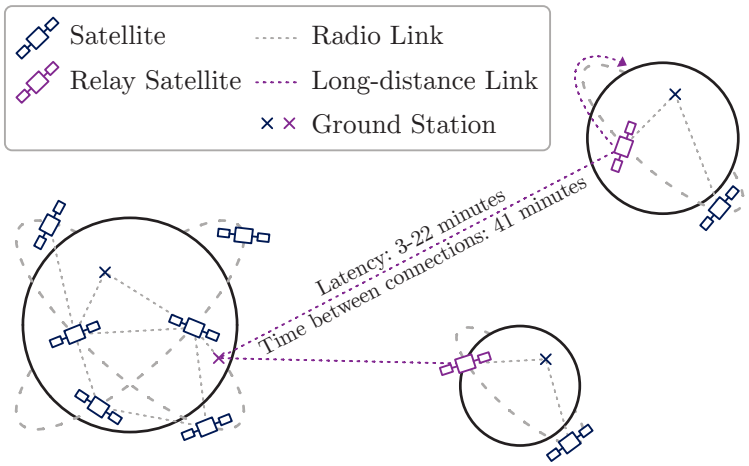


Figure 5.1: Example of an interplanetary network of satellites and the communication between them. There is a federated constellation of satellites around Earth (left), and relays connect the network to the Moon/Mars (right). These links are high latency and, due to satellite and planet movement, are not always available.

organisations like NASA and ESA using their respective relay satellite systems [37, 38], with upcoming proposals like NASA's LunaNet and ESA's Moonlight set to provide additional relays around the moon for lunar communication [12–14]. Federated Satellite Systems (FSSs) provide additional communication redundancy by enabling satellites to communicate with one another over Inter-Satellite Links (ISLs) beyond the constraints of traditional operator boundaries. There has been some recent interest in this concept, with works showing that such a network is feasible and useful, and beginning to consider protocols for negotiating federations on-the-fly to share bandwidth or payloads [196–200]. Supporting both of these is the increased deployment of terrestrial communication systems, with projects like Amazon's AWS Ground Station providing centralised control of satellites with broad coverage, using a network of ground-level antennas [35].

This results in a network topology that has a number of sporadic, high-latency links; this is illustrated in Figure 5.1. The network naturally partitions itself based on these long-distance links, with good connectivity within each partition. This type of network is considered to be a Delay-Tolerant Network (DTN), an architecture which has been explored extensively in related works. In DTNs, key management is considered to be an unsolved problem and is an active area of research [88].

The majority of research in this area focuses on the unpredictable movement and connectivity of the network, and makes use of techniques like gossip protocols and web of trust in order to provide probabilistic assurances of success [89–93].

However, although connectivity in satellite networks is intermittent, the entire network topology is predictable, including the movement of satellites and availability of links.[1] This differentiates satellite networks from other types of DTNs; as a result the underlying assumptions of key management in DTNs break down. Data does not need to be routed probabilistically and can instead be optimally routed to its destination, even if the network topology changes in-transit. In contrast to current consensus, we thus hypothesise that we can apply existing terrestrial PKI concepts.

5.1.1 Contributions

In this chapter, we outline the goals and requirements for deploying and operating PKI in large satellite networks, with a particular focus on interplanetary networks with long-distance, intermittent relays. We construct a standardised set of experiments for comparing different key management systems to one another for a given network topology, looking in particular at the performance of connection establishment procedures, and the speed at which key revocations and updates can occur so that the damage caused by a compromised key can be minimised. We evaluate these through the use of simulations, building the Deep Space Network Simulator (DSNS): an extensible system for the realistic simulation of large networks of satellites. Through this analysis we demonstrate, for the first time, that commonly used terrestrial PKI protocols can be used effectively in satellite networks. In doing so, rather than implementing novel protocols designed for DTNs, we enable interplanetary networks to maintain compatibility with terrestrial networks. We also establish configurations which optimise performance and security, and propose novel variations on these protocols which provide more effective revocations to protect remote segments more quickly. Finally, we discuss future directions of

[1] Of course, this does not account for random losses, or link/node loss caused by attacks; we discuss in Section 5.2 how an interplanetary network might recover in these cases without losing all the benefits of network predictability.

research, looking in particular at how interplanetary networks can recover from random or targeted losses of nodes or links in the network, without reverting to purely probabilistic methods, and drawing parallels to related problems in routing and critical message delivery.

5.2 Related Work

Due to its wide range of applications, the issue of security in DTNs and satellite networks has been well-explored, both in scientific literature and in internet drafts and RFCs. We look first at the drafts put forward by the security community, before moving on to relevant papers and academic works.

5.2.1 Internet Drafts

The Bundle Protocol (BP) is a standard protocol, designed to enable communication in satellite networks and other DTNs [201]. Unlike IP's best-effort delivery, BP is designed to be tolerant of intermittent connectivity, limited bandwidth, large delays, and high error rates. This is achieved by grouping blocks of data into bundles, ensuring each bundle contains enough information for an application to make meaningful progress. These bundles are routed in a store-and-forward manner, and can be sent across multiple paths if necessary (particularly if the bundle is marked as critical), enabling service guarantees even in highly distributed networks. The Consultative Committee for Space Data Systems (CCSDS) also provide standards for convergence layer adaptors, providing a bridge between BP and other protocols, including standard protocols used in terrestrial networks like TCP and UDP [15].

The most recent proposed standards for BP include the "BPSec" security extensions [202]. These provide secure communication through BP by adding two new blocks to the protocol: one for integrity, and one for confidentiality (i.e. encrypted communication). Previous versions of the BP standard have been accepted and used in a wide range of missions, so we expect that this is likely to be the standard for secure communication in future space DTNs. Notably, this standard does not recommend a specific method for key management, assuming

it to be handled separately as a part of network management [202]. In contrast, in our experiments we focus on the key management mechanisms, assuming the underlying cryptosystem to be implemented separately.

Other drafts assess the problem of key management in DTNs and space networks. The authors of [203] recognise key management to be an important open issue in DTNs – this is expanded on in [88], which argues that the long delays and expected disruption in DTNs makes traditional PKI methods unsuitable due to their requirement of short-turnaround communication with Certificate Authorities (CAs).

In [204], a basic set of high-level requirements is given for key management in DTNs; we draw particular attention to their note that systems must account for nodes with highly restricted connectivity, computation, or storage capacity. The authors of [205] also set out requirements for PKI in DTNs, noting that the system must not rely on time-constrained queries to a central authority or have a single point of failure, and that revocations must be delay tolerant. They propose their own protocol using ephemeral secret keys included with each bundle, signed by key authority nodes which issue revocations as periodic multicast bulletins.

In [206], the CCSDS give a high-level overview of many potential key management concepts in space DTNs. They note that key management in large space networks is significantly more complex than in current space networks, but do not suggest any particular concept as the optimal solution. There is also an experimental CCSDS specification designing a cloud-based CA for satellite systems, to be used as a common route of trust particularly in shared systems [207]. Results from our simulations will help to inform this and other projects – this is discussed further in Section 5.8.

In [208], the authors lay out five potential design patterns for PKI in DTNs, which we use as a basis for some of our simulations. Of particular interest are the "request-response" pattern, matching behaviour seen in OCSP and OCSP Stapling, and "publish with proxy subscribe", in which certificates are sent via a CA which validates the certificate en route. This eliminates the need for a handshake with the authority, and allows recipients to subscribe to updates and revocations of the

certificate, enabling updates to be sent without flooding the network. We also use the "publish-subscribe" (OCSP, but the recipient is informed about updates to the certificate) and "blacklist broadcast" (broadcast CRLs) patterns from this work.

5.2.2 Academic Works

The authors of [92] provide a good overview of existing works looking at key management in DTNs. Some of these focus on space networks or are network-agnostic, but the majority look at less relevant applications of DTNs such as rural networks.

A number of works focus on security initialisation or authenticated key exchange – that is, initiating a secure channel where one did not previously exist by exchanging or establishing keys [209]. We consider these to be out of scope of this work, and focus on communication and updates over an already established network.

In [210] the authors recognise that terrestrial protocols do not perform well in DTNs and smaller satellite networks, due to their reliance on constant end-to-end connectivity between points in the network, low link delays, and low error rates. They also acknowledge that any key management system will need to account for the types of nodes and heterogeneity in the network. The DSNS simulator described later in this chapter can be used to assist research in this area by assessing the differences between satellite networks, and setting out guidelines that apply to satellite systems in general.

In [211] the authors focus on real-world assessments of various cryptosystems on embedded hardware. Additionally, they discuss the requirements of security protocols in federated satellite systems. They propose a high-level PKI-based protocol to guarantee confidentiality, integrity, and authenticity, and rely on a centralised authority to verify and distribute keys.

Some works such as [212] adapt PKI for use in satellite networks – their system is very similar to OCSP stapling systems, with nodes requesting an authentication pass from a CA ahead of communication. On the other hand, [213] argues that PKI is not suitable for use in DTNs, instead suggesting the use of Hierarchical Identity

Based Cryptography (HIBC), in which keys are derived from public information about nodes, and a hierarchy of authorities manages keys. This is argued to be better than traditional PKI, as the public parameters do not need to be verified as frequently [214], at the cost of requiring distribution of additional secrets out of channel. We argue that this is not necessary for satellite networks, as each network segment is sufficiently connected that it is always possible to query a trusted authority, provided there are sufficiently many distributed throughout the network. We demonstrate this using simulations in Section 5.7.

In [93] a decentralised key management system is proposed in which chains of certificates are passed around the network, with nodes added to the chain when another node trusts it. This has the benefit of being fully decentralised, but has no mechanisms for key revocation or centralised/federated management.

Finally, some works extend existing PKI concepts to make them more suitable for use in DTNs. The authors of [215] and [216] build upon traditional CRLs, reducing message size and storage by sending "delta CRLs" containing incremental updates and by using hash tables. In [217], the authors also extend PKI concepts, building a consensus protocol to remove the need for a single root CA, and testing a number of revocation methods including OCSP Stapling and Bloom filters. The authors of [218] recognise that a single centralised CA is not enough for PKI in DTNs, and propose the use of distributed CAs to ensure reachability. In [219] this concept is extended to "adaptive CAs", in which any node can become a CA, and decisions are made dynamically.

5.2.3 Simulators

Some of these works are evaluated using simulators such as ns-3 [97], the ONE Simulator [100], and OMNeT++ [101] – these are compared and discussed in more detail in Section 2.4.1. We also discuss the limitations of each simulator; in particular, in Table 2.2. From this we see that there are limitations to each of the existing simulation offerings – some struggle to scale to large numbers of nodes, do not simulate mobility, or do not support dynamic links, limiting the range

of network configurations that can be tested. Others that do have some of these features are unnecessarily bloated, simulating the entire network stack when a much smaller number of layers is sufficient for the vast majority of simulations, or do not support the protocols we are interested in testing in the first place.

In addition to network simulators specialised for satellites, some works look at DTNs in a more general sense. In evaluations of DTN protocols, it is standard to simulate nodes moving randomly around a space, connecting to nearby nodes (for example, [220]) – whilst this is sufficient for some terrestrial DTNs, it does not capture the interesting mobility properties of a satellite network. We have designed DSNS such that constellation topology and connectivity can be simulated as realistically as possible, enabling us to perform a convincing real-world evaluation of key management in satellite systems, mirroring real-world deployments and proposals for future systems.

5.3 Threat Model

In this chapter we focus on mitigating threats involving compromised keys in larger satellite networks, rather than the physical layer threats addressed in earlier chapters. Our threat model therefore differs quite substantially from the generalised model in Section 2.3.7, although the overall motivation remains consistent. The goal of an attacker in this context is to gain access to the key of a legitimate user or device on the network, enabling them to craft messages as if they were the legitimate user. This could be achieved in a number of ways: for instance, compromising a satellite (e.g. by compromising telecommand authentication, taking control of a satellite via a compromised ground segment, or via a malicious hosted payload), intercepting improperly encrypted messages, compromising the supply chain, etc. Depending on the target, this could have a range of impacts, including producing forged measurement data, impersonating users in communication, or forging telecommand packets to damage the device itself.

When an attack has been discovered, the compromised key must be revoked. The long distances between nodes in an interplanetary network and inherent speed-of-light limit means the goal is not to prevent the attacker entirely, but instead to minimise the extent of the damage from the moment the attack is identified. This is achieved by propagating the revocation throughout the network as efficiently as possible, and constructing infrastructure in such a way as to minimise the effective reach of a key once this information has propagated.

The attacker's primary capability is sending messages on the network using their compromised key. This is a reasonable assumption to make as it lines up with common usage patterns, and enables us to generalise between attackers located in the ground segment and the space segment. In Section 5.8.1 we discuss the potential impact of jamming attacks, examining ways in which PKI and other critical message delivery systems might be affected by targeted link losses caused by jamming.

5.4 Framework Design

In this section we establish the goals and requirements for key management in satellite networks, constructing from them a set of standardised experiments which can be used to compare protocols against one another. Rather than being limited to one network topology or configuration, these are instead designed to be high-level tests that can be used on any network to compare different protocols side by side. This allows network operators to test protocols in the context of their own network configuration, enabling well-informed decision making about which configuration to choose, balancing performance against security. In Section 5.6 we design experiments using standard terrestrial PKI protocols in interplanetary satellite networks, and evaluate improvements tailored to the unique topology of these networks.

We state the following goals for PKI in satellite networks:

- **Low latency:** The latency of messages should be as close as possible to the shortest path between the two nodes. For some PKI systems this will not be a concern, but configurations which require messages to be sent via an authority

node should be configured to minimise the additional distance traversed by the message.

- **Low establishment overhead:** Establishment overhead T_E is computed to be the time difference between a full connection handshake between two nodes, and the time to send a single message. The long distances traversed by messages between planetary segments, combined with the short duration some relay links may be available, makes minimising this crucial.

- **Fast revocation coverage:** In large satellite networks with many users, it is inevitable that keys will need to be updated and revoked. We consider revocation coverage time $T_R(s)$ to be the time at which all of the nodes in segment s are protected, due to revocation information having propagated sufficiently far (e.g. to the relevant CA). There will be a necessary lower bound to this value, but protocols and network configuration can be tweaked to optimise it.

- **Low attack penetration:** In cases where authentication data has been cached, we also evaluate the attack penetration $P(s)$ to be the proportion of nodes in segment s which accept an attacker's message sent at the same time as a revocation was issued.

We also propose the following secondary goals, which are not required but may be desirable:

- **Distributed:** In order to achieve good performance across long distances, authority should be distributed throughout the network to enable querying without excessive latency. Later on we discuss the trade-off between centralised control and performance.

- **Intercompatible:** We argue that in order for interplanetary connectivity to be effective, protocols will need to work alongside existing internet infrastructure. Protocols may be designed to work seamlessly with existing protocols,

or utilise a translation layer. Protocols should also integrate with BPSec [202], which provides cryptosystems but leaves key management as an open question.

Our focus is on adapting PKI mechanisms to the environment of interplanetary networks, rather than on the underlying cryptosystem itself. While considerations such as storage, computational requirements, and bandwidth usage are crucial in this context, they are inherently tied to the choice of cryptosystem and are therefore outside the scope of this work. By concentrating on the feasibility of applying terrestrial PKI mechanisms in space, we establish a foundation for future research, which can then focus on optimising the cryptosystem and its components to meet the unique demands of interplanetary networks.

5.4.1 Scenarios

We construct two scenarios to evaluate the above goals: connection establishment, and key revocation.

Connection Establishment

In the first scenario one node wants to communicate with another, but must first prove its identity. To do so it sends a signed message with a certificate attached. The recipient validates the certificate via validity information either stapled to the message or requested from a CA.

This adds a delay before communication can occur, which can be quite significant in satellite networks with long-distance links. This is a particular concern when communication occurs over a sporadic link between network segments; any time spent establishing the sender's identity is time in which communication cannot occur, taking away from the total duration of the link's uptime.

In order to eliminate the need for traffic modelling we consider a connection establishment between every pair of communicating nodes in the network, giving a distribution of values for each metric. We focus on the following metrics:

I. Dropped messages (%);

II. Latency (s);

III. Establishment overhead (s);

IV. Total hops.

Key Revocation

In the second scenario we consider a key in use by a malicious actor. The key is revoked, and we aim to minimise the penetration of the attack beyond this point, and ensure the revocation propagates across the network as quickly as possible.

Coverage time $T_R(s)$ (for segment s) is assessed relative to revocation time t_0 by finding the time at which each segment's CA received the revocation message, adjusting to account for query time, and taking the greatest such time within the segment. As a result, for certain segments the coverage time may be negative – if a revocation reaches the CA quickly, but it takes a long time for the attacker's message to reach that same CA (for instance, if there is a long-distance link between them), then nodes under that CA can be protected even from messages initialised before the revocation was issued. Also note that in some cases such as OCSP Stapling, coverage depends on the attacker's location; in these cases, results are computed for every possible attacker location and aggregated by segment. We also assess attack penetration $P(s)$ for cases where certificate validity information is cached, with a fixed attacker location chosen to be in an orbital plane opposite the CA in segment s.

5.5 Experiment Design

In this section we design experiments to assess the viability of terrestrial PKI techniques in interplanetary network topologies, building upon the framework established in Section 5.4. We outline the key management systems tested, including the certificate revocation mechanisms and approach to distribution of authority within the network. Our experiments consider a range of network topologies, focusing on a representative Earth-Moon-Mars constellation alongside standard constellations

and federations of satellites, in order to provide a thorough understanding of the feasibility and performance of PKI techniques in these emerging network environments.

5.5.1 Key Management Systems

In our experiments we evaluate a number of PKI systems based on the existing and proposed mechanisms discussed in Section 5.2. In each case, nodes have a public/private keypair which is used to sign and encrypt messages. These keypairs are accompanied by a certificate – that is, a chain of signatures tracing back to a Certificate Authority (CA) vouching for its legitimacy. We specifically consider the following systems to verify a certificate's validity, ensuring it has not been revoked:

- **Certificate Revocation Lists (CRLs):** CAs each maintain a list of revoked certificates, which are sent to nodes on request. Any certificates on this list are rejected, and others are accepted provided there is a trusted CA in the certificate chain. CRLs are cached, with a new CRL requested on the next message received after expiry. To reduce size, CRLs can be built using hash tables [216] or using delta CRLs [215].
- **CRL Broadcast:** Similar to the above, but CAs broadcast CRLs at regular intervals, instead of on request.
- **Online Certificate Status Protocol (OCSP):** On receipt of a signed message, nodes request the certificate's validity information from their CA. The response is cached to reduce the overhead on subsequent messages.
- **OCSP Stapling:** Similar to the above, but the node sending the message requests its CA for a signed message with validity information. This is sent alongside the original message, reducing the burden on the recipient.
- **OCSP with Validators:** Proposed in [208], messages are instead sent via a CA, which staples the certificate validity message to the original message in-transit. This removes the need for a full handshake with the CA, and can significantly reduce latency if the CA is en route to the message destination.

This shares some similarities with the proposed "Revocation in the Middle" technique, using middleboxes to provide revocation information [221].

- **OCSP Hybrid:** We propose a new system, combining and extending the above OCSP concepts. We note that in many cases, in-transit OCSP validation results in increased congestion due to all traffic being sent via the CA. However, when traffic is travelling via a small set of relays anyway (for example, between Earth and Mars), the level of congestion is unchanged, in addition to improving latency by removing unnecessary handshakes. We therefore propose a hybrid system in which standard OCSP Stapling is used for traffic within a segment, but any cross-segment traffic uses in-transit validation, with CAs located at the relays between network segments.

We describe how these protocols are implemented in DSNS in Section 5.6.2, and give sequence diagrams for each protocol in Appendix C.

We evaluate each validity mechanism under a "centralised" model, with a single terrestrial CA, and a "distributed" model, in which multiple CAs are present throughout the network. In the latter case, we place one CA in each network segment (i.e. in each layer of satellites in the network, or group of ground systems), designated as sub-CAs of a single terrestrial root CA. Authority is distributed such that any CA can provide validity information for any other certificate, and nodes attempt to use the CA closest to them. This has the potential to significantly reduce the latency overhead of validating cross-segment messages, although the threat of attackers using revoked certificates from other segments is also increased. We demonstrate in Section 5.7 that it is possible to configure the system to minimise the threat of these attacks.

When implementing the revocation scenario, we consider two cases: one in which nodes have already cached the compromised certificate as valid, and one where they have not. This enables us to see the difference in effect between the situation where attacker messages are accepted by default, and one in which up-to-date revocation information must be retrieved. It also enables us to consider an alternative PKI

Table 5.1: The number of nodes in each of the network topologies used in our experiments. Relays and multiple segments are separated by "+".

	Earth		Moon		Mars		Relay
Network	Ground	Space	Ground	Space	Ground	Space	
Earth/Moon/Mars	256+3	66	12	8	12	66	2
Earth/Mars	256+3	66			12	66	1
Earth/Moon	256+3	66	12	8			1
Iridium	256	66					
Starlink	256	1584					
CubeSat	256	121					
LEO/LEO	256	66+1584					
LEO/MEO	256	66+13					
LEO/GEO	256	66+3					
LEO/MEO/GEO	256	66+13+3					

configuration; in OCSP we have the option of subscribing nodes to updates, with CAs remembering which nodes have been sent certificate validity information and passing on certificate updates to the relevant nodes. This allows revocations to take place more quickly without requiring broadcasts across the whole network.[2]

Finally, we propose and evaluate a new technique: using interplanetary relay nodes as a firewall. We note that in interplanetary networks all cross-segment traffic must travel via relays, providing an excellent opportunity to remove messages from revoked keys before they reach their destination. This can significantly reduce load on the relay links by dropping attackers' messages much sooner, and may also bring earlier revocation coverage, reducing the number of nodes reachable by an attacker with a revoked key.

5.5.2 Network Topologies

We want to ensure our experiments remain as true to real-world networks as possible, so we base our network topologies on currently deployed constellations, proposed future systems, and plausible extensions of existing systems. Table 5.1 gives some information about each scenario.

[2]This option can be used for each OCSP configuration except OCSP Stapling – the CA does not know where the certificate status will be sent, so it cannot subscribe the recipient to updates.

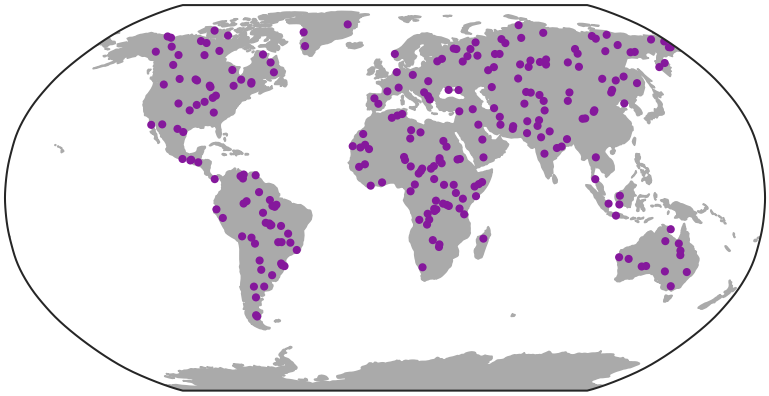


Figure 5.2: 256 ground stations are placed at random points on the Earth's land surface to produce a simulated network with global communication. Locations are fixed across all experiments for the sake of consistency.

Earth/Moon/Mars

For the majority of this work, we focus on a representative network with communication between Earth, the Moon, and Mars. We have designed this network to match existing proposals for future interplanetary networks, such as NASA's "LunaNet" proposal, in which relays enable communication between nodes around Earth and the Moon [12].

The network contains 66 satellites arranged in a Walker constellation matching the orbits of the Iridium satellites, and 256 randomly placed ground stations, the positions of which are shown in Figure 5.2.[3] For the sake of this experiment, we assume no terrestrial network exists between the ground stations; they must use the satellites to communicate with one another.

Around Mars we place a further 66 satellites and 12 ground stations. We also put 8 satellites and 12 ground stations on the Moon.

To relay data between segments, we place ground stations on Earth matching NASA's Deep Space Network (DSN). This has 3 ground stations spaced out around the Earth to provide a good view in all directions – while this exact network may not be used for all future communication, we can be confident that a similar system

[3]Each ground station has a minimum elevation angle of 8.2°, matching Iridium specifications [222].

would be used.[4] We place a single relay satellite around the Moon and Mars each, with orbital periods of approximately 3.3 hours and 2.5 hours respectively.[5] These short orbits were picked to ensure each experiment includes times in which the relay is disconnected. The relay satellites can connect to Earth's DSN nodes and to satellites in their segment. Although it would be possible to enable near-constant communication by adding more relay satellites, in this work we are particularly interested in ensuring interplanetary communication systems do not break down when connectivity between segments is interrupted. Furthermore, even a constellation with good relay coverage will face some unavoidable interruptions – for instance, every 26 months the line of sight between Earth and Mars is blocked for 2 weeks by the Sun [224].

We evaluate this network topology under various starting conditions to ensure the results are complete. To account for the dynamic nature of the relay links, we conduct experiments with four different starting configurations, each with a different start time. This allows us to capture all possible states of the relay links at the beginning of the experiment. Furthermore, when using decentralised PKI, we repeat the revocation experiments for each segment, since the behaviour and impact of revocation vary depending on the segment from which it originates. This enables us to cover all starting cases, gaining a comprehensive understanding of the system's performance.

Earth Constellations

We evaluate single-constellation networks as a control, to compare the properties of well-connected networks against interplanetary networks. We take two Low Earth Orbit (LEO) constellations at different scales: the Iridium constellation with 66 satellites, and Starlink's first orbital shell, with 1584 satellites [225]. Satellites are again connected to 256 ground stations.[6]

[4]The location of each DSN ground station and the minimum elevation of the antennas (10.5°) is taken from [223]. We also assume the DSN stations communicate with Earth's satellites, acting as a relay.

[5]The Moon relay has an orbital period of 11 941 s and the Mars relay has an orbital period of 8829 s.

[6]Starlink antennas have a minimum elevation of 25° [225].

Federation of CubeSats

We next design a federated satellite network based on 121 CubeSats in orbit as of 2024, connected to 256 ground stations.[7] Each CubeSat is equipped with an ISL with a maximum range of 2500 km, using technologies based on Opportunistic Space-Division Multiple Access (OSDMA) and beamforming to connect to multiple neighbours at once [226]. This forms a dynamic mesh in which nodes are not always reachable, providing an interesting challenge for PKI. Although the majority of deployed CubeSats are not yet equipped with ISLs, this gives a good idea of what future networks might look like. To enable such a large number of independently operated satellites to communicate securely, it is crucial that keys and certificates are properly handled.

Federation of Constellations

Finally, we consider networks composed of existing constellations talking to one another. Unlike the CubeSat constellation discussed above, composed almost exclusively of satellites in LEO, this enables us to form a multi-layer satellite network, with satellites in Low Earth Orbit (LEO), Medium Earth Orbit (MEO), and/or Geosynchronous Equatorial Orbit (GEO), to increase coverage, reliability, and performance [227]. This architecture is rapidly approaching, with networks like Starlink planning to add additional layers of satellites to provide connectivity at higher latitudes [225]. Although MEO and GEO constellations tend to provide good global coverage by themselves, using LEO satellites can reduce power requirements for ground terminals – indeed, LEO has been recently explored as an option to expand cellular and 5G connectivity [228–230].

We consider the following case studies:

- **LEO/LEO:** Two constellations in LEO communicating with one another to provide shorter traffic paths and better coverage. In this work we consider the Iridium and Starlink constellations talking to one another.

[7] Positions are derived from Two-Line Element (TLE) sets provided by celestrak.org on 2024-03-26 and propagated from 2024-03-26T00:00:00Z. Up-to-date TLEs may be used, but results will not be identical.

- **LEO/MEO:** A constellation in MEO can provide much more consistent coverage than LEO, with a small number of high-bandwidth satellites providing wide area coverage. The LEO satellites pick up the slack by providing coverage of areas further from the equator and lower latency connections, at the cost of a greater hop count. We consider Iridium working alongside the O3b mPOWER MEO constellation (operated by SES, in mid-deployment as of 2024) consisting of 13 satellites at an altitude of 8000 km [231, 232].
- **LEO/GEO:** GEO satellites provide even wider coverage than MEO, at the cost of much higher latency. We consider Iridium connecting to the Viasat-3 GEO satellites, consisting of 3 satellites spaced out to provide global coverage.
- **LEO/MEO/GEO:** Iridium, mPOWER, and Viasat-3 all working together.

In each of these cases the satellites can connect to the ground directly, and additional inter-layer links provide connectivity between each layer.

5.5.3 Routing

Routing in DTNs and satellite networks is a difficult problem, and has been explored extensively in recent literature [233–238]. Its particular difficulties stem from constantly changing connectivity and many-hop paths, requiring routing that can react to changes more rapidly than update messages can traverse the network.

The topology of many interplanetary networks makes it easier to route traffic – the use of only a small number of relays between segments means traffic only needs to be routed within each segment. Messages bound for other planets simply need to route to the relay, which can handle routing for the remainder of the journey.

In our experiments we do not employ any particular routing algorithm, considering it to be beyond the scope of this topic. To prevent results from being affected by a particular choice of routing algorithm or strategy, we assume all nodes have access to optimal routing information at all times, including looking ahead to the future state of the network in store-and-forward routing. This provides similar results to source routing, in which a message's route is computed at the source

then simply forwarded along the computed path. This is trivial to implement for best-effort routing (at the cost of computational power), but additional complexity is introduced by the store-and-forward strategy employed by our networks: the routing must look forward to the future state of the network to determine whether a message needs to be stored. This can be implemented by precomputing the state a fixed number of timesteps ahead.

The design of DSNS allows the routing algorithm to be easily changed, providing the possibility for future work to evaluate the effectiveness of proposed future routing algorithms across a wide range of satellite network topologies. We discuss this further in Section 5.8.1.

5.5.4 Traffic Modelling

As discussed in Section 5.4, we do not rely upon randomised traffic modelling, instead considering pairwise communication between every pair of non-relay nodes in the network. This provides us with a distribution for each performance metric, which can be further partitioned into network segments, enabling a comprehensive analysis of the network's performance under each PKI configuration.

5.6 The Deep Space Network Simulator (DSNS)

To run these experiments, a simulator is required. Such a simulator must be able to handle arbitrary satellite constellations and interplanetary networks, scale to large numbers of nodes, and be easily extensible to support new protocols and configurations. We found in Section 5.2 that existing simulators were not suitable due to their computational inefficiency, lack of support for the network configurations tested, and unnecessary complexity for our experimental setup.

We therefore present the Deep Space Network Simulator (DSNS): a new event-based simulator for satellite networks, supporting the required features for our experimental framework. Unlike existing simulators, DSNS can scale to large numbers of nodes, with computational requirements governed only by the number of events generated by the simulation. It supports arbitrary orbital configurations

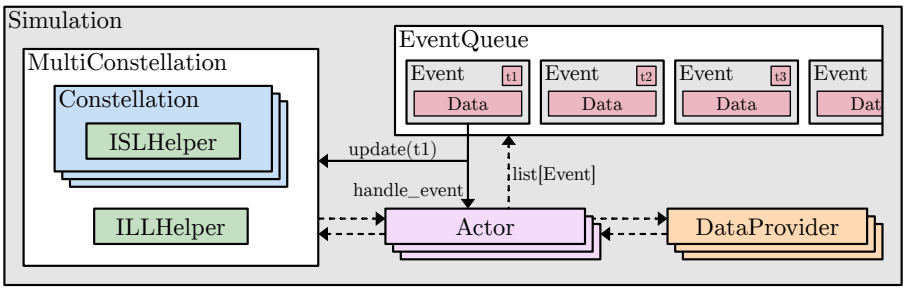


Figure 5.3: Overall structure of DSNS and its high-level operation.

and network topologies, and provides a simple interface by which new protocols and behaviour can be implemented.

In this section we describe the broad design and structure of DSNS, the functionality added in order to support the experiments described in the previous section, and how it can be extended to support new functionality.

5.6.1 Architecture

Figure 5.3 shows the overall architecture of DSNS. The simulation is highly modular, underpinned by a simple event-based simulation, and components can be switched out to alter behaviour, or entirely new components can be added for additional functionality. Mobility and connectivity are handled by the `MultiConstellation` class, which simulates any number of constellations, the Inter-Satellite Links (ISLs) within constellations, and the Inter-Layer Links (ILLs) between constellations. Constellations can be created from fixed points, Walker constellation parameters, or by importing Two Line Element (TLE) sets for in-orbit satellites. Each of these inherits from the same `Constellation` class, defining the positions of satellites in the constellation or segment over time. Each constellation also has an `OrbitalCenter`, defined to be the parent around which satellites orbit, enabling complex movement and simulation of satellites around many different planets or other bodies, without any loss of precision when communicating over short distances.

Inter-satellite links can be fixed (for ground systems or Walker constellations) or dynamic (connecting to satellites within view), by using one of the pre-built `ISLHelper` classes or defining the links manually. The `ILLHelper` works almost

identically, but defines instead the links between different constellations or planets. Each time the simulation time is updated, the positions of satellites and connectivity of links are updated to reflect the new state.

All other functionality is handled by an event-based simulator – events are stored in a priority queue and processed by `Actor`s in the simulation, which in turn produce additional events to add to the queue. As each event is removed from the queue and processed, mobility and connectivity of the constellation is updated to reflect the state at the time of the event, and the event is passed to all actors, which pattern match on events to decide how to handle them. Each actor may process the event, optionally communicating with the mobility and connectivity model or `DataProvider`s for additional features such as routing, before optionally returning a list of new events, which are added to the queue. This enables complex functionality like message routing and broadcasts in an entirely modular fashion, and makes it easy to add new features without requiring a deep understanding of the rest of the simulation.

For example, message delivery is handled by a `MessageRoutingActor`, which can deliver messages between any pair of nodes between the network, in addition to supporting broadcast messages. It does this by taking any `MessageCreatedEvent`s (indicating a message has been newly created), and creating an initial `MessageSentEvent` across the first hop of the message's path. Each time a message is received (`MessageReceivedEvent`), a subsequent `MessageSentEvent` is created to send it on its next hop, before finally generating a `MessageDeliveredEvent` at the destination. If store-and-forward message routing is used, messages can also be held at a node for later delivery, and sent when the link becomes available (`LinkUpEvent`). If instead best-effort routing is used, messages which cannot be forwarded over the desired route, or for which no route exists, are dropped.

The routing for this actor is handled by a `RoutingDataProvider`, an instance of the more generic `DataProvider` (any actor that can provide information or functionality to other actors). Each time the simulation is updated, the routing system builds a connectivity graph for the network in its current state, enabling

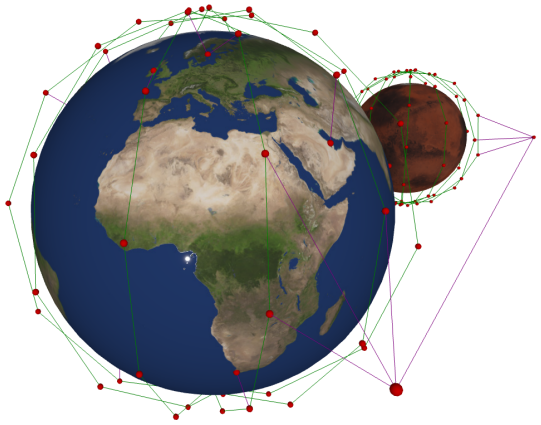


Figure 5.4: An example visualisation of an interplanetary network in DSNS. Earth satellites are connected to Mars satellites via a relay link. Nodes (satellites and ground systems) are in red, inter-satellite links in green, and inter-layer and interplanetary links in magenta.

it to compute optimal next-hop routing for any message that needs to be routed. We also provide a routing system with lookahead for store-and-forward message delivery, which computes routes by looking ahead at the future state of the network. This allows it to find routes involving links which are not yet up – in this case, messages travel as far as they can, and are then stored until the link goes up. Each of these mechanisms assume each node knows the state of the network and can thus perform perfect routing; this provides similar outcomes to source routing, in which the source node computes the full route of a message on creation. We discuss in Section 5.8.1 how future work might extend this system to provide more realistic implementations of routing protocols, and what performant routing in interplanetary networks might look like.

We also provide a visualiser using the "pyrender" library, to make it easier to understand how satellite network topologies evolve over time, and to construct new constellations. An example visualisation can be seen in Figure 5.4.

Due to the extensibility of DSNS, it will be useful for future research in new space systems, particularly in evaluating the performance of other protocols. This is discussed further in Section 5.8.1.

5.6.2 PKI Functionality

On top of the base DSNS simulation framework, we implement new functionality specifically for the PKI protocols described in Section 5.5. We achieve this by adding two new actors: the `CertificateActor` implements OCSP-like behaviour, and the `CertificateCRLActor` implements CRL-like behaviour; both of these inherit from a `GenericCertificateActor`. Thanks to the architecture of DSNS, each of these modules does not need to worry about routing or any lower-level functionality, since this is handled by other actors in the simulation.

Although it would be possible to implement the protocols in full detail down to the byte, we are primarily focused on the behaviour of the PKI systems at the message level. We therefore focus instead on implementing the message-level functionality of the protocols.

At its core, the new functionality is underpinned by two new classes: the `Certificate` and the `SignedMessage`. At the start of the simulation, a hierarchy of certificates is created, with a root CA defined and all other CAs (and regular communicating nodes) deriving their certificate from some CA. When an unsigned message is created, the actor decides how to respond, typically creating a `SignedMessage` encapsulating the original message, and potentially generating additional events (for example, to request the certificate status or CRL from the CA). Additional procedures are also carried out on receipt of a signed message (unless the certificate status has already been cached by the node), using a `CertificateQueryMessage` to query the CA about the certificate's status in OCSP, and a `CRLRequestMessage` to request a CRL. A `ValidatorQueryMessage` is used to indicate that the CA should staple the certificate status to the message in the OCSP Validator configuration. Finally, the actor also implements functionality to handle certificate updates and revocations, with update messages propagated to all CAs, all nodes, or nodes which are known to have the revoked certificate's status cached, depending on implementation.

Sequence diagrams for each of these protocols are given in Appendix C.

5.6.3 Extending DSNS

Due to its modular design, adding new functionality to DSNS is a reasonably straightforward process. We demonstrate this by outlining how a user might add support for a new network layer above or below those already simulated, a new routing strategy, and new constellations within the mobility and connectivity model.

Since the behaviour of each component is defined by Python functions, rather than a domain-specific language, customisation options are almost unlimited: as long as the desired functionality can be expressed in code, it can be added.

Higher Layer Protocols

It is easy to add new protocols at higher levels by extending the `BaseMessage` to add additional fields, encapsulating the original message as a field within the new message. We can already see this in practice in the functionality added by our PKI experiments; the `SignedMessage` adds additional information to the base message class, enabling a `KeyManagementActor` to both add this information to a message and to act on it.

However, users need to take care that layers are appropriately ordered by ensuring each layer correctly pattern matches on messages from the layer above it, and that a `MessageCreatedEvent` is only generated for the lowest layer event. For more complex simulations with multiple layers, a layer manager actor might be used in order to make sure layers are correctly ordered and handle events between them.

Physical Layer

Of course, alongside higher layer functionality DSNS also supports implementing additional features at lower layers. This might involve adding additional parameters such as message size, priority, or physical layer modelling for message loss. To modify the message structure, `BaseMessage` can be extended once again to add the new fields. Following this, `MessageRoutingActor` could be extended to modify physical layer delivery mechanisms – for example, the `handle_message_sent_event` function can be modified to support bandwidth limiting or message loss modelling.

Routing Strategies

At first glance, adding a new routing strategy appears quite simple, requiring that users merely extend `RoutingDataProvider` with the new strategy. However, this obscures a fair amount of complexity – the current routing strategies assume perfect knowledge about the state of the network, and do not involve any amount of message passing. In order to implement a routing protocol that involves an exchange of information between nodes, functionality must be added to the data provider for passing data between nodes. This cannot use the regular message delivery system, as this itself relies upon the routing system, but it can use the same underlying message structure.

The routing system could also be extended to support new functionality; for instance, if the data provider returns more than one next hop, the message routing system could be configured to deliver messages along multiple paths to increase the chance of successful delivery.

Mobility and Connectivity

Finally, we consider extensions to the mobility and connectivity model used by DSNS. These are the easiest parts of the simulator to modify, as they are unaffected by the rest of the simulation. To add a new type of orbit, a user simply extends the `Constellation` class to provide the desired functionality, defining `update_positions` to correctly set their positions at each timestep. Similarly, `ISLHelper` can be extended to provide new connectivity models, with `get_isls` defining how satellites in a constellation ought to be connected at any given point in time. Finally, `ILLHelper` (and corresponding function `get_ills`) defines the connections between layers and planetary segments.

Once these have been defined, the simulator handles all the low-level functionality, solving for routes and generating appropriate link up/down events as links become available and unavailable.

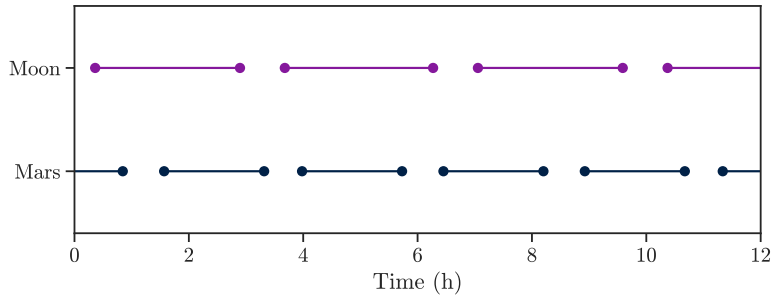


Figure 5.5: State of the relays from Earth to the Moon and Mars over time. Gaps indicate that the link is not available.

Summary

As we have seen, DSNS supports a wide range of useful functionality, and is easily extended to add even more new features. By making this tool freely and openly available, we are confident that it will become a highly useful research tool in the future, enabling a wide range of interesting projects in this rapidly growing field.

5.7 Results

Next we use DSNS to build and run the simulations and experiments designed in Section 5.5, and analyse the results.

5.7.1 Connectivity

Before looking at properties of the PKI systems, we must first understand the network topologies themselves. We are particularly interested in how the partitions of the network change over time: connectivity between interplanetary segments is possible if and only if relays are available. This is visualised in Figure 5.5: when relays are up, the system joins into a single unbroken network (although there is still high latency across the relays), and as the relays periodically pass behind the planet and become unavailable, it is split into multiple parts. This can also be seen in Figure 5.6, measuring the time taken in the Earth/Moon/Mars interplanetary network for a message sent from a given segment to reach a ground station in a different segment. We see some interesting properties here, with spikes in latency between planets as the relay links are periodically cut off from the rest of the

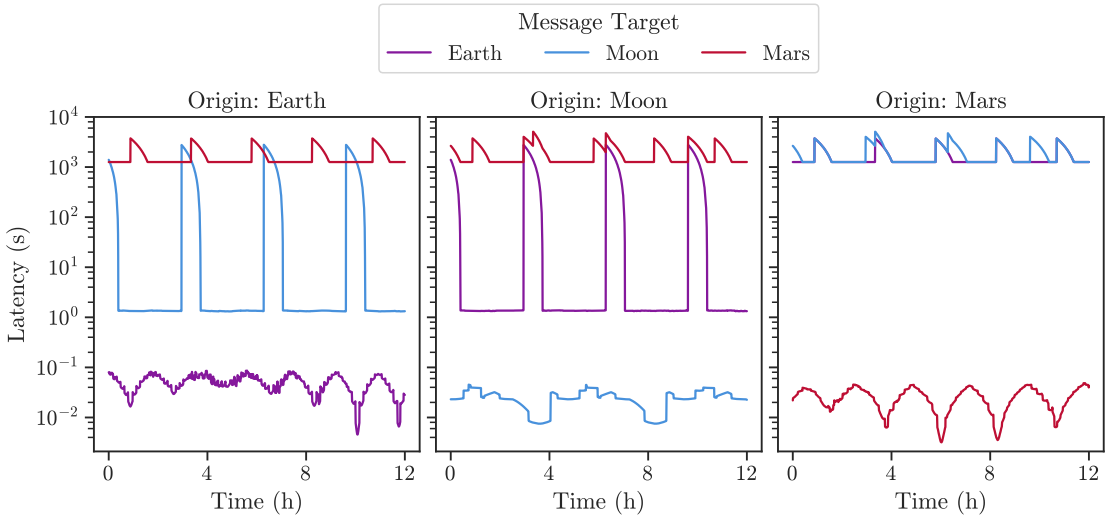


Figure 5.6: Latency over time for an Earth/Moon/Mars interplanetary network, passing messages between nodes in each segment. Latency is plotted on a log-scale.

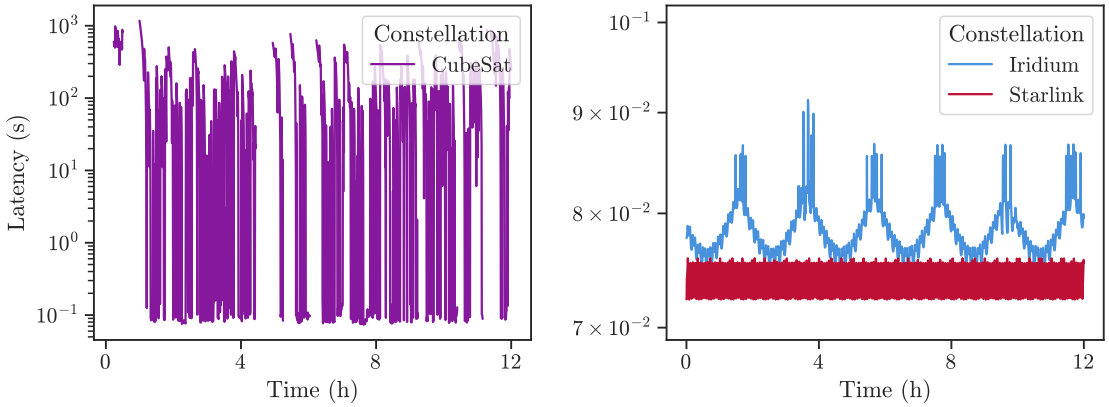


Figure 5.7: Latency of messages between ground stations on opposite sides of the globe for non-federated terrestrial networks.

network – this reinforces the importance of minimising the number of unnecessary interplanetary messages in a connection handshake, as each message sent between planetary segments incurs a significant penalty (in this case, over 1000 s). The latency of connections between segments is relatively consistent (as any small variations are dominated by the long-distance link), whereas connections within a given segment are low, but subject to significant variation as the satellites progress in their orbits.

Figure 5.7 shows how latency between two ground stations changes over time for the Iridium and Starlink constellations, and the federation of CubeSats introduced earlier. We can see that the latency for Iridium and Starlink is a highly predictable

Table 5.2: Cost of establishing a new connection under each constellation topology. Standard OCSP is used in each case, with a single centralised CA.

	Dropped (%)	Latency (s)				Establishment Overhead (s)			Hops			
Constellation		Min	Max	Mean	Std	Max	Mean	Std	Min	Max	Mean	Std
CubeSat	32.97	130.0	599.0	162.7	79.34	592.0	101.3	83.89	2	79	29.44	14.72
Iridium	0.000	0.009542	1.201	0.1483	0.09414	0.1702	0.1001	0.03613	2	28	15.93	4.000
Starlink	24.16	0.004351	229.1	4.148	22.19	1.115	0.09849	0.1181	2	179	54.98	30.26
LEO/LEO	0.000	0.004351	1.200	0.1572	0.1969	1.113	0.09043	0.08545	2	177	40.38	27.65
LEO/MEO	0.000	0.009542	1.201	0.1482	0.09361	0.1702	0.1001	0.03613	2	28	15.89	3.966
LEO/GEO	0.000	0.009542	1.201	0.1483	0.09414	0.1702	0.1001	0.03613	2	28	15.93	4.000
LEO/MEO/GEO	0.000	0.009542	1.201	0.1482	0.09361	0.1702	0.1001	0.03613	2	28	15.89	3.966
Earth/Moon/Mars	0.04608	0.005875	6248	393.7	1203	4989	110.5	524.6	2	54	18.02	6.758

pattern, due to the repeating orbits of satellites. On the other hand, the federation of CubeSats exhibits a more chaotic pattern, with overall high latency punctuated by brief periods of low latency when instantaneous paths exist between points.[8]

We can also observe latency statistics from the connection establishment results in Table 5.2, further discussed later in the section. We see that Starlink offers good coverage with low latency due to its use of many satellites in low orbits, but as a result, data must travel over significantly more hops to reach its destination. We also note that Starlink struggles to connect to ground stations in higher latitudes due to the satellites' orbits, resulting in dropped messages and high latency for these ground stations. The CubeSat network also struggles; with only 121 satellites and no consistent pattern to their movement, there is insufficient coverage to provide consistent low-latency routing across the globe. However, connectivity is still possible, and it is likely that latency could be brought down with the addition of more satellites, or some relay satellites in MEO or GEO. The Iridium constellation provides good connectivity: higher orbits and fewer satellites result in lower hop counts at the cost of slightly greater latency compared to Starlink.

The federated networks of LEO, MEO, and/or GEO satellites perform very similarly to their standard counterparts, with almost identical latencies. This is because long-distance messages will preferentially travel over the LEO satellites, particularly when there is no fixed cost for each hop taken, and when bandwidth and onboard processing power are not a concern.

[8]Of course, the specifics of the results in the CubeSat configuration are highly dependent on the choice of ISL configuration. Allowing more ISLs or longer-distance links can improve connectivity, but this starts to stretch the bounds of realism – the default configuration of 2500 km is already a long distance for ISLs.

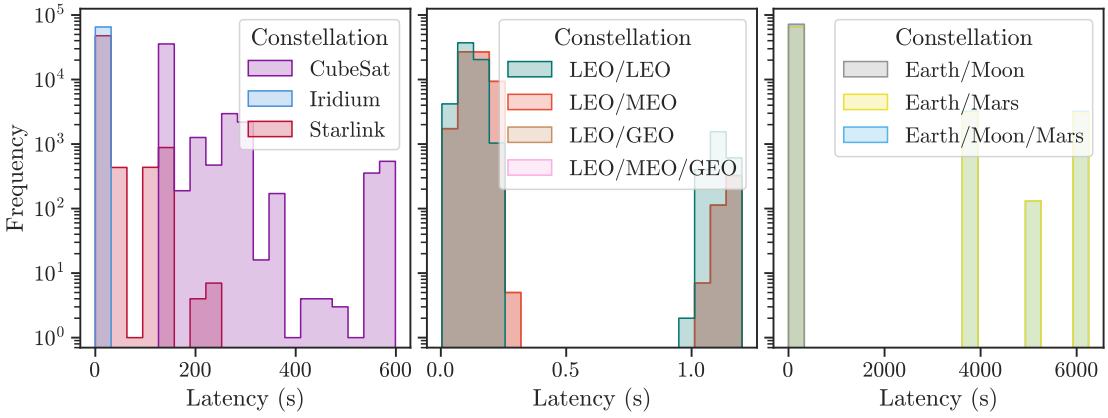


Figure 5.8: Distribution of latencies when establishing a new connection under each constellation topology, using standard OCSP with a single centralised CA.

5.7.2 Connection Establishment

We look next at the cost of establishing a new connection under each configuration. To evaluate this, we ran 202 simulations, aggregating the results into useful insights. Table 5.2 shows the latency, establishment overhead, and number of hops required to initiate communication under each network topology when using the OCSP protocol under its default configuration. We can also see the distribution of latencies in Figure 5.8. From these we gain some interesting insights about the effectiveness of various constellations. We can see that the CubeSat network has high latency, and struggles to route all messages – this is likely due to the satellites' limited and patchy coverage. If simulations run for long enough it will likely eventually route messages, so such a network may be suitable for certain use cases, but making calls back to a centralised CA will incur significant latency costs. We can also see that Starlink achieves very low latencies on many messages but struggles to route some in a timely manner, due to the Phase 1 satellites struggling to reach ground stations at higher latitudes. This is mitigated in the LEO/LEO network, which can make use of Iridium to expand coverage – this federated network performs better than either constellation acting by itself, reducing the overhead of connection establishment. However, the other federated networks do not make significant use of the higher layer satellites, so the results are almost identical to Iridium acting alone. We also see that the interplanetary network performs significantly worse, with a very high

Table 5.3: Full results for the connection establishment scenario on the terrestrial and federated constellations, for each PKI configuration tested.

Configuration			Dropped (%)	Latency (s)				Establishment Overhead (s)			Hops			
Distributed	PKI	Variant		Min	Max	Mean	Std	Max	Mean	Std	Min	Max	Mean	Std
	OCSP	–	0.04608	0.005875	6248	393.7	1203	4989	110.5	524.6	2	54	18.02	6.758
	OCSP	Stapling	0.000	0.005875	6250	396.1	1207	4989	202.8	846.0	2	54	18.03	6.663
	OCSP	Validator	0.000	0.005875	4989	293.0	870.7	1296	3.986	70.14	2	48	17.90	5.452
	CRL	–	0.000	0.005875	4989	395.9	1152	4989	111.0	383.5	2	53	18.01	6.603
	CRL	Broadcast	0.000	0.005258	3985	284.9	855.9	—	—	—	2	24	6.047	2.999
✓	OCSP	–	0.000	0.005875	3985	285.1	856.0	30.01	0.1635	0.8505	2	37	16.49	4.494
✓	OCSP	Stapling	0.000	0.005875	3985	285.1	855.9	230.1	0.3703	6.873	2	36	16.56	4.371
✓	OCSP	Validator	0.000	0.005875	3985	285.0	855.9	10.07	0.03374	0.09609	2	36	16.78	4.244
✓	OCSP	Hybrid	0.000	0.005875	3985	285.0	855.9	230.1	0.1538	1.491	2	41	16.05	4.143
✓	CRL	–	0.000	0.005875	6248	303.0	924.2	6248	18.12	216.0	2	51	16.72	4.585
✓	CRL	Broadcast	0.000	0.005258	3985	284.9	855.9	—	—	—	2	32	11.38	3.296

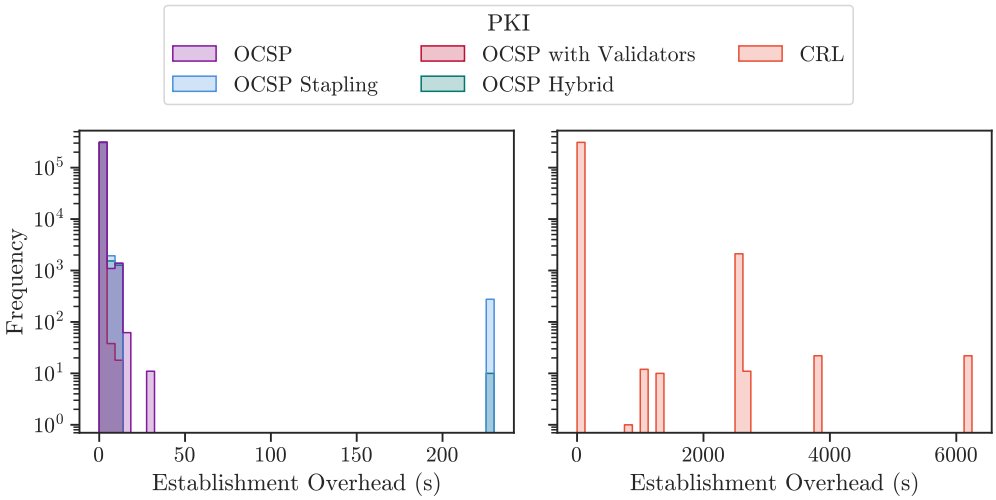


Figure 5.9: Establishment overhead of initiating a connection in the Earth/Moon/Mars network under the distributed CA configuration, for each PKI configuration tested.

time overhead and a large number of messages dropped. This is due to each message requiring a handshake with an Earth-based CA, even those received by nodes on other planets – this can be mitigated by distributing CAs throughout the network.

Next, we compare each of the different PKI configurations, focusing on the Earth/Moon/Mars network. We can see these summarised in Table 5.3. In the centralised configuration of each protocol, latency and establishment overhead are both so high as to be unusable, since every connection establishment requires communication with an Earth-based CA. However, the distributed case shows a significant improvement in establishment overhead, with results comparable to those of the Earth-based constellations! This reduction in overhead is further illustrated in Figure 5.9 (we also see the number of hops for each configuration in Figure 5.10),

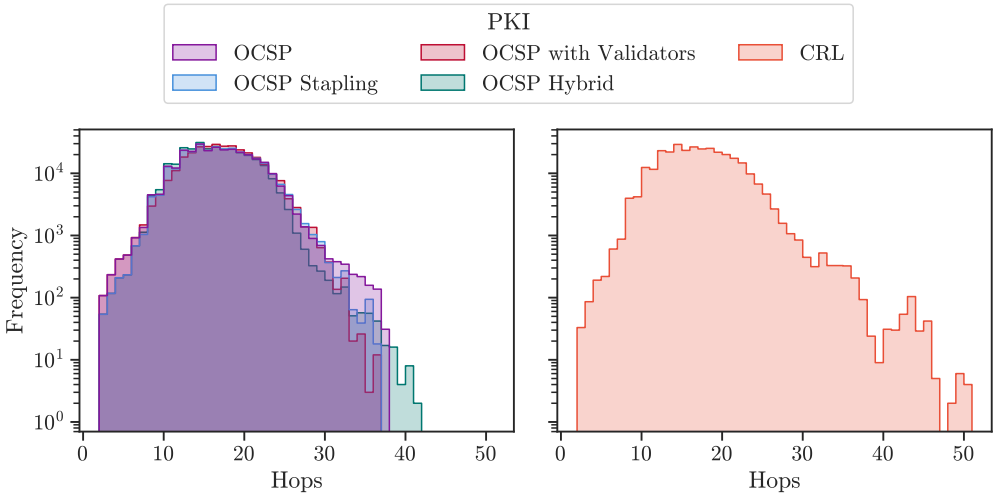


Figure 5.10: Overall hop count of messages involved in initiating a connection in the Earth/Moon/Mars network under the distributed CA configuration, for each PKI configuration tested.

enabling efficient communication across the entire network.

Although all the OCSP variants in the distributed configuration result in good performance, establishment overhead is minimised by using the "Validator" configuration, in which the certificate validity information is stapled to the message in-transit. This is unsurprising, as it removes the need for a round trip during establishment. However, it does risk link congestion by sending a high volume of traffic via the CAs – to mitigate this, the "Hybrid" approach uses in-transit validation for messages traversing the relay links, and normal OCSP Stapling elsewhere. This retains the benefits of low overhead for cross-segment messages, providing better performance than OCSP Stapling while minimising the risk of congestion.

Finally, we discuss the differences between CRL-based systems and OCSP. Using CRLs can reduce the number of messages sent – although the statistics may appear similar to OCSP, it is important to remember that the establishment exchange only needs to be done for the first message sent within the CRL's expiry window, rather than being required for every new node initiating communication. Any subsequent messages do not require a CRL to be requested, as it is already cached. This can significantly lower the number of messages sent, at the cost of security – if a CRL has been cached, then messages using a revoked key will be accepted until the next

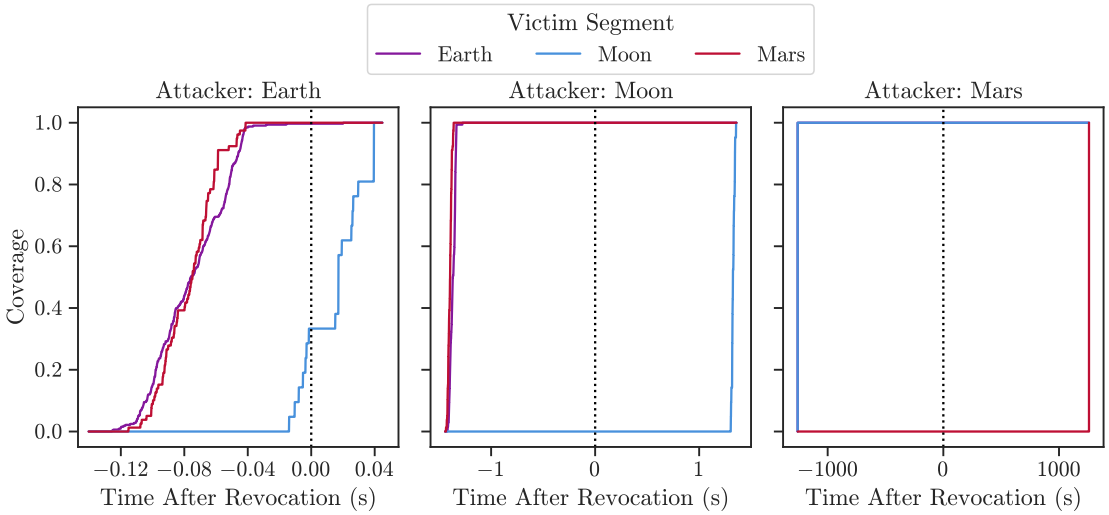


Figure 5.11: Coverage over time for a revocation, relative to the time at which the revocation was issued. Revocation originates from an Earth-based CA, under the base OCSP configuration with distributed CAs.

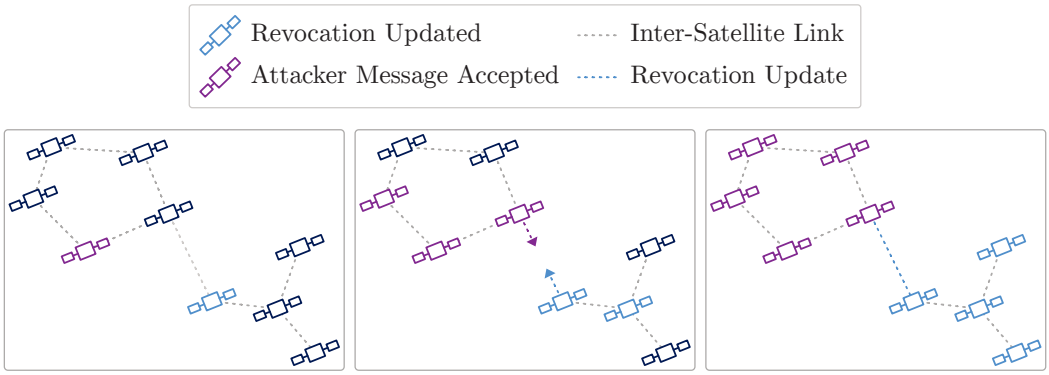


Figure 5.12: Example of the race between attacker and CA, when an attack and revocation originate in different network segments. The attacker's message can propagate quickly within its own segment, but by the time its message traverses the relay link, the revocation has already been distributed within the second segment.

CRL is received. Additionally, although the number of messages is lower, each CRL will be much larger than a single OCSP message as it contains the full list of revoked certificates. This may be mitigated through the use of techniques like "delta CRLs" to reduce the size of each message [215], although these risk opening up further threats through attackers blocking specific portions of the CRL from being delivered.

Table 5.4: Coverage results (time taken for revocation to reach all nodes) for simulations on the Iridium constellation, grouped by attacker location and CA from which the revocation originates.

Configuration		Attacker Segment	Coverage Time (s)	
PKI	Variant		Earth Origin	Space Origin
CRL	–	Ground	0.02236	0.004453
CRL	–	Space	0.01338	0.0008970
CRL	Broadcast	Ground	0.06511	0.08838
CRL	Broadcast	Space	0.07358	0.09156
OCSP	–	Ground	0.02236	0.004453
OCSP	–	Space	0.01338	0.0008970
OCSP	Stapling	Ground	0.02233	0.004423
OCSP	Stapling	Space	0.01334	−0.04131
OCSP	Validator	Ground	0.02233	0.004423
OCSP	Validator	Space	0.01334	−0.04131

5.7.3 Key Revocation

For this scenario, we focus on the interplanetary Earth/Moon/Mars network, with Iridium as a reference terrestrial satellite network. We ran 1044 simulations with different configurations, discussing the most interesting results in this section.

Figure 5.11 shows the coverage time for a revocation under OCSP for each segment in the network. We can see that the coverage time is often negative for segments in which the attacker is not present – this is due to the long distances between segments, enabling the revocation to reach the CA long before the attacker's message traverses the relay link. As a result, only messages sent significantly before of the revocation will be accepted in that segment, resulting in a negative coverage time. This race between the attacker's message and a revocation from a different segment is illustrated in Figure 5.12 – due to the vast distances involved, each segment is rapidly covered by its respective message, but cross-segment messages incur a significant delay. Consequently, a revocation can be sent well after the attacker's message and still protect the victim node in question, as the attacker's message will take much longer to reach the victim CA – this is how negative coverage times can arise in the results.

In contrast, when the attacker and revoker are located in the same segment, coverage time is primarily influenced by each node's proximity to the CA, leading to

Table 5.5: Results (time taken for revocation to cover all nodes) for simulations on the Earth/Moon/Mars constellation with a distributed CA. Attacker is in the Earth segment, results are grouped by revocation origin and victim segment.

Configuration			Coverage Time (s)		
PKI	Variant	Revocation	Earth	Moon	Mars
OCSP	–	Earth	0.01617	3.947	4.835
OCSP	–	Moon	1003	−997.5	682.9
OCSP	–	Mars	2474	1840	−2467
OCSP	Stapling	Earth	0.01379	2.242	0.2759
OCSP	Stapling	Moon	1003	1003	1003
OCSP	Stapling	Mars	2474	2474	2475
OCSP	Validator	Earth	0.01379	2.242	0.2759
OCSP	Validator	Moon	1003	1003	1003
OCSP	Validator	Mars	2474	2474	2475
OCSP	Hybrid	Earth	0.01379	2.242	0.2729
OCSP	Hybrid	Moon	1003	996.1	1002
OCSP	Hybrid	Mars	2474	2472	1322
CRL	–	Earth	0.01617	3.787	3.918
CRL	–	Moon	1003	−998.8	682.1
CRL	–	Mars	2474	1840	−2471
CRL	Broadcast	Earth	0.08711	3.787	3.775
CRL	Broadcast	Moon	1003	−350.4	1005
CRL	Broadcast	Mars	2474	2162	1846

rapid overall coverage with some nodes protected later than others. Our objective is to optimise the PKI configuration to achieve the earliest possible coverage in each scenario, minimising damage within the speed-of-light constraints inherent in long-distance communication.

We can see aggregated coverage information for a wider range of configurations in Tables 5.4 (Iridium) and 5.5 (Earth/Moon/Mars). In these tables, the coverage time is the time after which every node in the segment is protected by the revocation. We can once again see a significant difference between the cases where the attack originates from the same segment as the revocation, and when it originates from a different segment – in the latter case it takes significantly longer for the segments to be covered. This is because the revocation does not always have time to propagate to the segment in which the attacker is present, and must instead race the attacker's messages.

The overall coverage statistics are quite similar between configurations, but there are some differences – the base OCSP and CRL configurations perform

Table 5.6: Results for the key revocation on Earth/Moon/Mars, when CAs subscribe nodes to updates on requested certificates using OCSP.

Configuration		Accepted (%)					
		No Subscribe			Subscribe		
OCSP Variant	Revocation	Earth	Moon	Mars	Earth	Moon	Mars
–	Earth	4.785	39.58	22.92	4.785	39.58	22.92
–	Moon	99.32	0.000	50.00	99.32	0.000	50.00
–	Mars	100.0	77.08	0.000	100.0	77.08	0.000
Validator	Earth	0.2930	6.250	4.167	0.2930	6.250	4.167
Validator	Moon	100.0	91.67	95.83	99.90	91.67	81.25
Validator	Mars	100.0	100.0	91.67	100.0	83.33	91.67
Hybrid	Earth	0.000	0.000	0.000	0.000	0.000	0.000
Hybrid	Moon	100.0	0.000	52.08	100.0	0.000	52.08
Hybrid	Mars	100.0	87.50	0.000	100.0	87.50	0.000

better in some cases, and OCSP Hybrid provides improvements over the Stapling and Validator configurations.

We have already established that for interplanetary networks a single centralised CA is not feasible due to high latency. We therefore focus on decentralised configurations in this analysis – in many of the centralised cases all the attacker's messages can be rejected, but the latency overhead for communication is too high to be useful. Similarly, we focus on the best-case scenario for the attacker, in which nodes already have the attacker's certificate status cached, so they will not immediately request up-to-date revocation information from the CA.

Subscribe to Updates

Having noted that a large number of messages get accepted if the attack and revocation originate from different segments of the network, we evaluate potential methods of mitigating this threat. One such method requires OCSP CAs to register subscriptions for nodes that request a certificate status, forwarding any updates or revocations directly to that node. This is not possible for certain configurations (e.g. OCSP Stapling, as the CA does not know the destination of its stapled certificate status) but is a good practice in supported cases, as caches of compromised certificates can be cleared sooner, reducing the attacker's window of opportunity.

We can see in Table 5.6 that this reduces the number of vulnerable nodes for the "OCSP Validator" configuration, but does not have a significant effect on the other

Table 5.7: Results for key revocation when the relay nodes are used as a firewall in the "OCSP Hybrid" configuration, compared to the base results.

Origin		Coverage Time (s)					
		No Firewall			Firewall		
Attack	Revocation	Earth	Moon	Mars	Earth	Moon	Mars
Earth	Earth	0.01379	2.242	0.2729	0.003222	−0.03161	−0.03161
Earth	Moon	1003	996.1	1002	1003	−998.8	679.7
Earth	Mars	2474	2472	1322	2474	1839	−2471
Moon	Earth	714.5	907.4	5.739	−1002	806.6	−999.6
Moon	Moon	1.058	−2.149	0.7961	−2.687	−2.150	−324.3
Moon	Mars	2641	2789	637.0	1473	2687	−3472
Mars	Earth	2320	2319	2334	−2471	−2467	2450
Mars	Moon	3010	3010	3025	−1468	−3469	3129
Mars	Mars	3.741	2.552	0.1273	−0.003290	−633.6	0.1309

configurations. However, this is still beneficial as it enables the system to converge faster following a revocation, minimising the duration for which a compromised key can be used while still using caches to improve efficiency.

Relay Firewall

Finally, we propose a new technique in which the relay nodes are used as a firewall to filter out messages which use revoked certificates. Under this configuration, the relay filters all messages passing through which are making use of revoked keys, stopping them from reaching their destination or marking them as invalid (depending on implementation). This does not affect the message's path, as all interplanetary traffic travels via the relays anyway, and in a number of cases (OCSP with Validators, OCSP Hybrid) the certificates are also validated by them. The technique has the potential to reduce the race between the revocation and the attacker's messages when messages are waiting to be forwarded at a relay, allowing all of these messages to be dropped by the firewall instead. It also allows relays to drop messages from an attacker even if a valid certificate status has already been stapled to the message, reducing the reach of the attacker across long-distance relays.[9]

The results can be seen in Table 5.7. We can see that this technique has a huge impact on the coverage time of the revocations, particularly when the attack

[9]Depending on implementation, the firewall may inform the recipient about dropped messages, or simply label the message as unauthenticated so the recipient can choose what to do with it.

and revocation originate from different segments: the attacker's segment is largely unaffected, but the other segments in the network can be protected substantially earlier. This brings us closer to the expected best-case performance of revocations under distributed PKI, bounding the attacker's reach to the smallest segment possible and covering the other segments in the network as early as possible. This technique also reduces load on the interplanetary relays, preventing denial of service by attackers using compromised keys to exhaust the bandwidth of relay links.

5.8 Discussion

Related works hypothesise that there is no "one size fits all" approach to PKI in satellite networks [205, 208]. Our results further confirm this: distributed OCSP with firewalls on each of the relays can provide high performance with fast revocation coverage, but some high security applications may prefer centralisation, even at the cost of increased latency. Our results enable satellite operators to make informed decisions about PKI implementation based on their specific needs and constraints: DSNS can be used to test the specifics of a PKI system against a network topology before launch and deployment, ensuring the desired balance between security and performance can be achieved. Additionally, our testing framework can be used (alongside DSNS or another network simulator) to test newly proposed key management systems against a wide range of network topologies, measuring its performance and security characteristics.

When it comes to security, satellite operators are likely to have overlapping and occasionally incompatible needs, adding complexity when systems are networked together. We therefore propose a mixed-protocol system, in which key management systems are designed to work alongside one another. Both OCSP and CRLs are already highly intercompatible, as they are based upon existing internet technologies, and we have shown in our "OCSP Hybrid" protocol that mixed approaches are also possible in an interplanetary system. This would present the opportunity for organisations to choose PKI requirements on a per-device basis. A system that prioritises security at all costs could use a single centralised CA with no caching,

to ensure all communication is centrally authorised at the cost of substantially increased latency. On the same network, a less security critical application could make use of the OCSP Hybrid protocol with long cache times to prioritise latency while still securing communication. This permits a high degree of control on each device without sacrificing intercompatibility, enabling nodes to continue to communicate with one another.

We must also consider the complexity of implementation of these technologies. Some works evaluated in Section 5.2 introduce new protocols; although this enables more significant changes, these new protocols have not been tested as extensively. By using protocols that have already been standardised and are in wide use across the terrestrial internet, we minimise the risk of security threats in the PKI system itself. We also benefit from the fact that standards already exist, so standardisation bodies such as the CCSDS do not need to build new standards from scratch, instead building off what already exists. This could involve further additions to the "BPSec" proposed standard [202], adding new block definitions and behaviours to enable the new functionality. By building standards with good performance and security properties we can ensure better compatibility between systems, taking another step towards the eventual goal of a unified interplanetary internet.

5.8.1 Limitations and Future Work

The results in this chapter come with one notable limitation: latency and revocation time are not perfect metrics when it comes to evaluating PKI systems. As previously discussed, an attacker must first be discovered before a certificate can be revoked, which can sometimes take a very long time – sometimes on the order of days or weeks. When operating on this timescale, it may not seem as significant to ensure a revocation occurs a few thousand seconds sooner or later. However, these optimisations can still save hours of time, which could prevent attackers from making last-ditch efforts to exploit their position once discovered.

Furthermore, improving coverage in this way can have a positive impact beyond the scope of PKI. There are a number of other areas which rely upon delivery of

critical messages in a timely manner: in particular, anything involving propagating configuration changes across the network, updating routing information to exclude faulty or compromised devices, or alerting operators to changes elsewhere in the network. By combining speed-optimised message delivery with network-level traffic prioritisation, we can ensure these critical messages reach their destination as quickly as possible so that disruption is minimised.

However, computational impact, bandwidth, and storage use are all important factors in a PKI context, particularly when satellites have limited hardware and communicate over constrained radio links. Future work should therefore aim to extend DSNS to simulate each of these factors. This will enable regulators to better understand how PKI protocols should be developed and implemented, providing the fast connection establishment and effective revocations seen in this chapter while remaining within the constraints of satellite hardware. It will also enable the concrete benefits of federated constellations to be demonstrated, making use of a small number of powerful satellites to provide connectivity to a large number of devices while using LEO satellites for short-distance connections with low latency. We have shown in Section 5.6.3 that DSNS can be easily extended to support new features and protocols, facilitating this research and providing real-world usefulness to the space security community.

Future research may also consider testing "optimistic" protocols – that is, protocols which initially assume the certificate to be valid, rolling back the state if this is later shown not to be the case – or protocols like QUIC, which enable communication with zero round-trip-time setup [239]. However, this raises additional security concerns: firstly, the system must be designed in such a way that state can be efficiently rolled back without losing other data or leaving artefacts behind. Many satellites have tight resource constraints, and it is vital that adversaries are prevented from exploiting these resources without proper authorisation by opening temporary communication channels with invalid credentials. If these protocols are to be used, there must still be a mechanism for establishing contracts such that resources are not wasted on unauthorised users. Similarly, there is scope for

evaluation of the scope and limitations for retroactive revocations, considering which actions can be safely rolled back without affecting general operation.

Another area of future research is the exploration of mechanisms that do not rely on revocation, such as short-lived certificates [240]. These certificates, which are already gaining popularity in terrestrial networks, have a limited validity period (typically ranging from 24 to 72 hours) and can reduce the need for revocation lists – the damage that can be caused by a compromised certificate with a short lifespan is limited, as it will expire soon anyway. This approach could be particularly useful in satellite communications, particularly when nodes or small sections of the network remain disconnected for significant amounts of time, and still need to operate independently of a CA. However, before deploying this at scale, operators will need to quantify the likely damage that could be caused in a real-world setting by a compromised certificate before its validity period naturally ended, to see if allowing this is worth the risk. Additionally, operators will need to figure out the optimal certificate lifespan for a given network topology, in order to ensure certificates do not expire before they can be renewed when links go down. The good news is that short-lived certificates can be implemented alongside techniques based on OCSP or CRLs without affecting compatibility – the exact same certificates will be used, simply with shorter lifespans.

Routing

As we have already discussed, one area of research in which DSNS can be particularly useful is routing. As satellite networks grow larger, many traditional routing paradigms start to break down – many terrestrial routing protocols assume link loss is unpredictable, and require any changes to the network topology to be discovered and communicated across the network. At the same time, gossip-based protocols popular in related works squander most of the benefits of space networking: even though the movement and connectivity of satellites is entirely predictable, these protocols assume all data must be routed probabilistically [234, 236, 241]. Purely predictive or source-based routing also has problems in the case of unexpected

link or node loss (caused by random failure or targeted attack), since it cannot recover or reroute if the network topology changes.

A routing protocol optimised for space might take inspiration from the Border Gateway Protocol (BGP), in common use across the terrestrial internet [242]. BGP differentiates between routing within an autonomous system (e.g. on a local network) and routing between autonomous systems (e.g. across the internet), thus reducing the number of updates that need to be propagated across the entire network. This type of mechanism has the potential to work particularly well in satellite systems, and particularly interplanetary networks – each segment is defined as a separate autonomous system, with relay nodes sitting at the borders between them. Each segment then only needs to handle routing and updates within the segment, with the borders handling routing between segments. The main problem that remains to be solved is differentiating between predicted and unexpected link losses: the protocol needs to be able to discover and report unexpected losses across the network segment, so traffic can reroute around them. Some preliminary work has been done in this area [243], but it currently remains limited to simple grid topologies. If this can be solved for arbitrary topologies, then routing systems can maintain the efficiency and performance gains of optimal point-to-point routing combined with fast recovery in the case of unexpected failure or loss, without resorting to probabilistic delivery mechanisms. However, BGP is not without its problems – its versatility in terrestrial networks has caused a number of routing issues including highly suboptimal routes, intentional traffic misdirection by nation state actors, and traffic blocking [244]. Regulatory agencies like the CCSDS ought to take inspiration from BGP and other protocols, but optimise routing mechanisms for the satellite network use case.

There are also some parallels to the related area of critical message delivery. The Bundle Protocol standard describes an "extended class of service" extension, which adds fine-grained control of bundle priority, affecting message delivery behaviour [245]. Any bundle marked as critical will be sent across *every* path that could get it to its destination, rather than the optimal path only. The extension also enables bundles to be marked as "best-effort" (disabling any retransmission

mechanisms) or as requiring reliable transmission, in which case loss detection and retransmission is required to be used. Similarly, the Licklider Transmission Protocol (LTP) is designed to provide mechanisms for reliable delivery across extremely high-latency networks, enabling per-hop retransmission and the prioritisation of specific parts of the message [246]. Through the use of network simulation, the parameters of these protocols can be optimised to work under a wide range of known attack strategies, and new techniques may be developed to mitigate specific areas of vulnerability.

DSNS can be a highly useful tool in comparing, developing, and testing these different approaches to routing – not only have we shown that it is easy to implement and modify new protocols, but it is also effortless to swap out different network topologies, simulate link and node failures, and run large numbers of simulations to gather a wide range of performance metrics on each protocol. As interplanetary networks grow in significance and research interest turns to optimising various aspects of networking in this new domain, DSNS is well positioned as a tool to accelerate research in this and other areas.

5.9 Conclusion

In this chapter, we evaluated the feasibility of using terrestrial PKI protocols in interplanetary networks. We set out the problem of key management in these networks, highlighting in particular the difficulty of querying a central authority in a network with sporadic high-latency links. We outlined the goals of a key management system in this context, and designed a number of standardised scenarios which can be used to assess the suitability of a protocol in an interplanetary network, or tweak its configuration to optimise it for a given network. To achieve this, we built the Deep Space Network Simulator (DSNS), a new network simulator designed to efficiently handle the simulation of arbitrary satellite networks, including interplanetary networks, which can easily be extended to support new features and protocols. Using DSNS, we implemented the OCSP and CRL protocols used for communication on the terrestrial internet, and show through simulations that

they can be adapted to work efficiently in interplanetary settings, with minimal message overhead. We also proposed and evaluated a hybrid technique incorporating multiple aspects of OCSP, handling cross-segment messages differently from intra-segment communication, increasing efficiency. Finally, we tested a new configuration in which interplanetary relay nodes are additionally used as a firewall, filtering out invalid messages to further improve performance by reducing the reach of an attacker, and minimising their load on the relay links.

In contrast to the current consensus, we have shown that terrestrial PKI can be used effectively in current and future satellite networks, ensuring efficient, low-latency connection establishment by distributing certificate authorities across the network. This removes the need for purpose-built protocols designed for DTNs which rely on probabilistic guarantees of success. By adapting existing terrestrial protocols we ensure compatibility between terrestrial and interplanetary networks, bringing us one step closer to a unified interplanetary internet.

5.9.1 Availability

The source code of DSNS will be made available upon publication of the tool release paper, under the GPLv3 license to ensure it can be used and extended to enable future research. Until then, it is available on request, and has already been released in this manner for a number of student projects.

6 Conclusion

Contents

6.1 Summary of Results 185
6.2 Future Work 186
6.3 Final Remarks 187

6.1 Summary of Results

In this thesis, we have achieved significant improvements in authentication for both Old and New Space systems, enhancing the security and resilience of satellite communication. Through exploration of physical layer fingerprinting and Public Key Infrastructure (PKI), we have identified complementary techniques that can be applied across a diverse range of contexts, including the authentication of individual satellites, large constellations, and even interplanetary networks, to distinguish legitimate from malicious communication and ensure the secure exchange of information.

In Chapter 3, we showed that satellite transmitters could be authenticated using their physical layer transmitter fingerprints, looking in particular at the message headers at a high sample rate. We captured the largest known dataset of high sample rate satellite signals, comprising over 10 million messages from Iridium transmitters, and used this data to train the SatIQ system to differentiate transmitters from one another. We demonstrated its capability in this area, and showed that it is even more effective at detecting simple replay attacks than the prior state of the art.

In Chapter 4, we investigated attacks on fingerprinting systems in even greater detail, looking at an attacker targeting the fingerprinting system directly. We established a range of attacks, including simple jamming, optimised jamming and spoofing, and data poisoning, and tested all of these attacks against the SatIQ system. Our results showed that an attacker targeting the fingerprinting system directly can cause moderate to severe disruption, particularly in a jamming context, and highlighted the importance of deploying fingerprinting as part of a suite of attack countermeasures and signal intelligence tools. In carrying out these experiments, we also uncovered a new technique which can be used to train fingerprinting systems with only a single transmitter's worth of data, optimised for the detection of attacks rather than simply to distinguish legitimate transmitters. We demonstrated the effectiveness of this technique, even when detecting replay attacks on previously unseen transmitters, thus enabling fingerprinting to be deployed in a wider range of contexts across modern and legacy systems.

In Chapter 5, we demonstrated the effectiveness of terrestrial approaches to PKI in interplanetary networks, and established configuration requirements for this to be the case. We also presented optimisations to these protocols, enabling increased efficiency and improved security, minimising the reach of an attacker and their load on relay links within the network. Finally, we built the Deep Space Network Simulator (DSNS), optimised for testing protocols such as these in large interplanetary networks, and have made it openly available to aid future research.

6.2 Future Work

We have already discussed some interesting avenues of future work in previous chapters. Of particular importance among these is the combination of multiple defence strategies to provide broad coverage – attacker capabilities and goals are ever improving and changing, and it is no longer sufficient to deploy a single source of protection against attacks. We have shown that physical layer fingerprinting is not immune to attacks, and even when proper cryptographic protection can be deployed vulnerabilities still exist in the supply chain and human factors. Satellite operators should therefore focus not on a single strategy but instead on combining multiple sources of intelligence to provide a broad understanding of attacks, and deploy a range of countermeasures depending on the threat currently being faced. Research can support this by looking at how well different physical layer techniques work together, and organisations like ESA or standards bodies like CCSDS can provide recommendations for operators on which techniques are most effective and how they can be combined. These best practices may differ between modern and legacy systems, making a particular distinction between systems in which the space segment can be updated to support new behaviour, and those in which it cannot, in which operators must instead rely upon existing hardware and infrastructure.

As fingerprinting and other physical layer techniques start to be deployed as security measures, standardisation also becomes increasingly vital. A standard suite of tests, in which a satellite system could be subjected to a range of attacks to assess its reliance, would be immensely useful in testing the deployment of

countermeasures and in the development of new techniques. Such a system would also be a big step towards fixing the problem of reproducibility in physical layer security, enabling academics and operators alike to deploy, test, and compare techniques in a standardised manner.

There are also a number of open questions surrounding the topic of PKI. For instance, protocols ought to consider the process of securely bootstrapping a certificate – once a certificate has been revoked, it may be difficult to re-establish connectivity. Current systems often rely upon trusted keys which may only be used for rekeying, or provide mechanisms to downgrade security to an unauthenticated link, but both of these mechanisms are quite fragile. A better solution would enable new certificates to be issued and installed without increasing the attack surface or requiring authentication to be temporarily disabled.

Another avenue worth exploring, beyond those already discussed, is intercompatibility between terrestrial and space systems. This is not as simple as ensuring space protocols are sufficiently similar to their terrestrial counterparts, although this is useful – the requirement for radio links in space means there will still need to be some gateway devices which connect the two together. Depending on the level of similarity, this may require some amount of encapsulation or conversion. One way in which this could be developed is by connecting a network simulator such as DSNS to a real-world terrestrial network, via a simulation actor which translates between real network traffic and simulation events. This would enable researchers to test against a larger satellite network without causing any impact to deployed systems, iterating upon a compatibility protocol and building towards useful standards.

6.3 Final Remarks

This thesis has demonstrated the potential of physical layer fingerprinting and PKI to enhance the security of satellite communication systems. However, as attacker capabilities and goals continue to evolve, it is essential that the techniques and strategies used to secure these systems also adapt and change, particularly as the industry moves towards increasingly interconnected systems.

Fingerprinting and other physical layer information will play a crucial role in this effort, but to be maximally effective they must be deployed as part of a comprehensive suite of countermeasures. By combining multiple defence strategies and physical layer techniques, operators can stay ahead of emerging threats.

As the space industry continues to grow, the importance of secure communication protocols will only increase, and the ability to establish trust and authenticate identities over long distances will be essential to supporting this growth. Furthermore, standardisation will be crucial in enabling seamless communication between space systems, terrestrial systems, and ultimately laying the foundation for a unified interplanetary internet. It is hoped that the findings in this thesis will inform and shape upcoming standards, furthering the security of future space systems.

Appendices

A
Signal Processing

This appendix provides a useful overview of a number of hardware and software signal processing concepts used elsewhere in this thesis.

A.1 Frequency Bands

The radio spectrum is divided up into a number of frequency bands. Within these bands, frequency ranges and narrow channels are allocated for specific applications, or left open for public or amateur use. These are summarised in Table A.1.

Almost all of these frequency bands are used in satellite communication, particularly between VHF and K_u, with the exception of K-band, due to atmospheric attenuation from water vapour in the atmosphere. Some notable examples of different band usage include the L-band used by GPS for navigation, C-band for satellite television, and X-band for military applications [248]. The "SatNOGS" community-operated network of satellite ground stations reports over 2400 satellite transmitters operating in UHF, as well as over 400 in VHF and 800 in S-band, at the time of writing [249].

A.2 Antennas

Antennas are used to convert radio waves into electric current, or vice versa, in order to receive or transmit signals respectively. The design of the antenna affects its frequency response (the range of frequencies it transmits or receives well), radiation pattern (gain in each direction), directivity, and polarisation of the radio waves [250].

Table A.1: Radio frequency bands as allocated by IEEE [247].

Band	Frequency range (GHz)	
	Lower	Upper
HF	0.003	0.03
VHF	0.03	0.3
UHF	0.3	1
L	1	2
S	2	4
C	4	8
X	8	12
K_u	12	18
K	18	27
K_a	27	40
V	40	75
W	75	110
mm	110	300
THz	300	1000

There are a wide range of antenna designs in use, each of which provide different properties and are useful in different contexts [251]. These include the following:

- The mast antenna, or omnidirectional antenna, which radiates power equally in all directions perpendicular to its axis with low gain. Its length affects its frequency response; a quarter-wave monopole is commonly used, with a length equal to one quarter of the desired wavelength.

- The dipole antenna, composed of two conducting elements pointed away from one another. The dipole also radiates equally in directions perpendicular to its elements, and the most commonly used configuration has a total length equal to half the desired wavelength.

- The Yagi-Uda antenna, which consists of a number of parallel rods. The rearmost of these acts as a reflector, in front of which is the actively driven element and one or more director elements. The directional gain of this antenna increases with the number of directors via constructive and destructive interference. The length, spacing, and number of elements varies depending on design. These antennas are widely used in the HF, VHF, and UHF bands.

- The parabolic reflector antenna, providing very high gain within a narrow beam by focusing energy from a specific direction onto a focal point. A

"feed" antenna is placed at the focal point of the dish. Larger dishes provide narrower beams, and can receive lower frequency signals. Alongside satellite communication, these antennas are used in radio telescopes for deep space observation.

- Patch antennas, which consist of a patch of metal (usually on a printed circuit board) mounted above a ground plane. The shape of the patch and the spacing between it and the ground plane affect the frequency response of the antenna and its radiation pattern.

Additionally, phased array antennas may be used, composed of multiple smaller (usually patch) antennas positioned in an array. By shifting the phase of each antenna's signal, the beam can be steered electronically without any moving components, in both transmitting and receiving signals. These are popular in small satellites and CubeSats due to their small size [252], and may also be used in ground terminals [253].

Directional antennas direct the majority of radiation in one direction (the main lobe), but usually create additional "sidelobes" at other angles, particularly directly behind the main lobe (the back lobe). These are undesired, as they waste energy in transmission and may increase interference when receiving. We discuss in Section 2.3 how this may be exploited by an attacker for eavesdropping or spoofing.

A.3 Filters

Filters are widely used in radio systems in order to attenuate or remove unwanted frequencies. These are usually implemented through hardware, but may also be implemented in software during digital signal processing. Filters provide one of four different functions:

- Low-pass filter: remove all frequencies above a cutoff, allowing only lower frequencies to pass.

- High-pass filter: remove all frequencies below a cutoff, allowing only higher frequencies to pass.
- Band-pass filter: allow only frequencies within a desired band to pass.
- Band-stop filter: remove all frequencies within a specified band, allowing everything else to pass.

A.4 Amplifiers

Amplifiers are used to convert low-power signals into high-power signals. These come in two main forms. The Low Noise Amplifier (LNA) is designed to amplify signals from a very low starting power while introducing as little noise as possible, thus minimally affecting the signal's SNR. These are commonly used in receiver systems. Power amplifiers, on the other hand, provide a much higher output power in order to drive a transmitting antenna.

An amplifier will only work over a defined range of frequencies, which differs depending on construction. Frequencies outside this range will not be sufficiently amplified, and may be attenuated or even damage the amplifier. It is therefore important that unwanted frequencies are filtered out prior to amplification.

A.5 Attenuators

An attenuator performs the opposite function of an amplifier, reducing a signal's power while minimising distortion. Unlike amplifiers, these are typically passive and do not require power, although active cooling may be required in some cases to dissipate the excess energy. They are often used in order to reduce the power of a signal, preventing damage to components.

A.6 Digital Signal Processing

Signals may be processed in the time domain, looking at the amplitude over time, or in the frequency domain, in which the signal is decomposed into its frequency

components through a Fourier transform, giving the amplitude and phase of each frequency in the signal [254]. The latter, sometimes called spectral analysis, is particularly useful in digital signal processing, as it is easy to see which frequencies are present (or absent) in the input signal. It is also easy to see the noise floor – the amount of background noise introduced by a range of sources such as thermal noise in the receiver or unwanted devices – and the Signal to Noise Ratio (SNR), the level of the signal compared to the background noise. These are measured in decibels (dB), a measure of the power ratio between two signals. The unit is logarithmic in scale, with a change in power by a factor of 10 corresponding to a change of 10 dB. Absolute signal strength can be expressed in decibel watts (dBW, the strength relative to one watt), or decibel milliwatts (dBm, relative to one milliwatt).

B
Extended Attack Results

In this appendix we outline some of the additional results from our experiments in Chapter 4 which were not crucial to the outcome, but are interesting nonetheless.

B.1 Jamming: Phase Synchronisation

Results for the jamming experiment when phase synchronisation is enabled can be seen in Figure B.1. These results are at best comparable to the original results, and in many cases produce worse performance, particularly at higher power levels. It is likely this is caused by overfitting, with random phase shifts serving to force the jamming signal to be more generalisable – this means the attacker does not need to worry about phase alignment, and can simply generate a jamming signal and align it at the symbol level.

B.2 Identity Shift

Full results for the identity shift experiment under each configuration are given in Figure B.2. We can see that performance is improved significantly when phase synchronisation is permitted. Interestingly, we also see that performance is not greatly affected by the choice of starting transmitter (random, or from the same specific transmitter), suggesting that the attacker's signal can add a new fingerprint without having to worry too much about counteracting the original fingerprint – once transmit power is high enough, the attacker's signal dominates the old data.

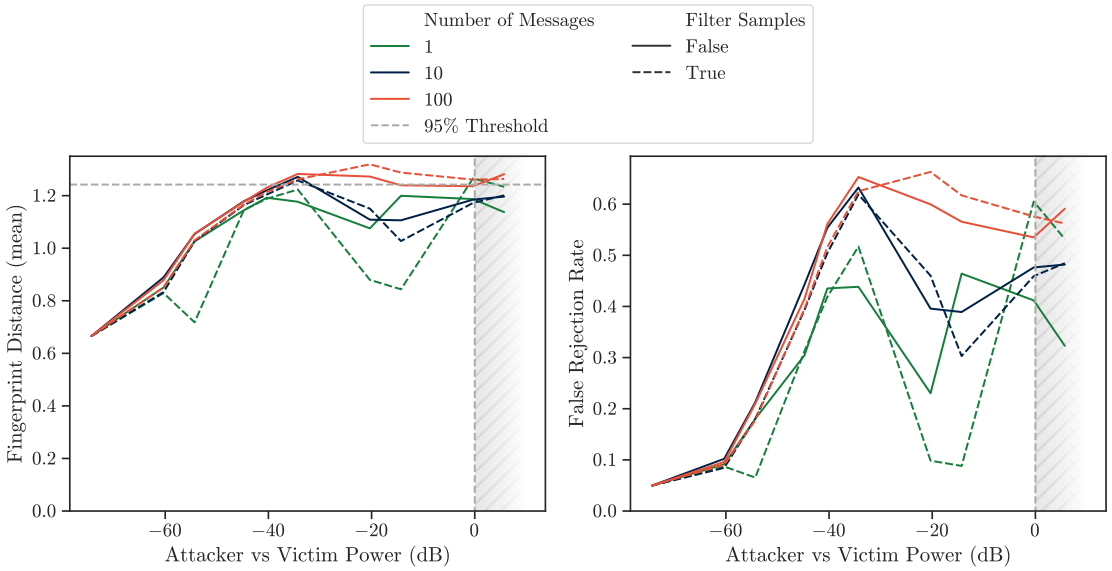


Figure B.1: Results (mean fingerprint distance and false rejection rate) from the jamming experiments, when the attacker can achieve phase synchronisation with the victim. Acceptance threshold is set such that 95% of legitimate messages are accepted. Above 0 dB, jamming is significantly more likely to affect the decoder rather than the fingerprint; this region has been indicated in grey.

B.3 Fingerprint Masking

Full results for the fingerprint masking experiment using the fixed spoofing signal (i.e., without using a GAN) are given in Figure B.3. Results are comparable to the corresponding identity shift experiments.

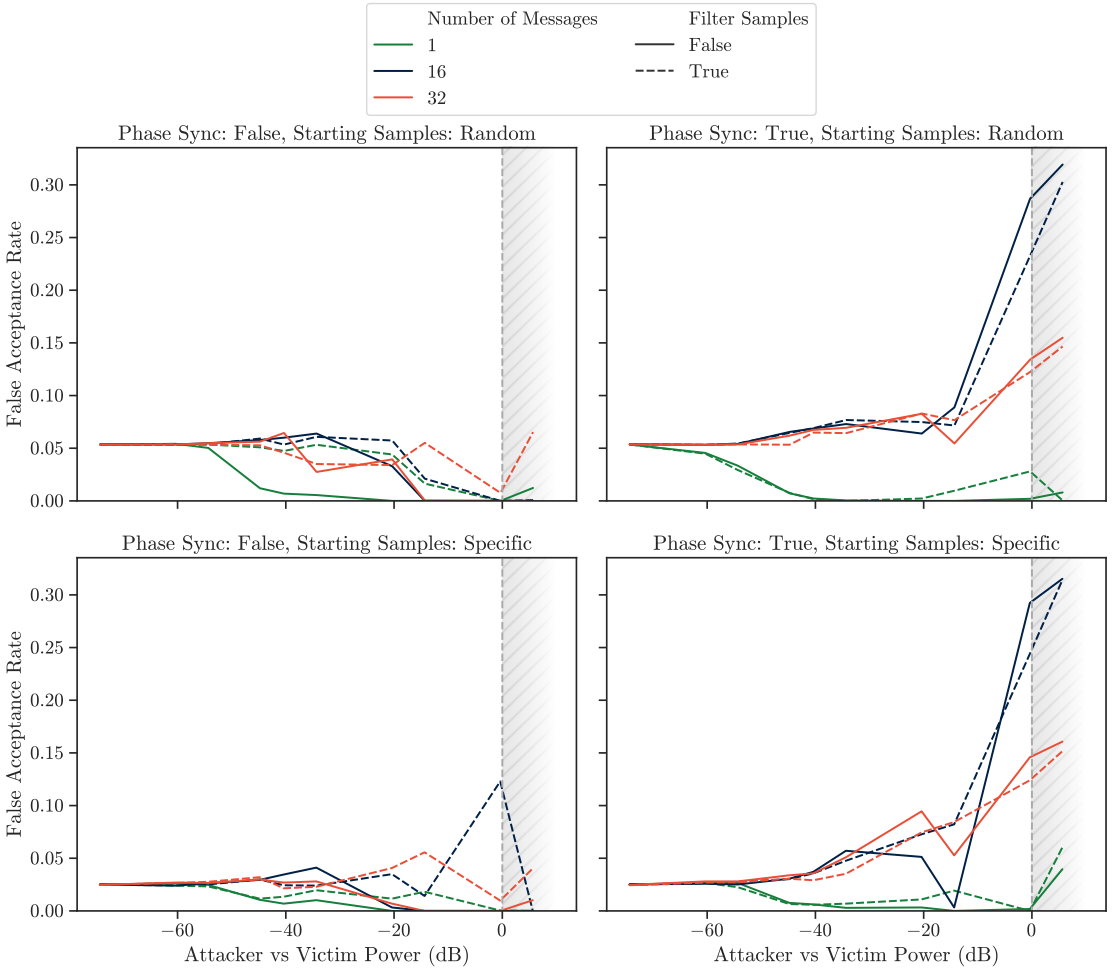


Figure B.2: False Acceptance Rate for the identity shift experiments under each configuration, as the power of the spoofing signal increases. False acceptance rates are given relative to an initial acceptance threshold, set such that 95% of illegitimate messages are rejected. Above 0 dB has been marked in grey, since spoofing is more effective than identity shifting above this point.

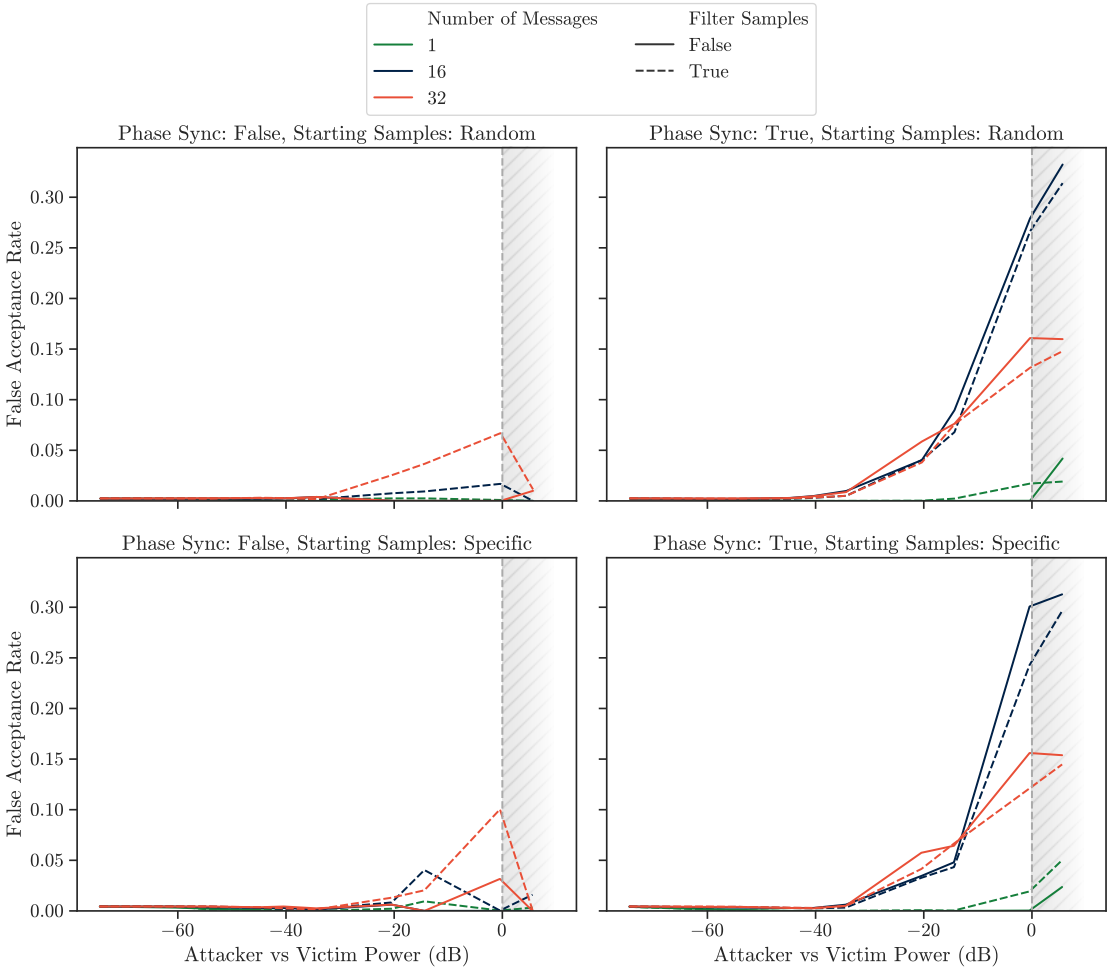


Figure B.3: False Acceptance Rate for the fingerprint masking experiments under each configuration, as the power of the spoofing signal increases, when a GAN is not used. False acceptance rates are given relative to an initial acceptance threshold, set such that 95% of illegitimate messages are rejected. Above 0 dB has been marked in grey, since spoofing is more effective than identity shifting above this point.

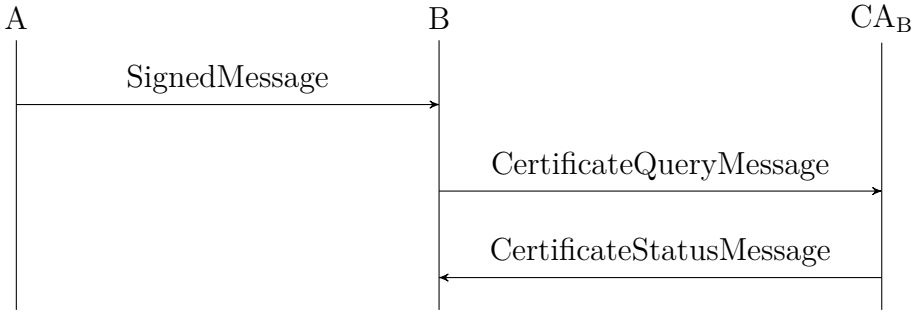
C

PKI Protocol Sequence Diagrams

In Chapter 5 we describe a number of terrestrial PKI protocols based on Certificate Revocation Lists and the Online Certificate Status Protocol. In this appendix we demonstrate nominal functionality of each of these protocols through the use of message sequence diagrams. These diagrams are shown for the distributed case, in which there are multiple CAs and authority is delegated. OCSP Hybrid is not shown here, since behaviour is identical to OCSP Stapling for communication within a segment, and identical to OCSP with Validators for communication between segments.

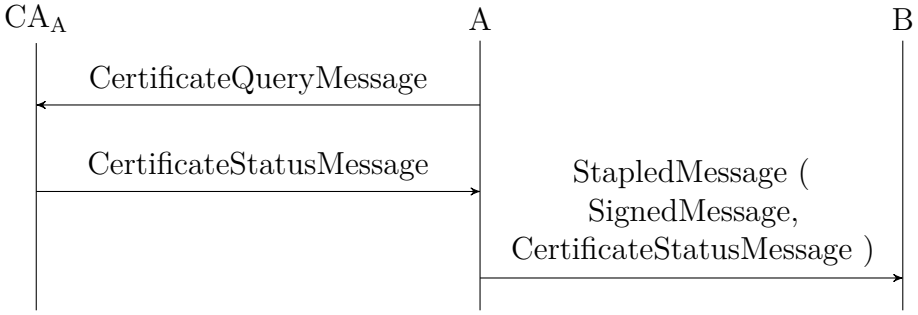
Full protocol implementations are given in the source code for DSNS, distributed alongside the original paper [20].

C.1 OCSP



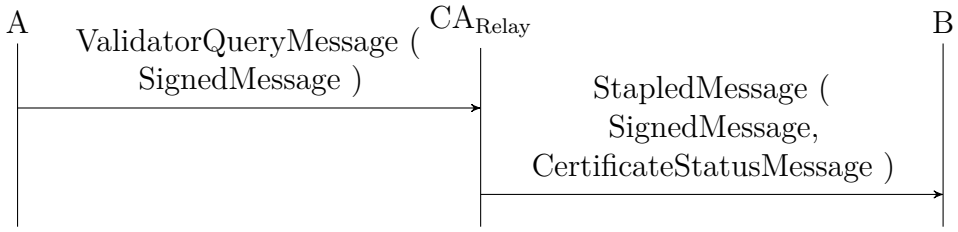
Under OCSP, when a node receives a signed message, a query is sent to the recipient's CA to ensure the sender's certificate is valid and has not been revoked. This results in quick validation as long as the recipient has its CA located nearby, but can result in longer revocations if messages need to be sent across network segments. Upon recipient of a certificate status message, the node caches the certificate for a configurable amount of time, so that on subsequent messages from the same sender it does not need to check with its CA again. Under the "subscribe" configuration evaluated in Chapter 5, the CA will remember this and update the recipient directly if the certificate is later revoked.

C.2 OCSP Stapling



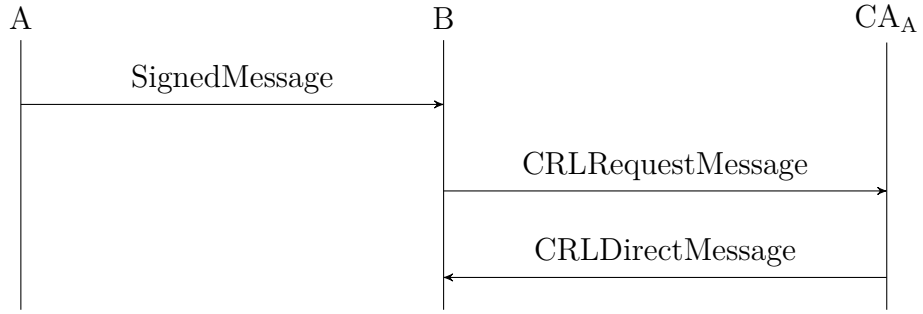
OCSP Stapling functions similarly to basic OCSP, but the sender instead queries its own CA before it can send the message. Once the status message for its own certificate has been received, it is stapled to (and sent alongside) the original message to its recipient. This shifts the burden of verifying the legitimacy of the certificate onto the sender, reducing the potential impact of message flooding attacks.

C.3 OCSP with Validators



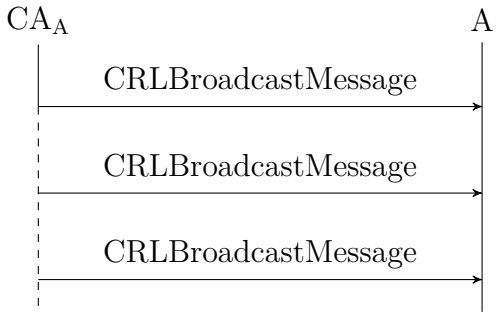
When the CA is located along the path between a message's source and destination, the overhead of OCSP can be reduced by sending query messages via the CA, which staples the validation information to the signed message en route. This is particularly useful in interplanetary networks, where all messages between segments must travel via one of a small number of relay nodes – these can act as a CA and staple certificates to messages as they pass through.

C.4 CRL



CAs maintain a Certificate Revocation List (CRL), which is sent to any nodes on request. On receipt of a message, if a node does not have an up-to-date CRL from the CA corresponding to the sender's certificate, it requests it, and processes the original message on receipt. Although CRLs are larger than OCSP responses, the advantage is that they cover a much larger number of nodes in a single CRL, so fewer need to be sent overall in situations where messages are received from many different nodes.

C.5 CRL Broadcast



Finally, the CRL broadcast configuration sends the latest CRL to all subscribed nodes at regular intervals, instead of using a request system.

References

[1] N2YO.com. *Browse Satellites by Launch Date*. 2025. URL: https://www.n2yo.com/browse/ (visited on 06/11/2025).

[2] Edouard Mathieu, Pablo Rosado, and Max Roser. "Space Exploration and Satellites". In: *Our World in Data* (2022). URL: https://ourworldindata.org/space-exploration-satellites (visited on 06/11/2025).

[3] Novaspace. *Space Economy Report, 11th Edition*. 2025. URL: https://nova.space/hub/product/space-economy-report/ (visited on 06/11/2025).

[4] Charlotte Van Camp and Walter Peeters. "A World without Satellite Data as a Result of a Global Cyber-Attack". In: *Space Policy* 59 (2022), p. 101458.

[5] Military + Aerospace Electronics. *GPS Jamming amid Wars Playing Havoc with Airline Navigation – Report*. 2024. URL: https://www.militaryaerospace.com/commercial-aerospace/article/55234380/gps-jamming-israel-ukraine (visited on 06/11/2025).

[6] "Russia, in New Push, Increasingly Disrupts Ukraine's Starlink Service". In: *The New York Times* (2024). URL: https://www.nytimes.com/2024/05/24/technology/ukraine-russia-starlink.html (visited on 06/11/2025).

[7] "Cyber Attack on TV Channel BabyTV: Toddlers Suddenly Exposed to Russian Propaganda". In: *NL Times* (2024). URL: https://nltimes.nl/2024/04/06/cyber-attack-tv-channel-babytv-toddlers-suddenly-exposed-russian-propaganda (visited on 06/11/2025).

[8] Mary Pat Flaherty, Jason Samenow, and Lisa Rein. "Chinese hack U.S. weather systems, satellite network". In: *The Washington Post* (2014). URL: https://www.washingtonpost.com/local/chinese-hack-us-weather-systems-satellite-network/2014/11/12/bef1206a-68e9-11e4-b053-65cea7903f2e_story.html (visited on 06/11/2025).

[9] Edd Salkield, Sebastian Köhler, Simon Birnbach, Richard Baker, Martin Strohmeier, and Ivan Martinovic. "Firefly: Spoofing Earth Observation Satellite Data through Radio Overshadowing". In: *Workshop on Security of Space and Satellite Systems (SpaceSec)*. 2023.

[10] sam210723. *COMS-1 LRIT Key Decryption*. 2018. URL: https://vksdr.com/lrit-key-dec (visited on 06/11/2025).

[11] sam210723. *Receiving Images from Geostationary Weather Satellite GEO-KOMPSAT-2A*. 2020. URL: https://vksdr.com/xrit-rx (visited on 06/11/2025).

[12] David J Israel, Kendall D Mauldin, Christopher J Roberts, Jason W Mitchell, Antti A Pulkkinen, D Cooper La Vida, Michael A Johnson, Steven D Christe, and Cheryl J Gramling. "LunaNet: A Flexible and Extensible Lunar Exploration Communications and Navigation Infrastructure". In: *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–14.

[13] NASA. *LunaNet Interoperability Specification Document*. 2022. URL: `https://www3.nasa.gov/sites/default/files/atoms/files/lunanet_interoperability_specification_version_4.pdf` (visited on 06/11/2025).

[14] European Space Agency. *Moonlight*. 2024. URL: `https://www.esa.int/Applications/Connectivity_and_Secure_Communications/Moonlight` (visited on 06/11/2025).

[15] CCSDS. *Overview of Space Communications Protocols*. 2014.

[16] Joshua Smailes, Sebastian Köhler, Simon Birnbach, Martin Strohmeier, and Ivan Martinovic. "Watch This Space: Securing Satellite Communication through Resilient Transmitter Fingerprinting". In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. CCS '23. Copenhagen, Denmark: Association for Computing Machinery, 2023, pp. 608–621.

[17] Joshua Smailes, Sebastian Köhler, Simon Birnbach, Martin Strohmeier, and Ivan Martinovic. "SatIQ: Extensible and Stable Satellite Authentication using Hardware Fingerprinting". In: *ACM Transactions on Privacy and Security* 29.1 (Nov. 2025).

[18] Joshua Smailes, Edd Salkield, Sebastian Köhler, Simon Birnbach, Martin Strohmeier, and Ivan Martinovic. "Sticky Fingers: Resilience of Satellite Fingerprinting against Jamming Attacks". In: *Workshop on Security of Space and Satellite Systems (SpaceSec)*. 2024.

[19] Joshua Smailes, Sebastian Köhler, Simon Birnbach, Martin Strohmeier, and Ivan Martinovic. "SATversary: Adversarial Attacks on Satellite Fingerprinting". In: *arXiv preprint arXiv:2506.06119* (2025). arXiv: `2506.06119`.

[20] Joshua Smailes, Sebastian Köhler, Simon Birnbach, Martin Strohmeier, and Ivan Martinovic. "KeySpace: Public Key Infrastructure Considerations in Interplanetary Networks". In: *arXiv preprint arXiv:2408.10963* (2024). arXiv: `2408.10963`.

[21] Joshua Smailes, Filip Futera, Sebastian Köhler, Simon Birnbach, Martin Strohmeier, and Ivan Martinovic. "DSNS: The Deep Space Network Simulator". In: *2025 Security for Space Systems (3S)*. IEEE. 2025.

[22] Joshua Smailes, Razvan David, Sebastian Köhler, Simon Birnbach, and Ivan Martinovic. "POSTER: spaceQUIC: Securing Communication in Computationally Constrained Spacecraft". In: *arXiv preprint arXiv:2305.12948* (2023). arXiv: `2305.12948`.

[23] Joshua Smailes, Edd Salkield, Sebastian Köhler, Simon Birnbach, and Ivan Martinovic. "Dishing out DoS: How to Disable and Secure the Starlink User Terminal". In: *arXiv preprint arXiv:2303.00582* (2023). arXiv: `2303.00582`.

[24] Edd Salkield, Marcell Szakály, Joshua Smailes, Sebastian Köhler, Simon Birnbach, Martin Strohmeier, and Ivan Martinovic. "Satellite Spoofing from A to Z: On the Requirements of Satellite Downlink Overshadowing Attacks". In: *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2023, pp. 341–352.

[25] Cédric Solenthaler, Joshua Smailes, and Martin Strohmeier. "OrbID: Identifying Orbcomm Satellite RF Fingerprints". In: *Workshop on Security of Space and Satellite Systems (SpaceSec)*. 2025.

[26] Simon Birnbach, Joshua Smailes, Richard Baker, and Ivan Martinovic. "Adaptable Hardware Fingerprinting for Radio Data Links and Avionics Buses in Adversarial Settings". In: *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*. IEEE, 2023, pp. 1–10.

[27] The CubeSat Program. *Cubesat Design Specification (Rev 14.1)*. 2022.

[28] Michael Swartwout. "Cubesats/Smallsats/Nanosats/Picosats/Rideshare(Sats) in 2022: Making Sense of the Numbers". In: *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022, pp. 1–10.

[29] Andrew Turner. "Constellation Design Using Walker Patterns". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. 2002, p. 4636.

[30] GPS.gov. *Space Segment*. 2025. URL: `https://www.gps.gov/systems/gps/space/` (visited on 06/11/2025).

[31] Dan Veeneman. *Iridium: Technical Details*. 2021. URL: `http://www.decodesystems.com/iridium.html` (visited on 06/11/2025).

[32] Tereza Pultarova. "Starlink Satellites: Facts, Tracking, and Impact on Astronomy". In: *Space.com* (2025). URL: `https://www.space.com/spacex-starlink-satellites.html` (visited on 06/11/2025).

[33] NASA Science. *The Mars Relay Network Connects Us to NASA's Martian Explorers*. 2021. URL: `https://mars.nasa.gov/news/8861/the-mars-relay-network-connects-us-to-nasas-martian-explorers` (visited on 06/11/2025).

[34] Kevin Fall. "A Delay-Tolerant Network Architecture for Challenged Internets". In: *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 2003, pp. 27–34.

[35] Amazon Web Services. *AWS Ground Station*. Amazon Web Services, Inc. 2023. URL: `https://aws.amazon.com/ground-station/` (visited on 06/11/2025).

[36] Viasat. *Viasat's Real-Time Earth (RTE) Satellite Ground Segment Service*. 2025. URL: `https://www.viasat.com/government/antenna-systems/real-time-earth/` (visited on 06/11/2025).

[37] Heather Monaghan. *Tracking and Data Relay Satellite (TDRS)*. NASA. Apr. 28, 2015. URL: `http://www.nasa.gov/directorates/heo/scan/services/networks/tdrs_main` (visited on 06/11/2025).

[38] European Space Agency. *European Data Relay Satellite System (EDRS) Overview*. 2022. URL: https://connectivity.esa.int/european-data-relay-satellite-system-edrs-overview (visited on 06/11/2025).

[39] Planet Labs. *Planet Labs Specifications: Spacecraft Operations & Ground Systems*. 2015.

[40] Maxar Technologies. *Spacecraft Platforms*. 2025. URL: https://www.maxar.com/maxar-space-systems/products/spacecraft-platforms (visited on 06/11/2025).

[41] EOS Data Analytics. *Satellite Monitoring for NGOs and NPOs*. 2025. URL: https://eos.com/products/crop-monitoring/non-profits/ (visited on 06/11/2025).

[42] World Population Review. *Military Satellites by Country 2025*. 2025. URL: https://worldpopulationreview.com/country-rankings/military-satellite-by-country (visited on 06/11/2025).

[43] Matthew Mowthorpe. *The Russian Space Threat and a Defense Against It with Guardian Satellites*. 2022. URL: https://www.thespacereview.com/article/4401/1 (visited on 06/11/2025).

[44] Gunter D. Krebs. *Lume 1*. 2018. URL: https://space.skyrocket.de/doc_sdat/lume-1.htm (visited on 06/11/2025).

[45] Gunter D. Krebs. *KSAT (Hayato)*. 2010. URL: https://space.skyrocket.de/doc_sdat/ksat.htm (visited on 06/11/2025).

[46] Gunter D. Krebs. *O/OREOS*. 2010. URL: https://space.skyrocket.de/doc_sdat/ooreos.htm (visited on 06/11/2025).

[47] Gunter D. Krebs. *INVADER (ARTSAT 1, CO 77, CubeSat-OSCAR 77)*. 2014. URL: https://space.skyrocket.de/doc_sdat/invader.htm (visited on 06/11/2025).

[48] Wikipedia Contributors. "List of CubeSats". In: *Wikipedia, The Free Encyclopedia* (2025). URL: https://en.wikipedia.org/wiki/List_of_CubeSats (visited on 06/11/2025).

[49] European Space Agency. *OPS-SAT*. 2024. URL: https://www.esa.int/Enabling_Support/Operations/OPS-SAT (visited on 06/11/2025).

[50] European Space Agency. *Call for Ideas: Cybersecurity Experiments in Orbit*. 2025. URL: https://security4space.esa.int/2025/cfi-cybersecurity-experiments-in-orbit/ (visited on 06/11/2025).

[51] Hack-A-Sat. *World's First CTF in Space*. 2023. URL: https://hackasat.com/ (visited on 06/11/2025).

[52] SatNOGS. *SatNOGS: Open Source Global Network of Satellite Ground-Stations*. 2022. URL: https://satnogs.org/ (visited on 06/11/2025).

[53] NASA. *The Core Flight System (cFS)*. 2025. URL: https://github.com/nasa/cFS (visited on 06/11/2025).

[54] NASA. *CryptoLib*. 2025. URL: https://github.com/nasa/CryptoLib (visited on 06/11/2025).

[55] United States Department of Agriculture. *Satellite Imagery*. URL: `https://ipad.fas.usda.gov/remote.htm` (visited on 06/11/2025).

[56] Claude E Shannon. "Communication in the Presence of Noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[57] Marc Lichtman. "IQ Sampling". In: *PySDR: A Guide to SDR and DSP Using Python*. 2021.

[58] Arne Svensson. "An Introduction to Adaptive QAM Modulation Schemes for Known and Predicted Channels". In: *Proceedings of the IEEE* 95.12 (2007), pp. 2322–2336.

[59] Great Scott Gadgets. *HackRF One*. 2021. URL: `https://greatscottgadgets.com/hackrf/one/` (visited on 06/11/2025).

[60] James Pavur, Daniel Moser, Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. "A Tale of Sea and Sky On the Security of Maritime VSAT Communications". In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 1384–1400.

[61] Benedikt Driessen. "Eavesdropping on Satellite Telecommunication Systems". In: *Cryptology EPrint Archive* (2012).

[62] NASA. *NASA's Earth Observing System: Project Science Office*. 2024. URL: `https://eospso.gsfc.nasa.gov/` (visited on 06/11/2025).

[63] Hemani Kaushal and Georges Kaddoum. "Optical Communication in Space: Challenges and Mitigation Techniques". In: *IEEE Communications Surveys & Tutorials* 19.1 (2016), pp. 57–96.

[64] Daniel Moser, Vincent Lenders, and Srdjan Capkun. "Digital Radio Signal Cancellation Attacks: An Experimental Evaluation". In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. 2019, pp. 23–33.

[65] Marc Lichtman and Jeffrey H Reed. "Analysis of Reactive Jamming against Satellite Communications". In: *International Journal of Satellite Communications and Networking* 34.2 (2016), pp. 195–210.

[66] SaiDhiraj Amuru and R. Michael Buehrer. "Optimal Jamming Against Digital Modulation". In: *IEEE Transactions on Information Forensics and Security* 10.10 (2015), pp. 2212–2224.

[67] Edd Salkield, Sebastian Köhler, Simon Birnbach, and Ivan Martinovic. "SpaceJam: Protocol-aware Jamming Attacks against Space Communications". In: *Proceedings of the 18th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec '25. Association for Computing Machinery, 2025.

[68] Andrea J Goldsmith and S-G Chua. "Adaptive Coded Modulation for Fading Channels". In: *IEEE Transactions on communications* 46.5 (1998), pp. 595–602.

[69] Edd Salkield, Sebastian Köhler, Simon Birnbach, and Ivan Martinovic. "Security Risks of Adaptive Coding and Modulation in Space Systems". In: *2024 Security for Space Systems (3S)*. IEEE, 2024, pp. 1–10.

[70] Mark L Psiaki and Todd E Humphreys. "GNSS Spoofing and Detection". In: *Proceedings of the IEEE* 104.6 (2016), pp. 1258–1270.

[71] NASA. *X-Band Direct Readout Sites Worldwide*. 2022. URL: https://directreadout.sci.gsfc.nasa.gov/?id=dspContent%5C&cid=78 (visited on 06/11/2025).

[72] Eric Jedermann, Martin Strohmeier, Matthias Schäfer, Jens Schmitt, and Vincent Lenders. "Orbit-based Authentication using TDOA Signatures in Satellite Networks". In: *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2021, pp. 175–180.

[73] Gildas Avoine, Muhammed Ali Bingöl, Ioana Boureanu, Srdjan Čapkun, Gerhard Hancke, Süleyman Kardaş, Chong Hee Kim, Cédric Lauradoux, Benjamin Martin, and Jorge Munilla. "Security of Distance-Bounding: A Survey". In: *ACM Computing Surveys (CSUR)* 51.5 (2018), pp. 1–33.

[74] Mohsen Riahi Manesh, Jonathan Kenney, Wen Chen Hu, Vijaya Kumar Devabhaktuni, and Naima Kaabouch. "Detection of GPS Spoofing Attacks on Unmanned Aerial Systems". In: *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2019, pp. 1–6.

[75] Damian Miralles, Aurelie Bornot, Paul Rouquette, Nathan Levigne, Dennis M Akos, Yu-Hsuan Chen, Sherman Lo, and Todd Walter. "An Assessment of GPS Spoofing Detection Via Radio Power and Signal Quality Monitoring for Aviation Safety Operations". In: *IEEE Intelligent Transportation Systems Magazine* 12.3 (2020), pp. 136–146.

[76] Xiao Wei and Biplab Sikdar. "Impact of GPS Time Spoofing Attacks on Cyber Physical Systems". In: *2019 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2019, pp. 1155–1160.

[77] Maryam Motallebighomi, Harshad Sathaye, Mridula Singh, and Aanjhan Ranganathan. "Cryptography Is Not Enough: Relay Attacks on Authenticated GNSS Signals". In: *arXiv preprint arXiv:2204.11641* (2022). arXiv: 2204.11641.

[78] The Associated Press. "Video Pirate Interrupts HBO". In: *The New York Times* (1986). URL: https://www.nytimes.com/1986/04/28/arts/video-pirate-interrupts-hbo.html (visited on 06/11/2025).

[79] Inquirer Wire Services. "Bogus 'Max Headroom' Interrupts Broadcasts On 2 Chicago Stations". In: *Philly.Com* (1987). URL: https://web.archive.org/web/20160909140638/http://articles.philly.com/1987-11-24/news/26174863_1_max-headroom-video-pirate-broadcasts (visited on 06/11/2025).

[80] James Pavur, Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. "In the Same Boat: On Small Satellites, Big Rockets, and Cyber Trust". In: *2021 13th International Conference on Cyber Conflict (CyCon)*. IEEE, 2021, pp. 151–169.

[81] Pieter Maene, Johannes Götzfried, Ruan De Clercq, Tilo Müller, Felix Freiling, and Ingrid Verbauwhede. "Hardware-Based Trusted Computing Architectures for Isolation and Attestation". In: *IEEE Transactions on Computers* 67.3 (2017), pp. 361–374.

[82] Gabriel Honrada. "Russia's Kosmos-2558 May Hunt and Kill US Spy Satellites". In: *Asia Times* (2022). URL: https://asiatimes.com/2022/08/russias-kosmos-2558-may-hunt-and-kill-us-spy-satellites/ (visited on 06/11/2025).

[83] "Case Study: Viasat". In: *CyberPeace Institute* (2022). URL: https://cyberconflicts.cyberpeaceinstitute.org/law-and-policy/cases/viasat (visited on 06/11/2025).

[84] Gustavus J Simmons. "Symmetric and Asymmetric Encryption". In: *ACM Computing Surveys (CSUR)* 11.4 (1979), pp. 305–330.

[85] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Request for Comments RFC 5280. Internet Engineering Task Force, May 2008. URL: https://datatracker.ietf.org/doc/rfc5280 (visited on 06/11/2025).

[86] Stefan Santesson, Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. Request for Comments RFC 6960. Internet Engineering Task Force, June 2013. URL: https://datatracker.ietf.org/doc/rfc6960 (visited on 06/11/2025).

[87] Donald E. Eastlake 3rd. *Transport Layer Security (TLS) Extensions: Extension Definitions*. Request for Comments RFC 6066. Internet Engineering Task Force, Jan. 2011. URL: https://datatracker.ietf.org/doc/rfc6066 (visited on 06/11/2025).

[88] Fred Templin. *Delay Tolerant Networking Security Key Management - Problem Statement*. Internet Draft draft-templin-dtnskmps-00. Internet Engineering Task Force, Mar. 12, 2014. 6 pp. URL: https://datatracker.ietf.org/doc/draft-templin-dtnskmps-00 (visited on 06/11/2025).

[89] Rohit Khare and Adam Rifkin. "Weaving a Web of Trust". In: *World Wide Web Journal* 2.3 (1997), pp. 77–112.

[90] Simson Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly Media, Inc., 1995.

[91] Laurent Eschenauer and Virgil D Gligor. "A Key-Management Scheme for Distributed Sensor Networks". In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*. 2002, pp. 41–47.

[92] Sofia Anna Menesidou, Vasilios Katos, and Georgios Kambourakis. "Cryptographic Key Management in Delay Tolerant Networks: A Survey". In: *Future Internet* 9.3 (2017), p. 26.

[93] Diogo De Andrade and Luiz Carlos Pessoa Albini. "Fully Distributed Public Key Management through Digital Signature Chains for Delay and Disrupt Tolerant Networks". In: *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2016, pp. 316–324.

[94] Chris I Djamaludin, Ernest Foo, and Peter Corke. "Establishing Initial Trust in Autonomous Delay Tolerant Networks without Centralised PKI". In: *Computers & Security* 39 (2013), pp. 299–314.

[95] Jason Anderson, Sherman Lo, Andrew Neish, and Todd Walter. "Authentication of Satellite-Based Augmentation Systems with Over-the-Air Rekeying Schemes". In: *NAVIGATION: Journal of the Institute of Navigation* 70.3 (2023).

[96] CCSDS. *Space Data Link Security Protocol—Extended Procedures*. 2020.

[97] George F Riley and Thomas R Henderson. "The Ns-3 Network Simulator". In: *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 15–34.

[98] Jani Puttonen, Budiarto Herman, Sami Rantanen, Frans Laakso, and Janne Kurjenniemi. "Satellite Network Simulator 3". In: *Workshop on Simulation for European Space Programmes (SESP)*. Vol. 24. 2015, p. 26.

[99] Simon Kassing, Debopam Bhattacherjee, André Baptista Águas, Jens Eirik Saethre, and Ankit Singla. "Exploring the "Internet from Space" with Hypatia". In: *Proceedings of the ACM Internet Measurement Conference*. 2020, pp. 214–229.

[100] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. "The ONE Simulator for DTN Protocol Evaluation". In: *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. 2009, pp. 1–10.

[101] Andras Varga. "OMNeT++". In: *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 35–59.

[102] NASA Technology Transfer Platform. *NASA Operational Simulator for Small Satellites (NOS^3)*. 2025. URL: `https://software.nasa.gov/software/GSC-17737-1` (visited on 06/11/2025).

[103] Daniel Fischer, Mariella Spada, and David Koisser. "SpaceSecLab: A Modular Environment for Prototyping Space-Link Security Protocols". In: *14th International Conference on Space Operations*. 2016, p. 2391.

[104] Anu Jagannath, Jithin Jagannath, and Prem Sagar Pattanshetty Vasanth Kumar. "A Comprehensive Survey on Radio Frequency (RF) Fingerprinting: Traditional Approaches, Deep Learning, and Open Challenges". In: *Computer Networks* 219 (2022), p. 109455.

[105] Naeimeh Soltanieh, Yaser Norouzi, Yang Yang, and Nemai Chandra Karmakar. "A Review of Radio Frequency Fingerprinting Techniques". In: *IEEE Journal of Radio Frequency Identification* 4.3 (2020), pp. 222–233.

[106] Crystal Bertoncini, Kevin Rudd, Bryan Nousain, and Mark Hinders. "Wavelet Fingerprinting of Radio-Frequency Identification (RFID) Tags". In: *IEEE transactions on industrial electronics* 59.12 (2011), pp. 4843–4850.

[107] M Barbeau, J Hall, and E Kranakis. "Detection of Rogue Devices in Bluetooth Networks Using Radio Frequency Fingerprinting". In: *Proceedings of the 3rd IASTED International Conference on Communications and Computer Networks (CCN)*. 2006, pp. 4–6.

[108] William C Suski II, Michael A Temple, Michael J Mendenhall, and Robert F Mills. "Using Spectral Fingerprints to Improve Wireless Network Security". In: *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE, 2008, pp. 1–5.

[109] Jeyanthi Hall, Michel Barbeau, and Evangelos Kranakis. "Detection of Transient in Radio Frequency Fingerprinting Using Signal Phase". In: *Wireless and Optical Communications* (2003), pp. 13–18.

[110] Lianfen Huang, Minghui Gao, Caidan Zhao, and Xiongpeng Wu. "Detection of Wi-Fi Transmitter Transients Using Statistical Method". In: *IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC)*. IEEE, 2013, pp. 1–5.

[111] K. J. Ellis and N. Serinken. "Characteristics of Radio Transmitter Fingerprints". In: *Radio Science* 36.4 (2001), pp. 585–597.

[112] Kasper Bonne Rasmussen and Srdjan Capkun. "Implications of Radio Fingerprinting on the Security of Sensor Networks". In: *Third International Conference on Security and Privacy in Communications Networks (SecureComm)*. IEEE, 2007, pp. 331–340.

[113] Mahsa Foruhandeh, Abdullah Z. Mohammed, Gregor Kildow, Paul Berges, and Ryan Gerdes. "Spotr: GPS Spoofing Detection via Device Fingerprinting". In: *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec '20. Association for Computing Machinery, 2020, pp. 242–253.

[114] Weidong Wang and Lu Gan. "Radio Frequency Fingerprinting Improved by Statistical Noise Reduction". In: *IEEE Transactions on Cognitive Communications and Networking* (2022).

[115] Gabriele Oligeri, Savio Sciancalepore, Simone Raponi, and Roberto Di Pietro. "PAST-AI: Physical-layer Authentication of Satellite Transmitters via Deep Learning". In: *IEEE Transactions on Information Forensics and Security* 18 (2022), pp. 274–289.

[116] Irwin O Kennedy, Patricia Scanlon, Francis J Mullany, Milind M Buddhikot, Keith E Nolan, and Thomas W Rondeau. "Radio Transmitter Fingerprinting: A Steady State Frequency Domain Approach". In: *68th Vehicular Technology Conference*. IEEE, 2008, pp. 1–5.

[117] Joshua Bassey, Damilola Adesina, Xiangfang Li, Lijun Qian, Alexander Aved, and Timothy Kroecker. "Intrusion Detection for IoT Devices Based on RF Fingerprinting Using Deep Learning". In: *Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2019, pp. 98–104.

[118] Francesco Restuccia, Salvatore D'Oro, Amani Al-Shawabka, Mauro Belgiovine, Luca Angioloni, Stratis Ioannidis, Kaushik Chowdhury, and Tommaso Melodia. "DeepRadioID: Real-time Channel-Resilient Optimization of Deep Learning-Based Radio Fingerprinting Algorithms". In: *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 2019, pp. 51–60.

[119] Kunal Sankhe, Mauro Belgiovine, Fan Zhou, Luca Angioloni, Frank Restuccia, Salvatore D'Oro, Tommaso Melodia, Stratis Ioannidis, and Kaushik Chowdhury. "No Radio Left Behind: Radio Fingerprinting Through Deep Learning of Physical-Layer Hardware Impairments". In: *IEEE Transactions on Cognitive Communications and Networking* 6.1 (2020), pp. 165–178.

[120] Boris Danev, Heinrich Luecken, Srdjan Capkun, and Karim El Defrawy. "Attacks on Physical-Layer Identification". In: *Proceedings of the Third ACM Conference on Wireless Network Security*. 2010, pp. 89–98.

[121] Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. "On the Security of the Automatic Dependent Surveillance-Broadcast Protocol". In: *IEEE Communications Surveys Tutorials* 17.2 (2015), pp. 1066–1087.

[122] Marc Lichtman, Roger Piqueras Jover, Mina Labib, Raghunandan Rao, Vuk Marojevic, and Jeffrey H Reed. "LTE/LTE-A Jamming, Spoofing, and Sniffing: Threat Assessment and Mitigation". In: *IEEE Communications Magazine* 54.4 (2016), pp. 54–61.

[123] Gyuhong Lee, Jihoon Lee, Jinsung Lee, Youngbin Im, Max Hollingsworth, Eric Wustrow, Dirk Grunwald, and Sangtae Ha. "This Is Your President Speaking: Spoofing Alerts in 4G LTE Networks". In: *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. 2019, pp. 404–416.

[124] João Gaspar, Renato Ferreira, Pedro Sebastião, and Nuno Souto. "Capture of UAVs Through GPS Spoofing Using Low-Cost SDR Platforms". In: *Wireless Personal Communications* 115 (2020), pp. 2729–2754.

[125] NASA. *Fire Information for Resource Management System (FIRMS)*. 2022. URL: `https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms` (visited on 06/11/2025).

[126] Esri. *Esri Releases Updated Land-Cover Map with New Sets of Global Data*. 2022. URL: `https://www.esri.com/about/newsroom/announcements/esri-releases-updated-land-cover-map-with-new-sets-of-global-data/` (visited on 06/11/2025).

[127] Meta. *High Resolution Population Density Maps*. 2022. URL: `https://dataforgood.facebook.com/dfg/tools/high-resolution-population-density-maps` (visited on 06/11/2025).

[128] Cloud to Street. *Cloud to Street*. 2022. URL: `https://www.cloudtostreet.ai/` (visited on 06/11/2025).

[129] ÖH Tekbaş, Oktay Üreten, and Nur Serinken. "Improvement of Transmitter Identification System for Low SNR Transients". In: *Electronics Letters* 40.3 (2004), pp. 182–183.

[130] European Space Agency. *Applications for Radio Frequency Fingerprinting in Satellite Communications (ARTES FP 1B.146)*. 2025. URL: `https://esastar-publication-ext.sso.esa.int/ESATenderActions/details/140407` (visited on 06/11/2025).

[131] Expedition Technology. *Expedition Technology, Inc. Awarded DARPA Phase 3 Contract for RF Machine Learning*. 2021. URL: `https://www.exptechinc.com/expedition-technology-awarded-darpa-phase-3-contract-for-rf-machine-learning/` (visited on 06/11/2025).

[132] Jiabao Yu, Aiqun Hu, Fen Zhou, Yuexiu Xing, Yi Yu, Guyue Li, and Linning Peng. "Radio Frequency Fingerprint Identification Based on Denoising Autoencoders". In: *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2019, pp. 1–6.

[133] Joshua Bassey, Xiangfang Li, and Lijun Qian. "Device Authentication Codes Based on RF Fingerprinting Using Deep Learning". In: *arXiv preprint arXiv:2004.08742* (2020). arXiv: `2004.08742`.

[134] Qi Jiang and Jin Sha. "Radio Frequency Fingerprint Identification Based on Variational Autoencoder for GNSS". In: *IEEE Geoscience and Remote Sensing Letters* (2024).

[135] Davide Chicco. "Siamese Neural Networks: An Overview". In: *Artificial Neural Networks* (2021), pp. 73–94.

[136] Jinting Zhu, Julian Jang-Jaccard, and Paul A Watters. "Multi-Loss Siamese Neural Network With Batch Normalization Layer for Malware Detection". In: *IEEE Access* 8 (2020), pp. 171542–171550.

[137] Cheng Zhang, Wu Liu, Huadong Ma, and Huiyuan Fu. "Siamese Neural Network Based Gait Recognition for Human Identification". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2832–2836.

[138] Mingtao Pei, Bin Yan, Huiling Hao, and Meng Zhao. "Person-Specific Face Spoofing Detection Based on a Siamese Network". In: *Pattern Recognition* 135 (2023), p. 109148.

[139] Kaavya Sriskandaraja, Vidhyasaharan Sethu, and Eliathamby Ambikairajah. "Deep Siamese Architecture Based Replay Detection for Secure Voice Biometric". In: *Interspeech*. 2018, pp. 671–675.

[140] Yu Mao, Yang-Yang Dong, Ting Sun, Xian Rao, and Chun-Xi Dong. "Attentive Siamese Networks for Automatic Modulation Classification Based on Multitiming Constellation Diagrams". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[141] Louis Morge-Rollet, Frédéric Le Roy, Denis Le Jeune, and Roland Gautier. "Siamese Network on I/Q Signals for RF Fingerprinting". In: *Actes de La Conférence CAID*. 2020, p. 152.

[142] Zachary Langford, Logan Eisenbeiser, and Matthew Vondal. "Robust Signal Classification Using Siamese Networks". In: *Proceedings of the ACM Workshop on Wireless Security and Machine Learning*. 2019, pp. 1–5.

[143] Kian Ahrabian and Bagher BabaAli. "Usage of Autoencoders and Siamese Networks for Online Handwritten Signature Verification". In: *Neural Computing and Applications* 31 (2019), pp. 9321–9334.

[144] Tobias Schneider and Stefan Zehl. *gr-iridium: GNU Radio Iridium Out Of Tree Module*. Chaos Computer Club München. 2022.

[145] Sarang Narkhede. "Understanding AUC - ROC Curve". In: *Towards Data Science* 26.1 (2018), pp. 220–227.

[146] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2013). arXiv: `1312.6114`.

[147] Bechir Hamdaoui and Abdurrahman Elmaghbub. "Deep-Learning-Based Device Fingerprinting for Increased LoRa-IoT Security: Sensitivity to Network Deployment Changes". In: *IEEE Network* 36.3 (2022), pp. 204–210.

[148] Amani Al-Shawabka, Francesco Restuccia, Salvatore D'Oro, Tong Jian, Bruno Costa Rendon, Nasim Soltani, Jennifer Dy, Stratis Ioannidis, Kaushik Chowdhury, and Tommaso Melodia. "Exposing the Fingerprint: Dissecting the Impact of the Wireless Channel on Radio Fingerprinting". In: *IEEE Conference on Computer Communications (INFOCOM)*. 2020, pp. 646–655.

[149] Visual Crossing Corporation. *Visual Crossing Weather*. 2024. URL: `https://www.visualcrossing.com/` (visited on 06/11/2025).

[150] Jonathan McDowell. *Starlink Statistics*. 2024. URL: `https://planet4589.org/space/con/star/stats.html` (visited on 06/11/2025).

[151] Gabriele Oligeri and Savio Sciancalepore. *Physical Layer Data Acquisition of IRIDIUM Satellites Broadcast Messages*. 2022. URL: `https://data.mendeley.com/datasets/xcxspv8c2r/1` (visited on 06/11/2025).

[152] Bingyin Ren, Hailong Ge, Guangfei Xu, and Yongxin Zhang. "Anti-Jamming Analysis and Application of Starlink System". In: *International Conference on Networking, Informatics and Computing (ICNETIC)*. 2023.

[153] Valerie Insinna. *SpaceX Beating Russian Jamming Attack was 'Eyewatering': DoD Official*. 2022. URL: `https://breakingdefense.com/2022/04/spacex-beating-russian-jamming-attack-was-eyewatering-dod-official/` (visited on 06/11/2025).

[154] Saeed Ur Rehman, Kevin W Sowerby, and Colin Coghill. "Analysis of Impersonation Attacks on Systems Using RF Fingerprinting and Low-End Receivers". In: *Journal of Computer and System Sciences* 80.3 (2014), pp. 591–601.

[155] Matthew Edman and Bülent Yener. "Active Attacks against Modulation-Based Radiometric Identification". In: *Rensselaer Institute of Technology, Technical report* (2009), pp. 09–02.

[156] Damilola Adesina, Chung-Chu Hsieh, Yalin E Sagduyu, and Lijun Qian. "Adversarial Machine Learning in Wireless Communications Using RF Data: A Review". In: *IEEE Communications Surveys & Tutorials* 25.1 (2022), pp. 77–100.

[157] Joshua H Tyler, Mohamed KM Fadul, and Donald R Reising. "Considerations, Advances, and Challenges Associated with the Use of Specific Emitter Identification in the Security of Internet of Things Deployments: A Survey". In: *Information* 14.9 (2023), p. 479.

[158] Brian Kim, Yalin E Sagduyu, Kemal Davaslioglu, Tugba Erpek, and Sennur Ulukus. "Over-the-Air Adversarial Attacks on Deep Learning Based Modulation Classifier over Wireless Channels". In: *2020 54th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2020, pp. 1–6.

[159] Brian Kim, Yalin E Sagduyu, Kemal Davaslioglu, Tugba Erpek, and Sennur Ulukus. "Channel-Aware Adversarial Attacks Against Deep Learning-Based Wireless Signal Classifiers". In: *IEEE Transactions on Wireless Communications* 21.6 (2021), pp. 3868–3880.

[160] Silvija Kokalj-Filipovic, Rob Miller, and Joshua Morman. "Targeted Adversarial Examples Against RF Deep Classifiers". In: *Proceedings of the ACM Workshop on Wireless Security and Machine Learning*. 2019, pp. 6–11.

[161] Yi Shi, Yalin E Sagduyu, Tugba Erpek, Kemal Davaslioglu, Zhuo Lu, and Jason H Li. "Adversarial Deep Learning for Cognitive Radio Security: Jamming Attack and Defense Strategies". In: *IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[162] Francesco Restuccia, Salvatore D'Oro, Amani Al-Shawabka, Bruno Costa Rendon, Kaushik Chowdhury, Stratis Ioannidis, and Tommaso Melodia. "Generalized Wireless Adversarial Deep Learning". In: *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*. 2020, pp. 49–54.

[163] Samurdhi Karunaratne, Enes Krijestorac, and Danijela Cabric. "Penetrating RF Fingerprinting-based Authentication with a Generative Adversarial Attack". In: *IEEE International Conference on Communications (ICC)*. IEEE, 2021, pp. 1–6.

[164] Muhammad Irfan, Savio Sciancalepore, and Gabriele Oligeri. "Preventing Radio Fingerprinting through Friendly Jamming". In: *arXiv preprint arXiv:2407.08311* (2025). arXiv: `2407.08311`.

[165] Liting Sun, Da Ke, Xiang Wang, Zhitao Huang, and Kaizhu Huang. "Robustness of Deep Learning-Based Specific Emitter Identification under Adversarial Attacks". In: *Remote Sensing* 14.19 (2022), p. 4996.

[166] Boyang Liu, Haoran Zhang, Yiyao Wan, Fuhui Zhou, Qihui Wu, and Derrick Wing Kwan Ng. "Robust Adversarial Attacks on Deep Learning Based RF Fingerprint Identification". In: *IEEE Wireless Communications Letters* (2023).

[167] Ildi Alla, Selma Yahia, Valeria Loscri, and Hossien Eldeeb. "Robust Device Authentication in Multi-Node Networks: ML-Assisted Hybrid PLA Exploiting Hardware Impairments". In: *Annual Computer Security Applications Conference (ACSAC)*. 2024.

[168] Gabriel Resende Machado, Eugênio Silva, and Ronaldo Ribeiro Goldschmidt. "Adversarial Machine Learning in Image Classification: A Survey Toward the Defender's Perspective". In: *ACM Computing Surveys (CSUR)* 55.1 (2021), pp. 1–38.

[169] Stefano Marrone and Carlo Sansone. "An Adversarial Perturbation Approach Against CNN-based Soft Biometrics Detection". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[170] Marcos Martinez-Diaz, Julian Fierrez, Javier Galbally, and Javier Ortega-Garcia. "An Evaluation of Indirect Attacks and Countermeasures in Fingerprint Verification Systems". In: *Pattern Recognition Letters* 32.12 (2011), pp. 1643–1651.

[171] Alejandro Gomez-Alanis, Jose A Gonzalez-Lopez, and Antonio M Peinado. "GANBA: Generative Adversarial Network for Biometric Anti-Spoofing". In: *Applied Sciences* 12.3 (2022), p. 1454.

[172] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. "A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning". In: *ACM Computing Surveys* 55.8 (2022), pp. 1–35.

[173] Zhibo Wang, Jingjing Ma, Xue Wang, Jiahui Hu, Zhan Qin, and Kui Ren. "Threats to Training: A Survey of Poisoning Attacks and Defenses on Machine Learning Systems". In: *ACM Computing Surveys* 55.7 (2022), pp. 1–36.

[174] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. "Can Machine Learning Be Secure?" In: *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*. 2006, pp. 16–25.

[175] Giulio Lovisotto, Simon Eberz, and Ivan Martinovic. "Biometric Backdoors: A Poisoning Attack Against Unsupervised Template Updating". In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 184–197.

[176] Battista Biggio, Giorgio Fumera, Paolo Russu, Luca Didaci, and Fabio Roli. "Adversarial Biometric Recognition: A Review on Biometric System Security from the Adversarial Machine-Learning Perspective". In: *IEEE Signal Processing Magazine* 32.5 (2015), pp. 31–41.

[177] Samuel Lefcourt, Nathaniel Gordon, Hanting Wong, and Gregory Falco. "Space Cognitive Communications: Characterizing Radiofrequency Interference to Improve Digital Space Domain Awareness". In: *2022 International Conference on Localization and GNSS (ICL-GNSS)*. IEEE. 2022, pp. 1–7.

[178] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. "Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks". In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 321–338.

[179] Matthew Walters and Sujoy Sinha Roy. "Constant-time BCH Error-correcting Code". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2020, pp. 1–5.

[180] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. "CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2745–2754.

[181] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *arXiv preprint arXiv:1411.1784* (2014). arXiv: `1411.1784`.

[182] Paola Martinelli, Ernestina Cianca, Mauro De Sanctis, Lanfranco Di Paolo, Alessandro Pisano, and Lorenzo Simone. "Robustness of Satellite Telecommand Links to Jamming Attacks". In: *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*. IEEE, 2012, pp. 1–6.

[183] Junyuan Xie, Linli Xu, and Enhong Chen. "Image Denoising and Inpainting with Deep Neural Networks". In: *Advances in Neural Information Processing Systems* 25 (2012).

[184] Aya Saleh Ahmed, Wessam H El-Behaidy, and Aliaa AA Youssif. "Medical Image Denoising System Based on Stacked Convolutional Autoencoder for Enhancing 2-Dimensional Gel Electrophoresis Noise Reduction". In: *Biomedical Signal Processing and Control* 69 (2021), p. 102842.

[185] Debjani Bhowick, Deepak K Gupta, Saumen Maiti, and Uma Shankar. "Stacked Autoencoders Based Machine Learning for Noise Reduction and Signal Reconstruction in Geophysical Data". In: *arXiv preprint arXiv:1907.03278* (2019). arXiv: 1907.03278.

[186] Guoqing Jin, Shiwei Shen, Dongming Zhang, Feng Dai, and Yongdong Zhang. "APE-GAN: Adversarial Perturbation Elimination with GAN". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3842–3846.

[187] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *arXiv preprint arXiv:1706.06083* (2017). arXiv: 1706.06083.

[188] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks". In: *2016 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2016, pp. 582–597.

[189] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified Adversarial Robustness via Randomized Smoothing". In: *International Conference on Machine Learning*. PMLR, 2019, pp. 1310–1320.

[190] Zhitao Gong and Wenlu Wang. "Adversarial and Clean Data Are Not Twins". In: *Proceedings of the Sixth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. 2023, pp. 1–5.

[191] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. "On the (Statistical) Detection of Adversarial Examples". In: *arXiv preprint arXiv:1702.06280* (2017). arXiv: 1702.06280.

[192] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. "On Detecting Adversarial Perturbations". In: *arXiv preprint arXiv:1702.04267* (2017). arXiv: 1702.04267.

[193] Silvija Kokalj-Filipovic, Rob Miller, Nicholas Chang, and Chi Laung Lau. "Mitigation of Adversarial Examples in RF Deep Classifiers Utilizing Autoencoder Pre-training". In: *2019 International Conference on Military Communications and Information Systems (ICMCIS)*. IEEE, 2019, pp. 1–6.

[194] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. "Ensemble Adversarial Training: Attacks and Defenses". In: *arXiv preprint arXiv:1705.07204* (2017). arXiv: 1705.07204.

[195] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified Defenses against Adversarial Examples". In: *arXiv preprint arXiv:1801.09344* (2018). arXiv: 1801.09344.

[196] Giuseppe Araniti, Antonio Iera, Antonella Molinaro, Sara Pizzi, and Federica Rinaldi. "Opportunistic Federation of CubeSat Constellations: A Game-Changing Paradigm Enabling Enhanced IoT Services in the Sky". In: *IEEE Internet of Things Journal* (2021).

[197] Ignasi Lluch, Paul T Grogan, Udrivolf Pica, and Alessandro Golkar. "Simulating a Proactive Ad-Hoc Network Protocol for Federated Satellite Systems". In: *2015 IEEE Aerospace Conference*. IEEE, 2015, pp. 1–16.

[198] Joan A Ruiz-de-Azua, Lara Fernandez, Joan F Muñoz, Marc Badia, Ricard Castella, Carlos Diez, Andrea Aguilella, Simone Briatore, Nicola Garzaniti, and Anna Calveras. "Proof-of-Concept of a Federated Satellite System between Two 6-Unit CubeSats for Distributed Earth Observation Satellite Systems". In: *2019 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2019, pp. 8871–8874.

[199] Joan A Ruiz-De-Azua, Anna Calveras, and Adriano Camps. "A Novel Dissemination Protocol to Deploy Opportunistic Services in Federated Satellite Systems". In: *IEEE Access* 8 (2020), pp. 142348–142365.

[200] Joan A Ruiz-de-Azua, Nicola Garzaniti, Alessandro Golkar, Anna Calveras, and Adriano Camps. "Towards Federated Satellite Systems and Internet of Satellites: The Federation Deployment Control Protocol". In: *Remote Sensing* 13.5 (2021), p. 982.

[201] Scott Burleigh, Kevin Fall, and Edward J. Birrane. *Bundle Protocol Version 7*. Request for Comments RFC 9171. Internet Engineering Task Force, Jan. 2022. 53 pp. URL: `https://datatracker.ietf.org/doc/rfc9171` (visited on 06/11/2025).

[202] Edward J. Birrane and Kenneth McKeever. *Bundle Protocol Security (BPSec)*. Request for Comments RFC 9172. Internet Engineering Task Force, Jan. 2022. 35 pp. URL: `https://datatracker.ietf.org/doc/rfc9172` (visited on 06/11/2025).

[203] Stephen Farrell, Susan Symington, Howard Weiss, and Peter Lovell. *Delay-Tolerant Networking Security Overview*. Internet Draft draft-irtf-dtnrg-sec-overview-06. Internet Engineering Task Force, Mar. 8, 2009. 29 pp. URL: `https://datatracker.ietf.org/doc/draft-irtf-dtnrg-sec-overview-06` (visited on 06/11/2025).

[204] Stephen Farrell. *DTN Key Management Requirements*. Internet Draft draft-farrell-dtnrg-km-00. Internet Engineering Task Force, June 19, 2007. 9 pp. URL: `https://datatracker.ietf.org/doc/draft-farrell-dtnrg-km-00` (visited on 06/11/2025).

[205] Fred Templin and Scott C. Burleigh. *DTN Security Key Management - Requirements and Design*. Internet Draft draft-templin-dtn-dtnskmreq-00. Internet Engineering Task Force, Apr. 6, 2016. 12 pp. URL: `https://datatracker.ietf.org/doc/draft-templin-dtn-dtnskmreq-00` (visited on 06/11/2025).

[206] CCSDS. *Space Missions Key Management Concept*. 2011.

[207] CCSDS. *Intergovernmental Certification Authority*. 2024.

[208] Kapali Viswanathan and Fred Templin. *Architecture for a Delay-and-Disruption Tolerant Public-Key Distribution Network (PKDN)*. Internet Draft draft-viswanathan-dtn-pkdn-00. Internet Engineering Task Force, Mar. 8, 2016. 18 pp. URL: `https://datatracker.ietf.org/doc/draft-viswanathan-dtn-pkdn-00` (visited on 06/11/2025).

[209] Sofia Anna Menesidou and Vasilios Katos. "Authenticated Key Exchange (AKE) in Delay Tolerant Networks". In: *27th IFIP TC 11 Information Security and Privacy Conference (SEC)*. Springer, 2012, pp. 49–60.

[210] N Bhutta, G Ansa, E Johnson, N Ahmad, M Alsiyabi, and Haitham Cruickshank. "Security Analysis for Delay/Disruption Tolerant Satellite and Sensor Networks". In: *2009 International Workshop on Satellite and Space Communications*. IEEE, 2009, pp. 385–389.

[211] Olga von Maurich and Alessandro Golkar. "Data Authentication, Integrity and Confidentiality Mechanisms for Federated Satellite Systems". In: *Acta Astronautica* 149 (2018), pp. 61–76.

[212] Enyenihi Johnson, Haitham Cruickshank, and Zhili Sun. "Providing Authentication in Delay/Disruption Tolerant Networking (DTN) Environment". In: *International Conference on Personal Satellite Services (PSATS)*. Springer, 2013, pp. 189–196.

[213] Aaditeshwar Seth and Srinivasan Keshav. "Practical Security for Disconnected Nodes". In: *1st IEEE ICNP Workshop on Secure Network Protocols (NPSec)*. IEEE, 2005, pp. 31–36.

[214] N Asokan, Kari Kostiainen, Philip Ginzboorg, Jorg Ott, and Cheng Luo. "Towards Securing Disruption-Tolerant Networking". In: *Nokia Research Center, Tech. Rep. NRC-TR-2007-007* (2007).

[215] David A Cooper. "A More Efficient Use of Delta-CRLs". In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2000, pp. 190–202.

[216] Muhammad Nasir Mumtaz Bhutta, Haitham Cruickshank, and Zhili Sun. "Public-key Infrastructure Validation and Revocation Mechanism Suitable for Delay/Disruption Tolerant Networks". In: *IET Information Security* 11.1 (2017), pp. 16–22.

[217] David Koisser, Daniel Fischer, Marcus Wallum, and Ahmad-Reza Sadeghi. "TruSat: Building Cyber Trust in Collaborative Spacecraft Networks". In: *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022, pp. 1–12.

[218] Jun Luo, J-P Hubaux, and Patrick Th Eugster. "DICTATE: DIstributed CerTification Authority with probabilisTic frEshness for Ad Hoc Networks". In: *IEEE Transactions on Dependable and Secure Computing* 2.4 (2005), pp. 311–323.

[219] Ren Fang and Fan Jiulun. "An Adaptive Distributed Certificate Management Scheme for Space Information Network". In: *IET Information Security* 7.4 (2013), pp. 318–326.

[220] Chris I Djamaludin, Ernest Foo, Seyit Camtepe, and Peter Corke. "Revocation and Update of Trust in Autonomous Delay Tolerant Networks". In: *Computers & Security* 60 (2016), pp. 15–36.

[221] Pawel Szalachowski, Laurent Chuat, Taeho Lee, and Adrian Perrig. "RITM: Revocation in the Middle". In: *36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2016, pp. 189–200.

[222] Stephen R Pratt, Richard A Raines, Carl E Fossa, and Michael A Temple. "An Operational and Performance Overview of the IRIDIUM Low Earth Orbit Satellite System". In: *IEEE Communications Surveys* 2.2 (1999), pp. 2–10.

[223] Deep Space Network. *301 Coverage and Geometry*. 810-005, 301, Rev. N. Apr. 15, 2022.

[224] Ahmad Alhilal, Tristan Braud, and Pan Hui. "The Sky Is NOT the Limit Anymore: Future Architecture of the Interplanetary Internet". In: *IEEE Aerospace and Electronic Systems Magazine* 34.8 (2019), pp. 22–32.

[225] Space Exploration Holdings, LLC. *SpaceX Non-Geostationary Satellite System, Attachment A: Technical Information to Supplement Schedule S*. Federal Communications Commission, 2020.

[226] Wan Choi, Antonio Forenza, Jeffrey G Andrews, and Robert W Heath. "Opportunistic Space-Division Multiple Access with Beam Selection". In: *IEEE Transactions on Communications* 55.12 (2007), pp. 2371–2380.

[227] Ian F Akyildiz, Eylem Ekici, and Michael D Bender. "MLSR: A Novel Routing Algorithm for Multilayered Satellite IP Networks". In: *IEEE/ACM Transactions on Networking* 10.3 (2002), pp. 411–424.

[228] Tasneem Darwish, Gunes Karabulut Kurt, Halim Yanikomeroglu, Michel Bellemare, and Guillaume Lamontagne. "LEO Satellites in 5G and Beyond Networks: A Review From a Standardization Perspective". In: *IEEE Access* 10 (2022), pp. 35040–35060.

[229] Boya Di, Lingyang Song, Yonghui Li, and H Vincent Poor. "Ultra-Dense LEO: Integration of Satellite Access Networks into 5G and Beyond". In: *IEEE Wireless Communications* 26.2 (2019), pp. 62–69.

[230] Starlink. *Starlink: Direct to Cell*. 2023. URL: `https://www.starlink.com/gb/business/direct-to-cell` (visited on 06/11/2025).

[231] Patrick Gannon. *SES Networks (Formerly O3b) Update*. 2023. URL: `https://www.bcsatellite.net/blog/ses-networks-formerly-o3b-update/` (visited on 06/11/2025).

[232] SES. *Insight Paper - O3b mPOWER*. 2022.

[233] Samo Grasic, Elwyn Davies, Anders Lindgren, and Avri Doria. "The Evolution of a DTN Routing Protocol – PRoPHETv2". In: *Proceedings of the 6th ACM Workshop on Challenged Networks*. 2011, pp. 27–30.

[234] Jérémie Leguay, Timur Friedman, and Vania Conan. "DTN Routing in a Mobility Pattern Space". In: *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*. 2005, pp. 276–283.

[235] Giuseppe Araniti, Nikolaos Bezirgiannidis, Edward Birrane, Igor Bisio, Scott Burleigh, Carlo Caini, Marius Feldmann, Mario Marchese, John Segui, and Kiyohisa Suzuki. "Contact Graph Routing in DTN Space Networks: Overview, Enhancements and Performance". In: *IEEE Communications Magazine* 53.3 (2015), pp. 38–46.

[236] CC Sobin, Vaskar Raychoudhury, Gustavo Marfia, and Ankita Singla. "A Survey of Routing and Data Dissemination in Delay Tolerant Networks". In: *Journal of Network and Computer Applications* 67 (2016), pp. 128–146.

[237] QI Xiaogang, MA Jiulong, WU Dan, LIU Lifang, and HU Shaolin. "A Survey of Routing Techniques for Satellite Networks". In: *Journal of Communications and Information Networks* 1.4 (2016), pp. 66–85.

[238] Fatih Alagoz, Omer Korcak, and Abbas Jamalipour. "Exploring the Routing Strategies in Next-Generation Satellite Networks". In: *IEEE Wireless Communications* 14.3 (2007), pp. 79–88.

[239] Jana Iyengar and Martin Thomson. *QUIC: A UDP-Based Multiplexed and Secure Transport*. Request for Comments RFC 9000. Internet Engineering Task Force, 2021. URL: https://datatracker.ietf.org/doc/rfc9000 (visited on 06/11/2025).

[240] Jeremy Rowley. *How Short-Lived Certificates Improve Certificate Trust*. 2016. URL: https://www.digicert.com/blog/short-lived-certificates (visited on 06/11/2025).

[241] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. "DTN Routing as a Resource Allocation Problem". In: *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 2007, pp. 373–384.

[242] Yakov Rekhter, Susan Hares, and Tony Li. *A Border Gateway Protocol 4 (BGP-4)*. Request for Comments RFC 4271. Internet Engineering Task Force, 2006. URL: https://datatracker.ietf.org/doc/rfc4271 (visited on 06/11/2025).

[243] Stefano Vissicchio and Mark Handley. "Reliable Low-Delay Routing in Space with Routing-Oblivious LEO Satellites". In: *arXiv preprint arXiv:2401.11490* (2024). arXiv: 2401.11490.

[244] Doug Madory. *A Brief History of the Internet's Biggest BGP Incidents*. 2025. URL: https://www.kentik.com/blog/a-brief-history-of-the-internets-biggest-bgp-incidents/ (visited on 06/11/2025).

[245] CCSDS. *CCSDS Bundle Protocol Specification*. 2015.

[246] Scott C. Burleigh, Stephen Farrell, and Manikantan Ramadas. *Licklider Transmission Protocol - Specification*. Request for Comments RFC 5326. Internet Engineering Task Force, 2008. URL: https://datatracker.ietf.org/doc/rfc5326 (visited on 06/11/2025).

[247] "IEEE Standard Letter Designations for Radar-Frequency Bands". In: *IEEE Std 521-2019 (Revision of IEEE Std 521-2002)* (2020), pp. 1–15.

[248] European Space Agency. *Satellite Frequency Bands*. 2025. URL: https://www.esa.int/Applications/Connectivity_and_Secure_Communications/Satellite_frequency_bands (visited on 06/11/2025).

[249] SatNOGS. *Statistics*. 2022. URL: https://db.satnogs.org/stats (visited on 06/11/2025).

[250] Andrea M. Mitofsky. "Antenna Characteristics". In: *Direct Energy*. LibreTexts Engineering, 2018. Chap. 4.4. URL: https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electro-Optics/Direct_Energy_(Mitofsky)/04%3A_Antennas/4.04%3A_Antenna_Characteristics (visited on 06/11/2025).

[251] Thomas A Milligan. *Modern Antenna Design*. John Wiley & Sons, 2005.

[252] Suhila Abulgasem, Faisel Tubbal, Raad Raad, Panagiotis Ioannis Theoharis, Sining Lu, and Saeid Iranmanesh. "Antenna Designs for CubeSats: A Review". In: *IEEE Access* 9 (2021), pp. 45289–45324.

[253] Guolong He, Xin Gao, Liangliang Sun, and Rentian Zhang. "A Review of Multibeam Phased Array Antennas as LEO Satellite Constellation Ground Station". In: *IEEE Access* 9 (2021), pp. 147142–147154.

[254] William T Cochran, James W Cooley, David L Favin, Howard D Helms, Reginald A Kaenel, William W Lang, George C Maling, David E Nelson, Charles M Rader, and Peter D Welch. "What Is the Fast Fourier Transform?" In: *Proceedings of the IEEE* 55.10 (1967), pp. 1664–1674.