A^2RID - Anonymous Direct Authentication and Remote Identification of Commercial Drones

Eva Wisse, Pietro Tedeschi, Savio Sciancalepore, and Roberto Di Pietro

***Abstract*—The recent worldwide introduction of RemoteID (RID) regulations forces all Unmanned Aircrafts (UAs), a.k.a. drones, to broadcast in plaintext on the wireless channel their identity and real-time location, for accounting and monitoring purposes. Although improving drones' monitoring and situational awareness, the RID rule also generates significant privacy concerns for UAs' operators, threatened by the ease of tracking of UAs and related confidentiality and privacy concerns connected with the broadcasting of plaintext identity information.**
In this paper, we propose A^2RID, a protocol suite for anonymous direct authentication and remote identification of heterogeneous commercial UAs. A^2RID integrates and adapts protocols for anonymous message signing to work in the UA domain, coping with the constraints of commercial drones and the tight real-time requirements imposed by the RID regulation. Overall, the protocols in the A^2RID suite allow a UA manufacturer to pick the configuration that best suits the capabilities and constraints of the drone, i.e., either a processing-intensive but memory-lightweight solution (namely, $CS-A^2RID$) or a computationally-friendly but memory-hungry approach (namely, $DS-A^2RID$). Besides formally defining the protocols and formally proving their security in our setting, we also implement and test them on real heterogeneous hardware platforms, i.e., the Holybro X-500 and the ESPcopter, releasing open-source the produced code. For all the protocols, we demonstrated experimentally the capability of generating anonymous RemoteID messages well below the time bound of 1 second required by RID, while at the same time having quite a limited impact on the energy budget of the drone.

***Index Terms*—Unmanned Aerial Vehicles; Privacy; Security; Privacy-Enhancing Technologies; Applied Security and Privacy.**

I. Introduction

Unmanned Aircrafts (UAs), a.k.a. drones, are gaining increasing momentum in Industry and Academia, thanks to the flexibility and enhanced mobility features they provide in several key application domains, e.g., surveillance, goods delivery, and search-and-rescue, to name a few [1]. In addition, leading business forecasting companies estimate the drones' market to grow from USD 8.15 billion in 2022 to USD 47.38 billion by 2029, with a CAGR of approx. 28.58%, estimating more than 9.64 million drones flying around by the same time [2], [3].

Such large numbers motivated recent significant efforts by several regional authorities to integrate UAs within the local airspace, for traffic management and safety issues. In this context, the US-based Federal Avionics Administration (FAA) was the first to take action, by introducing the Remote Identification (RID) regulation [4]. In a nutshell, RID requires all UAs to broadcast on the wireless channel, from take-off to landing, information such as the identity and location of the UA, with a minimum rate of 1 msg/s (see Sec. II-A for more details). At the same time, initiatives similar to RID are also planned in EU and China [5], [6].

Although solving traffic management and safety concerns, RID regulations also introduce significant privacy issues [7], [8]. Indeed, by simply eavesdropping on the wireless channel, passive adversaries might easily get the unique identity of the UA, its real-time location, and other sensitive information, such as the location of the related Ground Control Station (GCS). Through a longer, fully passive and stealthy observation, the adversary can also track the drone during regular operations and infer more private information about their operators, such as the place where they live, the usual flight source, path, and destination, as well as the location of storage sites of large commercial delivery companies (e.g., in case of drone-based deliveries by goods distribution companies) [9]. Such threats are also magnified by recent reports, documenting the leakage of drones identifiers on the public Internet [10]. In this context, if the broadcast RID messages were anonymous, UAs could protect the privacy of their operator(s), and make indiscriminate tracking much more challenging. At the same time, any solutions for anonymous remote identification should also guarantee the disclosure of the identity of possibly misbehaving UAs, i.e., when drones invade (accidentally or not) no-fly-zones. Moreover, such solutions should also cope with the limitations of UAs, mainly in terms of available processing and energy capabilities.

At the time of this writing, very few scientific contributions investigated anonymous remote identification of UAs within the framework of the RID regulation. In this context, a conference paper [11] of ours proposed *ARID*, the first solution for anonymous remote identification of UAs. ARID allows UAs to broadcast anonymous RID messages, where the long-term identity of the emitter is never revealed on the wireless channel. At the same time, whenever the invasion of

This is a personal copy of the authors. Not for redistribution. The final published version of the paper accepted in IEEE Internet of Things Journal is available through the IEEExplore Digital Library, with the DOI: 10.1109/JIOT.2023.3240477.

Eva Wisse is with the Eindhoven University of Technology (TU/e), Department of Mathematics and Computer Science, Eindhoven, Netherlands. e-mail: e.m.c.wisse@student.tue.nl.

Pietro Tedeschi is with the Technology Innovation Institute, Autonomous Robotics Research Center, Abu Dhabi, United Arab Emirates. e-mail: pietro.tedeschi@tii.ae

Savio Sciancalepore is with the Eindhoven University of Technology (TU/e), Department of Mathematics and Computer Science, Eindhoven, Netherlands. e-mail: s.sciancalepore@tue.nl.

Roberto Di Pietro is with the Division of Information and Computing Technology (ICT), College of Science and Engineering (CSE), Hamad Bin Khalifa University (HBKU), Doha, Qatar. e-mail: rdipietro@hbku.edu.qa.

a no-fly area is detected, Critical Infrastructure (CI) operators might forward the received messages to a Trusted Third-Party (TTP), such as the FAA, to disclose the long-term identity of the misbehaving UA and take action. However, ARID provides a brokered message authentication. Indeed, entities receiving ARID messages cannot directly and autonomously verify message authenticity, but they have to interact with the TTP to verify that received messages are neither forged nor replayed. As a result, the deployment of ARID and its integration into the RID framework might pose excessive management overhead on regulatory authorities. Moreover, being based on different entities, the networking architecture required by ARID does not match with current standardization activities, such as the ones carried out by the IETF WG *drip*, working on the standardization of the components and the network architecture for the integration of RID into national airspaces.

Contribution. In this paper, we make the following contributions.

- We propose A^2RID (acronym for Anonymous direct Authentication and Remote IDentification), the first protocol suite for anonymous direct authentication and remote identification of heterogeneous commercial drones.
- Within the A^2RID protocol suite, we propose and define three protocols, namely: (i) $CS-A^2RID$, for high-end UAs equipped with regular processing and energy capabilities, capable of running pairing-based cryptography schemes on board; (ii) $DS-CCA2-A^2RID$, for UAs with low processing and energy availability, but equipped with large storage space; and, (iii) $DS-CPA-A^2RID$, for UAs characterized by severely constrained processing, storage, and energy availability.
- Through the protocols listed above, we provide a solution for the UAs to broadcast anonymous RID messages, protecting their long-term unique identity from malicious eavesdroppers while being compliant with current RID regulations.
- For all the protocols above listed, we provide a rigorous protocol description within the network architecture for UA remote identification defined by the IETF Working Group (WG) *drip*, as well as a formal security proof, via the well-known automated verification tool ProVerif.
- To show the viability of the proposed solution, we implemented the protocols in the A^2RID protocol suite on heterogeneous commercial UAs, i.e., the Holybro X500 and the ESPcopter, characterized by very different processing, storage, and energy capabilities. We also released the corresponding source code as open-source, to foster the reproducibility and re-usability of our code and results [12], [13], as well as to stimulate further research in the field.
- Finally, we also report the results of an extensive experimental performance assessment of our solutions when run on real heterogeneous hardware, demonstrating that it is possible to achieve anonymous remote identification of UAs in ≈ 0.017 seconds on the Holybro X500 and within 0.22 seconds on the ESPcopter, i.e., well below the time limit of 1 second imposed by the RID regulation, even with severely constrained UAs.

Roadmap. The rest of this paper is organized as follows. Sec. II introduces preliminary notions, Sec. III reviews the related work, Sec. IV outlines the scenario and the adversarial model considered in our work, Sec. V provides the details of our solution, Sec. VI discusses the security features offered by our solution, Sec. VII provides a thorough performance evaluation, both via simulations and a real experimentation and, finally, Sec. VIII concludes the paper.

II. Background and Preliminaries

In this section, we introduce background material that will be helpful for the sequel of the manuscript. Sec. II-A provides an overview of the RID regulation, while Sec. II-B summarizes cryptography techniques and notions used in this manuscript.

A. RemoteID Specification

The RID rule was published first in April 2021 by the US-based FAA, and it is set to become mandatory for all UAs from September 2022 [4]. According to the RID specification, all UAs, almost independently from their weight and usage, should broadcast on the wireless channel the following information: (i) unique identifier, (ii) timestamp, (iii) current location, (iv) current speed, (v) location of any GCS, and finally, (vi) emergency status. Such information should be broadcasted in plaintext, from take-off to landing time, with a minimum rate of one message per second. The rule also suggests the adoption of the WiFi standard for messages broadcasting, due to its reasonable range and widespread adoption. Besides the *broadcast mode*, RID also defines a *unicast* mode, where UAs might be available on a given port to answer requests about their identity and location. At the same time, RID does not force UAs to integrate an Internet connection, but only to feature a module for the broadcast of wireless messages. When such a module is unavailable, the manufacturer can provide dedicated external modules after the deployment to make UAs compliant with RID.

Overall, the aim of the RID rule is to set a framework for accountability of UA operations, as well as identification of the owner of any flying UAs. However, note that network security issues connected with the integration of RID are specifically not addressed in the FAA rule. Finally, it is worth noting that the overall problem of UAs remote identification goes beyond the US borders, and also other geographical airspaces such as the EU, Russia, and China are taking initiatives toward regulating drones' flights [5], [6].

B. Cryptography Techniques and Notions

In this section, we recall as preliminaries the main building blocks used throughout the manuscript.

Public Key Encryption. Public Key Encryption (PKE) schemes allow to encrypt a message M using the public key of the recipient pk to a ciphertext c, such that the recipient only, in possession of the corresponding secret key sk, can decrypt the ciphertext and recover the plaintext message M.

Definition II.1. A public key encryption algorithm PKE consists of the following algorithms:
$PKE.KGen(1^k)$: on input a security parameter k, it outputs a secret decryption key sk and a public encryption key pk.
$PKE.Enc(pk, m)$: on input a plaintext message m and a public key pk, it outputs a ciphertext c.
$PKE.Dec(sk, c)$: on input a ciphertext c and a secret decryption key sk, it outputs the corresponding plaintext m.

Although any public-key encryption scheme can be used, in this manuscript, we use the well-known Elliptic Curve Integrated Encryption Scheme (ECIES) scheme [14].

Decisional Diffie-Hellman (DDH). The Decisional Diffie–Hellman (DDH) is an assumption commonly used in cryptography on the computational hardness of solving discrete logarithms problems in cyclic groups. Such an assumption is at the roots of the security of many protocols, including Cramer–Shoup cryptosystems (see below). Assume $\mathbb{G}$ is a cyclic group of order q, with generator g, and a, b, c are random values $\in \mathbb{Z}_q$. According to the DDH assumption, given the distributions $\langle g^a, g^b, g^{ab}\rangle$ and $\langle g^a, g^b, g^c\rangle$, they are computationally indistinguishable in the security parameter $n = log(q)$ [15].

Cramer-Shoup Cryptosystem. Assume $\mathbb{G}$ is a cyclic group of prime order q, where q is large, m is a plaintext message encoded as an element of $\mathbb{G}$, and H a universal family of one-way hash functions mapping bit-strings into elements of $\mathbb{Z}_q$ [16].

Definition II.2. A Cramer-Shoup public key encryption algorithm CSC consists of the following algorithms:
$CSC.KGen(\mathbb{G}, \mathbb{Z}_q)$: on input a group $\mathbb{G}$ of prime order q, it generates random elements $g_1, g_2 \in \mathbb{G}$, and $x_1, x_2, y_1, y_2, z \in \mathbb{Z}_q$. Next, it computes the elements $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z$. The generated public key is the tuple $pk = (g_1, g_2, c, d, h, H)$, and the private key is $sk = (x_1, x_2, y_1, y_2, z)$.
$CSC.Enc(pk, m, r)$: on input a plaintext message $m \in \mathbb{G}$, a public key pk, and $r \in \mathbb{Z}_q$ it outputs the ciphertext $c = (u_1, u_2, e, \psi)$, where $u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = H(u_1, u_2, e), \psi = c^r d^{r\alpha}$.
$CSC.Dec(sk, c)$: on input a ciphertext $c = (u_1, u_2, e, \psi)$ and a secret key $sk = (x_1, x_2, y_1, y_2, z)$, the decryption algorithm computes $\alpha = H(u_1, u_2, e)$ and tests if $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} \stackrel{?}{=} \psi$. If this condition is verified, the algorithm outputs the plaintext $m = \frac{e}{u_1^z}$; otherwise it outputs $\perp$.

Digital Signature Schemes. Digital Signature (DSig) schemes allow a sender to produce a signed value σ for a message m, demonstrating to be the actual sender of the message.

Definition II.3. A digital signature scheme $DSig$ consists of the following algorithms:
$DSig.KGen(1^K)$: on input a security parameter k, it outputs a secret signing key sk and a public verification key pk.
$DSig.Sign(sk, m)$: on input a message m and a signing key sk, it outputs a signature σ.
$DSig.Vrf(pk, m, \sigma)$: on input a message m, a public key pk, and a signature σ, it outputs a bit $b \in 0, 1$, where 0 indicates that σ is not verified and 1 indicates that σ is verified.

Although any DSig scheme on elliptic curves can be used, in this manuscript, we use the scheme proposed by Boneh, Lynn, and Shacham [17].
Bilinear Pairings. Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative groups of prime order q, and g be a generator of $\mathbb{G}$. A map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is called a bilinear map if it satisfies the following properties:

1) Bilinearity: $\hat{e}(g^\alpha, g^\beta) = \hat{e}(g, g)^{\alpha\beta}$ for all $\alpha, \beta \in \mathbb{Z}_{||}^*$.
2) Non-degeneracy: There exist $\alpha, \beta \in \mathbb{G}$ such that $\hat{e}(\alpha, \beta) = 1$.
3) Computability: There exists an efficient algorithm to compute $\hat{e}(\alpha, \beta)$ for any $\alpha, \beta \in \mathbb{G}$.

Several bilinear pairing types exist, i.e., Type-1 (symmetric), Type-2, Type-3, and Type-4. In this paper, we use Type-1 pairing ($\mathbb{G}_1 = \mathbb{G}_2$) and Type-3 pairing ($\mathbb{G}_1 \neq \mathbb{G}_2$ and absence of any computable isomorphism). Interested readers can refer to the article by the authors in [18] and [19] for more details.
Structure-Preserving Signatures on Equivalence Classes. Structure-preserving signatures on equivalence classes allow, among other properties, to generate unlinkable message-signature pairs on elements of the same equivalence classes.

Definition II.4. A structure-preserving signatures on equivalence classes scheme S consists of the following algorithms.
$S.BGGen(1^K)$: on input a security parameter k, it outputs a bilinear group BG.
$S.KGen(BG, l)$: on input a bilinear group BG and an integer l, it outputs a secret key sk and a public key pk.
$S.Sign(m, sk)$: on input a message m and a secret key sk, it outputs a signature σ.
$S.ChgRep(m, \sigma, \rho, pk)$: on input a message m, a signature σ on m, a scalar ρ, and a public key pk, it outputs a message-signature pair (M', ρ'), being $M' = \rho \cdot M$.
$S.Vrf(m, \sigma, pk)$: on input a message m, a signature σ, and a public key pk, it outputs a bit $b \in 0, 1$, where 0 indicates that σ is not verified and 1 indicates that σ is verified.
$S.VKey(sk, pk)$: on input a secret key sk and a public key pk, it outputs a bit $b \in 0, 1$, where 0 indicates that the keys are not related to each other, while a 1 indicates that they are related to each other.

In this paper, we use the structure-preserving signatures on equivalence classes scheme reported in [20], and originally defined by the authors in [21].

Non-Interactive Zero-Knowledge Proofs. Non-Interactive Zero Knowledge Proofs (NIZKP) schemes allow a sender to prove a statement to a verifier, while allowing the sender to create proofs for such a statement offline, without an online interaction with the verifier.

Definition II.5. A NIZKP scheme NZ consists of the following algorithms.
$NZ.Setup(1^k)$: on input a security parameter k, it outputs a

common reference string crs.
$NZ.Proof(crs, x, w)$: on input a common reference string crs, a statement x, and a witness w, it outputs a proof π.
$NZ.Vrf(crs, x, \pi)$: on input a common reference string crs, a statement x, and a proof π, it outputs a bit $b \in 0, 1$, where 0 indicates that the statement is not verified, while a 1 indicates that the statement is verified.

In this paper, we use and adapt to our problem the Schnorr NIZKP scheme [22].

Signatures of Knowledge. Signature of Knowledge (SoK) schemes allow a sender, that has knowledge of a word, to sign a message while allowing a receiver to verify the knowledge of such a statement.

Definition II.6. A SoK scheme SoK consists of the following algorithms.
$SoK.Setup(1^k)$: on input a security parameter k, it outputs a common reference string crs.
$SoK.Sign(crs, x, w, m)$: on input a common reference string crs, a word x, a witness w, and a message m, it outputs a signature σ.
$SoK.Vrf(crs, x, m, \sigma)$: on input a common reference string crs, a word x, a message m, and a signature σ, it outputs a bit $b \in 0, 1$, where 0 indicates that the knowledge of the word x is not verified, while a 1 indicates that the knowledge of the word x is verified.

In this paper, we use the SoK scheme reported in [20] and initially defined by the authors in [23].
CPA-Full and CCA2-Full Anonymity. In line with the current literature on anonymous group signatures, in this work, we distinguish between CPA-Full and CCA-2 Full Anonymity [20].

Definition II.7. We define a group signature scheme as *CPA-Full Anonymous* if the scheme guarantees signer anonymity provided that the adversary cannot issue opening requests for specific signed messages.

Definition II.8. We define a group signature scheme as *CCA2-Full Anonymous* if the scheme guarantees signer anonymity also when the adversary can issue opening requests for specific signed messages.

It is worth noting that, without loss of generality, CCA2-Full Anonymity is stronger than CPA-Full Anonymity, as the former does not imply any constraints on the interactions between entities in the signature scheme. At the same time, as will become evident from our experimental evaluation (Sec. VII), CCA2-Full anonymity schemes are usually more processing-intensive and energy-hungry than CPA-Full anonymity schemes, which might become relevant in constrained scenarios.

III. Related Work

A few contributions investigated security and privacy issues connected with the adoption of the RID regulation.

The authors in [24] and [25] tried to integrate the concept of mix zones, well-known in the Vehicular Ad-Hoc Networks (VANET) area for pseudonym exchange, within the UA research domain. However, mix zones require communication with infrastructure elements, which might not always be available in UA operations. The authors in [26] proposed a decentralized traffic management protocol providing many security services, including integrity and confidentiality, and mainly focus on the security of the interactions between regulatory entities. Thus, drones' anonymity and authenticity are not considered. The authors in [27] propose the integration of the Hyperledger Iroha blockchain for the management of drones' remote identification. Based on the proposed system architecture, drones register to the blockchain using their public key and related certificate. At run-time, Internet-connected drones provide remote identification information directly on the blockchain, while drones not equipped with an Internet connection delegate the closer GCS to write such information on the blockchain. Thus, although providing authentication and integrity of drones' messages, anonymity and privacy of drones are not considered in the design of such a solution. The authors in [28] took into account the RID regulation, but focused on location privacy rather than anonymity. To this aim, they integrate the Differential Privacy (DP) tool into the RID regulation, allowing drones to broadcast an obfuscated location, so as to preserve location privacy. Thus, the authors did not investigated specifically anonymity and message authenticity.

Besides scientific contributions, a few commercial solutions for drone remote identification are starting to appear on the market. Examples include ScaleFyt designed by Thales [29], the Broadcast Location and Identification Platform (BLIP) designed by Unifly [30], and the Secure Airspace Integrated Management (SIAM) tool provided by RelmaTech [31]. All these solutions rely on the authentication and anonymity services provided by the LTE cellular technology, which is, however, not the communication technology currently envisioned by the RID regulation and the DRIP WG of the IETF.

It is worth noting that anonymity for broadcasting devices has also been considered by a few works outside the UA area. In the area of VANET, many contributions investigated anonymity issues for vehicles, such as [32], [33], and [34], just to name a few. However, such schemes either require a persistent external connection or infrastructure elements, which might not always be available for UA operations, e.g., in remote areas.

In the avionic research domain, the authors in [35] proposed to generate pseudonyms for online aircrafts using an on-purpose Trusted Registration Authority, assumed to be always available online. Such an approach is not feasible for UAs, as most of them cannot rely on a persistent external connection. Similarly, in the maritime research domain, the authors in [36] proposed to anonymize vessels identity using pseudonyms generated by an online trusted authority, while the authors in [37] proposed to adopt the IEEE P1609.02 scheme for pseudonymous generation and disclosure. Note that standards available for pseudonyms management, such as the IEEE P1609.02, consider unicast scenarios, and not broadcast-type interactions like the ones in RID.

Finally, we highlight that this contribution extends and

improves our previous conference paper presented in [11] by: (i) adapting heterogeneous protocols for anonymous group signatures, traditionally adopted for online voting, to work in real-time scenarios for the anonymous identification of UAs; (ii) allowing generic receivers, namely, *observers*, to verify autonomously the authenticity of the messages received by the UA, without relying on a TTP constantly online; (iii) reducing further the overhead of UA anonymous identification on avionics authorities, such as the FAA, by removing the need for expensive message authenticity management issues; and (iv) integrating our proposed A^2RID protocol suite within the standardized network architecture currently defined by the IETF WG *drip*, specifically meant to ease the worldwide adoption of the RID regulation. Therefore, our proposed solution emerges as the first of its kind. We will provide more details on these aspects in the next sections.

IV. Scenario and Adversary Model

In this section, we introduce the scenario (Sec. IV-A) and adversarial model (Sec. IV-B) considered in our work. Moreover, we also derive the most important design and security requirements for an anonymous RID solution (Sec. IV-C).

A. System Model

The scenario assumed in this work, in line with the architecture defined by the Drone Remote Identification Protocol (DRIP) WG [38], is depicted in Figure 1.

Figure 1. Reference scenario. Several UAs fly in a given area and broadcast messages compliant with the RID rule. Observers, such as CIs, listen to RID messages to monitor such areas and enforce protection of no-fly areas.

We assume several UAs, flying around a given area to complete their intended mission. The UAs are produced by different manufacturers and operated by several service providers, not trusting each other. In line with common capabilities and equipment, we assume the UAs to be equipped with a Global Positioning System (GPS) module for precise location estimation, and a wireless communication module, allowing them to emit broadcast RID messages. Such a communication module is typically a Wi-Fi transceiver, allowing messages to be received in a radius of up to 1 km around the UAs' location. In line with the reference architecture in [38], we do not assume any Internet connectivity on board of UAs. This is consistent with the most general and challenging scenario, where the UA has to operate in a remote region, not offering support for Internet connectivity.

We also assume the presence of a trusted authority, namely, the Unmanned Service Supplier (USS). In line with the definition in [38], the USS sits in between the UAs operators and the traffic management entities, providing services such as real-time traffic monitoring and planning, data archiving, and airspace and violation control. As such, the USS hosts a Public Information Registry (PbIR), i.e., a registry of public UA' information, to be used by observers (see definition below) to verify RID messages emitted by UAs.

Our scenario also includes *observers*, i.e., generic wireless receivers able to listen to the radio channel used by UAs to deliver wireless RID messages. Such observers can be either standalone (i.e., smartphones of generic users and hobbyists), namely, *General Public Observers*, or associated to more extensive sensor networks, such as in the case of monitoring networks of CIs, namely, *Public Safety Observers*. In the latter, they monitor invasions of specific no-fly areas. They can interact online with the USS to report such invasions and allow the USS to obtain information about the identity of the UA violating flight restrictions on a given no-fly area. As an essential requirement, we also require such observers to be able to verify directly, i.e., autonomously, the authenticity of received RID messages. Thus, although they might interact with the USS once to obtain public identification parameters of UAs stored in the PbIR, they cannot contact the UAS regularly to verify RID messages' authenticity. Note that this is a reasonable requirement, as having each observer interacting with the USS regularly to check every message would overload the USS, as well as require powerful resources to handle frequent cryptography operations and Internet traffic. For the sake of better readability, from now on we will always refer to *Public Safety Observers* as simply observers.

Finally, for the readers' convenience, we report the main notation used below in Tab. I, along with a brief description.

B. Adversary Model

The adversary assumed in this contribution ($\mathcal{A}$) has two main objectives, i.e., obtaining the long-term identity of a specific UA, and impersonating a specific UA while carrying out malicious activities using its (spoofed) identity.

To reach such objectives, $\mathcal{A}$ can carry out both passive and active attacks. On the one hand, $\mathcal{A}$ is a global, frequency-unbounded, and spatially-unlimited passive eavesdropper, capable of detecting and receiving any message broadcasted by the UA, independently from the communication frequency and modulation. We also assume $\mathcal{A}$ can receive such wireless messages independently from the location of the UA, i.e., through a network of receiving antennas deployed on the ground. On the other hand, $\mathcal{A}$ can also replay previously received packets and generate ad-hoc rogue messages, trying to impersonate legitimate UAs. As such, $\mathcal{A}$ possesses all the capabilities to be in line with the so-called *Dolev-Yao* attacker model, commonly assumed among the most powerful attacker models considered in the literature [39].

Note that we do not make any distinction between internal and external adversaries. Indeed, as shown in Sec. VI, being a member of a group does not provide any advantage to the

Table I
NOTATION AND BRIEF DESCRIPTION.

Notation	Description
UA_i	Generic Unmanned Aircraft.
rcv	Generic observer.
T	Unmanned Service Supplier.
H	Hashing function.
p	Groups order.
$\mathcal{G}_1$, $\mathcal{G}_2$	Bilinear groups, with generators P and $\hat{P}$.
$\mathcal{G}_T$	Prime number of k bits.
e	Pairing operation.
sk_R, pk_R	Private and public key of the structure-preserving signatures on equivalence classes scheme in $DS-A^2RID$.
sk_O, pk_O	Private and public key of the PKE scheme in $DS-A^2RID$.
sk_i, pk_i	Private and public key of the DSig scheme for the i-th UA in $DS-A^2RID$.
crs_j, crs_o	Reference strings of the NIZKP scheme in $DS-A^2RID$.
crs_S	Reference string of the SoK scheme in $DS-A^2RID$.
gpk	Group Public Key in $DS-A^2RID$.
ik	Issuing key in $DS-A^2RID$.
ok	Opening key in $DS-A^2RID$.
q ,r, χ, y, ρ_k	Nonces extracted in $\mathcal{Z}_p$.
$\hat{C}_{j_i}$	Encrypted witness in $DS-A^2RID$.
σ_{j_i}	Signed witness in $DS-A^2RID$.
reg_i	Registration Table entry for UA_i.
σ'	Signed UA credential in $DS-A^2RID$.
gsk_i	Group secret key for UA_i in $DS-A^2RID$.
t_k	Timestamp of a RID message.
pos_k	Location of UA_i at time t_k.
v_k	Speed of UA_i at time t_k.
em_k	Emergency status of UA_i at time t_k.
$\sigma_{1,k}$, $\sigma_{2,k}$	Signatures deliverd by UA_i at time t_k in $DS-A^2RID$.
ID_g	Group Identifier.

adversary, and neither allows $\mathcal{A}$ to successfully carry out the attacks previously mentioned.

In line with the main objectives of the RID specification, our contribution does not take into account Denial of Service (DoS) attacks. Indeed, the RID architecture has been designed specifically to make spoofing and replay attacks very hard, while not dealing with DoS attacks. In this context, our work aims to strengthen protection against replays and spoofing, while also providing anonymity to emitting UAs. Finally, note that our solution cannot unconditionally protect an UA against passive tracking. Indeed, as acknowledged by several contributions in the literature, broadcasted locations can be used to link messages among them, and track a specific UAs even without being aware of its long-term identity. We refer the readers to specific works protecting locations disclosed in RID messages for solutions to the cited specific threat [28].

C. Design and Security Requirements

Based on the scenario and adversarial model described above, we can derive the following system and security requirements for a secure and anonymous RID-compliant solution.

- **UA Anonymity.** To avoid privacy issues, the UA should stay anonymous while emitting RID messages. Such a requirement implies that the long-term identity of a specific UA should never be available to observers, and observers should not be able to detect if two RID messages are emitted by the same UA. As the observers do not need UA identification information, both CPA-Full and CCA2-Full anonymity might fulfill this requirement.
- **UA Messages Authenticity.** All messages emitted by legitimate UAs should be verifiable as authentic, i.e., emitted by a legitimate entity and not forged. Although replay attacks are always possible due to the broadcast nature of RID communications, they should be mitigated as much as possible. The fulfillment of this requirement is necessary to avoid false invasion claims and attributions (i.e., attempts to falsely locate a drone where it should not be allowed to fly), aimed at unveiling the identity of a specific UA.
- **Direct Message Verification on Observers.** Observers receiving RID messages should be able to verify the authenticity of such messages directly and autonomously, without interacting with third-parties. Such a requirement is necessary to consider observers with scarce or unstable Internet connections, as well as to avoid performance bottlenecks and excessive management on the USS.
- **UA Identity Disclosure upon Invasion Detection.** Upon invasion detection by an observer, the USS should always be able to: (i) verify the invasion, and (ii) disclose the long-term identity of the UA causing the invasion, for further action.

In the following sections, we describe several solutions fulfilling all the above-described requirements, characterized by different features and trade-offs.

V. THE A^2RID PROTOCOL SUITE

In this section, we provide the details of A^2RID, our novel protocol suite for anonymous direct authentication and remote identification of commercial UAs. We first introduce the involved actors in Sec. V-A. We then describe the $CS-A^2RID$ protocol in Sec. V-B, and the two variants of the $DS-A^2RID$ protocol in Sec. V-C.

A. Actors

A^2RID involves the following actors.

- **UA**. It is a generic drone, following the RID specification. As such, it emits periodic broadcast wireless messages reporting, among other information, its location and identification. The scope of our work is to provide anonymity to the emitted messages, such that message authenticity can be verified while not revealing the long-term identity of the drone.
- **USS T.** It is an authority, in charge of releasing cryptography parameters to the UA for anonymous remote identification. The manufacturer of the UA interacts with T only offline, in the Setup phase. At the same time, T is available online on a public interface for queries by observers deployed by CI operators, to verify invasions of no-fly areas and unveil the location of the misbehaving UAs.

- **Observer** rcv**.** It is a generic receiver, deployed to monitor invasions of no-fly areas of a CI. It can directly verify the authenticity of broadcast messages emitted by UA, without being aware of the long-term identity of the emitting entity. In case an invasion of the related no-fly area is detected, the observer can forward the received messages to the USS, for identity disclosure and enforcement of sanctions.

In the following, we describe two variants of the A^2RIDprotocol, namely, $CS-A^2RID$ and $DS-A^2RID$, suitable for integration in medium-end and low-end UAs, respectively.

B. Option #1: Camenisch-Lysyanskaya (CS) Scheme

The $CS-A^2RID$ scheme, inspired by the protocol in [40], is a dynamic scheme with distributed authorities. The security parameter $\kappa \in \mathbb{N}$ uses cyclic groups $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T = \langle g_T \rangle$ of prime order $\mathcal{Q}$ with $|\mathcal{Q}| = \kappa$, a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, and the generator $g_T = \hat{e}(g, g)$. Moreover, it also relies on a collision-resistant hashing function $\mathcal{H} : 0, 1^* \rightarrow \mathbb{Z}_Q$. We identify four phases, i.e, the *Setup Phase*, *UA Joining Phase*, *Online Phase*, and *Disclosure Phase*.

Setup Phase. This phase is executed at the boot of the overall system, to setup the necessary cryptography parameters on the relevant entities, i.e., the USS and the UA. Specifically, this phase includes the following operations.

1) The USS selects a security parameter κ, $x, y \in_R \mathbb{Z}_Q$, and sets $X = g^x, Y = g_T^y$.
2) The USS also selects $h \in_R \mathbb{G}_T \backslash \{1_{\mathbb{G}_T}\}, x_1, \ldots, x_5 \in_R \mathbb{Z}_Q$ and sets $\mathbf{y}$ as in Eq. 1.

$$\begin{cases} y_1 = g_T^{x_1} h^{x_2}, \\ y_2 = g_T^{x_3} h^{x_4}, \\ y_3 = g_T^{x_5}. \end{cases} \tag{1}$$

   Then, the USS outputs (gpk, ik, ok, reg) such that:
   - the group public key is set to $gpk = (\mathcal{Q}, \mathbb{G}, \mathbb{G}_T, g, g_T, \hat{e}, X, Y, h, y_1, y_2, y_3)$
   - the secret issuing key is set to $ik = (gpk, x, y)$
   - the secret opening key is set to $ok = (gpk, x_1, \ldots, x_5)$
   - the registration list reg is initially empty.
3) Finally, the USS publishes the group public key gpk online, to make it available to all UAs interesting in joining the system and to observers aiming to verify anonymous RID broadcasts.

As the USS is trusted, we assume that this phase is performed in a trusted way. Thus, the elements $x, y, h, x_1, \ldots, x_5$ are chosen independently at random from $\mathbb{Z}_Q$, $\mathbb{G}_T \backslash, \{1_{\mathbb{G}_T}\}$, respectively. Note that the tuple (h, y_1, y_2, y_3) is a public key of the *Cramer-Shoup* encryption scheme over the group $\mathbb{G}_T$ and $(x_1, \ldots, x_5)$ is the corresponding private key [16].

UA Joining Phase. This phase is executed offline, every time a new UA UA_i would like to join the system, i.e., to start operating anonymously while complying with the RemoteID specification. This phase includes the following operations.

1) UA_i picks a nonce $k_i \in_R \mathbb{Z}_Q$, and it sets $P_{i,1} = g^{k_i}$. The UA_i also picks a nonce $rk \in_R \mathbb{Z}_Q$, sets $R = g^{rk}$, and it computes the hash $\eta_1 = \mathcal{H}(g, R)$. Further, it sets $Sk = \eta_1 k_i + rk$, and sends the tuple $\eta_1, Sk, P_{i,1}$ via a secure channel to the USS.
2) The USS T executes the following procedures: (i) it proceeds to verify if $P_{i,1} \in \mathbb{G}$ is a point on the curve, and (ii) it computes the value $\gamma = \frac{g^{Sk}}{P_{i,1}^{\eta_1}}$ and the hash $\eta_2 = \mathcal{H}(g, \gamma)$. If $\eta_1 \stackrel{?}{=} \eta_2$ holds, it picks a nonce $r \in_R \mathbb{Z}_Q$ and computes $a_i = g^r$, $b_i = a_i^y$, and $c_i = a_i^x P_{i,1}^{rxy}$. Then, it delivers to UA_i the membership certificate (a_i, b_i, c_i) via a secure channel.
3) Finally, the USS T computes $P_{i,2} = \hat{e}(P_{i,1}, g)$ and stores an entry for UA_i in the registration list as $reg_i = (P_{i,1}, P_{i,2})$.
4) At message reception from the USS, UA_i stores its secret signing key as $gsk_i = (gpk, k_i, a_i, b_i, c_i)$.

Note that the tuple (a_i, b_i, c_i), part of the secret signing key gsk_i, represents an ordinary Camenisch-Lysyanskaya signature on the element k_i. If all the verification steps described above end successfully, UA_i stores the pair (gsk_i, gpk) as its own private and public group keys, to be used for anonymously signing messages when deployed.

Online Phase. This phase is executed at runtime, both on any generic observer rcv and operational UA UA_i, which would like to send RID messages, while staying anonymous. It involves the following operations.

1) Assume that at the time t_k the UA UA_i is required to deliver a *RemoteID* message. We denote with ID_g the UA group ID, $pos_k = (lat_k, lon_k, alt_k)$ the latitude, longitude, and altitude coordinates of the location occupied by UA_i, as obtained through the integrated GPS module, $pos_{u,k} = (lat_{u,k}, lon_{u,k}, alt_{u,k})$ the latitude, longitude, and altitude coordinates of the location occupied by UA Ground Station (or controller) at the time t_k, $v_k = (v_{x,k}, v_{y,k}, v_{z,k})$ as the 3-D vector of the speed in m/s of the UA as obtained through integrated modules at the time t_k, and em_k the emergency status value of the UA. We denote $m_k = (ID_g, pos_k, v_k, pos_{u,k}, t_k, em_k)$.
2) The signature generation algorithm takes as input the secret signing key gsk_i of UA_i, the group public key gpk, and a generic message $m \in \{0, 1\}^*$. First, UA_i computes $P_{i,2} = g_T^{k_i} = \hat{e}(P_{i,1}, g)$. Then, UA_i encrypts $P_{i,2}$ under the group opener's public key, i.e., it chooses $u \in_R \mathbb{Z}_Q$ and computes the elements of the vector $\mathbf{T}$ as in Eq. 2.

$$\begin{cases} T_1 = g_T^u, \\ T_2 = h^u, \\ T_3 = y_1^u P_{i,2}, \\ T_4 = y_2^u y_3^{u\mathcal{H}(T_1 || T_2 || T_3)}. \end{cases} \tag{2}$$

3) Then, UA_i selects $r, r' \in_R \mathbb{Z}_Q$ and computes a blinded version of the certificate, namely, $\tilde{\sigma} = (T_5, T_6, T_7)$,

according to Eq. 3.

$$\begin{cases} T_5 = a_i^{r'}, \\ T_6 = b_i^{r'}, \\ T_7 = c_i^{r'r}. \end{cases} \tag{3}$$

4) UA_i also picks a nonce $\rho, \mu, \nu \in_R \mathbb{Z}_Q$, and performs the operations in Eq. 4.

$$\begin{cases} R_1 = \frac{\hat{e}(g,T_7)^{\rho}}{\hat{e}(X,T_6)^{\mu}}, \\ R_2 = g_T^{\nu}, \\ R_3 = h^{\nu}, \\ R_4 = y_1^{\nu} g_T^{\mu}, \\ R_5 = y_3^{\nu \mathcal{H}(T_1||T_2||T_3)}. \end{cases} \tag{4}$$

Then, using the plaintext RID message m_k, the UA_i computes the hash of the generated parameters and the message m_k as in Eq. 5.

$$\begin{cases} R = (R_1, R_2, R_3, R_4, R_5), \\ c = \mathcal{H}(R, g, g_T, X, Y, h, y_1, y_2, y_3, m_k), \\ S_\rho = \frac{c}{r} + \rho, \\ S_\mu = ck_i + \mu, \\ S_\nu = cu + \nu. \end{cases} \tag{5}$$

5) Finally, UA_i computes the signature σ_k over the message m_k as in Eq. 6.

$$\sigma_k = (S_\rho, S_\mu, S_\nu, T_1, T_2, T_3, T_4, T_5, T_6, T_7). \tag{6}$$

UA_i delivers in broadcast the message m_k, the signature σ_k, and the value $mode_k = 0$, indicating the usage of the $CS - A^2RID$ protocol of the A^2RID protocol suite. Note that, in the above signature generation algorithm, the values (T_1, T_2, T_3, T_4) are a Cramer-Shoup encryption of $P_{i,2}$ under the group opener's public key (h, y_1, y_2, y_3) [16]. The ciphertext can be decrypted using the private key $(x_1, \ldots, x_5)$. Thus, the signature S proves that the signer is in possession of a k_i, satisfying $P_{i,1} = g^{k_i}$ and proving, in turn, that the signer has a valid signing key gsk_i.

6) On the receiving side, every observer rcv receiving the message can directly authenticate it and verify its contents. Specifically, rcv extracts the message m_k and parses the signature σ_k and the mode $mode_k$. If $mode_k = 1$, indicating that the message is secured with $CS - A^2RID$, it proceeds as in the next steps.
7) First, rcv computes the hash $H = \mathcal{H}(T_1||T_2||T_3)$, and the following parameters, as in Eq. 7.

$$\begin{cases} R_1' = \frac{\hat{e}(g,T_7)^{S_\rho}}{\hat{e}(X,T_6)^{S_\mu} \hat{e}(X,T_5)^{H}}, \\ R_2' = \frac{g_T^{S_\nu}}{T_1^H}, \\ R_3' = h^{S_\nu} - T_2^H, \\ R_4' = (y_1^{S_\nu} g_T^{m_k}) - T_3^H, \\ R_5' = \frac{y_2^{S_\nu} y_3^{H S_\nu}}{T_4^H}. \end{cases} \tag{7}$$

8) Then, rcv computes the hash of the message as in the following Eq. 8.

$$\begin{cases} R' = (R_1', R_2', R_3', R_4', R_5'), \\ c' = \mathcal{H}(R', g, g_T, X, Y, h, y_1, y_2, y_3, m_k). \end{cases} \tag{8}$$

9) If $c \stackrel{?}{=} c'$ applies and $\hat{e}(T_5, Y) \stackrel{?}{=} \hat{e}(g, T_6)$, the signature is deemed valid and authentic. Otherwise, the signature algorithm outputs an error, rejecting the message.

Disclosure Phase. This phase is triggered by the generic observer rcv and executed on the USS, whenever an invasion of the no-fly area of rcv by an anonymous UA is detected. It involves the following operations.

1) Assume rcv realizes that the received message $(m, mode, \sigma_{\mathbf{k}})$, verified as authentic, determines an invasion of the monitored no-fly zone. Then, rcv forwards this message to the USS T.
2) The USS retrieves the secret opening key ok and the group public key gpk corresponding to the message m_k, as well as the registration list reg. First, the USS verifies the signature σ_k, by executing the signature verification algorithm, checking that $T_4 \stackrel{?}{=} T_1^{x_3 + x_5 H} T_2^{x_4}$. If the check is verified, it proceeds further; otherwise, it outputs an error and rejects the received message.
3) Then, the USS computes the value $P_{i,2} = \frac{T_3}{(T_1^{x_1} T_2^{x_2})}$, and it finds the UA UA_i such that $reg_i = (P_{i,1}, P_{i,2})$. If no such UA_i exists, it outputs an error. Otherwise, the USS can identify UA_i and return an acknowledgment to rcv, without revealing the identity of such UA.

C. Option #2: Derler-Slamanig (DS) Schemes

Although the $CS - A^2RID$ scheme described in the previous section allows UAs to sign RID messages anonymously, it requires pairing operations during signature generation, hardly achievable on very constrained UAs within the time bounds defined by the RID specification (1 s). To this aim, in this section, we introduce the adaptation of the Derler-Slamanig (DS) schemes proposed in [20] to our scenario, namely, $DS - CCA2 - A^2RID$ and $DS - CPA - A^2RID$. We anticipate that, as a distinguishing feature, both $DS - CCA2 - A^2RID$ and $DS - CPA - A^2RID$ do not require the signing UA to use pairing operations, thus being feasible for integration into constrained UAs for anonymous remote identification. In the following, we describe all the operations required by the protocols in our setting. We delve into the details of the instantiation of the structure-preserving signatures on equivalence classes, SoK, and NIZKP scheme while, due to their more generic features and ease of notation, we report the public-key encryption setup, public-key encryption, public key decryption, digital signature scheme setup, digital signature, and digital signature verification operations through the general notation $PKE.KGen(\cdot)$ $PKE.Enc(\cdot)$, $PKE.Dec(\cdot)$, $DSig.KGen(\cdot)$, $DSig.Sgn(\cdot)$, and $DSig.Vrf(\cdot)$, respectively. Interested readers can find the details of the used PKE and DSig schemes in [14] and [17], respectively.

In line with the previous scheme, we identify four protocol phases, i.e, the *Setup Phase*, *UA Joining Phase*, *Online Phase*, and *Disclosure Phase*.

Setup Phase. This phase is executed at the boot of the overall system, to setup the necessary cryptography parameters on the relevant entities, i.e., the USS and the UA. Specifically, this phase includes the following operations.

- The USS T takes a bilinear group $BG = (p, \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, e, P, \hat{P})$, where p is the order of the groups $\mathcal{G}_1$ and $\mathcal{G}_2$, $\mathcal{G}_T$ is a prime number of bit-size κ, e is a pairing, and P and $\hat{P}$ are generators of $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively.
- The USS T generates the key-pair of the structure-preserving signatures on equivalence classes scheme, namely, (sk_R, pk_R). To this aim, on input BG and a vector length l, it chooses $(x_i)_{i\in|l|} \leftarrow (\mathcal{Z}_p)^l$, sets $sk_R = (x_i)_{i\in|l|}$, and computes $pk_R = (x_i\hat{P})_{i\in|l|}$.
- The USS T initializes the key pair of the PKE algorithm, namely, (sk_O, pk_O), as $(sk_O, pk_O) \leftarrow PKE.KGen(1^k)$.
- The USS T generates two reference strings of the NIZKP scheme, i.e., crs_j and crs_O, through the application of a string generation algorithm $CRSGen(gk)$, being $gk = (\mathcal{G}, q, g)$, where $\mathcal{G}$ is a group of prime order q and generator g.
- The USS T generates the reference string of the SoK scheme, i.e., crs_S, through the string generation algorithm $CRSGen(gk)$.
- The USS sets the group public key $gpk = (pk_R, pk_O, crs_j, crs_O, crs_S)$, the issuing key $ik = sk_R$, and the opening key $ok = sk_O$.
- Finally, the USS publishes the group public key gpk online, to make it available to all UAs interested in joining the system and to observers aiming to verify anonymous RID broadcasts.

At the same time, the generic UA UA_i initializes the key pair of the DSig algorithm, namely, (sk_i, pk_i), as $(sk_i, pk_i) \leftarrow DSig.KGen(1^K)$. Then, it makes pk_i publicly available.

UA Joining Phase. This phase is executed offline, every time a new UA UA_i would like to join the system, i.e., to start operating anonymously while complying with the RemoteID specification. This phase includes the following operations.

- Assume UA_i would like to obtain the cryptography materials necessary to join the group. UA_i first extracts two nonces $(q, r) \leftarrow \mathcal{Z}_p^*$. Then, it computes the elliptic curve points $Q = qP$ and $U_i = r \cdot qP$. Then, it generates the encrypted witness $\hat{C}_{j_i}$ as $\hat{C}_{j_i} = PKE.Enc(pk_O, r\hat{P})$, and generates the signed witness σ_{j_i} as $\sigma_{j_i} = DSig.Sgn(sk_i, \hat{C}_{j_i})$. Finally, UA_i generates the proof π_{j_i} by applying the proof algorithm of the NIZKP scheme. To this aim, it extracts a nonce $\chi \leftarrow Z_p$, computes $A = \chi Q$, $B = \chi\hat{P}$, $c = H(Q, U, \hat{P}, C_{j_i}, A, B)$, and $s = \chi - c \cdot r$, and sets $\pi_{j_i} = (c, s)$.
  The values $M_j = ((U_i, Q), \hat{C}_{j_i}, \sigma_{j_i}, \pi_{j_i})$ and $st = (gpk, q, U_i, Q)$ are then delivered to the authority T.
- At reception of the message, the USS first stores in the *Registration Table* the entry $reg_i = (\hat{C}_{j_i}, \sigma_{j_i})$.
- Then, the USS T generates the signature σ' by applying the structure-preserving signatures on equivalence classes scheme. Specifically, on input the plaintext $m = ((U_i, Q))$ and the secret key $sk_R = (x_i)_{i\in|l|}$, the USS extracts $y \leftarrow Z_p$ and generates the signature $\sigma = (Z, Y, \hat{Y})$, being $Z = y\sum_{i\in|l|} x_i M_i$, $Y = \frac{1}{y}P$, and $\hat{Y} = \frac{1}{y}\hat{P}$.
- Then, the USS T generates a new anonymous identifier for UA_i, by applying the algorithm $S.ChgRep$. Specifically, on input the message $m = (U_i, Q)$, the signature $\sigma' = (Z, Y, \hat{Y})$, the scalar q^{-1}, and the public key pk_R, the USS picks $\psi, \mu \leftarrow \mathcal{Z}_p$ and outputs $\sigma = (\psi\mu Z, \frac{1}{\psi}Y, \frac{1}{\psi}\hat{Y})$. The USS then constructs group secret key for UA_i as $gsk_i = ((R = rP, P), \sigma)$.
- The signature σ' and the group secret key gsk_i are then delivered to UA_i.
- Finally, UA_i verifies all the received signatures. First, it verifies the NIZKP signature π_{j_i}. To this aim, it first parses $\pi_{j_i} = (c, s)$, and then computes $\hat{A} = (sQ + cU)$ and $\hat{B} = (sP + cC_{j_i})$. Then, it checks if $c \stackrel{?}{=} H(Q, U, \hat{P}, C_{j_i}, \hat{A}, \hat{B})$. If the check is verified, it proceeds further.
  Second, it verifies the digital signature σ_{j_i}. To this aim, it checks that $DSig.Vrf(pk_i, \hat{C}_{j_i}, \sigma_{j_i}) \stackrel{?}{=} 1$.
  Finally, it verifies the signature of the structure-preserving signature on equivalence classes scheme. To this aim, on input $m = (m_i)_{i\in l} = (U_i, Q)$, the signature $\sigma' = (Z, Y, \hat{Y})$, and the public key pk_R, it checks that the following Eq. 9 holds.

$$\prod_{i\in|l|} e(m_i, \hat{X}_i) = e(Z, \hat{Y})$$
$$e(Y, \hat{P}) = e(P, \hat{Y}), \tag{9}$$

  being $\hat{X}_i \in \mathcal{G}_2^*$, and $e(\cdot, \cdot)$ the bilinear pairing operation.

If all the verification steps described above end successfully, UA_i stores the pair (gsk_i, gpk) as its own private and public group keys, to be used for anonymously signing messages when deployed.

Online Phase. This phase is executed at runtime, both on any generic observer rcv and operational UA UA_i, which would like to send RID messages, while staying anonymous. It involves the following operations.

- Assume that at the time t_k the UA UA_i is required to deliver a *RemoteID* message. We denote with ID_g the UA group ID, $pos_k = (lat_k, lon_k, alt_k)$ the latitude, longitude, and altitude coordinates of the location occupied by UA_i, as obtained through the integrated GPS module, $pos_{u,k} = (lat_{u,k}, lon_{u,k}, alt_{u,k})$ the latitude, longitude, and altitude coordinates of the location occupied by UA Ground Station (or controller) at the time t_k, $v_k = (v_{x,k}, v_{y,k}, v_{z,k})$ as the 3-D vector of the speed in m/s of the UA as obtained through integrated modules at the time t_k, and em_k the emergency status value of the UA. We denote $m_k = (ID_g, pos_k, v_k, pos_{u,k}, t_k, em_k)$. The following operations depend on the selected operational mode, i.e., $DS - CCA2 - A^2RID$ or $DS - CPA - A^2RID$.
- $\boldsymbol{DS - CCA2 - A^2RID}$. UA_i extracts a nonce $\rho_k \leftarrow Z_p$, and computes two signatures for the time-slot

k, namely, $\sigma_{1,k}$ and $\sigma_{2,k}$.
$\sigma_{1,k}$ is obtained by applying the randomization algorithm of the structure-preserving signatures on equivalence classes scheme on the group secret key gsk_i, i.e., it sets $\sigma_k = \sigma = (Z, Y, \hat{Y})$, extracts $\phi_k \leftarrow Z_p$, and computes $\sigma_{1,k} = (\phi_k \mu Z, \frac{1}{\phi_k} Y, \frac{1}{\phi_k} \hat{Y}) = ((R', P'), \sigma')$.
$\sigma_{2,k}$ is obtained by signing the message m_k through the signature algorithm of the SoK scheme, Specifically, UA_i extracts $u_k, v_k, \eta_k \leftarrow Z_p$, computes $(\hat{C_1}, \hat{C_2}) = (u_k \hat{Y}, \rho \hat{P} + u \hat{P})$, $N = v_k P$, $\hat{M_1} = \eta_k \hat{Y}$,$\hat{M_2} = (v_k + \eta_k)\hat{P}$, $c = H(N||\hat{M_1}||\hat{M_2}||\sigma_{1,k}||m)$, $z_1 = v_k + c \cdot \rho_k$, $z_2 = \eta + c \cdot u_k$, and finally, it sets $\sigma_{2,k} = (\hat{C_1}, \hat{C_2}, c, z_1, z_2)$.
Finally, UA_i delivers the RemoteID message $(m_k, mode = 1, \sigma_{\mathbf{k}} = (\sigma_{1,k}, \sigma_{2,k}))$.
- $\boldsymbol{DS - CPA - A^2RID}$. UA_i extracts two nonces $\rho_k, v_k \leftarrow Z_p$, and computes two signatures for the time-slot k, namely, $\sigma_{1,k}$ and $\sigma_{2,k}$.
$\sigma_{1,k}$ is obtained as for the $DS - CCA2 - A^2RID$ scheme above. $\sigma_{2,k}$ is the pair (c_k, z_k), where $c_k = H(N||\sigma_{1,k})||m_k)$, with $N = v_k P$, and $z_k = v_k + c_k \cdot \rho_k$. Finally, UA_i delivers the RemoteID message $(m_k, mode = 2, \sigma_{\mathbf{k}} = (\sigma_{1,k}, \sigma_{2,k}))$.
- Assume the generic observer rcv receives the message $(m_k, mode, \sigma_{\mathbf{k}}$, and would like to verify its authenticity.
- If the group public key corresponding to the group identifier ID_g is not stored locally, rcv retrieves it online from the USS T. If gpk has already been retrieved, this step is not necessary.
- If $mode = 1$, the message was broadcasted in CCA2 mode. Thus, rcv verifies the signature components $(\sigma_{1,k}, \sigma_{2,k})$ by applying individually the verification algorithms of the structure-preserving signatures on equivalence classes and SoK schemes on $\sigma_{1,k}$ and $\sigma_{2,k}$, respectively.
First, rcv checks the authenticity of $\sigma_{1,k}$. To this aim, it sets $\sigma_{1.k} = (R', P'), \sigma = (Z, Y, \hat{Y})$ and $pk_R = (\hat{X}_i)_{i \in |l|}$, and checks that Eq. 9 reported above holds. In case it does not hold, the RemoteID message is not verified, and it is reported as anomalous.
If Eq. 9 holds, then rcv sets $N' = z_1 P - cP'$, $M_1 = z_2 \cdot \hat{Y} - c \cdot \hat{C_1}$, and $\hat{M_2} = (z_1 + z_2) \cdot \hat{P} - c \cdot \hat{C_2}$, and checks if $c \stackrel{?}{=} H(N'||\hat{M_1}||\hat{M_2}||\sigma_{1,k})||m_k)$. If it holds, the message m is authenticated. Otherwise, it is discarded as not authentic.
- If $mode = 2$, the message was broadcasted in CPA mode. As previously-described, rcv verifies the signature components $(\sigma_{1,k}, \sigma_{2,k})$ individually. Thus, rcv checks the authenticity of $\sigma_{1,k}$, using the same procedure described above for the $DS - CCA2 - A^2RID$. If Eq. 9 does not hold, it sets $N' = zP - cP'$, and checks if $c \stackrel{?}{=} H(N'||\sigma_{1,k}||m_k)$. If the equality holds, the message is authenticated, otherwise, the message is not authenticated and, it is discarded.

Disclosure Phase. This phase is triggered by the generic observer rcv and executed on the USS, whenever an invasion of the no-fly area of rcv by an anonymous UA is detected. It involves the following operations.

- Assume rcv realizes that the received message $(m, mode, \sigma_{\mathbf{k}})$, verified as authentic, determines an invasion of the monitored no-fly zone. Then, rcv forwards this message to the USS T.
- On message reception, T finds the entry $reg_i \leftarrow (\hat{C}_{j_i}, \sigma_{j_i})$ such that the following Eq. 10 applies.

$$\hat{R} = PKE.Dec(ok, \hat{C}_{j,i})$$
$$e(\sigma_{1,k}[1][1], \hat{P}) = e(\sigma_{1,k}[1][2], \hat{R}). \quad (10)$$

- T finally returns an ack message to the observer, to confirm the correct execution of identification operations. Recall that T does not reveal the identity of the invading UA, as rcv is not the entity in charge of enforcing sanctions, charges, or bans.

VI. Security Considerations

In this section, we discuss the security features offered by the protocols in A^2RID. Sec. VI-A provides the big picture of the security services offered by the A^2RID protocol suite, while Sec. VI-B reports the details of their formal security verification conducted using ProVerif.

A. Security Services

Overall, the protocols in A^2RID provide the following security properties.

UA Anonymity. One of the most important security properties provided by the protocols in the A^2RID protocol suite is the sender's full anonymity in the Random Oracle Model under the DDH assumption in $\mathbb{G}_T$. In the $CS - A^2RID$ scheme, even assuming that the adversary knows the secret signing keys gsk_i of all group members, he/she cannot distinguish if the signature $\sigma = (S_\rho, S_\mu, S_\nu, T_1, T_2, T_3, T_4, T_5, T_6, T_7)$ has been generated by two different signers. Indeed, the corresponding signature features the statistical Zero Knowledge property, i.e., it does not reveal any information about the secret signing key. Thus it is computationally hard (under the DDH assumption) for the adversary to distinguish which group secret key has been used to compute the elements T_1, T_2, T_3, T_4. Moreover, since the values T_5, T_6, T_7 are blinded, the adversary cannot extract any information about the membership certificate of the user. Similarly, in the two variants of the $DS - A^2RID$ scheme, the two signature components $\sigma 1, k$ and $\sigma 2, k$ are obtained by using the group public key and a pseudo-random transformation of the group secret key gsk_i through the structure-preserving signatures on equivalence classes algorithm. By definition, two elements of the same equivalent class are unlinkable, i.e., it is not possible to attribute any of these elements to a single (or multiple) entity(ies), thus providing sender anonymity. In Sec. VI-B, we formally verify the anonymity of the sender for all the above protocols using ProVerif.

UA Message Authenticity. Another important security feature offered by A^2RID is the capability of generic observers to authenticate directly the received anonymous broadcast messages, without relying on any TTPs. In the $CS - A^2RID$

scheme, the observer uses the group public key gpk to verify that a legitimate signer, in possession of the group secret key gsk_i, signed the message m, thus proving the authenticity of the message. Similarly, in the two variants of the $DS-A^2RID$ scheme, the observers can use the group public key gpk to ensure that a legitimate correspondent group secret key gsk_i has been used to sign the message, thus guaranteeing message authenticity. This is a significant security feature, especially for deployability purposes, as it allows observers not to be Internet-connected and to detect misbehaving UAs immediately. We also formally verify the authenticity of received messages for all the above protocols using ProVerif in Sec. VI-B.

Protection against Replay Attacks. As A^2RID messages are broadcast, replay attacks cannot be avoided, but only detected and mitigated through careful protocol integration and deployment. In A^2RID protocols, protection against replay attacks is provided thanks to the timestamp t_s included in regular standard-compliant RID messages. At message reception time, the observer decodes the timestamp and, if the difference between the information in the message and the actual reception time is higher than a threshold τ, it discards the message. At the same time, thanks to the message authenticity feature discussed above, any malicious modification of the timestamp in the message leads to the failure of the verification of the message signature(s), leading to message discarding.

Partial Protection against UA tracking. Although A^2RID specifically addresses sender anonymity, in our context, it also provides partial protection against passive tracking of the UA. Indeed, in the standard setup where the eavesdropper does not know how many UAs are flying in the area corresponding to its reception range, it is not possible to associate two (or more) messages to the same UA with 100% assurance.

However, as highlighted also by the authors in [41] and [42], working on disclosed locations, an eavesdropper with capabilities to run data analysis might be able to infer the track of a given UA with remarkable accuracy, resorting to state-of-the-art Artificial Intelligence (AI) techniques. We acknowledge the likely effectiveness of these solutions for UA tracking and thus, we define our protection against tracking as *partial*. However, we recall that the aim of the A^2RID protocol suite is to provide UAs anonymity and messages authenticity for UAs compliant with the RID specification. Therefore, location privacy is out of scope and can be provided through different solutions, easily integrable in RID messages [28].

B. Formal Verification using ProVerif

We formally verified the security features offered by A^2RID, i.e., UA anonymity and RID message authenticity, through the tool ProVerif [43]. Note that the rigorous security proofs of the CS and DS schemes adopted in our manuscript have been provided by the authors in [40] and [20], respectively. However, the application of such schemes in the drone ecosystem and their combination with additional security properties might, in principle, *logically* affect the security of such schemes. In this context, using automated security protocol verification tools such as ProVerif is the correct way to check if the way such protocols are used does not affect their security properties.

In brief, ProVerif is an automatic cryptographic protocol verifier widely adopted in the literature to formally verify the security properties achieved by cryptographic protocols [44], [45], [46].

Specifically, ProVerif assumes the Dolev-Yao attacker model, i.e., the attacker can read, modify, delete, and forge new packets to be delivered on the communication channel. Assuming the aforementioned conditions, ProVerif checks if the attacker can break the security goals of the protocol, as defined by the user. In case an attack is found, ProVerif also provides a step-by-step description of the attack. Internally, ProVerif uses an abstract representation of the protocols through Horn clauses, and it adopts a logical resolution algorithm on these clauses to prove the security properties of the protocol or to find attacks. ProVerif also translates the security properties of interest into *derivability queries* on the generated Horn clauses, and it finds attacks (if any) working on the reachability of such clauses. Interested readers can find all the details on the internal working logic of ProVerif in the seminal work in [47].

We formally verified A^2RID protocols in ProVerif to prove: (i) the secrecy of the long-term identity of the UA; and, (ii) the authenticity of the message broadcasted by an UA. Therefore, according to the logic of the ProVerif tool, we defined two main events.

1) *acceptUA(id)*, indicating that an UA with long-term identity ID_n is running a protocol in A^2RID.
2) *termAuth(id)*, indicating that the USS finished successfully executing a protocol in A^2RID, verifying that the UA ID_n was the one to generate the message.

Moreover, we proved the message authenticity property by verifying security properties such as *sender authentication* and *impersonation resistance*. As standard mechanisms in ProVerif we checked that *event(acceptUA(id))* cannot be executed after the execution of *event(termAuth(id))*.

Furthermore, we verified the strong secrecy of the long-term identity of UA_i, ensuring that the attacker cannot to distinguish when the secret changes, and that the attacker cannot obtain UA_i from the messages exchanged on the wireless communication channel. The following output messages are provided by ProVerif to identify the fulfillment of the security properties of our interest.

- *event(last_event ()) $\Longrightarrow$ event(previous_event ()) is true*: meaning that the function *last_event* is executed only when another function, namely *previous_event*, is really executed;
- *not attacker(elem[]) is true*: meaning that the attacker is not in possession of the value of *elem*;
- *Non-interference elem[] is true*: meaning that an attacker cannot infer the value of *elem* from the eavesdropped messages.

We anticipate that the interested readers can verify our claims and reproduce our results by downloading the source code of the security verification in ProVerif of both the $CS-A^2RID$

protocol and the $DS - A^2RID$ protocols at [12] and [13], respectively.

***CS − A²RID* Scheme.** We first implemented $CS-A^2RID$ in ProVerif and, we tested the aforementioned two security properties. An excerpt of the output of the ProVerif tool for the case of $CS-A^2RID$ is shown in Figure 2. The tool confirms

Verification Summary - $CS - A^2RID$

```
Query event(termAuth(id)) ==>
event(acceptUA(id)) is true.
Query Querynotattacker(IDA[]) is true.
Non-interference IDA is true.
```

Figure 2. Excerpt of the output provided by the ProVerif tool for verifying the security properties of the $CS - A^2RID$ protocol.

that: (i) message authenticity is verified; (ii) the attacker cannot obtain the long-term identity of the UA, namely, UA_i; and, (iii) the attacker cannot even distinguish when the emitter of a given message changes. Thus, $CS - A^2RID$ achieves the security properties of our interest.

***DS − A²RID* Schemes.** We also implemented both the $DS - A^2RID$ schemes in ProVerif, and we tested also for these protocols the same security properties verified for the $CS - A^2RID$ scheme. We report in Fig. 3 the excerpt of the output of the ProVerif tool for both the $DS-CCA2-A^2RID$ and the $DS - CPA - A^2RID$ protocols.

Verification Summary - $DS - A^2RID$

```
Query event(sig_verified(id)) ==>
send_message(id)) is true.
Query Querynotattacker(secret_gsk[]) is true.
Non-interference secret_gsk is true.
```

Figure 3. Excerpt of the output provided by the ProVerif tool for verifying the security properties of the $DS - CCA2 - A^2RID$ protocol.

We first notice that whenever a message by the UA id is authenticated by an observer (event(sig_verified(id)), that message has been actually delivered by the UA with unique identifier id, thus verifying message authenticity. Such authenticity verification always occurs directly, without relying on trusted third parties. Moreover, we also notice that the group secret key $secret_gsk$ used by the UA to anonymize its messages is never leaked to the adversary (Querynotattacker(secret_gsk[]) is *true*), and that such an adversary cannot even discriminate if the key used to anonymize two or more messages is the same or not (Non-interference $secret_gsk$ is *true*). Thus, the main security properties to be provided by $DS-CCA2-A^2RID$ have been verified. Note that the same properties have also been verified for the $DS-CPA-A^2RID$ scheme. The interested readers can verify our claims and reproduce our results by downloading the source code of the security verification in ProVerif at the link provided in [13].

VII. Performance Assessment

In this section, we present the results of our extensive experimental performance assessment, performed on actual UAs. Sec. VII-A provides the implementation details, while Sec. VII-B presents all the experimental results.

A. Implementation Details

CS Scheme. We implemented a prototype of the $CS - A^2RID$ protocol on the Holybro X-500 commercial drone [48]. The Holybro X-500 features a mission computing unit UP Xtreme i7 8665UE, connected to a *Pixhawk 4 Autopilot*. The *Pixhawk 4* is connected, in turn, to a Holybro M8N GPS module. It also includes a quad-core processor 1.70 GHz Intel Core i7, 64 GB of eMMC memory, and 16,384 MB of RAM, being definitely a high-end UA. As for the Operative System (OS), the drone runs *Ubuntu 20.04 LTS Focal Fossa* [49].

We implemented $CS - A^2RID$ in *C++*, and we integrated it at the IEEE 802.11b MAC-layer. In particular, we integrated A^2RID within a custom IEEE 802.11b MAC-layer by using the network packet sniffing and crafting library libtins 4.0 [50] and the lightweight Micro Air Vehicle Message Marshalling Library MAVSDK 1.4.3 [51], used to acquire the GPS coordinates and drone speed. We also recall that libtins allows full customization of IEEE 802.11 raw MAC-layer messages, such as broadcast data frames encapsulated within IEEE 802.11 standard with a Maximum Transmission Unit (MTU) of 2,312 B.

We report the structure of the IEEE 802.11b MAC-layer raw frame (customized for A^2RID) in Fig. 4, while Tab. II provides additional configuration details. Note that, we customized the Protocol Data Unit (PDU) with dedicated a IEEE 802.11 Message ID (`0xa21d`). Fields such as the Group ID, dependent on the deployment of the scheme, can be enlarged or reduced as necessary.

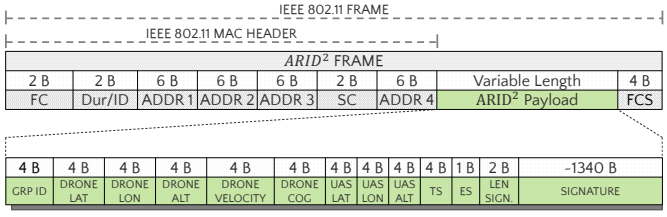

Figure 4. IEEE 802.11b custom Frame and A^2RID Payload Notation.

For the pairing operations, we used the Pairing-Based Cryptography (PBC) Library v0.5.14 [52]. We adopted the symmetric pairings over Type-A (supersingular) curves defined in the PBC library with the default parameters, which offers the highest efficiency. In our implementation, p is a 160-bit Solinas prime, which offers 1,024 bits of discrete-logarithm security. With this Type-A curves setting in PBC, elements of groups $\mathbb{G}$ and $\mathbb{G}_T$ are represented through 512 and 1,024 bits, respectively. We selected the cited curve because it provides a security level of 80 bits, avoiding message fragmentation (not desirable). In particular, starting from a signature of 27,392 B, we encoded such a string in `base58`, reducing

Table II
IEEE 802.11B MAC-LAYER CUSTOM FRAME AND A^2RID PAYLOAD NOTATION.

Name Field	Content/Size	Description
FC	2 B	Frame Control.
Duration/ID	2 B	Payload Length.
Address 1	FF:FF:FF:FF:FF:FF	Receiver MAC Address.
Address 2	00:00:00:00:00:00	Sender MAC Address.
Address 3	6 B	N/A.
Sequence Control	2 B	Sequence Control Field.
Address 4	6 B	N/A.
Payload	1386-1390 B	A^2RID message.
FCS	2 B	Frame Check Sequence.
A^2RID **Payload**		
GRP_{ID}	4 B	Drone Group ID.
D_{LAT}	4 B	Drone Latitude.
D_{LON}	4 B	Drone Longitude.
D_{ALT}	4 B	Drone Altitude.
D_{VEL}	4 B	Drone Speed.
D_{COG}	4 B	Drone Course Over Ground.
U_{LAT}	4 B	Ground Station Latitude.
U_{LON}	4 B	Ground Station Longitude.
U_{ALT}	4 B	Ground Station Altitude.
TS	4 B	Message Timestamp.
ES	1 B	Emergency Code.
L_{SIG}	2 B	Signature Length.
SIG	1343-1347 B	A^2RID Signature.

it to $1,343 \sim 1,347$ B, i.e. 95.08% less.
As for the supporting cryptographic operations, we used the $\mathsf{SHA}-1$ as the hashing function, and a sound cryptographic Pseudo Random Number Generator (PRNG) (/dev/urandom) as a source of random bits. Finally, we implemented the generic observer and the Authority as separated processes on a Raspberry Pi.
We also released at [12] a custom Wireshark dissector plugin for the proposed protocol, allowing observers to identify the customized IEEE 802.11 raw frames of $CS-A^2RID$ immediately.
Our signature implementation on the Holybro X500 requires 149.025 kB of Flash Memory and 3.88 kB of RAM. We also released the source code of A^2RID as open-source, to allow interested researchers and readers to verify our claims and possibly extend A^2RID with additional features [12]. We remark that our implementation leverages popular open-source tools, such as the Ubuntu OS, libtins, MAVSDK, and PBC Cryptography Library, supported by a large variety of commercial UAs.

DS Scheme. We also implemented both the $DS-CPA-A^2RID$ and the $DS-CCA2-A^2RID$ protocols on a reference constrained platform, i.e., the ESPcopter [53]. The ESPcopter is a wirelessly networkable small-size programmable mini-drone, powered by the microcontroller ESP8266 produced by EspressIf [53]. As such, it features an L106 32-bit RISC microprocessor core running at up to 160 MHz, a WiFi 802.11 b/g/n module for wireless communications, 16 MB of Flash, and 160 kB of RAM, being one of the most constrained commercial platforms where to implement, run, and test our protocols. Starting from the ESPcopter code base available at [54], we imported the MIRACL [55] library for cryptography operations on pairing-friendly curves. We remark that, although pairing operations are not executed on the UA during the online phase, the protocols described in Sec. V-C require the usage of pairing-friendly curves, whose implementation cannot be found in other cryptographic libraries for constrained devices. Finally, note that for the implementation of the protocols, we used the curve *BN254*, providing a state-of-the-art security level of 128 bits.
As for the $CS-A^2RID$ scheme, also the source code of the implementation of the $DS-CPA-A^2RID$ and the $DS-CCA2-A^2RID$ protocols on the ESPcopter has been released as open-source at [13].

B. Experimental Results

We report here the performance assessment of the $CS-A^2RID$ protocol on the Holybro X500 and the $DS-A^2RID$ protocols on the ESPcopter. All the time measurements have been obtained through the function `std::chrono::high_resolution_clock::now()`, with an accuracy of 1 ns. Moreover, for all the results, we report the 95% confidence interval as obtained through the related Matlab routine.

$\boldsymbol{CS-A^2RID}$. We report in Fig. 5, Fig. 6, and Fig. 7 the average time and the average energy (due to processing and radio operations) required to execute the cryptographic operations of the $CS-A^2RID$ protocol of the A^2RID protocol suite over $1,000$ tests (with 95% confidence intervals). We notice that the most time-consuming operation is on the *Disclosure Phase*, requiring an average time of 26.498 ms. The *Signature Generation* procedure in the Online Phase requires 17.335 ms on average, while the *Signature Verification* requires 15.477 ms, on average.

As for the energy consumption, to take the measurements, we used a Keysight E36232A DC power supply set, with a voltage of 14.8 V. Specifically, we measured the difference in the electrical current drained by the drone between two

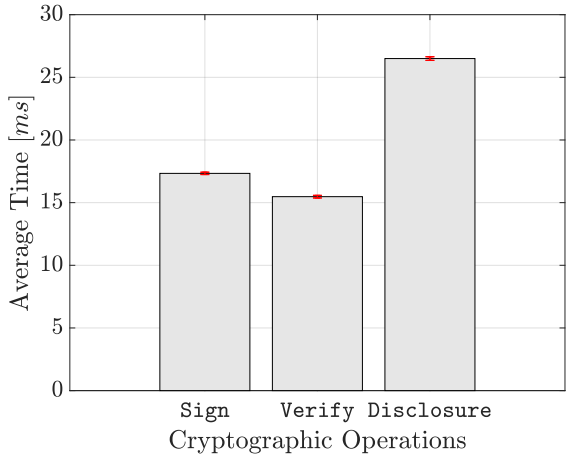

Figure 5. Time required to execute the $CS-A^2RID$ protocol of the A^2RID protocol suite on the HolyBro X500 with mission computer UP Xtreme i7 8665UE.

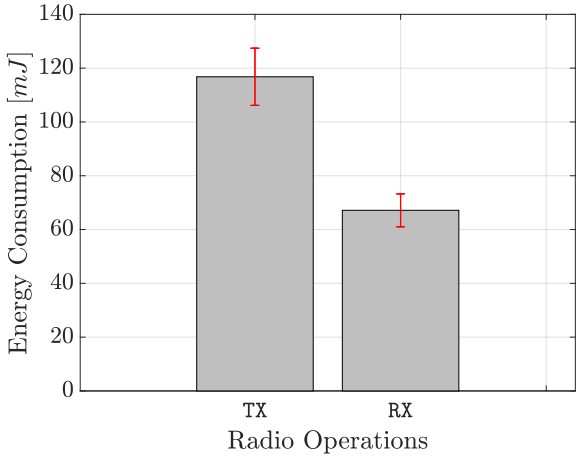

Figure 7. Radio Energy required to transmit and receive A^2RID custom raw IEEE 802.11 frame.

different states: (i) at rest; and, (ii) during the execution of A^2RID. We computed an average difference of ≈ 423 mA in the electric current drained by the drone over $1,000$ runs, and from such a measure, we extracted energy consumption values. As reported in Fig. 6, the most energy-consuming operation is

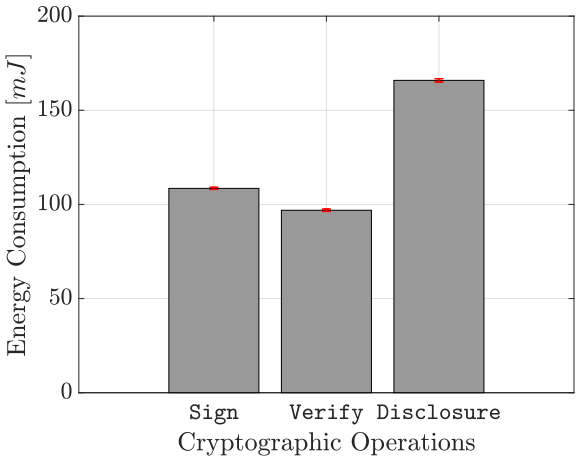

Figure 6. Energy required to execute the $CS - A^2RID$ protocol of the A^2RID protocol suite on the HolyBro X500 with mission computing unit UP Xtreme i7 8665UE.

always the *Disclosure* procedure, requiring an average energy of 165.888 mJ. The *Signature Generation* procedure requires 108.525 mJ on average, while the *Signature Verification* procedure requires 96.895 mJ. To estimate the energy consumption of the radio operations on the Holybro X500, we used an Alfa AWUS036ACH card, capable of running in monitor mode and featuring packet injection capabilities. Such a device works with an input voltage of 5 V, consuming ≈ 600 mA in TX mode and ≈ 345 mA in RX mode, with the IEEE 802.11b protocol [56] at 2.4 GHz. We also assumed that a packet is modulated through the standard Direct Sequence Spread Spectrum (DSSS) modulation using Differential Binary Phase-Shift Keying (DBPSK), a Transmission Rate of 1.0 Mbps on the 22 MHz channel bandwidth, and a Short Guard Interval of 800 ns. In this configuration, the $CS - A^2RID$ protocol in A^2RID consumes only ≈ 116.7 mJ per instance in TX mode (i.e., delivered A^2RID frame), and only ≈ 67.7 mJ in RX mode, considering that a frame is transmitted and received in ≈ 38.9 ms. Considering that the overall capacity of the battery powering the Holybro X500 drone is $266,400$ J ($5,000$ mAh at 14.8 V), the $CS-A^2RID$ protocol consumes on average only $\approx 4.073761 \cdot 10^{-5}\%$ of the battery of the drone for each instance. Overall, such results prove that the energy consumption of the $CS-A^2RID$ protocol in A^2RID is reasonable, and that anonymous RID operations can be executed reliably and in a lightweight fashion, while fully adhering to the RID specification.

DS − A²RID. We also evaluated the time and energy cost of the two variants of $DS - A^2RID$, namely, $DS - CPA - A^2RID$ and $DS - CCA2 - A^2RID$, on the ESPcopter. For the online phase of such protocols, we investigated the performance of two implementation strategies, i.e., without and with pre-computations. The implementation strategy without pre-computations, namely, *plain*, requires the run-time execution of all the modular multiplications and exponentiations detailed in the description of the *Online Phase* in Sec. V-C. Conversely, the implementation strategy with pre-computations requires an offline pre-computation and storage of a number of elliptic curve points correspondent to the ones necessary to run the *Online Phase* of the protocols for a given time. Here, the intuition is that it is possible to trade-off processing time and energy with storage, accelerating the execution of the protocol. We report in Fig. 8 the execution times of all the phases of the $DS - A^2RID$ protocols, including the two implementation strategies mentioned above. While the online phase has been executed on the ESPcopter, the join, verify, and open phases have been executed on a HP ZBook Studio G5 laptop equipped with two Intel(R) Core(TM)

i7-9750H CPUs running at 2.60 GHz.

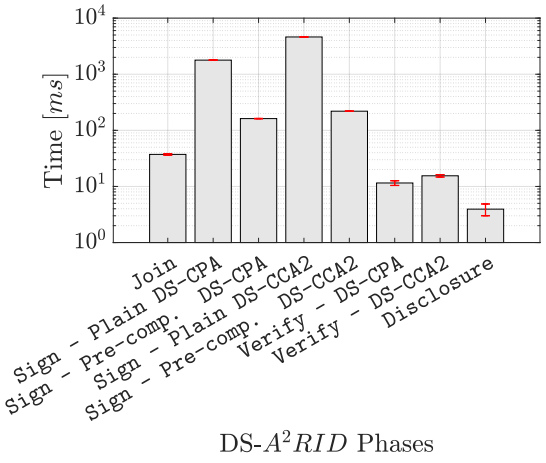

Figure 8. Time required to execute all the phases of the $DS-A^2RID$ protocols of the A^2RID protocol suite, with and without pre-computations, on the ESPcopter and the HP ZBook Studio G5 laptop.

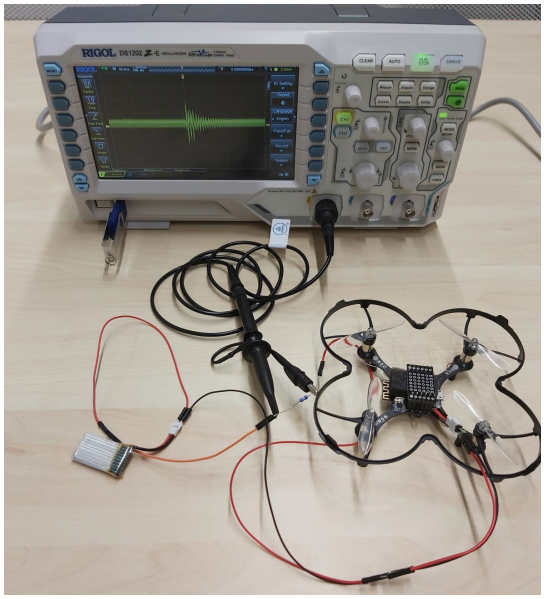

Figure 9. Experimental setup used for acquisition of energy consumption measurements.

Overall, we notice that the execution of the *plain* versions of the $DS-A^2RID$ protocols requires more than 1 s (specifically, ≈ 4.616 seconds for the $DS-CCA2-A^2RID$ and $\approx$ 1.784 seconds for the $DS-CPA-A^2RID$), not allowing to fulfill the time requirement set in the RID specification. Conversely, as modular multiplication and exponentiation operations are the most time-consuming ones, through pre-computations, the ESPcopter can complete the $DS-CCA2-A^2RID$ in 0.22 seconds and the $DS-CPA-A^2RID$ in 0.16 seconds, definitely fulfilling the above-mentioned requirement.

We also measured the energy consumption required by the signing operations on the ESPcopter, both in the *plain* mode and with pre-computations. Fig. 9 shows the testbed used for energy consumption measurements.

We used the oscilloscope RIGOL DS1202Z-E, acquiring samples with a real-time sample rate of up to 1 Gsa/s on two parallel channels [57]. As the oscilloscope measures voltage only, in line with the methodology used in [58], we sampled the voltage drop to the terminals of a shunting resistor of 0.1 Ω, bridging the pins in series with the chipset of the ESP8266 and the battery powering the ESPcopter. We acquired several runs of each protocol, and we post-processed the acquired samples through Matlab. Fig. 10 shows the results of our analysis.

In line with the time measurements, using pre-computations reduces the energy consumption significantly. Looking at the $DS-CPA-A^2RID$ solution, using pre-computed values requires only 9.25 μJ per message signing, on average, down 98.75 % from the 0.74 mJ required for the same scheme, without pre-computations. At the same time, using $DS-CCA2-A^2RID$ with pre-computations requires 12.74 μJ, 37.73 % more than $DS-CPA-A^2RID$ with pre-computations. Considering that the ESPcopter is powered by a tiny Li-Po battery with a capacity of 260 mAh, signing a message with

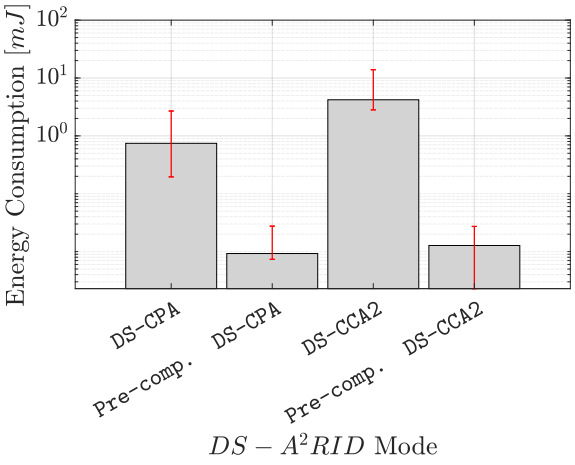

Figure 10. Energy required to generate anonymous RID messages through $DS-A^2RID$ protocols on the ESPcopter, with and without pre-computations.

$DS-CPA-A^2RID$ requires only $9.63 \cdot 10^{-6}$ % of the overall available energy, introducing very limited energy overhead.

At the same time, we highlight that the price to pay for the acceleration of signature operations is on the memory requirements of the protocol, which increase with the amount of information to be stored on the UA and thus, on the duration of the flight. Figure 11 shows the memory space required to pre-compute the values necessary to run both the $DS-CCA2-A^2RID$ and the $DS-CPA-A^2RID$ protocols, with different flight times, assuming to deliver exactly one RID message per second, i.e., the minimum requirement to adhere to the RID rule.

As the $DS-CCA2-A^2RID$ protocol requires more modular multiplications and exponentiation operations at run-

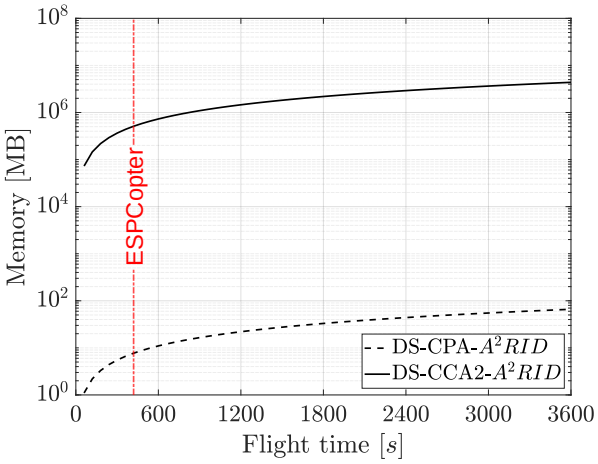

Figure 11. Memory space required to run the $DS-CCA2-A^2RID$ and the $DS-CPA-A^2RID$ protocols, with different flight times, assuming to deliver exactly one RID message per second.

time, it requires more memory than the $DS-CPA-A^2RID$. For a flight time of 420 seconds, $DS-CCA2-A^2RID$ requires 510,049 MB, while $DS-CPA-A^2RID$ requires 7.70 MB only, achieving a consistent reduction.

Overall, these results demonstrate that, using pre-computations, it is possible to achieve anonymous direct authentication and remote identification of commercial UAs, while fulfilling all the requirements imposed by the RID regulation, at the cost of a variable memory increase. In this context, we highlight that this is a reasonable trade-off for commercial drones. Indeed, while processing capabilities require expensive and possibly bulky hardware, with more significant energy consumption and bulkier batteries, increasing memory size typically requires only attaching a bigger SD card, which is less energy-demanding and lightweight to carry around.

C. Comparison

In this section, we compare the performance of the protocols in the A^2RID protocol suite against a benchmarking solution, i.e., the legacy version of the ARID protocol in [11]. Note that, to the best of our knowledge, as summarized in Sec. III, the protocol in [11] is the only current solution providing anonymity for RemoteID-compliant drones, being the straightforward choice for our comparison.

For the comparison, we implemented and ran the *Online Phase* of all the protocols on a common platform, i.e., a Dell XPS 9570 laptop, integrating an Intel(R) Core(TM) i7-8750H CPU running at 2.20 GHz, and equipped with the Ubuntu 22.04 LTS operating system and 32 GB of RAM. We highlight that the hardware selected for comparison is very similar to the ones available on the Holybro X500 drone. At the same time, running all the protocols on a constrained platform such as the ESPcopter is not possible, as such a platform (and similar ones) cannot run processing-intensive approaches.

Figure 12 reports the average time (and related 95% confidence intervals) required to generate anonymous RID messages on the mentioned hardware platform, using the approach in [11], the $CS-A^2RID$ protocol (Sec. V-B), and the $DS-A^2RID$ protocols (Sec. V-C), without and with pre-computations. Note that, for all the protocols excluding $CS-A^2RID$, we considered the configurations providing the same security level, i.e., 256 bits. For $CS-A^2RID$, instead, we considered a security level of 80 bits.

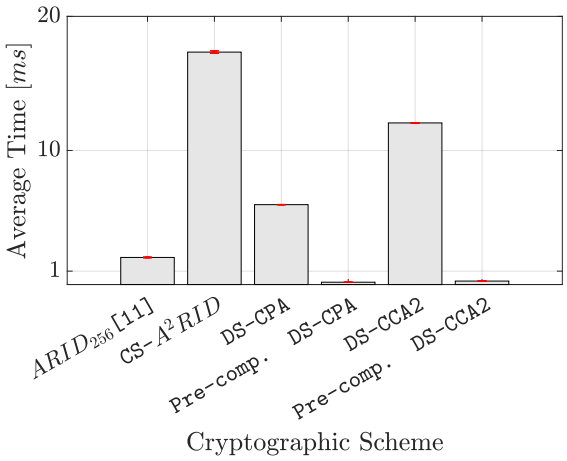

Figure 12. Time required for anonymous signature generation in the protocols of the A^2RID suite and the approach in [11].

We notice that, although being less secure (80 bits instead of 256), the $CS-A^2RID$ protocol is the most time-consuming solution, taking on average 17.34 msec to generate anonymous RemoteID messages. This result is consistent with the cryptography operations required by such a protocol, involving bilinear pairing operations during anonymous signature generation. At the same time, when considering the respective memory-friendly versions, although not relying on bilinear pairing operations at signature generation time, the two $DS-A^2RID$ protocols are more time-consuming than the approach in [11], requiring 12.05 msec ($DS-CCA2-A^2RID$) and 5.95 msec ($DS-CPA-A^2RID$), respectively, against the 2.02 msec of the approach in [11]. This additional time (and energy) overhead is the price to pay to provide direct authentication, instead of the brokered authentication provided by [11]. However, by adopting pre-computations, the $DS-A^2RID$ protocols trade off computation time with memory storage, allowing to significantly reduce the time to generate anonymous RemoteID-compliant messages at the cost of increased memory footprint. In particular, by adopting pre-computations, the $DS-CCA2-A^2RID$ and $DS-CPA-A^2RID$ protocols take on average only 0.26 msec and 0.17 msec. We finally highlight that, compared to the solution in [11], the protocols in the A^2RID suite also allow direct authentication of the messages, with significant advantages in terms of ease of deployability and maintenance.

VIII. Conclusion

In this paper, we proposed A^2RID, a protocol suite for anonymous direct authentication and remote identification of Unmanned Aircrafts. By adopting the protocols in A^2RID, UAs can maintain anonymity while broadcasting standard-compliant Remote Identification messages. At the same time, on reporting by observers, the Unmanned Service Supplier can recover the identity of misbehaving UAs, for follow-up actions. The protocols in A^2RID are characterized by different requirements, being able to run even on very constrained UAs, with minimal processing, storage, and energy load. As a reference, through smart pre-computations, the most lightweight protocol, namely $DS-CPA-A^2RID$, can provide CPA-full anonymity for constrained UAs while requiring only 0.16 seconds for signature generation on the ESPcopter, as well as a minimal energy toll. Thanks to the native integration into the standardized *drip* networking architecture, A^2RID enjoys a straightforward adaptation into actual deployments, contributing to enforce drones' anonymity and invasion accountability. Finally, we highlight that we also released the source code of A^2RID as open-source, to allow interested researchers and industry to verify our claims and possibly extend A^2RID with additional features [12] [13], as well as to check the viability of further research directions. Future works include evaluating the feasibility of A^2RID for improving location privacy of UAs, as well as assessing its feasibility for mitigating UAs tracking. We also plan to evaluate the feasibility of selective online disclosure of UA identity to specific parties.

Acknowledgements

The authors would like to thank the anonymous reviewers, that helped improving the quality of the paper. This work has been partially supported by the INTERSECT project, Grant No. NWA.1162.18.301, funded by Netherlands Organisation for Scientific Research (NWO). This publication was also supported by Technology Innovation Institute - Abu Dhabi, and awards NPRP-S-11-0109-180242 from the QNRF-Qatar National Research Fund, a member of The Qatar Foundation, and NATO Science for Peace and Security Programme - MYP G5828 project "SeaSec: DronNets for Maritime Border and Port Security". The findings reported herein are solely responsibility of the authors.

References

[1] L. Abualigah, A. Diabat, P. Sumari, and A. H. Gandomi, "Applications, Deployments, and Integration of Internet of Drones (IoD): A Review," *IEEE Sensors J.*, vol. 21, no. 22, pp. 25 532–25 546, 2021.

[2] Fortune Business Insights, "Commercial Drones Market Size," https://www.fortunebusinessinsights.com/commercial-drone-market-102171, 2022, (Accessed: 2023-Jan-14).

[3] Statista, "Consumer drone unit shipments worldwide from 2020 to 2030," https://www.statista.com/statistics/1234658/worldwide-consumer-drone-unit-shipments/, 2022, (Accessed: 2023-Jan-14).

[4] FAA, "UAS Remote Identification Overview," 2021, Available Online: https://www.faa.gov/uas/getting_started/remote_id/.

[5] E. Commission, "Detailed rules on unmanned aircrafts," EU, Tech. Rep., May 2021.

[6] K. Belwafi, R. Alkadi, S. A. Alameri, H. A. Hamadi, and A. Shoufan, "Unmanned Aerial Vehicles' Remote Identification: A Tutorial and Survey," *IEEE Access*, vol. 10, pp. 87 577–87 601, 2022.

[7] Drone DJ, "Letter to FAA: Please reconsider your Remote ID proposal," https://tinyurl.com/r7v24eva, 2021, (Accessed: 2023-Jan-14).

[8] AIN Online, "NBAA: Remote ID Drone Rule Raises Privacy Concerns," https://tinyurl.com/3vzzv39d, 2021, (Accessed: 2023-Jan-14).

[9] Drone DJI, "This free app tracks nearby drone flights using Remote ID data," https://dronedj.com/2022/10/04/remote-id-drone-tracking-app/?utm_source=dlvr.it&utm_medium=twitter&s=08, 2022, (Accessed: 2023-Jan-14).

[10] DroneXL, "80,000 Drone IDs Exposed in DJI Aeroscope Data Leak," https://dronexl.co/2022/10/17/80000-drone-ids-dji-aeroscope-data-leak/?utm_source=rss&utm_medium=rss&utm_campaign=80000-drone-ids-dji-aeroscope-data-leak, 2022, (Accessed: 2023-Jan-14).

[11] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "ARID: Anonymous Remote IDentification of Unmanned Aerial Vehicles," in *ACSAC*. ACM, Dec. 2021.

[12] P. Tedeschi, "Open-source code of A^2RID - CS variant," https://github.com/pietrotedeschi/arid2, 2022, (Accessed: 2023-Jan-14).

[13] E. Wisse, "Open-source code of A^2RID - DS variant," https://github.com/EvaWisse/RemoteID_gs, 2022, (Accessed: 2023-Jan-14).

[14] Certicom, Security, "SEC 1: Elliptic curve cryptography," *Certicom Research v2*, vol. 137, 2009.

[15] D. Boneh, "The decision Diffie-Hellman problem," in *Int. algorithmic number theory symposium*. Springer, 1998, pp. 48–63.

[16] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," Cryptology ePrint Archive, Paper 1998/006, 1998.

[17] D. Boneh, et al., "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in *Int. Conf. on the theory and applications of cryptographic techniques*. Springer, 2003, pp. 416–432.

[18] S. Galbraith, et al., "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008.

[19] S. Chatterjee, D. Hankerson, and A. Menezes, "On the efficiency and security of pairing-based protocols in the type 1 and type 4 settings," in *Arithmetic of Finite Fields*, M. A. Hasan and T. Helleseth, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 114–134.

[20] D. Derler and D. Slamanig, "Highly-Efficient Fully-Anonymous Dynamic Group Signatures," in *Proc. of the 2018 on Asia Conf. on Computer and Communications Security*, 2018, pp. 551–565.

[21] G. Fuchsbauer, C. Hanser, and D. Slamanig, "Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials," *Journ. of Cryptology*, vol. 32, no. 2, pp. 498–546, 2019.

[22] F. Hao, "Schnorr Non-interactive Zero-Knowledge Proof," Tech. Rep., 2017.

[23] S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi, "On the Non-malleability of the Fiat-Shamir Transform," in *Int. Conf. on Cryptology in India*. Springer, 2012, pp. 60–79.

[24] A. Svaigen, et al., "MixDrones: A Mix Zones-based Location Privacy Protection Mechanism for the Internet of Drones," in *Proc. of Int. ACM Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2021, pp. 181–188.

[25] A. Svaigen, et al., "BioMixD: A Bio-Inspired and Traffic-Aware Mix Zone Placement Strategy for Location Privacy on the Internet of Drones," *Computer Communications*, 2022.

[26] R. Alkadi and A. Shoufan, "Unmanned Aerial Vehicles Traffic Management Solution Using Crowd-sensing and Blockchain," *IEEE Trans. on Network and Service Management*, pp. 1–1, 2022.

[27] Y. Hashem, E. Zildzic, and A. Gurtov, "Secure Drone Identification with Hyperledger Iroha," in *Proc. of the 11th ACM Symposium on Design and Analysis of Intelligent Veh. Networks and Applications*, 2021, pp. 11–18.

[28] A. Brighente, M. Conti, and S. Sciancalepore, "Hide and Seek–Preserving Location Privacy and Utility in the Remote Identification of Unmanned Aerial Vehicles," in *Proc. of ARES*, 2022.

[29] Thales Group, "ScaleFyt — Remote ID Identification & Tracking," https://www.scaleflyt.com/remoteid, 2022, (Accessed: 2023-Jan-14).

[30] Unifly, "Broadcast Location & Identification Platform (BLIP)," https://unifly.aero/products/blip, 2022, (Accessed: 2023-Jan-14).

[31] Relmatech, "Relmatech)," https://www.relmatech.com/, 2022, (Accessed: 2023-Jan-14).

[32] G. Calandriello, et al., "Efficient and Robust Pseudonymous Authentication in VANET," in *Proc. of ACM Int. Works. on Veh. Ad Hoc Networks*, 2007, p. 19–28.

[33] L. Benarous, B. Kadri, and S. Boudjit, "Alloyed Pseudonym Change Strategy for Location Privacy in VANETs," in *IEEE Annual Consumer Communications Networking Conf. (CCNC)*, 2020, pp. 1–6.
[34] X. Li et al., "PAPU: Pseudonym Swap With Provable Unlinkability Based on Differential Privacy in VANETs," *IEEE Internet of Things J.*, vol. 7, no. 12, 2020.
[35] A. Asari, et al., "A new provable hierarchical anonymous certificateless authentication protocol with aggregate verification in ADS-B systems," *Computer Networks*, vol. 185, p. 107599, 2021.
[36] A. Goudossis and S. K. Katsikas, "Towards a secure automatic identification system (AIS)," *Journ. of Marine Science and Technology*, vol. 24, no. 2, pp. 410–423, 2019.
[37] J. Hall, J. Lee, J. Benin, C. Armstrong, and H. Owen, "IEEE 1609 influenced automatic identification system (AIS)," in *IEEE Veh. Technology Conf. (VTC Spring)*. IEEE, 2015, pp. 1–5.
[38] S. Card, A. Wiethuechter, R. Moskowitz, S. Zhao, A. Gurtov, "Drone Remote Identification Protocol (DRIP) Architecture - draft-ietf-drip-arch-24," Tech. Rep., 2022.
[39] D. Dolev and A. Yao, "On the Security of Public Key Protocols," *IEEE Trans. on information theory*, vol. 29, no. 2, pp. 198–208, 1983.
[40] M. Manulis, N. Fleischhacker, F. Günther, F. Kiefer, and B. Poettrering, "Group signatures: Authentication with privacy," *Bundesamt fur Sicherheit in der Informationstechnik, Bonn, Germany, Tech. Rep*, 2012.
[41] M. Sun, M. Li, and R. Gerdes, "A data trust framework for VANETs enabling false data detection and secure vehicle tracking," in *IEEE Conf. on Communications and Network Security*. IEEE, 2017, pp. 1–9.
[42] S. Khakpour, R. W. Pazzi, and K. El-Khatib, "Using clustering for target tracking in Veh. ad hoc networks," *Veh. communications*, vol. 9, pp. 83–96, 2017.
[43] B. Blanchet, "Automatic Verification of Correspondences for Security Protocols," *Journ. of Comp. Security*, vol. 17, no. 4, pp. 363–434, 2009.
[44] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "PPCA - Privacy-Preserving Collision Avoidance for Autonomous Unmanned Aerial Vehicles," *IEEE Trans. on Dependable and Secure Comput.*, 2022.
[45] J. Wu et al., "Formal Model-Driven Discovery of Bluetooth Protocol Design Vulnerabilities," in *IEEE Symp. on Security and Privacy*, 2022, pp. 879–897.
[46] X. Zhang, H. Zhong, C. Fan, I. Bolodurina, and J. Cui, "CBACS: A Privacy-Preserving and Efficient Cache-Based Access Control Scheme for Software Defined Vehicular Networks," *IEEE Trans. on Information Forensics and Security*, vol. 17, pp. 1930–1945, 2022.
[47] B. Blanchet, "The Security Protocol Verifier ProVerif and its Horn Clause Resolution Algorithm," in *European Joint Conferences on Theory and Practice of Software*. Springer, 2022.
[48] Holybro, "X500 V2 Kit," http://www.holybro.com/product/x500-v2-kit/, 2022, (Accessed: 2023-Jan-14).
[49] Ubuntu, "Focal Fossa 20.04 LTS," https://releases.ubuntu.com/20.04/, 2022, (Accessed: 2022-08-09).
[50] M. Fontanini, "Libtins: packet crafting and sniffing library," 2016, (Accessed: 2023-Jan-14).
[51] D. Foundation, "MAVSDK," https://mavsdk.mavlink.io, 2022, (Accessed: 2023-Jan-14).
[52] B. Lynn, "PBC Library 0.5.14," 2013, (Accessed: 2023-Jan-14).
[53] ESPcopter, "ESPcopter: Programmable MiniDrone," https://espcopter.com/, 2022, (Accessed: 2023-Jan-14).
[54] ESPcopter, "ESPcopter: Code Releases," https://espcopter.com/code-release/, 2022, (Accessed: 2023-Jan-14).
[55] MIRACL, "MIRACL Core," https://github.com/miracl/core, 2022, (Accessed: 2023-Jan-14).
[56] S. Keranidis, et al., "Experimental Evaluation and Comparative Study on Energy Efficiency of the Evolving IEEE 802.11 Standards," in *Proc. of Int. Conf. on Future Energy Systems*, 2014, pp. 109–119.
[57] RIGOL, "Digital Oscilloscope DS1202Z-E," hhttps://rigolshop.eu/digital-oscilloscope-ds1202z-e.html, 2021, (Accessed: 2023-Jan-02).
[58] S. Sciancalepore, G. Oligeri, and R. Di Pietro, "Strength of Crowd (SOC)—Defeating a Reactive Jammer in IoT with Decoy Messages," *Sensors*, vol. 18, no. 10, p. 3492, 2018.