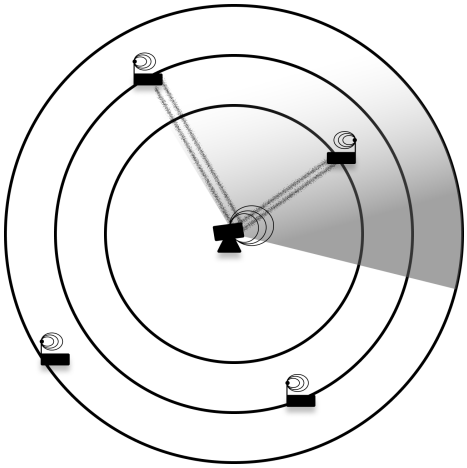
CHALMERS
UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG



ABLA: Asymmetric Backscatter Lightweight Attestation

A Lightweight Attestation Protocol for Radar and Backscatter Transponder Links

Master's thesis in Computer science and engineering

Arthur Alexandersson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

ABLA: Asymmetric Backscatter Lightweight Attestation

A Lightweight Attestation Protocol for Radar and Backscatter Transponder Links

Arthur Alexandersson

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

ABLA: Asymmetric Backscatter Lightweight Attestation A Lightweight Attestation Protocol for Radar and Backscatter Transponder Links
Arthur Alexandersson

© Arthur Alexandersson, 2026.

Supervisor: Md Masoom Rabbani, Chalmers University of Technology, Department of Computer Science and Engineering
Advisor: Johan Bodin and Pontus Johannisson, Saab AB
Examiner: Romaric Duvignau, Chalmers University of Technology, Department of Computer Science and Engineering

Master's Thesis 2026
Chalmers University of Technology, Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Radar and backscatter transponder network where illuminated transponders have an established communication link.

Typeset in LaTeX
Gothenburg, Sweden 2026

ABLA: Asymmetric Backscatter Lightweight Attestation A Lightweight Attestation Protocol for Radar and Backscatter Transponder Links
Arthur Alexandersson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Augmenting existing surveillance systems with complementary devices that extend their capabilities introduces new challenges. Such an auxiliary device typically shares the hostile environment of the system it serves, so it must be authenticated and shown to be running uncompromised firmware before its reports can be trusted. Remote attestation addresses this: a verifier checks a prover's integrity, e.g. through a hash of the software running on the device. This thesis, carried out in collaboration with Saab AB, examines one such case: the integration of a device intended to complement a rotating radar. The device is networked and assumed to be exposed to adversaries capable of eavesdropping, replay, spoofing, jamming, and physical capture. The communication link is, however, severely asymmetric: the downlink carries only tens of bits per second, while the backscatter uplink operates at a few hundred. We propose ABLA: *Asymmetric Backscatter Lightweight Attestation*, a five-phase remote attestation protocol designed for this setting. The protocol uses only symmetric primitives on the device side and combines challenge–response attestation with a radar-based proximity check step that reuses the radar's existing range and bearing measurements to detect relay attacks and physical displacement at no additional cryptographic cost. ABLA's cryptographic security is analyzed symbolically in the Tamarin prover; the model covers the cryptographic core of Phases 3 and 4 only, establishing key secrecy, message authentication, verifier-commit uniqueness per session tuple, and a minimal attestation-soundness property under a Dolev–Yao adversary. The protocol's operational behavior is evaluated through discrete-event simulation in OMNeT++ for deployments of up to 32 transponders.

Keywords: Remote attestation, lightweight attestation, attestation, radar, backscatter, network, security, network security.

Acknowledgements

I would like to extend my sincere thanks to my supervisor at Chalmers University of Technology, Md. Masoom Rabbani, and to my industrial supervisors at Saab AB, M.Sc Johan Bodin and Dr. Pontus Johannisson, for their guidance throughout this thesis. Their combined expertise in remote attestation and radar technology has been invaluable, and this work would not have taken its present shape without it.

I would also like to thank my family and friends for their invaluable support during this work, the work marking the end of five years of studies at Chalmers University of Technology.

Arthur Alexandersson, Gothenburg, 2026-06-30

Contents

List of Figures

List of Tables

Acronyms

f_{PRF} Pulse Repetition Frequency. 8, 9, 30, 48

ASK Amplitude-shift Keying. 12

BASK Binary Amplitude-shift Keying. 12

BC Backscatter Communication. xiii, 1, 2, 5, 10, 11, 13, 21, 28, 36, 48

BER Bit-Error Rate. 57, 61, 65

BPSK Binary Phase-shift Keying. 12

CRA Collective Remote Attestation. 21

DC Duty Cycle. 8, 13

DICE Device Identifier Composition Engine. 18, 19, 29

DMA Direct Memory Access. 25

DoS Denial-of-service. 21, 22, 25, 26

EA-MPU Execution Aware Memory Protection Unit. 24, 25

ECC Elliptic Curve Cryptography. 27

EM Electromagnetic. 5

IFF Identification Friend or Foe. 9, 10

IoT Internet of Things. 1, 10, 11, 16, 17, 19, 22, 23, 25

IPC Interprocess communication. 25

LTL Linear Temporal Logic. 26

MAC Message Authentication Code. 15, 22, 25, 26

MPU Memory Protection Units. 19

NIST National Institute of Standards and Technology. 2, 28

OAS Optimistic Aggregate Signature. 21

PCRs Platform Configuration Registers. 19

PL Pulse Length. 8, 30

PQC Post-Quantum Cryptography. 28, 62

PRF pseudorandom function. 53

PRI Pulse Repetition Interval. 8, 9

RA Remote Attestation. 18, 25, 26

Radar Radio Detection And Ranging. xiii, xv, 1, 2, 5–7, 9, 11–13, 16, 28, 30–34, 36–40, 46, 48, 51, 53, 55, 56, 63, 65

RATS Remote Attestation Procedures. 29

RCS Radar Cross Section. 5–7, 11, 13

RF Radio-Frequency. 1, 7, 9, 48

RFID Radio-Frequency Identification. 1, 9, 10, 12, 16, 18, 27, 62

RIS Reconfigurable Intelligent Surface. 28

ROM Read-Only Memory. 19, 23, 25

RTM Root-of-Trust for Measurement. 25

RTT round-trip-time. xv, 51–58

SNR Signal-to-Noise Ratio. 7–9, 13

TCB Thread Control Block. 25

TCG Trusted Computing Group. 18, 29

TOCTOU Time-of-Check Time-of-Use. 47

TPM Trusted Platform Module. 18, 19

UDS Unique Device Secret. 19

WSN Wireless Sensor Network. 26

1

Introduction

Modern surveillance systems rely on a constant flow of trustworthy information [49]. In advanced *RAdio Detection and Ranging* (Radar) platforms, e.g., the system does not operate in isolation. Its functionality depends on continuous exchanges with surrounding equipment, sensors, and auxiliary devices. As these systems grow more interconnected, the communication links that bind them together become increasingly more exposed. Adversaries today are no longer limited to simple noise jammers, they can also impersonate legitimate nodes, replay previously captured messages, or strategically disrupt the network to degrade system performance at critical moments or obtain sensitive information [14, 50].

These challenges create a compelling need for communication networks that can maintain such security and reliability even under hostile conditions. However, achieving this becomes more difficult when the devices participating in the network do not share equal capabilities. In many real-world scenarios, such as *radio-frequency identification* (RFID) like devices, Radar peripherals, or *Internet of Things* (IoT) nodes, some must operate with limited computational power, restricted memory, or constrained energy budget [40]. Designing a secure architecture that remains robust despite these asymmetries is therefore both technically demanding and crucial for operational resilience.

This project was conducted in collaboration with Saab AB and explored how such a secure, attack-resilient network can be built over an asymmetric communication link. The focus was on creating a general purpose network that ensures high data security, integrity, and reliability, while remaining resistant to attacks such as spoofing, repetition, jamming, interference, and denial of service. The communication link considered in this thesis is provided and derived from data provided by Saab AB, which defines the available communication rates between the Radar and the transponder. The two directions operate at significantly different rates, making the link inherently asymmetric, a constraint that shapes much of the protocol design in this thesis. The network itself consists of a single central node, i.e. a rotating Radar, and several leaf nodes implemented as transponders using *backscatter communication* (BC) [42], a technique allowing a device to communicate by reflecting an existing *radio-frequency* (RF) signal. Communication with the leaf nodes is treated as a secondary function of the Radar, layered onto its primary surveillance role without interrupting it. As a consequence, the Radar can only exchange data with a given transponder while its antenna is pointed at that transponder, and the duration

of each such illumination window is set by the Radar's rotation rate.

To address the challenges faced, the thesis explored how attestation protocols could be used to strengthen trust across such networks. According to the *National Institute of Standards and Technology* (NIST) [36], attestation is defined as follows; "The process of providing a digital signature for a set of measurements securely stored in hardware, and then having the requester validate the signature and the set of measurements". That is, attestation offers a systematic way to verify that devices are running correct firmware, behaving as expected, and have not been compromised; i.e. an authentication of the device plus a state proof. This makes it especially appealing for systems where trust must be established at scale and under limited physical availability. This definition is assumed throughout the thesis. By examining state-of-the-art attestation schemes and adapting them for asymmetric devices within the specific ecosystem, the work aims to contribute a security architecture capable of performing in contested environments.

In doing so, this research situates itself within a broader technological trend; the rapid expansion of secure communication methods for resource-constrained devices. The ability to guarantee trustworthy operation at scale is becoming essential. A solution that succeeds in the Radar domain, therefore, has implications far beyond a single application.

1.1 Research questions

The questions explored in this thesis are as follows:

- How can a remote attestation protocol be designed to operate over an asymmetric, low-bitrate backscatter link between a rotating Radar and resource-constrained transponders?
- What security guarantees can such a protocol provide against an adversary capable of eavesdropping, replay, spoofing, jamming, and physical capture of transponders?

1.2 Thesis outline

This thesis is organized into seven chapters. Chapter 1 introduces the motivation behind the project and states the research questions. Chapter 2 provides the background needed for the chapters that follow, covering the fundamentals of Radar, transponders, and BC, the security and cryptographic concepts relevant to this work. Chapter 3 covers a survey of remote attestation, including state-of-the-art protocols for resource-constrained and swarm settings. Chapter 4 describes the methodology: it defines the system and adversary models, derives the resulting security requirements, presents the proposed ABLA protocol, and outlines the simulation- and formal-verification-based evaluation approach. Chapter 5 reports the results of the OMNeT++ performance evaluation and the Tamarin symbolic verification.

Chapter 6 discusses the findings from Chapter 5 and their implications. Chapter 7 concludes the thesis and contains directions for future work.

2

Background

This chapter provides the necessary background on Radar, transponders, and BC needed to understand the limitations of the data link which the proposed attestation protocol will operate over.

2.1 Radar

Radar [52] systems operate by transmitting *electromagnetic* (EM) waves in a direction, based on the direction of the antenna, and then listen for reflections, called echoes. Much like sound reflecting off a surface, a Radar echo returns only a fraction of the transmitted signal, with the returned power determined by the reflective properties of the target. These reflective properties are formally captured by its *radar cross section* (RCS), which is dependent on factors such as size, material, and orientation relative to the Radar. A large RCS indicates the object is easily detectable, while a small one makes it harder to detect. If the echo is sufficiently strong relative to noise, it can be used to estimate properties of the target such as range, direction, and radial velocity.

The earliest practical Radar systems were developed independently in several nations during the 1930s, driven primarily by the need for early warning against aircraft, and saw rapid refinement during the Second World War [52]. Since then, Radar has become indispensable across a broad range of civilian and military applications, including air traffic control, weather observation, maritime navigation, remote sensing of the Earth and planetary surfaces, automotive driver-assistance systems, and industrial level and distance sensing.

2.1.1 Electromagnetic Waves

Radar systems can in principle operate anywhere in the electromagnetic spectrum, see Figure 2.1, but in practice they are confined to the radio and microwave regions, spanning roughly 3 MHz to 300 GHz. The choice of operating frequency is a fundamental design parameter. Lower frequencies propagate with less attenuation and penetrate obstacles and weather more easily, while higher frequencies enable smaller antennas, wider bandwidths, and finer spatial resolution [52]. To organize this frequency spectrum, the IEEE has standardized a set of letter-designated frequency bands [26]. These are L-band (1 GHz to 2 GHz), S-band (2 GHz to 4 GHz), C-band

(4 GHz to 8 GHz), X-band (8 GHz to 12 GHz), and K-band (18 GHz to 27 GHz), each historically associated with particular applications, from long-range air surveillance at the lower end to automotive and imaging Radar at the higher end.

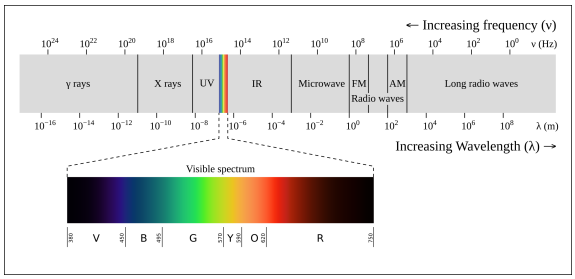


Figure 2.1: The electromagnetic spectrum [53].

Electromagnetic waves consist of coupled oscillating electric and magnetic fields, see Figure 2.2, that sustain one another as they propagate, allowing them to travel through vacuum as well as through many material media [52]. The two fields are mutually perpendicular and also perpendicular to the direction of propagation, so an electromagnetic wave is a transverse wave. The orientation of the electric field vector defines the *polarization* of the wave, which may be linear, circular, or elliptical, and which strongly influences how the wave interacts with a target: the effective RCS of an object and the efficiency with which a receiving antenna captures the echo both depend on the alignment between the incident polarization and the geometry of the scatterer or antenna.

In vacuum, electromagnetic waves propagate at the speed of light, which is defined as $c_0 = 299\,792\,458\,\mathrm{m/s}$. This is the basis for using time-of-flight measurements to estimate range to an object [52]. The frequency f and wavelength λ of the wave are related through

$$\lambda = \frac{c_0}{f}, \tag{2.1}$$

so that a 10 GHz X-band signal has a wavelength of roughly 3 cm. The wavelength sets the characteristic physical scale of antennas, resonant structures, and scattering objects, and therefore plays a central role in the design of both the Radar itself and any cooperative target such as a backscatter transponder.

The range of a target is obtained from the time-of-flight of the echo according to:

$$R = \frac{c_0 \tau}{2}, \tag{2.2}$$

where τ is the round-trip time, i.e., the time taken from transmitting the signal to receiving the echo. The factor of two accounts for the fact that the wave traverses the distance to the target and back.

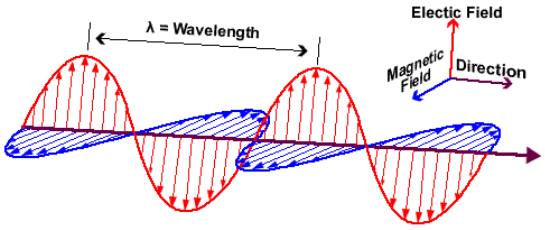


Figure 2.2: Illustration of an electromagnetic wave [45].

2.1.2 The Radar Equation

The amount of power returned to the Radar from a target is described by the *radar equation*, which combines the geometry of transmission, scattering, and reception into a single expression. For a monostatic Radar, where the same antenna is used to transmit and receive, the received echo power can be written as

$$P_r = \frac{P_t\, G_t\, G_r\, \lambda^2\, \sigma}{(4\pi)^3\, R^4}, \tag{2.3}$$

where P_t is the transmitted power, G_t and G_r are the transmit and receive antenna gains, λ is the wavelength, σ is the RCS of the target, and R is the range [52]. The characteristic $\frac{1}{R^4}$ dependence reflects the two-way path: the wave spreads as $\frac{1}{R^2}$ on the way out, illuminates the target, and the scattered field spreads as $\frac{1}{R^2}$ again on the way back. As a consequence, doubling the range reduces the received power by a factor of sixteen, which is the dominant factor limiting the maximum detection distance of any Radar system.

The ability of a Radar to detect a target is ultimately determined by the ratio of the received echo power to the noise power at the receiver output, i.e. the *signal-to-noise ratio* (SNR). Thermal noise, quantified by $k_B T B$ where k_B is the Boltzmann constant, T the effective system noise temperature, and B the receiver bandwidth, sets a fundamental floor, while additional losses arise from the antenna, the propagation medium, and the receiver electronics. Target detection is therefore a statistical decision problem in which a threshold is applied to the received signal to balance the probability of detection against the probability of false alarm [52].

2.1.3 Pulsed Radar

In a pulsed Radar, the transmitter emits a short burst of RF energy, the *pulse*, and then falls silent while the receiver listens for returning echoes [52]. After a fixed interval, the next pulse is transmitted and the cycle repeats. This time-sharing between transmission and reception avoids the need to separate the weak returning echo of the crosstalk from a continuously radiating, much stronger transmit signal, and it directly ties the measurement of target range to the simple time-of-flight expression in Equation (2.2).

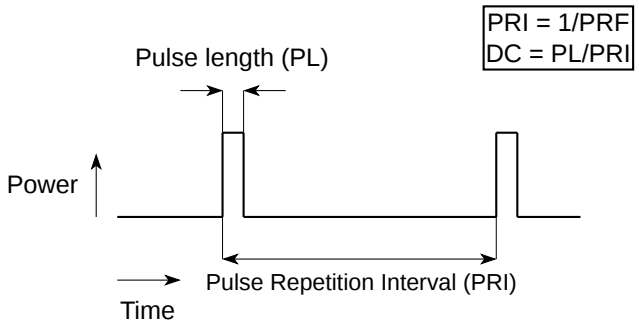


Figure 2.3: Illustration of a pulsed wave and the description of the different parameters.

A pulsed waveform is characterized by three principal parameters, see Figure 2.3: the *pulse length* (PL), denoted τ_p, which is the duration of the transmitted burst; the *Pulse Repetition Interval* (PRI), which is the time between the start of successive pulses; and the *Pulse Repetition Frequency* (f_{PRF}), which is simply the inverse of the PRI, the number of pulses transmitted per second [52]. The ratio of PL to PRI is known as the *duty cycle* (DC) and sets the relationship between the peak transmitted power and the average transmitted power, according to:

$$P_{\text{avg}} = P_t \frac{\tau_p}{\text{PRI}} = P_t \tau_p \text{PRF}. \tag{2.4}$$

Because the detection performance is governed by the energy in each echo rather than by the peak power alone, a longer pulse delivers more energy and therefore has a higher SNR, which is the level of the desired signal compared to the noise signal.

This advantage, however, comes at the cost of range resolution. The *range resolution* ΔR is the minimum separation at which two targets along the same line of sight produce distinguishable echoes, and for an unmodulated pulse of width τ_p it is given by [52]:

$$\Delta R = \frac{c_0 \tau_p}{2}. \tag{2.5}$$

A 10 ns pulse, e.g., yields a resolution of 1.5 m. To escape the direct coupling between pulse energy and PL, modern pulsed radars frequently employ *pulse compression*, in which a long pulse is internally modulated so that different portions of it can be distinguished from one another. A common choice is a linear frequency sweep, or *chirp*, in which the carrier frequency rises (or falls) smoothly across the pulse; an alternative is a *phase code*, a predetermined sequence of abrupt phase shifts (typically 0 and π) applied at fixed sub-intervals of the pulse. At the receiver, the echo is correlated against a stored replica of the transmitted waveform, a process known as *matched filtering*, which concentrates the energy of the long pulse into a much shorter peak. The resulting range resolution is then governed by the signal bandwidth B rather than the physical PL:

$$\Delta R = \frac{c_0}{2B}, \tag{2.6}$$

which allows long, high-energy pulses to achieve the fine resolution otherwise associated with very short pulses [51].

The PRI also sets an upper bound on the range that can be measured without ambiguity. An echo returning after a time greater than the PRI will be received during the listening window of a subsequent pulse and will therefore appear at an apparent range that is shorter than its true value. The *unambiguous range* is accordingly:

$$R_{\mathrm{ua}} = \frac{c_0 \mathrm{PRI}}{2} = \frac{c_0}{2\,\mathrm{f_{PRF}}}. \tag{2.7}$$

A f_{PRF} of 10 kHz, e.g., corresponds to an unambiguous range of 15 km. Increasing the f_{PRF} increases the rate at which the target scene is sampled but reduces the unambiguous range, while lowering the f_{PRF} has the opposite effect, illustrating the fundamental trade-off between range coverage and update rate in pulsed Radar design [52].

2.2 Transponder Fundamentals

In short, a transponder is a device that receives a signal and automatically emits a different signal in response. The term itself comes from a combination of *transmitter* and *responder*, transponder [46]. The concept originated during the Second World War with the development of *Identification Friend or Foe* (IFF) systems, where aircraft were equipped with devices that replied to interrogating Radar pulses with a coded signal to distinguish allied aircraft from hostile ones [60]. Since then, transponders have become ubiquitous in modern technology, appearing in applications as diverse as air traffic control, satellite communications, RFID, keyless entry systems, and wireless sensor networks.

Transponders can be broadly categorized according to how they generate their response signal [46]. Active transponders contain their own power source, typically a battery, and actively generate and transmit a new RF signal upon receiving an interrogation. This approach allows for long communication ranges and high SNR at the Radar, but at the cost of complexity, and a finite operational lifetime limited by the energy source. Passive transponders, in contrast, have no internal power source for transmission and instead derive the energy required for their response from the interrogating signal itself. Semi-passive transponders occupy an intermediate position, using a battery to power on-board electronics (such as sensors or microcontrollers) while still relying on the interrogator's signal for the communication itself.

When the target of interest is not a passive scatterer but an active or semi-passive transponder, the timing of the pulsed Radar takes on an additional meaning. The transponder introduces a delay τ_d between the arrival of the interrogating pulse and

the moment the response begins, so the measured round-trip time becomes:

$$\tau = \frac{2R}{c_0} + \tau_d, \tag{2.8}$$

and the transponder delay must either be calibrated out of the range estimate or explicitly subtracted within the protocol [60]. Any variability in this delay from one interrogation to the next, i.e., timing jitter, translates directly through Equation (2.2), into a range error of $c_0/2$ per unit of timing uncertainty, corresponding to approximately 15 cm per nanosecond. The stability of the transponder timing therefore becomes a central factor in the achievable range accuracy and is a primary driver of the protocol design choices discussed in later chapters.

2.3 Backscatter Fundamentals

BC, in its simplest form, is a way of taking an incident signal and then modulating and reflecting it back, embedded with additional data [42]. That is, the backscatter device does not generate its own carrier signal, rather it reflects the existing energy in the received signal. As aforementioned, this technology can be dated back all the way to the Second World War where this technology was used in IFF systems. This technology then led to the development of RFID systems where the receiver and transmitter are used in proximity to reduce path losses.

The attraction of BC lies in its asymmetry. Because the BC device does not need a local oscillator, power amplifier, or up-conversion chain, its transmit path can be reduced to little more than an antenna and a controllable impedance. This makes it possible to build communicating devices that consume orders of magnitude less power than conventional radios, at the cost of shifting most of the hardware complexity, and essentially all the transmit power, onto the interrogator side of the link [32]. As a result, BC is today the technology of choice for applications in which the tagged device must be small, inexpensive, and energy-constrained, most notably passive RFID, wireless sensing, and emerging low-power IoT scenarios [42].

2.3.1 System Architectures

Backscatter systems are conceptually simple, consisting of three subsystems: a carrier signal source, a backscatter transceiver (tag), and a backscatter receiver. Depending on how the underlying systems are distributed between physical devices, three principal architectures can be distinguished, see Figure 2.4. In a) a *monostatic* configuration [35], the carrier source and the backscatter receiver are co-located and share a single antenna, as is typical for commercial RFID readers. This is the most compact arrangement, but it forces the receiver to extract a weak modulated echo in the presence of the much stronger transmitted carrier, which leads to self-interference.

In b) a *bistatic* configuration [21], the carrier emitter, backscatter tag, and receiver are implemented as separate, spatially distributed devices. By placing the carrier emitter in proximity to the tag, the tag receives a stronger incident signal, improving

the strength of the backscattered signal at the reader. This separation also decouples the forward and backscatter links, alleviating the self-interference constraints present in monostatic systems and reducing the near-far effect.

Finally, in c) an *ambient* backscatter configuration [32], no dedicated carrier source is used at all; instead, the backscatter device modulates signals that already exist in the environment, such as television, cellular, or Wi-Fi transmissions. Ambient backscatter removes the need for a dedicated illuminator but makes the link inherently dependent on signals outside the designer's control. Based on a survey of the ambient backscatter communications literature [68], the technology presents a compelling solution for battery-free, low-power IoT devices by leveraging existing wireless signals as both an energy source and communication medium. However, practical deployment at scale remains constrained by unresolved issues in receiver compatibility, interference with incumbent systems, and underdeveloped downlink communication.

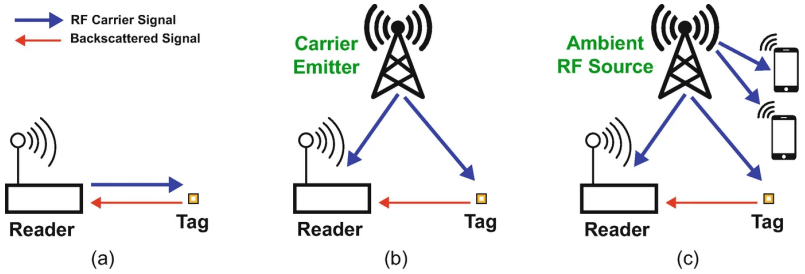


Figure 2.4: High-level block diagram of three different BC systems [1]. a) monostatic backscatter, b) bistatic backscatter, and c) ambient backscatter.

2.3.2 The Backscatter Link Budget

In contrast to the one-way Radar-range expression, a backscatter link involves two wireless paths: a forward path from the carrier source to the tag, over which the tag is illuminated, and a return path from the tag to the receiver. When the illuminator is a pulse-Doppler Radar, the same instrument serves both as the carrier source and as the receiver, and the tag appears as a target in a single range-Doppler cell determined by its range and radial velocity. The average received power per pulse from a tag at range R in a monostatic geometry follows the Radar equation,

$$P_r = \frac{P_t\, G_t\, G_r\, \lambda^2\, \sigma_{\text{tag}}}{(4\pi)^3\, R^4}, \tag{2.9}$$

where σ_{tag} is the RCS of the tag [52, 22]. For a bistatic setup with different transmitter-to-tag and tag-to-receiver distances R_1 and R_2, the R^4 factor generalizes to $R_1^2 R_2^2$, and the link becomes sensitive to the geometry of the three nodes.

Two properties of Equation (2.9) are worth highlighting in the context of this thesis. First, the received echo power decreases with the fourth power of distance in the monostatic case, which makes the achievable communication range fundamentally

shorter than that of a conventional one-way radio link at equal transmit power. Note, however, that this decay bounds the range over which the tag's modulated payload can be decoded; target detection, ranging, and velocity estimation are carried out by the host Radar's standard pulse-Doppler processing on the carrier return and are generally feasible at longer ranges than payload demodulation. Second, the tag conveys information by varying its scattering response between pulses, so that the complex return in its range-Doppler cell is modulated across the coherent processing interval. The Radar recovers the payload by tracking this pulse-to-pulse variation rather than by measuring absolute echo power, which means the communication performance is governed by the depth of the tag's scattering modulation and not by the bulk size of its antenna [22].

In a passive tag, the forward path imposes an additional constraint that has no counterpart in ordinary communication links. The power delivered to the tag's rectifier must exceed the minimum activation threshold of its electronics before any communication can take place [46]. The operational range of a passive system is therefore determined by whichever of the two paths becomes the bottleneck. In short–range, low-sensitivity implementations, the bottleneck is typically the forward, energy-harvesting path, while in carefully designed systems with efficient rectifiers it can shift to the return path, i.e., to the reader's ability to detect the modulated echo above noise.

In an active tag, the power supply constraint of the passive case is replaced by an on-board energy source, typically a battery, which decouples the tag's ability to operate from the strength of the incident Radar signal [46]. The operational range of an active system is therefore determined solely by the return path, i.e., by the reader's ability to detect the modulated echo above the noise floor. This removes the forward-path bottleneck entirely and allows the tag to maintain a consistent modulation depth and switching behavior regardless of its distance from the Radar. The tradeoff is one of practicality rather than physics: the introduction of a dedicated power source increases the tag's cost, size, and maintenance burden, and in applications involving large numbers of tags or inaccessible deployment locations, these factors can outweigh the range and reliability benefits that active operation provides.

2.3.3 Modulation and Encoding

From a communications standpoint, the tag's role is to modulate information on the reflected wave by switching its antenna between two or more load states, each corresponding to a different reflection coefficient. The simplest choice, and the one used in the overwhelming majority of passive RFID tags, is a binary scheme in which the reflection coefficient is toggled between two values to produce *amplitude-shift keying* (ASK) or *phase-shift keying* (PSK) on the backscattered signal, i.e., *binary amplitude-shift keying* (BASK) or *binary phase-shift keying* (BPSK) [63, 41]. The distance between the two reflection states in the complex plane, together with the incident carrier amplitude, sets the magnitude of the useful signal at the receiver and thus the available link margin.

Several encoding and framing techniques are often combined with this basic scheme

to improve detection. Subcarrier modulation, in which the tag switches its load at a rate much higher than the data rate, shifts the backscattered signal away from the strong carrier leakage near DC and simplifies the separation of the echo from self-interference at the reader. Line codes such as Manchester or FM0 ensure that the backscattered waveform has no long runs of constant value, which aids symbol synchronization and DC-balance at the receiver [23]. Higher-order modulations such as quadrature amplitude modulation, obtained by switching between more than two impedance states, have been demonstrated as a route to higher data rates on the return link, at the cost of increased tag complexity and reduced robustness at low SNR [63].

2.3.4 Backscatter Transponders

A backscatter transponder is a passive or semi-passive device that communicates by modulating the reflection of an incident electromagnetic wave rather than by generating its own carrier signal. Instead of actively transmitting, the transponder alters the way its antenna reflects the incoming wave, using the aforementioned technologies, and the resulting variation in the reflected signal encodes the information to be transmitted back to the interrogator [15].

The physical principle underlying BC is the dependence of an antenna's scattered field on the impedance of the load connected to its terminals. When an antenna is illuminated by an incident wave, a portion of the incoming energy is re-radiated as a scattered field. The amplitude and phase of this scattered field are governed by the reflection coefficient at the antenna port,

$$\Gamma = \frac{Z_L - Z_0^*}{Z_L + Z_0}, \tag{2.10}$$

where Z_L is the load impedance connected to the antenna and Z_0 is the antenna impedance, with Z_0^* denoting its complex conjugate [41].

The power budget of a backscatter link is governed by a modified form of the Radar equation, in which the received power at the reader decreases with the fourth power of the distance to the transponder, as the signal traverses the reader-to-transponder and transponder-to-reader paths successively. This rapid decay with distance, together with the low sensitivity of passive transponder front-ends, is the principal factor limiting the operational range of backscatter systems and motivates much of the ongoing research into improved antenna designs, energy harvesting circuits, and receiver architectures.

2.3.5 Practical Considerations

In practice, non-idealities impact the performance of real backscatter systems. Because the tag antenna both harvests energy and radiates the backscattered response, its impedance must be jointly optimized for power transfer in one load state and for a large average RCS between states, and these two objectives are generally in tension. Multipath propagation in indoor or cluttered environments introduces fading that

can temporarily suppress the forward path, the return path, or both, and leads to characteristic read-range nulls. However, in a monostatic case all of these are identical and will have the same fading. In the presence of multiple tags, protocol-level mechanisms such as framed slotted ALOHA or tree-based anti-collision algorithms are required to arbitrate access to the shared backscatter channel [23]. Finally, regulatory constraints on the emitted power of the interrogator, which vary significantly between regions and frequency bands, place a hard upper bound on the illuminating field strength and therefore on the achievable range. These considerations are revisited in later chapters when the specific transponder protocol used in this thesis is introduced.

2.4 Security

Information security has grown from a specialist concern of governments and banks into a pervasive requirement of everyday technology. As computing and communication have become embedded in vehicles, medical devices, industrial controls, payment systems, and the fabric of personal and social life, the consequences of security failures have grown correspondingly. A single compromised device can expose private data, disrupt critical services, inflict physical harm, or serve as an entry point into a larger system. At the same time, the number of connected devices has grown rapidly, and is projected to reach tens of billions globally [18], many of which are small, inexpensive, and operate with little or no human oversight. This combination of ubiquity, stakes, and heterogeneity is what makes security a central concern in the design of modern communication systems, including the class of backscatter-based devices considered in this thesis.

Security is not a single property but a collection of properties that a system should preserve in the presence of an adversary. It is therefore meaningful only in relation to an explicit description of what is being protected, from whom, and under what assumptions. Such a description is commonly referred to as a *threat model*, and it specifies the assets of value, the capabilities of the attacker, and the boundaries of trust within the system. Without a threat model, statements about whether a system is "secure" are unfalsifiable; with one, security becomes a concrete engineering question about whether the defenses in place are sufficient against the adversary that has been assumed.

2.4.1 The CIA Triad

A widely used conceptual foundation for information security is the so-called *CIA triad*, which distills the goals of a secure system into three complementary properties: *confidentiality*, *integrity*, and *availability* [59].

Confidentiality requires that information is disclosed only to parties authorized to receive it. In a communication system, this is typically achieved by encrypting messages so that they are meaningful only to holders of the appropriate key. A violation of confidentiality need not alter the system's behavior in any visible way;

an eavesdropper who silently records transmissions can compromise confidentiality without being noticed.

Integrity requires that information cannot be altered, forged, or replayed without detection. This applies both to data in transit and to the origin of messages, and is typically enforced through cryptographic mechanisms such as *message authentication codes* (MAC) or digital signatures, together with freshness indicators such as nonces (numbers used once), sequence numbers, or timestamps. An attacker who cannot read a message may nevertheless be able to modify or retransmit it, and integrity mechanisms are what allow the receiver to distinguish genuine traffic from such manipulations.

Availability requires that the system remains usable for its legitimate users. Threats to availability include denial-of-service attacks, deliberate jamming of the wireless medium, and resource-exhaustion attacks that force a device to spend its limited energy or memory on spurious requests. For battery-free or energy-harvesting devices, availability is closely coupled to the energy budget: an adversary who can force the device to perform unnecessary work can effectively silence it without ever breaking any cryptographic primitive.

These three properties are often supplemented by further goals such as *authenticity*, i.e. assurance of the identity of the communicating party, and *non-repudiation*, i.e., the inability of a party to deny having sent a message. For the purposes of this thesis, the most directly relevant goals are authenticity, integrity, and confidentiality, which together determine whether a reader can trust that it is communicating with the intended transponder and that the exchanged data has not been disclosed or tampered with.

2.4.2 Authentication and Identification

Identification is the act of a party claiming an identity, while *authentication* is the process by which that claim is verified. The distinction matters because identification alone provides no security guarantee; anyone can claim to be a particular device, and only authentication establishes that the claim is genuine. In wireless systems, authentication is commonly realized through *challenge–response* protocols, in which the verifier sends an unpredictable challenge and the prover returns a response that could only have been computed by a party in possession of a shared secret or a corresponding private key.

A particular hazard in identification systems is the *cloning* or *spoofing* of devices. *Cloning* refers to the creation of an exact, unauthorized copy of a legitimate device, in which the adversary extracts the identifying information from an authorized tag and reproduces it on a blank or programmable one. *Spoofing*, by contrast, only requires the fabrication of a plausible-looking identifier, the replay of a previously captured response, or the guessing of valid IDs based on knowledge of the numbering scheme. The key distinction is that spoofing does not require access to any specific authorized device. If a tag transmits only a fixed identifier, an attacker who observes a single transaction can subsequently impersonate it by replaying the captured iden-

tifier. This class of attack is well documented against early RFID deployments and motivates the use of cryptographic authentication, in which the response depends not only on the device's secret but also on a fresh, verifier-chosen challenge, ensuring that past responses cannot be reused.

2.4.3 Confidentiality and Encryption

Confidentiality in a wireless channel is fundamentally a matter of encryption, since the physical medium itself is open to any receiver within range [56]. Encryption schemes are broadly divided into *symmetric* schemes, in which the same secret key is used to encrypt and decrypt, and *asymmetric* (public-key) schemes, in which the two operations use distinct but mathematically related keys. Symmetric encryption is generally much cheaper in computation [24] and energy, which makes it the natural default for constrained devices, while asymmetric schemes are primarily used for key exchange, digital signatures, and scenarios in which no shared secret can be pre-established. For this thesis a directed wireless link will be assumed, i.e., the communication will only be possible when the Radar is illuminating the transponder, which enables more protection against an adversary which is unaware of the location of the transponder.

For constrained devices, the cost of encryption is not just theoretical. The energy consumed by a cryptographic operation, the silicon area required to implement it, and the latency it introduces all directly affect the feasibility of the system. This has motivated a dedicated research effort on *lightweight cryptography*, which seeks algorithms that provide meaningful security guarantees within the tight resource budgets of small IoT devices [33].

2.4.4 Integrity and Replay Protection

Even when confidentiality is ensured, an attacker with access to the wireless channel can still attempt to modify messages in flight or to retransmit earlier captured messages. Integrity mechanisms, most commonly message authentication codes built from symmetric primitives, allow the receiver to detect such modifications with high probability. Replay protection, however, requires an additional element of *freshness*: a message that is cryptographically valid when first sent remains cryptographically valid if it is simply retransmitted later, and only the inclusion of a *nonce*, *counter*, or *timestamp* allows the receiver to distinguish the original from the replay. A nonce is a value intended to only be used once. A counter, in simple terms is a mechanism which produces new inputs for cryptographic functions.

For ranging and localization systems, a particularly important variant of replay is the *relay attack*. Here, an adversary forwards the legitimate exchange between two parties over a path that differs from what they believe they are using. This can make the verifier conclude that a distant prover is nearby, or vice versa. Because the exchanged messages are genuine and cryptographically valid, classical authentication alone does not detect relay attacks. Defending against them requires the verifier to reason about the *time* of the exchange, not only its contents, which is the domain

of *distance-bounding* [8].

2.4.5 Security in Constrained IoT and Backscatter Devices

The aforementioned principles can apply to any system, but their implementation becomes markedly harder as the device shrinks. Small IoT and backscatter devices are constrained in several simultaneous dimensions: available energy, computational throughput, memory, silicon area, and unit cost. A passive backscatter tag, in particular, may have only microwatts of harvested power with which to execute its entire protocol, and its communication is inherently broadcast to anyone within range of the interrogator.

These constraints interact with security in characteristic ways. Cryptographic algorithms designed for general-purpose processors are often too heavy to run on a tag, both in energy and on reconfigurable hardware, which has driven the development of lightweight ciphers and hash functions mentioned above. The tag's inability to generate its own strong randomness limits its capacity for protocol freshness and biases designs toward verifier-chosen challenges. Physical access to deployed devices is frequently easy for an attacker, which makes *side-channel* attacks, in which secrets are inferred from power consumption, timing, or electromagnetic emissions, a realistic concern that has no real equivalent in purely remote threat models. The broadcast nature of the wireless link means that eavesdropping, replay, and relay attacks can be mounted without any physical contact at all.

At the same time, the applications envisioned for such devices are often precisely the ones where security failures matter most: access control, vehicle keys, payment, supply-chain integrity, medical implants, and industrial sensing. A transponder that can be silently cloned, impersonated, or relayed at distance can defeat the very property it was deployed to provide. Security in this setting is therefore not an optional add-on layer but a design constraint that must be reasoned about from the outset, alongside power, range, and cost. The remainder of this thesis returns to these considerations in the context of the specific transponder protocol under study, where the goals of reliable range estimation and of resistance to impersonation and relay must be achieved within the limits of a passive backscatter device.

2.5 Attestation

The security properties discussed in the previous section, such as authentication and integrity, rest on an implicit assumption: that each party in a protocol executes the software it is supposed to execute. An authenticated message only means what the protocol specifies if the device that produced it has not been silently modified, reprogrammed, or otherwise compromised. In practice this assumption is fragile, particularly for devices that are physically accessible, connected to untrusted networks, or deployed in the field for long periods without direct supervision. *Attestation* is the class of techniques that allow one party to verify that another is in a trustworthy software state, and thereby restore the link between cryptographic identity and actual behaviour [36].

Attestation is particularly relevant for small, connected devices. A backscatter transponder, an RFID-based access token, or a low-power sensor node is typically easy to reach physically, runs firmware that may be updated in the field, and may be one of many nominally identical devices deployed at scale. An attacker who modifies the firmware of such a device can subvert higher-level security guarantees without breaking any cryptographic primitive. Attestation provides a mechanism by which a remote party can gain assurance, before relying on a device, that the device is running the expected software. In the following, the focus is on *remote* attestation, where the Verifier (a trusted entity) and the Prover (the assumed to be untrusted entity) are separated by a communication channel.

Remote attestation (RA) is the process by which a Verifier establishes trust in the software state of a remote device, the Prover. Several frameworks have been proposed for reasoning about the roles and message flows involved; this thesis adopts the terminology of the *Trusted Computing Group* (TCG) *Device Identifier Composition Engine* (DICE) architecture [64], which is well-suited to resource-constrained devices and provides a clean separation between the components that produce attestation evidence and those that consume it. The architecture defines several roles, of which the most important for the present discussion are the *device* – consisting of an *Attesting Environment* that collects claims and a *Target Environment* about which claims are collected – and the *Verifier*, which evaluates the resulting Evidence. Additional roles (the *Endorser*, who vouches for the device's hardware; the *Relying Party*, who consumes the Verifier's appraisal; and the *Owner*, who configures policy) round out the model but are not central to the protocols considered in this thesis. The attesting environment packages its claims about the target environment as integrity-protected and authenticated evidence. DICE further supports *layered devices*: each layer L_{n-1} acts as the attesting environment for the next layer L_n, attesting L_n before passing control to it, so the chain of trust extends upward through the software stack.

The DICE architecture further distinguishes between *implicit* and *explicit* attestation. *Implicit attestation* applies when a device is expected to be in a static condition and a simple verification approach suffices: rather than transmitting a full description of the device's state, the Verifier infers correctness from the device's ability to participate in a protocol that depends, directly or indirectly, on a key derived from a measurement of the device. If the device's software has been tampered with, the derived key changes and the protocol fails, so the very success of the interaction acts as evidence of integrity. *Explicit attestation*, by contrast, applies when a device has a known configuration and runs known software, and evidence – typically a signed report describing measurements of the loaded code and configuration – is conveyed to a Verifier and compared against reference values. These two forms are not mutually exclusive, and a single device may use implicit attestation for routine operation and explicit attestation when a Verifier needs a more detailed account of its state.

A key distinguishing feature of the DICE architecture is that it does not require a *Trusted Platform Module* (TPM), which is a dedicated security co-processor, standardized by the Trusted Computing Group [65]. A TPM provides a hardware root of trust on a wide range of computing platforms, most prominently *Personal Com-*

puter (PCs) and servers. It offers a small set of well-defined services: secure generation and storage of cryptographic keys that never leave the chip in plaintext, a set of *Platform Configuration Registers* (PCRs) into which measurements of the boot chain and loaded software are extended (i.e., hashed cumulatively) to record the platform's state, signed attestation quotes over those PCRs that allow a remote verifier to reason about what code was loaded, and sealing operations that bind data to a specific platform state so it can only be decrypted when the same measurements are present. TPMs are typically realized as discrete chips connected over a low-bandwidth bus, or as firmware TPMs running inside a trusted execution environment on the main CPU. In both cases they add cost, board area, power draw, and integration complexity that are often unacceptable for low-end IoT devices, where the entire microcontroller may have only tens of kilobytes of memory and a very low unit cost. DICE derives its root of trust from a *Unique Device Secret* (UDS)–a cryptographic key fused into the silicon that gives the device a unique identity–combined with a small amount of immutable boot code, rather than from a dedicated security coprocessor, i.e. a TPM. This approach provides comparable assurances about device identity and software integrity while remaining applicable to IoT devices with minimal silicon resources [64].

DICE is best understood within a broader landscape of attestation approaches, which can be categorized along the lines of where the root of trust is anchored. *Software-based* attestation requires no secure hardware or cryptographic secrets, making it attractive for the cheapest devices; it instead relies on tight timing assumptions, where the Prover is expected to compute a checksum over its memory within a bound so strict that any malicious modification of the code would visibly slow the response. This approach is limited to one-hop network topologies due to those strict round-trip time constraints, and its security guarantees have been challenged in the literature [2]. *Hardware-based* attestation places the entire root of trust in dedicated silicon, such as a TPM, providing strong guarantees at the cost of the silicon, power, and integration overheads discussed above. *Hybrid* (hardware-software co-design) attestation, which leverages modest hardware features such as *Read-Only Memory* (ROM) and *Memory Protection Units* (MPU) together with carefully isolated software, offers a more robust alternative than software-based attestation while remaining far cheaper than full TPM-based designs. DICE falls into this hybrid category, and the protocols considered in the remainder of this thesis assume a hybrid attestation model.

3

Related Work

This chapter covers related work in remote attestation and reviews existing protocols relevant to the design of an attestation protocol for BC devices operating over asymmetric, low bit rate data links.

3.1 State of the art: Collective and Swarm Attestation Protocols

Collective Remote Attestation (CRA) protocols enable the verification of software integrity across networks of devices rather than individual nodes. An overview of the state of the art in CRA is provided by Ambrosin et al. [3].

The *Secure and Scalable Aggregate Network Attestation* (SANA) protocol [2] is designed to efficiently aggregate attestation results from large-scale networks, demonstrating the ability to attest up to one million devices in 2.5 seconds. SANA provides guarantees of unforgeability, completeness, scalability, and *denial-of-service* (DoS) limiting under a strong adversary model, in which the attacker fully controls the network and may have compromised an arbitrary subset of devices, including their secret keys. The protocol operates in three phases: *Initialization*, where each Prover receives an *Optimistic Aggregate Signature* (OAS) key pair and an identity certificate signed by the Owner. Leaf *nodes generate* software configuration measurements and create OAS signatures over the configuration, counter for replay protection and *counter* value. While SANA demonstrates impressive scalability, its communication cost is high, and the large multi-hop network it targets places it outside the scope of the scenario considered in this thesis.

The *Scalable Embedded Device Attestation* (SEDA) protocol [7] targets embedded devices and builds upon the *SMART* [17] and *TrustLite* [29] security frameworks. SMART provides attestation for low-end embedded devices, while TrustLite offers a broader security architecture. Like many attestation protocols, SEDA assumes no physical tampering is possible and restricts the adversary to software-only attacks. The protocol consists of an offline initialization phase, where devices receive signing key pairs and certificates from the Operator, and an online attestation phase. During initialization, each device also receives the public key of the Operator and the certificates of its neighbors. In the attestation phase, each device performs the `attestDev` function with its neighbors; when a Verifier initiates attestation, a spanning tree is

built, and results are aggregated toward the root. The computational cost is dominated by cryptographic operations: the root node computes one ECDSA signature and verifies up to $2g$ signatures (where g is its neighbor count), while other nodes compute and verify HMAC-SHA1 MACs. With a MAC length of $l_{\text{mac}} = 160$ bits and a signature length of $l_{\text{sign}} = 320$ bits, the maximum communication overhead for the root device D_1 is $48g + 176$ bytes (sending) and $20 + 56g$ bytes (receiving). Computational time scales linearly with the number of devices in a star topology.

Scalable Lightweight Attestation Protocol For the Internet of Things (SlimIoT) [5] takes a cluster-based approach to network attestation. The protocol divides the network into clusters and maintains attestation results for each cluster over time as a 3-tuple $A_t^i = \langle C_i, S_t^i, T_{\text{last}} \rangle$, where C_i identifies the cluster, S_t^i is a boolean indicating whether the cluster contains any untrusted devices, and T_{last} is the timestamp of the last attestation. Cluster selection for each attestation round is formulated as a minimization problem that balances the probability of compromise against coverage and freshness constraints. The protocol is deemed secure if the probability of an adversary succeeding is negligible, under the conditions that the attestation time does not exceed the adversary's attack time and that the underlying *Pseudorandom Number Generator* (PRNG), MAC, and aggregation schemes are secure and selectively forgery resistant. While SlimIoT demonstrates an interesting optimization-based approach to attestation scheduling, it assumes a large network divisible into clusters, making it inapplicable to the smaller-scale scenario explored in this thesis.

The *Hashgraph-based Attestation for Groups with Autonomous Reporting* (HAGAR) protocol [67] employs techniques similar to those used in blockchain to enable fast and secure continuous attestation aggregation in large IoT networks. HAGAR supports efficient attestation in dynamic and unstructured networks, decentralized verification where each node can attest others, and system-wide consensus. The protocol operates in three phases: *Deployment*, an offline phase where the Operator bootstraps devices and manages keys; *Message Aggregation and Attestation*, where devices exchange self-attestation results along with timestamps, network views, and MACs computed as $\sigma \leftarrow \text{MAC}_k(T_s \| Att_s \| H_s \| ID_s)$; and *Verification*, where the Verifier initiates contact with a device, receives its attestation result and log, and determines whether the device is compromised. HAGAR does not require devices to be constantly online or reachable, and provides robustness against DoS attacks. However, the protocol requires inter-device communication, which is not possible in the network topology considered in this thesis, where nodes can only communicate with the Verifier.

3.2 State of the Art: Attestation for Resource-Constrained and Intermittent Devices

This section surveys remote attestation work relevant to the scenario considered in this thesis, organized into three groups. The first covers attestation protocols for resource-constrained devices, ranging from heterogeneous swarm schemes to protocols that tolerate intermittent power or interrupted execution, to lightweight pro-

tocols that minimize cryptographic or communication overhead. The second covers the hybrid hardware/software *architectures* that supply the primitives—atomic execution, key isolation, secure measurement—on which many of those protocols implicitly depend. The third revisits the on-demand assumption itself and describes a self-measurement alternative. Among the protocols reviewed below, the *Scalable Heterogeneous Layered Attestation* (SHeLA) protocol is the most applicable to the scenario considered in this thesis, and is introduced first.

3.2.1 Swarm and Network-Level Protocols

SHeLA [48] accommodates heterogeneous networks where multiple less powerful devices connect via one-to-one links to more powerful intermediate devices. These intermediate devices may communicate with each other to share information and build a larger picture of the overall network state, and are in turn connected to a base station or Verifier. Critically, the protocol assumes that the more powerful intermediate devices are trusted, an assumption that aligns with the trust model adopted in this thesis.

The *Lightweight Swarm Attestation* (LISA) protocols, LISAα and LISAs [12], provide additional approaches to swarm attestation in IoT networks. Both build on the SMART+ architecture, which provides atomic (uninterruptible) attestation code, a secret key stored in ROM and accessible only to the attestation code, and a fixed-size block of RAM—also accessible only to the attestation code—that holds a counter or timestamp of the last attestation. LISAα is the simpler of the two: it introduces minimal changes over SMART+ and achieves lower end-to-end attestation times, but at the cost of significant communication overhead, since each device may forward up to n reports in the worst case. LISAs, by contrast, reduces bandwidth substantially through in-network aggregation of attestation reports and offers greater flexibility in the level of QoSA provided to the verifier. These benefits come at the cost of increased code complexity, a larger write-protected memory footprint that grows linearly with the swarm size, and longer total attestation time due to the synchronous waiting required at each node.

3.2.2 Intermittent and Interruptible Attestation

The *Remote Attestation of Intermittent IoT Devices* (RESERVE) protocol [47] addresses remote attestation for intermittent devices—devices that do not have constant power and may rely on energy harvesting—and is designed for scenarios where attestation cannot be completed in a single execution cycle. Like many remote attestation protocols, RESERVE assumes no physical adversary is present and restricts threats to software-based attacks, whether remote or local. It assumes certain hardware capabilities, including ROM, a Memory Protection Unit, a number generator, and a real-time clock. The protocol has a communication overhead of 500 bits and assumes a packet size of 128 bytes, consistent with common IoT communication protocols.

The *Interruptible Remote Attestation Through Cuckoo Filters* (ITERATOR) proto-

col [58] addresses the same challenge of interrupted attestation, but without relying on additional hardware, instead using Cuckoo filters to maintain integrity across attestation rounds. The system model includes a trusted Verifier, an untrusted Prover, and a network Operator. During initialization, the Operator provisions each device with a private key, the attestation protocol, and parameters such as block size and the number of blocks to attest per round, and constructs the associated Cuckoo filter. During attestation, ITERATOR attests a subset of memory blocks per round through two actions: ATTEST, which checks the next memory block based on the attestation progress variable $p \in [0, n]$, and CHECK, which re-attests a random selection of k previously attested blocks where $k \in [0, p]$. The Cuckoo filter is queried to verify whether a block is present; if not, the device is deemed compromised.

3.2.3 Lightweight Protocols

Two further protocols target lightweight attestation along different axes. The *Radio Context Attestation in Cognitive Radio Networks* (ROSTER) protocol [69] targets attestation on cognitive radios—radios that dynamically change frequency to find the optimal one—and achieves a communication overhead of 536 bits, making it one of the more lightweight protocols in the literature. The *Lightweight and Transparent Remote Attestation* (LTRAA) protocol [57] reduces computational rather than communication cost, replacing public-key encryption with symmetric polynomials to avoid the burden of asymmetric cryptography.

3.2.4 Hybrid Security Architectures

The protocols above largely assume an underlying hybrid security architecture that supplies the basic primitives they depend on: atomic execution of attestation code, isolated storage of attestation keys, and a trustworthy measurement path. The remainder of this section reviews the architectures that supply those primitives.

Tiny Trust Anchor for Tiny Devices (TyTAN) [9] is a security architecture for low-end embedded devices that simultaneously delivers strong task isolation, dynamic task management, attestation, and hard real-time guarantees—capabilities that prior designs like SMART and TrustLite each provide only in part. The architecture targets a multi-stakeholder model in which the device manufacturer provides trusted hardware and a small set of trusted software components, the owner controls an untrusted operating system, and multiple mutually distrusting task providers can deploy code on the same device. Its key hardware ingredient is an *Execution-Aware Memory Protection Unit* (EA-MPU) inherited from TrustLite, which enforces fine-grained, code-identity-based access control: a given memory region is reachable only when the CPU is executing instructions inside an authorized code region, tasks can only be entered through a designated entry point, and interrupt handling is mediated so that a malicious handler cannot peek at a suspended task's state. Building on top of secure boot and a platform key, TyTAN runs a small set of trusted services: an EA-MPU driver that dynamically (re)configures protection rules as tasks are loaded and unloaded; an *interrupt multiplexer* (Int Mux) that securely saves and wipes the context of secure tasks on interrupts; an *Interprocess communication*

(IPC) proxy that mediates authenticated synchronous or asynchronous inter-task messaging and can set up shared memory for bulk transfer; a *Root-of-Trust-for-Measurement* (RTM) [13] task that hashes each loaded task's code and initial state to produce a position-independent identity; a RA task that uses a key derived from the platform key to MAC reports for remote verifiers; and a secure storage task that seals data under per-task keys. A modified FreeRTOS with an ELF loader handles relocation, scheduling, and preemption of both normal and secure tasks while preserving interruptibility of every TyTAN component, including the RTM hash computation.

Where TyTAN leans heavily on custom hardware features like the EA-MPU, *Hybrid Design for Remote Attestation* (HYDRA) [16] pushes most of the security-critical functionality into software. HYDRA is a hybrid RA design for embedded, cyber-physical, and IoT devices that builds on the seL4 microkernel [54], a general-purpose microkernel whose binary has been formally verified down to the C and assembly level, with machine-checked proofs of functional correctness and of access-control properties (authority confinement, integrity, and confidentiality). The authors use seL4's capability-based isolation to realize the standard set of hybrid RA requirements identified by Francillon et al. [20]: exclusive access to the attestation key, no leakage of K-derived intermediate values, immutability of the attestation binary, uninterruptible execution, controlled invocation, and verifier authentication. The attestation process is launched as the initial user-space process and runs at the highest scheduling priority, which seL4 guarantees no other process can exceed. It holds the only capabilities to its own *Thread Control Block* (TCB), i.e., no other process can suspend the attestation process. The thread also holds its own **Virtual Space** (VSpace), i.e, no other process can inspect its memory, and the binary containing the key, and authenticating incoming requests from the verifier with a MAC plus a freshness check based on a real-time clock to prevent DoS, replay, reorder, and delay attacks. The only hardware feature HYDRA really relies on is a secure boot mechanism, instantiated via NXP's High Assurance Boot [62] on the Sabre Lite board, which establishes a chain of trust from ROM through the verified seL4 binary and into the attestation process.

HYDRA inherits its security guarantees from seL4's formal verification, but it does not itself prove that the resulting RA scheme satisfies the high-level soundness and security definitions of remote attestation. *Verifiable Remote Attestation for Simple Embedded Devices* (VRASED) [44] closes that gap, and is presented as the first end-to-end formally verified RA architecture. Designed as a hybrid hardware/software co-design, it targets low-end embedded devices like the TI MSP430 [61]. Remote attestation itself lets a trusted verifier check the software state of an untrusted remote device—for example, an IoT gadget—to detect malware, typically via a challenge–response protocol where the device returns an authenticated integrity check (an HMAC) over its memory. VRASED splits responsibilities between a hardware module (HW-Mod), which enforces key access control, atomicity, controlled invocation, secure reset, and *Direct Memory Access* (DMA) protection, and a software component, built on the formally verified HACL* [70] HMAC-SHA256 implementation, which computes the actual attestation report. The authors decompose end-to-end

RA soundness and security into sub-properties expressed in *Linear Temporal Logic* (LTL), verify each hardware sub-module as a finite state machine using the NuSMV model checker [19], and then use a theorem prover to show that the composition of these verified pieces implies the high-level soundness and security definitions—the latter via a reduction to HMAC's existential unforgeability.

3.2.5 Beyond On-Demand Attestation

All of the architectures above, and most of the protocols introduced earlier, follow the conventional on-demand RA model, in which the verifier triggers a fresh measurement whenever it wants to check the prover. *Efficient Remote Attestation via Self-Measurement for Unattended Settings* (ERASMUS) [11] departs from that model. The authors argue that on-demand RA has two problems for unattended devices: it can miss "mobile malware" that infects the prover and then leaves before the next request, and it forces the prover to perform an expensive cryptographic computation at potentially inconvenient times, which is bad for safety-critical or real-time devices. ERASMUS instead splits RA into two phases. In the measurement phase, the prover periodically and securely—using hybrid architectures like SMART+ or HYDRA as a base—computes timestamped, MAC-protected self-measurements of its own state and stores them in a rolling circular buffer in insecure memory. In the collection phase, the verifier occasionally fetches the most recent measurements. Because measurements are already authenticated with a shared key that malware cannot access, collection requires no cryptographic work on the prover and no authentication of the verifier's request, which also eliminates the computational DoS concern that previously motivated verifier authentication.

3.2.6 Summary

The landscape surveyed here varies along several axes: network topology (single device vs. swarm), execution model (atomic vs. interruptible, on-demand vs. self-measurement), cryptographic primitives (symmetric vs. asymmetric, aggregated vs. per-device), and the balance of trust placed in hardware versus software. SHeLA's layered trust model—in which more powerful intermediate devices are trusted relative to the constrained leaf devices—most closely matches the scenario considered in this thesis, and informs the design choices made in subsequent chapters. The hybrid architectures (TyTAN, HYDRA, VRASED) provide the primitives that any concrete deployment ultimately rests on, while ERASMUS's self-measurement paradigm offers a useful alternative when devices are unattended or when on-demand cryptography is too expensive to schedule reliably.

3.3 Security in Wireless Sensor Networks

*Wireless Sensor Network*s (WSN)s face a wide range of security threats, as comprehensively surveyed by [40]. Small sensor devices are particularly vulnerable to jamming, eavesdropping, and message injection. Furthermore, these devices often

lack the resources to employ computationally heavy cryptographic functions such as public key cryptography.

Several attack types are especially relevant at the datalink layer. *Sybil attacks* involve an adversary creating many pseudonymous identities to gain disproportionate influence in a network. *Unfairness attacks* aim to degrade the system to gain control of resources. *Collision attacks* exploit hash collisions to forge authentication credentials. *Interrogation attacks* involve eavesdropping on or intercepting data transferred over the data link. Energy depletion attacks also pose a significant threat to battery-powered sensor networks.

The choice of encryption mechanism is critical for sensor networks, as any form of encryption adds communication overhead. In large networks this leads to increased delays, jitter, and packet losses, though this may be less of a concern in the smaller network considered in this thesis. Protocols such as SET-IBS and SET-IBOOS have been proposed to address the deficiencies of symmetric key cryptography through signature-based authentication of encrypted data, though the overhead of asymmetric cryptography remains a limiting factor.

3.4 Security in Military Sensor Networks

Security in military sensor network applications introduces additional requirements, as discussed in [27]. While RSA is computationally heavy and impractical for resource-constrained devices, *Elliptic Curve Cryptography* (ECC) offers comparable security with lower power consumption, albeit with reduced arithmetic primitive support. The authentication protocol presented in [27] consists of two phases. In the registration phase, a new node requests a digital certificate from the base station, sending its identity, public key, and nonce encrypted with the base station's pre-embedded public key. In the subsequent *Authentication* phase, nodes mutually authenticate using these certificates. The total communication cost for a complete authentication cycle is 1172 bit, comprising identity fields (14 bit), timestamps (64 bit), RSA ciphertexts (128 bit), MD5 hash digests (128 bit), and digital certificates (206 bit). At the low data rates characteristic of the links considered in this thesis (6 bps to 75 bps), this translates to authentication times of 15.6–195.3 seconds, which is prohibitively slow for practical deployment.

3.5 Cryptographic Considerations

The security properties discussed so far depend not only on the choice of cryptographic primitives but also on the care with which they are implemented. A well-designed algorithm can be rendered useless by an insecure implementation, and this is particularly acute in resource-constrained devices, where pressure on silicon area, power, and cost often pushes designers toward shortcuts that erode the underlying security.

A striking illustration is provided by the reverse engineering of cryptographic RFID

tags in [43], where the cipher used in a widely deployed family of tags was shown to be breakable within seconds using a precomputed rainbow table. The vulnerability was not caused by a fundamental weakness in the abstract cipher but by concrete implementation choices: a flawed random number generator and a key length of only 48 bit, both of which fall far below modern minimum expectations. The example illustrates that the presence of cryptographic hardware is no guarantee of security; what matters is whether the primitives, their parameters, and the surrounding system have been designed and implemented to a sound standard.

Looking further ahead, *post-quantum cryptography* (PQC) is becoming an increasingly relevant consideration for long-lived secure communication systems, including those built on Radar and sensor technologies [14]. Shor's algorithm, an algorithm used to efficiently factor large prime numbers, when executed on a sufficiently large and stable quantum computer, would break the public-key schemes that underpin most of today's key establishment and digital signatures, notably RSA and elliptic-curve-based schemes. Symmetric primitives such as AES are affected much less severely: Grover's algorithm provides only a quadratic speed-up, so doubling the key length restores the original security margin. The principal threat therefore lies in the asymmetric layer of existing protocols rather than in their symmetric building blocks.

In response to this prospect, NIST has standardized a first set of quantum-resistant algorithms. FIPS 203 [38] specifies ML-KEM, a lattice-based key-encapsulation mechanism derived from CRYSTALS-Kyber, while FIPS 204 [37] and FIPS 205 [39] specify the lattice-based ML-DSA and the hash-based SLH-DSA digital signature schemes, derived from CRYSTALS-Dilithium and SPHINCS+ respectively. Although adoption of these schemes lies beyond the scope of this thesis, they represent the direction in which attestation and related protocols are expected to migrate, and designing such protocols with clear separation between authentication, key establishment, and integrity mechanisms simplifies later substitution of PQ-secure primitives.

An alternative line of work addresses the security of BC not through stronger cryptography but through the physical layer itself. Techniques based on *Reconfigurable Intelligent Surfaces* (RIS), for example, have been proposed as a means of shaping the propagation environment so that legitimate receivers observe a stronger channel than eavesdroppers, thereby providing a form of confidentiality without additional cryptographic cost on the tag [28]. Such approaches are complementary to, rather than in competition with, the cryptographic mechanisms considered in this thesis, and are likewise outside its scope.

4

Methods

This chapter describes the design and evaluation of our protocol ABLA. It begins by outlining the overall research approach, followed by a formal characterization of the system and adversary models that define the operating environment and threat landscape. From these models, a set of security requirements is derived, which in turn drive the protocol design. The protocol itself is presented in detail across its five phases, from provisioning through verification, after which a security analysis evaluates its resilience against each identified threat. The chapter concludes with a description of the simulation-based performance evaluation methodology.

4.1 Research approach

The research process was structured as follows. First, a literature review and state-of-the-art analysis was conducted to identify existing attestation protocols, their assumptions, and their limitations with respect to the target scenario. Second, established attestation frameworks were studied, in particular the TCG DICE architecture [64] and the IETF *Remote Attestation Procedures* (RATS) architecture [10], in order to ground the protocol design in well-defined terminology, roles, and architectural patterns. Third, a system model and threat model were defined to capture the specific constraints and adversarial environment of the target deployment. Fourth, security requirements and design constraints were derived from these models. Fifth, the protocol was designed to satisfy the stated requirements within the identified constraints. Sixth, a security analysis was performed to evaluate the protocol against the threat model. Finally, the protocol's performance was evaluated through discrete-event simulation using OMNeT++.

4.2 System Model and Adversary Model

This section describes the environment in which the attestation protocol is intended to operate. The *System Model* characterizes the devices involved, the properties and limitations of the communication link between them, and the resulting constraints that any protocol running on this system must respect. The *Threat Model* then specifies the adversary against which the protocol is designed to be resilient, including the attacker's assumed capabilities and the attacks that fall within and outside the scope of this work. Together, the two models define both the design space available

to the protocol and the security properties it is expected to provide.

4.2.1 System Model

The goal is not to maximize the bandwidth, but rather to include communication in an already existing system while still allowing it to operate normally without being disturbed by the Radar-to-transponder communication. Hence, due to technical limitations, the potential of how much data can be included in the transmitted waves is limited. The leaf nodes are backscatter transponders which rather than generating their own radio frequency signal communicate by modulating and reflecting the carrier signal transmitted by the central node. The backscattered signal is able to carry more data than the carrier signal generated by the Radar system due to it not being limited by some normal operation. This results in an inherently asymmetric datalink with very low data transfer rates.

The system under consideration consists of a star network with a single powerful central node and multiple resource constrained leaf nodes, see Figure 4.1. The system is capable of a *downlink* communication rate, i.e. from the central node to the leaf node, of approximately 6 bps to 75 bps. The *uplink*, i.e. from the leaf node to the central node via backscatter, operates at a higher data rate of approximately 170 bps to 900 bps. The central node is a rotating Radar system capable of transmitting a carrier signal encoded with data to the leaf nodes. At a functional level, the Radar is capable of changing the f_{PRF} and the PL, which demodulated by the transponder can be interpreted as data bits depending on the chosen f_{PRF} and PL. Sequences of different PRFs can also be interpreted as a specific signal by the transponder.

4.2.2 Adversary Model

Subject to the trust assumptions stated in Section 4.2.3, the goal of an adversary is to inflict as much damage as possible on the system and network while remaining undetected. In line with other schemes [48, 31, 25, 7, 30, 4, 2] that consider an adversary capable of compromising the software and network, this thesis additionally considers an adversary capable of physically compromising the devices. Specifically, the adversary has the following capabilities:

- Eavesdrop on all wireless communication in both the downlink and uplink directions.
- Replay previously captured messages in an attempt to pass old attestation results as current.
- Inject and spoof messages on the communication link, including forged attestation responses.
- Jam the communication channel to disrupt the attestation process.
- Physically capture leaf nodes, given their deployment in hostile or unattended environments.

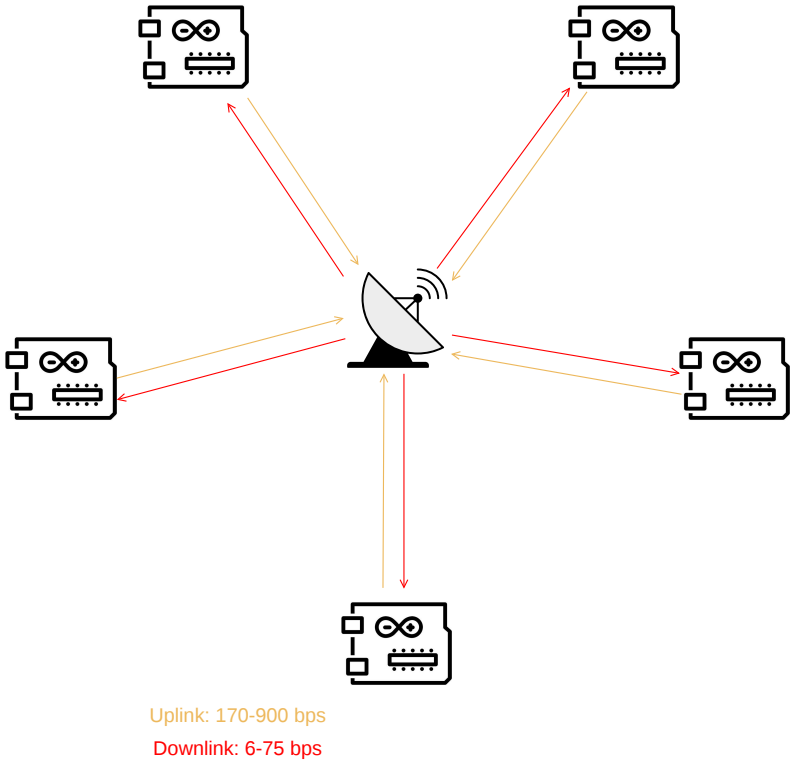


Figure 4.1: The figure displays a high level overview of the network topology and the available data rates over the datalink.

- Tamper with the firmware or software configuration of captured leaf nodes.
- Clone a captured leaf node and attempt to introduce the clone into the network.
- Relay messages between a legitimate leaf node and the central node, for instance to make a distant or displaced node appear to be in its expected location.

4.2.3 Assumptions

This subsection covers *trust* assumptions, *scope* assumptions about what the analysis does and does not cover, and *simplifications* adopted for the present work, are all collected in this subsection.

The following trust assumptions are made:

- The central node (Radar) is trusted. It is physically secured, and its integrity is not in question.
- The provisioning phase takes place in a trusted environment prior to deployment. Keys and credentials established during provisioning are assumed to be securely installed.

- Leaf nodes are considered untrusted after deployment. Once deployed in the field, any leaf node may have been compromised.

The following are out of scope:

- The technicalities of how the Radar encodes data onto the carrier signal are out of scope. The model treats the resulting data link only at the bit-rate level given in Section 4.2.1.
- Attacks targeting the physical layer beyond those listed in Section 4.2.2, such as side-channel attacks on the central node or exploitation of the node, are out of scope.

The following simplifications are made:

- Although each transponder could in principle communicate with multiple radars, only a single Radar is considered in this thesis.
- The link bit-rates given in Section 4.2.1 are taken from a parallel project at Saab AB and assumed to be representative of the deployment environment.
- The leaf nodes communicate only with the central node and have no ability to communicate with one another. This rules out the use of collective attestation protocols that rely on communication for aggregation or consensus.

The following assumptions and limitations describe the Tamarin model used to verify the cryptographic core of the challenge–response and verification phases of the protocol. The analysis covers only Phases 3 and 4 of ABLA under a Dolev–Yao adversary. Phase 0 provisioning is abstracted as a single key-creation rule, and Phases 1 and 2 are omitted entirely. HMAC-SHA-256 is represented as an uninterpreted function symbol satisfying EUF-CMA in the symbolic model, and the contents of the prover's memory are abstracted to a binary firmware state, good or bad, bound into a public digest function. Under these abstractions, Tamarin discharges key secrecy in the absence of an explicit reveal. It also discharges uniqueness of each verifier commit on its session tuple, and a minimal attestation-soundness property. Acceptance implies the responding prover held unmodified firmware. Carrier authentication, proximity verification, finite–counter wrap–around, side–channel attacks, physical capture beyond key reveal, memory contents beyond the binary firmware state, and implementation–level attacks fall outside the model.

4.3 Security Requirements

The following security requirements are derived from the threat model. Each requirement addresses one or more of the identified adversarial capabilities, and together they define the security properties that the protocol must provide.

1. Integrity: The protocol must detect unauthorized modification of a leaf node's firmware, software configuration, or boot configuration. This addresses the threat of physical tampering and software exploitation.

2. Authenticity: The protocol must verify the identity of each leaf node and prevent impersonation. This addresses the threats of spoofing, cloning, and message injection.

3. Freshness: The protocol must ensure that attestation results are current and cannot be replayed from a previous epoch.

4. Confidentiality: The protocol must protect the attestation exchange from eavesdroppers, preventing an adversary from learning the attestation state of leaf nodes or the internal structure of attestation messages. This addresses the eavesdropping threat.

5. Mobility awareness: The protocol must not rely on static location or a fixed network topology. It should detect physical displacement of leaf nodes and relay attacks. This addresses relay and displacement threats.

4.4 Constraints

This section will present constraints and assumptions made on the overall system model which will determine how the protocol is implemented and designed. Firstly, it will cover the communication rates, both up- and downlink, over the datalink. Secondly, limitations in the network topology and scale of the network will be set. Lastly, the hardware and its capabilities on both the Radar and the transponders will be set.

4.4.1 Datalink communication rates

The datalink transfer rates are, as mentioned under Section 4.2.1, 6 bps to 75 bps downlink and 170 bps to 900 bps uplink. For the sake of this thesis the rates will be assumed to be 20 bps downlink and 300 bps uplink. These values are picked as a valid estimate based on data from the company. The downlink will allow a smaller initialization message to be sent, as well as a message containing all necessary information to instantiate attestation can be sent to the transponder device. However, the low rate will inherently remove the capabilities of using any safe cryptography scheme or MACs on the message before transmitting it, hence, data will travel unencrypted. On the other hand, being able to operate at 300 bps, the uplink will be able to carry encrypted messages or MACs within a reasonable time frame.

4.4.2 Network topology and network scale

The topology as mentioned under Section 4.2.1 is of one-hop star constellation consisting of one central node, and several leaf nodes. The number of leaf nodes will be limited in this thesis, with the maximum amount of nodes greatly less than what is presented in other papers such as those presented under Section 2.

4.4.3 Hardware

An Arduino nano with an ESP32 [6] will be assumed to be used for the system. However, testing on hardware will not be conducted within this study itself. Rather the results achieved from this study, using different simulation methods, can be compared to what is theoretically possible on an Arduino nano with an ESP32, see Section 4.7.

4.5 ABLA

This section presents the design of our protocol *Asymmetric Backscatter Lightweight Attestation protocol* (ABLA). The protocol consists of five phases: provisioning (Section 4.5.1), carrier authentication (Section 4.5.2), proximity verification (Section 4.5.3), challenge–response attestation (Section 4.5.4), and verification (Section 4.5.5). Sections 4.5.1 through 4.5.3 lay the trust foundation, while Section 4.5.4 and 4.5.5 constitute the core attestation cycle. All cryptographic operations on the leaf nodes are symmetric, HMAC-SHA-256 is used for authentication, and SHA-256 for integrity measurement. Table 4.1 summarizes the notation used throughout this section.

Table 4.1: Notation used in the protocol specification.

Symbol	Meaning
V	Verifier (Radar), trusted
P_i	Prover (backscatter transponder) with identifier $i \in \{0, \ldots, 31\}$
K_i	Symmetric key, shared between V and P_i, distributed offline
K_{bc}	Key shared between all backscatter transponder devices
CH	Challenge nonce, 5 bits, drawn at random by V per attestation
TS_v	Verifier-side timestamp, 6 bits, monotonic counter local to V
TR	Trigger / coverage selector, 4 bits
SW_i	Software / memory image of P_i
B_i	Ordered set of attestable memory blocks of P_i, $\lvert B_i \rvert = m$
$S \subseteq B_i$	Subset of blocks selected for attestation in a given round
D_i	The hash of the concatenation of selected blocks
H	SHA-256
HMAC	HMAC-SHA-256
PRF	Keyed pseudorandom function
Att_{req}	Attestation request, 20 bits
Att_{res}	Attestation response, 256 bits

4.5.1 Phase 0: Provisioning

Provisioning is performed in a trusted environment prior to deployment, see Figure 4.2. During this phase, cryptographic material and reference measurements are established for each leaf node P_i. For each device P_i, the following values are

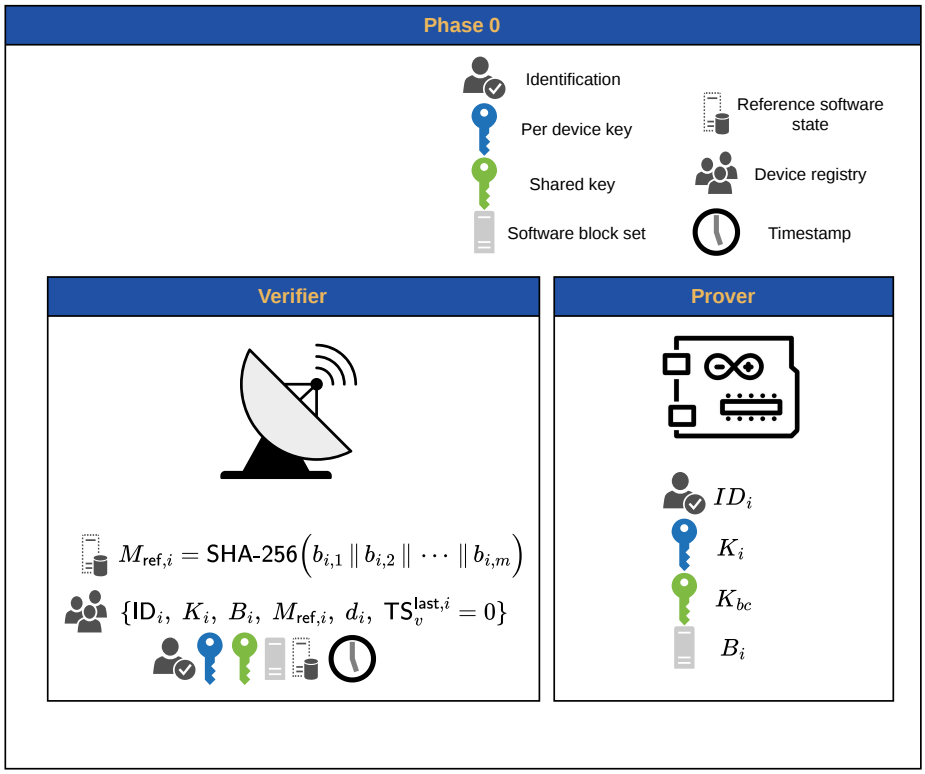


Figure 4.2: Phase 0: Provisioning illustration

generated:

$$\text{ID}_i \leftarrow \{0,1\}^5 \tag{4.1}$$

$$K_i \xleftarrow{\$} \{0,1\}^{256} \tag{4.2}$$

$$K_{\text{bc}} \xleftarrow{\$} \{0,1\}^{256} \tag{4.3}$$

where $\xleftarrow{\$}$ denotes uniform random sampling. The unique transponder identifiers ID_i are chosen to be distinct to avoid collision. The shared key K_{bc} is identical across all leaf nodes and is generated once. The per-device key K_i is used for both the attestation HMAC and (where applicable) derivation of the coverage selector seed via PRF_{K_i}.

A reference measurement is computed over the known-good software state of each device, partitioned into the ordered block set $B_i = (b_{i,1}, \ldots, b_{i,m})$:

$$M_{\text{ref},i} = \text{SHA-256}\left(b_{i,1} \,\|\, b_{i,2} \,\|\, \cdots \,\|\, b_{i,m}\right) \tag{4.4}$$

The block partitioning is identical at the prover and the verifier and is fixed at provisioning time. The verifier additionally stores the block-level reference values needed to recompute D'_i for any TR-selected subset (see Section 4.5.4).

Each leaf node P_i is provisioned with $\{\mathsf{ID}_i, K_i, K_{\mathsf{bc}}, B_i\}$, stored in secure storage (e.g., eFuse-protected or flash-encrypted memory). The total secure key storage required per device is 517 bits: 512 bits of key material and a 5-bit identifier.

The central node $\mathcal{V}$ maintains a device registry containing, for each device:

$$\{\mathsf{ID}_i,\ K_i,\ B_i,\ M_{\mathsf{ref},i},\ d_i,\ \mathsf{TS}_v^{\mathsf{last},i} = 0\}$$

along with the global key K_{bc}. The value $\mathsf{TS}_v^{\mathsf{last},i}$ is the highest verifier timestamp accepted from P_i and is used for replay protection.

4.5.2 Phase 1: Carrier Authentication

Carrier authentication prevents leaf nodes from accepting a rogue carrier signal, i.e. a signal from an unauthorized Radar. The central node $\mathcal{V}$ broadcasts an authenticated carrier signal each epoch, and leaf nodes validate the signal before initiating any BC.

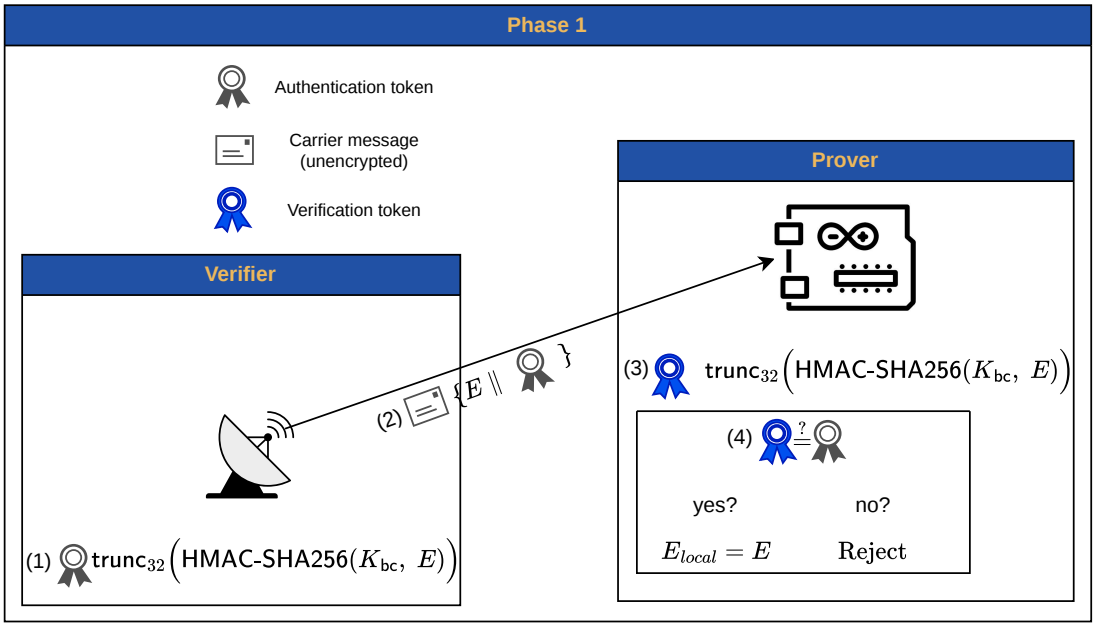


Figure 4.3: Phase 1: Carrier authentication illustration

See Figure 4.3 for an illustration of the steps conducted during this phase. Phase (1): At the beginning, before communication is initiated, $\mathcal{V}$ computes an authentication token:

$$\mathsf{token} = \mathsf{trunc}_{32}\Big(\mathsf{HMAC\text{-}SHA256}(K_{\mathsf{bc}},\ E)\Big) \tag{4.5}$$

where $\mathsf{trunc}_{32}(\cdot)$ denotes truncation to the 32 most significant bits, and E is a 16-bit counter that increments every T_{epoch} seconds. The carrier payload is:

$$\mathsf{carrier} = \{E \parallel \mathsf{token}\} \tag{4.6}$$

with a total size of 48 bits, requiring 2.4 s at 20 bps. This payload is broadcast continuously and updated when the epoch increments.

Upon receiving the carrier signal (3), leaf node P_i performs the following verification:

$$\mathsf{trunc}_{32}\Big(\mathsf{HMAC\text{-}SHA256}(K_{\mathsf{bc}},\ E)\Big) \stackrel{?}{=} \mathsf{token} \tag{4.7}$$

If the token verifies (4), P_i additionally checks that the epoch is fresh, $E > E_{\mathsf{local}}$, before accepting the carrier as legitimate and updating its local epoch $E_{\mathsf{local}} \leftarrow E$. The freshness check is necessary because the HMAC authenticates the value E but does not by itself bind the carrier to the present moment: a previously valid $\{E \,\|\, \mathsf{token}\}$ pair would otherwise verify correctly if replayed. Storing the last accepted epoch and rejecting any $E \leq E_{\mathsf{local}}$, or equivalently accepting only epochs within a forward window of the last accepted value, closes this gap. As E is a 16 bit monotonic counter, wrap-around is remote over realistic deployment lifetimes, and where it is a concern it is handled as for the verifier timestamp in Phase 4.5.5. If either the token check or the freshness check fails, P_i ignores the signal and does not respond to subsequent challenges, thereby preventing interaction with a rogue central node. The carrier authentication protocol is summarized in Figure 4.4.

Central node $\mathcal{V}$:

(1) $\mathsf{token} \leftarrow \mathsf{trunc}_{32}(\mathsf{HMAC\text{-}SHA256}(K_{\mathsf{bc}}, E))$

(2) $\mathcal{V} \xrightarrow{E \,\|\, \mathsf{token}\ \ (48\ \text{bits})} P_i$

Leaf node P_i :

(3) $\mathsf{token}' \leftarrow \mathsf{trunc}_{32}(\mathsf{HMAC\text{-}SHA256}(K_{\mathsf{bc}}, E))$

(4) **if** $\mathsf{token}' = \mathsf{token}$ **and** $E > E_{\mathsf{local}}$ **then** $E_{\mathsf{local}} \leftarrow E$
else reject carrier

Figure 4.4: Phase 1: Carrier authentication.

4.5.3 Phase 2: Proximity Verification

Since the communication link is between a Radar as interrogator and a backscatter transponder as the interrogated, each transponder is associated with a range and a bearing once they are detected, see Figure 4.5. Hence, the proximity of a transponder will be extractable, and an adversary trying to impersonate, produce relay attacks, or physically displace the transponder, renders the resulting anomaly detectable, subject to assumptions made below. This verification will not require any extra data to be transmitted from the Radar, and will be verifiable internally at the Radar.

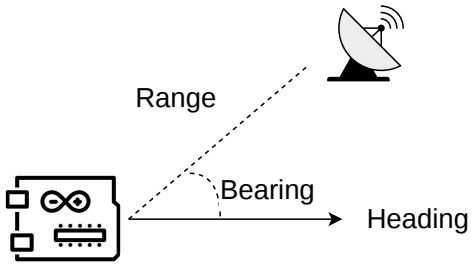


Figure 4.5: The figure illustrates the theoretical range and bearing between two objects, a Radar and a backscatter transponder.

See Figure 4.6 for an illustration of the proximity verification step. Upon detecting a transponder (1), the Radar obtains its range d_i and bearing θ_i directly from the echo (2). These two quantities are estimated internally by the signal processing chain and require no additional data exchange with the transponder. The central node retrieves the distance d_i and the bearing from the return echo s_{rx} of the transponder (4). It then calculates the delta of the measured bearing and the pointing direction of the main beam (4):

$$(d_i,\, \theta_i) = \mathsf{Localize}(s_{\mathsf{rx}}) \tag{4.8}$$

$$\Delta\theta = |\theta_i - \theta_{\mathsf{beam}}| \tag{4.9}$$

where s_{rx} is the received backscatter signal and θ_{beam} is the pointing direction of the main beam.

The central node evaluates the proximity result according to the following decision logic (5):

$$\mathsf{ProximityResult} = \begin{cases} \text{PASS} & \text{if } d_i \leq d_{\max} \text{ and } \Delta\theta \leq \tau_\theta \\ \text{SUSPICIOUS} & \text{if } d_i > d_{\max} \text{ and } \Delta\theta \leq \tau_\theta \\ \text{ANOMALOUS} & \text{if } \Delta\theta > \tau_\theta \end{cases} \tag{4.10}$$

where $d_{\max}$ is the maximum plausible range for a legitimate transponder and τ_θ is a configurable angular tolerance threshold. A SUSPICIOUS result, where the response originates within the main beam but at an implausibly long range, indicates a possible relay attack or physical displacement of the device. An ANOMALOUS result, where the response arrives from outside the main beam, indicates a spoofed return generated by an adversary that knows the waveform and transponder response but cannot place itself on the legitimate bearing.

Additionally, the following conditions are checked:

- Clone detection: If two valid responses are received for the same ID_i from distinct (d_i, θ_i) positions within a short time window, a CLONE_DETECTED

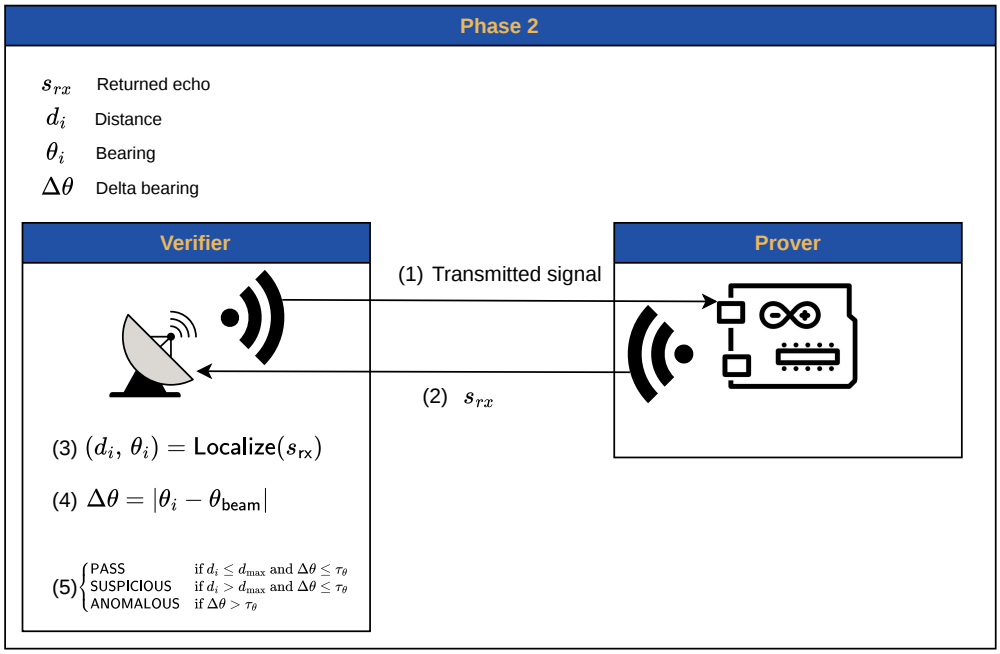


Figure 4.6: Phase 2: Proximity verification illustration

event is raised, indicating two physical devices operating with the same identity.

- Movement anomaly: If the range d_i or bearing θ_i changes faster than a plausible movement rate across successive Radar scans, a MOVEMENT_ANOMALY is flagged.

The guarantees provided here are those of a radar-based proximity check and movement-anomaly detection, not those of cryptographic distance bounding. Their soundness rests on a few assumptions. First, a relay or displacement attack cannot be mounted without introducing additional delay. I.e. the legitimate transponder applies a fixed, calibrated processing delay, so any relay or signal-processing path inserted by the adversary adds round-trip time that the Radar observes as an increase in apparent range d_i beyond the range resolution of the Radar. Second, the range and bearing estimates are accurate to within the range and angular resolution of the Radar, and the thresholds $d_{\max}$ and τ_θ are set relative to these resolutions. Finally, each device is associated with an expected position. An adversary able to relay with delay below the range resolution, or to place a spoofed return within both the expected range gate and the main beam, is not detected by this mechanism. The proximity verification protocol is summarized in Figure 4.7.

Central node $\mathcal{V}$ (Radar):
Transmit interrogation waveform

(1) $\mathcal{V} \xrightarrow{\text{waveform}} P_i$

Leaf node P_i (transponder):
Backscatter modulated response

(2) $P_i \xrightarrow{\text{backscatter response}} \mathcal{V}$

Central node $\mathcal{V}$:

(3) $(d_i, \theta_i) \leftarrow \mathsf{Localize}(s_{\mathsf{rx}})$

(4) $\Delta\theta \leftarrow |\theta_i - \theta_{\mathsf{beam}}|$

(5) Evaluate according to Equation 4.10

Figure 4.7: Phase 2: Proximity verification.

4.5.4 Phase 3: Challenge–Response Attestation

In the challenge–response attestation phase, leaf node P_i receives the 20 bit challenge message from $\mathcal{V}$ (Table 4.2), computes an integrity measurement over the subset of memory blocks selected by the trigger field TR, and returns a 256 bit authentication tag bound to the challenge, timestamp, and trigger. This phase is initiated by the central node and does not complete within a single illumination, rather under two illuminations. The first illumination will allow the Radar to transmit the challenge, and the other illumination will allow the transponder to send the response back, see Figure 4.8 for an illustration.

Table 4.2: Radar to Transponder challenge message format.

Field	Name	Size (bits)	Description
ID	Target device ID	5	Addresses the transponder being interrogated
CH	Challenge	5	Random challenge value sent by the Radar
TS_v	Timestamp	6	Current timestamp registered by the Radar
TR	Trigger	4	Coverage selector
Total		**20**	

Before sending the challenge to the transponder P_i (1), the verifier V generates a challenge consisting of 5 bit, it reads the current timestamp TS_v, and chooses a trigger TR.

The Radar sends $\mathsf{ID} \,\|\, \mathsf{CH} \,\|\, \mathsf{TS}_v \,\|\, \mathsf{TR}$ (2), the leaf node checks $\mathsf{ID} \neq \mathsf{ID}_i$ (3). If the identifier does not match, the message is silently dropped. If $\mathsf{TR} = \texttt{1111}$ (4), the

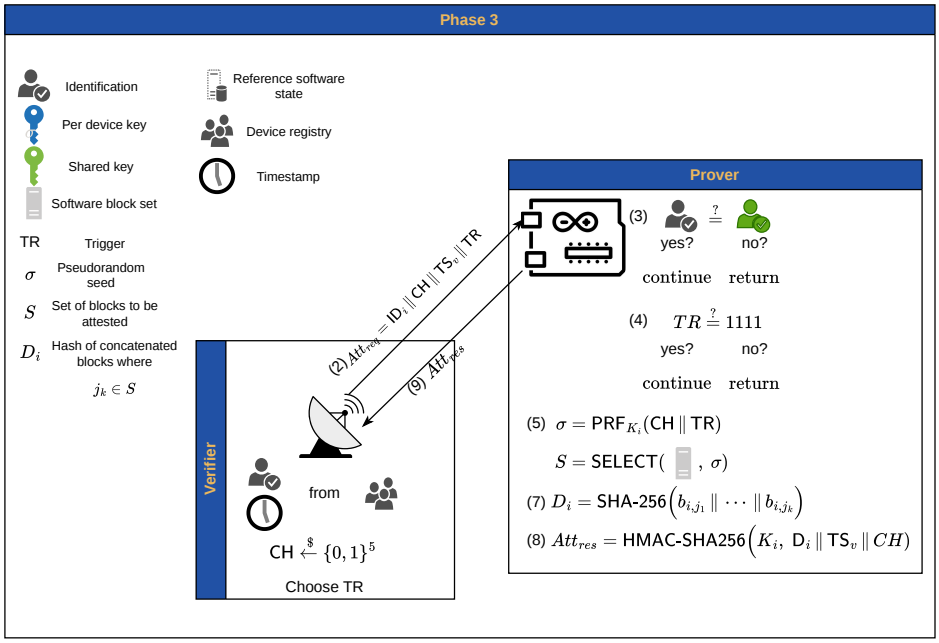


Figure 4.8: Phase 3: Challenge–response attestation step illustration.

leaf node performs no attestation and does not transmit a response but the message is not dropped.

The set $S \subseteq B_i$ of blocks to be attested is determined by TR:

$$S = \begin{cases} B_i & \text{if } \mathsf{TR} = 0000 \\ \mathsf{SELECT}(B_i,\ \sigma) & \text{otherwise, with } \sigma = \mathsf{PRF}_{K_i}(\mathsf{CH} \,\|\, \mathsf{TR}) \end{cases} \tag{4.11}$$

where SELECT$(\cdot,\cdot)$ is a deterministic, publicly specified function that, given the block layout B_i and a pseudorandom seed σ, returns a subset of fixed cardinality $|S| = k$ (a system parameter, e.g. $k = \lceil m/8 \rceil$) (6). PRF_{K_i} is instantiated as HMAC-SHA256 (5) truncated as required by SELECT.

The leaf node then computes the digest over the selected blocks in canonical order (7):

$$D_i = \mathsf{SHA\text{-}256}\big(b_{i,j_1} \,\|\, b_{i,j_2} \,\|\, \cdots \,\|\, b_{i,j_k}\big), \qquad \{j_1, \ldots, j_k\} = S \tag{4.12}$$

Thereafter, the 256 bit attestation response is computed (8) as:

$$Att_{res} = \mathsf{HMAC\text{-}SHA256}\big(K_i,\ \mathsf{ID}_i \,\|\, \mathsf{CH} \,\|\, \mathsf{TS}_v \,\|\, \mathsf{TR} \,\|\, \mathsf{D}_i\big) \tag{4.13}$$

Including ID_i and TR explicitly in the tag binds the response to the addressed prover and to the coverage selector, rather than relying on TR being implicitly determined through D_i. This also matches the inputs of the digest function used in the symbolic model (Appendix A).

The 256 bit key provides 128 bit security against quantum key recovery via Grover's algorithm. Resistance to forgery rests on HMAC-SHA-256 being a secure (EUF-CMA) message authentication code: without knowledge of K_i, an adversary cannot compute a valid tag Att_{res} for a fresh challenge–response transcript, even after observing arbitrarily many valid $(challenge, Att_{res})$ pairs.

Lastly, the response Att_{res} is transmitted via backscatter back to the verifier (9):

$$P_i \to \mathcal{V}: \quad Att_{res} \tag{4.14}$$

At the uplink rate of 300 bps, the 256 bit response requires approximately 0.85 s to transmit. The device identifier is not transmitted in the response: the verifier already addressed the prover and knows which K_i to use for verification within the illumination.

The attestation protocol is summarized in Figure 4.9.

Central node $\mathcal{V}$:

(1) $\mathsf{CH} \xleftarrow{\$} \{0,1\}^5$, read TS_v, choose TR

(2) $\mathcal{V} \xrightarrow{\mathsf{ID}_i \,\|\, \mathsf{CH} \,\|\, \mathsf{TS}_v \,\|\, \mathsf{TR} \;\; (20 \text{ bits})} P_i$

Leaf node P_i :

(3) **if** $\mathsf{ID} \neq \mathsf{ID}_i$: drop

(4) **if** $\mathsf{TR} = \mathtt{1111}$: halt

(5) $\sigma \leftarrow \mathsf{PRF}_{K_i}(\mathsf{CH} \,\|\, \mathsf{TR})$ (omit if $\mathsf{TR} = \mathtt{0000}$)

(6) $S \leftarrow \mathsf{SELECT}(B_i, \sigma)$ (or B_i if $\mathsf{TR} = \mathtt{0000}$)

(7) $D_i \leftarrow \mathsf{SHA\text{-}256}\big(b_{i,j_1} \,\|\, \cdots \,\|\, b_{i,j_k}\big)$

(8) $Att_{res} \leftarrow \mathsf{HMAC\text{-}SHA256}(K_i,\ \mathsf{ID}_i \,\|\, \mathsf{CH} \,\|\, \mathsf{TS}_v \,\|\, \mathsf{TR} \,\|\, \mathsf{D}_i)$

(9) $P_i \xrightarrow{Att_{res} \;\; (256 \text{ bits})} \mathcal{V}$

Figure 4.9: Phase 3: Challenge–response attestation.

4.5.5 Phase 4: Verification

Upon receiving the response Att_{res} (0) within the expected uplink slot of the illumination addressed to ID_i, the central node $\mathcal{V}$ performs the following verification procedure seen in Figure 4.10.

The first step involves retrieving all necessary information from the registry (1). The freshness check is the initial check the verifier does. The verifier checks that the timestamp issued in the challenge has not been previously accepted from this prover (2):

$$\mathsf{TS}_v > \mathsf{TS}_v^{\mathsf{last},i} \tag{4.15}$$

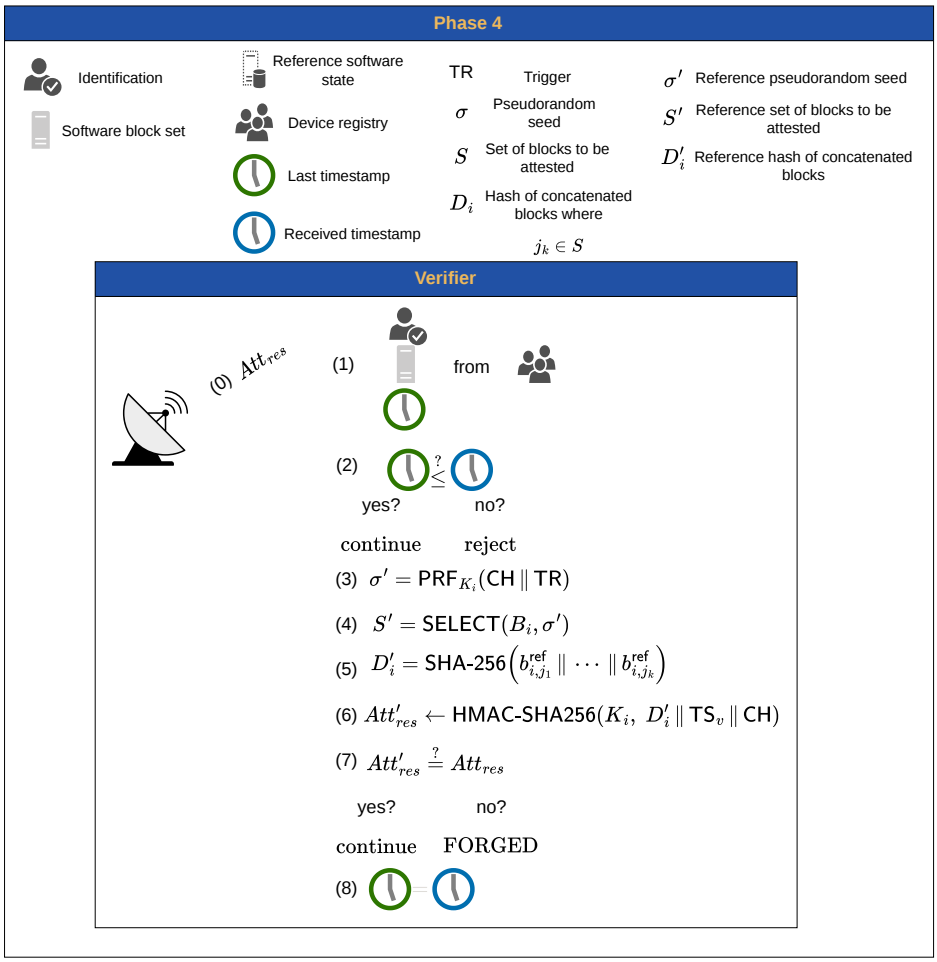


Figure 4.10: Phase 4: Verification step illustration

The 6-bit verifier timestamp wraps every $2^6 = 64$ accepted attestations from a given prover. The strict inequality $\mathsf{TS}_v > \mathsf{TS}_v^{\mathsf{last},i}$ is therefore interpreted modulo 2^6: a candidate TS_v is accepted if and only if $(\mathsf{TS}_v - \mathsf{TS}_v^{\mathsf{last},i}) \bmod 2^6 \in \{1, \ldots, W\}$ for a sliding-window parameter $W \leq 32$. In deployments where the per-prover attestation rate risks wrap-around within the operational window, the field width is increased rather than the window relaxed. If the check fails, the response is rejected as a replay. Provided the per-prover attestation rate does not exhaust the 2^6 counter values within the operational window, no wrap-around occurs and the modular test reduces to the simple comparison $\mathsf{TS}_v > \mathsf{TS}_v^{\mathsf{last},i}$ against the last accepted value for P_i. The pseudocode in Figure 4.11 shows this non-wrapping case; when wrap-around is in scope, step (2)'s plain comparison is replaced by the modular window test $(\mathsf{TS}_v - \mathsf{TS}_v^{\mathsf{last},i}) \bmod 2^6 \in \{1, \ldots, W\}$ given above.

Using the registry entry for P_i, the verifier computes the same pseudorandom seed σ' (3), then the verifier reconstructs the same subset S that the prover would have selected (4):

$$S' = \begin{cases} B_i & \text{if } \mathsf{TR} = \mathsf{0000} \\ \mathsf{SELECT}(B_i,\ \sigma') & \text{otherwise, with } \sigma' = \mathsf{PRF}_{K_i}(\mathsf{CH} \,\|\, \mathsf{TR}) \end{cases} \tag{4.16}$$

Thereafter, using the stored block-level reference values for P_i, the verifier computes (5):

$$D_i' = \mathsf{SHA\text{-}256}\Big(b_{i,j_1}^{\mathsf{ref}} \,\|\, b_{i,j_2}^{\mathsf{ref}} \,\|\, \cdots \,\|\, b_{i,j_k}^{\mathsf{ref}}\Big), \qquad \{j_1, \ldots, j_k\} = S' \tag{4.17}$$

The verifier then recomputes the expected response and compares it to the received attestation response (6):

$$Att_{res}' = \mathsf{HMAC\text{-}SHA256}\Big(K_i,\ \mathsf{ID}_i \,\|\, \mathsf{CH} \,\|\, \mathsf{TS}_v \,\|\, \mathsf{TR} \,\|\, \mathsf{D}_i'\Big) \stackrel{?}{=} Att_{res} \tag{4.18}$$

If Att_{res}' and Att_{res} do not match (7), the report is rejected, indicating either tampering, a software-state divergence from reference software state $M_{\mathsf{ref},i}$, or a forgery attempt.

The acceptance predicate $Accept_i(\mathsf{CH}, \mathsf{TS}_v, \mathsf{Att}_{res})$ holds if and only if all of (4.15), the reconstruction of (4.16)–(4.17), and the equality (4.18) succeed.

If the response comparison in Equation 4.18 succeeds, the verifier commits the accepted timestamp (8):

$$\mathsf{TS}_v^{\mathsf{last},i} \leftarrow \mathsf{TS}_v \tag{4.19}$$

This update sets the freshness check in (1) of subsequent rounds and ensures that any later replay of Att_{res} bound to the same TS_v is rejected as REPLAYED.

The latest timing result for ID_i from Phase 2 is retrieved and combined with the cryptographic verification result (9).

The final trust state is determined by combining the results of all verification steps, as summarized in Table 4.3 (10). The complete verification procedure is summarized in Figure 4.11.

Table 4.3: Verifier trust decision logic, each column is one specific verdict and each row a check performed during verification.

Signal / Criterion	TRUSTED	FORGED	REPLAYED	DESYNCED	DISPLACED	CLONED
MAC valid	✓	✗	✓	✗	✓	✓
Freshness	✓	✓	✗	✗	✓	✓
Timing in bounds	✓	—	—	—	✗	✗
Unique ID	✓	✓	✓	✓	✓	✗

Central node $\mathcal{V}$ receives Att_{res} on the slot for ID_i :

(1) Look up $\{K_i,\ B_i,\ \mathsf{TS}_v^{\mathsf{last},i}\}$ from the registry

(2) **if** $\mathsf{TS}_v \le \mathsf{TS}_v^{\mathsf{last},i}$: reject as REPLAYED

(3) $\sigma' \leftarrow \mathsf{PRF}_{K_i}(\mathsf{CH} \,\|\, \mathsf{TR})$ (omit if $\mathsf{TR} = \mathtt{0000}$)

(4) $S' \leftarrow \mathsf{SELECT}(B_i, \sigma')$ (or $S' \leftarrow B_i$ if $\mathsf{TR} = \mathtt{0000}$)

(5) $D_i' \leftarrow \mathsf{SHA\text{-}256}\big(b_{i,j_1}^{\mathsf{ref}} \,\|\, \cdots \,\|\, b_{i,j_k}^{\mathsf{ref}}\big),\ \{j_1, \ldots, j_k\} = S'$

(6) $Att'_{res} \leftarrow \mathsf{HMAC\text{-}SHA256}(K_i,\ \mathsf{ID}_i \,\|\, \mathsf{CH} \,\|\, \mathsf{TS}_v \,\|\, \mathsf{TR} \,\|\, D_i')$

(7) **if** $Att'_{res} \neq Att_{res}$: reject as FORGED

(8) $\mathsf{TS}_v^{\mathsf{last},i} \leftarrow \mathsf{TS}_v$

(9) Cross-check with the latest Phase 2 timing result for ID_i

(10) Emit trust decision per Table 4.3

Figure 4.11: Phase 4: Verification procedure at the central node.

4.5.6 Communication Overhead Summary

Table 4.4 summarizes the communication overhead of each protocol phase.

Table 4.4: Communication overhead per protocol phase.

Phase	**Direction**	**Size (bits)**	**Time**
1: Carrier auth.	$\mathcal{V} \to P_i$	48	2.4 s (20 bps)
2: Proximity	$\mathcal{V} \rightleftharpoons P_i$	0	negl
3: Challenge	$\mathcal{V} \to P_i$	20	1.0 s (20 bps)
3: Response	$P_i \to \mathcal{V}$	256	0.85 s (300 bps)
Total per cycle		≤ 324	≤ 4.25 s

The total attestation overhead per device per cycle is at most 324 bit, with no per-

device downlink payload beyond the broadcast carrier signal. This compares favorably with existing protocols: RESERVE reports 500 bits of overhead [47], ROSTER reports 536 bits [69], and the authentication protocol in [27] requires 1172 bits. Critically, the protocol achieves this with only 48 bits on the constrained downlink, and these 48 bits are broadcast once to all nodes rather than sent individually.

4.6 Security Analysis

This section examines the security of the proposed protocol against the threats identified in the threat model. The analysis is organized by attack class rather than by protocol phase, since a single adversary capability may bear on several phases at once. Each subsection states the assumptions made about the adversary, identifies which protocol mechanisms are relevant, and assesses the extent to which the threat is prevented, detected, or merely contained. Throughout, a deliberate distinction is drawn between cryptographic and operational guarantees. Cryptographic guarantees hold by virtue of the underlying primitives and the secrecy of key material. Operational guarantees arise from the physical properties of the Radar channel or from the correlation of events across phases. The protocol does not claim to prevent every attack outright; in several cases, particularly hardware compromise and jamming, the design goal is to bound the consequences of a successful attack and to surface it through complementary detection signals rather than to exclude it entirely. The subsections that follow address hardware attacks, software compromise, replay, spoofing, eavesdropping, and denial-of-service through jamming in turn. The section also covers the security of the underlying cryptography, i.e. the properties proven in the Tamarin Prover.

4.6.1 Security against hardware attacks

The protocol assumes that the per-device key K_i, the shared carrier key K_{bc}, and the reference block layout B_i are stored in protected memory. This provides resistance against software-level extraction and physical inspection, but does not constitute tamper resistance: an adversary with sufficient laboratory capability is assumed to be able to extract key material from a captured device. The protocol is therefore designed to bound the consequences of single-device compromise rather than prevent it.

Because K_i is unique to P_i, extraction of K_i from a single captured transponder does not enable forgery of attestation responses for any other prover P_j, $j \neq i$. The verifier's registry isolates devices: a compromised K_i allows the adversary to produce valid responses only for ID_i, and only until the operator revokes the entry $\{\mathsf{ID}_i, K_i, B_i, M_{\mathsf{ref},i}, d_i, \mathsf{TS}_v^{\mathsf{last},i}\}$ from the registry.

The key K_{bc} is replicated across all provers and is therefore the most attractive single target for hardware extraction. Its scope is, however, limited to the carrier authentication phase (Phase 4.5.2): an adversary holding K_{bc} can fabricate a valid epoch token and induce provers to accept a rogue carrier, but cannot produce valid attestation responses, which require the per-device key K_i. Mitigation requires global

re-keying of K_{bc}, which is performed offline through the same trusted provisioning used in Phase 4.5.1. An alternative is to derive K_{bc} online via an authenticated key-exchange protocol, eliminating the long-lived shared secret at the cost of additional rounds and asymmetric operations per epoch; this trade-off is left to future work.

A prover that has been physically compromised holds K_i and can therefore compute

$$\mathsf{HMAC\text{-}SHA256}(K_i,\ \mathsf{ID}_i \,\|\, \mathsf{CH} \,\|\, \mathsf{TS}_v \,\|\, \mathsf{TR} \,\|\, \mathsf{D}_i)$$

for any digest D_i of its choosing. Software attestation alone cannot distinguish a compromised prover that reports $D_i = D_i^{\mathsf{ref}}$ over malicious memory contents from an honest prover that reports D_i^{ref} over genuine memory. The verifier-driven coverage selector TR (Section 4.5.4) raises the cost of *Time–Of–Check Time–Of–Use* (TOCTOU)–style evasion by preventing the prover from anticipating which subset S will be attested, but it does not by itself defeat a fully compromised device. Stronger guarantees in this area require execution-time bounds (as in SWATT [55]) or hardware-rooted attestation primitives (e.g. a root-of-trust producing the digest), neither of which is assumed by ABLA.

Although hardware compromise of an individual prover cannot be ruled out cryptographically, several phases of the protocol provide indirect detection. Phase 4.5.3 (proximity verification) detects relay attacks and device cloning through the range and bearing recovered from the backscattered return, which are physical properties of the propagation geometry and not under the adversary's cryptographic control. The CLONE_DETECTED event in particular flags the case where a compromised device has been duplicated and operated alongside the original. Section 4.6.6 (liveness monitoring) flags devices that cease producing valid responses, including those whose keys have been erased or whose firmware has been corrupted by an unsuccessful extraction attempt. Operational correlation across these signals provides defense-in-depth that a single phase does not.

4.6.2 Security against software compromises

The devices will be vulnerable to malicious attacks, such as malware or other changes that compromise the integrity of the software which is running on the devices. However, as previously mentioned, the protocol will utilize HMACs for the attestation response. These HMACs are compared against reference HMACs computed in the verification step, i.e., Phase 4. Any small changes in the software will be detectable if any of the input values in the HMAC function is modified, e.g. the memory blocks holding the software. In the symbolic model this is captured in the attestation_soundness lemma, see Appendix B. This lemma proves that an accepted response could only be produced by the addressed prover holding unmodified firmware.

4.6.3 Security against replay attacks

Replay attacks, in which an adversary delays or retransmits valid messages, could trick the verifier into believing the scheme has completed correctly. To prevent this, the protocol uses timestamped messages and unique challenges, which assumes a

monotonic counter and a source of randomly generated challenges. As mentioned, the verifier sends a timestamped message containing the challenge, the device id, and the trigger bits, which the device verifies before accepting the request. The device, in turn, includes a timestamp in its attestation report. The verifier accepts the report only if its timestamp falls within a predefined threshold; otherwise, the report is rejected as either invalid or a suspected replay attack. Replay attacks are also protected against through the functionality of the Radar.

4.6.4 Security against spoofing

Through the use of symmetric key cryptography integrity is ensured by allowing both parties in the network, that is the transponder and the Radar, to authenticate themselves. The adversary will not be able to send spoofed data without having the correct key. Furthermore, where the specific setup of the Radar and BC transponder is used the ability for the Radar to authenticate itself to the BC transponder is possible. The Radar can select from a set of f_{PRF} and transmit them in a predetermined sequence; a transponder pre-programmed with that sequence will respond only to radars that reproduce it.

4.6.5 Security against eavesdropping

Since the system will be run in a hostile environment without supervision, the possibility of malicious actors listening in on the communication is likely. Hence, the risk of eavesdropping becomes an issue. Under standard assumptions, the messages sent from the transponders will not reveal the key or allow forgery, this is because HMAC. However, this is a HMAC-SHA-256 requiring 256 bit, which is only possible to transmit in uplink and all messages transmitted in downlink will be in plain sight. Therefore, an adversary could interpret the message from the Radar and gain knowledge that an attestation request is sent. It could also acquire metadata such as timestamp, id.

4.6.6 Liveness and Jamming Detection

The protocol does not provide cryptographic protection against denial-of-service via RF jamming, which is outside the threat model of channel-layer attestation. Instead, the verifier monitors prover liveness: each prover P_i is expected to produce a valid attestation response within a sliding window of N revolutions. If no valid response is received from P_i within this window, $\mathcal{V}$ raises a LIVENESS_LOST event for that prover. This event indicates one of: (i) sustained jamming of the link to P_i, (ii) device failure or power loss, (iii) physical removal of the device, or (iv) a compromise that prevents the device from producing valid responses (e.g. key erasure, firmware corruption that breaks the attestation routine). The verifier cannot distinguish these cases from the radio signal alone; correlating LIVENESS_LOST with SUSPICIOUS or ANOMALOUS results from the Phase described under Section 4.5.3, or with simultaneous loss across multiple provers (suggesting jamming rather than per-device failure), provides operational disambiguation.

4.6.7 Freshness

The cryptographic core of the protocol is verified symbolically in the Tamarin Prover under a Dolev–Yao adversary. Among the properties established is freshness of the responses returned by the BC transponders, expressed through the `no_replay` lemma, see Appendix B.6. Tamarin proves that the timestamp TS_v accompanying each accepted response is unique to its session, so the verifier never commits to two distinct responses that share a (P, TS_v) pair. This establishes a necessary but not sufficient condition for replay resistance: the symbolic model represents timestamps as globally unique fresh names and therefore captures neither the 6-bit wrapping counter, the modular arithmetic, nor the sliding-window parameter W used by the implementation. The correctness of the wrap-around behavior must be argued separately.

4.7 Performance evaluation

This section evaluates the proposed attestation protocol along two complementary dimensions. The first concerns its operational performance: how the protocol behaves as the network grows in size, and what overhead it imposes in terms of communication and latency. The second concerns its security: whether the protocol actually achieves the guarantees it was designed to provide, in particular authentication of leaf nodes and secrecy of the attestation evidence. Because no single tool addresses both dimensions adequately, two distinct evaluation methodologies are employed. Operational performance is assessed empirically through discrete-event simulation in OMNeT++, which permits the protocol to be exercised over many network configurations in a controlled and reproducible setting. Security is assessed through formal verification with the Tamarin Prover, which establishes whether the desired properties hold against a symbolic adversary for an unbounded number of sessions. Each approach carries its own modelling assumptions and limitations, discussed in the respective subsections below, and the two are intended to be read as mutually reinforcing rather than redundant: simulation demonstrates that the protocol is practical to deploy, while the formal verification demonstrates that the cryptographic core of Phase 3 and 4 is sound under the symbolic model. It does not provide soundness of the full protocol. Phases 1 and 2 are out, side channels, physical capture, and implementation–level attacks lie outside the model.

4.7.1 OMNeT++

The performance of the attestation protocol is evaluated in the simulation tool OMNeT++ [66]. Since it is a simulation it does have some shortcomings, for example; it does not take computational times into consideration, all randomness is pseudo-random and therefore does not reflect true sources of entropy, and the modelled network conditions are necessarily a simplification of a real deployment. These limitations are accepted in exchange for the ability to evaluate the protocol over a wide range of network sizes and topologies in a reproducible manner. The results obtained are described in the next chapter.

4.7.2 Tamarin Prover

The authentication and freshness layer utilized within the protocol, together with a minimal attestation-soundness property over an abstract firmware state, are covered by the Tamarin Prover [34], and not the attestation protocol as a whole. Tamarin Prover is a tool for the symbolic analysis of cryptographic protocols. The protocol is specified as a multiset rewriting system, a formalism in which the global state is represented as a multiset of facts and labelled rewrite rules describe how this state evolves through protocol execution and adversary actions. The desired security properties are expressed as lemmas in a first-order logic over traces, where atomic formulas refer to events occurring at specific time points in an execution, allowing properties such as secrecy and authentication to be stated as quantified formulas that must hold for every possible trace. Tamarin then attempts to either prove these lemmas or refute them by producing a counterexample trace. As with any verification approach, this methodology has limitations: the analysis is carried out in the symbolic, Dolev–Yao model, a model which has unlimited power, and that idealizes cryptographic primitives as perfect black boxes and therefore abstracts away from attacks on the underlying cryptography itself. Since protocol verification is undecidable in general, Tamarin may fail to terminate on a given lemma and require manual proof guidance or auxiliary lemmas; and the adversary and protocol models are necessarily a simplification of any concrete deployment. These limitations are accepted in exchange for unbounded verification, meaning that a successful proof holds for an arbitrary number of protocol sessions and participants rather than for a finite set of test cases.

5

Results

The developed ABLA protocol has been simulated in the discrete event simulator OMNeT++. The simulation has been run through parameter sweeps covering Radar bitrate, response bitrate, re-attestation interval and bit error rate. The results from these simulations will be presented in this chapter. Furthermore, the cryptographic core of Phase 3 and 4 of the protocol has been verified in Tamarin Prover over six lemmas, the findings are presented in this chapter.

5.1 Simulation results

The ABLA attestation protocol is evaluated along two complementary axes: a network-level performance study in OMNeT++, and a symbolic security analysis in the Tamarin prover. The OMNeT++ study runs a rotating Radar deployment with ten legitimate transponders as a baseline over a 120 s simulation horizon. The baseline configuration uses a 20 bps challenge channel, a 300 bps response channel, and a reattestation interval of five rotations, and is then swept along five independent axes: challenge bitrate, response bitrate, reattestation interval, prover population, and channel bit-error rate. A separate attacker-scenarios configuration replaces approximately half of the population with rogue transponders provisioned with a key the verifier does not hold, exercising the FORGED classification path that the all-legitimate sweeps leave dormant. For each run the simulator records, per prover, the number of attestation cycles classified as TRUSTED, FORGED, REPLAYED, or DESYNCED, together with the round-trip time (RTT) of each cycle and the corresponding detection counts. The Tamarin analysis, in turn, models Phases 3 and 4, i.e., those described under Section 4.5.4 and Section 4.5.5, of the protocol under a Dolev–Yao adversary and discharges six lemmas covering sanity, key secrecy, non-injective agreement, commit uniqueness, freshness, and attestation soundness. The analysis covers the cryptographic core of the protocol and attestation soundness is established under a binary firmware–state abstraction rather than over actual memory contents. The remainder of this section reports the OMNeT++ outcomes for the baseline configuration first, then for each parameter sweep in turn, and concludes with the Tamarin verification results.

5.1.1 OMNeT++ Simulation assumptions

Table 5.1: Assumptions and parameters of the OMNeT++ simulation of ABLA. Baseline values are those held fixed while a single knob is swept; the swept range is given alongside each knob.

Parameter	Value
Radar geometry and rotation	
Verifier	single rotating Radar
Rotation period T_{rot}	60 RPM
Detection model	beam-pass illumination triggers a cycle
Illumination window	1 s
Node placement and topology	
Topology	one-hop star, single verifier
Population	10 baseline; sweep $\{3, 8, 16, 32\}$
Placement	random placement within boundary
Datalink and message sizes	
Challenge Att_{req}	20 bit
Response Att_{res}	256 bit
Challenge bitrate	20 bps baseline; sweep $\{10, 20, 40, 80\}$
Response bitrate	300 bps baseline; sweep $\{100, 300, 600, 1200\}$
Attestation schedule	
Reattestation interval	5 rotations baseline; sweep $\{1, 2, 5, 10\}$
Cycles per run (baseline)	40 (4 per prover $\times$ 10 provers)
Block selection	$\sigma = \text{HMAC}(K_i, CH \parallel TR)$, $k = 4$ blocks; $TR \in \{0, \ldots, 14\}$
Classification	
Outcome classes	TRUSTED, FORGED, REPLAYED, DESYNCED
Attacker configuration	$\approx$50% rogue provers holding a key V does not

5.1.2 OMNeT++

The baseline configuration in this subsection runs ten legitimate transponders against a single rotating verifier for 120 s of simulated time, at a challenge bitrate of 20 bps, a response bitrate of 300 bps, and a reattestation interval of five rotations. Under these settings each prover is attested four times within the run, yielding 40 attestation cycles in total. The five subsequent sweeps each vary one parameter at a time about this baseline, holding the others fixed. A separate attacker–scenarios run, described at the end of this subsection, replaces half of the provers with rogues to exercise the classification path. Before turning to the verification outcomes, it is useful to decompose where the per-cycle time is spent, since the per-cycle RTT is what bounds how aggressively the interval can be tightened. Each prover is verified multiple times over a run.

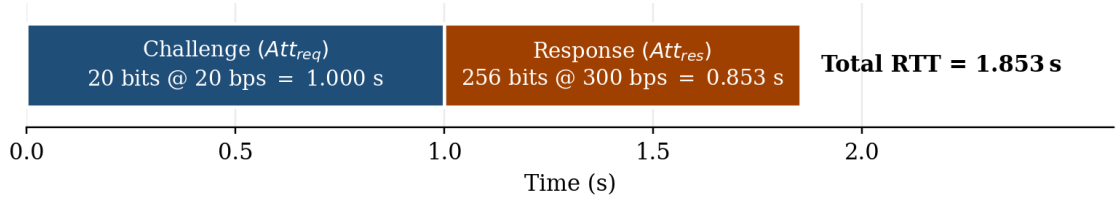


Figure 5.1: Communication time of one Phase 3 attestation cycle. The 20 bit challenge transmitted at 20 bps and the 256 bit response transmitted at 300 bps together account for the full 1.853 s RTT.

Over the 120 s baseline run, all 40 attestation cycles are classified TRUSTED; no FORGED, REPLAYED, or DESYNCED outcomes occur and the per-cycle RTT is 1.853 s in every cycle, see Figure 5.1. The attestation layer is triggered 241 times during the run by beam-pass illumination at the Radar layer, corresponding to roughly six beam–pass detections per attestation interval. The correspondence between detections, completed cycles, and trusted outcomes is examined more directly in the population sweep, see Figure 5.3. In general the number of total outcomes is approximately equal to (rounds per prover) × (number of provers)

To verify that the PRF-keyed block-selection path of Phase 3 agrees between prover and verifier for arbitrary trigger codes, the baseline was re-run with TR sampled uniformly from $\{0, \ldots, 14\}$ on every challenge. Outcome counts and RTT are unchanged from the $TR = 0$ baseline (40 trusted, 0 of all other classes, 1.853 s mean RTT over 241 detections), which confirms that the deterministic block-selection function $\sigma = \mathrm{HMAC}(K_i, CH \| TR)$ evaluated independently by prover and verifier yields the same $k = 4$ blocks in both cases.

Figure 5.2 reports the two bitrate sweeps, uplink and downlink, and the reattestation sweep in a single three-panel view, also see Table 5.2 and Table 5.3. The Radar (challenge) bitrate sweep uses the canonical population and shows a clean illumination-overrun shortfall at the slowest setting: at 10 bps the per-cycle RTT extends to 2.853 s and only 34 of the 40 expected cycles complete within the illumination window; from 20 bps upward all 40 cycles complete and the trusted curve sits exactly on the expected ceiling. The response (transponder) bitrate sweep is shown here at pop = 32, where the across-seed variance is at its smallest; the RTT story is identical, falling from 3.560 s at 100 bps to 1.213 s at 1200 bps, while the trusted count drops modestly from 308.6 at the baseline to 275.4 at the slowest rate. Across all bitrate values and all populations the FORGED, REPLAYED, and DESYNCED counts remain zero, so the shortfall at low bitrate manifests as missing cycles rather than misclassified ones.

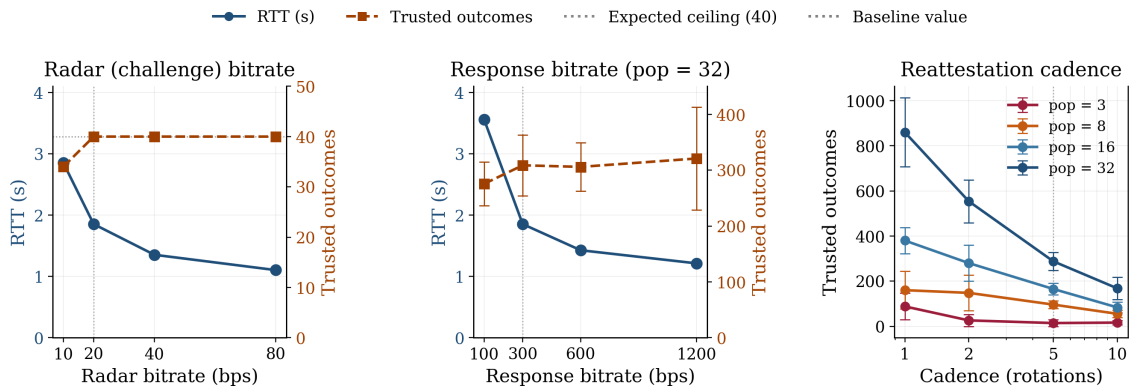


Figure 5.2: Parameter sweeps over the three operational parameters of the deployment. Each panel shows RTT (left axis, solid blue) and total trusted outcomes within the 120 s run (right axis, dashed rust). The vertical dotted line marks the baseline; the horizontal dotted line in the two bitrate panels marks the expected cycle ceiling of 40. Cycles that fail to complete within the illumination window appear as shortfalls below this ceiling at the slowest bitrate in each sweep.

Table 5.2: Radar (challenge) bitrate sweep. FORGED, REPLAYED, and DESYNCED outcome counts are zero in every row and are omitted. The shortfall at 10 bps corresponds to attestation cycles that fail to complete within the illumination window.

Radar rate (bps)	Trusted	Shortfall	RTT (s)
10	34	−6	2.853
20	**40**	**0**	**1.853**
40	40	0	1.353
80	40	0	1.103

Table 5.3: Response (transponder) bitrate sweep, trusted outcomes by population and bitrate (mean ± standard deviation across seeded replications). `FORGED`, `REPLAYED`, and `DESYNCED` outcome counts are zero in every cell. Per-cycle RTT depends only on bitrate and is reported in the bottom row.

Population	100 bps	300 bps	600 bps	1200 bps
3	28.8 ± 17.6	14.2 ± 14.5	15.4 ± 15.7	30.2 ± 17.1
8	53.8 ± 29.2	79.6 ± 43.0	101.4 ± 20.2	105.6 ± 28.9
16	121.0 ± 26.6	156.0 ± 44.3	174.2 ± 25.3	157.4 ± 44.8
32	275.4 ± 38.9	308.6 ± 54.5	305.8 ± 43.1	320.8 ± 92.2
RTT (s)	3.560	1.853	1.427	1.213

The right panel of Figure 5.2 plots trusted-outcome count against the interval for each of the four populations tested. The four curves are monotonically decreasing in reattestation interval, as expected: an interval of one reattests every rotation and

produces the maximum cycle count, while an interval of ten reattests every tenth rotation. The scaling between interval values is approximately, but not exactly, inverse-proportional; the deviation grows at smaller populations where the position-randomness effect (see below) dominates the variance budget. Table 5.4 reports the full 4×4 matrix. Per-cycle RTT is 1.853 s in every cell because the reattestation interval governs when attestation is scheduled, not how long an individual cycle takes.

Table 5.4: Reattestation interval × population with standard deviations.

Population	**Interval 1**	**Interval 2**	**Interval 5**	**Interval 10**
3	88.0 ± 59.3	26.0 ± 26.1	14.2 ± 14.5	16.2 ± 9.1
8	160.0 ± 83.7	148.0 ± 79.5	96.0 ± 16.3	55.2 ± 15.4
16	380.0 ± 58.3	280.0 ± 79.7	165.2 ± 25.1	83.6 ± 23.6
32	860.0 ± 153.0	554.0 ± 94.5	287.8 ± 40.8	168.0 ± 48.8

The dedicated population sweep, summarized in Table 5.5 and plotted in Figure 5.3, characterizes how trusted-outcome count scales with prover count at the baseline configuration. The two larger populations sit on a straight line with slope approximately 9.1 trusted outcomes per added prover, which captures the cycles-per-prover budget of the baseline; the two smaller populations sit visibly below this line and exhibit much larger relative variance, since with only a handful of provers a single seeded placement that falls outside the Radar's effective range removes a large fraction of the workload. The 95% confidence interval on the pop = 3 mean overlaps zero, which is the practical signature of this regime; deployments at small population should therefore not rely on the linear-scaling envelope predicted by larger populations. The absolute outcome counts reported across these sweeps follow from independent runs whose realized attestation schedule depends on the per-run Radar–prover geometry: a prover contributes completed cycles only while its randomly drawn placement keeps it within the Radar's effective range over the horizon, so the per-run workload is shaped jointly by the reattestation schedule and by placement. The counts are therefore best read as indicators of the scaling each sweep isolates rather than as exact, cross-comparable totals, a caution that applies most strongly at small populations, where a single placement seed can move a large fraction of the workload in or out of range and inflates the across-seed variance accordingly.

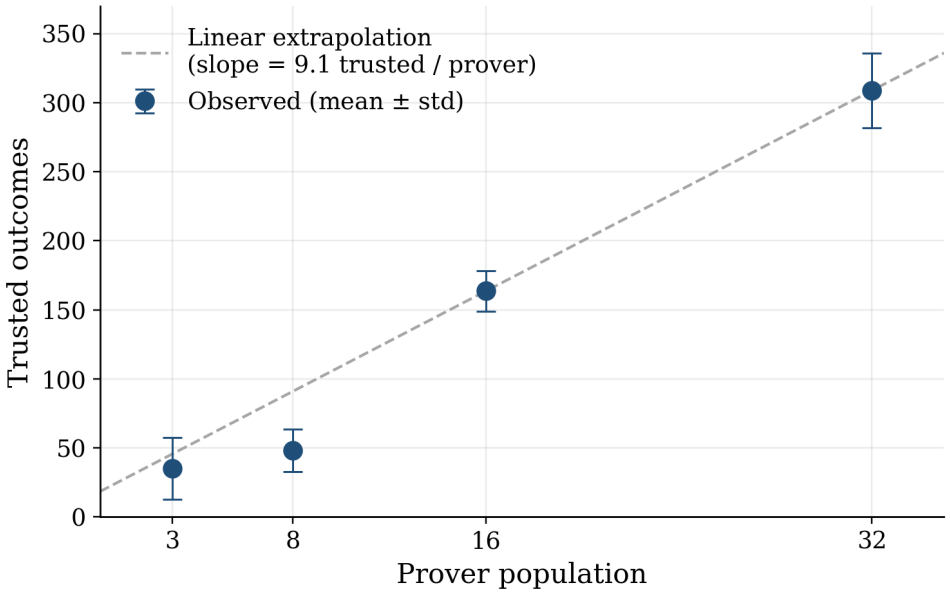


Figure 5.3: Trusted-outcome count versus prover population, with error bars indicating across-seed standard deviation. The dashed line is the linear extrapolation through pop $\in \{16, 32\}$, the two cleanest data points; pop $= 3$ and pop $= 8$ sit visibly below this line because random position seeds occasionally place provers outside the Radar's effective range, removing a disproportionate fraction of the small-population workload.

Table 5.5: Population sweep at the baseline configuration. FORGED, REPLAYED, and DESYNCED counts are zero in every row; per-cycle RTT is 1.853 s in every row.

Population	**TRUSTED**
3	35.2 ± 22.2
8	48.2 ± 15.3
16	163.6 ± 14.8
32	308.8 ± 27.2

The standard sweeps above place only legitimate provers in the field, which exercises the TRUSTED branch of the verifier's classification logic but leaves FORGED classifications unexercised. The attacker-scenarios configuration restores this coverage by replacing approximately half of the population with rogue transponders that hold a key the verifier does not. Table 5.6 reports the resulting outcomes over the same 120 s horizon. The 21 TRUSTED and 19 FORGED outcomes correspond to the legitimate and rogue halves of the population respectively; no rogue cycle is ever classified TRUSTED and no legitimate cycle is ever classified FORGED.

Table 5.6: Attacker-scenarios run with approximately equal numbers of legitimate and rogue provers. TRUSTED and FORGED outcomes together account for all 40 cycles; no REPLAYED or DESYNCED outcomes occur, and per-cycle RTT remains 1.853 s.

Detections	TRUSTED	FORGED	REPLAYED	DESYNCED
361	21	19	0	0

The deterministic channel model used above isolates the protocol's logical behavior, but operational deployments expose the response link to bit errors. To characterize the protocol's response to channel noise, the response link was modelled as a binary symmetric channel with *bit-error rate* (BER) varied from 10^{-4} to 10^{-2} in five steps, with five seeded replications at each BER, see Table 5.7. The outcome shifts shown in Figure 5.4 are entirely between the TRUSTED and FORGED classes: the REPLAYED and DESYNCED counts remain zero at every noise level, so the protocol always fails toward rejection. At BER $= 10^{-4}$, 97% of cycles still verify; the trusted and forged curves cross near BER $\approx 2.5 \times 10^{-3}$, and by BER $= 10^{-2}$, fewer than one cycle in ten still verifies. The shift is monotonic in BER and the within BER standard deviation never exceeds 3.3 cycles, consistent with the analytical prediction that a single bit error anywhere in the 256 bit MAC tag causes the verifier-side recomputation to disagree and the cycle to be classified FORGED. The practical reliability ceiling for this configuration is therefore in the neighborhood of BER $\approx 10^{-3}$, below which more than 75% of cycles complete successfully on the first attempt.

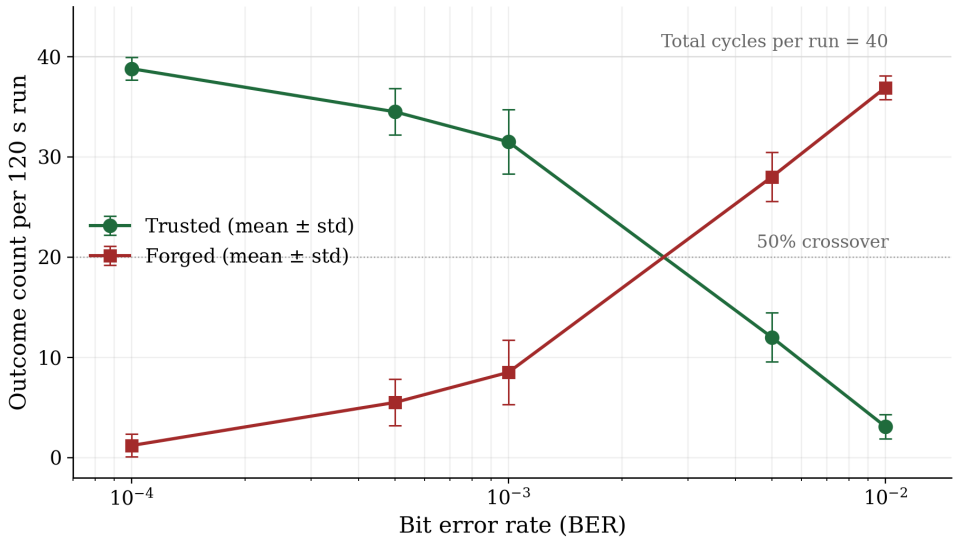


Figure 5.4: Trusted (green) and forged (red) outcome counts as a function of the response-channel BER, with error bars from five seeded replications per BER value. The total TRUSTED + FORGED remains 40 at every noise level and neither REPLAYED nor DESYNCED classifications occur, so cycles corrupted by bit errors are rejected rather than accepted. The crossover at BER $\approx 2.5 \times 10^{-3}$ marks the point at which the channel destroys more cycles than it preserves.

Table 5.7: Noise sweep on the response channel, modelled as a binary symmetric channel. REPLAYED and DESYNCED counts are zero in every row; per-cycle RTT is 1.853 s in every row.

BER	Trusted	Forged
10^{-4}	38.8 ± 1.1	1.2 ± 1.1
5×10^{-4}	34.5 ± 2.3	5.5 ± 2.3
10^{-3}	31.5 ± 3.2	8.5 ± 3.2
5×10^{-3}	12.0 ± 2.4	28.0 ± 2.4
10^{-2}	3.1 ± 1.2	36.9 ± 1.2

5.1.3 Tamarin prover

Phases 3 and 4 of the ABLA protocol are modelled in the Tamarin prover under a Dolev–Yao network adversary, with Phase 0 provisioning abstracted as a one-shot rule that creates a fresh long-term key K_i shared between the verifier and prover. The MAC primitive is represented as an uninterpreted function symbol $hmac/2$, capturing the *Existential Unforgeability under Chosen Message Attack* (EUF-CMA) property of HMAC-SHA256 in the symbolic model. The prover's firmware is abstracted as a binary state, *good* or *bad*: the digest is modelled as a public function over this state, a dedicated compromise rule lets the adversary flip a prover from *good* to *bad* without extracting its key, and the verifier recomputes the expected MAC against the reference state *good*.

As stated, the model covers Phase 3 and Phase 4 of the protocol only. Under these assumptions, Tamarin establishes (i) secrecy of K_i, that is the per–device key stays secret in the absence of an explicit compromise event; (ii) unique verifier commit (ts, ch, tr) tuple per session; (iii) freshness, i.e., no two verifier commits ever share the same timestamp (P, TS_v); (iv) attestation soundness, i.e., every accepted response was computed by the addressed prover, in this session, while holding unmodified firmware. The model does not cover carrier authentication (Phase 1), proximity verification (Phase 2), finite 6-bit wrap-around behavior, side-channel attacks, physical capture beyond key reveal, memory contents beyond the binary firmware state, or implementation-level attacks. The full set of six lemmas executes in 1.12 s, with no manual proof guidance.

6

Discussion

This chapter interprets the results of Chapter 4 in the context of the research questions posed in Chapter 1 and the related work surveyed in Chapter 2. It then addresses the limitations of the study and outlines directions for future work.

6.1 Interpretation of results

The OMNeT++ evaluation establishes that the ABLA protocol fits within the timing and bandwidth budget imposed by a rotating-radar deployment. In the baseline configuration, a complete Phase 3 challenge–response cycle requires 276 bit and completes in 1.853 s, of which 1.000 s is consumed by the 20 bps downlink challenge and 0.853 s by the 256 bit backscatter response at 300 bps. The rogue transponder provisioned with a key not held by the verifier is detected as FORGED on every cycle, while the legitimate transponders are classified as TRUSTED on every cycle.

The parameter sweeps confirm that the protocol behaves as the analytical model predicts. Per-cycle RTT is independent of both population and reattestation interval: it is set by the bit-level transit times on the asymmetric channel and not by the rate at which cycles are scheduled. The challenge–bitrate sweep shows the expected linear relationship between downlink rate and RTT, with a reduction of approximately 1.75 s when the radar rate is raised from 10 to 80 bps. The response-bitrate sweep behaves symmetrically. The population sweep, repeated with five seeded replications per population size, shows that detection density scales roughly linearly with population while per-cycle RTT remains constant, and that full trust coverage is maintained even at 32 transponders. None of these results is surprising in isolation, but together they provide evidence that the protocol's behavior matches the design assumptions that motivated it.

The Tamarin verification covers Phases 3 and 4 of the protocol only, where the lemmas verify key secrecy under no key reveal, verifier commit uniqueness of the (ts, ch, tr) tuple, freshness, and attestation soundness. The six lemmas discharge automatically in 1.12 s with no proof guidance. In the symbolic model, the protocol therefore guarantees that the per-device key remains secret in the absence of an explicit compromise event, that every verifier commit corresponds to a unique prover session with matching parameters, and that no two commits share the same (prover, timestamp) pair. The freshness property in particular is the symbolic counterpart of the monotonic-counter check in Phase 4, Step 1, and provides assurance that

the design choice of a verifier-issued timestamp is sufficient to prevent replay at the protocol level. The soundness lemma, in turn, is the symbolic counterpart of the digest reconstruction in Phase 4 Steps 3–7.

6.2 Relation to Research Questions

With respect to RQ1, the protocol described in Chapter 4 demonstrates that a remote attestation exchange can be carried out over a 20 bps downlink and a 300 bps backscatter uplink within two illuminations. The design uses only symmetric primitives on the prover side (HMAC-SHA-256, SHA-256), broadcasts the 48 bit carrier-authentication token once per epoch rather than per device, and confines per-device downlink traffic to a 20 bit challenge. The total per-cycle overhead of at most 324 bit compares favorably with prior protocols targeting comparable settings: RESERVE [47] reports 500 bit, ROSTER [69] reports 536 bit, and the military authentication scheme of [27] requires 1172 bit, which at the bitrates considered here would translate to authentication times in the tens to hundreds of seconds. The bandwidth and timing constraints imposed by the rotating-radar link are therefore not prohibitive: a complete attestation cycle fits within two illuminations at 1.85 s of RTT, leaving sufficient margin for the proximity-verification exchange of Phase 2 and for scheduling overhead at the verifier.

With respect to RQ2, the protocol addresses each of the adversarial capabilities enumerated in the threat model through a combination of cryptographic and non–cryptographic mechanisms, and these mechanisms are evaluated through two complementary methodologies. Eavesdropping is mitigated by transmitting a 256 bit MAC in uplink, while accepting that the broadcast carrier signal is observable; replay is prevented by the verifier-issued monotonic timestamp and the freshness check at Phase 4 Step 1; spoofing of attestation responses is likewise prevented by the HMAC tag bound to the verifier's challenge and timestamp; jamming is treated as outside the cryptographic threat model but is partially addressed through liveness monitoring, which surfaces sustained loss of valid responses. Relay attacks and physical displacement are addressed by Phase 2 distance bounding, which exploits the radar's existing range measurement rather than introducing additional cryptographic machinery on the prover. Physical capture of an individual transponder cannot be prevented cryptographically; the protocol instead bounds the consequences of such compromise by isolating per-device keys, and provides indirect detection through clone detection, movement-anomaly flags, and liveness monitoring.

These guarantees are evaluated along two axes. The symbolic analysis in Tamarin establishes the security of the cryptographic core, Phase 3 and Phase 4, under a Dolev–Yao network adversary. All six lemmas, sanity, key secrecy, non-injective agreement, unique commit on the (ts, ch, tr) tuple, freshness, and attestation soundness, discharge automatically in 1.12 s without additional proof guidance. In the symbolic model, the protocol therefore guarantees that the per–device key remains secret in the absence of an explicit compromise event, that no two commits share the same timestamp, and that the verifier and prover agree on the timestamp, challenge and trigger. The discrete-event simulation in OMNeT++, in turn, evaluates

the protocol's operational behavior: it confirms that detection and trust outcomes scale predictably across BER, bitrate, reattestation interval, and population, and that trust is maintained at up to 32 transponders. Taken together, the two evaluations cover the abstract security of the protocol and its concrete behavior under different configurations, but neither captures side-channel attacks, implementation flaws in the underlying cryptography, or the full set of physical-layer effects in a real deployment. This is future work.

6.3 Relation to Prior Work

The closest match in the prior literature is SHeLA [48], whose layered trust model where more capable intermediate nodes are trusted relative to constrained leaf nodes directly mirrors the asymmetric trust relationship between the radar and the transponders considered here. ABLA can be seen as an adaptation of that trust model to a single-hop, broadcast-illumination setting, where the "intermediate" layer collapses into the verifier itself and where the bandwidth asymmetry is dictated by the physical layer rather than by node capability. The hybrid architectures of TyTAN [9], HYDRA [16], and VRASED [44] provide the on-device primitives: atomic execution, key isolation, and secure measurement that any concrete realization of ABLA would ultimately rest on. However, they are orthogonal to the channel-level design considered in this thesis.

The collective and swarm attestation protocols surveyed in Section 3.1, such as SANA [2], SEDA [7], and HAGAR [67], target a fundamentally different problem: aggregating attestation evidence across multi-hop networks of mutually communicating nodes. The star topology and one-hop constraint of the deployment considered here rule out such aggregation schemes, and the small network size (≤ 32 transponders) means that the scalability concerns that motivate them do not apply.

6.4 Limitations

Three limitations of this study deserve explicit acknowledgement. First, no hardware implementation was performed. The Arduino Nano ESP32 referenced in Section 4.6.1 was identified as a plausible target platform, but the protocol was not deployed, profiled, or measured on real hardware. The cryptographic cost of HMAC-SHA-256 and SHA-256 on such a platform, the energy consumed by an attestation cycle, the timing jitter introduced by interrupt handling and clock variability, and the realistic memory layout of a deployed transponder are all matters that a hardware study would resolve and that the present simulation-based evaluation cannot. Claims about real-world feasibility should therefore be read as conditional on a successful hardware validation, which can be seen as the most important next step.

Second, the security analysis is partial in two distinct senses. The Tamarin verification covers only Phases 3 and 4, leaving Phase 1 (carrier authentication) and Phase 2 (proximity verification) outside the symbolic model. Extending the model to cover these phases – in particular, modelling the radar-based proximity check of

Phase 2, which depends on physical timing rather than only on message contents – is non-trivial and was not attempted here. Separately, even within the Phase 3 and 4 model, the analysis is symbolic: HMAC-SHA-256 is treated as an uninterpreted function symbol satisfying EUF-CMA, and the model captures neither side-channel attacks nor implementation flaws in the underlying cryptographic primitives. The reverse-engineering of cryptographic RFID tags discussed in Section 3.5 [43] illustrates how a sound abstract design can be undermined by concrete implementation choices, and the Tamarin proofs do not protect against that class of failure. In addition, the attestation–soundness result rests on a binary firmware–state abstraction: the model distinguishes only modified from unmodified firmware and does not represent actual memory contents, so the lemma should be read as a minimal soundness guarantee rather than full memory–attestation soundness.

Third, the evaluation considered at most 32 transponders. While this is consistent with the deployment scale anticipated by the project, it is well below the scales targeted by SANA, SEDA, or HAGAR, and the absence of contention or attestation-request collisions at 32 nodes does not imply the same will hold at, say, 256 or 1024. The protocol does not currently include a MAC-layer arbitration mechanism, and at higher densities the question of how to schedule attestation cycles without echo collisions becomes a design problem in its own right.

6.5 Future Work

Several directions follow naturally from the limitations above. The most immediate is a hardware implementation on the Arduino Nano ESP32 or a comparable platform, which would establish whether the symbolic and simulation-level results translate to a working system and would quantify the energy, timing, and memory costs that the present evaluation abstracts away.

A second direction is to extend the formal model to cover Phases 1 and 2. Carrier authentication is a straightforward extension of the existing Tamarin theory, but proximity verification requires modelling physical timing constraints, which lies at the boundary of what symbolic tools can express cleanly. Radar-based proximity analyses in the computational model, or hybrid approaches combining symbolic verification with timing assumptions, are candidate methodologies.

A third direction concerns PQC readiness. The protocol's reliance on symmetric primitives means that Grover's algorithm imposes only a quadratic speed-up, and the 256 bit key length already accounts for this. The asymmetric layer that would be needed to replace the long-lived shared carrier key K_{bc} with an online authenticated key exchange, however, is currently absent; introducing such a layer with a PQ-secure key-encapsulation mechanism (FIPS 203's ML-KEM [38]) or a hash-based signature scheme (FIPS 205's SLH-DSA [39]) would eliminate the most attractive single target for hardware extraction and align the protocol with the migration trajectory described in Section 3.5.

A fourth direction is to relax the assumption of always–on, energy–rich transponders. The intermittent attestation approaches of RESERVE [47] and ITERATOR [58] ad-

dress scenarios in which a single attestation cycle cannot be completed in a single execution window, and adapting such ideas to a rotating Radar setting would broaden the protocol's applicability to energy-harvested or duty-cycled deployments.

A fifth direction is to make fuller use of the Radar's sequence-identification mechanism, in which a transponder pre-programmed with a particular sequence of pulse-repetition frequencies responds only to a Radar that reproduces it. Performing this check in the transponder hardware could make the protocol more lightweight, replacing the cryptographic carrier authentication of Phase 1 with a filter the device already applies at the physical layer. It would also give each transponder a cheap first-line means of rejecting illuminations from radars that do not present the expected sequence, before any attestation exchange or communication begins.

Finally, larger-scale evaluations – both in simulation and on hardware – would clarify how the protocol behaves under contention, and would establish whether MAC-layer scheduling or some form of slotted access is required at higher transponder densities.

7

Conclusion

This thesis investigated how a remote attestation protocol can be designed to operate over an asymmetric, low-bitrate backscatter link between a rotating Radar and a small population of resource–constrained transponders. The resulting protocol, ABLA, organizes attestation into five phases; provisioning, carrier authentication, proximity verification, challenge–response attestation, and verification. The protocol uses only symmetric primitives on the prover side. The Phase 2 distance-bounding step reuses the Radar's existing range measurement to provide a non-cryptographic check against relay attacks and physical displacement, complementing the cryptographic guarantees of Phases 3 and 4.

The protocol was evaluated along two axes. A discrete-event simulation in OMNeT++ showed that a complete attestation cycle fits within two complete illuminations at 1.85 s of round-trip time, that trust coverage is achieved across a 120 s run with up to 32 transponders, and that outcome counts scale predictably with BER, reattestation interval, and bitrate. A symbolic analysis in the Tamarin prover discharged six lemmas covering key secrecy, agreement, unique commit for the tuple (ts, ch, tr), freshness, and attestation soundness for Phases 3 and 4 under a Dolev–Yao adversary.

The work is best read as a feasibility study. It demonstrates that the asymmetric, low-bitrate setting does not, in principle, preclude meaningful attestation, and it provides a concrete design that could be implemented and tested. Translating that demonstration into an operational system will require hardware validation, extension of the formal model to cover the full protocol, and evaluation at scales beyond those considered here. These steps are the natural continuation of the work and are left for future investigation.

Bibliography

[1] Aida Aberra et al. “Backscatter Communication for Biomedical Devices”. In: *Handbook of Biochips: Integrated Circuits and Systems for Biology and Medicine*. Ed. by Mohamad Sawan. New York, NY: Springer New York, 2020, pp. 1–18. ISBN: 978-1-4614-6623-9. DOI: 10.1007/978-1-4614-6623-9_39-1. URL: https://doi.org/10.1007/978-1-4614-6623-9_39-1.

[2] M. Ambrosin et al. “SANA: Secure and Scalable Aggregate Network Attestation”. In: *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. Vienna, Austria, Oct. 2016.

[3] Moreno Ambrosin et al. “Collective Remote Attestation at the Internet of Things Scale: State-of-the-Art and Future Challenges”. In: *IEEE Communications Surveys & Tutorials* 22.4 (2020), pp. 2447–2461.

[4] Moreno Ambrosin et al. “PADS: Practical Attestation for Highly Dynamic Swarm Topologies”. In: *2018 International Workshop on Secure Internet of Things (SIoT)* (2018), pp. 18–27. URL: https://api.semanticscholar.org/CorpusID:49268741.

[5] Mahmoud Ammar et al. “SlimIoT: Scalable Lightweight Attestation Protocol for the Internet of Things”. In: *2018 IEEE Conference on Dependable and Secure Computing (DSC)*. 2018, pp. 1–8. DOI: 10.1109/DESEC.2018.8625142.

[6] Arduino. *Nano ESP32*. https://docs.arduino.cc/hardware/nano-esp32/. Accessed: 2026-05-12. 2026.

[7] N. Asokan et al. “SEDA: Scalable Embedded Device Attestation”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS ’15. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 964–975. ISBN: 9781450338325. DOI: 10.1145/2810103.2813670. URL: https://doi.org/10.1145/2810103.2813670.

[8] Stefan Brands and David Chaum. “Distance-Bounding Protocols”. In: *Advances in Cryptology — EUROCRYPT ’93*. Ed. by Tor Helleseth. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 344–359. ISBN: 978-3-540-48285-7.

[9] Ferdinand Brasser et al. “TyTAN: Tiny trust anchor for tiny devices”. In: *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2015, pp. 1–6. DOI: 10.1145/2744769.2744922.

[10] F. Bruckert et al. *Remote ATtestation Procedure Architecture (RATS)*. RFC 9334. 2023.

[11] Xavier Carpent, Gene Tsudik, and Norrathep Rattanavipanon. “ERASMUS: Efficient remote attestation via self-measurement for unattended settings”. In:

2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). 2018, pp. 1191–1194. DOI: 10.23919/DATE.2018.8342195.

[12] Xavier Carpent et al. "Lightweight Swarm Attestation: A Tale of Two LISA-s". In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ASIA CCS '17. Abu Dhabi, United Arab Emirates: Association for Computing Machinery, 2017, pp. 86–100. ISBN: 9781450349444. DOI: 10.1145/3052973.3053010. URL: https://doi.org/10.1145/3052973.3053010.

[13] David Cooper et al. *BIOS Protection Guidelines*. Tech. rep. NIST SP 800-147. National Institute of Standards and Technology, Apr. 2011. DOI: 10.6028/NIST.SP.800-147.

[14] Marius Cotiga et al. "Towards Resilient and QuantumSafe RADAR Communication". In: *Information Systems Technology Panel*. STO, Nov. 2025. ISBN: 978-92-837-2605-0.

[15] Daniel M. Dobkin. *The RF in RFID: UHF RFID in Practice*. 2nd. Oxford, U.K.: Newnes (Elsevier), 2012.

[16] Karim Eldefrawy, Norrathep Rattanavipanon, and Gene Tsudik. "HYDRA: hybrid design for remote attestation (using a formally verified microkernel)". In: *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec '17. Boston, Massachusetts: Association for Computing Machinery, 2017, pp. 99–110. ISBN: 9781450350846. DOI: 10.1145/3098243.3098261. URL: https://doi.org/10.1145/3098243.3098261.

[17] Karim Eldefrawy et al. "SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust". In: *Network and Distributed System Security Symposium (NDSS)*. 2012.

[18] Ericsson. *Ericsson Mobility Report*. https://www.ericsson.com/en/reports-and-papers/mobility-report. Accessed: 2026-04-21. 2023.

[19] Fondazione Bruno Kessler. *NuSMV: a new symbolic model checker*. https://nusmv.fbk.eu/. Accessed: 2026-05-12. 2025.

[20] Aurélien Francillon et al. "A minimalist approach to remote attestation". In: *Proceedings of the Conference on Design, Automation & Test in Europe*. DATE '14. Dresden, Germany: European Design and Automation Association, 2014. ISBN: 9783981537024.

[21] Anjali Subhash Gajbhiye and Gayatri Bhoyar. "Bistatic Backscatter Communication: The Review Paper". In: *2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA*. 2022, pp. 1–5. DOI: 10.1109/ICCUBEA54992.2022.10011052.

[22] Joshua D. Griffin and Gregory D. Durgin. "Complete Link Budgets for Backscatter-Radio and RFID Systems". In: *IEEE Antennas and Propagation Magazine* 51.2 (2009), pp. 11–25. DOI: 10.1109/MAP.2009.5162013.

[23] GS1 EPCglobal. *EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID Standard: Specification for RFID Air Interface Protocol for Communications at 860 MHz–960 MHz*. Release 2.1. 2018.

[24] Basel Halak, Yildiran Yilmaz, and Daniel Shiu. "Comparative Analysis of Energy Costs of Asymmetric vs Symmetric Encryption-Based Security Appli-

cations". In: *IEEE Access* 10 (July 2022), pp. 1–1. DOI: 10.1109/ACCESS.2022.3192970.

[25] Ahmad Ibrahim et al. "DARPA: Device Attestation Resilient to Physical Attacks". In: *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '16. Darmstadt, Germany: Association for Computing Machinery, 2016, pp. 171–182. ISBN: 9781450342704. DOI: 10.1145/2939918.2939938. URL: https://doi.org/10.1145/2939918.2939938.

[26] *IEEE Standard Letter Designations for Radar-Frequency Bands*. IEEE, 2019. DOI: 10.1109/IEEESTD.2020.8999849.

[27] Usha Jain and Muzzammil Hussain. "Securing Wireless Sensors in Military Applications through Resilient Authentication Mechanism". In: *Procedia Computer Science* 171 (2020). Third International Conference on Computing and Network Communications (CoCoNet'19), pp. 719–728. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.04.078. URL: https://www.sciencedirect.com/science/article/pii/S1877050920310462.

[28] Masoud Kaveh et al. *Secure Backscatter Communications Through RIS: Modeling and Performance*. 2024. arXiv: 2410.01829 [cs.IT]. URL: https://arxiv.org/abs/2410.01829.

[29] Patrick Koeberl et al. "TrustLite: A Security Architecture for Tiny Embedded Devices". In: *Proceedings of the Ninth European Conference on Computer Systems (EuroSys)*. 2014. DOI: 10.1145/2592798.2592824.

[30] Florian Kohnhäuser, Niklas Büscher, and Stefan Katzenbeisser. "SALAD: Secure and Lightweight Attestation of Highly Dynamic and Disruptive Networks". In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ASIACCS '18. Incheon, Republic of Korea: Association for Computing Machinery, 2018, pp. 329–342. ISBN: 9781450355766. DOI: 10.1145/3196494.3196544. URL: https://doi.org/10.1145/3196494.3196544.

[31] Florian Kohnhäuser et al. "SCAPI: a scalable attestation protocol to detect software and physical attacks". In: *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec '17. Boston, Massachusetts: Association for Computing Machinery, 2017, pp. 75–86. ISBN: 9781450350846. DOI: 10.1145/3098243.3098255. URL: https://doi.org/10.1145/3098243.3098255.

[32] Vincent Liu et al. "Ambient Backscatter: Wireless Communication Out of Thin Air". In: *Proc. ACM SIGCOMM*. Hong Kong, China, Aug. 2013, pp. 39–50. DOI: 10.1145/2486001.2486015.

[33] Kerry A. McKay et al. *Report on Lightweight Cryptography*. Tech. rep. NISTIR 8114. National Institute of Standards and Technology, 2017. DOI: 10.6028/NIST.IR.8114.

[34] Simon Meier et al. "The TAMARIN Prover for the Symbolic Analysis of Security Protocols". In: *Computer Aided Verification*. Ed. by Natasha Sharygina and Helmut Veith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 696–701. ISBN: 978-3-642-39799-8.

[35] Deepak Mishra and Erik G. Larsson. “Monostatic Backscattering Detection by Multiantenna Reader”. In: *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. 2019, pp. 697–701. DOI: 10.1109/IEEECONF44664.2019.9048853.

[36] National Institute of Standards and Technology. *Attestation — NIST CSRC Glossary*. Computer Security Resource Center, https://csrc.nist.gov/glossary/term/attestation. Accessed: 2026-05-05. 2025.

[37] National Institute of Standards and Technology. *Module-Lattice-Based Digital Signature Standard*. Tech. rep. FIPS Publication 204. U.S. Department of Commerce, Aug. 2024. DOI: 10.6028/NIST.FIPS.204.

[38] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Tech. rep. FIPS Publication 203. U.S. Department of Commerce, Aug. 2024. DOI: 10.6028/NIST.FIPS.203.

[39] National Institute of Standards and Technology. *Stateless Hash-Based Digital Signature Standard*. Tech. rep. FIPS Publication 205. U.S. Department of Commerce, Aug. 2024. DOI: 10.6028/NIST.FIPS.205.

[40] Dung T. Nguyen et al. “Security Issues in IoT-Based Wireless Sensor Networks: Classifications and Solutions”. In: *Future Internet* 17.8 (2025). ISSN: 1999-5903. DOI: 10.3390/fi17080350. URL: https://www.mdpi.com/1999-5903/17/8/350.

[41] Pavel V. Nikitin and K. V. S. Rao. “Theory and Measurement of Backscattering from RFID Tags”. In: *IEEE Antennas and Propagation Magazine* 48.6 (Dec. 2006), pp. 212–218. DOI: 10.1109/MAP.2006.323323.

[42] Jin-Ping Niu and Geoffrey Ye Li. “An Overview on Backscatter Communications”. In: *Journal of Communications and Information Networks* 4.2 (2019), pp. 1–14. DOI: 10.23919/JCIN.2019.8917868.

[43] Karsten Nohl et al. “Reverse-engineering a cryptographic RFID tag”. In: *Proceedings of the 17th Conference on Security Symposium*. SS’08. San Jose, CA: USENIX Association, 2008, pp. 185–193.

[44] Ivan De Oliveira Nunes et al. “VRASED: A Verified Hardware/Software Co-Design for Remote Attestation”. In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1429–1446. ISBN: 978-1-939133-06-9. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/de-oliveira-nunes.

[45] philip_0008. *How do light waves get their size?* Physics Stack Exchange. Accessed: 2016-06-23. 2016. URL: https://physics.stackexchange.com/q/258561.

[46] Stevan Preradovic, Nemai C. Karmakar, and Isaac Balbin. “RFID Transponders”. In: *IEEE Microwave Magazine* 9.5 (2008), pp. 90–103. DOI: 10.1109/MMM.2008.927637.

[47] Md Masoom Rabbani et al. “RESERVE: Remote Attestation of Intermittent IoT devices”. In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. SenSys ’21. Coimbra, Portugal: Association for Computing Machinery, 2021, pp. 578–580. ISBN: 9781450390972. DOI: 10.1145/3485730.3493364. URL: https://doi.org/10.1145/3485730.3493364.

[48] Md Masoom Rabbani et al. “SHeLA: Scalable Heterogeneous Layered Attestation”. In: *IEEE Internet of Things Journal* 6.6 (2019), pp. 10240–10250. DOI: 10.1109/JIOT.2019.2936988.

[49] Reeshen Reddy and Saurabh Sinha. “State-of-the-Art Review: Electronic Warfare Against Radar Systems”. In: *IEEE Access* 13 (2025), pp. 57530–57567. DOI: 10.1109/ACCESS.2025.3555493.

[50] Théobald de Riberolles et al. “Characterizing Radar Network Traffic: a first step towards spoofing attack detection”. In: *2020 IEEE Aerospace Conference*. 2020, pp. 1–8. DOI: 10.1109/AERO47225.2020.9172292.

[51] Mark A. Richards. *Fundamentals of Radar Signal Processing*. 2nd. New York, NY, USA: McGraw-Hill Education, 2014.

[52] Mark A. Richards and William L. Melvin, eds. *Principles of Modern Radar: Basic Principles*. 2nd. Radar, Sonar and Navigation. SciTech Publishing, 2023, p. 1152. ISBN: 9781839533815.

[53] Philip Ronan. *Electromagnetic spectrum*. Wikimedia Commons. Licensed under CC BY-SA 3.0. 2007. URL: https://commons.wikimedia.org/w/index.php?curid=2521356.

[54] seL4 Foundation. *The seL4 Microkernel*. https://sel4.systems/. Accessed: 2026-05-12. 2026.

[55] A. Seshadri et al. “SWATT: softWare-based attestation for embedded devices”. In: *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. 2004, pp. 272–282. DOI: 10.1109/SECPRI.2004.1301329.

[56] Jaleel Shaheen et al. “Confidential and Secure Broadcast in Wireless Sensor Networks”. In: *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*. 2007, pp. 1–5. DOI: 10.1109/PIMRC.2007.4394560.

[57] Tao Shen et al. “LTRAA: Lightweight and transparent remote attestation with anonymity”. In: *Journal of Information Security and Applications* 97 (2026), p. 104352. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2025.104352. URL: https://www.sciencedirect.com/science/article/pii/S2214212625003886.

[58] Nicoló Sponziello et al. “ITERATOR: Interruptible Remote Attestation Through Cuckoo Filters”. In: *IEEE Internet of Things Journal* 12.24 (2025), pp. 54746–54758. DOI: 10.1109/JIOT.2025.3621016.

[59] William Stallings. *Cryptography and Network Security: Principles and Practice*. 7th. Boston, MA, USA: Pearson, 2017.

[60] George W. Stimson. *Introduction to Airborne Radar*. 2nd. Raleigh, NC, USA: SciTech Publishing, 1998.

[61] Texas Instruments. *MSP430 MCU design & development*. 2026. URL: https://www.ti.com/design-development/embedded-development/msp430-mcus.html (visited on 05/12/2026).

[62] Bryan Thomas. *Understanding HAB and Code Signing*. NXP Technical Presentation, AMF-AUT-T2794. NXP FTF/Connects technical session. NXP Semiconductors, Aug. 2017.

[63] Stewart J. Thomas et al. “Quadrature Amplitude Modulated Backscatter in Passive and Semipassive UHF RFID Systems”. In: *IEEE Transactions on Mi-*

crowave Theory and Techniques 60.4 (Apr. 2012), pp. 1175–1182. DOI: 10.1109/TMTT.2012.2185810.

[64] Trusted Computing Group. *DICE Attestation Architecture*. Specification Version 1.2. Trusted Computing Group (TCG), Apr. 2025.

[65] Trusted Computing Group. *Trusted Platform Module (TPM) Work Group*. https://trustedcomputinggroup.org/work-groups/trusted-platform-module/. Accessed: 2026-05-12. 2026.

[66] András Varga and Rudolf Hornig. "An overview of the OMNeT++ simulation environment". In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Simutools '08. Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. ISBN: 9789639799202.

[67] Jo Vliegen et al. "HAGAR: Hashgraph-based Aggregated Communication and Remote Attestation". In: *Proceedings of the 21st ACM International Conference on Computing Frontiers: Workshops and Special Sessions*. CF '24 Companion. Ischia, Italy: Association for Computing Machinery, 2024, pp. 10–16. ISBN: 9798400704925. DOI: 10.1145/3637543.3654655. URL: https://doi.org/10.1145/3637543.3654655.

[68] Weiqi Wu et al. "A survey on ambient backscatter communications: Principles, systems, applications, and challenges". In: *Computer Networks* 216 (2022), p. 109235. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2022.109235. URL: https://www.sciencedirect.com/science/article/pii/S1389128622003139.

[69] Ning Zhang et al. "ROSTER: Radio Context Attestation in Cognitive Radio Network". In: *2018 IEEE Conference on Communications and Network Security (CNS)*. 2018, pp. 1–9. DOI: 10.1109/CNS.2018.8433187.

[70] Jean-Karim Zinzindohoué et al. "HACL*: A Verified Modern Cryptographic Library". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 1789–1806. ISBN: 9781450349468. DOI: 10.1145/3133956.3134043. URL: https://doi.org/10.1145/3133956.3134043.

A

Tamarin proof source

The file models Phases 3 (challenge–response attestation) and 4 (verification) of ABLA under a Dolev–Yao network adversary, with an abstracted Phase 0 (provisioning) rule that mints fresh long-term keys. Phases 1 (carrier authentication) and 2 (proximity verification) are out of scope and not modeled. All six lemmas stated below close automatically; the verification was performed with Tamarin Prover 1.12.0 and Maude 3.5.1.

Listing A.1: Tamarin theory file `ABLA_Phase34.spthy`.

```
/* ================================================================
 *  ABLA Phase 3 / Phase 4  --  symbolic security model for Tamarin
 *  ----------------------------------------------------------------
 *  Verified with Tamarin Prover 1.12.0 /
 *  Maude 3.5.1; all six lemmas below close automatically.
 *
 *  Scope
 *  -----
 *  Models Phase 3 (challenge--response) and Phase 4 (verification)
 *  of the ABLA radar-backscatter attestation protocol, under a full
 *  Dolev--Yao network adversary.  Phase 0 (provisioning) is captured
 *  by a single setup rule that mints fresh long-term keys; Phases 1
 *  (carrier authentication) and 2 (proximity verification) are out
 *  of scope for this theory.
 *
 *  Abstractions
 *  ------------
 *    hmac(K, m)     keyed MAC, declared as an uninterpreted function
 *                   symbol.  Without K the adversary cannot produce
 *                   hmac(K, m) for any m, capturing EUF-CMA security
 *                   in the symbolic model.
 *    digest(P, tr, state)
 *                   public uninterpreted function, freely computable
 *                   by any party including the adversary.  `state` is
 *                   an abstract firmware state ('good' or 'bad'); the
 *                   prover digests whatever state it is actually in,
 *                   while the verifier recomputes against the
 *                   reference state 'good'.  Memory *contents* are
 *                   still not modelled -- only the binary distinction
 *                   between unmodified and modified firmware -- but
 *                   this lifts the theory from pure MAC
 *                   authentication to (minimal) attestation
 *                   soundness: acceptance implies the responding
```

```
34  *                   prover held good firmware.
35  *    ~ts            fresh TS_v.  Symbolic surrogate for the 6-bit
36  *                   monotonic counter in the implementation;
37  *                   Symbolic freshness is a simplification, not a
38  *                   stronger guarantee. It removes wrap-around
39  *                   behavior entirely. The 6-bit counter with
40  *                   modular arithmetic and sliding window W is NOT
41  *                   modelled here.
42  *
43  *
44  *
45  *  Identities
46  *  ----------
47  *    $V             public verifier identity (radar).
48  *    ~P, ~K         fresh prover identity and shared long-term key.
49  *    ~ch, ~tr       fresh challenge and trigger drawn by the
50  *                   verifier on each invocation.
51  *
52  *  Lemmas (all verified)
53  *  ---------------------
54  *    1. executable           protocol has an accepting trace.
55  *    2. key_secrecy          K_i remains secret absent compromise.
56  *    3. agreement            Accept implies P ran with same params.
57  *    4. unique_commit        No two verifier commits share the same (ts, ch,
    tr) tuple.
58  *    5. no_replay            no two Accepts share (V, P, TS_v).
59  *    6. attestation_soundness
60  *                            Accept implies the prover answered this
61  *                            session while holding good firmware.
62  * ================================================================== */
63
64  theory ABLA_Phase34
65  begin
66
67  functions: hmac/2, digest/3
68
69
70  /* ---------------- Phase 0 (abstracted): provisioning -------------- *
71   *  Mints a fresh prover identity and a fresh shared long-term key,
72   *  persisted at both ends.  The 5-bit prover ID is published.
73   * ------------------------------------------------------------------ */
74
75  rule Provision:
76      [ Fr(~P), Fr(~K) ]
77    --[ Provision(~P, ~K) ]->
78      [ !ProverKey(~P, ~K)
79      , !VerifierKey($V, ~P, ~K)
80      , ProverState(~P, 'good')
81      , Out(~P)
82      , LastAccepted(~P, 'init')
83      ]
84
85
86  /* ---------------- Adversary: firmware compromise ------------------- *
87   *  One-way transition from good to bad firmware.  Deliberately does
88   *  NOT reveal the key: it models a software-level compromise that
```

```
89  *  alters the measured memory without extracting the long-term
90  *  secret from the (assumed tamper-resistant) key store.  Full
91  *  device compromise is Reveal_Key below.
92  * ------------------------------------------------------------------ */
93
94  rule SoftwareCompromise:
95      [ ProverState(P, 'good') ]
96    --[ BadMemory(P) ]->
97      [ ProverState(P, 'bad') ]
98
99
100 /* ---------------- Phase 3: verifier issues challenge -------------- *
101  *  TS_v, CH and TR are drawn fresh and sent in the clear; Phase 3
102  *  does not authenticate the verifier (that is Phase 1's job).
103  * ------------------------------------------------------------------ */
104
105 rule V_Challenge:
106     [ !VerifierKey(V, P, K)
107     , Fr(~ts), Fr(~ch), Fr(~tr)
108     ]
109   --[ Issued(V, P, ~ts, ~ch, ~tr) ]->
110     [ Out(<P, ~ts, ~ch, ~tr>)
111     , V_Pending(V, P, ~ts, ~ch, ~tr)
112     ]
113
114
115 /* ---------------- Phase 3: prover responds ------------------------ *
116  *  MAC under the long-term key over (digest, TS_v, CH).  The prover
117  *  digests its *current* firmware state, consumed from and re-emitted
118  *  to ProverState.  Running records the parameters the prover
119  *  believed it was answering; ProverRanWith records the firmware
120  *  state the response was computed in.
121  * ------------------------------------------------------------------ */
122
123 rule P_Respond:
124     [ !ProverKey(P, K)
125     , ProverState(P, state)
126     , In(<P, ts, ch, tr>)
127     ]
128   --[ Running(P, <ts, ch, tr>)
129     , ProverRanWith(P, state)
130     ]->
131     [ ProverState(P, state)
132     , Out(hmac(K, <digest(P, tr, state), ts, ch>))
133     ]
134
135
136 /* ---------------- Phase 4: verifier accepts ----------------------- *
137  *  The Eq action becomes the equality mac = hmac(K, ...) via the
138  *  restriction below, capturing the verifier's MAC reconstruction
139  *  check; Neq enforces ts != prev_ts (the monotonic-counter check).
140  *  The verifier recomputes the MAC against the *reference* state
141  *  'good', so a response digested in state 'bad' can never verify.
142  *  Accept and Commit both fire on success: Commit drives the
143  *  agreement lemmas, Accept the sanity check.
144  * ------------------------------------------------------------------ */
```

```
145
146 rule V_Accept:
147     [ V_Pending(V, P, ts, ch, tr)
148     , !VerifierKey(V, P, K)
149     , In(mac)
150     , LastAccepted(P, prev_ts)
151     ]
152   --[ Eq(mac, hmac(K, <digest(P, tr, 'good'), ts, ch>))
153     , Neq(ts, prev_ts)
154     , Accept(V, P, ts, ch, tr)
155     , Commit(V, P, <ts, ch, tr>)
156     ]->
157     [ LastAccepted(P, ts)
158     ]
159
160
161 /* ---------------- Adversary: long-term key compromise ------------- *
162  *  Models a fully compromised prover.  Every lemma except
163  *  `executable` is guarded by `not (Ex #r. Reveal(P) @ r)` so that
164  *  compromise does not vacuously falsify properties for honest
165  *  participants.
166  * ------------------------------------------------------------------ */
167
168 rule Reveal_Key:
169     [ !ProverKey(P, K) ]
170   --[ Reveal(P) ]->
171     [ Out(K) ]
172
173
174 /* ============================ Restrictions ======================== */
175
176 restriction Eq_check:
177     "All x y #i. Eq(x, y) @ i ==> x = y"
178
179 restriction Neq_check:
180     "All x y #i. Neq(x, y) @ i ==> not (x = y)"
181
182
183 /* ============================== Lemmas ============================ */
184
185 // (1) Sanity: the protocol can run to completion with no compromise.
186 lemma executable: exists-trace
187     " Ex V P ts ch tr #i.
188         Accept(V, P, ts, ch, tr) @ i
189       & not (Ex Q #r. Reveal(Q) @ r & #r < #i) "
190
191
192 // (2) Secrecy of the long-term key K_i.
193 lemma key_secrecy:
194     " All P k #i.
195         Provision(P, k) @ i
196       & not (Ex #r. Reveal(P) @ r )
197       ==> not (Ex #j. K(k) @ j) "
198
199
200 // (3) Non-injective agreement: if the verifier commits with
```

```
201 //     parameters (ts, ch, tr), then the prover actually ran with
202 //     those same parameters earlier in the trace.
203 lemma agreement:
204     " All V P ts ch tr #i.
205         Commit(V, P, <ts, ch, tr>) @ i
206       & not (Ex #r. Reveal(P) @ r & #r < #i)
207       ==> Ex #j. Running(P, <ts, ch, tr>) @ j & j < i "
208
209
210 // (4) Unique Commit: No two verifier commits share the same (ts, ch, tr)
211 //                    tuple for the same prover.
212 lemma unique_commit:
213     " All V P ts ch tr #i.
214         Commit(V, P, <ts, ch, tr>) @ i
215       & not (Ex #r. Reveal(P) @ r & #r < #i)
216       ==> (Ex #j. Running(P, <ts, ch, tr>) @ j & j < i)
217         & not (Ex V2 #i2.
218                   Commit(V2, P, <ts, ch, tr>) @ i2
219                 & not (#i = #i2)) "
220
221
222 // (5) Freshness / replay-resistance: no two Commits ever share the
223 //     same (verifier, prover, TS_v) -- captures Phase 4 Step 1's
224 //     monotonic-counter check.
225 lemma no_replay:
226     " All V V2 P ts ch1 tr1 ch2 tr2 #i1 #i2.
227         Commit(V,  P, <ts, ch1, tr1>) @ i1
228       & Commit(V2, P, <ts, ch2, tr2>) @ i2
229       ==> #i1 = #i2 "
230
231
232 // (6) Attestation soundness: if the verifier accepts, the prover
233 //     computed *this session's* response while holding good firmware.
234 //     The state is bound into the MAC via digest(P, tr, state), so a
235 //     'bad'-state response cannot match the verifier's reference
236 //     recomputation, and (absent key compromise) nobody else can
237 //     forge a 'good'-state MAC.
238 lemma attestation_soundness:
239     " All V P ts ch tr #i.
240         Accept(V, P, ts, ch, tr) @ i
241       & not (Ex #r. Reveal(P) @ r & #r < #i)
242       ==> Ex #j. Running(P, <ts, ch, tr>) @ j
243                & ProverRanWith(P, 'good') @ j
244                & #j < #i "
245
246 end
```

B

Tamarin proof output

The output below was produced by `tamarin-prover -prove abla_phase34.spthy` using Tamarin 1.12.0 and Maude 3.5.1. The full theory source is listed in Appendix A. Each section reproduces the proof script that Tamarin emits for one lemma; the final section gives the verdict summary.

B.1 Summary

```
summary of summaries:

analyzed: abla_phase34.spthy

  processing time: 1.12s

  executable (exists-trace): verified (16 steps)
  key_secrecy (all-traces): verified (3 steps)
  agreement (all-traces): verified (7 steps)
  unique_commit (all-traces): verified (17 steps)
  no_replay (all-traces): verified (10 steps)
  attestation_soundness (all-traces): verified (7 steps)
```

B.2 Lemma: `executable`

Sanity check confirms the protocol admits at least one accepting trace without prover compromise.

```
lemma executable:
  exists-trace
  "Ex V P ts ch tr #i.
    (Accept( V, P, ts, ch, tr ) @ #i)
    (not (Ex Q #r. (Reveal( Q ) @ #r) & (#r < #i)))"
/*
guarded formula characterizing all satisfying traces:
"Ex V P ts ch tr #i.
  (Accept( V, P, ts, ch, tr ) @ #i)
 &
```

```
  all Q #r. (Reveal( Q ) @ #r) => not (#r < #i)"
*/
simplify
solve( V_Pending( V, P, ts, ch, tr ) -> #i )
  case V_Challenge
  solve( !VerifierKey( $V, ~P, K ) -> #i )
    case Provision
    solve( !KU( hmac(~K, <digest(~P, ~tr, 'good'), ~ts, ~ch>) ) @ #vk )
      case P_Respond
      solve( LastAccepted( ~P, prev_ts ) -> #i )
        case Provision
        solve( ProverState( ~P, 'good' ) -> #vr.2 )
          case P_Respond
          solve( ProverState( ~P, 'good' ) -> #vr.3 )
            case P_Respond
            solve( ProverState( ~P, 'good' ) -> #vr.4 )
              case P_Respond
              solve( ProverState( ~P, 'good' ) -> #vr.5 )
                case P_Respond
                solve( ProverState( ~P, 'good' ) -> #vr.6 )
                  case P_Respond
                  solve( ProverState( ~P, 'good' ) -> #vr.7 )
                    case Provision
                    solve( !KU( ~P ) @ #vk.2 )
                      case Provision
                      solve( !KU( ~ts ) @ #vk.9 )
                        case V_Challenge
                        solve( !KU( ~ch ) @ #vk.11 )
                          case V_Challenge
                          solve( !KU( ~tr ) @ #vk.12 )
                            case V_Challenge
                            SOLVED // trace found
                          qed
                        qed
                      qed
                    qed
                  qed
                qed
              qed
            qed
          qed
        qed
      qed
    qed
  qed
qed
```

B.3 Lemma: `key_secrecy`

The long-term key K_i remains unknown to the adversary unless the explicit `Reveal_Key` rule fires.

```
lemma key_secrecy:
  all-traces
  "all P k #i.
    ((Provision( P, k ) @ #i) & (not (Ex #r. Reveal( P ) @ #r)))
    (not (Ex #j. K( k ) @ #j))"
/*
guarded formula characterizing all counter-examples:
"Ex P k #i.
  (Provision( P, k ) @ #i)
 &
  (all #r. (Reveal( P ) @ #r) => F) & (Ex #j. (K( k ) @ #j))"
*/
simplify
solve( !KU( ~K ) @ #vk )
  case Reveal_Key
  by contradiction /* from formulas */
qed
```

B.4 Lemma: `agreement`

A verifier `Commit` on (TS_v, CH, TR) implies the prover ran with the same parameters. The proof splits on `case P_Respond` (honest origin) and `case c_hmac` (adversary constructed the MAC), the latter closing under the secrecy of K_i.

```
lemma agreement:
  all-traces
  "all V P ts ch tr #i.
    ((Commit( V, P, <ts, ch, tr> ) @ #i) &
     (not (Ex #r. (Reveal( P ) @ #r) & (#r < #i)))) =>
    (Ex #j. (Running( P, <ts, ch, tr> ) @ #j) & (#j < #i))"
/*
guarded formula characterizing all counter-examples:
"Ex V P ts ch tr #i.
  (Commit( V, P, <ts, ch, tr> ) @ #i)
 &
  (all #r. (Reveal( P ) @ #r) => not (#r < #i)) &
  (all #j. (Running( P, <ts, ch, tr> ) @ #j) => not (#j < #i))"
*/
simplify
solve( V_Pending( V, P, ts, ch, tr ) -> #i )
  case V_Challenge
  solve( !VerifierKey( $V, ~P, K ) -> #i )
```

```
    case Provision
    solve( !KU( hmac(~K, <digest(~P, ~tr, 'good'), ~ts, ~ch>) ) @ #vk )
      case P_Respond
      by contradiction /* from formulas */
    next
      case c_hmac
      solve( !KU( ~K ) @ #vk.1 )
        case Reveal_Key
        by contradiction /* from formulas */
      qed
    qed
  qed
qed
```

B.5 Lemma: `unique_commit`

Unique commit: every `Commit` corresponds to a unique prover session, even across multiple verifiers.

```
lemma unique_commit:
  all-traces
  "all V P ts ch tr #i.
    ((Commit( V, P, <ts, ch, tr> ) @ #i) &
     (not ( #r. (Reveal( P ) @ #r) & (#r < #i)))) =>
    ((Ex #j. (Running( P, <ts, ch, tr> ) @ #j) & (#j < #i)) &
     (not (Ex V2 #i2. (Commit( V2, P, <ts, ch, tr> ) @ #i2) & (not (#i = #i2)))))"
/*
guarded formula characterizing all counter-examples:
"Ex V P ts ch tr #i.
  (Commit( V, P, <ts, ch, tr> ) @ #i)
 &
  (all #r. (Reveal( P ) @ #r)  not (#r < #i)) &
  (((all #j. (Running( P, <ts, ch, tr> ) @ #j) => not (#j < #i)) |
    (Ex V2 #i2. (Commit( V2, P, <ts, ch, tr> ) @ #i2) & not (#i = #i2))))"
*/
simplify
solve( (all #j. (Running( P, <ts, ch, tr> ) @ #j) => not (#j < #i))  |
       (Ex V2 #i2. (Commit( V2, P, <ts, ch, tr> ) @ #i2) & not (#i = #i2)) )
  case case_1
  solve( V_Pending( V, P, ts, ch, tr ) -> #i )
    case V_Challenge
    solve( !VerifierKey( $V, ~P, K ) -> #i )
      case Provision
      solve( !KU( hmac(~K, <digest(~P, ~tr, 'good'), ~ts, ~ch>) ) @ #vk )
        case P_Respond
        by contradiction /* from formulas */
      next
```

```
        case c_hmac
        solve( !KU( ~K ) @ #vk.1 )
          case Reveal_Key
          by contradiction /* from formulas */
        qed
      qed
    qed
  qed
next
  case case_2
  solve( (#i < #i2)  (#i2 < #i) )
    case case_1
    solve( V_Pending( V, P, ts, ch, tr ) -> #i )
      case V_Challenge
      solve( !VerifierKey( $V, ~P, K ) -> #i )
        case Provision
        solve( V_Pending( V2, ~P, ~ts, ~ch, ~tr ) -> #i2 )
          case V_Challenge
          by contradiction /* cyclic */
        qed
      qed
    qed
  next
    case case_2
    solve( V_Pending( V, P, ts, ch, tr ) -> #i )
      case V_Challenge
      solve( !VerifierKey( $V, ~P, K ) -> #i )
        case Provision
        solve( V_Pending( V2, ~P, ~ts, ~ch, ~tr ) -> #i2 )
          case V_Challenge
          by contradiction /* cyclic */
        qed
      qed
    qed
  qed
qed
```

B.6 Lemma: `no_replay`

No two verifier commits share the same (P, TS_v), i.e. the symbolic counterpart of the Phase 4 Step 1 freshness check.

```
lemma no_replay:
  all-traces
  " V V2 P ts ch1 tr1 ch2 tr2 #i1 #i2.
    ((Commit( V, P, <ts, ch1, tr1> ) @ #i1) &
     (Commit( V2, P, <ts, ch2, tr2> ) @ #i2)) =>
```

```
    (#i1 = #i2)"
/*
guarded formula characterizing all counter-examples:
"Ex V V2 P ts ch1 tr1 ch2 tr2 #i1 #i2.
  (Commit( V, P, <ts, ch1, tr1> ) @ #i1) |
  (Commit( V2, P, <ts, ch2, tr2> ) @ #i2)
 &
  not (#i1 = #i2)"
*/
simplify
solve( (#i1 < #i2)  | (#i2 < #i1) )
  case case_1
  solve( V_Pending( V, P, ts, ch1, tr1 ) -> #i1 )
    case V_Challenge
    solve( !VerifierKey( $V, ~P, K ) -> #i1 )
      case Provision
      solve( V_Pending( V2, ~P, ~ts, ch2, tr2 ) -> #i2 )
        case V_Challenge
        by contradiction /* cyclic */
      qed
    qed
  qed
next
  case case_2
  solve( V_Pending( V, P, ts, ch1, tr1 ) -> #i1 )
    case V_Challenge
    solve( !VerifierKey( $V, ~P, K ) -> #i1 )
      case Provision
      solve( V_Pending( V2, ~P, ~ts, ch2, tr2 ) -> #i2 )
        case V_Challenge
        by contradiction /* cyclic */
      qed
    qed
  qed
qed
```

B.7 Lemma: attestation_soundness

```
lemma attestation_soundness:
  all-traces
  "all V P ts ch tr #i.
    ((Accept( V, P, ts, ch, tr ) @ #i) &
     (not ( #r. (Reveal( P ) @ #r) & (#r < #i)))) =>
    (Ex #j.
      ((Running( P, <ts, ch, tr> ) @ #j) & (ProverRanWith( P, 'good' ) @ #j)) &
      (#j < #i))"
```

```
/*
guarded formula characterizing all counter-examples:
"Ex V P ts ch tr #i.
  (Accept( V, P, ts, ch, tr ) @ #i)
 &
  (all #r. (Reveal( P ) @ #r) => not (#r < #i)) &
  (all #j.
    (Running( P, <ts, ch, tr> ) @ #j) & (ProverRanWith( P, 'good' ) @ #j)
   =>
    not (#j < #i))"
*/
simplify
solve( V_Pending( V, P, ts, ch, tr ) -> #i )
  case V_Challenge
  solve( !VerifierKey( $V, ~P, K ) -> #i )
    case Provision
    solve( !KU( hmac(~K, <digest(~P, ~tr, 'good'), ~ts, ~ch>) ) @ #vk )
      case P_Respond
      by contradiction /* from formulas */
    next
      case c_hmac
      solve( !KU( ~K ) @ #vk.1 )
        case Reveal_Key
        by contradiction /* from formulas */
      qed
    qed
  qed
qed
```