IoD-Crypt: A Lightweight Cryptographic Framework for Internet of Drones

Muslum Ozgur Ozmen, Rouzbeh Behnia, Attila A. Yavuz, Member, IEEE

Abstract—Internet of Drones (IoD) is expected to play a central role in many civilian and military applications, that require sensitive and mission-critical information to be processed. It is therefore vital to ensure the security and privacy of IoD. However, unlike traditional networks, IoD has a broader attack surface and is highly energy-constrained, which hinder the direct adoption of standard cryptographic protocols for IoD.

We propose an energy-efficient cryptographic framework (namely IoD-Crypt), which can potentially meet the requirements of battery-limited IoD. Specifically, IoD-Crypt utilizes special precomputation techniques and self-certified primitives to gain significant computation and communication efficiency over the standard public key cryptography (PKC) suites. Our integrations and optimizations are broadly applicable to key exchange, digital signature and public key encryption schemes that encompass generic applications of PKC in IoD. We prove that IoD-Crypt is secure in the random oracle model. We fully implemented IoD-Crypt on two common drone processors, namely 8-bit AVR and 32-bit ARM, and conducted an in-depth energy analysis. Our experiments (on both platforms) showed that IoD-Crypt offers up to 48% less energy consumption compared to standard techniques. We have open-sourced our implementations for wide adoption and public testing purposes.

Index Terms—internet of drones, drone security, network security, public key cryptography, lightweight cryptography

I. INTRODUCTION

Internet of Drones (IoD), as an emerging mobile Internet of Things (IoT) system, is a layered network control architecture that is designed to control and coordinate unmanned aerial vehicles (UAVs or drones) [1]. IoD has many applications including, but not limited to, military operations, package delivery, traffic control, environmental monitoring and disaster recovery [1], [2], [3], [4], [5], [6]. Due to the sensitive and strategic information involved in IoD, it is essential to guarantee their security and privacy. However, due to the low computational power, the energy and bandwidth limitations, this becomes a highly challenging task [3].

While there have been extensive studies on the optimization of security mechanisms on IoT, their direct adoption for IoD might not yield feasible and secure solutions. This has been further emphasized in a recent study that investigated the attacks (and their potential countermeasures) on IoT applications [7]. In [7], the authors point out that the majority of the attack vectors could be mitigated if the existing security mechanisms and standards were properly implemented considering the requirements and characteristics of such systems. Therefore, there is a critical need to harness the state-of-the-art security mechanisms and algorithmic optimizations to offer a viable cryptographic solution for IoD, with minimal impact to the battery lives of resource-constrained drones.

We outline the security vulnerabilities and the state-of-the-art solutions that are considered for aerial drones as follows.

A. Overview of Existing Cryptographic Approaches

- **Security Vulnerabilities and Solutions:** In [8], Pleban et al. showed a successful hack and hijack of an AR.Drone 2.0 (equipped with a 32-bit ARM processor) resulted from the lack of cryptographically secure communication channels. Son et al. [9] pointed out that most of the commodity (civilian) drone telemetry systems do not use cryptography to secure the communication. Thus, they proposed a fingerprinting method that may provide some authentication for drones.

There has been a lot of work on the identification of the security vulnerabilities of the current drone configurations [10], [11], [12]. For instance, Shin et al. [10] showed how to extract frequency hopping sequence of FHSS-type drone controllers using a software defined radio. Son et al. [11] demonstrated attacks that can cause drones to crash by exploiting the resonance frequency of MEMS gyroscopes. Habibi et al. [12] demonstrated stealth attacks (with potential defense strategies combining software and hardware) that alter the sensors’ data and even change the direction of the drone. Another vulnerability outlined in [13] targets the MAVLink protocol and results in disabling the ongoing mission of the attacked drone. Garg et al. [14] proposed a framework based on probabilistic data structures that considers UAVs as intermediate aerial nodes to facilitate real-time analysis and cyber-threat detection mechanisms.

Recently, Lin et al. [3] identified the differences of drone networks from traditional ones and the potential challenges towards securing drones. Their study showed that the design and deployment of specifically tailored lightweight cryptographic protocols are essential for energy-constrained drones.

- **Cryptographic Techniques:** The earlier studies [15], [16] suggest the adoption of conventional cryptographic techniques such as RSA and AES on FPGAs to provide efficient solutions. Symmetric ciphers with white-box cryptography [17] were also considered in drone settings to target the mitigations of drone capturing attacks. Recently, certificateless cryptographic protocols were proposed for drone-based smart city applications [18]. These protocols aim to minimize the certification
overhead (e.g., transmission) that might be costly for drones. However, they might still be computationally expensive for energy-constrained drones since they include multiple elliptic curve (EC) scalar multiplications [18]. Cheon et al. [19] proposed a linearly homomorphic authenticated encryption scheme to secure drone systems, but this scheme might also be too costly for resource-constrained drones, due to heavy homomorphic computations.

There is a significant need for an open-source, energy-efficient, and comprehensive cryptographic framework which can vastly enhance the state-of-the-art crypto schemes.

B. Our Contributions

In this paper, we created a cryptographic framework that we refer to as IoD-Crypt, which can meet some of the stringent energy and bandwidth needs of IoD applications.

Below, we give some desirable features of IoD-Crypt.

- **A Comprehensive Energy-Efficient Cryptographic Framework:** IoD-Crypt exploits the synergies among various cryptographic primitives such as efficient elliptic curves and constant storage precomputation techniques while avoiding the costly certification overheads by employing a self-certified cryptographic system. We apply our optimizations to a wide-variety of standard suites such as a key exchange, a digital signature, and a public key encryption. Our theoretical and experimental analysis, inline with the requirements of IoD outlined in [1], [3], confirms that IoD-Crypt can meet the needs of IoD, specifically, in drone-to-drone and drone-to-infrastructure scenarios.

- **Improved Side-Channel Resiliency:** Some of the optimization techniques we employed in our framework aim to speed up EC scalar multiplication by converting it to EC additions [20], [21]. For instance, the digital signature (BPV-SC-FourQ-Schnorr) and public key encryption (DBPv-SC-FourQ-ECIES) schemes that are proposed as a part of the IoD-Crypt framework, do not require any EC scalar multiplication in their signing and encryption, respectively. Therefore, they achieve improved side-channel resiliency as compared to the base schemes by avoiding the attacks that target EC scalar multiplication [22], [23].

- **Cryptographic Energy Usage Analysis on Two Common Platforms:** We identify two microprocessors, namely, 8-bit AVR [12], [24], [25] and 32-bit ARM [26], [27], from different families that are commonly used in small aerial drones (e.g., Crazyflie [26]). We implemented our framework on both of these processors and assessed the energy consumption. Moreover, we tested the standard techniques and provided an in-depth comparison in §VI. Our results showed that IoD-Crypt offers up to $13 \times$ and $48 \times$ lower energy consumption on 8-bit AVR and 32-bit ARM, respectively.

- **Broad Applicability to Various Cryptographic Constructs:** We identify various other (Elliptic Curve) Discrete Logarithm Problem (DLP) based schemes that can benefit from our proposed methods (see §IV-A). We discuss how IoD-Crypt can enhance the performance of such schemes.

- **Open-Source Cryptographic Framework for Wide Adoption:** We open-sourced our optimized framework for wide adoption and testing purposes at the link below.

[https://github.com/ozgurozmen/IoD-Crypt](https://github.com/ozgurozmen/IoD-Crypt)

- **Potential Applications to IoTs:** The techniques used in IoD-Crypt can be useful for other IoT applications that require energy efficiency. Specifically, any IoT application relying on energy-limited devices (e.g., medical implantables, wireless sensors in smart cities/homes) can benefit from IoD-Crypt. Moreover, 8-bit AVR and 32-bit ARM processors, on which we tested IoD-Crypt, are not only common in drones but also in various other IoT applications [28], [29].

**Differences between this article and its preliminary versions in [2], [21]:** In this article, we harness the proposed techniques in two preliminary works to propose an integrated framework, IoD-Crypt. We provide improved implementations on drone microcontrollers that includes algorithmic optimizations. (i) We improve our framework, Dronecrypt, proposed in [2] to further enhance its efficiency and energy consumption, with the techniques in [21]. (ii) In Dronecrypt, only 32-bit ARM processor was considered and in [21], only 8-bit AVR processor was considered. Whereas, in this version,
we implement our improved framework on both 8-bit AVR drone processor [12], [24], [25], and 32-bit ARM processor (that Crazyflie 2.0 is equipped with [26], [27]). We evaluate our improved framework in terms of efficiency, energy consumption, and storage overhead. (iii) We provide the security proof of self-certified keys in [30] for the first time in this paper, that were proposed/adopted without a security proof in [21], [30]. (iv) We provide a formal security analysis of IoD-CryptHy capturing the security properties of its underlying cryptographic primitives. (v) We propose new parameter sets for our optimizations that offers higher security. (vi) We discuss how IoD-Crypt meets the well-established requirements of IoD networks presented in [1], [3] and elaborate on the broader impacts of our optimizations and proposed framework, specifically, we highlight some other EC-based schemes that can benefit from our optimization.

II. BUILDING BLOCKS

We first outline our notation in Table I.

| Notation | Description |
|----------|-------------|
| $P_p$    | Finite Field |
| $G$      | Generator Group Point |
| $N$      | A large prime |
| $(x, U)$ | Private/Public key pair |
| $(d, D)$ | System Wide Private/Public key pair |
| $\Gamma$ | Precomputation Table |
| $D_k$    | IND-CPA Encryption via key $k$ |
| $\times$ | Elliptic Curve Scalar Multiplication (Emul) |
| KDF      | Key Derivation Function |
| $H$      | Cryptographic Hash Function $H: \{(0,1)^* \rightarrow Z_N$ |

TABLE I: Notation followed to describe schemes.

1) **FourQ Curve [31]**: FourQ is a special EC that is defined by the complete twisted Edwards equation $E/F_{p^2}$: $-x^2 + y^2 = 1 + dx^2 y^2$. FourQ is known to be one of the fastest elliptic curves that admits 128-bit security level [31]. Moreover, with extended twisted Edwards coordinates, FourQ offers the fastest EC addition algorithms [31], that is extensively used in our optimizations. All of our schemes are realized on FourQ.

2) **Boyko-Peinado-Venkatesan (BPV) Generator [20]**: BPV generator is a precomputation technique that converts an EC scalar multiplication to multiple EC additions with the cost of a small constant-size table storage. The security of BPV on EC discrete logarithm problem (DLP) setting is well-investigated [33], [34] and depends on the affine hidden subset sum problem. The offline (key generation) and online algorithms of BPV are described in Algorithm 1.

3) **Arazi-Qi (AQ) Self-Certified Key Exchange [30]**: AQ self-certified key exchange minimizes the certificate transmission and verification overhead. In Algorithm 2, we give key generation and shared secret algorithms where the keys are generated and distributed by a key generation center (KGC).

Algorithm 1 BPV Generator [20]

1. Generate BPV parameters $(v, k)$, where $k$ and $v$ are the number of pairs to be precomputed and the number of elements to be randomly selected out of $k$ pairs, respectively, for $2 < v < k$.
2. $r_i \leftarrow Z_N$, $R_i \leftarrow r_i \times G$, $i = 0, \ldots, k - 1$.
3. Set precomputation table $\Gamma = \{R_i, i = 0, \ldots, k - 1\}$.

Algorithm 2 AQ Self-Certified (SC) Scheme [30]

1. Generate a random set $S \subset [0, k - 1]$, where $|S| = v$.
2. $r \leftarrow \sum_{i \in S} r_i \mod N$, $R \leftarrow \sum_{i \in S} R_i$.

A. **System Model**

Self-certified cryptography requires a trusted key generation center to generate and distribute the keys to the entities in the system. Therefore, we assume that there is a trusted authority that can compute and distribute the self-certified keys to the drones. This is a feasible assumption as Federal Aviation Administration (FAA) requires that all drones between 0.6 to 55 lb should be registered with the U.S. Department of Transportation [1]. For instance, as the trusted authority, U.S. Department of Transportation (or FAA) can issue the keys to the drones.

As depicted in Figure 2, our main system components are the drones, and there are also recipients that communicate with the drones. These recipients may include entities such as Zone Service Providers (ZSPs), control stations or servers. For example, ZSPs provide navigation information and coordinate drones in a predetermined area, control stations can be used to
send commands to non-autonomous drones, and servers can be used to store the sensors’ data received from the drones [1], [3], [4]. The communication among these entities and drones occurs over a (insecure) broadcast channel and therefore it should be secured. Note that IoD networks are energy constrained, the components are mobile and the communication range is limited [1], [3].

B. Security Model

We assume that only the KGC is trusted. Following [36], we consider a probabilistic polynomial time (PPT) adversary \( A \) which is able to tap the communications between parties and intercept/modify messages. \( A \) is also able to request the key pairs of any user ID in the system via the \((U_{ID},x_{ID}) \leftarrow \text{KeyQry}(ID)\) oracle. When the key pair of a user is exposed via the KeyQry oracle the user will be marked as corrupted. Since the keys are self-certified, we assume that the adversary is able to verify the validity of the keys via the KeyVer oracle.

Inspired by the work of Canetti and Krawczyk [36], we define the security of a key generation scheme in the following definition. To capture a real-world scenario, given the system-defined the security of a key generation scheme in the following

Definition 1. A self-certified key generation scheme \( SC-KG = (\text{Setup}, \text{Kg}) \) is secure if \( A \) has a negligible advantage in the following experiment \( \text{Exp}^{SC-KG} \) with a challenger \( C \).

\[
\begin{align*}
&- \text{(params)} \leftarrow \text{SC-KG.Setup}(1^n) \\
&- (U_T,x_T) \leftarrow A^{\text{KeyQry}(),\text{KeyVer}()}(\text{params}) \\
&- \text{If } 1 \leftarrow \text{KeyVer}(U_T,x_T), \text{return } 1. \text{ Else return } 0.
\end{align*}
\]

\( A \) wins if user \( T \) was never queried to the KeyQry oracle. The advantage \( \text{Adv}_A^{SC-KG} \) of \( A \) is defined as \( \Pr[\text{Exp}^{SC-KG} = 1] \).

We define the notion of existential unforgeability under chosen message attack (EU-CMA) for a signature scheme \( SGN = (\text{Kg}, \text{Sig}, \text{Ver}) \) through the following definition.

Definition 2. Existentially Unforgeability under Chosen Message Attack (EU-CMA) experiment \( \text{Exp}^{EU-CMA}_{SGN} \) is defined as follows.

\[
\begin{align*}
- (sk, pk) \leftarrow SGN.Kg(1^n) \\
- (m^*, \sigma^*) \leftarrow A^{SGN.\text{Sig}()}(pk) \\
- \text{A wins the above experiment if } 1 \leftarrow SGN.\text{Ver}(m^*, \sigma^*, pk) \text{ and } m^* \text{ was not queried to } SGN.\text{Sig}(). \text{ The EMU-CMA advantage } \text{Adv}^{EU-CMA}_{SGN} \text{ of } A \text{ is defined as } \Pr[\text{Exp}^{EU-CMA}_{SGN} = 1]
\end{align*}
\]

We define the notion of indistinguishability under chosen message attack (IND-CPA) for encryption scheme \( \Sigma = (\text{Kg}, \text{Enc}, \text{Dec}) \) through the following definition.

Definition 3. IND-CPA experiment \( \text{Exp}^{IND-CPA}_{\Sigma} \) for an encryption scheme \( \Sigma = (\text{Kg}, \text{Enc}, \text{Dec}) \) between the adversary \( A \) and a challenger \( C \) is defined as follows.

\[
\begin{align*}
- (sk, pk) \leftarrow \Sigma.\text{Kg}(1^n) \\
- (m_0, m_1) \leftarrow A^{\Sigma.\text{Enc}()}(pk) \\
- C \text{ flips a coin } b \leftarrow \{0, 1\}, \text{ computes } c_b \leftarrow \Sigma.\text{Enc}(m_b, pk) \text{ and returns } c_b \text{ to } A. \\
- b^* \leftarrow A^{\Sigma.\text{Enc}()}(pk) \\
- A \text{ wins the above experiment if } b = b^*. \text{ The IND-CPA advantage } \text{Adv}^{IND-CPA}_{\Sigma} \text{ of } A \text{ is defined as } \Pr[b = b^*] \leq \frac{1}{2} + \epsilon.
\end{align*}
\]

IV. PROPOSED FRAMEWORK

Design Rationale: Our main idea is as follows.

(i) The transmission and verification of certificates introduce a significant communication and computation burden which might be expensive for many IoD applications. Moreover, drones are required to be registered with central authorities (e.g., FAA). We observed that self-certified cryptographic keys as in [30] can be used to bridge this gap and enhance the security and performance of IoD. Specifically, we integrate self-certified keys into Schnorr signatures [37] and Elliptic Curve Integrated Encryption Scheme (ECIES) [38] to achieve efficient authentication and confidentiality tools. We refer to such improved schemes as BPV-SC-FourQ-Schnorr and DBPV-SC-FourQ-ECIES, respectively. Therefore, our approach eliminates the burden of certificates and harnesses a central trusted authority for the initial key distribution that fits IoD in compliance with the FAA requirements.

(ii) We alleviate the computational overhead of our schemes by harnessing special precomputation techniques such as the BPV precomputation technique (see Algorithm 1). We adopt a variation of the BPV technique, the Designated BPV (i.e., DBPV) [21], which permits an efficient use of BPV in ECIES. Finally, we realize all of our improved schemes on FourQ curve [39], which is one of the most computationally efficient ECs. Specifically, the efficiency of EC additions in FourQ complements BPV/DBPV techniques.
(iii) The AQ self-certified system [30] has not shown to be provably secure. We fill this research gap by providing a formal security proof (in the random oracle model) for the AQ self-certified keys. We then analyze the security of our proposed schemes by capturing the integration of self-certified keys and DBPV into Schnorr and ECIES. Our proposed framework consists of a signature scheme and a CPA secure encryption scheme to achieve authentication and confidentiality, respectively.

1) BPV-SC-FourQ-Schnorr: We adopt the BPV technique in the Schnorr signature scheme to speed up the EC scalar multiplication required in Schnorr. Moreover, we use AQ public/private key pair in our Schnorr algorithms to avoid the burden of communicating certificates by achieving self-certification. In PKI-based schemes, before the signature verification, the public key of the signer (i.e., via its certificate) should be verified. Note that \([H(ID_u||U_b) \times U_b + D]\) part in the verification step is equivalent to verifying a certificate without any extra communication cost. Moreover, it can be calculated and stored before the online steps. Our scheme is provided in Algorithm 3.

**Algorithm 3 BPV-SC-FourQ-Schnorr with AQ Keys**

\[(\Gamma, x, U) \leftarrow \text{BPV-SC-FourQ-Schnorr.Kg}(1^n):\]

1. KGC generates its keys \(d \leftarrow \mathbb{Z}_N^*, D \leftarrow d \times G\).
2. \(b \leftarrow \mathbb{Z}_N^*, U \leftarrow b \times G\).
3. \(x \leftarrow [H(ID, U) \times b + d]\).
4. Generate \((v, k)_1\) as in BPV.Offline() Step 1
5. \(r'_i \leftarrow \mathbb{Z}_N^*, R'_i \leftarrow r'_i \times G, i = 0, \ldots, k - 1\).
6. Set precomputation table \(\Gamma = \{r'_i, R'_i\}_{i=0}^{k-1}\).

\[(s, e) \leftarrow \text{BPV-SC-FourQ-Schnorr.Sig}(m, x, \Gamma):\]

1. Generate a random set \(S \subset [0, k - 1]\), where \(|S| = v\).
2. \(r \leftarrow \sum_{i \in S} r'_i \mod N, R \leftarrow \sum_{i \in S} R'_i\).
3. \(e \leftarrow H(m||R), s \leftarrow (r - e \times x) \mod N\).

\[b \leftarrow \text{BPV-SC-FourQ-Schnorr.Ver}(m, \{s, e\}, U, ID):\]

1. \(R' \leftarrow e \times [H(ID||U) \times U + D] + s \times G\).
2. If \(e = H(m||R')\) then set \(b = 1\) as valid, else \(b = 0\).

2) DBPV-SC-FourQ-ECIES: ECIES is a public key encryption scheme also included in TinyECC [40], that offers forward-secure authentication and encryption. In the standard ECIES scheme, two EC scalar multiplications are required, one to generate ephemeral key pair, and one to generate a one-time key. We also adopt the Designated BPV technique that allows to speed up both of these EC scalar multiplications in ECIES by replacing them with EC additions [21].

The BPV precomputation technique is extended for “Designated” elliptic curve points, where not only the EC scalar multiplication over the generator (G) but also over other precomputed curve points are stored [21]. This significantly benefits the public key encryption schemes (e.g., ECIES [38]) where an EC scalar multiplication is required over the receiver’s public key. Although DBPV eliminates EC scalar multiplications, the caveat is the small storage overhead. However, public key encryption schemes are usually suitable for applications where multiple entities (e.g., drones) send messages/reports to a single command center/base station (e.g., ZSPs) using their public key. This allows easy adoption of DBPV to public key encryption schemes (especially in the IoD networks) since the storage overhead introduced would be tolerable, which is confirmed by our experiments (see §VI).

For the ECIES scheme, as in Algorithm 4, we generate the designated precomputation table over \([H(ID||U) \times U + D]\), that is based on the public key of the receiver.

**Algorithm 4 DBPV-SC-FourQ-ECIES with AQ Keys**

\[(\Gamma, x, U) \leftarrow \text{DBPV-SC-FourQ-ECIES.Kg}(1^n):\]

1. KGC generates its keys \(d \leftarrow \mathbb{Z}_N^*, D \leftarrow d \times G\).
2. \(b \leftarrow \mathbb{Z}_N^*, U \leftarrow b \times G\).
3. \(x \leftarrow [H(ID, U) \times b + d]\).
4. Generate \((v, k)_1\) as in BPV.Offline() Step 1
5. \(r'_i \leftarrow \mathbb{Z}_N^*, R'_i \leftarrow r'_i \times G, S'_i \leftarrow r'_i \times [H(ID||U) \times U + D], i = 0, \ldots, k - 1\).
6. Set precomputation table \(\Gamma = \{r'_i, R'_i, S'_i\}_{i=0}^{k-1}\).

\[(c, d, R) \leftarrow \text{DBPV-SC-FourQ-ECIES.Enc}(m, U, ID, \Gamma):\]

1. Generate a random set \(S \subset [0, k - 1]\), where \(|S| = v\).
2. \(r \leftarrow \sum_{i \in S} r'_i \mod N, R \leftarrow \sum_{i \in S} R'_i, S \leftarrow \sum_{i \in S} S'_i\).
3. \((k_{enc}, k_{MAC}) \leftarrow \text{KDF}(S)\).
4. \(c \leftarrow e_{k_{enc}}(m)\).
5. \(d \leftarrow \text{MAC}_{k_{MAC}}(c)\).

\[m \leftarrow \text{DBPV-SC-FourQ-ECIES.Dec}(x, \{c, d, R\}):\]

1. \(S' \leftarrow y \times R\).
2. \((k_{enc}, k_{MAC}) \leftarrow \text{KDF}(S')\).
3. If \(d \neq \text{MAC}_{k_{MAC}}(c)\) return invalid
4. \(m \leftarrow D_{k_{enc}}(c)\)

We also complement our optimized public-key primitives by adopting lightweight symmetric ciphers. We consider Chacha stream cipher and Poly1305 authenticator [41], due to their fast operations and well-studied security [42].

A. Broad Application to IoD and EC-based Schemes

We discuss how IoD-Crypt can meet some of the needs of IoD networks as outlined in [1], [3]. First and foremost, the energy efficiency of IoD-Crypt makes it a suitable choice for IoD. Our experiments showed that IoD-Crypt offers high energy efficiency on small drones that are equipped with 8-bit and 32-bit processors (see §VI). Since IoD networks consist of energy-constrained small drones [3], IoD-Crypt techniques are preferable over the standard techniques. Moreover, since the IoD networks usually use broadcasting for message delivery and may carry highly sensitive information, public key cryptographic tools may be useful due to their scalability, non-repudiation and public verifiability.
Moreover, the proposed techniques in IoD-Crypt framework can benefit various other ECDLP based schemes, that can be useful for securing IoD networks. For instance, in [18], certificateless key encapsulation and encryption techniques are proposed to secure drone communications in smart cities. The EC scalar multiplications in the certificateless hybrid encryption scheme in [18] can be accelerated with DBPV. Similarly, it can also benefit from the adoption of FourQ curve, especially when considered with the DBPV due to its optimized EC addition. In [43], [44], efficient digital signature and signcryption schemes were proposed. Although these schemes include bilinear pairing and therefore, cannot adopt FourQ curve, they can benefit from IoD-Crypt optimizations. In [43], signer needs to perform an EC scalar multiplication that can be accelerated with BVIP, and in [44], at the signcryption algorithm, DBPV can be utilized to speed up a couple of EC scalar multiplications that are necessary to generate an ephemeral key.

B. Limitations and Potential Remedies

High energy and computational efficiency of IoD-Crypt comes with a trade-off due to the adopted optimization methods. First, the precomputation technique used in IoD-Crypt requires storing a constant-size table at the signer/sender. However, we show that with the correct parameter choices, it is even possible to store this table in highly resource-constrained 8-bit AVR processors. Moreover, IoD-Crypt requires a trusted authority (e.g., KGC) to compute and deliver keys to the drones, to remove the certificate transmission and verification authority (e.g., KGC) to compute and deliver keys to the drones, to remove the certificate transmission and verification overhead. As explained in our system model in §III-A, since drones should be registered with authorities (e.g., FAA or U.S. Department of Transportation), we believe this is a plausible assumption for the envisioned IoD applications.

V. Security Analysis

Theorem 1. If an adversary \(A\) wins the experiment in Definition 1 with non-negligible probability after making \(q_k\) key queries and \(q_h\) hash queries, then one can build another algorithm \(C\) that runs \(A\) as a subroutine and can solve the Diffie-Hellman Problem with a non-negligible probability.

Proof. \(C\) runs the AQ.Setup() algorithm and passes \(D\) to \(A\). \(C\) aims to solve the Discrete Logarithm (DL) problem on the input of \(D\). \(C\) keeps tables \(L_h\) and \(L_k\) to keep track of random oracle queries and key queries, respectively.

Setup RO(·) Oracle: \(C\) implements a function H-Sim to handle RO(·) queries to random oracles \(H\). That is, the cryptographic hash functions \(H\) are modeled as random oracles via H-Sim as follows.

\[
i) \ h \leftarrow H-Sim(ID||U_{1D}, L_h): \text{If } R \in L_h \text{ then } H-Sim \text{ returns the corresponding value } h \leftarrow L_m(ID||U_{1D}). \text{ Otherwise, it returns } h \leftarrow 0.5 \cdot Z_N^* \text{ as the answer, and inserts } (ID||U_{1D}, h) \text{ into } L_m.
\]

Queries of \(A\): \(A\) can query RO(·) and KeyQry(·) oracles on any message of its choice up to \(q_k\) and \(q_h\) times, respectively.

1) **Handle RO(·) queries:** \(A\) ’s queries on \(H\) is handled by H-Sim function as described above.

2) **Handle KeyQry(·) queries:** To answer \(A\) ’s key queries on any user of its choice \(ID\), \(C\) inserts \(ID\) into \(L_k\) and continues as follows.

\[
i) \ \text{Pick } x_{1D} \leftarrow Z_N \text{ and } h \leftarrow 0.5 \cdot Z_N^*.
\]

\[
ii) \ \text{Compute } U_{1D} \leftarrow h^{-1} \times (x_{1D} \times G - D).
\]

\[
iii) \ \text{C checks if } L_k(ID||U_{1D}) \neq \bot, \text{ C aborts and outputs 0.}
\]

\[
\text{Else, it sets } L_k(ID||U_{1D}) \leftarrow h.
\]

\[
iv) \ \text{Lastly, } C \text{ outputs } (x_{1D}, U_{1D}) \text{ and sets } L_k(ID) \leftarrow (x_{1D}, U_{1D}).
\]

Output of \(A\): Finally, \(A\) outputs a forgery key pair for any user of its choice \(ID^*\) as \((x_{1D^*}, U_{1D^*})\). By Definition 1, \(A\) wins if the below conditions hold.

\[
i) \ 1 \leftarrow \text{KeyVer}(x_{1D^*}, U_{1D^*}, ID^*)
\]

\[
ii) \ ID^* \notin L_k
\]

\(C\)’s Solution to the DH problem: If \(A\) fails to output a key pair, \(C\) also fails in solving the DL problem, and therefore, \(C\) aborts and returns 0.

Otherwise, if \(A\) outputs a successful forgery \((x_{1D^*}, U_{1D^*})\), using the forking lemma [45], \(C\) can rewind \(A\) to get a second forgery for user \(ID^*\) as \((\tilde{x}_{1D^*}, \tilde{U}_{1D^*})\) where \(x_{1D^*} \neq \tilde{x}_{1D^*}\) and \(U_{1D^*} = \tilde{U}_{1D^*}\) with an overwhelming probability. For the forgeries \((x_{1D^*}, U_{1D^*})\) and \((\tilde{x}_{1D^*}, \tilde{U}_{1D^*})\) on \(ID^*\), based on [45], we know that \(H(ID^*)||U_{1D^*} \neq H(ID^*)||\tilde{U}_{1D^*}\).

Given 1 \( \leftarrow \text{KeyVer}(x_{1D^*}, U_{1D^*}, ID^*)\) and 1 \( \leftarrow \text{KeyVer}(\tilde{x}_{1D^*}, \tilde{U}_{1D^*}, ID^*)\), \(C\) computes \(b \leftarrow H(ID^*||U_{1D^*}) + H(ID^*||\tilde{U}_{1D^*})\) and solve for the discrete logarithm of \(D\) as \(d \leftarrow x_{1D^*} \cdot H(ID^*||U_{1D^*}) \cdot b\). \(\blacksquare\)

Theorem 2. The signature scheme proposed in Algorithm 3 is EU-CMA secure in the sense of Definition 2.

Proof. Based on the results of Theorem 1, we know that the self-certified key used in Algorithm 3 is secure. Moreover, The distribution of keys generated by Algorithm 2 is indistinguishable from uniform random due [37]. As shown in the previous works [20], [33], one can see that the distribution of BPV output is statistically close to the uniform random with the appropriate choices of parameters \((v, k)\). Following this, if the ephemeral randomness \(r\) in BPV-SC-FourQ-Schnorr.Sig Step 2 is obtained from BPV generator, then BPV-SC-FourQ-Schnorr is secure given the hardness of the Affine Hidden Subset Sum problem [33], [46]. Therefore, based on these results, the signature scheme proposed in Algorithm 3 is secure in the sense of Definition 2 under the DL problem. \(\blacksquare\)

Theorem 3. The encryption scheme proposed in Algorithm 4 is IND-CPA secure in the sense of Definition 3.

Proof. The security and properties of the self-certified keys adopted in Algorithm 4 follows from the first part of the proof in Theorem 2. The original BPV, as shown in Algorithm 1 computes \(R\) by multiplying a random \(r\) with the generator of the group \(E(F_p^\times[N])\). However, due to the property of the group \(E(F_p^\times[N])\), in the DBPV method used in Algorithm 4, the public key of the user, \((x \cdot H(ID||U) \cdot b) \times G\), is also a generator of the group. Therefore, proving the randomness of
the BPV output (as also studied in [20], [33]) for the DBPV method is identical to the original BPV method. Based on these results, the encryption scheme proposed in Algorithm 4 is secure in the sense of Definition 3, and the proof can be obtained identical as in [38].

Parameter Choice: The security of our proposed schemes depend on the underlying curve and BPV/DBPV parameter choice of \((v,k)\). As the underlying curve, we select FourQ, that offers 128-bit security level and very fast curve operations such as EC addition [31]. Security of BPV/DBPV depends on the \(v\)-out-of-\(k\) different combinations that could be created with the precomputed table. Moreover, the selection of \(v\) and \(k\) have a trade-off between storage and computation, where increasing \(k\) increases the storage and increasing \(v\) increases the computation. We offer two parameter sets, specifically \(v = 28\), \(k = 256\) and \(v = 18\), \(k = 1024\) that offer 2\(^{128}\) different combinations, and therefore 128-bit security level. From these two, we focus on \(v = 28\), \(k = 256\) since it requires \(4\times\) less storage, which can easily fit highly resource-constrained processors such as 8-bit AVR.

VI. PERFORMANCE ANALYSIS AND COMPARISON

We fully implemented IoD-Crypt on two common drone microprocessors and assessed the performance and energy efficiency of our schemes. We measured the energy consumption with the formula \(E = V \cdot I \cdot t\), following the works [21], [2], [47]. We compare the costs of our optimized techniques with the standard ones, specifically, we implemented ECDH, ECDSA and ECIES protocols on the standardized secp256k1 curve, and adopted AES and HMAC as the symmetric ciphers.

A. Performance on 8-bit AVR Processor

Hardware Configurations: We implemented IoD-Crypt targeting an 8-bit AVR ATmega 2560 processor. ATmega 2560 has a maximum frequency of 16 MHz and it is equipped with 8 KB SRAM and 256 KB flash memory. ATmega 2560 operates at \(V = 5\) V and \(I = 20\) mA while running at 16 MHz (full capacity), that is used to estimate the energy consumption. Since we used IAR Embedded Workbench to develop IoD-Crypt, our codes can be easily tested/used in other 8-bit AVR processors.

Implementation: We implemented the public key primitives of IoD-Crypt using the open-source implementation of FourQ library on microprocessors [47], which provides the basic EC operations such as EC scalar multiplication, EC addition. As for the symmetric primitives, we used Weatherley’s library [48]. We used the cycle-accurate simulator of IAR Embedded Workbench to benchmark our schemes, as in [47]. For the standard public key primitives, we used micro-ecc [49], that offers the lightweight implementations of standard elliptic curves for microprocessors.

The results of our experiments are presented in Table II for the public key primitives and in Table III for the symmetric ones. As shown, IoD-Crypt primitives outperform their counterparts in computation time and energy consumption. On the other hand, they require a storage of 16 KB for the adoption of BPV and 24 KB for DBPV. However, these precomputation tables are stored as a part of the flash memory of ATmega 2560 and they take less than 10% of the memory. Moreover, the standard public key techniques take a few seconds, that might be infeasible for time-critical IoD applications, whereas IoD-Crypt signature generation takes 150 ms and public key encryption takes 270 ms.

B. Performance on 32-bit ARM Processor

Hardware Configurations: We targeted an actual drone for our tests on 32-bit ARM processor, namely Crazyflie 2.0 [26].

| Protocol       | CPU Cycles | CPU Time (s) | Memory (Byte) | Bandwidth (Byte) | Energy Consumption (mJ) | Certificate Overhead |
|----------------|------------|--------------|---------------|------------------|-------------------------|---------------------|
| ECDH           | 25,840,000 | 1.61         | 32            | 32               | 161.52                  | Yes                 |
| Ephemeral ECDH | 54,380,000 | 3.40         | 32            | 32               | 339.86                  | Yes                 |
| ECDSA-Sign     | 28,370,000 | 1.77         | 32            | 64               | 177.30                  | Yes                 |
| ECDSA-Verify   | 28,960,000 | 1.81         | 32            | 64               | 181.01                  | Yes                 |
| ECIES-Encrypt  | 54,450,000 | 3.40         | 32            | 32 + lcl + IMACI | 340.34                  | Yes                 |
| ECIES-Decrypt  | 25,920,000 | 1.62         | 32            | 32 + lcl + IMACI | 161.99                  | Yes                 |

IoD-Crypt

| Protocol       | CPU Cycles | CPU Time (s) | Memory (Byte) | Bandwidth (Byte) | Energy Consumption (mJ) | Certificate Overhead |
|----------------|------------|--------------|---------------|------------------|-------------------------|---------------------|
| AQ             | 6,940,000  | 0.43         | 32            | 32               | 43.38                   | No                  |
| BPV-AQ-Hang    | 9,140,000  | 0.57         | 32            | 32               | 57.14                   | No                  |
| BPV-SC-FourQ-Schnorr.Sig | 2,490,000  | 0.15         | 32            | 64               | 15.57                   | No                  |
| BPV-SC-FourQ-Schnorr.Ver | 8,310,000  | 0.52         | 32            | 64               | 51.94                   | No                  |
| DBPV-SC-FourQ-ECIES.Enc | 4,290,000  | 0.27         | 24608         | 32 + lcl + IMACI | 26.80                   | No                  |
| DBPV-SC-FourQ-ECIES.Dec | 6,980,000  | 0.44         | 32            | 32 + lcl + IMACI | 43.61                   | No                  |

† Memory denotes private key size for sign/encrypt schemes as signer/sender stores it, memory denotes public key size for verify/decrypt schemes as verifier/receiver stores it.

TABLE II: Comparison of Public Key Primitives on 8-bit AVR Processor

| Protocol       | (KB/s) | CPU Cycles | CPU Time (ms) | Energy Cons. (mJ) |
|----------------|--------|------------|--------------|-------------------|
| AES            | 29.339 | 19,500     | 1.22         | 122.31            |
| AES-GCM        | 8.890  | 76,500     | 4.78         | 478.15            |
| HMAC†          | 22.017 | 182,300    | 11.39        | 1138.69           |

IoD-Crypt

| Protocol       | (KB/s) | CPU Cycles | CPU Time (ms) | Energy Cons. (mJ) |
|----------------|--------|------------|--------------|-------------------|
| CHACHA20       | 65.670 | 8,300      | 0.52         | 521.89            |
| CHACHA-POLY    | 23.687 | 35,500     | 2.22         | 222.03            |
| POLY1305       | 37.169 | 21,400     | 1.34         | 133.81            |

† CPU cycles presented here are for a 32-byte message.
‡ SHA256 is used as the standard hash function for HMAC.
Crazyflie 2.0 is equipped with an STM32F4 processor (32-bit ARM Cortex M-4 architecture) that has 192 KB SRAM and 1 MB flash memory, and operates at 168 MHz. At max frequency, the processor requires 3.3 V and 40 mA.

**Implementation:** We used the Microsoft FourQ library to implement IoD-Crypt for 32-bit ARM processor [31], [47]. This library provided the basic EC operations which we used to build the protocols with the optimization techniques. For our counterparts, we used micro-ecc [49] and for symmetric primitives, we used Wolfcrytp library [50].

Table IV and Table V show the benchmarks of IoD-Crypt and the standard techniques on STM32F4 processor. Our experiments showed that IoD-Crypt offers up to 48× less energy consumption. Moreover, IoD-Crypt can support to very high message throughputs due to its high efficiency. More specifically, with IoD-Crypt, an STM32F4 processor can compute 555 signatures and encrypt 448 messages per second. In our experiments, we stored the precomputation tables on the SRAM.

**C. Performance on Commodity Hardware**

Considering the recipient side (e.g., command centers, ZSPs, and servers) that are involved in IoD networks are equipped with high-end processors, we also measured the costs of IoD-Crypt on a commodity hardware. In our experiments, we used an Intel i7 Skylake 2.6 GHz CPU running with Linux 16.04. Our results showed that these processors can support up to a very large number of message throughputs so that they can communicate with a large drone fleet easily. More specifically, signing a message (with BPV-SC-FourQ-Schnorr.Sig) only takes around 10 µs and encrypting a message (with DBPV-SC-FourQ-ECIES.Enc) takes around 15 µs. Verification and decryption algorithms in IoD-Crypt only take 40–45 µs. Therefore, IoD-Crypt supports the secure communication of thousands of messages, and thereby permits a better quality-of-service and response time, wherein recipients (e.g., ZSPs) may handle many drones simultaneously. We open-sourced our codes on the 64-bit processor as well, to complete our framework.

**VII. Conclusion**

In this paper, we propose IoD-Crypt, an efficient public-key based cryptographic framework that is tailored for the stringent energy and network requirements of IoD. IoD-Crypt eliminates (i) the conventional PKI overhead with self-certified keys and (ii) EC scalar multiplications with BPV and DBPV techniques, and adopts FourQ curve that offers very fast curve operations (e.g., EC addition). We prove that our improved schemes in IoD-Crypt are secure, and provide their implementation on two common drone processors, namely 8-bit AVR and 32-bit ARM. Our experiments showed that IoD-Crypt offers up to 13× and 48× less energy consumption compared to standard techniques, on 8-bit AVR and 32-bit ARM, respectively.

**REFERENCES**

[1] M. Gharibi, R. Boutaba, and S. L. Waslander, “Internet of drones,” *IEEE Access*, vol. 4, pp. 1148–1162, 2016.

[2] M. O. Ozmen and A. A. Yavuz, “Dronecrypt - an efficient cryptographic framework for small aerial drones,” in *Milcom 2018 Track 3 - Cyber Security and Trusted Computing (Milcom 2018 Track 3)*, Los Angeles, USA, Oct. 2018.

[3] C. Lin, D. He, N. Kumar, K. R. Choo, A. Vinel, and X. Huang, “Security and privacy for the internet of drones: Challenges and solutions,” *IEEE Communications Magazine*, vol. 56, no. 1, pp. 64–69, Jan 2018.

[4] R. Clarke, “Understanding the drone epidemic,” *Computer Law & Security Review*, vol. 30, no. 3, pp. 230 – 246, 2014.

[5] P. Kumar, S. Garg, A. Singh, S. Batra, N. Kumar, and I. You, “Mvo-based 2-d path planning scheme for providing quality of service in uav environment,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1698–1707, June 2018.
Muslum Ozgur Ozmen received the bachelor’s degree in electrical and electronics engineering from the Bilkent University, Turkey and the M.S. degree in computer science from Oregon State University. He is currently pursuing a PhD degree in computer science with the Department of Computer Science and Engineering, University of South Florida. His research interests include lightweight cryptography for IoT systems (drones and medical devices), digital signatures, privacy enhancing technologies (dynamic symmetric and public key based searchable encryption) and post-quantum cryptography.

Rouzbeh Behnia received an M.S. degree from Multimedia University, Malaysia in 2013. He was a recipient of an FRGS grant during his time as a lecturer in Multimedia University and was awarded a Researcher Award in 2016. In 2016, he received an Outstanding Graduate Scholar award from the School of Electrical Engineering and Computer Science, Oregon State University. He is currently pursuing a Ph.D. in Computer Science at the University of South Florida. His research interests include post-quantum cryptography, privacy enhancing technologies and efficient authentication schemes.

Attila Altay Yavuz (M ’11) is an Assistant Professor in the Department of Computer Science and Engineering, University of South Florida (August 2018). He was an Assistant Professor in the School of Electrical Engineering and Computer Science, Oregon State University (2014-2018). He was a member of the security and privacy research group at the Robert Bosch Research and Technology Center North America (2011-2014). He received his PhD degree in Computer Science from North Carolina State University in August 2011. He received his MS degree in Computer Science from Bogazici University (2006) in Istanbul, Turkey. He is broadly interested in design, analysis and application of cryptographic tools and protocols to enhance the security of computer networks and systems. Attila Altay Yavuz is a recipient of NSF CAREER Award (2017). His research on privacy enhancing technologies (searchable encryption) and intra-vehicular network security are in the process of technology transfer with potential world-wide deployments. He has authored more than 40 research articles in top conferences and journals along with several patents. He is a member of IEEE and ACM.