SEDA: Secure Over-The-Air Code Dissemination Protocol for the Internet of Things

Jun Young Kim, Wen Hu, *Senior Member, IEEE,* Hossein Shafagh, Sanjay Jha, *Senior Member, IEEE*

Abstract—The capability to securely (re)program embedded devices over-the-air is a fundamental functionality for the emerging Internet of Things (IoT). Current approaches work efficiently by exploiting the *homogeneity* within sensing devices, where all nodes require the update and participate in the process. Due to the *heterogeneity* of IoT deployments, where the type of devices and program images vary, existing solutions suffer from severe performance degradation. We address this shortcoming with our system SEDA, which leverages a *secure multicast approach*. SEDA outperforms existing systems since the program image is securely delivered and processed on targeted nodes only, hence reducing the overhead for not involved nodes. In order to enable the multicast approach, we introduce an *asymmetric broadcast encryption* primitive, which we have optimized towards constrained nodes to reduce the communication/computation overhead as compared to existing approaches. With an extensive experimental study on a public testbed in several practical settings, we show SEDA's efficient performance compared to state-of-the-art approaches. Finally, our theoretical security analysis shows SEDA's security against identified adversary models.

Index Terms—Dissemination, Over-the-air update, Internet of Things, Multicast, Security, Group key distribution.

1 Introduction

The Internet of Things (IoT) encompasses a wide range of devices enabling the interconnection of the physical world to the digital world. Typical application scenarios include health monitoring systems (e.g., wearables), automated homes, and enterprise (e.g., hospitals). However, many devices are launched in the market without considering security during the design phase and hence can be easily compromised [36]. This makes IoT devices an attractive target for attackers. For instance, IoT devices constitute 38% of the victims of a cryptocurrency mining worm [5]. Hence, this endangers the vision of IoT to fall down into the *Internet of Stupid Things* [1]. This problem will exacerbate when the number of such poorly designed, faulty, or malicious devices start to wreck havoc in the future. This motivates the need for secure over-the-air (re)programming capabilities as a basic functionality for IoT devices.

The Thread[1] group [9] depicts a technological effort, which has the goal to securely connect/manage a large number (+250) of various IoT devices in home networks. Similar to our work, this group aims to design energy-efficient security solutions, where battery-powered devices can run for years. Low-power and secure over-the-air update, referred to as code dissemination, is a key feature in achieving the security goals. In order to design energy efficient applications, the group uses Public Key Cryptography (PKC) as authentication and symmetric key encryption such as AES for the content encryption. The group also adopts low-power-multi-hop networks such as IPv6 Low-power Wireless Personal Area Networks (6LoWPAN) to minimize the energy consumption from wireless communications. Communication and computation are two premium resources to be considered in an energy-efficient design, as radio activities and security operations consume the majority of energy on constrained devices. Unfortunately, existing over-the-air (re)programming solutions suffer from severe performance degradation especially due to the *heterogeneity* of devices. We elaborate this in the next subsection.

1.1 Heterogeneity of IoT Deployments

Code dissemination has been already studied in the context of Wireless Sensor Networks (WSN) [23], [25]. However, the heterogeneity of devices (e.g., variety of platforms) presents a new challenge when considering IoT deployments. A secure code dissemination protocol should be communication/energy efficient, since these devices are inherently resource constrained, remotely deployed, and may require multi-hop packet forwarding capability without any permanent power source.

Existing WSNs over-the-air (re)programming protocols, such as Deluge [23], are based on the epidemic communication [28] which assumes homogenous sensor nodes in the network with all nodes participating in the (re)programming process. Epidemic code dissemination approaches perform efficiently in homogeneous networks by exploiting *spatial multiplexing*, i.e., parallel transmissions in different parts of the network. Given the multitude of IoT applications, the epidemic approach would require non-target nodes to participate in the propagation resulting in unnecessary computation intensive cryptographic operations and additional packet forwarding overhead. This has a negative impact on the sparse resources of such networks, specifically with regards to energy.

- *Jun Young Kim, Wen Hu, and Sanjay Jha are with the Department of Computer Science and Engineering, UNSW Australia, and data61, CSIRO Sydney Australia,*
  E-mail: jun.kim@unsw.edu.au
- *Hossein Shafagh is with the Department of Computer Science at ETH Zurich, Switzerland.*

Manuscript received February 1, 2016; revised December 12, 2016.

1. The Thread group is organized by major IT companies such as ARM, Qualcomm, and Samsung. It focuses on connectivity in the IoT.

1.2 Group Key Distribution

In this paper, we adopt a multicast communication approach [2], where only code-image-update-involved target nodes (targets hereafter) actively take part in the process. Many studies [33], [37] already showed that the multicast approach outperforms the epidemic propagation. Therefore, we can enhance the propagation efficiency while providing a secure dissemination function with the multicast approach. The rationale for our design is that we significantly reduce the communication and computation overhead compared to epidemic approaches by avoiding unnecessary communication and computation-intensive cryptographic operations at both target and non-target nodes.

In order to enable a secure multicast, a fresh and secure group key distribution to arbitrary targets is a precondition. Existing group key distribution schemes, however, assume known/fixed group or neighbor pairing for efficient key distribution [16], [31]. These schemes fall back to the naive approach of using multiple secure unicasts to emulate a secure multicast group key distribution when supporting dynamic groups [2]. Based on our study on group key distribution schemes, we propose to use a public key cryptography broadcast encryption scheme, which efficiently distributes the shared group key to a dynamic number of target nodes.

This paper is motivated by the development of reference models and benchmarks to ease the burden of reproducible experiments on complex systems. With this in mind, we make the following contributions:

1) We develop SEDA[2], a SEcure over-the-air code Dissemination Architecture for the IoT, which adopts multicast communication to improve the protocol efficiency in heterogeneous IoT applications. SEDA demonstrates superior performance over existing methods.
2) Our design focuses on the selection, implementation, and optimization of a public key cryptographic broadcast encryption scheme (BGW_t) for efficient group key distribution. Our optimizations overcome performance and memory restrictions on constrained IoT nodes.
3) We identify emerging security threats, then provide counter-measures to the program image update process. Our security analysis demonstrates that our security properties are not compromised by identified adversary models.
4) We experimentally validate SEDA through a prototype IoT platform and demonstrate the efficiency of SEDA in practical settings. We publicly release our implementation as an open-source code[3].

The paper is organized as follows. Sec. 2 presents related research on secure code dissemination systems and key distribution. Sec. 3 presents our design criteria and Sec. 4 sets the security goals with adversary models. Sec. 5 introduces the SEDA protocol. We provide implementation, evaluation, and security analysis of the SEDA protocol in Sec. 6, 7, and 8, respectively. Sec. 9 concludes the paper.

2. SEDA means strong in Korean.
3. SEDA is available in https://github.com/jun-kim/SEDA

2 Related Work and Background

In this section, we discuss existing over-the-air update systems and their shortcomings in heterogeneous IoT applications. We then discuss a number of group key distribution techniques relevant to our work.

In commercial IoT deployments, over-the-air update can be performed via lightweight secure End-to-End (E2E) protocols [24], [39], where the server propagates a new image to targets one by one. Although these E2E protocols are lightweight (for instance consisting of ZigBee [4] as the wireless link communication technology, Datagram TLS (DTLS) [6] as the secure transport layer protocol, and Constrained Application Protocol (CoAP) [40] as the application layer protocol), E2E propagation exhibits unscalable linear overhead with the increase of the number of nodes and communication hops.

Deluge [23] is a well-known code dissemination protocol for WSNs. Many efforts have been made to secure Deluge such as Seluge [25]. These extensions add authenticity, integrity, and known attack protections to the dissemination process by leveraging various techniques such as digital signature, Merkle hash tree, one-way hash functions [35], and pairwise encryption [31]. They inherently follow Deluge's epidemic architecture [28], where a node with the newer version program image becomes a sender and a node with an older version becomes the receiver. Although recent dissemination efforts (e.g., Pando [19], Splash [15]) introduced enhanced dissemination techniques such as constructive interference, scheduling, and pipelining, these techniques are out of scope of this paper since they are not considering security and heterogeneous environments.

As discussed earlier, the use of epidemic communication in heterogeneous IoT environments is inefficient, mainly due to the unnecessary packet delivery and security operations held on both non-target and target nodes. As an alternative for the IoT context, we explore multicast communication primitives, known to be more efficient and better suited for heterogeneous nodes [33].

Group Domain of Interpretation (GDOI) [2] relies on the multicast communication. In GDOI, a trusted key distribution server sends advertisements to all legitimate receivers and senders to securely distribute a fresh shared group key. The senders then multicast the content encrypted by the shared group key to the corresponding set of receivers. To increase the key distribution efficiency, GDOI adopts lightweight secure E2E protocols such as DTLS. Similar to the E2E propagation, E2E key distribution approaches exhibit linear overhead with the increased number of targets and hops when accessing the trusted key distribution server. Our evaluation in Sec. 7 will illustrate this inefficient linear overhead compared to SEDA's stable performance.

In the context of WSNs, many approaches have been proposed to establish a shared group key. However, the majority of these approaches require *fixed* grouping [16] or neighboring authentication such as pairwise encryption [31]. In a heterogeneous IoT deployment, the number of target nodes (e.g., the number of smart light bulbs in a building) can change dynamically. Devices may not form a fully connected network by themselves, requiring other smart objects to bridge the communication. Furthermore,

some constrained non-target nodes may have to help forward packets, which requires unnecessary, computationally expensive cryptographic operations. Thus, existing secure group key distribution systems in WSN are not ideal for heterogeneous applications.

Fiat and Naor [21] first introduced Broadcast Encryption (BE) to broadcast a shared secret key to a group of targets on the same channel. Only targets in a dynamically defined group can decrypt the shared key using their private keys. In BE, each node is required to store $O(k \log k \log n)$ keys and the sender must broadcast $O(k^2 \log^2 k \log n)$ messages to establish the shared key, where k is the maximum number of non-target nodes that cannot collude to learn the secret and n is the number of total nodes. BE works well when k is small; otherwise, BE produces significant communication/storage overhead. Although a number of approaches have been proposed to reduce communication and/or storage overhead, they are still very often either linearly or exponentially k-dependent, as it is challenging to set a k value that balances security and overhead in practice.

A number of Public key cryptographic-based Broadcast Encryption schemes (PBE), such as [13], [14], [41] were proposed to accommodate the possible large number of nodes and to provide full collusion resistance. PBEs offer various distinctions for the over-the-air (re)programming protocol design regarding heterogeneous IoT applications for several reasons:

- **Dynamic grouping:** PBEs have the capability to target any combination/number of target nodes.
- **Shorter/fixed-size ciphertext (keying material):** Among various tradeoffs of PBEs, SEDA chooses a scheme with short ciphertext size and lower decryption overhead to increase communication/energy efficiency on the constrained node side.
- **Full collusion resistance:** Traditional BEs are k-resilient, which means $k+1$ compromised nodes can collude to generate the key while PBEs are fully collusion resistant.

3 Design Goals

In this section, we discuss the design of SEDA. One of our major design goals is minimizing the consumption of the two premium resources, namely communication and computation while providing a secure code dissemination function.

3.1 Shared Group Key Distribution and Decryption

We first discuss an efficient and secure group key distribution, which we utilize to enable the multicast approach. The design space of PBEs for shared group key distribution includes public key size, private key size, ciphertext size, encryption complexity, and decryption complexity. Based on our extensive survey on group key distribution schemes [26], [27], we present the comparison of our short-listed PBEs in Table 1. All of these schemes employ a hybrid encryption paradigm [10], where the server broadcasts the encrypted shared group key, which is later used to encrypt the broadcast information.

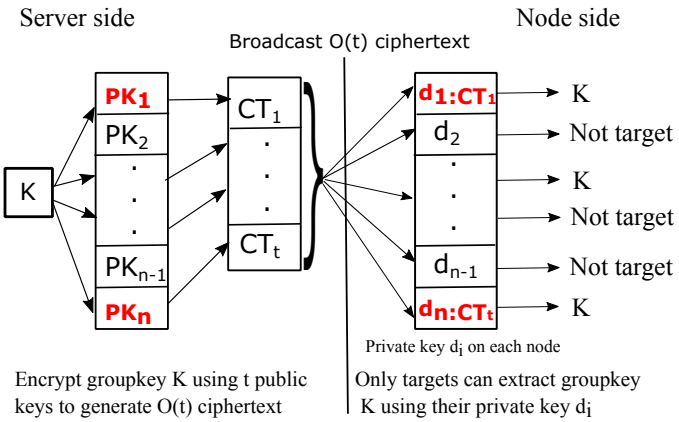


Fig. 1. Trivial broadcast encryption scheme in an n-nodes network targeting t nodes (t is a dynamic multicast group).

To elaborate the PBE concept, let us consider the Trivial [41] method. Trivial produces ciphertext size of $O(t)$ in a network that has n nodes and t dynamic targets. In Trivial, each node possesses a private key and the key distribution server sequentially encrypts the shared group key using t public keys to generate a concatenation of t ciphertext. After broadcasting the linear ciphertext, *each target node* decrypts its share with its own private key and obtains the shared group key (see Fig. 1). Trivial is simple and effective, however, the ciphertext size is proportional to the number of target nodes. Similarly, the ciphertext size is inversely proportional to the number of targets for trapdoor-based schemes, such as Delerablée$_1$ [14]. These linear ciphertext distinctions are not suitable for IoT applications, especially for a multicast approach due to the increased communication overhead.

Based on our study, we found BGW_t [13] proposed by Boneh et al. to be the most suitable scheme. BGW_t results in additional storage but improves overall communication efficiency. In addition, it offers reduced decryption complexity on the IoT device side with increased encryption complexity on the server side.

We provide our rationale for selecting BGW_t as our group key distribution scheme below:

1) **The shortest ciphertext size**: Shorter or fixed size ciphertext is a desirable feature, as it is directly proportional to the communication overhead for the group key distribution. BGW_t achieves the shortest ciphertext size of $O(1)$, but trades off linear storage to store all other nodes' public keys. The public key can be stored in inexpensive ROM or flash disks. Thus, SEDA sacrifices linear storage for communication efficiency. We evaluate the communication/storage tradeoff in Sec. 7.
2) **Lower decryption complexity**: On a dissemination protocol, encryption is performed at the level of a resource-rich trusted server while decryption is performed at the level of constrained nodes. Since longer public keys are pre-computed/installed and used on nodes, decryption on the node side requires significantly less complexity. Thus, BGW_t has higher encryption complexity of $O(n)$ while lower decryption complexity of $O(1)$, which enhances energy consumption/latency on the constrained nodes.

	Public key	Private key	Ciphertext	Encryption complexity	Decryption complexity
Trivial [41]	$O(n)$	$O(1)$	$O(\mathbf{t})$	$O(t)$	$O(1)$
Delerablée$_1$ [14]	$O(n)$	$O(1)$	$O(\mathbf{r})$	$O(r^2)$	$O(\mathbf{r})$
Delerablée$_2$ [14]	$O(n)$	$O(n)$	$O(1)$	$O(r^2)$	$O(\mathbf{r}^2)$
BGW [13]	$O(\sqrt{n})$	$O(1)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
BGW_t [13] (SEDA)	$O(\mathbf{n})$	$O(1)$	$O(\mathbf{1})$	$O(\mathbf{n})$	$O(\mathbf{1})$

TABLE 1
Comparison of our shortlist PBEs [14] in an n-node network targeting arbitrary t nodes, r non-target nodes.

3) **Fixed size private key**: IoT nodes are vulnerable to physical node capture attack, where adversaries compromise nodes to extract pre-installed keys. Short size private keys are desirable since they can be stored in the tamper-resistant storage (if applicable) to effectively mitigate physical attacks. BGW_t's $O(1)$ private key size is another benefit for the more secure application design.

In the following, we discuss other PBE candidates and show why they are not suitable for our protocol design (see Table 1). Trivial and Delerablée$_1$ have $O(1)$ private key and lower decryption complexity, but their linear ciphertext is not suitable for communication efficiency. Delerablée$_2$ has $O(1)$ ciphertext but its private key size is linear and decryption is exponential. Boneh et al.'s first BGW system offers well-balanced ciphertext/public key size, but communication is more precious resource than storage on constrained nodes. Considering this, BGW_t is the best match for our protocol design in terms of ciphertext size, decryption complexity and private key size. Although we chose BGW_t as the scheme of our system, we show the communication efficiency of Trivial and Delerablée$_1$ in Sec. 7.1, which can be suitable for alternative application scenarios with fixed number of target nodes. For example, if we wish to design an IoT application to target almost every node without pre-key-installation, Delerablée$_1$ would be a suitable choice. However, if the number of targets is always small, Trivial can be an option.

3.2 Implementation and Optimization

Asymmetric cryptography is still an expensive security primitive for constrained IoT nodes. Bringing BGW_t type heavy crypto to the IoT domain may not be feasible due to lack of computation power and memory. We implement and optimize our proposed BGW_t scheme on a constrained node to show its application in the IoT domain. We evaluate our protocol implementation on a public testbed to quantify the performance of SEDA in practical settings.

4 Prerequisites

Now we briefly discuss important prerequisites about communication scenarios and security, before diving into SEDA's design in the next section.

4.1 Network and Protocol Scenarios

SEDA is designed to (re)program deployed nodes via a secure over-the-air protocol. It maintains a multicast session for one type/group of nodes with the group membership being dynamically set during the network lifetime. SEDA assumes a network model that consists of *heterogeneous* wireless nodes, using IP protocols, i.e., IPv4/IPv6, on 6LoWPAN. We assume the trusted server/cloud (server hereafter) is resource-rich in terms of computation, communication bandwidth and power. Nodes have different computation and power resources and run a variety of Operating Systems (OS) that support multicast communication services[4]. Nodes can perform a limited number of asymmetric key cryptography operations such as Elliptic Curve Cryptography (ECC) and RSA (Rivest Shamir Adleman) either with or without hardware cryptographic support. We use ECC as the asymmetric cryptographic primitives for SEDA. This design decision is mainly driven by the efficiency of ECC for embedded devices, as ECC supports the same level of security with shorter keys than RSA. However, SEDA can also operate with other asymmetric cryptographic primitives such as RSA. We further assume that nodes have pre-installed PBE public and private key pairs and the public key of a trusted server.

Nodes join the multicast communication tree in which the trusted server is a member and register their profiles with the trusted server using protocols such as Universal Plug and Play (UPnP)[5]. The trusted server has then all relevant profile information such as node type, multicast group membership, and public key of all legitimate nodes in the network.

4.2 Adversary Model

We assume an adversary model that follows the Dolev-Yao security model [17] with additional physical attack capabilities. Adversaries can intercept, modify and transmit messages over wireless channels and they can perform effective computations in polynomial time. We assume that adversaries can be either outsiders or insiders since they may have the ability to physically compromise one or few deployed nodes. The goal of an adversary is to compromise a large number of nodes[6] in the network by attacking the dissemination protocol.

4. Either standard IP multicasting (http://tools.ietf.org/html/rfc1112) or light-weight low-power multicast services such as those available in popular embedded device Operating Systems like Contiki (https://github.com/adamdunkels/contiki-fork/tree/master/examples/ipv6/multicast), and TinyOS [37].

5. ISO/IEC standard on UPnP device architecture makes networking simple and easy. http://www.iso.org/iso/news.htm?refid=Ref1185

6. physically compromising a larger number of nodes would increase risks for an adversary to be caught.

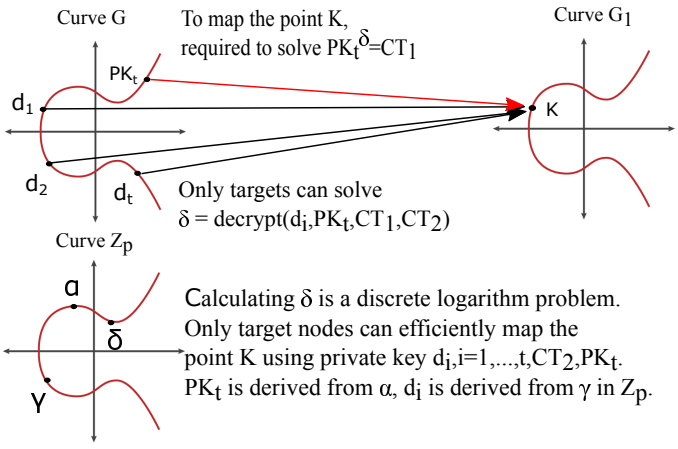


Fig. 2. A simple description of BGW_t. Only targets have advantage in calculating δ to map the point K using their own private key d_i.

4.3 Security Goals

Compromising the server can be a single point of failure resulting in the failure of the entire system [38]. Thus, even under fully distributed scenarios, we assume the server is a powerful node, hosted securely and cannot be compromised. SEDA has the following security goals:

- The adversary cannot extract the group key.
- A colluding group of adversaries outside the group of target nodes cannot access the shared group key. Further, compromised nodes in the target group can be immediately revoked as soon as they are identified (identifying compromised nodes is beyond the scope of this paper).
- Nodes can identify any fake code dissemination session.
- Targets can mitigate Denial of Service (DoS) attacks [35] by identifying forged dissemination advertisements.
- Extracting the disseminated code image is computationally expensive for adversaries.

4.4 Cryptographic Primitives and Key Management

In this section, we briefly introduce the cryptographic primitives of BGW [13] and discuss a variant BGW_t used in our system. In brief, BGW is a bilinear maps and groups based PBE system for dynamic targets with well-balanced sizes of $O(\sqrt{n})$ ciphertext/public key (see Table 1). Bilinear maps systems are widely used for Identity Based Encryption (IBE) [12] and Attribute Based Encryption (ABE) [11]. The notion of bilinear maps and groups system is that given two multiplicative cyclic groups G and G_1 of prime order p, there exists an efficiently computable bilinear map $e : G \times G \rightarrow G_1$, which is symmetric, as $e(u^a, v^b) = e(u, v)^{ab} = e(u^b, v^a)$, $e(u, u) \neq 1$ for all $u, v \in G$ and $a, b \in Z_p$. We can simply refer to e as an admissible pairing that can map from two points in G to a point in G_1.

BGW provides a tradeoff between the ciphertext size and public key size. With a longer public key size, we can achieve a constant/short ciphertext size, which results in an improved communication efficiency. However, the downside of BGW is that it requires nodes to additionally store all multicast group nodes' public keys in the process of decrypting the fixed size ciphertext. Put simply, this would allow to achieve a fixed size ciphertext at the expense of linear storage overhead. This extreme tradeoff case is referred as BGW_t and is used in SEDA for an improved group key distribution efficiency.

Since we propose to use BGW_t as our group key primitive, we briefly describe its security and three key functions: *Setup*, *Encryption* (performed at the server), and *Decryption* (performed at resource-constrained nodes).

The key generation function Setup(n) picks a random generator $g \in G$ of a cyclic group G and $\alpha, \gamma \in Z_p$. Then computes $g_i = g^{\alpha^i} \in G$ for $i = 1, ..., 2n$ and $v = g^\gamma \in G$. The generated public keys for all n nodes (PK) and private key (d_i) for each node $i = 1, ..., n$ are derived as per [13]:

$$PK \Leftarrow (g, g_1, ...g_{2n}, v) \in G^{(2n+1)} \quad (1)$$

$$d_i \Leftarrow g_i^\gamma, \ for\ i = 1, ..., n \in G \quad (2)$$

The Encryption(t, PK_t) generates the group key (K) and ciphertext (CT_1, CT_2) for arbitrary targets t using a random generator $\delta \in Z_p$, all targets' public keys (PK_t), and the bilinear map function $e(,)$. Note that K is an elliptic point in G_1.

$$K \Leftarrow e(g_{n+1}, g)^\delta \in G_1 \quad (3)$$

$$CT_1 \Leftarrow g^\delta, \ \ CT_2 \Leftarrow (v \cdot \prod_{j \in t}(g_{(n+1-j)}))^\delta \quad (4)$$

The security of BGW_t is based on the Bilinear Diffie-Hellman Exponent assumption (BDHE), which is known as a discrete logarithmic problem. By capturing CT_1, CT_2 and having access to targets' public keys PK_t as input, adversaries try to map point K in G_1 by calculating δ from $PK_t^\delta = CT_1$. There is no known algorithm with non-negligible probability to compute δ in polynomial time. BGW_t is semantically Chosen Ciphertext Attack (CCA) secure with this property of bilinear maps system (see Fig. 2).

On the other hand, target nodes with their own private key have advantage in solving the BDHE problem via Decryption($d_i, CT_{1,2}, PK_t$) and the group key K:

$$K \Leftarrow e(g_i, CT_2) \ / \ e(d_i \cdot \prod_{j \in t, b \neq j}(g_{(n+1-j+b)}), CT_1) \quad (5)$$

Since PK_t ($\prod_{j \in t, b \neq j}(g_{(n+1-j+b)})$ is required in (5), nodes are required to store all other nodes' public keys. This distinction results in linear storage overhead, which we will address in the following section.

5 SEDA

In this section, we first address two drawbacks of the BGW_t scheme in the IoT domain then present our efforts towards more efficient design. We introduce the basic architecture of SEDA, followed by a detailed description of the protocol.

5.1 Addressing two critical drawbacks of BGW_t

Although BGW_t scheme is the most suitable one for the IoT domain, there are two critical drawbacks to be used in the IoT domain:

1) **Public key pre-installation and management**: In many IoT applications, an incremental node addition is desirable. However, the nature of pre-public-key-installation restricts this functionality and it is challenging to manage linear public keys after deployment. This makes the management architecture more complicated and restricted.
2) **Higher key decryption complexity on constrained nodes**: We noticed that computing PK_t ($\prod_{j\in t, b\neq j}(g_{(n+1-j+b)})$) in (5) dominates the decryption time, especially when there is a large number of nodes. This makes the decryption complexity more complicated and requires more memory as well.

To address these two troublesome problems, we eliminate the public key pre-installation restriction by adding one ciphertext CT_3:

$$CT_3 \Leftarrow \prod_{j\in t, b\neq j} g_{(n+1-j+b)} \tag{6}$$

This solution can also solve the higher key decryption complexity problem, as the value is pre-computed in CT_3 by the resource-rich server. Then the key K can be decrypted more efficiently without pre-installed public keys:

$$K \Leftarrow e(g_i, CT_2) \,/\, e(d_i \cdot CT_3, CT_1) \tag{7}$$

We will elaborate this optimization in Sec. 7.2.

5.2 Efficient Key Revocation

Key revocation is necessary when a key is lost or a node is compromised. The majority of BE schemes require linear or logarithmic overhead for the key revocation [34]. However, SEDA's revocation is simple and only requires updating two ciphertext CT_2, CT_3 regardless of the number of revoked keys. Since CT_1, CT_2, CT_3 are generated as $CT_1 \Leftarrow g^{\delta}$, $CT_2 \Leftarrow (v\cdot \prod_{j\in t}(g_{(n+1-j)}))^{\delta}, CT_3 \Leftarrow \prod_{j\in t, b\neq j}(g_{(n+1-j+b)}$ in (4,6), we can simply revoke any user u:

$$\begin{aligned} CT_2' &\Leftarrow CT_2 \,/\, g^{\delta}_{(n+1-u)}, for\ u \in 1, ..., n \\ CT_3' &\Leftarrow CT_3 \,/\, g_{(n+1-u)}, for\ u \in 1, ..., n \end{aligned} \tag{8}$$

Alternatively, a re-keying process can be initiated by server, which results in a key distribution session after removing the user u from the target list. This re-keying process only requires updating three ciphertext CT_1, CT_2 and CT_3.

5.3 Elliptic Curve Point Compression

We applied the elliptic curve point compression technique [32] to reduce the communication and storage overhead. The notion of elliptic curve point compression is that an elliptic point (x, y) can be represented by only x value with a single bit indicating which direction of y to calculate the point. Although it requires a point decompression process, we can reduce the element size by almost half. The main gain from this technique is that we only need to multicast almost half ciphertexts to the network at lower decompression cost (e.g., average 10 ms is required for a point decompression using 160 bits generated curves). We will elaborate the storage gain and point decompression cost from the elliptic point compression technique in Sec. 7.3.

5.4 Feedback Suppression and Chaining

In our initial experiments, we noticed that our multicast propagation approach suffers severely from feedback related issues resulting in severely increased overhead and delay in the dissemination process. For instance, propagating 10 kB of image to 33 devices required over 55 seconds (now 29 seconds on average). We noticed that propagating 90 % of image only requires 20 seconds, but filling the rest 10 % requires more time due to feedback related issues. We briefly describe the feedback suppression and chaining technique to minimize two feedback-related issues.

1) **Feedback implosion.** Multicast propagation is prone to the feedback implosion problem when many receivers send feedback to the sender such as ACK or Negative ACK. This problem is not only energy inefficient, but also causing channel interference problems especially in dense networks.
2) **Long tail problem.** Achieving 100% reliability on all nodes will cause backoff delay and the long tail problem, where 100% reliability requires orders of magnitude longer time than the time for achieving 80% reliability.

As Pando [19] showed methods in solving these two issues in dissemination protocols, we apply the feedback suppression and chaining technique. Nodes hold their feedback until the time threshold for the arrival of the last propagation packet. Then send feedback for missing or HMAC verification failed packets for retransmission to the parent of the multicast tree. All parents nodes chain their children's feedback to generate a chain of feedback. This technique is especially effective in dense networks. However, the epidemic propagation cannot support the feedback suppression and chaining technique since it requires sequential packet transfer.

Key	size	Description
d_i	$1 \times m$	BGW_t Private key
GD_S	$1 \times m$	Server's ECDSA public key

TABLE 2
Pre-installed keys in each node. m: BGW_t one element key size (e.g., 22 bytes or 176 bits).

Notation	Meaning
$A \parallel B$	Concatenation of A and B
$\|M\|$	The length of M (bytes)
$D_k(M)$	Symmetric decryption using the key k
$E_k(M)$	Symmetric encryption using the key k
$BD_k(M)$	BGW_t decryption to extract the group key K
$H(M)$	HMAC generation using K
$Sig(M)$	Server/cloud's ECDSA signature
$Ver(M)$	Signature or HMAC verification

TABLE 3
Notation summary.

5.5 Initial Setup

In initial setup, a server (S) sets two important parameters then generates BGW_t public/private key pairs:

1) **The maximum size of the network** n: SEDA generates n pairs of BGW_t keys to be pre-installed.

2) **The key size** r: We set the level of security by setting the key size of the elliptic curve used in the BGW_t system. The key size results in the element size (m), which represents an elliptic point on the BGW_t scheme. All keys and ciphertext are represented using this element. For instance, for r=160 bit curve, one BGW_t element size is 42 bytes, which we reduce to 22 bytes using elliptic curve point compression [32].

After pre-installing and deploying all nodes, each node in the network maintains two keys:

1) The server's Elliptic Curve Digital Signature Algorithm (ECDSA) public key GD_S for signature-based authenticity guarantee of the advertisements: Stored in ROM or flash disk;
2) Each node's own BGW_t private key (d_i, $i = 1, ..., n$, where n is the number of total nodes): Stored in a tamper-resistant storage if available;

Table 2 shows the details of the pre-installed keys in each node. Deployed nodes register themselves to S and join the multicast group according to our network scenario.

5.6 An Overview of the SEDA Protocol

At a high level, SEDA consists of 3 phases, namely the initial setup, the advertisement phase, and the actual code image propagation (see Fig. 3). The code image dissemination to dynamic targets t follows the following 4 steps:

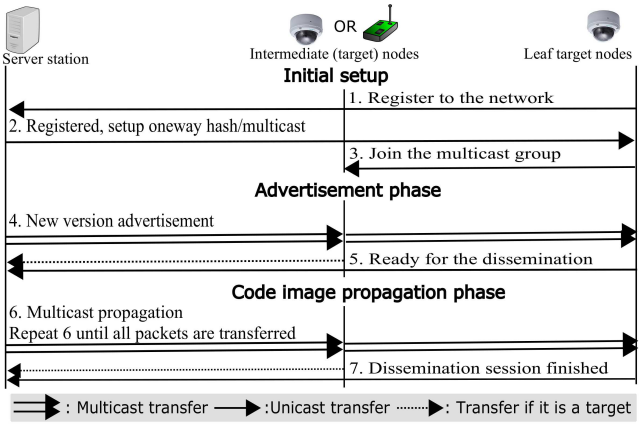


Fig. 3. The protocol flow of SEDA.

1) The server S advertises a new image dissemination via multicast to an arbitrary number of target nodes (t) (e.g., all the smart light bulbs in the building). This advertisement contains important information such as the shared session group key encrypted as CT_1, CT_2, CT_3, and a cryptographic signature to provide the authenticity and integrity to the advertisement packet.
2) Intermediate nodes just relay multicast/unicast packets, and do not perform any cryptographic computations. Only nodes in t can decrypt the group key K from ciphertexts, then reply with a ready message to the server S.
3) The server multicasts the program image to the network.
4) Nodes in t can verify the integrity of the image with the Hash-based Message Authentication Code (HMAC) using the group key K. Finally, if encrypted, the image can be decrypted with K and the update procedure initiated.

In the following, we describe the details of the two phases. The notations used in this work are summarized in Table 3.

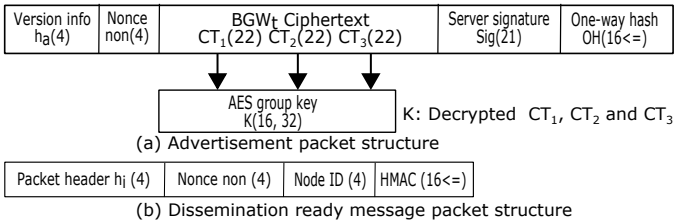


Fig. 4. Advertisement packet structure (bytes).

Algorithm 1: Advertisement phase for the server station

Data: Plain code image, public key of target nodes
Result: Advertisement header ad
1 $K \Leftarrow$ `Generate_key(`PK,*random value*`)`
2 $CI \Leftarrow E_K$(Plain code image) (optional)
3 $h_a \parallel non \Leftarrow$ `Generate_Header(`t,CI`)`
4 $CT_{1,2,3} \Leftarrow$`Generate_ciphertext(`K, PK_t, t`)`
5 $OH \Leftarrow$ `OnewayHash(`$h_a \parallel non \parallel CT_{1,2,3} \parallel Sig$`)`
6 $ad \Leftarrow h_a \parallel non \parallel CT_{1,2,3,} \parallel Sig \parallel OH$
7 **while** *Waiting for dissemination ready from* $\forall t$ **do**
8 `Multicast(`ad`)`
9 **if** *Received ready message from* $\forall t$ **then**
10 go to code image propagation phase
11 **else**
12 go to `Multicast(`ad`)` after a certain period

5.7 Advertisement Phase

The code propagation process is triggered when there is a new program image available for a group of target nodes t. The server S generates an advertisement packet (ad) (see Fig. 4 for the packet format). The ad packet includes a new image version header h_a, a one-way hash (OH), $CT_{1,2,3}$, and the signature of S Sig. The advertisement phase at the server consists of the following steps (see Alg. 1):

1) S picks up a group key K that will be used to encrypt the plain code image to generate the-group-key-encrypted code image CI (this is optional according to the characteristic of the application). It then generates h_a and non (line 1-3 in Alg. 1).
2) Ciphertexts are generated using K and a set of target t's public keys (PK_t). S signs the packet to generate Sig and generates one-way hash OH. It then generates ad as a concatenation of all fields (line 4-6 in Alg. 1).
3) Lastly, S multicasts ad using a periodic timer until all nodes in t acknowledge readiness for the dissemination session (line 7-12 in Alg. 1).

On the receiver side, each node determines whether it is a target by parsing h_a. All target nodes t decrypt K and send a ready message to S. However, non-target/intermediate

Algorithm 2: Advertisement phase for all nodes

Data: Advertisement packet ad
Result: Group key K, transfer ready message to S
1 **if** *NotTarget(h_a)* **then**
2 return NOT_TARGET
3 **if** `FAILED(`*Ver(OH)* $\|$ *Ver(Sig)*`)` **then**
4 return INVALID_ADVERTISEMENT
5 $K \Leftarrow BD_{PK_t \| d_i}(CT_1 \| CT_2 \| CT_3)$ for all i,i=1,...,t
6 `Send_Ready_Message(`S, (i)`)`

nodes just forward the packet without processing any security operations. This is the computational gain of SEDA compared to epidemic protocols, where all nodes should perform computationally expensive security operations. The receiver side advertisement phase, as described in Alg. 2, consists of the following steps:

1) Parse the version header h_a to check the relevance of the code update. Non-targets just forward the packet (line 1-2 in Alg. 2).
2) Target nodes verify OH to prevent DoS attacks on the advertisement packet and verify Sig for signature-based authentication (line 3-4 in Alg. 2).
3) Perform BGW_t decryption $K \Leftarrow BD_{PK_t \| d_i}(CT_1 \| CT_2 \| CT_3)$ to generate the group key K (line 5 in Alg. 2).
4) This phase ends when all nodes in t transfer the ready message to S (line 6 in Alg. 2). We assume that reliability is provided by underlying multicasting protocol.

Algorithm 3: Code image propagation phase for the server

Data: (Groupkey-encrypted) code image CI
Result: Multicast all packets P_i
1 $h_i \Leftarrow$ `Generate_Multicast_Header(`CI`)`
2 $P_0 \Leftarrow h_i \| 0 \| |CI| \| H(CI) \| HMAC$
3 $|Pkt| \Leftarrow MTU - |h_i| - |seq| - |HMAC|$
4 **for** $i = 1$ **to** $|CI| \ / \ |Pkt|$ **do**
5 $Pkt_i \Leftarrow CI[(|Pkt| * i)]$
6 $h_i \Leftarrow$`Generate_Packet_Header(`Pkt_i`)`
7 $P_i \Leftarrow h_i \| i \| Pkt_i \| HMAC$
8 **for** $i = 0$ **to** $|CI| \ / \ |Pkt|$ **do**
9 `Multicast(`P_i`)`

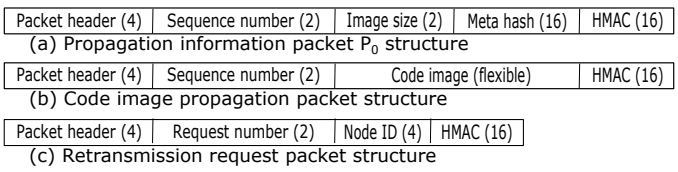


Fig. 5. Multicast propagation packet structure.

5.8 Code Image Propagation Phase

In this phase, S multicasts the (group-key-encrypted) code image to the multicast group. Upon receiving the ready message from all targets, S commences the multicast propagation following Alg. 3:

1) The first multicast packet P_0 includes the description of the new image CI such as total size and the meta hash $H(CI)$ with sequence number (seq) (line 1-2 in Alg. 3). The structure of the propagation packet is described in Fig. 5.
2) $CI \Leftarrow E_K$(plain code image) is divided into fixed size packets Pkt_i, $i = 1, ..., |CI| \ / \ |Pkt|$, where the size of $|Pkt_i|$ is flexible according to the MTU of the network (line 3-5 in Alg. 3).
3) Multicast packet P_i is generated using packet header h_i, seq, for each packet Pkt_i with HMAC (line 6-7 in Alg. 3).
4) All multicast packets $P_i, i = 0, ..., |CI| \ / \ |Pkt|$ are propagated via the multicast channel.

Algorithm 4: Code image propagation phase for targets

Data: All propagation packet $\sum_{i=1}^{m} P_i$
Result: New code image NI
1 $m \Leftarrow |CI| \ / \ |Pkt|$
2 **for** $i = 0$ **to** m **do**
3 **if** `NoPacket(`Pkt_i`)` $\|$ `FAILED(`*Ver(H(P_i))*`)` **then**
4 return `Retrans_Request(`i, t`)`
5 **else**
6 $CI[i] \Leftarrow Pkt_i$
7 **if** $D_K(E_K(H(CI))) \neq H(\sum_{i=1}^{m} Pkt_i)$ **then**
8 return NO_HASH_MATCH
9 **for** $i = 1$ **to** m **do**
10 $NI[i] \Leftarrow D_K(Pkt_i)$
11 `Dissemination_Finished_Message(`*t,HMAC*`)`

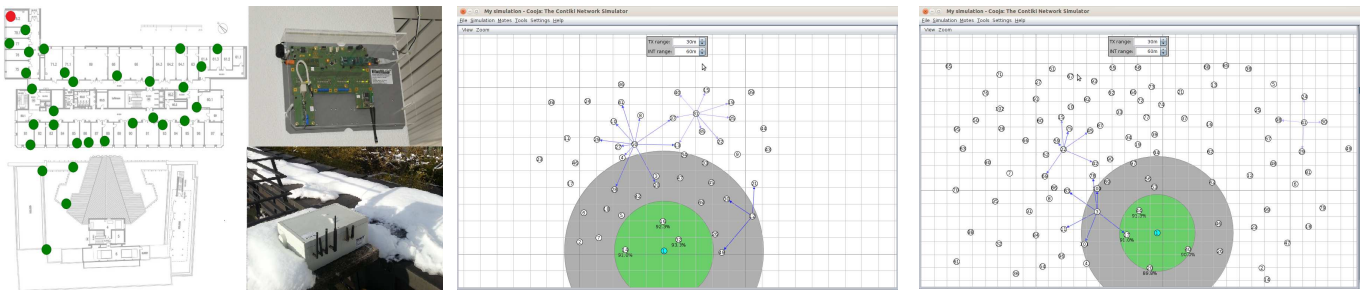
On the receiver side, targets buffer all HMAC-verified propagation packets while queuing retransmission requests for missing or HMAC check failed packets. Upon receiving all multicast packets, targets extract the new code image NI following Alg. 4:

1) If there is a missing or/and HMAC mismatch packet, targets request retransmission to the multicast group until all legitimate packets are received (line 1-6 in Alg. 4).
2) Targets perform meta hash verification after extracting all encrypted code image CI from Pkt_i, which is fulfilled if the meta hash matches with the hash of the received image, $D_K(E_K(H(CI))) = H(\sum_{i=1}^{m} Pkt_i)$ (line 7-8 in Alg. 4).
3) If the image is encrypted, targets can perform the symmetric decryption $D_K(\sum_{i=1}^{m} Pkt_i)$ to extract the code image (line 9-10 in Alg. 4).

The entire dissemination session is finalized when all targets transmit the dissemination complete message to the server.

6 Implementation and Experimental Setup

Our prototype implementation of SEDA consists of server side and node-side components. We implemented our server-side BGW_t key generation module using the type-A



(a) Flocklab testbed with in/outdoor nodes. (b) Cooja simulator for a 50 node network. (c) Cooja simulator for a 100 node network.

Fig. 6. Experimental setup. (a) the Flocklab testbed (b) Cooja simulator (n=50) (c) Cooja simulator (n=100)

pairing in the Pairing Based Cryptography (PBC) library [3], [32]. We have implemented the node-side component of SEDA in the Contiki 2.7 distribution [20] for the Openmote [7] platform. Implementing BGW_t type heavy cryptography on constrained nodes is restricted due to lack of performance and memory. As lower memory footprint is desirable in constrained devices, we optimized our node-side pairing-based-arithmetic operations by applying elliptic curve optimization techniques [30], [42] with BGW_t specific optimization techniques. We will elaborate our optimization in Sec. 7.2.

Openmote represents our constrained IoT device model. We show our superior security by securing Openmote type constrained IoT devices. Openmote is equipped with Cortex-M3 processor with up to 32 Mhz clock, 512 kB ROM, 32 kB RAM, cryptographic accelerator, and tamper resistant key storage. Although current constrained IoT devices such as TelosB and medical implant devices are not equipped with cryptographic accelerator and tamper-resistant key storage, we believe smart-home IoT applications would require similar capabilities to protect from emerging attacks such as physical compromise attacks.

We employ Stateless Multicast RPL Forwarding (SMRF) [37] multicast protocol in combination with NullRDC duty cycle. SMRF protocol is more suitable and efficient for the dissemination protocol design with the following multi-fold benefits:

1) Multicast packets only travel downwards of the tree, which is more suitable for dissemination protocols.
2) Stateless protocols require no additional per-packet information.
3) The multicast tree is determined dynamically before the dissemination session.
4) SMRF protocol requires no extra maintenance packets since it works on top of the existing Routing Protocol for Low-Power and Lossy Networks (RPL) messages.

Module	ROM	RAM (static)
PBC, GMP library	53.8 kB	3.9 kB
SEDA Dissemination architecture	26.8 kB	4.8 kB
SMRF multicast and routing	2.3 kB	1.3 kB
Total code size	83.0 kB	10.2 kB

TABLE 4
Memory footprint of SEDA on Openmote.

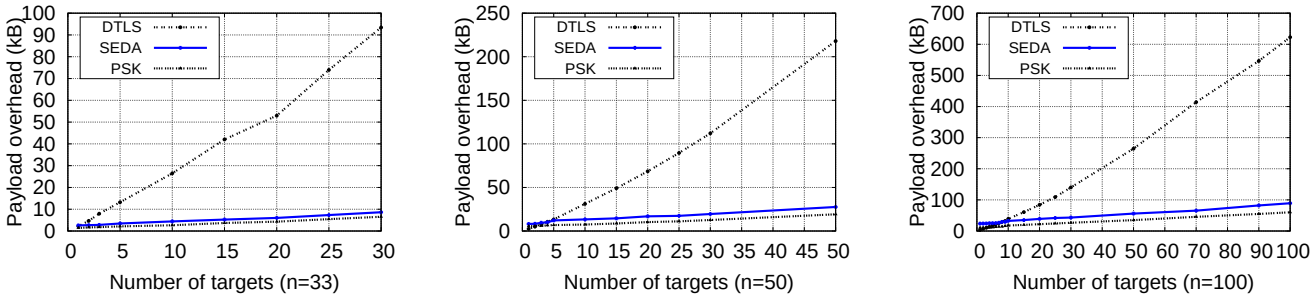
SEDA supports AES 128/256 bits key symmetric cryptography and HMAC using SHA 128/256 bits hash functions. Table 4 shows the memory footprint of our node-side implementation. We evaluated our system on Flocklab [29], a public testbed, equipped with 33 indoor/outdoor heterogeneous nodes such as Tmote Sky, ACM2, Opal, CC430, TinyNode, Wismote, and Openmote. Flocklab provides Internet connectivity for the sensors, which allows us to emulate real world IoT settings with accurate measurements such as power profiling, GPIO actuation, adjustable supply voltage, and serial I/O functions on a web interface. We evaluated the 33 node case in Flocklab, and simulated 50, 100 node cases using Cooja, the Contiki simulator [20] (see Fig. 6. Unit Disk Graph Medium, transfer range: 30 m, interference range: 60 m, NullRDC, IEEE 802.15.4.).

7 Evaluation

7.1 Communication Overhead

In order to show the key distribution efficiency, we first compare SEDA to two other group key distribution approaches based on: DTLS and Pre-Shared-Key (PSK). Then, we show the superior performance of SEDA compared to a state-of-the-art epidemic approach. Lastly, we show the group key establishment efficiency of BGW_t over other PBEs we presented. To evaluate the communication overhead, we measured the *transmitted payload* overhead. We included both multicast and unicast payloads, but excluded routing management packets. We use $r = 160$ bit generated elliptic curves or equivalent key size for the evaluation.

We start with comparing SEDA to the GDOI scheme, which leverages DTLS as a secure E2E protocol for authentication and group key distribution. Fig. 7 shows the payload overhead to complete the advertisement phase regarding the number of target nodes. DTLS protocol is designed to offer lightweight E2E security by means of PKC, but the overhead is larger given an increasing number of targets and communication hops. Even though the communication overhead of E2E protocols can be decreased via various approaches, such as header compression, the nature of E2E protocols still makes them unscalable. We also included a PSK approach for comparison, where there is a fixed grouping and the group key is already deployed on each node. PSK causes lower communication overhead than SEDA, but lacks efficient key revocation. Consequently, with the disclosure of the pre-shared-key, e.g., by node compromise, the entire network is jeopardized. As we observe in Fig. 7, the E2E approach is only scalable when the number of targets is small, while SEDA causes stable distribution overhead

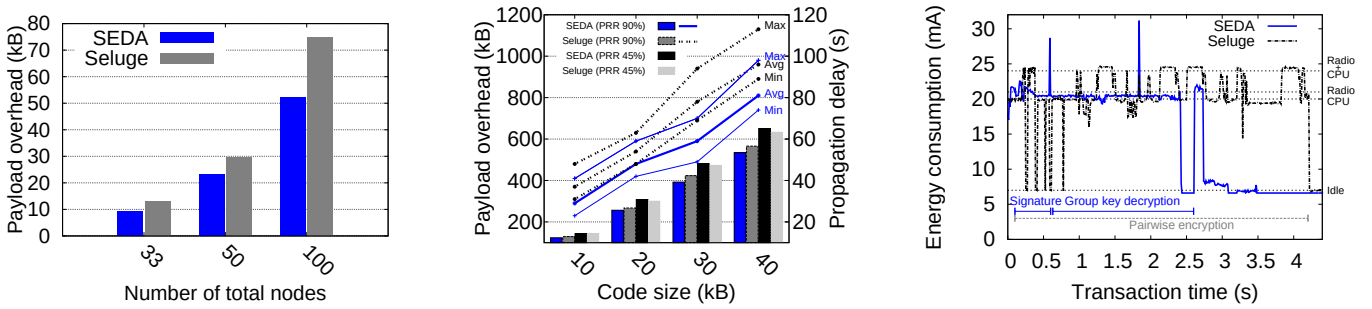


(a) SEDA yields better communication efficiency than DTLS over 2 targets.

(b) SEDA yields better communication efficiency than DTLS over 3 targets.

(c) SEDA yields better communication efficiency than DTLS over 8 targets.

Fig. 7. Payload overhead of advertisement phase compared to the DTLS and PSK approach in three network scenarios. DTLS is causing linear communication overhead to the number of targets, while SEDA has a lower slope even if the number of targets increases.



(a) Key establishment overhead comparison.

(b) Propagation overhead (bar, left axis) and delay (line, right axis) comparison.

(c) Average energy overhead comparison for key distribution.

Fig. 8. Comparison between SEDA and Seluge.

regardless of the number of targets. Note that SEDA shows less communication overhead than DTLS when the number of targets is over 6-8% of total, i.e., over 2, 3 and 8 target nodes for a 33, 50, 100 node network, respectively.

	SEDA	**Seluge**
Propagation	Multicast	Epidemic
Targeting	Arbitrary target	All nodes
Security operations	Targets only	All nodes
Authenticity	ECDSA	ECDSA
Integrity	HMAC	Merkle tree
Key distribution	PBE	Pairwise
Pre-key-installation	1 private key	Logarithmic
Collusion resistance	Full	Weak
Key revocation	2 messages	linear

TABLE 5
The general comparison between SEDA and Seluge.

Now we discuss the efficiency gain of SEDA over Seluge [25], a state-of-the-art epidemic approach (Default Seluge setup: 48 packets per page, 102 bytes MTU, 8 leaves for the Merkle hash tree). Table 5 shows the general comparison between SEDA and Seluge. Seluge's cluster key (Pairwise) exhibits logarithmic pre-key-installation, but it suffers from weak collusion resistance while SEDA requires only 1 private key installation with full collusion resistance. Seluge enhances the dissemination efficiency and security by exploiting the sequential packet delivery distinction of the epidemic approach. The Merkle hash tree is an efficient integrity guarantee method as long as the delivery is sequential. However, this sequential delivery restriction becomes an additional burden in the propagation since it causes an increasing number of Selective Negative ACK (SNACK) packets. SEDA leverages per-packet HMAC using the group key, in which the sequential delivery is not necessary.

Directly comparing SEDA to Seluge is not trivial since they have different assumptions. To make this comparison, we assume SEDA is targeting all nodes in the network, which is the worst case performance for SEDA. Fig. 8(a) depicts SEDA's superior key distribution performance in medium-sized networks. The efficiency gain gap between SEDA and Seluge is proportional to the size of the network since Seluge's key establishment overhead depends on the number of neighboring nodes. The bar graph with left axis in Fig. 8(b) depicts the propagation efficiency comparison in the Flocklab. When SEDA is targeting all nodes, SEDA and Seluge share similar propagation strategies, but different retransmission methods for lost packets. To show the impact of different re-transmissions, we emulated physical layer packet loss by dropping packets randomly to obtain 90% (good link quality) and 45% (poor to moderate link quality) Packet Reception Ratio (PRR). For the code propagation overhead, SEDA performs slightly better with good link quality and Seluge is more efficient when the link quality is bad. Again, this performance is Seluge's best case and SEDA's worst case since SEDA can choose a specific target group while Seluge cannot. The line graph with right axis depicts the maximum, average, and minimum propagation delay according to the size of the code image. As aforemen-

tioned, Seluge's sequential delivery restriction causes more SNACK packets, which results in increased propagation delay, while SEDA does not require sequential transfer. Note that SEDA's communication/delay efficiency are further increased, when we target a subset of nodes instead of all nodes, which is the typical heterogeneous IoT scenario SEDA is designed for.

We also compared the energy consumption for the key establishment (see Fig. 8(c)). Seluge shows a higher level of energy consumption pattern due to constant packet exchange between neighboring nodes. SEDA consumes in average 117.6 mJ while Seluge consumes 164.85 mJ for the key distribution phase. We will elaborate this consumption in Sec. 7.4.

To conclude our discussion about communication overhead, we now discuss the evaluation of employing other PBE candidates instead of BGW_t. As mentioned in Sec. 3.1, E2E is infeasible in multi hop communication and the communication overhead of both Trivial and Delerablée$_1$ schemes is linear due to the dynamic number of target nodes. To elaborate, the ciphertext size of Trivial is proportional to the number of targets t, Delerablée$_1$ is proportional to the number of non-target nodes r, and none of both can be deterministic in the number of targets. Nevertheless, BGW_t offers constant ciphertext size. Fig. 9(a) shows the communication overhead evaluation of the advertisement phase in case we employ other schemes on the SEDA protocol design. Both Trivial and Delerablée$_1$ are only efficient when we have a small number of targets or non-targets, while SEDA is stable regardless of the number of targets.

7.2 Computation Overhead

Two important design goals of SEDA were to avoid unnecessary security operations as the epidemic approach, and to decrease expensive computations on the resource-constrained IoT devices. As we described in Sec. 4.4, BGW_t is a tradeoff case that enables this by transferring the large share of computation to the two functions; Setup and Encryption, which are performed on the server side. Deployed nodes leverage server-generated pre-computed ciphertext and perform decryption in $O(1)$. Our BGW_t improvements in Sec. 5 further enhanced the decryption efficiency since the decryption time is not dominated by the number of nodes anymore with the pre-computation technique. However, generating elliptic curves and performing the bilinear pairing to map the point K is still a computationally intensive operation on constrained nodes. Inspired by TinyECC [30] and TinyPairing [42], which demonstrated the optimization of elliptic curve based operations on constrained nodes, we optimized our node-side BGW_t implementation along with the GMP library for efficient pairing-based-arithmetic operations. As a result, the one-time-per-session group key decryption becomes efficient on the constrained node side (see Fig. 9(b) and Table 6). For instance, the average group key decryption time for r=160 bits is optimized to 1,864 ms from 2,982 ms.

Hu et al. [22] showed that asymmetric cryptography can be used and enhanced in WSNs with the assistance of a cryptographic accelerator, which costs significantly less energy on common sensor nodes. Utilizing a cryptographic accelerator for the bilinear pairing can further enhance the performance of SEDA. Regarding other security operations, our evaluation platform Openmote can enhance cryptographic operations with the on-board cryptographic accelerator that supports HMAC, ECDSA, and AES. However, we also included the computation overhead of the software implementation to show the impact of the cryptographic accelerator. Table 7 shows the processing time of all cryptographic operations with their energy consumption on Openmote. Software implementation consumes more energy since it requires longer operation time than the cryptographic accelerator.

7.3 Storage Overhead Gain

In this section, we show how much storage overhead reduction is achieved with our BGW_t drawbacks addressing described in Sec. 5.1. In the original BGW_t scheme, the ciphertext is fixed to 2 elements, the private key is 1 element while the public key includes $(2n+1)$ elements. The element size increases, as we choose longer r bits generated elliptic curves. For example, if we choose a $r = 160$ bit generated elliptic curve, 1 element size is 42 bytes (see Table 6). We reduce this element size to 22 bytes with the elliptic curve point reduction and compression [32] to increase the communication efficiency.

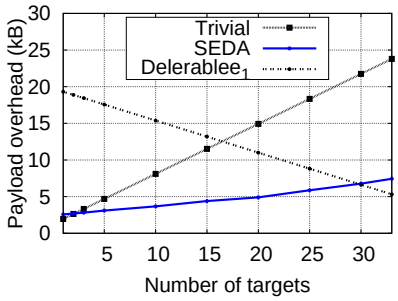
Fig. 10 shows the removed storage overhead with our modification with and without elliptic curve point compression. For example, when we use r=160 bits generated curves, the storage requirement is about 5 kB with compression, 10 kB without compression in a 100 node network. Furthermore, each node is required to store only one element private key d_i in a tamper-resistant storage if applicable, to mitigate physical attacks.

7.4 Energy Consumption

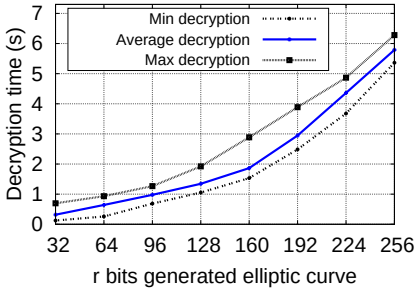
We measured the energy consumption for performing security operations during the advertisement phase such as one-way hash/ECDSA verification, group key decryption, and AES decryption. We used a power analyzer to measure the current draw on a 10 Ω shunt resistance. The energy consumption is calculated as: current (mA) $\times$ transaction time (sec) $\times$ input voltage (2.1 V). Note that only 1 dissemination ready message transmission is performed and radio is mostly listening during the advertisement phase. Table 7 shows the energy consumption of each security operation and the total advertisement phase. The advertisement phase consumes a total of 117.6 mJ.

7.5 Latency

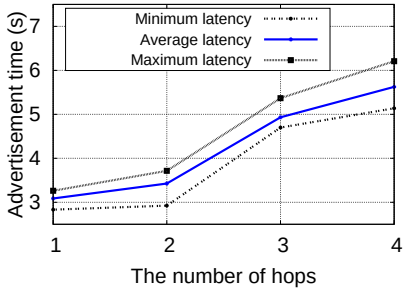
We measured the average advertisement phase completion time according to the number of hops in the Flocklab testbed (see Fig. 9(c)). Given the multicast and lossy network, the round trip time varies as the number of hops grows. This latency includes the whole advertisement phase such as multicasting advertisement packet, signature verification, group key decryption, and transferring the ACK message to the server. Note that the average advertisement packet processing time on the node side is 2,281 ms, which is only performed once per session. SEDA's superior latency over



(a) Payload overhead of advertisement phase in Flocklab, in case of adopting other candidate PBEs.



(b) Group key decryption time(ms) on Openmote regarding the r bits generated elliptic curve.



(c) Advertisement phase completion time in Flocklab (n=33).

Fig. 9. Evaluation of the SEDA protocol. (a) Comparison with other PBEs (b) group key decryption overhead (c) latency.

Elliptic curve	Element size	Compressed Element	Decompression	Initial Decryption	Optimized Decryption
128 bits	34 bytes	18 bytes	8 ms	1,931 ms	1,345 ms
160 bits (≡RSA1024)	42 bytes	**22 bytes**	10 ms	2,982 ms	**1,864 ms**
192 bits	50 bytes	26 bytes	16 ms	4,260 ms	2,947 ms
224 bits (≡RSA2048)	58 bytes	30 bytes	21 ms	5,801 ms	3,862 ms
256 bits (≡RSA3072)	66 bytes	34 bytes	23 ms	8,231 ms	5,789 ms

TABLE 6
Key element size and processing time. For $r = 160$ bit setting (the level of security is equivalent to RSA 1024 bits) with point compression, SEDA needs 22 bytes private key, 66 bytes ciphertext.

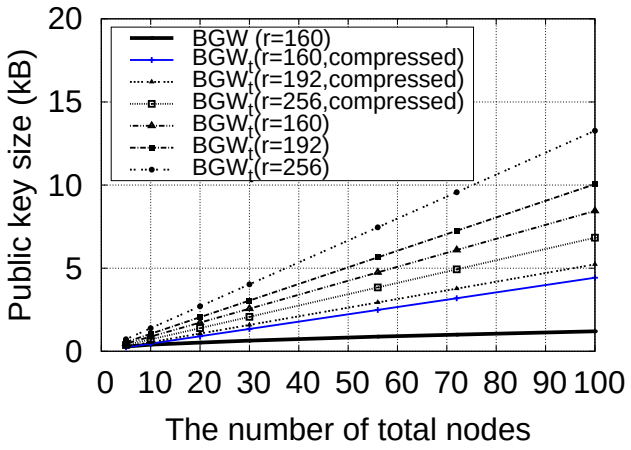


Fig. 10. The storage overhead gain from storing the public key according to the r generated elliptic curve

Module	Type	Current	Time, Energy
Hash verification (73 bytes payload)	HW SW	24.3 mA 20.3 mA	25 μs, 1.2 μJ 721 μs, 30 μJ
ECDSA verification (160 bits)	HW SW	25.9 mA 20.7 mA	148 ms, 8.0 mJ 393 ms, 17.1 mJ
Elliptic curve point decompression	SW	20.5 mA	10 ms, 430 μJ
Group key decryption	SW	21.5 mA	1,864 ms, 84.1 mJ
AES-128 decryption (80 bytes payload)	SW	20.4 mA	214 μs, 9.1 μJ
AES-256 decryption (80 bytes payload)	HW SW	23.2 mA 20.8 mA	31 μs, 1.5 μJ 456 μs, 19.9 μJ
HMAC verification (84 bytes payload)	HW SW	24.3 mA 20.8 mA	27 μs, 1.3 μJ 780 μs, 34.0 μJ
Radio active		24.7 mA	421 ms, 21.8 mJ
Total	SW	21.7 mA	2,582 ms, 117.6 mJ

TABLE 7
Average security operation processing time and energy consumption on Openmote with hardware crypto accelerator (HW) or in software implementation (SW).

DTLS and Seluge is achieved via the small advertisement packet size and minimized ACK messages. However, Seluge's Merkle hash tree requires linear packet size and DTLS suffers from E2E communication in multi hop environment.

8 Theoretical Security Analysis

8.1 Authenticity and Signature based Attack Mitigation on the Advertisement

The new dissemination advertisement packet ad includes the following fields: new version advertisement, server's ECDSA signature, ciphertext, and one-way hash. Although there are many efficient ECDSA verification implementations, it is still a resource consuming task. Since SEDA leverages the ECDSA signature as an immediate authentication, adversaries can launch signature-based attacks to cripple nodes by spending energy on signature verification. Therefore, SEDA utilizes a lightweight one-way hash for initial verification. Upon receiving the ad, target nodes first verify the one-way hash (which is cheap to compute compared to a signature verification) then verify the signature to authenticate the advertisement message. The same approach applies to the most expensive operation, namely group key decryption, as it is performed after the signature verification is completed. Thus, SEDA achieves authenticity and signature-based attack mitigation via one-way hash and signature verification.

8.2 The Security of Group Key Distribution

Secure distribution of the group key plays an important role for a secure system. Thus, the security of the group key is a crucial part of our protocol design. As we already showed, adversaries may capture ad to extract ciphertext CT_1, CT_2, CT_3 and have access to the public keys. However, adversaries cannot extract any information to solve the BDHE problem in polynomial time to generate the group key. In addition, SEDA generates a fresh group key every

session and this one-timeness gives us even a higher level of security.

8.3 Integrity and DoS Attack Mitigation

Adversaries can inject forged packets to install malicious code or to sabotage nodes. In epidemic code dissemination systems, mechanisms such as the per-packet meta hash (Merkle hash tree) and one-way hash chain have been employed to verify the integrity and to mitigate DoS attacks [25]. We leverage HMAC in combination with the group key to prevent code injection attacks and to guarantee the integrity. HMAC is lightweight but robust against known attacks such as brute force and collision attacks as long as the key is not disclosed. SEDA also leverages the meta hash for the entire (encrypted) code image for additional integrity of the new code image. After receiving a propagation packet, target nodes perform HMAC verification for integrity which mitigates DoS attacks because it is hard for adversaries to inject forged or modified packets by tampering HMAC.

8.4 Confidentiality and Reverse Engineering Protection

In many IoT applications, adversaries can easily mount snooping attacks on the code image. This reverse engineering is an effective way to reveal any security holes in the source code. To achieve confidentiality and prevent reverse engineering, SEDA optionally allows encryption of code image using the symmetric group key K. Symmetric cryptography, such as AES, causes relatively small computation overhead for critical applications to preserve confidentiality of code (see Table 7). The use of a fresh group key per session allows a higher level of confidentiality against snooping attacks.

8.5 Full Collusion Resistance and Physical Attack Mitigation

By means of physical attacks, an attacker can extract the installed keys and launch an inside attack. In traditional BE, adversaries can use the extracted keys to collude and generate the group key. Acquiring higher collusion resiliency is infeasible in traditional BE schemes. Boneh et al. [13] proved that the bilinear maps systems based PBEs are fully collusion resistant as BDHE assumption is intractable. They proved that for all $g, h \in G_1$, $\delta \in Z_p$, and $K \in G_T$, there is no efficient algorithm for deciding whether $e(g,h)^{\delta} = K$. Thus, even the disclosure of all other key materials, BGW_t is fully collusion resistant as long as the private keys in the target group are not disclosed. If applicable, storing the private key in a tamper-resistant storage provides additional security against the node capture attack. Moreover, pre-installation of private keys in a tamper-resistant storage also provides security against identity-based Sybil attacks [18]. Thus, SEDA can achieve physical attack mitigation and full collusion resistance.

More importantly, we have tested SEDA using a formal security analysis tool, AVISPA [8] with its On-the-Fly-Model-Checker tool. The tool did not find any security threats.

9 Conclusion

In this paper, we introduced the design, implementation, and evaluation of SEDA, an efficient and secure over-the-air (re)programming protocol for heterogeneous IoT applications. We addressed the challenges of using current secure code dissemination protocols in heterogeneous IoT applications. To solve this problem, SEDA adopts the multicast communication approach to avoid unnecessary propagation and security operations on non-target nodes in the update process. In order to enable the secure multicast approach, we explored many well-known asymmetric broadcast encryption schemes then we choose the BGW_t scheme for improved communication and computation efficiency. We first analyzed two critical drawbacks of the original BGW scheme then presented a solution towards IoT applications. We performed extensive experiments in Flocklab, a public testbed, and with the Cooja simulator to show the superior efficiency of the SEDA protocol with defensive measures against adversary models.

References

[1] http://blog.apnic.net/2015/04/30/the-internet-of-stupid-things.
[2] http://datatracker.ietf.org/doc/rfc6407/.
[3] https://github.com/oliverguenther/PBC_BKEM.
[4] http://www.zigbee.org, 2006.
[5] http://www.symantec.com/connect/blogs/iot-worm-used-mine-cryptocurrency, 2015.
[6] https://tools.ietf.org/html/rfc6347, 2015.
[7] http://www.openmote.com/, 2015.
[8] Avispa. http://www.avispa-project.org/, 2015.
[9] The thread group. http://threadgroup.org/, 2015.
[10] M. Abe, R. Gennaro, K. Kurosawa, and V. Shoup. Tag-kem/dem: A new framework for hybrid encryption and a new analysis of kurosawa-desmedt kem. In *EUROCRYPT 2005*.
[11] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy*, pages 321–334. IEEE, 2007.
[12] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.
[13] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology–CRYPTO 2005*, pages 258–275, 2005.
[14] C. Delerablée, P. Paillier, and D. Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Pairing*, pages 39–59. 2007.
[15] M. Doddavenkatappa, M. C. Chan, and B. Leong. Splash: Fast data dissemination with constructive interference in wireless sensor networks. In *NSDI*, pages 269–282, 2013.
[16] I. Doh, J. Lim, and K. Chae. Code updates based on minimal backbone and group key management for secure sensor networks. *Mathematical and Computer Modelling*, 57(11):2801–2813, 2013.
[17] D. Dolev and A. C. Yao. On the security of public key protocols. *Information Theory*, 29(2):198–208, 1983.
[18] J. R. Douceur. The sybil attack. In *Peer-to-peer Systems*. Springer, 2002.
[19] W. Du, J. C. Liando, H. Zhang, and M. Li. When pipelines meet fountain: Fast data dissemination in wireless sensor networks. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 365–378. ACM, 2015.
[20] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *LCN*. IEEE, 2004.
[21] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology-CRYPTO*, 1994.
[22] W. Hu, H. Tan, P. Corke, W. C. Shih, and S. Jha. Toward trusted wireless sensor networks. *TOSN*, 7(1):5, 2010.
[23] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd Sensys*. ACM, 2004.

[24] R. Hummen, H. Shafagh, S. Raza, T. Voig, and K. Wehrle. Delegation-based authentication and authorization for the ip-based internet of things. In *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 284–292. Ieee, 2014.
[25] S. Hyun, P. Ning, A. Liu, and W. Du. Seluge: Secure and dos-resistant code dissemination in wireless sensor networks. In *ACM/IEEE IPSN*, 2008.
[26] J. Y. Kim. Secure and efficient management architecture for the internet of things. In *SenSys*, pages 499–500, 2015.
[27] J. Y. Kim, S. Jha, W. Hu, H. Shafagh, and M. A. Kaafar. Poster: Toward efficient and secure code dissemination protocol for the internet of things. In *Sensys*, pages 425–426. ACM, 2015.
[28] P. A. Levis, N. Patel, D. Culler, and S. Shenker. *Trickle: A self regulating algorithm for code propagation and maintenance in wireless sensor networks*. Computer Science Division, University of California, 2003.
[29] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *IPSN*. IEEE, 2013.
[30] A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *IPSN 2008*. IEEE, 2008.
[31] D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM TISSEC*, 2005.
[32] B. Lynn. The pairing-based cryptography library. *Internet: crypto. stanford. edu/pbc/[Mar. 27, 2013]*.
[33] A. Marchiori and Q. Han. Pim-wsn: Efficient multicast for ipv6 wireless sensor networks. In *WoWMoM*. IEEE, 2011.
[34] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology-CRYPTO 2001*, 2001.
[35] P. Ning, A. Liu, and W. Du. Mitigating dos attacks against broadcast authentication in wireless sensor networks. *ACM TOSN*, 4(1):1, 2008.
[36] S. Notra, M. Siddiqi, H. Habibi Gharakheili, V. Sivaraman, and R. Boreli. An experimental study of security and privacy risks with emerging household appliances. In *CNS*. IEEE, 2014.
[37] G. Oikonomou and I. Phillips. Stateless multicast forwarding with rpl in 6lowpan sensor networks. In *PERCOM Workshops*. IEEE, 2012.
[38] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 329–350. Springer, 2001.
[39] H. Shafagh, A. Hithnawi, A. Dröscher, S. Duquennoy, and W. Hu. Talos: Encrypted query processing for the internet of things. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 197–210. ACM, 2015.
[40] Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (coap). 2014.
[41] D. R. Stinson. *Cryptography: theory and practice*. CRC press, 2005.
[42] X. Xiong, D. S. Wong, and X. Deng. Tinypairing: a fast and lightweight pairing-based cryptographic library for wireless sensor networks. In *WCNC*. IEEE, 2010.

Jun Young Kim Jun Young Kim is a Ph.D. candidate and research assistant at the University of New South Wales (UNSW Australia), and Data61 CSIRO Sydney Australia. His main research interest includes securing wireless sensor networks and the Internet of Things, as he worked over 12 years in the embedded software industry before joining the Ph.D. program.

Wen Hu Dr. Wen Hu is a senior lecturer at School of Computer Science and Engineering, the University of New South Wales (UNSW). Much of his research career has focused on the novel applications, low-power communications, security and compressive sensing in sensor network systems and Internet of Things (IoT). Hu published regularly in the top rated sensor network and mobile computing venues such as ACM/IEEE IPSN, ACM SenSys, ACM transactions on Sensor Networks (TOSN), Proceedings of the IEEE, and Ad-hoc Networks.

Hu was a principal research scientist and research project leader at CSIRO Digital Productivity Flagship, and received his Ph.D from the UNSW. He is a recipient of prestigious CSIRO Office of Chief Executive (OCE) Julius Career Award (2012 - 2015) and multiple CSIRO OCE postdoctoral grants.

Hu is a senior member of ACM and IEEE, and is an associate editor of ACM TOSN, as well as serves on the organising and program committees of networking conferences including ACM/IEEE IPSN, ACM SenSys, ACM MobiSys, IEEE ICDCS, IEEE LCN, IEEE ICC, IEEE WCNC, IEEE DCOSS, IEEE GlobeCom, IEEE PIMRC, and IEEE VTC.

Hossein Shafagh is a Ph.D. candidate and research assistant at the Department of Computer Science of ETH Zurich, Switzerland. His research interests include designing and improving security protocols for wireless sensor networks and enabling secure Internet of Things.

Sanjay Jha Sanjay K. Jha is a Professor, head of the Networked Systems and Security Group (NetSyS) and director of CySPri Laboratory at the School of Computer Science and Engineering at the University of New South Wales. Prof. Jha holds a Ph.D. degree from the University of Technology, Sydney, Australia. Sanjay has published over 180 articles in high quality journals and conferences. He is the principal author of the book Engineering Internet QoS and a co-editor of the book Wireless Sensor Networks: A Systems Perspective. He has been very active in attracting research grants from ARC, industry and other funding agencies.