Journal of
Sensor and Actuator Networks

MDPI

Article

A Dynamic Light-Weight Symmetric Encryption Algorithm for Secure Data Transmission via BLE Beacons

Sam Banani, Surapa Thiemjarus *, Kitti Wongthavarawat and Nattapong Ounanong

Health Innovation and Information Assistive Technology and Medical Devices Research Center, National Science and Technology Development Agency, Pathum Thani 12120, Thailand; sam.ban@nstda.or.th (S.B.); kitti.won@nstda.or.th (K.W.); nattapong.oun@ncr.nstda.or.th (N.O.)
* Correspondence: surapa.thi@nstda.or.th; Tel.: +66-2564-6900 (ext. 2475)

Abstract: Pervasive sensing with Body Sensor Networks (BSNs) is a promising technology for continuous health monitoring. Since the sensor nodes are resource-limited, on-node processing and advertisement of digested information via BLE beacon is a promising technique that can enable a node gateway to communicate with more sensor nodes and extend the sensor node's lifetime before requiring recharging. This study proposes a Dynamic Light-weight Symmetric (DLS) encryption algorithm designed and developed to address the challenges in data protection and real-time secure data transmission via message advertisement. The algorithm uses a unique temporal encryption key to encrypt each transmitting packet with a simple function such as XOR. With small additional overhead on computational resources, DLS can significantly enhance security over existing baseline encryption algorithms. To evaluate its performance, the algorithm was utilized on beacon data encryption over advertising channels. The experiments demonstrated the use of the DLS encryption algorithm on top of various light-weight symmetric encryption algorithms (i.e., TEA, XTEA, PRESENT) and a MD5 hash function. The experimental results show that DLS can achieve acceptable results for avalanche effect, key sensitivity, and randomness in ciphertexts with a marginal increase in the resource usage. The proposed DLS encryption algorithm is suitable for implementation at the application layer, is light and energy efficient, reduces/removes the need for secret key exchange between sensor nodes and the server, is applicable to dynamic message size, and also protects against attacks such as known plaintext attack, brute-force attack, replaying attack, and differential attack.

Keywords: temporal encryption key; light-weight encryption; BSNs

Citation: Banani, S.; Thiemjarus, S.; Wongthavarawat, K.; Ounanong, N. A Dynamic Light-Weight Symmetric Encryption Algorithm for Secure Data Transmission via BLE Beacons. *J. Sens. Actuator Netw.* **2022**, *11*, 2. https://doi.org/10.3390/jsan11010002

Academic Editor: Ioannis Chatzigiannakis

Received: 1 October 2021
Accepted: 22 December 2021
Published: 27 December 2021

1. Introduction

Pervasive sensing with Body Sensor Networks (BSNs) represents the latest evolution of diagnostic tools from the traditional episodic management to continuous monitoring of patients' physical and physiological parameters under their natural conditions. Miniaturized sensors attached or worn on the user's body, along with ambient sensors, are a new generation of computers that can be tailored to tackle healthcare management challenges. Applications that have been developed include chronic and non-communicable disease monitoring [1–7], fall monitoring [8–10], emergency care management [11–13], rehabilitation [14–17], fitness and lifestyle tracking [18,19], and sport performance monitoring [20–22].

Wireless Body Sensor Network (WBSNs) communication technologies are mostly focused on low cost, low power, pervasive networking, and short-range communication. Some of the wireless technologies that support the communication between sensor nodes and gateways are Bluetooth, Bluetooth Low Energy (BLE), Ultra-Wideband (UWB), ZigBee (IEEE 802.15.4), and Z-wave. Each wireless technology has different characteristics and supports different network topologies, frequencies, and communication ranges. To understand the security requirements and possible attacks in a pervasive sensing system,

we need to understand the infrastructure of the network in use as well as device communication protocols. Figure 1 depicts an overview of a BSN-based pervasive sensing system architecture. A typical system consists of three network modules: (1) personal area network (BSN), (2) local area network, and (3) global network. The personal area network is the network of sensor nodes (e.g., motion, EMG, and ECG) for measuring/monitoring physical and physiological parameters of a patient. The sensory data or the derived information will be collected and forwarded to the local area network via the sensor gateways. The received data will then be transmitted to a server (local or cloud) via the internet gateway. Users (patients and/or caretakers) are allowed to configure and receive sensor feedback for patient monitoring and further data analysis at the client side through an application software. The sensitive medical data that are collected by IoT devices requires privacy and security to protect the data at different levels of access in the local and public cloud [23].

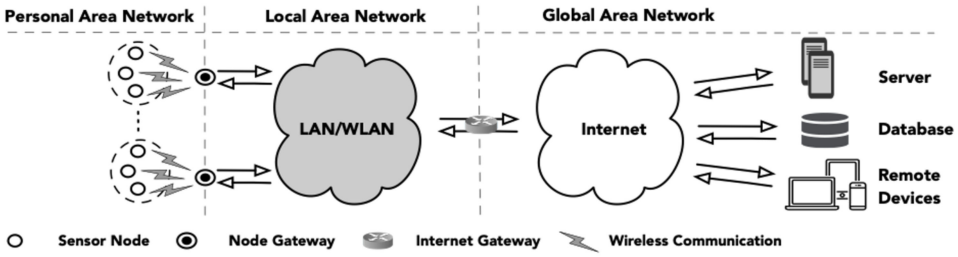

Figure 1. Overview of a typical pervasive sensing architecture.

In general, network attacks can be broadly classified as passive or active [23,24]. In passive attacks, the main goal of intruders is to monitor, learn, and exploit the information in the system (e.g., eavesdropping and traffic analysis). In active attacks, the main goal of intruders is to break into the network to alter system resources (e.g., message corruption, data modification, and replay attack) or interrupt its operation (e.g., resource exhaustion). Some of the possible network attacks are as follows:

- Eavesdropping [25–29]: attackers can capture data packets wirelessly sent from sensor nodes to understand and use the information;
- Data corruption [26,27,29,30]: attackers may delete partial or entire data making processing such data packets a waste of resources;
- Data modification [26,27,29]: attackers modify/remove partial or entire data in the data packets;
- Replay attack [30,31]: attackers eavesdrop and intercept data packets and fraudulently delay or resend them, causing confusion and additional resource consumption;
- Denial of Service (DoS) [30,32]: attackers transmit traffic to overload a machine or network to make services unavailable for legitimate users;
- Tracking attack [25]: attackers identify messages from the same origin in order to collect information about an individual or to trace messages for personal information such as the sender's location;
- Impersonation attack [27,29,33]: attackers introduce themselves as one of the legitimate nodes in a network or in a communication protocol.

While other parts of the network can be protected with generally well-known solutions, security design for sensor nodes is more challenging. Since a BSN's sensor nodes constantly share and transmit sensitive physiological, physical, and/or environmental information related to an individual to other devices in close proximity (a few meters) and eventually to the cloud, the privacy and security of the data is a major concern [34–39]. In addition, the miniaturized sensors have limited resources in terms of computation, battery power, memory, and network bandwidth [24,40].

To address the resource constraint issues, on-node processing and advertisement of digested information can be used instead of streaming raw data out of sensor nodes. With on-node processing and connectionless communication, network bandwidth and battery

consumption on the sensor node can be significantly reduced. Compared to classic Bluetooth, BLE beacons transfer small amount of data at a regular time intervals, thus consume much less power. In the past, BLE beacons were often used for proximity marketing or tracking applications. The data packet usually contains an ID with spatial data, status of the beacon (e.g., beacon temperature, battery status), and in the case of Eddystone beacons, a URL. However, the trend is changing. The low-power feature of BLE beacon advertisement is a promising enabling technology for secured transmission of digested sensing data in wearable and IoT applications. It allows sensor nodes to last weeks, months or even a year with a button cell depending on the transmission rate.

In the past, most studies on beacon security focused on protection of the sender's identity [41–45]. To our knowledge, apart from Google's Eddystone [43] which applies the Advanced Encryption Standard (AES) for data encryption in Eddystone-TLM (for broadcasting telemetry information about the beacon), there is limited or no other work on secured light-weight encryption of data transmitted via BLE beacon. The problem with AES is that it is not suitable for implementation on resource-constrained sensor nodes [46]. Several light-weight data encryptions for IoT applications have been previously proposed [46–50]; however, they mostly rely on a fixed encryption key, fixed block size, and key management.

Dynamic encryption [51] can enhance data security by changing the cipher on every data transaction. BRISK [46], proposed by Dwivedi, uses a cipher with simple operations but varying cipher specifications selected from a given small pool. The algorithm is designed targeting resource-constrained applications such as RFID tags. Unlike RFID, which can hold data up to a few thousand bytes, BLE allows a device to broadcast 25~28 bytes of data (manufacture specific data) depending on the advertising type [52]. BLE beacons may carry sensory information in a small packet broadcasted at a regular time interval. The length of the data embedded in each data frame is relatively small and varied depending on the application or platform. Most existing light-weight encryption algorithms use a fixed-sized cipher block (i.e., 32, 64, 128 bytes) and require padding unused space for data encryption. To generate the encrypted data that fits the beacon payload, the block size may need to be reduced, affecting the security of the encryption.

To enable secure data transmission via BLE beacon, this paper proposes a Dynamic Light-weight Symmetric (DLS) encryption algorithm for security enhancement of the existing data encryption algorithms with dynamic temporal encryption keys with small additional overhead. The contributions of this study are as follows:

- A light-weight cryptography for secure data transmission via BLE beacon that applies dynamic temporal encryption key generation on top of existing symmetric encryption algorithms for further security enhancement;
- Two levels of security for secret key generation based on fixed-node secret key and random numbers obtained periodically from the server (adjustable security level by changing the number of bits of random numbers and counters);
- Support dynamic size of messages without requiring padding (thus saving more energy compared to a standard block cipher) and yielding cyphertext with the same size as the input message;
- Demonstration of its application across baseline encryption algorithms;
- Demonstration of its application on beacon data encryption by assessing its performance when implemented in a miniaturized sensor node.

This paper is organized as follows. Section 2 provides a background review of some well-known symmetric encryption algorithms and existing encryption algorithms for sensor networks. Section 3 describes the proposed dynamic light-weight symmetric encryption algorithm. Section 4 demonstrates the experimental setup, results, and cryptanalysis. A comparative analysis between the proposed algorithm and some of baseline symmetric encryption algorithms in terms of memory consumption, processing overhead, power consumption, avalanche effect, key sensitivity, and randomness is presented. Section 5 concludes the paper.

2. Background and Related Work

In general, cryptographic algorithms can be broadly classified in two categories: asymmetric key cryptography and symmetric key cryptography [53]. Asymmetric key cryptography (public key algorithm) uses a key pair, namely, a public key and a private key. A public key that is used for data encryption is available to everyone. A private key, on the other hand, is used for generating a digital signature and decrypting the data which is encrypted by the public key. Symmetric key cryptography (secret key algorithm) uses the same key both for data encryption and decryption. Symmetric key algorithms typically consume less resources and are less complicated to implement compared to asymmetric encryption algorithms [53–56], making them more suitable for resource constrained systems. Symmetric key algorithms can be classified into stream cipher and block cipher. In stream cipher, the plaintext is converted to ciphertext one bit or byte at a time. In block cipher, the plaintext is broken down into fixed size block (i.e., 64 bits) before the conversion is performed. Although stream cipher is usually faster and requires fewer resources, block cipher is generally more secure and more popular, as it is highly analyzed and studied by cryptographers.

In 1994, the Tiny Encryption Algorithm (TEA) [57] was proposed by Wheeler and Needham based on the 64- or 32-round FN-structure. It uses a 64-bit cipher block and a 128-bit encryption key. TEA is one of the most efficient encryption algorithms in terms of memory consumption and processing time. Therefore, it can be easily implemented and integrated with embedded systems [53]. However, TEA is susceptible to attacks such as a related-key attack. eXtended TEA (XTEA) [47] was proposed to improve the security of TEA with a more complex key-schedule, shift rearrangement, XORs, and additions.

In 2011, Wenling and Zhang [58] proposed a light-weight block cipher called LBlock. LBlock uses 32 rounds on a 64-bit block cipher with an 80-bit encryption key length for encrypting plaintext. The LBlock round function uses SPN with small 4×4 S-boxes for the confusion layer and a 4-bit word permutation layer for diffusion. The 4-bit oriented structure of LBlock makes it optimal for software and hardware implementation. SIMON [59], proposed by Ray Beaulieu et al., is a 2n-bits FN-based block cipher with an encryption key length of nm-bits. The number of rounds for encryption can be varied from 32–72 rounds, depending on the block and encryption key size. In each round, simple operations such as bitwise XOR, AND, and circular shift are applied to the block. SIMON is vulnerable to attacks such as the key recovery attack [60] and differential fault attack [61].

In 2016, Bansod et al. [62] proposed PICO, a compact and ultra-light SPN-based block cipher, which uses a large number of active S-boxes in fewer rounds to provide protection against linear and differential attacks. Another SPN-based cipher called SKINNY [63] uses 64- or 128-bit block cipher with n, $2n$, and $3n$ (where n is size of block cipher) encryption key lengths. The number of rounds varies from 32–56 depending on the block and encryption key size. The S-box in SKINNY is designed to be compact, and a sparse diffusion layer with light key scheduling is used. SKINNY is vulnerable to attacks such as the differential faults attack [64] and impossible differential attack [65]. Li et al. [66] proposed QTL, a light-weight block cipher that improves the slow diffusion in traditional FN structures by changing all the block messages in each round. The fast diffusion of SPNs has improved the security of QTL; however, it is still not secure against differential and linear attacks [67].

In 2017, GIFT [68] was proposed to overcome the weakness of PRESENT. Based on PRESENT, it introduces permutation in conjunction with the Difference Distribution Table (DDT)/Linear Approximation Table (LAT) of the S-box [68]. GIFT is still vulnerable to attacks such as the related key attack [69], a side channel attack, and chosen plaintext attack [70]. Usman et al. [49] proposed the Secure IoT (SIT) algorithm for light-weight encrypted communication between IoT devices. SIT uses a mixture of FN and uniform SPN to encrypt a cipher block with a key size of 64 bits. SIT uses five-round encryption with five different keys. To improve the energy efficiency in each round, 4-bit mathematical functions are used to apply confusion and diffusion in the block cipher. LiCi [71], proposed by Patil et al., is a FN-based block cipher with a small footprint area that uses only 1153 GEs

(Gate Equivalents). The design of the key scheduling is inspired by PRESENT. It uses 4-bit S-boxes with operations such as XOR and left and right circular shift for data encryption.

In 2019, Sehrawat and Gill [50] proposed a FN-based block cypher called BRIGHT for resource-constrained IoT-enabled applications. It supports 64-bit and 128-bit block ciphers with an encryption key size ranging from 80 to 256 bits. It uses 32 to 37 rounds for encrypting a plaintext, depending on the cipher block and encryption key sizes. In each round, BRIGHT uses addition, rotation and XOR (ARX) operations, pre-key whitening, and a round of premutation to achieve fast diffusion and small code size. Aboshosha et. al. [72] proposed the Simple Light-weight Encryption Algorithm (SLEA), which uses compact S-boxes for achieving confusion and diffusion. It uses operands such as XOR, addition, and subtraction operations in each round to encrypt the data.

There are several recent research studies on light-weight cryptography algorithms. Thabit et al. performed a validation study on a new light-weight cryptographic algorithm (NLCA) [73] with general symmetric algorithms such RC4, HIGH, SF, AES, SIT, and DES for data transmission in cloud services. The results show that NLCA has the smallest execution encryption/decryption time and lower memory usage. However, there was no report on its performance on a sensor device with limited computational resources. Kumar V G et al. [74] proposed the BRISI light-weight algorithm, which is based on ARX (Addition-Modulo Rotation and XOR) operation, FN-structure, and a combination of BRIGHT and SIMON algorithms. Sophia et al. [75] proposed a fused secure and fuzzy combination framework over AES to improve privacy against unauthorized unknown access.

Many studies have focused on novel key generation methods. Mousavi et al. [76] proposed a hybrid model by using ABC (Artificial Bee Colony), ECC (Elliptic-Curve Cryptography), and SHA256 (Secure Hash Algorithm 256) to enhance data security in IoT applications. By generating a private key using an ABC genetic algorithm, encryption execution time was 52.31% lower than the RSA-AES method and total throughput could be increased over 50% both for encryption and decryption. Al-Husainy et al. [77] proposed a light-weight cryptography algorithm using the deoxyribonucleic acid (DNA) sequence for generating random encryption keys. This algorithm demonstrates better performance compared to AES in terms of execution time and immunity for statistical attacks. Sivasangari et al. [39] proposed the SPECG encryption algorithm, which generates a unique 256-bit encryption key based on ECG data and applies RC7 [78] for data encryption. Key generation using either a genetic algorithm or a DNA sequence can exhaustively consume computation resources and may not be suitable for on node implementation.

BRISK [46] is a FN-based block cipher that provides extra security with a dynamism concept. Designed for resource-constrained devices, instead of selecting a cipher from a large pool of ciphers [51], BRISK uses the same cipher with varying specifications selected from a small pool (e.g., number of rounds and cipher components) during encryption for each session. It uses a block size of 2n (where word size n is 16 bit) with an encryption key size of 80 bit. BRISK uses four S-Boxes with a size of 4-bit for confusion, and a 16-bit permutation layer for diffusion. The simple operations make it energy-efficient. However, relying on the small pool of specifications for encryption, an attacker might use a dictionary attack to achieve the necessary parameters to decrypt the transmitted messages.

Table 1 summarizes some of the existing light-weight block-based symmetric key encryption algorithms. Most of the existing encryption algorithms provide security for connection-oriented communication. To enhance the security of the network, a schedule for exchanging an encryption key is frequently required.

Table 1. Summary of some of the existing light-weight symmetric block cipher algorithms based on substitution–permutation network (SPN) and Feistel network (FN).

Encryption Algorithm	Tech	Key Size (bits)	Block Size (bits)	Number of Rounds	Possible Attacks
TEA [57], 1994	FN	128	64	32, 64	Equivalent keys attack [79]
XTEA [47], 1997	FN	128	64	32, 64	Related-key rectangle attack [80]
PRESENT [48], 2007	SPN	80, 128	64	31	Saturated, weak key, and linear attacks
LBlock [58], 2011	FN	80	64	32	Saturated attack [81]
SIMON [59], 2013	FN	64–256	32–128	32–72	Key recovery attack [60], differential fault attack [61]
PICO [62], 2016	SPN	128	64	32	-
SKINNY [63],2016	SPN	n, 2n, 3n	64, 128	32–56	Differential faults attacks [64], impossible differential attacks [65]
QTL [66], 2016	FN	64, 128	64	16, 20	Differential and linear cryptanalysis attack [67]
GIFT [68], 2017	SPN	128	64,128	28, 40	Related key attack [69], side channel attack, and chosen plaintext attack [70]
SIT [49], 2017	SPN, FN	64	64	5	-
LiCi [71], 2017	FN	128	64	31	-
BRIGHT [50],2019	FN	80–256	64, 128	32–37	-
SLEA [72], 2019	FN	2n	n	m	
NLCA [73], 2021	SPN, FN	128	128	10–20	-
SPECG [39], 2021	FN	256	256	0–255	-
BRISK [46], 2021	FN	80	2n	32	-

3. Dynamic Light-Weight Symmetric Encryption Algorithm

This section describes the proposed dynamic light-weight symmetric encryption algorithm and the parameters used for algorithm evaluation. The algorithm was designed for connectionless communication between sensor nodes and node gateways. The algorithm involves three types of keys, i.e., the master key, the node secret key, and the temporal encryption key. The master key and device information will be used to generate node secret keys for each device. Instead of scheduling encryption key exchange, a temporal encryption key can be dynamically generated from a node secret key, a counter, and a random number. Details of the proposed light-weight symmetric encryption algorithm are as follows.

3.1. Key Generation

Master key generation: a master key can be generated by passing a random long string through a hash function (e.g., MD5, SHA-1, or SHA-2), as shown in Figure 2. A master key can be shared among a group of sensors. The master key(s) must be kept in a secure place and only authorized people can have access to the key(s).

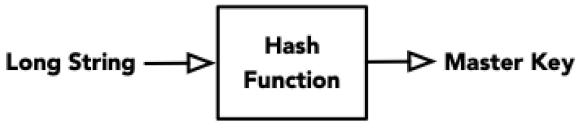

Figure 2. Master key generation.

Node secret key generation: Figure 3 shows an overview of the node secret key generation and management process. A node secret key for each sensor can then be obtained by applying a hash function on a string of a master key and a unique sensor's information (e.g., MAC address). The node secret key can be generated for each device and stored at the server for encryption key generation as messages arrive from each node. On the sensor side, the node secret key can be generated and preloaded onto the sensor node (e.g., embedded in sensor firmware) or periodically obtained from the server.

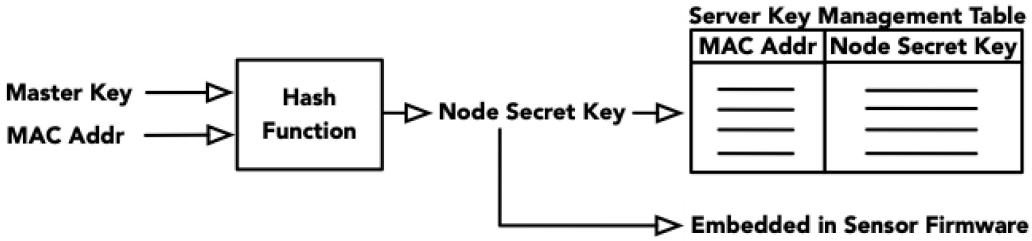

Figure 3. Node secret key generation.

Temporal encryption key generation: To prevent brute-force attacks and message replay, the proposed algorithm uses different encryption keys for ciphering consecutive messages. These "temporal" encryption keys are generated by applying a hash function on a concatenation of a node secret key, a random number, and a counter value, as depicted in Figure 4. Each message will be encrypted with different temporal encryption keys, which are changed with respect to the updated counter value. The hash function used for generating temporal encryption keys must use as little resources as possible. The possible combinations of temporal encryption keys depend on the number of bits used for counter representation. For instance, if a sensor broadcasts messages at a rate of 1 Hz, the temporal encryption keys will be reused after 12 and 194 days with 20-bit and 24-bit counters, respectively. Reuse of temporal encryption keys (i.e., after 12 or 194 days) is vulnerable to a key replay attack. Attackers could recover the temporal encryption key using pairs of known plaintexts and known ciphertexts and create a dictionary table of the temporal encryption keys associated with the counters. Implementing key management on the miniaturized sensor node can be computationally expensive. With constraints on power consumption, a light-weight mechanism such as random number exchange between the node and the server is used. The node random number can be updated and exchanged between the sensor and server before the exhaustion of the counter number, so that the temporal encryption key will never be reused.

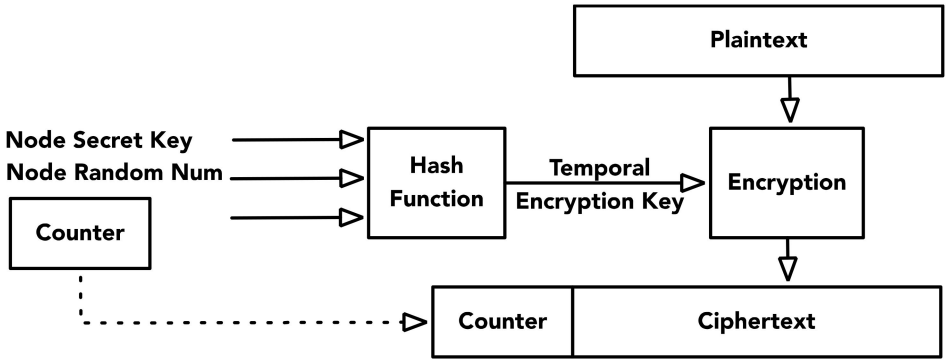

Figure 4. Temporal encryption key generation and message encryption.

3.2. Message Encryption/Decryption

The security of the proposed dynamic light-weight encryption algorithm relies on the generation of a unique encryption key for each message transmission. To address the resource constraint issue at the sensor node, a simple XOR function is used for data encryption at the sensor. The advantages of XOR encryption are (1) very low resource requirement and (2) suitability for different data sizes (no restriction on block size as in most block cipher symmetric algorithms). In addition, since the sensor may use only a part of a temporal encryption key to encrypt a message, it is more difficult for attackers to regenerate the keys and/or recover the messages. The sensor sends the counter value along with the ciphertext to a sensor gateway. The sensor gateway encapsulates the received information along with additional information (e.g., timestamp, RSSI, and sensor MAC address) and securely forward the packet to the server.

At the server, when an encrypted message is received, a sensor's specific information (i.e., MAC address) is used to look up the node secret key associated with the source node. The server uses the node secret key, the random number, and the counter value in the received packet to generate a temporal encryption key with a hash function. The temporal encryption key can then be used for decrypting the ciphertext. Figure 5 shows the message decryption process when a packet is received at a server.

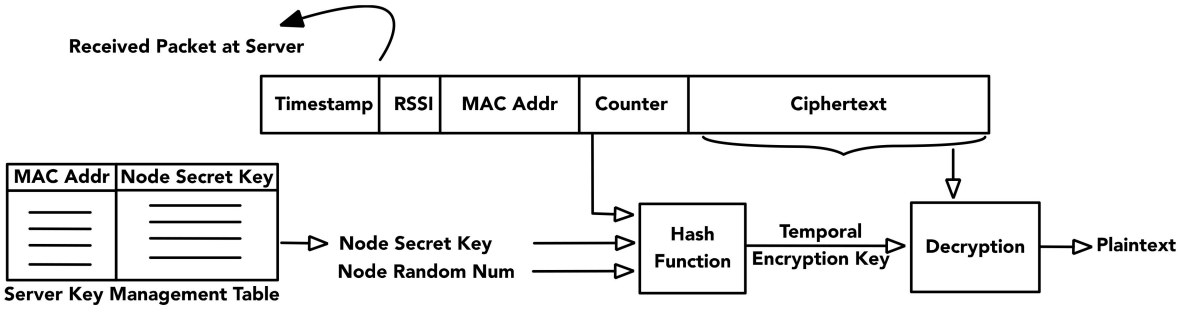

Figure 5. Server/cloud decryption process.

3.3. Evaluation Parameters

To evaluate the performance of the proposed DLS encryption algorithm, the following security criteria and performance metrics are considered:

3.3.1. Security Criteria

- Avalanche effect: in cryptography, the avalanche effect is a desirable property. It refers to changes in the output of the algorithm (ciphertext) when a small change is made (i.e., flipping a single bit) either on an encryption key or a plaintext. The avalanche effect can be measured using Equation (1). A good cipher should always have at least 50%

change in ciphertext bits with even a one-bit change in a plaintext or an encryption key [82]. This implies that the encryption algorithm can resist most possible attacks.

$$Avalanche\ Effect\ (AE) = \frac{Number\ of\ changed\ bits\ in\ ciphertext}{Number\ of\ bits\ in\ ciphertext} \times 100 \quad (1)$$

- Randomness: a randomness test can be used to determine the randomization in ciphertexts based on their diffusion characteristics. To pass the randomness test, there should not be any form of input/output relation in the cipher. The distribution of the number of zeros and ones in ciphertexts can be observed to check the randomness in ciphertexts.

3.3.2. Performance Metrics

- Memory consumption (byte): more memory used by an application running on a sensor usually leads to higher power consumption. In this study, memory usage is measured in terms of code size and RAM in use for encrypting a plaintext.
- Execution time (second): the time required to encrypt a plaintext. Longer execution time implies higher power consumption and a delay in data transmission. In a real-time system with a low data latency requirement, encryption with lower execution time is preferred.
- Power consumption (mV/hr): a sensor with low power consumption has a longer battery-life and does not require frequent recharging.

4. Experiments and Results

4.1. Experiment Setup

The proposed DLS encryption algorithm adds additional security on top of existing light-weight encryption algorithms and does not change the core functions of the encryption algorithms used. To evaluate the performance of the proposed DLS encryption algorithm over light-weight encryption algorithms, we implemented the proposed DLS encryption algorithms using TEA, XTEA, PRESENT, and MD5 as hash functions for the temporal encryption key generation step. To generate temporal encryption keys, we used a 128-bit node secret key and a 64-bit input generated based on a concatenation of the sensor MAC address and a 16-bit counter, as shown in Figure 6. Seven algorithms were compared, namely, (1) PRESENT, (2) XTEA, (3) TEA, (4) DLS over PRESENT (DLS-PRESENT), (5) DLS over XTEA (DLS-XTEA), (6) DLS over TEA (DLS-TEA), and (7) DLS over MD5 (DLS-MD5). All the encryption algorithms were implemented on a 64-bit processor in C language. The experiments were performed on a MacBook Air with a 1.7 GHz dual-core Intel Core i7 processor and 4 GB of RAM.

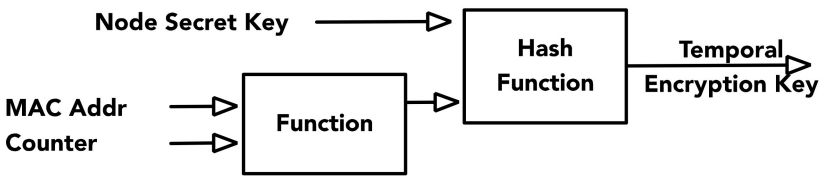

Figure 6. Temporal encryption key generation using a symmetric encryption algorithm (i.e., XTEA).

In another experimental setting, the TEA, XTEA, and DLS over XTEA (DLS-XTEA) algorithms were implemented on the AiR (Autonomous intelligent Recognition) node [83] for power consumption evaluation. The AiR node, as shown in Figure 7, is a miniaturized wearable activity and behavior monitoring sensor for elderly care, fall prevention, and chronic disease management. The sensor is embedded with a motion sensor, a 70 mAhr Lithium-ion battery, and the nRF51822 chipset, which supports the use of Bluetooth Low Energy (BLE) for wireless communication. In our study, the AiR node performed on-node processing and broadcast a 15-byte data packet via advertising channels at a rate of 0.5 Hz.

The data packet contained 48 bits of inferred user information, a 16-bit counter, and zero-paddings. Since the advertising channel is not designed for sensitive data transmission, an encryption algorithm was applied for data protection. To measure power consumption, the on-board function was used to obtain the voltage level. The same AiR node was fully charged and kept in the controlled environment for all the experiments. To ensure a similar starting voltage level, data collection started when the battery voltage was reduced to 4.1 volts and left for 48 h before the battery voltage level was reassessed. A fixed secret key of 128 bits was used for encrypting a 64-bit plaintext. For smaller data sizes, the input requires padding with random numbers. In our study, the 48-bit data were padded with random numbers to make a plaintext of 64 bits.

Figure 7. AiR node for monitoring activity and behavior parameters of patients.

4.2. Experimental Results

4.2.1. Security Criteria

In this section, we evaluate if the seven algorithms pass the strict avalanche criterion, i.e., plaintext sensitivity test, key sensitivity test, and randomness test.

Plaintext Sensitivity Test: Table 2 demonstrates a specific example of how the avalanche effect is measured when a one-bit change is made on a plaintext. The seven encryption algorithms were applied using a 128-bit node secret key on a plaintext and its one-bit modified version. The number of bits changed between the two ciphertexts were calculated by the sum of their XOR result. For each algorithm, the process was repeated over 50,000 random plaintexts. Table 3 shows a summary of average number of bits changed and avalanche effect due to a one-bit change over all the random plaintexts (The 128-bit secret encryption key is fixed to 0×000041A7 10D63AF1 60B7ACD9 3AB50C2A). The results show that approximately half of the ciphertext bits are changed and ~50% avalanche effect values are obtained on average for all algorithms.

Key Sensitivity Test: An algorithm is key sensitive if retrieving a plaintext is not possible even with a one-bit change in the original encryption key. To measure key sensitivity, the avalanche test is used to observe the changes in ciphertext [50]. Table 4 demonstrates a specific example of how the avalanche effect is measured when a one-bit change is made on the encryption key. Table 5 shows the summary of the average number of bits changed and the avalanche effect due to a one-bit change over all the random encryption keys (The plaintext is fixed to 0×00000000 00000000). The results show that approximately half of the ciphertext bits are changed and ~50% avalanche effect values are obtained on average for all algorithms.

Table 2. Avalanche effect calculated when a one-bit change is made on a specific plaintext.

Encryption Algorithm	Plaintext	Ciphertext	Bits Changed	Avalanche Effect
PRESENT	0×00000000 4431B782 0×00000000 4431B783	0×1089e85d 077ed1be 0×4c07946e f36dac79	36	56.25%
XTEA		0×C0037A37 6BC98019 0×4E0F748A 96239243	33	51.56%
TEA		0×DF6F3D7E 722BCE66 0×B1218894 9A57CB71	34	53.12%
DLS-PRESENT		0×efbc5f36 d12fc516 0×d486981b 2836795c	35	54.68%
DLS-XTEA		0×6AB29657 E7DBE6C5 0×168FF88F 16114132	39	60.00%
DLS-TEA		0×8cefa414 fa32ba72 0×0548e8bc d1b1505e	29	45.31%
DLS-MD5		0×c665c7f4 d0f0993a 0×f39b4a36 9ef7d0cd	35	54.68%

Table 3. A summary of average number of bits changed and avalanche effect measured over 50,000 random plaintexts.

Encryption Algorithm	Average Bits Changed	Average Avalanche Effect
PRESENT	32.01	50.01%
XTEA	32.00	50.00%
TEA	31.96	49.95%
DLS-PRESENT	31.98	49.98%
DLS-XTEA	32.01	50.01%
DLS-TEA	32.00	50.00%
DLS-MD5	31.98	49.97%

Table 4. Avalanche effect calculated when a one-bit change is made on a specific encryption key.

Encryption Algorithm	Encryption Key	Ciphertext	Bits Changed	Avalanche Effect
PRESENT	0×000041A7 10D63AF1 60B7ACD9 4431B782	0×a6096f79 60a39158 0×cbb6a496 074871c0	41	64.06%
XTEA		0×322E8C81 A941DA47 0×47892894 E99202F3	37	46.87%
TEA		0×E7239A15 7C4601DD 0×95AB191E CCD20FC3	25	39.06%
DLS-PRESENT	0×000041A7 10D63AF1 60B7ACD9 4431B783	0×5611add9 e3cf63ec 0×00af538 ddeb18660	39	60.93%
DLS-XTEA		0×9fe73cda 5046966c 0×f69477c6 27cc670b	35	54.68%
DLS-TEA		0×3dda8056 4c345547 0×abb8187c 2aa15d37	25	39.06%
DLS-MD5		0×fe3ba2d6 43541e23 0×6b05a37d 40778e7b	25	39.06%

Table 5. A summary of the average number of bits changed and the avalanche effect measured over 50,000 random encryption keys.

Encryption Algorithm	Average Bits Changed	Average Avalanche Effect
PRESENT	32.03	50.04%
XTEA	32.01	50.01%
TEA	32.00	50.01%
DLS-PRESENT	31.97	49.96%
DLS-XTEA	31.97	49.95%
DLS-TEA	31.99	49.98%
DLS-MD5	31.97	49.96%

Randomness Test: Table 6 shows the average number of ones and zeros resulting from the ciphertexts generated during the plaintext sensitivity and key sensitivity experiments. The average number of zeros and ones for a cipher block of 64 bits in each experiment was approximately ~32 bits (i.e., ~50% chance of occurrences for both zeros and ones) for all seven algorithms. The results indicate that there is no pattern (unpredictable) in the ciphertexts. Therefore, even with simple XOR encryption with temporal encryption keys, DLS will generate very secure ciphertext.

Table 6. Average number of zeros and ones in 64-bit ciphertext block when there is a single-bit change in plaintext (plaintext sensitivity) and key (key sensitivity).

Encryption Algorithm	Plaintext Sensitivity Average Number of		Key Sensitivity Average Number of	
	Zeros	Ones	Zeros	Ones
PRESENT	31.97	32.02	32.02	31.97
XTEA	31.97	32.02	31.98	32.01
TEA	31.99	32.00	31.97	32.02
DLS-PRESENT	31.99	32.00	31.99	32.00
DLS-XTEA	31.99	32.00	32.01	31.98
DLS-TEA	32.00	31.99	32.00	31.99
DLS-MD5	31.97	32.02	32.01	31.98

4.2.2. Resource Consumption

Figure 8a,b show memory consumption (code size and RAM), and execution time during the encryption process, respectively. The results show that the TEA encryption algorithm uses the lowest resources, while DLS-MD5 uses the highest memory resources compared to other algorithms. Applying DLS over TEA, XTEA, and PRESENT adds on average an additional ~694 bytes to the code size, ~2% more RAM, and ~3% longer execution time compared to the original encrypting algorithms.

4.2.3. Power Consumption

From the above results, we observe that (1) DLS-XTEA adds only slight overhead compared to TEA in a trade of higher security, (2) DLS-MD5 uses 2.27 times more memory resources and has a 2.37-times longer execution time compared to DLS-XTEA, and (3) DLS-PRESENT uses 1.6 times more memory resources and has a 13.5-times longer execution time. Taking the data latency, security, and memory constraints into consideration, DLS-XTEA appears to be a suitable candidate for enhancing security in pervasive sensing applications and was explored further for power consumption evaluation. Table 7 shows the power consumption required for encrypting a plaintext on an AiR node. The results show that DLS-XTEA uses approximately 8.83% and 5.66% more battery power compared to the TEA and XTEA algorithms, respectively. The AiR node with DLS-XTEA implementation can continuously broadcast data at a rate of 0.5 Hz for 7 days without recharging.

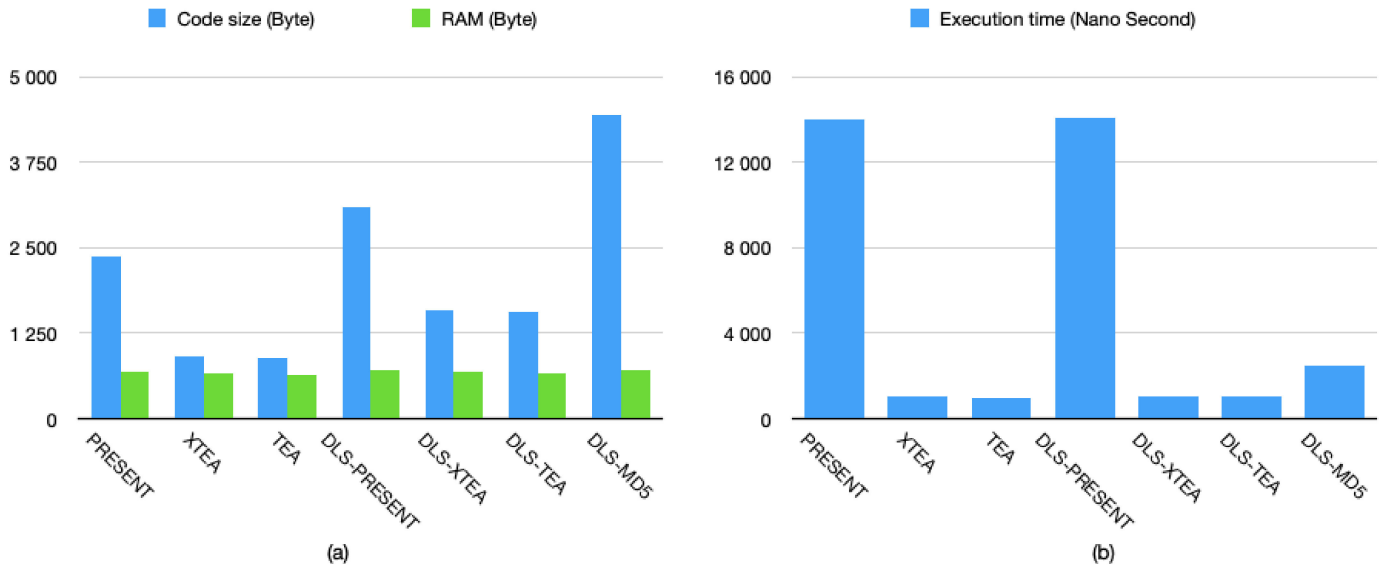

Figure 8. The average resource consumption for different encryption algorithms. (**a**) Code size and memory, and (**b**) Execution time.

Table 7. The power consumption on an AiR node for different encryption algorithms.

Encryption Scheme	DLS-XTEA	TEA	XTEA
Power consumption (mV/hr)	6.16	5.66	5.83

4.3. Cryptanalysis

In this section, we will analyze the strength of the proposed DLS encryption algorithms against different types of attacks.

Brute-force attack (exhaustive search): this type of attack checks all possible keys until the correct key is founded. In symmetric encryption algorithms, a secret key size will determine the feasibility of this attack. The number of attacks for a key with a length of k will be 2^k to break the ciphertext with the algorithm complexity of $O\left(2^k\right)$. Our proposed DLS encryption algorithm uses temporal encryption keys for encrypting messages. The temporal encryption key is generated by using a random number and a counter. Since the random number of length r is secret, the brute-force attack for DLS encryption algorithm with the encryption key of size k will have the complexity of $O\left(2^{rk}\right)$. The brute-force attack must finish within a short period of time (repeated counter duration).

Replay attack: the attacker delays or reuses the same messages transmitted to a receiver; since messages are correctly encrypted, the receiver might treat the messages as correct requests. Our proposed DLS can detect and prevent replay attacks. Since the DLS encryption algorithm uses a unique temporal encryption key for encrypting each transmitted message, and each message include a timestamp and a checksum, the receiver can detect and prevent the replay attack using the information.

Differential attack: when ciphertexts show non-random behavior, attackers can find the relationship between plaintexts and ciphertexts to recover the encryption key. The experiment result in Section 4 shows that our proposed DLS encryption algorithm achieves ~50% SAC on average and the distribution of zeros and ones on ciphertext is ~50%. Since our proposed algorithm uses a different temporal encryption key to encrypt each message, it would be difficult to recover the relationship between plaintext and ciphertext.

Eavesdropping: the rate of message transmission and the size of c determines the time period attackers require to collect all the ciphertext from a specific node for further analysis. With the specific node secret key (and node random number), the strength of the

encryption algorithm is further enhanced, as collecting messages from a different node will not directly contribute to the analysis of the secret key of another node.

Data modification/data corruption: when a bit in the ciphertext is modified, ~50% of bits in the decrypted message will also be modified and make the decrypted message uninterpretable. Malicious attackers may also modify counter values in the packet, causing synchronization problems. The server can detect data modification/data corruption when the decrypted message falls out of the expected pattern. Prevention of such attacks, however, needs to be handled with other security schemes (e.g., adding authentication to the communication).

Known plaintext attack: an attack model where both the plaintext and ciphertext are known to attackers [84]. DLS uses different temporal encryption keys per message, which are both secret and random. As a result, encrypting a number of the same plaintext will give different unpredictable ciphertext every time. Even if the attacker can recognize the temporal encryption key, only one key is compromised. It is very difficult to work backwards to retrieve the random numbers and node secret keys.

Denial of Service (DoS): An attack model where attackers transmit traffic to overload a machine or network to make services unavailable for legitimate users. In BSNs, wireless sensors normally have a short range of communication (a few meters) to the sensor gateway. The other parts of network, such as connections between gateways and the server, are protected with standard security techniques, which are not within the scope of this study. If the server detects a problematic sensor—for example, the sensor's transmission rate being too high—the specific sensor can be blocked at the gateway where the attack is detected. Unless the messages are from the same gateway, which means the attacker is in the vicinity of the sensor node, the original node will not be affected.

Secret key generation in DLS is designed to have two levels of security. The node secret key is fixed and embedded in the firmware. Physical security can be achieved by sealing the circuit with some coating material. This makes the node secret key extremely difficult to retrieve, since when the attacker opens the coating, the circuit will also be destroyed. The random number, on the other hand, is changed periodically. The random number is a part of the key dynamic that helps to prevent dictionary attack and replay attack of the secret key. If the random number is compromised, the node secret key still provides another layer of security.

In the proposed DLS algorithm, the level of security can also be adjusted as needed by changing the numbers of node random number bits and counter bits. To achieve higher security, node random number bits can be increased and counter bits can be reduced (which causes the exchange of random numbers with the server to avoid repeated use of keys and counter combinations for data encryption). This increases the total bits of secret key (node secret key bits + node random bits) and decreases the time duration opportunity, determined by the number of counter bits, to break the key. To compromise the security of the DLS algorithm, attackers need to recover the random number as well as the node secret key.

In dynamic key cryptography, each dynamic key is used to encrypt only one message. This makes it extremely difficult to analyze encrypted messages for common patterns to break the cryptography. However, in the case that the same known plaintext is repeatedly sent, attackers have a higher chance to derive dynamic encryption keys and eventually use a dictionary attack to derive the secret key. Once the secret key is known, attackers will be able to derive the next dynamic encryption keys in the sequence and break the cryptographic system.

The length of the counter and the speed of the message transmission determines how fast the dynamic encryption keys in the sequence are used up. As in other dynamic key cryptography, the risk of hijacking is higher when the counter is repeated. Therefore, the secret key needs to be updated, for example, by exchanging the node random number with the server regularly to avoid reusing dynamic encryption keys. Secure key exchange with the server to generate a new dynamic key sequence can incur considerable overhead and

power consumption. A possible trick is to perform key exchange between a sensor node and a server during the charging period. The key exchange with the server can either be implemented on the charging device and the key transferred to the sensor node during charging, or charging is used to trigger the key exchange (during charging the node should have sufficient power to complete the task).

5. Conclusions and Discussion

In this paper, we proposed a Dynamic Light-weight Symmetric (DLS) encryption algorithm for secure data transmission via BLE beacon. We implemented our proposed algorithm on AiR nodes that utilize an advertising channel to transmit the on-node processed data to the cloud server. The proposed algorithm can use any light-weight hash function or light-weight encryption algorithm to generate temporal encryption keys for plaintext encryption on the sensor node. The temporal encryption keys are dynamically generated from the node secret key, the counter, and a node random number. The number of bits in the random number and counter can be adjusted to obtain the needed security level. The random number can be exchanged between the node and the server periodically to avoid reusing temporal encryption keys. In addition, we demonstrated the use of a light-weight encryption algorithm as a hash function instead of a standard hashing algorithm (such as MD5) to further optimize computational resources. The experiment results show that the proposed DLS algorithm over existing light-weight hash functions or symmetric encryption algorithms is able to satisfy the test for strict avalanche criterion (i.e., ~50%), the plaintext sensitivity test, key sensitivity test, and randomness test with a marginal increase in resource usage. Applying DLS over TEA, XTEA, or PRESENT adds an average of ~694 bytes to code size, uses ~2% more RAM, and requires a ~3% longer execution time compared to the original encrypting algorithms. Experimental results on an AiR node show that DLS-XTEA uses approximately 8.83% and 5.66% more battery power compared to the TEA and XTEA algorithms, respectively. It only adds a small amount of overhead to the existing base encryption algorithms to effectively protect the data against attacks such as brute-force attack, replay attack, differential attack, and known plaintext attack. The proposed encryption algorithm is light, energy-efficient, and suitable for implementation at the application layer to achieve a significantly higher level of security for pervasive sensing applications.

Author Contributions: K.W. conceived the main algorithm design. S.B. and N.O. implemented and carried out the experiments. S.B. and S.T. designed the experiments, analyzed the results, and wrote the paper. S.B., S.T. and K.W. edited the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in parts by a National Science and Technology Development Agency (NSTDA) grant ID: P1951069, a Newton Fund Institutional Links grant ID:330760239 and 527636592, under the Newton-Thailand Research Fund (TRF) and the Office of National Higher Education Science Research and Innovation Policy Council partnership. The Newton Fund Institutional Links grant is funded by the UK Department of Business, Energy and Industrial Strategy (BEIS) and TRF and delivered by the British Council. For further information, please visit www.newtonfund.ac.uk (accessed on 24 December 2021).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data underlying this article will be shared on reasonable request from the corresponding author.

Acknowledgments: We thank Sarocha Paopas for her support in measuring the power consumption technique.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mundt, C.W.; Montgomery, K.N.; Udoh, U.E.; Barker, V.N.; Thonier, G.C.; Tellier, A.M.; Ricks, R.D.; Darling, R.B.; Cagle, Y.D.; Cabrol, N.A.; et al. A Multiparameter Wearable Physiologic Monitoring System for Space and Terrestrial Applications. *IEEE Trans. Inf. Technol. Biomed.* **2005**, *9*, 382–391. [CrossRef] [PubMed]
2. Sung, M.; Marci, C.; Pentland, A. Wearable Feedback Systems for Rehabilitation. *J. NeuroEngineering Rehabil.* **2005**, *2*, 17. [CrossRef] [PubMed]
3. Oliver, N.; Flores-Mangas, F. HealthGear: A Real-Time Wearable System for Monitoring and Analyzing Physiological Signals. In Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks, Cambridge, MA, USA, 3–5 April 2006.
4. Pop, V.; Francisco, R.D.; Pflug, H.; Santana, J.; Visser, H.; Vullers, R.; De Groot, H.; Gyselinckx, B. Human++: Wireless Autonomous Sensor Technology for Body Area Networks. In Proceedings of the Asia and South Pacific Design Automation Conference, Yokohama, Japan, 25–28 January 2011; pp. 561–566.
5. Caldara, M.; Colleoni, C.; Guido, E.; Rosace, G.; Re, V.; Vitali, A. A Wearable Sensor Platform to Monitor Sweat pH and Skin Temperature. In Proceedings of the International Conference on Body Sensor Networks, Cambridge, MA, USA, 6–9 May 2013; pp. 1–6.
6. Kirk, M.A.; Amiri, M.; Pirbaglou, M.; Ritvo, P. Wearable Technology and Physical Activity Behavior Change in Adults with Chronic Cardiometabolic Disease: A Systematic Review and Meta-Analysis. *Am. J. Health Promot.* **2018**, *33*, 1–14. [CrossRef] [PubMed]
7. Wang, G.; Zhang, S.; Dong, S.; Lou, D.; Ma, L.; Pei, X.; Xu, H.; Farooq, U.; Guo, W. Stretchable Optical Sensing Patch System Integrated Heart Rate, Pulse Oxygen Saturation, and Sweat pH Detection. *IEEE Trans. Biomed. Eng.* **2018**, *66*, 1000–1005. [CrossRef]
8. Fortino, G.; Gravina, R. Fall-MobileGuard: A Smart Real-Time Fall Detection System. In Proceedings of the EAI International Conference on Body Area Networks, Sydney, Australia, 28–30 September 2015.
9. Mozaffari, N.; Rezazadeh, J.; Farahbakhsh, R.; Yazdani, S.; Sandrasegaran, K. Practical Fall Detection Based on IoT Technologies: A Survey. *Internet Things* **2019**, *8*, 100124. [CrossRef]
10. Ramachandran, A.; Karuppiah, A. A Survey on Recent Advances in Wearable Fall Detection Systems. *BioMed Res. Int.* **2020**, *2020*, 2167160. [CrossRef]
11. Manimaraboopathy, M.; Vijayalakshmi, S.; Hemavathy, D.; Priya, A. A Wearable Multiparameter Medical Monitoring and Alert System with First Aid. *Int. J. Smart Sens. Intell. Syst.* **2017**, *10*, 446–458. [CrossRef]
12. Garbern, S.C.; Mbanjumucyo, G.; Umuhoza, C.; Sharma, V.K.; Mackey, J.; Tang, O.; Martin, K.D.; Twagirumukiza, F.R.; Rosman, S.L.; McCall, N.; et al. Validation of a Wearable Biosensor Device for Vital Sign Monitoring in Septic Emergency Department Patients in Rwanda. *Digit. Health* **2019**, *5*, 1–13. [CrossRef]
13. Wu, R.C.; Ginsburg, S.; Son, T.; Gershon, A.S. Using Wearables and Self-Management Apps in Patients with COPD: A Qualitative Study. *Eur. Respir. J.* **2019**, *5*, 00036-2019. [CrossRef] [PubMed]
14. Jovanov, E.; Milenkovic, A.; Otto, C.; De Groen, P. A Wireless Body Area Network of Intelligent Motion Sensors for Computer Assisted Physical Rehabilitation. *J. NeuroEngineering Rehabil.* **2005**, *2*, 6. [CrossRef]
15. Bavan, L.; Surmacz, K.; Beard, D.; Mellon, S.; Rees, J. Adherence Monitoring of Rehabilitation Exercise with Inertial Sensors: A Clinical Validation Study. *Gait Posture* **2019**, *70*, 211–217. [CrossRef] [PubMed]
16. Herrera-Luna, I.; Rechy-Ramirez, E.J.; Rios-Figueroa, H.V.; Marin-Hernandez, A. Sensor Fusion Used in Applications for Hand Rehabilitation: A Systematic Review. *IEEE Sens. J.* **2019**, *19*, 3581–3592. [CrossRef]
17. Qiu, S.; Wang, Z.; Zhao, H.; Liu, L.; Wang, J.; Li, J. Gait Analysis for Physical Rehabilitation via Body-Worn Sensors and Multi-information Fusion: Technology, Communications and Computing. In *Advances in Body Area Networks*; Springer: Cham, Switzerland, 2019; pp. 139–148. [CrossRef]
18. Patel, S.; Park, H.; Bonato, P.; Chan, L.; Rodgers, M. A Review of Wearable Sensors and Systems with Application in Rehabilitation. *J. NeuroEngineering Rehabil.* **2012**, *9*, 21. [CrossRef]
19. Shin, G.; Jarrahi, M.H.; Fei, Y.; Karami, A.; Gafinowitz, N.; Byun, A.; Lu, X. Wearable Activity Trackers, Accuracy, Adoption, Acceptance and Health Impact: A Systematic Literature Review. *J. Biomed. Inform.* **2019**, *93*, 103153. [CrossRef] [PubMed]
20. Lo, B.P.; Atallah, L.; Crewther, B.; Spehar-Deleze, A.M.; Anastasova, S.; West, A.A.; Conway, P.; Cook, C.; Drawer, S.; Vadgama, P.; et al. Pervasive Sensing for Athletic Training. In *Delivering London 2012: ICT Enabling Game*; Institution of Engineering and Technology: London, UK, 2011. [CrossRef]
21. Aroganam, G.; Manivannan, N.; Harrison, D. Review on Wearable Technology Sensors Used in Consumer Sport Applications. *Sensors* **2019**, *19*, 1983. [CrossRef] [PubMed]
22. Sperlich, B.; Aminian, K.; Düking, P.; Holmberg, H.C. Editorial: Wearable Sensor Technology for Monitoring Training Load and Health in the Athletic Population. *Front. Physiol.* **2020**, *10*, 1520. [CrossRef] [PubMed]
23. Aileni, R.M.; Suciu, G.; Rajagopal, M.; Pasca, S.; Valderrama Sukuyama, C.A. *Data Privacy and Security for IoMWT (Internet of Medical Wearable Things) Cloud*; Springer: Cham, Switzerland, 2020. [CrossRef]
24. Kompara, M.; Hölbl, M. Survey on Security in Intra-Body Area Network Communication. *Ad Hoc Netw.* **2018**, *70*, 23–43. [CrossRef]

25. Tan, C.C.; Wang, H.; Zhong, S.; Li, Q. IBE-Lite: A Lightweight Identity-Based Cryptography for Body Sensor Networks. *IEEE Trans. Inf. Technol. Biomed.* **2009**, *13*, 926–932. [CrossRef] [PubMed]
26. Li, M.; Lou, W.; Ren, K. Data Security and Privacy in Wireless Body Area Networks. *IEEE Wirel. Commun.* **2010**, *17*, 51–58. [CrossRef]
27. Liu, J.; Kwak, S.K. Hybrid Security Mechanisms for Wireless Body Area Networks. In Proceedings of the International Conference on Ubiquitous and Future Networks, Jeju, South Korea, 16–18 June 2010; pp. 98–103.
28. Huang, C.; Lee, H.; Lee, D.H. A Privacy-Strengthened Scheme for E-Healthcare Monitoring System. *J. Med. Syst.* **2012**, *36*, 2959–2971. [CrossRef]
29. Al Ameen, M.; Liu, J.; Kwak, S.K. Security and Privacy Issues in Wireless Sensor Networks for Healthcare Applications. *J. Med. Syst.* **2012**, *36*, 93–101. [CrossRef] [PubMed]
30. Ali, A.; Khan, F.A. Key Agreement Schemes in Wireless Body Area Networks: Taxonomy and State-of-the-Art. *J. Med. Syst.* **2015**, *39*, 115. [CrossRef] [PubMed]
31. Yao, L.; Liu, B.; Yao, K.; Wu, G.; Wang, J. An ECG-Based Signal Key Establishment Protocol in Body Area Networks. In Proceedings of the International Conference on Ubiquitous Intelligence and Computing and International Conference on Autonomic and Trusted Computing, Xian, China, 26–29 October 2010; pp. 233–238.
32. Drira, W.; Renault, E.; Zeghlache, D. A Hybrid Authentication and Key Establishment Scheme for WBAN. In Proceedings of the International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, UK, 25–27 June 2012; pp. 78–83.
33. Pathan, A.S.K.; Lee, H.W.; Hong, C.S. Security in Wireless Sensor Networks: Issues and Challenges. In Proceedings of the International Conference Advanced Communication Technology, Phoenix Park, South Korea, 20–22 February 2006.
34. Zhang, Z.; Cho, M.C.Y.; Wang, C.; Hsu, C.; Chen, C.; Shieh, S. IoT Security: Ongoing Challenges and Research Opportunities. In Proceedings of the Service-Oriented Computing and Applications, Matsue, Japan, 17–19 November 2014; pp. 230–234.
35. Guan, Z.; Yang, T.; Du, X.; Guizani, M. Secure Data Access for Wireless Body Sensor Networks. In Proceedings of the Wireless Communications and Networking, Doha, Qatar, 3–6 April 2016.
36. Fortino, G.; Russo, W.; Savaglio, C.; Viroli, M.; Zhou, M. Modeling Opportunistic IoT Services in Open IoT Ecosystems. In Proceedings of the Workshop from Objects to Agents (WOA), Catania, Italy, 29–30 July 2017; pp. 90–95.
37. Bordel, B.; Alcarria, R.; De Andres, D.M.; You, I.; Martin, D. Securing Internet-of-Things Systems Through Implicit and Explicit Reputation Models. *IEEE Access* **2018**, *6*, 47472–47488. [CrossRef]
38. Frustaci, M.; Pace, P.; Aloi, G.; Fortino, G. Evaluating Critical Security Issues of the IoT World: Present and Future Challenge. *IEEE Internet Things J.* **2018**, *5*, 2483–2495. [CrossRef]
39. Sivasangari, A.; Ananthi, A.; Deepa, D.; Rajesh, G.; Mercilin Raajini, X. *Chapter 3-Security and Privacy in Wireless Body Sensor Networks Using Lightweight Cryptography Scheme*; Academic Press: Cambridge, MA, USA, 2021. [CrossRef]
40. Botta, M.; Simek, M.; Mitton, N. Comparison of Hardware and Software-Based Encryption for Secure Communication in Wireless Sensor Networks. In Proceedings of the International Conference on Telecommunications and Signal Processing, Rome, Italy, 2–4 July 2013; pp. 6–10.
41. Zidek, A.; Tailor, S.; Harle, R. Bellrock: Anonymous Proximity Beacons From Personal Devices. In Proceedings of the Pervasive Computing and Communications, Athens, Greece, 19–23 March 2018.
42. Martin, P.D.; Rushanan, M.; Tantillo, T.; Lehmann, C.U.; Rubin, A.D. Applications of Secure Location Sensing in Healthcare. In Proceedings of the Bioinformatics Computational Biology, and Health Informatics, Seattle, WA, USA, 2–5 October 2016; pp. 58–67.
43. Khazanie, N.; Matias, Y. Growing Eddystone with Ephemeral Identifiers: A Privacy Aware & Secure Open Beacon Format. Available online: https://security.googleblog.com/2016/04/growing-eddystone-with-ephemeral-identifiers.html (accessed on 26 September 2021).
44. Lee, J.P.; Lee, J.G.; Lee, J.H.; Yoon, K.; Lee, J.K. Design of a Secure Disaster Notification System Using the Smartphone Based Beacon. In Proceedings of the Networks and Communications, Sydney, Australia, 23–24 December 2016; pp. 1–12.
45. Ziobro, A. Overview-Kontakt.io Secure for Beacons with Firmware 4.0+ (Firmware Preceeding the 1.x and 2.x from 2019 and 2020). Available online: https://support.kontakt.io/hc/en-gb/articles/206762009-Kontakt-io-Secure-Shuffling (accessed on 26 September 2021).
46. Dwivedi, A.D. BRISK: Dynamic Encryption Based Cipher for Long Term Security. *Sensors* **2021**, *21*, 5744. [CrossRef]
47. Needham, R.M.; Wheeler, D.J. *TEA Extensions, Technical Report*; Computer Laboratory, University of Cambridge: Cambridge, UK, 1997.
48. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.B.; Seurin, Y.; Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Vienna, Austria, 10–13 September 2007; pp. 450–466.
49. Usman, M.; Ahmed, I.; Aslam, M.I.; Khan, S.; Shah, U.A. SIT: A Lightweight Encryption Algorithm for Secure Internet of Things. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 402–411. [CrossRef]
50. Sehrawat, D.; Gill, N.S. Performance Evaluation of Newly Proposed Lightweight Cipher, BRIGHT. *Int. J. Intell. Eng. Syst.* **2019**, *12*, 71–80. [CrossRef]
51. Knudsen, L.R. Dynamic Encryption. *Cyber Secur. Mobil.* **2015**, *3*, 357–370. [CrossRef]
52. Lindh, J. *Bluetooth Low Energy Beacons*; Texas Instruments Incorporated: Dallas, TX, USA, 2015.

53. Rajesh, S.; Paul, V.; Menon, V.G.; Khosravi, M.R. A Secure and Efficient Lightweight Symmetric Encryption Scheme for Transfer of Text Files between Embedded IoT Devices. *Symmetry* **2019**, *11*, 293. [CrossRef]
54. Arora, R.; Parashar, A. Secure User Data in Cloud Computing Using Encryption Algorithms. *Int. J. Eng. Res. Appl.* **2013**, *3*, 1922–1926.
55. Ahmad, S.; Alam, K.M.R.; Rahman, H.; Tamura, S. A Comparison between Symmetric and Asymmetric Key Encryption Algorithm based Decryption Mixnets. In Proceedings of the Networking Systems and Security, Dhaka, Bangladesh, 5–7 January 2015; pp. 1–5.
56. Kumar, P.; Rawat, S.; Choudhury, T.; Pradhan, S. A Performance Based Comparison of Various Symmetric Cryptographic Algorithms in Run-Time Scenario. In Proceedings of the System Modeling Advancement in Research Trends, Moradabad, India, 25–27 November 2016; pp. 37–41.
57. Wheeler, D.J.; Needham, R.M. TEA, a Tiny Encryption Algorithm. In Proceedings of the International Workshop on Fast Software Encryption, Leuven, Belgium, 14–16 December 1995; pp. 363–366.
58. Wenling, W.; Zhang, L. LBlock: A Lightweight Block Cipher. In Proceedings of the Applied Cryptography and Network Security, Nerja, Spain, 7–10 June 2011; pp. 327–344.
59. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK Families of Lightweight Block Ciphers. In Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 7–11 March 2013.
60. Ashur, T. Improved Linear Trails for the Block Cipher Simon. *IACR Crypto* **2015**, *2015*, 285.
61. Tupsamudre, H.; Bisht, S.; Mukhopadhyay, D. Differential Fault Analysis on the Families of SIMON and SPECK Ciphers. In Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography, Busan, Korea, 23 September 2014; pp. 40–48.
62. Bansod, G.; Pisharoty, N.; Patil, A. PICO: An Ultra Lightweight and Low Power Encryption Design for Ubiquitous Computing. *Def. Sci. J.* **2016**, *66*, 259–265. [CrossRef]
63. Beierle, C.; Jean, J.; Kölbl, S.; Leander, G.; Moradi, A.; Peyrin, T.; Sasaki, Y.; Sasdrich, P.; Sim, S.M. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Proceedings of the Advances in Cryptology, Santa Barbara, CA, USA, 14–18 August 2016; pp. 123–153.
64. Vafaei, N.; Saha, S.; Bagheri, N.; Mukhopadhyay, D. Fault Attack on SKINNY Cipher. *J. Hardw. Syst. Secur.* **2020**, *4*, 277–296. [CrossRef]
65. Yang, D.; Wen-Feng, Q.; Hua-Jin, C. Impossible Differential Attacks on the SKINNY Family of Block Ciphers. *IET Inf. Secur.* **2017**, *11*, 377–385. [CrossRef]
66. Li, L.; Liu, B.; Wang, H. QTL: A New Ultra-Lightweight Block Cipher. *Microprocess. Microsyst.* **2016**, *45*, 45–55. [CrossRef]
67. Sadeghi, S.; Bagheri, N.; Abdelraheem, M.A. Cryptanalysis of Reduced QTL Block Cipher. *Microprocess. Microsyst.* **2017**, *52*, 34–48. [CrossRef]
68. Banik, S.; Pandey, S.K.; Peyrin, T.; Sasaki, Y.; Sim, S.M.; Todo, Y. GIFT: A Small Present Towards Reaching the Limit of Lightweight Encryption. In Proceedings of the Cryptographic Hardware and Embedded Systems, Taipei, Taiwan, 25–28 September 2017; pp. 321–345.
69. Zhao, B.; Dong, X.; Meier, W.; Jia, K.; Wang, G. Generalized Related-key Rectangle Attacks on Block Ciphers with Linear Key Schedule: Applications to SKINNY and GIFT. *Des. Codes Cryptogr.* **2020**, *13*, 1103–1126. [CrossRef]
70. Dalmasso, L.; Bruguier, F.; Benoit, P.; Torres, L. Evaluation of SPN-Based Lightweight Crypto-Ciphers. *IEEE Access* **2019**, *7*, 10559–10567. [CrossRef]
71. Patil, J.; Bansod, G.; Kant, K.S. LiCi: A New Ultra-Lightweight Block Cipher. In Proceedings of the Emerging Trends Innovation in ICT, Pune, India, 3–5 February 2017; pp. 40–45.
72. Aboshosha, B.W.; Dessouky, M.M.; Elsayed, A. Energy Efficient Encryption Algorithm for Low Resources Devices. *Acad. Res. Community Publ.* **2019**, *3*, 26. [CrossRef]
73. Thabit, F.; Alhomdy, S.; Jagtap, S. Security Analysis and Performance Evaluation of a New Lightweight Cryptographic Algorithm for Cloud Computing. *Glob. Transit. Proc.* **2021**, *2*, 100–110. [CrossRef]
74. Kumar, V.G.K.; Rai, C.S. Design and Implementation of Novel BRISI Lightweight Cipher for Resource Constrained Devices. *Microprocess. Microsyst.* **2021**, *84*, 104267. [CrossRef]
75. Sophia, B.; Jeril, L.; Kavin Harnesh, M.; Lalith Kumar, V. A Secure Remote Clinical Sensor Network Approach for Privacy Enhancement. In Proceedings of the 2021 International Conference on Computing, Communication, Electrical and Biomedical Systems (ICCCEBS), Coimbatore, India, 25–26 March 2021; Volume 1916, p. 012107. [CrossRef]
76. Mousavi, S.K.; Ghaffari, A. Data Cryptography in the Internet of Things Using the Artificial Bee Colony Algorithm in a Smart Irrigation System. *J. Inf. Secur. Appl.* **2021**, *61*, 102945. [CrossRef]
77. Al-Husainy, M.A.F.; Al-Shargabi, B.; Aljawarneh, S. Lightweight Cryptography System for IoT Devices Using DNA. *Comput. Electr. Eng.* **2021**, *95*, 107418. [CrossRef]
78. Rashmi Chawla, V.; Sehgal, R.; Nagpal, R. The RC7 Encryption Algorithm. *Int. J. Secur. Its Appl.* **2015**, *9*, 55–60. [CrossRef]
79. Kelsey, J.; Schneier, B.; Wagner, D. Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Proceedings of the Information and Communications Security, Beijing, China, 11–14 November 1997; pp. 233–246.
80. Lu, J. Related-Key Rectangle Attack on 36 Rounds of the XTEA Block Cipher. *Int. J. Inf. Secur.* **2008**, *8*, 1–11. [CrossRef]
81. Suzaki, T.; Minematsu, K.; Morioka, S.; Kobayashi, E. Twine: A Lightweight, Versatile Block Cipher. In Proceedings of the ECRYPT Workshop on Lightweight Cryptography, Louvain-la-Neuve, Belgium, 28–29 November 2011; pp. 146–169.

82. Webster, A.F.; Tavares, S.E. On the Design of S-Boxes. In Proceedings of the Advances in Cryptology — CRYPTO '85 Proceedings, Santa Barbara, CA, USA, 18–22 August 1986; pp. 523–534.
83. Thiemjarus, S. Pervasive Sensing for Fall Detection and Beyond. In Proceedings of the The International Advanced Medical Robotics Symposium, Bangkok, Thailand, 15–16 March 2019.
84. Yu, X.; Wang, C.; Zhou, X. Review on Semi-Fragile Watermarking Algorithms for Content Authentication of Digital Images. *Future Internet* **2017**, *9*, 56. [CrossRef]