Journal Paper

BloothAir: A Secure Aerial Relay System Using Bluetooth Connected Autonomous Drones

In press.

Kai Li*
Ning Lu
Jingjing Zheng*
Pei Zhang
Wei Ni
Eduardo Tovar*

*CI[STER Research Centre
CISTER-TR-210202

2021/02/09
BloothAir: A Secure Aerial Relay System Using Bluetooth Connected Autonomous Drones

Kai Li*, Ning Lu, Jingjing Zheng*, Pei Zhang, Wei Ni, Eduardo Tovar*

*CISTER Research Centre
Polytechnic Institute of Porto (ISEP P.Porto)
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
Portugal
Tel.: +351.22.8340509, Fax: +351.22.8321159
E-mail: kai@isep.ipp.pt, zheng@isep.ipp.pt, Wei.Ni@data61.csiro.au, emt@isep.ipp.pt
https://www.cister-labs.pt

Abstract
Thanks to flexible deployment and excellent maneuverability, autonomous drones have been recently considered as an effective means to act as aerial data relays for wireless ground devices with limited or no cellular infrastructure, e.g., smart farming in a remote area. Due to the broadcast nature of wireless channels, data communications between the drones and the ground devices are vulnerable to eavesdropping attacks. This paper develops BloothAir which is a secure multi-hop aerial relay system based on Bluetooth Low Energy (BLE) connected autonomous drones. For encrypting the BLE communications in BloothAir, a channel-based secret key generation is proposed, where received signal strength at the drones and the ground devices is quantized to generate the secret keys. Moreover, a dynamic programming based channel quantization scheme is studied to minimize the secret key bit mismatch rate of the drones and the ground devices by recursively adjusting the quantization intervals. To validate the design of BloothAir, we build a multi-hop aerial relay testbed by using the MX400 drone platform and the Gust radio transceiver which is a new lightweight onboard BLE communicator specially developed for the drone. Extensive real-world experiments demonstrate that the BloothAir system achieves significantly lower secret key bit mismatch rate than the key generation benchmarks which use the static quantization intervals. In addition, the high randomness of the generated secret keys is verified by the standard NIST test, thereby effectively protecting the BLE communications in BloothAir from the eavesdropping attacks.
**BloothAir: A Secure Aerial Relay System Using Bluetooth Connected Autonomous Drones**

KAI LI, Real-Time and Embedded Computing Systems Research Centre (CISTER), Portugal  
NING LU, AirMind LLC., China  
JINGJING ZHENG, CISTER, Portugal  
PEI ZHANG, Electrical and Computer Engineering, University of Michigan, USA  
WEI NI, Data61, Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia  
EDUARDO TOVAR, CISTER, Portugal

Thanks to flexible deployment and excellent maneuverability, autonomous drones have been recently considered as an effective means to act as aerial data relays for wireless ground devices with limited or no cellular infrastructure, e.g., smart farming in a remote area. Due to the broadcast nature of wireless channels, data communications between the drones and the ground devices are vulnerable to eavesdropping attacks. This paper develops BloothAir which is a secure multi-hop aerial relay system based on Bluetooth Low Energy (BLE) connected autonomous drones. For encrypting the BLE communications in BloothAir, a channel-based secret key generation is proposed, where received signal strength at the drones and the ground devices is quantized to generate the secret keys. Moreover, a dynamic programming based channel quantization scheme is studied to minimize the secret key bit mismatch rate of the drones and the ground devices by recursively adjusting the quantization intervals. To validate the design of BloothAir, we build a multi-hop aerial relay testbed by using the MX400 drone platform and the Gust radio transceiver which is a new lightweight onboard BLE communicator specially developed for the drone. Extensive real-world experiments demonstrate that the BloothAir system achieves significantly lower secret key bit mismatch rate than the key generation benchmarks which use the static quantization intervals. In addition, the high randomness of the generated secret keys is verified by the standard NIST test, thereby effectively protecting the BLE communications in BloothAir from the eavesdropping attacks.

**CCS Concepts:**  
- Computer systems organization → Embedded and cyber-physical systems;  
- Security and privacy → Mobile and wireless security.

**Additional Key Words and Phrases:** Autonomous drones, Aerial data relays, Bluetooth Low Energy, Wireless security, Experimental testbed

**ACM Reference Format:**

Authors’ addresses: Kai Li, Real-Time and Embedded Computing Systems Research Centre (CISTER), Porto, Portugal, kai@isep.ipp.pt; Ning Lu, AirMind LLC., Shanghai, China, ning.roland@mindpx.net; Jingjing Zheng, CISTER, Porto, Portugal, zheng@isep.ipp.pt; Pei Zhang, Electrical and Computer Engineering, University of Michigan, Ann Arbor, USA, peizhang@cmu.edu; Wei Ni, Data61, Commonwealth Scientific and Industrial Research Organization (CSIRO), Sydney, Australia, wei.ni@data61.csiro.au; Eduardo Tovar, CISTER, Porto, Portugal, emt@isep.ipp.pt.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Association for Computing Machinery.

XXXX-XXXX/2021/1-ART1 $15.00  
https://doi.org/10.1145/3448254

1 INTRODUCTION

Employing autonomous drones as data relays has the great potential to improve system connectivity and wireless coverage in smart sensing applications, such as environment monitoring in unpopulated regions [1], precision agriculture [2], 3D reconstruction [3], and disaster management [4]. The autonomous drones equipped with wireless communication modules fly along their predetermined trajectories in the area of interest, while relaying data of the ground devices, thanks to excellent mobility and maneuverability of the drones. The short distance line-of-sight (LoS) communication link between the drones and the ground devices can support high-speed data access and significant channel gain, hence saving energy consumption on data communication [5].

The flight of the drones is automated to relay data efficiently from ground devices to a sink by using airborne communications in our considered system. The accurate position information of the drone is crucial to the design and implementation of the trajectory. The information can be estimated based on the measurements of 3-axis accelerometers, gyroscopes, magnetometers, and global positioning system (GPS) [6], which have been widely installed on drones.

Several mobile communication techniques can be considered for wireless connection between the drones and ground devices, e.g., WiFi [7], Zigbee [8], or 4G Long-Term Evolution (LTE) [9]. However, due to limited non-overlapping frequency bands, WiFi or Zigbee can be interfered by other coexisting sensing devices that work in 2.4 GHz. Moreover, the large overhead of WiFi connections results in low spectrum utilization, and is not efficient for the aerial relay system that typically has small payload in data packets and bursty data traffic. Zigbee that uses direct sequence spread spectrum (DSSS) for data transmission can select the channel with the lowest interference. However, the selected channel is fixed for all future data transmissions even new interfering signals in the selected channel are detected. Thus, Zigbee is not applicable to the highly dynamic mobile scenarios such as aerial relay systems with drones. Despite LTE can extend the communication range of the drones, LTE connections require a vast 4G infrastructure coverage, which is hardly provided in unpopulated areas with little to no 4G/5G service.

In this paper, Bluetooth Low Energy (BLE) [10] is considered for data communications in aerial relay systems due to the following preferable features. As a new Frequency Hopping Spread Spectrum (FHSS) standard, BLE continuously adapts carrier frequency for anti-jamming capabilities, where the receiver recombines the received signals in the order determined by the transmitter. BLE has the maximum data rate of 2 Mbps, which enables high-speed data transmission at the drones. In addition, BLE detects and prevents interference at the edges of the 2.4 GHz ISM band and the neighboring LTE band, where a pseudo-random sequence is generated at the rate of 1600 hops per second over 40 channels.

Figure 1(a) illustrates the multi-hop secure aerial relay system, where two BLE-connected autonomous drones (drone A and drone B) are employed to fly over a ground data sender and an intended receiver, establishing a 3-hop aerial data relay link. Specifically, the data sender generates data packets at an application-specific sampling rate and transmits to the data receiver, i.e., the sink node. The data packets generated by the data sender can be heterogeneous, which contains plain text messages, voice messages from microphones, and/or images from cameras. The radio paths between the data sender and the sink node are obstructed, and the radio signal of the data sender is too weak at the sink node.

The trajectories of the drones are meticulously designed so that the data sender can be accessible by the drone along the flight. Moreover, the drone moves within the radio coverage of the ground device and the drone at the preceding hop, as well as the next hop, which maintains the aerial relay system’s connectivity. The drones forward the source data immediately after receiving the data to reduce the packet transmission latency.

Due to the broadcast nature of wireless channels, the BLE communication between the drones and ground devices in the aerial relay system is vulnerable to eavesdropping and replay attacks [11]. By decoding the eavesdropped data, adversaries can track the locations of the data senders of interest and abuse the mobility patterns
of the drones. Therefore, a secret key for data encryption/decryption is crucial to support data confidentiality, integrity, and sender authentication. In turn, it is also critical to the aerial relay system’s safety. As shown in Figure 1(a), at the first hop, a secret key is generated by the data sender to encrypt the source data packets which are sent to drone A. The same secret key has to be generated at drone A to decode the data. Similarly, another two pairs of secret keys are generated at the second hop (i.e., the link between drone A and drone B) and the third hop (i.e., the link between drone B and the data sink), respectively. The sender and the receiver in each hop have to agree upon a unanimous secret key so that the data can be successfully decoded by the receiver. Moreover, the secret key generated at the sender and the receiver cannot be transmitted over the insecure public channel that is observable to the eavesdropper, making it hard to unify the sender’s and receiver’s secret keys. A common method for establishing a secret key is by using public key cryptography. However, public key cryptography requires a fixed key management infrastructure, which is not applicable to real-time data transmission in mobile wireless environments. Although quantum cryptography [12] has started to appear recently, it is prohibitively expensive on the implementation.

Comparing to various physical layer information of radio channel (such as channel phase), received signal strength (RSS) can be measured by most of current off-the-shelf wireless devices without any modification, and thus presents significant cost savings. Generating the secret key with RSS measurements of the radio channel is a promising approach [13, 14], where two legitimate mobile users extract secret bits from the inherently random spatial and temporal variations of the reciprocal wireless channel between them. Moreover, the properties of the channel are unique to the locations of the users. An eavesdropper at different locations measures uncorrelated RSS values, which results in different quantization intervals. Thus, the eavesdropper is not able to generate the same secret key as the two users. In addition, RSS varies over time due to motion of the user and multipath propagation. The temporal and spatial variations of RSS can randomize the generated secret key, which enhances security of the RSS-based secret key generation.

In [15], a system is developed to use drones as communication relays for battery-free sensor networks. The drones are integrated with a Radio Frequency Identifier (RFID). In [16], a drone-enabled IoT relay system is developed to provide high-speed data collection in remote areas with no public network coverage. A drone-based onboard relay and a ground terminal collect the cached data of environmental monitoring sensors. In [17], a
path planning and navigation algorithm is studied for drones to relay computing tasks, which considers the environment, the flight conditions of the drones, and the quality of location estimation. The system relies on the collaboration of the drones to overcome the sensing limitations of the individuals, hence improving system-wide performance. In [18], the authors present a team-level programming model for a drone communication system, which expresses multiple sensing tasks with multiple drones. They also build a set of frameworks to support the team-level programming model on the drones. BLE-connected autonomous drones are exploited to build aerial relay communication networks [19]. A range-based localization algorithm is studied to reduce the localization errors of the drone based on the onboard sensory data and the RSS of the channel with the neighboring drones. Drones equipped with cameras are applied to cover live events in [20]. A centralized controller is developed to coordinate the drones as relay nodes to improve network throughput. In [21], a localization algorithm is presented for drones to find the location with the best link quality to relay data. The algorithm estimates the link qualities according to the sampled data, and utilizes tensor updates to make corrections to improve estimation accuracy.

An Internet of drones architecture is presented in [22] with an emphasis on security issues, such as privacy leakage, and the need for secure and efficient data sharing between drones. In [23], the movement of a drone is modeled by a set of rules. An intrusion detection and response scheme is developed to detect malicious anomalies by applying the rules to classify the monitored drones. An information fusion method is studied for the drone to detect GPS spoofing attacks by combining onboard visual sensors and the inertial measurement unit of the drone [24]. An image-based navigation system using error reduction on orientation features is also designed to assist the drone’s safe return after the GPS spoofing attack is detected. In [25], a security framework based on white-box cryptography is developed to protect the data and secret keys in the delivery drone system from white-box attacks. The security framework provides a public key infrastructure based authentication process for the drone to verify the identities of the customer, the delivery company, and the seller. A prototype drone monitoring system is built to detect the hardware failure and cyber attacks against the autopilot flight controller [26]. The monitoring system captures the real-time flight data and estimates the controller status and flight plan to track immediate changes of the flight parameters due to the hardware failures or cyber attacks. A legitimate wireless surveillance of the drone communications is demonstrated in [27, 28], where a legitimate drone is employed to track the flight of the suspicious drones. To obtain the flight information of the suspicious drones, the legitimate drone intentionally jams the suspicious receiver so as to force the suspicious transmitter to reduce its data rate, and hence increase the eavesdropping success. A tracking algorithm is developed for the legitimate drone to track the suspicious flight by comprehensively utilizing the intercepted packets, angle-of-arrival and received signal strength of the suspicious transmitter’s signal.

A BLE security scan framework is developed in [29] to examine the BLE applications in the absence of encryption or authentication. Taint analysis is used to identify the keys and nonces by exploring data flow patterns of the suspicious BLE applications. The authors of [30] design a key negotiation downgrade attack to break the security of BLE communications. An attacker can decrease the entropy of BLE long-term and session keys, and the attacker can enumerate the low-entropy keys with brute-force, and break the link-layer security. A custom profiling tool is presented to determine the minimum level of security [31]. Recommendations on how to reduce the security vulnerabilities of a device are provided.

The proposed BloothAir system makes the following key contributions:

- A secure BLE communication scheme based on received signal strength (RSS)-based key generation is developed to encrypt/decrypt the data transmission in each hop of the BloothAir system. The RSS measured at the drone or the ground device varies over time due to motions of the drones, which is quantized to generate the secret keys. The temporal and spatial variations of the RSS randomize the generated secret keys, which enhances security of the RSS-based key generation.
• A dynamic programming-based RSS quantization scheme is proposed to minimize the secret key bit mismatch rate (KBMR) of BloothAir, thereby improving the consistency of the generated keys. The RSS quantization intervals are recursively adjusted until a unanimous secret key is generated at the sender and the receiver in each hop of BloothAir. To the best of our knowledge, the proposed BloothAir is the first attempt to implement the RSS-based key generation with adaptive quantization intervals in multi-hop aerial relay networks.

• A new lightweight onboard Gust radio transceiver is specially developed to enable the BLE communication of the autonomous drones. The Gust radio transceiver works at 2.4GHz ISM band, and implements FHSS of BLE to reduce radio interference of the BLE communication with other wireless networks in the same frequency. Based on the Gust radio transceiver, BloothAir supports high-speed data transmission between the drones and the ground devices.

• To validate the effectiveness of BloothAir, a multi-hop aerial relay system testbed is demonstrated based on the MX400 drone platform [32] and the Gust radio transceivers, where the proposed dynamic programming-based channel quantization is implemented. Real-world experiments are conducted in outdoor environments to evaluate the RSS-based key generation, where the drones move along three different flight trajectories. The experiment results demonstrate that BloothAir achieves significantly lower secret key bit mismatch rate than the key generation benchmarks which use the static quantization intervals. Based on the data analysis, a tradeoff between the secret key bit mismatch rate and the secret key length is also revealed.

Some of our initial ideas on the proposed BLE-connected autonomous drones were briefly described in [33], while communication security, secret key generation, and implementation were not provided. Moreover, in [33], preliminary RSS measurements were presented to show the high correlation between the uplink and downlink of the airborne Bluetooth channel. In contrast, this paper presents in detail the RSS-based secret key generation scheme, as well as the implementation of the secure BloothAir system. We also reveal the optimal subproblem structure to qualify the use of dynamic programming, and achieve the optimal RSS quantization intervals. This paper also evaluates the BloothAir system in terms of secret key bits mismatch and randomness, and new results reveal the impact of quantization intervals on the end-to-end data delivery latency. All this is new, and was not provided in [33].

The rest of the paper is organized as follows. Section 2 illustrates the system architecture of the autonomous drone, and the secure BLE communication with the RSS-based secret key generation. Section 3 shows the implementation of the proposed BloothAir system. In Section 4, we demonstrate the design of the multi-hop aerial relay system testbed and the real-world experiments. The performance of BloothAir is also analyzed. Section 5 concludes this work.

2 SYSTEM OVERVIEW

In this section, we study the design of the BLE-connected autonomous drone which contains two system modules, i.e., autonomous flight control and secure BLE communications.

2.1 Autonomous flight control

Most off-the-shelf autonomous drones are also equipped with GPS and a large number of inertial sensors, a.k.a. Inertial Measurement Unit (IMU), e.g., three-axis accelerometers, gyroscopes, and magnetometers. The IMU continually monitors the rate of acceleration and variation in rotational attributes such as pitch, roll and yaw, while assisting calibration against orientation drift. To automatize the flight, the trajectory of the drone can be preset by a sequence of GPS coordinates in space, a.k.a. waypoints. The drone determines its current position by synthetically analyzing the sensory information from IMU and GPS. Moreover, onboard processors of the drone integrate the acceleration, together with an estimate of gravity and orientation, to control the real-time cruising
velocity and heading direction, so that the drone can timely and precisely arrive at the next waypoint on the trajectory.

Figure 1(b) depicts the autonomous flight control and BLE communication modules of the drone in BloothAir. The autonomous flight control of the drone is built based on the MindPX platform [34] which integrates the measurements of the onboard sensors, such as accelerometers, Gyro sensors, barometers, and compass sensors. Specifically, the MindPX platform contains a position and altitude estimator which analyzes the onboard sensor measurements to determine the current flight status of the drone. The position and altitude estimator also calculates the heading and accelerating values for the next time slot. Navigator is a storage element in the autonomous flight control module, which calculates and maintains flight waypoints (coordinates in 3D space) of the trajectory. Moreover, a position controller works with an altitude and rate controller to amend real-time rotational speeds of the propellers of the drone according to the estimated heading and accelerating values. As a result, the next waypoint which the drone heads to matches the flight trajectory stored in the navigator. In addition, the rotational speeds of the propellers are converted by a mixer to functional commands for physically controlling the motors connecting to the propellers. An actuator is embedded to control electric current input to the motors. Although the drones of BloothAir are in the autonomous flight mode, the navigator which interfaces with the Gust radio transceiver can receive control commands from other drones or ground remote controllers to adjust the trajectory during the flight. In addition, the waypoints (i.e., a set of 3D coordinates in the flying space) of the trajectory are predetermined, and can be optimized before the drone takes off for a specific purpose, e.g., covering the ground devices in a target field. The drone flies along the predetermined trajectory which consists of those potential waypoints. This paper focuses on the RSS-based secret key generation to secure the aerial relay system in the presence of the predetermined waypoints. The proposed RSS-based secret key generation in BloothAir is generic, and can work with other application-specific trajectories of the drones.

### 2.2 RSS-based secret key generation for secure BLE communications

In Figure 1(b), the RSS-based secret key generation is implemented in the communication module. Specifically, the drone and the ground device take turns to broadcast short channel sensing (CS) packets on each hop, via the Gust radio transceiver, to measure and sample the RSS readings. According to the reception of the CS packet, the RSS value can be measured by the drones and the ground devices. Therefore, the transmitter and the receiver at each hop (i.e., the data sender and drone A in the first hop, drone A and drone B in the second hop, and drone B and the data sink in the third hop) can have the RSS information of the channel between them. Since the time between transmissions of the CS packets at the drone or the ground device is much shorter than the inverse of the rate of change of the channel, the RSS readings at the sender or the receiver are typically identical. The CS packets have small length (less than 100 bits), which is much smaller than the size of a data packet. As an example, 10 CS packets are transmitted at each hop (5 packets from the transmitter or the receiver), and the length of a CS packet is 20 bits, which only contains the ID of the Gust radio transceiver. The total overhead of the CS packet is 200 bits, which is much smaller than the size of a data packet. Given the data rate of 2 Mbps, transmission time of the CS packets is about 0.1 ms. Thus, the overhead of CS is negligible due to the small amount of payload. Moreover, a channel quantizer discretizes the variation of the RSS over time into a sequence of binary secret bits. Particularly, the temporal and spatial variations of RSS can randomize the generated secret key, which enhances the security of the BLE communication between the drones and ground devices. Notations used in the paper are summarized in Table 1.

Let $m_s(t)$ and $m_r(t)$ denote the RSS measurement of the sender and the receiver at time $t$, respectively. $m_s(t), m_r(t) \geq H_{min}$. $H_{min}$ is the required minimum RSS for decoding the data packet. Given total $Q$ quantization intervals, the upper bound and lower bound of the $q$-th quantization interval ($1 \leq q \leq Q$) are denoted by $q^+$ and $q^-$, respectively. Thus, we have $q^- = H_{min}$ when $q = 1$. Let $\Delta(\cdot)$ denote the quantization function of the RSS measurement.
Table 1. The list of fundamental variables

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_s(t)$</td>
<td>RSS measurement at the sender</td>
</tr>
<tr>
<td>$m_r(t)$</td>
<td>RSS measurement of the receiver</td>
</tr>
<tr>
<td>$H_{min}$</td>
<td>required minimum RSS for decoding the data packet</td>
</tr>
<tr>
<td>$\Delta(\cdot)$</td>
<td>quantization function of the RSS measurements</td>
</tr>
<tr>
<td>$Q$</td>
<td>total quantization intervals</td>
</tr>
<tr>
<td>$S_q$</td>
<td>number of secret bits that mismatch at the $q$-th quantization interval</td>
</tr>
<tr>
<td>$q^+$</td>
<td>upper bound of the $q$-th quantization interval</td>
</tr>
<tr>
<td>$q^-$</td>
<td>lower bound of the $q$-th quantization interval</td>
</tr>
</tbody>
</table>

measurements, as given by

$$
\Delta(x, q^-, q^+) = \begin{cases} 
qu, & \text{if } q^- \leq x < q^+; \\
0, & \text{otherwise.} 
\end{cases}
$$

(1)

where $x$ takes either $m_s(t)$ or $m_r(t)$.

The sender and the receiver have to agree upon a unanimous secret key, so that the data of the sender can be successfully decoded by the receiver. $m_s(t)$ and $m_r(t)$ can be distorted by wireless channel fading and random noise, which results in independent channel quantizations at the sender and the receiver. As a result, the secret key bits generated at the sender can be inconsistent with the one generated at the receiver. To address this issue, dynamic programming based channel quantization is implemented in the BloothAir system, where the RSS quantization intervals, i.e., $\{q\}$, are recursively adjusted until the sender and the receiver generate a unanimous secret key. Let $S_q$ denote the number of secret bits that mismatch at the $q$-th quantization interval. $S_q$ can be given as the XOR value of $\Delta(m_s(t), q^-, q^+)$ and $\Delta(m_r(t), q^-, q^+)$, as given by

$$
S_q = \sum_{k=1}^{\text{KeyLength}} \oplus(\Phi(\Delta(m_s(t), q^-, q^+)), \Phi(\Delta(m_r(t), q^-, q^+)))_k,
$$

(2)

where $\oplus(\cdot)_k$ stands for the bitwise XOR operation of two binary sequences, and KeyLength denotes the length of the generated secret key. $\Phi(q)$ converts the quantized RSS value $q$ to a binary sequence with KeyLength bits. Moreover, KBMR can be given by $S_q$/KeyLength, and $0 < S_q$/KeyLength $\leq 1$.

Given $q^- = (q - 1)^+$, optimizing $q^+$ for minimizing $S_q$ can be defined as a dynamic programming problem $\eta_{q^+}$, which is

$$
\eta_{q^+} = \min_{q} \{S_q | q^- < q^+, q^- = (q - 1)^+\}.
$$

(3)

Based on the Bellman equation [35], the optimal solutions to the preceding subproblems $\eta_{(q+1)^+}$ can be used to solve $\eta_{q^+}$. Since $1^- = H_{min}$, the quantization intervals can be recursively adapted with the aim of minimizing $S_q$. Therefore, our proposed channel quantization scheme carries out backward induction in dynamic programming, which has been widely used to determine a sequence of optimal actions by reasoning backwards. According to (4),
the backward induction first assesses the upper bound of the last interval, i.e., $Q^+$, and then uses the outcome to determine the upper bound of the second-to-last interval, i.e., $(Q - 1)^+$. 

$$q^+ = \arg \min_{q^+ \in [1^+, Q^+]} \sum_{q=1}^{Q} S_q,$$  

(4)

This continues until the upper and lower bounds are decided for all the quantization intervals. The optimal solutions are $(q^-, q^+)$ ($1 \leq q \leq Q$), where $1^+ = H_{min}$. The proposed channel quantization scheme is summarized in Algorithm 1.

**Algorithm 1** Dynamic programming based channel quantization

1. **Initialize:** $Q, q^- = H_{min}$ when $q = 1$, and KeyLength.
2. **for** $q = 1$ to $Q$ **do**
3. $S_q \leftarrow (2)$.
4. **Backward induction** \rightarrow solving (3).
5. $\eta_{q^+} \rightarrow \min(S_q, \eta_{q^+})$, where the $q$-th quantization intervals has $(q^-, q^+)$. 
6. **end for**
7. $\Delta(x, q^-, q^+) \leftarrow (q^-, q^+)$. 
8. Quantize the RSS measurements at $t$ according to (1).
9. **Backward induction**
10. $q \leftarrow Q$.
11. **for** $q \geq 1$ **do**
12. Upper threshold: $q^+ \rightarrow (4)$.
13. **if** $q \neq 1$ **then**
14. **Trace backward:** $q^- \rightarrow (q - 1)^+$.
15. **else**
16. $q^- = H_{min}$.
17. **end if**
18. $\eta_{q^-} \leftarrow S_q$.
19. $q \leftarrow q - 1$.
20. **end for**

A secret key reconciliation scheme is critical to reconcile the mismatch on the secret bits between the sender and the receiver at each hop. Without loss of generality, we exploit the RSS-based secret key generation in conjunction with Cascade [36] in BloothAir. Cascade divides the secret bits into several blocks for bits permutation according to the predetermined permutation and parity. The ground devices and the drones permutate their secret bits in the same way, divide them into blocks, compute parities and check for parity mismatches. For each mismatch, the ground devices or the drones perform a binary search on the block to find if some bits can be changed to make the block match the parity. These steps are iterated until all blocks match, and the secret bit discrepancies are reconciled. Note that the BloothAir system is compatible with state-of-the-art secret key reconciliation schemes, such as BCH code [37], or Turbo code [38]. The overhead of the reconciliation can be reduced by taking advantage of a high key generation rate achieved by BloothAir. Therefore, the dynamic programming based channel quantization guarantees the secret key generation and correctness at the drones and the ground devices in the presence of the RSS measurement randomness.

For enhancing fault-tolerant capability of the BLE communication in BloothAir, a 24-bit Cyclic Redundancy Check (CRC) is appended to the end of the encrypted data packet in each hop for error detection. Moreover, data
whitening is applied to prevent long strings of 0’s and 1’s in the transmitted sequence. The data packet is then shifted to a convolutional forward-error-correction (FEC) encoder and a pattern mapper for improving receiver sensitivity and error correction. Furthermore, when a packet is received via the Gust radio transceiver, the drone carries out accordingly the pattern mapper, FEC decoding, dewhiteniing, and CRC check to retrieve the source packet. Then, the secret key is used to decrypt the packet.

2.3 Eavesdropping attacks

An eavesdropper is typically wavelengths away from the ground device or the drone, and experiences an independent radio channel. This is because the eavesdropper can be detected or noticed when it is too close (e.g., less than a few wavelengths) to the BloothAir system. An eavesdropper can move around the ground device or the drone, while keeping some distance to not be noticed.

The eavesdropper can overhear the CS packets transmitted by the ground device or the drone, measure the RSS variation, and quantize the RSS measurements in attempt to recover the secret key. The eavesdropper that attempts to decode the information of the data sender is not interested in disrupting the data transmission in BloothAir. Moreover, the eavesdropper is not able to possess the a-prior knowledge of RSS measurements between two arbitrary waypoints that the drones are, since such environmental sensitive information requires significant effort to obtain, e.g., recording RSS fingerprints of every movement of the drone along the flight trajectory in advance.

3 IMPLEMENTATION OF THE BLOOTHAIR SYSTEM

In this section, we present the implementation of the BloothAir system and the lightweight Gust radio transceiver specifically designed by AirMind Company [39] for the BLE-connected drones.

The BloothAir testbed is built with two autonomous drones and two ground devices, and all of which connect to the Gust transceiver, as shown in Figure 2(a). We develop Graphical User Interface (GUI) in Python to display the real-time flight trajectories of the drones, and the real-time system statistical information, e.g., packet loss, average RSS values, and packet delivery delay, at the sink node. We implement the autonomous flight control module, as shown in Figure 1(b), based on the MX400 platform [32], while the RSS-based secret key generation is carried out on the Gust radio transceiver to enable the secure BLE communication. The autonomous flight control and the BLE communication modules are programmed in C/C++.

Autonomous flight control. The general form of the autonomous flight control module is:

```plaintext
1: Procedure: AutonomousFlightControl()
2:   update_trajectory();
3:   if Onboard GPS is activated then
4:     takeoff();
5:     set_flight_mode();
6:     current_waypoint();
7:     if mission_complete() then
8:       return_to_base();
9:     if GPS matches the location of the base then
10:    landing();
11: end if
12: end if
13: end if
```

In AutonomousFlightControl(), the drone updates the waypoints of the flight trajectory which has been predetermined by the
user via update_trajectory(). When the onboard GPS signal is activated, the drone carries out takeoff() to take off, and the altitude of the drone is maintained according to the initial waypoint. The drone supports three different flight modes, i.e., autonomous flight, manual controlled flight, and altitude/position holding. Specifically, in the autonomous flight mode, the drone moves automatically along the waypoints of the trajectory. In the manual controlled flight mode, the flight of the drone has to be manually controlled by a remote controller. In the altitude/position holding mode, the drone hovers at a specific position and altitude. Particularly, set_flight_mode() in Line 5 is executed to enable the autonomous flight mode in BloothAir. Furthermore, current_waypoint() returns the current waypoint of the drone on the trajectory. mission_complete() checks if the flight mission of the drone is completed, and return_to_base() guides the drone to return to the start point of the trajectory once the flight mission is completed. The landing() function commands the drone to descend and land.
neighbor_gust_device_found() returns true. If discover_time_out() is not timeout, connect() is carried out to enable the BLE connection between the two transceivers. The neighboring Gust radio transceiver that receives the advertising packets transmit a connection request to directly establish a connection. If discover_time_out() is timeout, stop_discovering() stops the broadcast of the advertising packets. After a delay of discover_time_out(), the Gust radio transceiver calls start_discovering() to search the neighbors again.

Procedure: SetUpBLEconnections()

for each of the Gust radio transceivers do
start_discovering();
if discover_time_out() is not 0 and neighbor_gust_device_found() then
connect();
else
stop_discovering();
discover_time_out();
end if
end for

Once the BLE connections from the data sender to the sink are set up, StartDataTransmission() at the sender kicks off the data transmissions, where identity information of the Gust radio transceivers is logged in a vector connectedDeviceSet at the ground devices and the drones. Moreover, a log file is created at each Gust radio transceiver to log the transmitted data and received data by using create_log_file(). If the transmission of the source data is not completed, namely, if_end_of_file(source_data_file) returns 0, dt_send_data_packet(next_gust) continues to transmit the data to the next hop. The ground devices and the drones invoke StopDataTransmission() when all the data packets are transmitted. stop_logging() and close_log_file() are carried out to stop logging the data.

Procedure: StartDataTransmission()

all_acked = true;
for the Gust transceivers in connectedDeviceSet do
dt_start_logging_cmd();
if wait_for_ack() == failed then
all_acked = false;
end if
end for
if all_acked then
create_log_file();
while if_end_of_file(source_data_file) is not 0 do
dt_send_data_packet(next_gust);
end while
end if

Furthermore, we measure the transmit power of the Gust radio transceiver, as shown in Figure 2(b). The Gust radio transceiver is developed based on BLE which is a new Bluetooth Core Specification for long-distance communications [40]. Furthermore, the maximum transmit power of the Gust radio transceiver is up to 20.154 dBm, which leads to the BLE communication range of about 1 km [41]. In particular, the Gust radio transceiver
1: **Procedure**: StopDataTransmission()
2: for the Gust transceivers do
3:   stop_logging();
4: end for
5: close_log_file();

extends the communication range of the ground devices and the drones in BloothAir due to an onboard integrated radio power amplifier, compared to most of the off-the-shelf BLE transceivers. In addition, the transmit power of the Gust transceiver is adaptive to the channel conditions. Namely, Gust increases the transmit power whenever the channel is poor. As a software defined radio, radio stack firmwares of the Gust radio transceiver can be flashed to support multiple software defined 2.4 GHz physical layers, such as BLE, Thread and Zigbee.

4 EXPERIMENTAL EVALUATION OF THE BLOOTHAIR SYSTEM

In this section, we demonstrate testing scenarios and run-time measurements in real-world experiments. Moreover, we present the experimental evaluation of the BloothAir system in terms of KBMR, secret key bits randomness, end-to-end data delivery latency, and battery level monitoring of the drone. In particular, we define KBMR as a ratio of the data packet length and the number of data bits which are decoded at the sink, but mismatch the source data bits generated by the data sender. To further reveal the security of BloothAir, we demonstrate the KBMR of the secret key that is generated at the eavesdropper which overhears the CS packets of the ground devices and the drones.

4.1 Real-world experimental setup

We conduct experiments in a parkland (The Lake of Lotus, Nanjing, China). The data sender and the sink are obstructed by trees and bushes over a long distance. Hence, there is no LoS between the data sender and the sink. The payload of the data packet generated by the data sender has 300 bytes, while the maximum data rate of the Gust radio transceiver is 2 Mbps. The data rate and transmission power of all the Gust radio transceivers are set to the maximum level in our experiments.

Drone A and drone B patrol around the data sender and the sink along their predetermined trajectories. Three different flight trajectories are considered, namely, Sce-I, Sce-II, and Sce-III, as displayed on the GUI in Figures 3(a), 3(c), and 3(e), respectively. The trajectories are meticulously designed to prevent potential flight collisions of the drones, where the closest waypoints of the two trajectories are away from each other for at least 5 m. The size of the trajectory is about 500 m x 500 m, which is smaller than the communication range of the Gust radio transceiver. Therefore, the drones can maintain the BLE connections along the flight. Furthermore, the patrol velocity and the flight altitude of the drones are set to 5 m/s and 30 m, respectively.

By sampling and quantizing the RSS measurements, three pairs of secret keys are generated at the data sender, drone A, drone B, and the data sink. The transmission of the source data is initiated by the data sender. The packets are encrypted at the sender, and immediately transmitted to the next hop, i.e., drone A. Once the packet is successfully received by drone A, the secret key is used to decode the data. Acknowledgment is sent to the data sender, so that a new data packet can be transmitted. In the same way, the encrypted data packet is forwarded all the way to the data sink. Although increasing the key generation rate raises the randomness of the generated keys, the end-to-end data delivery time can be prolonged due to the key generation delay at the drones. Therefore, it is critical to properly configure the key generation rate according to the required delay tolerance of the data delivery.
Real-world experiments are conducted in a parkland (The Lake of Lotus, Nanjing, China). (a), (c) and (e) present three testing scenarios, namely, Sce-I, Sce-II, and Sce-III, with different flight trajectories of the autonomous drones. (b), (d), and (f) show the real-time RSS values of every BLE channel in BloothAir based on the reception of 2310 packets.

For performance comparison, the BloothAir system that carries out the proposed dynamic programming based channel quantization for the secret key generation is compared with two RSS-based secret key generation schemes in literature:

Secret key generation based on fixed quantization intervals (FQI). The quantization intervals of the RSS are predefined at the ground devices and the drones. The RSS measurements are individually quantized at the transmitter and the receiver in each hop according to the fixed quantization intervals [42]. Thus, the key generated by FQI is independent of the RSS variation.

Secret key generation based on median RSS quantization (MRQ). The transmitter and the receiver independently measure and quantize the RSS values. The quantization thresholds are calculated at the transmitter and the receiver according to median values of their local RSS measurements [43]. When the RSS measurement is higher than the median value, the key bit is encoded to 1, otherwise, the key bit is 0. Thus, the secret key generation with MRQ is based on the RSS variation that is locally observed by the transmitter or the receiver.

4.2 Real-time RSS along the flight trajectories of the drones

Figure 3(b) presents the RSS values of the three hops in Sce-I, which are measured at the data sender, drone A, drone B, or the sink. Similarly, the RSS values of the three hops in Sce-II and Sce-III are shown in Figure 3(d) and 3(f), respectively. Generally, the RSS measurements at the two ground devices and the two drones vary between -60 dB and -20 dB, and are dramatically affected by the trajectories of the drones. Particularly, the RSS at drone A and drone B in Sce-II periodically fluctuates around -20 dB. This is because the two drones move along one side of the square shaped trajectory, which has the shortest distance to each other. We also can see that the RSS between drone A and drone B in Sce-I is lower than the one in Sce-II and Sce-III. This is reasonable because the trajectories of drone A and drone B in Sce-I have the longest distance to each other, compared to Sce-II and Sce-III.

Figures 4(a), 4(b), and 4(c) present RSS correlations of the three hops in Sce-I, Sce-II, and Sce-III, respectively. As observed, the RSS measurements at the transmitter and the receiver of each hop have a strong correlation, higher than 0.975. We can also see that the correlation of the RSS measurements is highly affected by the movements of the drones. When the drone moves close to the ground device or the other drone, the correlation gradually grows. Otherwise, the correlation drops. Particularly, the correlation of the RSS at drone A and drone B has the highest fluctuation (from 0.977 to 0.999), compared to the Source-droneA and droneB-sink links. This is because movements of the two drones lead to a high RSS variation, from -60 dB to -10 dB (as shown in Figure 3), which is much higher than the RSS variation of the Source-droneA and droneB-sink links.

4.3 Secret key bit mismatch rate

Figure 5 evaluates the KBMR of the BloothAir system with regards to different secret key generation schemes, where the number of quantization intervals increases from 5 to 30. In particular, the key length of FQI and MRQ is fixed to 6 bits. In general, KBMR increases with the growth of $Q$ since the small number of quantization intervals makes the ground devices and the drones easy to agree upon a unanimous secret key. However, decreasing $Q$ results in a rise of security vulnerability on the generated secret key, where the eavesdropper may generate the same secret key as the BloothAir system. Therefore, it is important to balance the system security and the required KBMR in terms of the setting of $Q$.

We also observe in Figure 5 that BloothAir achieves a lower KBMR than FQI and MRQ. BloothAir with the key length of 6 bits outperforms FQI and MRQ with substantial gains of 54.2% and 42.5%, respectively, when $Q = 20$. Although FQI and MRQ can achieve a similar KBMR to BloothAir under $Q = 5$, the quantization errors of FQI and MRQ dramatically increase with $Q$. This is because the secret key generated by FQI is independent of the RSS variation due to the fixed quantization intervals, while MRQ generates the secret key based on the local RSS measurement. Therefore, small randomness of the RSS measurements results in a large number of quantization errors when $Q$ increases. In contrast, BloothAir carries out the dynamic programming-based channel
Fig. 4. Cross correlations of the RSS measurements between the data sender and drone A, drone A and drone B, and drone B and the data sink in Sce-I, Sce-II, and Sce-III.

quantization (as shown in Algorithm 1) to generate a unanimous secret key at the ground devices and the drones, which adapts the quantization intervals to the time-varying RSS.

Figure 5 also reveals a tradeoff between the KBMR and the secret key length. We see that BloothAir with the key length of 6 bits achieves about 16.2% lower KBMR than the key length of 18 bits. Therefore, shortening the
Fig. 5. Key bit mismatch rate at the data sink with regards to the secret key generation schemes, where the number of quantization intervals increases from 5 to 30. The error bars show the standard deviation over the independent experiments in Sce-I, Sce-II, and Sce-III.

secret key length of the ground devices and the drones reduces the KBMR in BloothAir. However, a drop in the number of secret bits results in a degradation of BLE communication security, where the same key can be generated by the eavesdropper to decode the data.

4.4 Randomness of the secret keys

In this experiment, the standard NIST statistical test suite [44] containing 15 different tests is employed to verify the randomness of the secret keys in the BloothAir system. Particularly, 8 of the 15 NIST tests are shown in Table 2. The reason of selecting the 8 NIST tests to assess BloothAir is because their recommended input size meets bit streams of the secret keys in our experiments. Note that the remaining 7 NIST tests require a very large input bit stream (more than 106 bits), where a large number of keys (in gigabytes) need to be generated.

In Table 2, the p-values [44, 45] of the secret keys are calculated in each of the tests. The p-value is the probability that a perfect random number generator would have produced a sequence less random than the input sequence that is tested. A p-value larger than 0.01 which is the threshold to pass the test indicates that the secret bit streams are random with a confidence of 99%.

In each NIST test in Table 2, 7 experiments, i.e., A to G, are conducted to obtain the p-values, where Q is set to 2, 5, 10, 15, 20, 25, or 30 in turn. We can observe that all the secret keys generated by BloothAir pass the test, and have much larger p-value than the threshold 0.01. Furthermore, the randomness of the keys generated by BloothAir substantially increases the time complexity of cracking the keys at the eavesdropper, hence protecting the data dissemination from the eavesdropping attacks.

4.5 KBMR of the eavesdropper

In this experiment, the experimental field is divided into 36 small blocks. The eavesdropper randomly moves within a block at a time while overhearing the CS packets of the ground devices and the drones, as shown in Figure 6. In each block, the eavesdropper overhears about 2 KB of data packets. The dynamic programming based channel quantization is used to generate the secret keys to decode the overheard data.
Table 2. P-values from NIST statistical test suite. To pass the test, all p-values must be greater than 0.01.

<table>
<thead>
<tr>
<th>Test</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>0.022796</td>
<td>0.036879</td>
<td>0.011412</td>
<td>0.026537</td>
<td>0.025389</td>
<td>0.033622</td>
<td>0.016246</td>
</tr>
<tr>
<td>Block Frequency</td>
<td>0.999999</td>
<td>0.999940</td>
<td>0.999853</td>
<td>0.999582</td>
<td>0.999817</td>
<td>0.999721</td>
<td>0.999882</td>
</tr>
<tr>
<td>Cumulative sums (Fwd)</td>
<td>0.038559</td>
<td>0.053713</td>
<td>0.013074</td>
<td>0.020848</td>
<td>0.018853</td>
<td>0.026545</td>
<td>0.024965</td>
</tr>
<tr>
<td>Cumulative sums (Rev)</td>
<td>0.041947</td>
<td>0.068227</td>
<td>0.013074</td>
<td>0.022824</td>
<td>0.017245</td>
<td>0.027245</td>
<td>0.014379</td>
</tr>
<tr>
<td>Non-overlapping Matching</td>
<td>0.861270</td>
<td>0.621560</td>
<td>0.890290</td>
<td>0.274253</td>
<td>0.509563</td>
<td>0.304055</td>
<td>0.577296</td>
</tr>
<tr>
<td>Longest run of ones</td>
<td>0.053256</td>
<td>0.038262</td>
<td>0.051872</td>
<td>0.015257</td>
<td>0.029642</td>
<td>0.046281</td>
<td>0.049283</td>
</tr>
<tr>
<td>FFT</td>
<td>0.042221</td>
<td>0.033709</td>
<td>0.029009</td>
<td>0.033005</td>
<td>0.011013</td>
<td>0.020259</td>
<td>0.033709</td>
</tr>
<tr>
<td>Approx. Entropy</td>
<td>0.295489</td>
<td>0.746822</td>
<td>0.813242</td>
<td>0.289959</td>
<td>0.562620</td>
<td>0.641605</td>
<td>0.208921</td>
</tr>
<tr>
<td>Serial</td>
<td>0.02, 0.01</td>
<td>0.04, 0.01</td>
<td>0.01, 0.04</td>
<td>0.02, 0.04</td>
<td>0.01, 0.02</td>
<td>0.05, 0.02</td>
<td>0.02, 0.02</td>
</tr>
</tbody>
</table>

Figures 6(a) and 6(b) show the average KBMR of the eavesdropper, given $Q = 10$ or $30$. We can see that the KBMR of the eavesdropper is higher than 75% in general. It confirms that the eavesdropper at a different location experiencing independent channel conditions is not able to obtain the same key as the ground devices or the drones. It also confirms that a higher $Q$ of the BloothAir system leads to the growth of KBMR of the eavesdropper (up to 98.6%), wherever the eavesdropper moves to. Furthermore, the KBMR of the eavesdropper is dramatically affected by the distance to the ground devices or the drones. The reason could be that the correlation of the RSS measurements drops when the eavesdropper is far from the ground devices or the drones.

### 4.6 Data delivery latency

Figure 7 depicts the end-to-end data delivery latency of the key generation schemes, where the number of quantization intervals increases from 5 to 30. Given the key length of 18 bits, the secret key generation of BloothAir takes between 0.35 second and 0.43 second, as $Q$ increase from 5 to 30. This is similar to the communication latency of MRQ and FQI in which the quantization of RSS is not optimized. Therefore, the dynamic programming based channel quantization in BloothAir does not cause extra communication latency on the ground devices and the drones, which is efficient to secure the BLE communication. When the key length is 6 bits, the delay of the BloothAir system is between 0.24 second and 0.29 second, which is lower than the one with 18 bits for about 0.12 second. This is reasonable, since a shorter key length takes less time on the secret key generation, and the decoding time of the drones and the sink node is also reduced by the short key length.

Table 3 shows the average hop-by-hop data delivery latency in BloothAir with respect to the number of quantization intervals. In general, the delay of the proposed dynamic programming based channel quantization is similar to the one achieved by FQI or MRQ. Particularly, the communication delay between drone A and drone B is about 0.15 second longer than the delay of the other two links, i.e., the link between the data sender and drone A, and the link between drone B and data sink. The reason is that drone A hovers around the data sender, while drone B hovers around the data sink. The link between the two drones experiences a high path loss when the two drones move far from each other. This prolongs the packet retransmission delay on the drones.
(a) KBMR of the eavesdropper when $Q = 10$. (b) KBMR of the eavesdropper when $Q = 30$. (c) The eavesdropper moves in one block at a time while overhearing the CS packets.

Fig. 6. KBMR of the eavesdropper given $Q = 10$ or 30. The eavesdropper moves around the ground devices and the drones while overhearing their CS packets.

4.7 Battery levels monitoring of the drone

Figure 8 shows the onboard battery level of the drone over about 14 minutes. It is observed that the energy consumption of the drone on the flight and the BLE communication grows linearly over time to 800 mAh. Moreover, the drone continuously hovers, which indicates a low energy consumption of the Gust radio transceiver. In particular, the battery level of the ground devices is not presented since the ground devices can be connected to persistent power supplies.

5 CONCLUSIONS

This paper proposed the BloothAir system for the secure multi-hop data relays with the BLE-connected autonomous drones. The RSS-based secret key generation scheme was designed and implemented to encrypt and decrypt the BLE communication, where the RSS quantization intervals are recursively adjusted until a unanimous secret key is generated for the drones and the ground devices at each hop in BloothAir. To validate performance of BloothAir, we developed a multi-hop aerial relay system testbed based on the MX400 platform and the new onboard Gust radio transceiver specially developed for the BLE communication of the drone. We also implemented a GUI to display the real-time trajectories as well as the packet reception rates and transmit power of the drones. Extensive real-world experiments demonstrate that the BloothAir system achieves a significantly lower KBMR than the key generation benchmarks which use the static quantization intervals. In addition, the high randomness
Fig. 7. End-to-end data relay latency with regards to the secret key generation schemes, where the number of quantization intervals increases from 5 to 30. The error bars show the standard deviation over the independent experiments in Sce-I, Sce-II, and Sce-III.

Table 3. Average hop-by-hop delivery latency in BloothAir.

<table>
<thead>
<tr>
<th>The hop and quantization intervals</th>
<th>data relay latency (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BloothAir (6 bits)</td>
</tr>
<tr>
<td>data sender and drone A</td>
<td></td>
</tr>
<tr>
<td>Q = 5</td>
<td>0.06</td>
</tr>
<tr>
<td>Q = 10</td>
<td>0.07</td>
</tr>
<tr>
<td>Q = 15</td>
<td>0.06</td>
</tr>
<tr>
<td>Q = 20</td>
<td>0.05</td>
</tr>
<tr>
<td>Q = 25</td>
<td>0.07</td>
</tr>
<tr>
<td>Q = 30</td>
<td>0.05</td>
</tr>
<tr>
<td>drone A and drone B</td>
<td></td>
</tr>
<tr>
<td>Q = 5</td>
<td>0.14</td>
</tr>
<tr>
<td>Q = 10</td>
<td>0.13</td>
</tr>
<tr>
<td>Q = 15</td>
<td>0.15</td>
</tr>
<tr>
<td>Q = 20</td>
<td>0.11</td>
</tr>
<tr>
<td>Q = 25</td>
<td>0.16</td>
</tr>
<tr>
<td>Q = 30</td>
<td>0.13</td>
</tr>
<tr>
<td>drone B and data sink</td>
<td></td>
</tr>
<tr>
<td>Q = 5</td>
<td>0.07</td>
</tr>
<tr>
<td>Q = 10</td>
<td>0.08</td>
</tr>
<tr>
<td>Q = 15</td>
<td>0.08</td>
</tr>
<tr>
<td>Q = 20</td>
<td>0.09</td>
</tr>
<tr>
<td>Q = 25</td>
<td>0.05</td>
</tr>
<tr>
<td>Q = 30</td>
<td>0.06</td>
</tr>
</tbody>
</table>

of the generated keys in BloothAir substantially increases the KBMR of the eavesdropper, hence protecting the BLE communication from the eavesdropping attacks.

Fig. 8. The changes of the consumption (mAh), voltage (V), and current (A) of the drone over time during the experiment.

ACKNOWLEDGMENTS

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (CEC/04234); also by national funds through the FCT, under CMU Portugal partnership, within project CMU/TIC/0022/2019 (CRUAV).

REFERENCES


