SEDIMENT: An IoT-device-centric Methodology for Scalable 5G Network Security

David Shur, Giovanni Di Crescenzo, Qinqing Zhang, Ta Chen,
Rajesh Krishnan, Yow-Jian Lin, Zahir Patni, Scott Alexander
Peraton Labs, 150 Mt. Airy Rd., Basking Ridge, NJ 07920
Email: {dshur,gdicrescenzo,qinqing.zhang,tchen,rkrishnan,ylin,zahir.patni,salexander}@peratonlabs.com
Gene Tsudik
University of California Irvine
Email: gts@ics.uci.edu

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771654

Abstract—**Advances in wireless networking, such as 5G, continue to enable the vision of the Internet of Things (IoT), where everything is connected, and much data is collected by IoT devices and made available to interested parties (i.e., application servers). However, events such as botnet attacks (e.g., [1]) demonstrate that there are important challenges in this evolution.**

In this paper we consider the problem of scalable and secure data publication from IoT devices, with included mechanisms that help towards device attacks prevention and detection. We propose SEDIMENT, a system and methodology which look more specifically at problems that arise in a network with a broad variety of devices, some of which have limited resources and some of which were designed for a less hostile environment. SEDIMENT uses a combination of software root of trust, remote attestation and resource-efficient cryptography, to build a system that scales across heterogeneous computing platforms. It allows for devices that range from battery-powered devices that are intended to operate for long periods up to server-class machines without power constraints. SEDIMENT provides a secure application layer that can be used for common communication paradigms such as publish-subscribe while following zero-trust principles in both protecting the end hosts from the network and other end hosts, as well as protecting the network from the end hosts.

I. Introduction

We envisage a future when 5G networks and their successors allow ubiquitous computing with substantially more IoT devices than currently exist. Effective consuming of the potentially enormous data produced by such devices requires scalable architectures, going beyond the natural approach of multiple dedicated producer-consumer sessions. Moreover, these future networks, being simultaneously critical infrastructure and the means used for cyberespionage and cyberwarfare, could present a significant threat to both government security and commercial intellectual property.

The limited SWaP (for Size, Weight and Power) resources on low-end battery powered devices offer the most challenges for deployment of advanced or even well-established security and cryptography solutions. Low-end device typically have limited security support: one cannot rely on the presence of modern security hardware such as a Trusted Platform Module (TPM) or a Trusted Execution Environment (TEE) for storing keys and sensitive code. In some cases, the processor may be a primitive embedded processor without rings or other process protection that can be used by the operating system. Additionally, these processors may have limited headroom to add security functionality, whether limited by the processor itself and/or the battery cost and/or the need to limit the impact of security or cryptography code execution on battery lifetime.

As traffic from a large variety of IoT devices is expected to go through 5G networks, it is likely that some will have limitations or bugs that allow them to be compromised. A major security challenge is that of preventing IoT devices from being accessed, compromised and managed to run botnet attacks to other network entities (see, *e.g.*, [1], [2]).

As attacks will inevitably happen, attack detection and reaction solutions remain of much interest. Another major security challenge is that of detecting attacks and/or reacting to them to allow for continued network operations.

To address all these challenges, we are building SEcure DIstributed IoT ManagemENT for 5G (SEDIMENT), which targets scalable data publication from IoT devices while satisfying desirable security and trust requirements, and device SWaP resource constraints. SEDIMENT operates across the entire scale of devices on a 5G network, with emphasis on resource-constrained IoT devices, and specifically on the under-served low-tier (*e.g.*, 8bit, 4K RAM) and mid-tier (*e.g.*, 32-bit Cortex-M3, 256K RAM) devices that lack advanced security features (*e.g.*, ARM TrustZone), not being addressed by the mainstream industry IoT security architectures, despite their proliferation. Key ingredients of our approach include:

1) Software-based root of trust for low- and mid-tier IoT devices that do not have hardware support such as a TPM or TEE. This ingredient helps preventing remote attacks to the IoT device, and securing cryptographic keys and code used in the next two ingredients.
2) Remote attestation of the software on the IoT device, that works in conjunction with a software-based root of trust (or hardware roots of trust if available). This ingredient helps detecting remote attacks to IoT devices.
3) Scalable secure many-to-many data publication based on resource-efficient attribute-based encryption from IoT devices. This ingredient helps achieving the desired secure many-to-many data publication functionality, and protecting data confidentiality, integrity and authenticity.

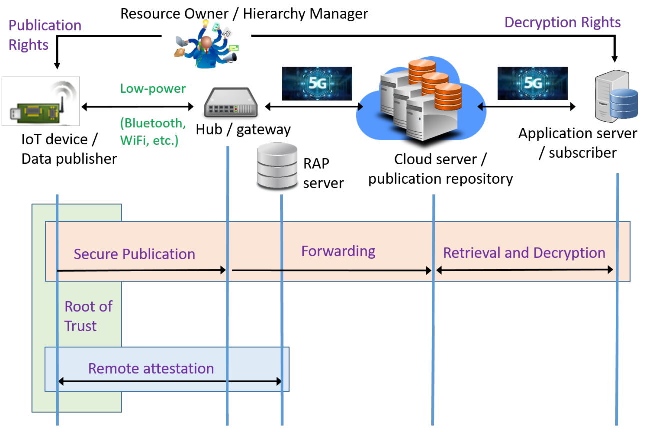

Fig. 1. SEDIMENT entities, interactions and protocol flows

It builds on and enhances the JEDI system [3] to improve IoT device runtime, battery consumption and lifetime.

The SEDIMENT design takes advantage of 5G resources, such as cloud servers, to store secure publications and make them available to subscribers, and Multi-access Edge Computing (MEC) servers, to route publication messages, and to perform intermediate computations, including remote attestation of IoT devices.

More specifically, the above 3 SEDIMENT ingredients are combined as follows (see also Figure 1).

1) IoT devices, acting as data publishers, issue secure publications for their measured data
2) Secure publications are (integrity-checked and) forwarded by hubs/gateways to cloud servers
3) Cloud servers make secure publications available to application servers, acting as subscribers
4) Remote attestation is periodically run between MEC servers and IoT devices, where the former attest authenticity of software stored on the latter
5) Roots of trust are used on IoT devices to protect symmetric cryptographic keys and code used in the above steps: i.e., to encrypt and authenticate data in secure publications and to hash memory in remote attestation.

The design of SEDIMENT ingredients also targets support of zero-trust security principles, as in NIST's recently proposed zero-trust security architecture [4], one of its main postulates being a cybersecurity paradigm shift from a single system perimeter defense to multiple resource-driven and minimum-privilege defense perimeters. Our supported principles and approaches to achieve this are summarized in Table I.

The 5G wireless environment is subject to eavesdropping, interference, losses, and intrusion vectors through the wireless link. In SEDIMENT, many-to-many end-to-end encryption and authentication protect against eavesdropping and message modification, and remote attestation protects against intrusion attacks that change either the firmware or application software. On the other hand, this paper does not focus on loss-resilient transport layers or physical layer robustness.

TABLE I
SUPPORT OF ZERO-TRUST PRINCIPLES IN SEDIMENT

Security principle	**SEDIMENT approach**
security of publication data	end-to-end encryption and authentication
authorization of decryption access	resource-owner decryption authorization and hierarchy-based access structures
security of device code/keys	tier-specific root of trust
continuous device security	remote attestation

TABLE II
AN IOT DEVICE CATEGORIZATION BASED ON MEMORY PROTECTION

Tier	**Constraints**	**SEDIMENT approach**
low	small amount of FLASH and no memory protection	Security Microvisor deployed via JTAG or similar to create privileged virtual ROM
mid	limited memory management such as a memory protection unit	Hybrid approach using availability security mechanisms, reduction in privileged code, and trust or verification for remaining privileged code
high	memory management unit and / or Trusted Execution Environment such as ARM TrustZone or Intel SGX	Leverage TEE hardware to create compatible RoT

In the rest of the paper we describe details of the SEDIMENT methodology and system ingredients: the software root of trust in Section II, the software remote attestation in Section III, and the secure data publication in Section IV.

II. SOFTWARE ROOT OF TRUST IN SEDIMENT

A critical component of secure remote attestation and secure data publication is to be able to keep the attestation/encryption functions and associated key(s) in a protected and immutable memory region via Memory Isolation and Protection or the Root of Trust (RoT). RoT is defined (in [5]) as "highly reliable hardware, firmware, and software components that perform specific, critical security functions." RoT provides a firm foundation to build security services, such as RA, secure boot, secure storage of cryptographic keys and code. Typically RoT is built into the silicon of microcontrollers or application processors and is not accessible directly from outside of the chip. However, most IoT devices either have no TPM or available/enabled TrustZone (due to cost and complexity). In SEDIMENT, to deal with this, we develop a minimalist Software Root of Trust (SRoT), to establish trust over a zero-trust network. The details of the SRoT are constrained by the support available on the device. For this purpose, we categorize IoT devices into three tiers, as shown in Table II. SEDIMENT supports all three tiers.

On *low-tier IoT devices*, we have developed and demonstrated the use of a software-based memory isolation hypervisor called Security MicroVisor (SμV) to address the RA needs of the low-tier IoT devices such as Arduino Uno. SμV is a lightweight, open-source software-based memory isolation technique [6], consisting of 510 lines of code, and occupying 1,070 Bytes of Flash. SμV turns part of the physical non-volatile memory into a virtual ROM, isolated and not accessible from untrusted application software. In particular, user applications are constrained to operate within their own subset of physical memory and cannot tamper with secrets, cryptographic data, and security-related functions, which are stored in the virtual ROM. SμV is installed prior to the deployment of the IoT device using a physical programming device, e.g., JTAG, before loading any user application. To create applications that do not violate memory isolation, a developer uses an extended toolchain consisting of a cross-compiler, linker, and standard libraries to produce the executable binary file. SμV provides memory protection by assembly-level code validation at load time and selective software virtualization, where the toolchain allows for replacing certain inherently unsafe instructions by safe virtualized instructions in the SRoT.

In recent work [7], our team has formally verified all the critical properties of the SμV code using VeriFast [8] (a separation logic based program verifier) such as atomic and timely execution, memory-safety, absence-of-crashes, and immutability of attestation code. The same formal verification process is applicable to the SEDIMENT code hosted within the SμV, and provides similar safety guarantees. SμV thereby provides a highly efficient and robust mechanism for trust bootstrap in 5G IoT deployments.

On *mid-tier IoT devices* equipped with an MPU but no MMU, SEDIMENT pursues a hybrid approach. Although the MPU can be used to separate security functions and services (e.g. RA, authentication keys, etc.) from other user applications, the MPU can be reconfigured at run time by privileged software. Because of this, bugs in privilege code, such as the operating system (OS), interrupts, event handlers, and RA code itself, could undermine the security of the RoT. In particular, the popular ARM Cortex-M4 design (present in the EFM32 Pearl Gecko and other popular IoT platforms) is not a sufficiently strong security architecture for SEDIMENT.

To fill this gap, our RoT design goal is to minimize privileged software. Thus, ISRs do not need privileged access since they are user dependent applications. However, other handlers such as fault handlers need privileged access. Depending on their complexity, they can either be trusted or formally verified.

We have been investigating two design approaches: (1) to implement the RoT on bare metal to keep minimum privileged software; (2) to use FreeRTOS, a minimal RTOS kernel, which has a dispatcher routine to "defer" any interrupts in user space. Our prior results, obtained with EFM32 Pearl Gecko with ARM Cortex-M4, showed that the idea is possible. However, it requires heavy reliance on the FreeRTOS kernel, which is undesirable. We investigated an alternative using small footprint MultiZone IoT firmware, optimized for ARM Cortex-M processor. MultiZone has a built-in TEE providing hardware-enforced separation of execution environments to isolate trusted services (such as RA) from untrusted third party libraries, as well as other less trusted applications.

We have developed a prototype RoT on a STM32F767ZI (aka ST NUCLEO-F767ZI) MCU with an ARM Cortex M7 core with MultiZone TEE. The RoT is implemented in Zone 3,

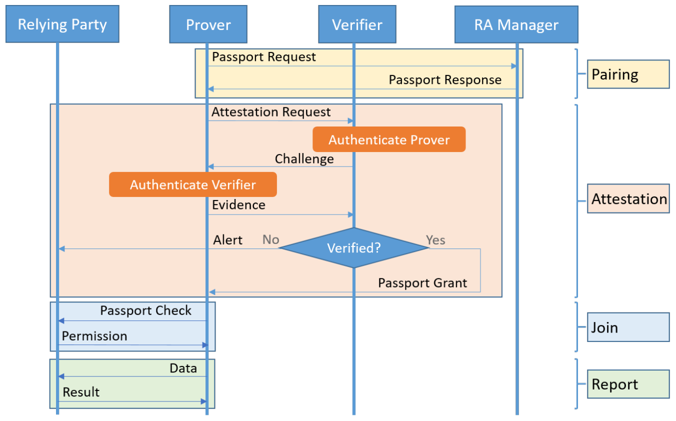

Fig. 2. SEDIMENT remote attestation procedure and message flows

in a MultiZone TEE separated from Zones 1 and 2 running other user level applications. Zone 3 is configured to access other zone's memory as read-only so that the RoT in Zone 3 can perform attestation of memory in other zones, but without the ability to modify that memory. This allows a verifier to attest Zone 1 memory via the RA software in Zone 3.

For *higher-tier IoT devices*, we are exploring the possibilities of leveraging their existing TEE supports, such as ARM TrustZone. For systems with more advanced OS with virtual memory, we are also looking into broadening the scope of attestation challenges, and investigating additional attestation aspects to ensure secure operations.

III. Software Remote Attestation in SEDIMENT

To help determining whether the intended, trustworthy software is running on IoT devices, SEDIMENT uses a *Software Remote Attestation Protocol* (SRAP) component to address this issue across heterogeneous nodes and networks. Remote Attestation (RA) is an interaction between a trusted verifier and one or more remote and potentially compromised devices. RA provides a means for the remote device to prove that it is running the intended software rather than a software compromise due to malware, configuration errors, or other problems. SRAP is effective on low-end microcontrollers that cannot support expensive security mechanisms efficiently.

SRAP Design. Figure 2 illustrates SEDIMENT's generalized SRAP procedure with modulized phases among RA logical entities. Its core attestation steps build upon our prior work called SIMPLE [7]. We choose the Prover to initiate the attestation phase, instead of continuously keeping a communication channel open waiting to be attested, so to minimize energy consumption needed to support RA tasks, particularly for low-end IoT devices. We also additionally use a relying party and the Passport model, using terminology from the IETF Remote ATtestation ProcedureS (RATS) Working Group [9].

Entities participating in SEDIMENT's SRAP include:

- *Prover:* runs on a targeted IoT End Device to supply evidence concerning properties of the target; evidence is gathered by means of a Network Endpoint Assessment, which can, e.g., compute HMAC of static and dynamic memory contents as an evidence of software state at the time. With various hardware/software Root of Trust (RoT) supports, the memory region of each IoT device conceptually has two separated areas: attestation and encryption codes, plus their associated keys, reside in a protected privileged area, whereas application code and data are loaded in an application area subject to malware attacks or unauthorized code/configuration changes.

- *Verifier:* runs on a RA Procedure (RAP) server to make an attestation decision about the well-being of a targeted IoT end device by evaluating supplied evidence against the Verifier's knowledge of the properties of the target.
- *Relying Parties:* these are entities that "rely" on attestation results to decide whether to interact with or admit traffic from attested IoT devices or to blacklist/whitelist their application data. Examples of such entities include application servers, and network access filtering servers.
- *RA manager:* serves in an administrative capacity to match Provers with Verifiers. It also includes a Key Manager component to generate and distribute authentication and attestation keys to each Verifier-Prover pair as needed.

Tasks involved in SEDIMENT's SRAP phases include:

- *Pairing:* An RA Manager entity pairs up a Verifier with a Prover dynamically to carry out attestation. Prover initializes the procedure by sending a Passport Request message to a designated RA Manager to authenticate itself and to acquire Verifier connection information. The RA Manager responds with a Passport Response message to supply connection information, including the authentication/attestation keys to be used by the pair. Note that this is an optional phase, and can be skipped if the pairing is statically administrated prior to the deployment of IoT devices, and their authentication/attestation keys need not be changed from time to time.
- *Attestation:* The attestation phase proceeds with Prover initiating the message exchanges. Prover sends Attestation Request, receives challenge, and produces evidence. Verifier evaluates evidence, and then either grants a passport to Prover or sends an Alert to Relying Party. Authentication digests signed with the shared authentication key enables both parties to authenticate.
- *Join:* Once Prover receives a granted passport, it submits the acquired passport to Relying Party to gain permission to communicate. If the submitted passport is not accepted, or if a Permission message is not received after n tries, Prover restarts the attestation procedure. Permission message may indicate a different Relying Party for Prover to interact with during the Report Procedure.
- *Report:* The IoT end device can start sending application data after Prover received a Permission message indicating the submit passport has been accepted. The IoT end device may need to retransmit if it did not receive a positive Result acknowledgment in time. Prover needs to restart the Attestation procedure if the IoT end device receives a negative Result or if it failed to receive any Result message after K tries.

SRAP Correctness, Security and Performance. Since both the attestation protocol and secret keys reside inside the secure memory area of SμV, we are able to prove all the critical properties of the attestation code. Given the bounded execution time and memory-safety properties, the functional correctness of SRAP is also provable and ensures that the attestation code completes correctly while maintaining keys confidentiality. SRAP requires $\leq 80B$ of permanent storage. The verified attestation code occupies about 6kB of Flash memory, with an additional 1.5kB of temporary storage (RAM) required to support attestation reports. On a representative IoT device (Arduino Uno ATmega 328p microcontroller running at 16MHz), the measured execution time was very small (2.64 secs to verify the integrity of 32kB of memory). It is thus extremely efficient in terms of storage, computation (and battery power), and thus is an excellent choice as a first-line defense in detecting and blocking malware implants. We also obtained preliminary performance measurement of the current prototype on the STM32 board, which has a Cortex-M7 processor with 16MHz frequency. Results are encouraging, as it takes 440ms to attest 32kB of RAM, and 60ms to attest 4kB, showing the possibility of a small-size trusted computing base without relying on unverified kernel code, as with EFM32.

IV. Scalable Secure Publications from Efficient Attribute-based Encryption Schemes

We considered the problem of scalable secure publications from IoT devices for distribution in a 5G network and consumption by a varying set of subscribing application servers. In the SEDIMENT methodology, we designed a system satisfying the following (sets of) requirements: (1) a publish-subscribe approach to state-of-the-art secure data encryption and expressive decryption access authorization; (2) use of cryptography algorithms for encryption and authentication/signing on resource-constrained devices; and (3) use of encryption and authorization solutions following zero-trust principles [4]. There are a number of systems in the literature that address a subset of these requirements. Among these, we selected the JEDI system [3] as our starting point, and applied to it various improvements and enhancements to address all of our requirements, as we discuss in the rest of this section.

Publish-subscribe encryption. In addition to resource-efficient symmetric encryption (e.g., AES-based encryption), SEDIMENT uses many-to-many, 'label-based', asymmetric encryption to encrypt AES keys, so that a ciphertext can be secured and published for any receivers subscribing to a particular content label (e.g., a topic, a hierarchy URI, a validity time interval). Asymmetric encryption schemes in the literature which could be used for this purpose include identity-based encryption [10], searchable encryption [11], and attribute-based encryption [12]. The SEDIMENT design can be based on any one of these encryption primitives, together with a (say, cloud-based) message queuing repository, which can store

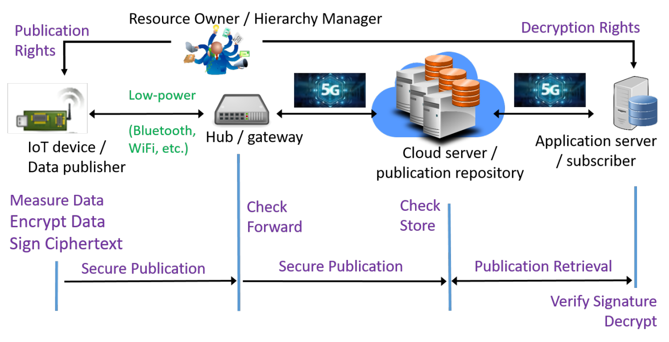

Fig. 3. SEDIMENT scalable encryption protocol flow

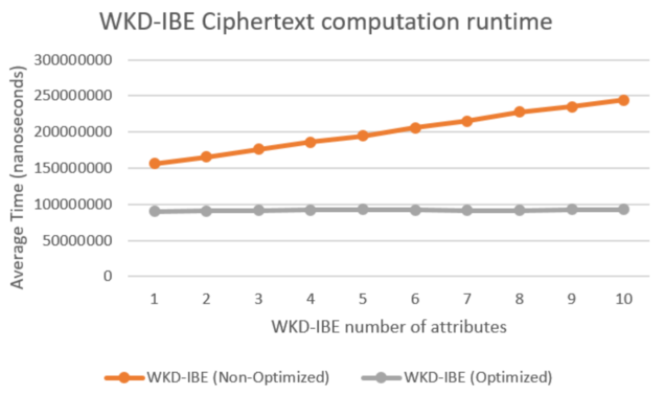

Fig. 4. Runtime of optimized wkd-ibe ciphertext computation on RPi0

secure publications and make them available to subscribers for pull-based (or even push-based) retrieval (see also Figure 3).

Cryptography for resource-constrained devices. To achieve a desirable balance between efficient encryption runtime and expressive decryption rights, in SEDIMENT we use wkd-ibe encryption [13], a type of attribute-based encryption with a number of desirable properties, including hierarchical and decentralized delegation of decryption rights. In JEDI [3], the authors remarkably showed that wkd-ibe encryption can be deployed on a number of IoT devices. In SEDIMENT we continue their work, and complement it with some enhancements to further reduce IoT device runtime and energy consumption.

One of our enhancements is on the key revocation mechanism. In [3] the authors implement a key revocation mechanism for stateless receivers which requires increased runtime and length of wkd-ibe ciphertexts, both impacting the IoT device energy consumption. Instead, we require the most expensive parts of the key revocation algorithm to be run by a power and computation rich MEC node, which updates only one of the attribute fields used by the wkd-ibe encryption algorithm, keeping a session nonce, for both IoT devices and non-revoked subscribers. This requires a much lower-order increase in runtime on the IoT device's wkd-ibe ciphertext.

Another enhancement is on ciphertext precomputation and adjustment (based on precomputed ciphertext material). The most expensive part in computing a wkd-ibe ciphertext consists of computing a number n of additions of scalar multiplications in an additive elliptic curve group, where n denotes the (constant) number of attribute fields. While a wkd-ibe ciphertext computation requires about $n + 3$ such multiplications, in [3] the authors implement a method for one-time precomputation and periodic adjustment of precomputed data (as the attribute fields slightly change on every data session) that only results in an amortized number of about 4 multiplications. We further studied this problem of selecting efficient precomputed summation values, so to later minimize the amortized amount of additional work (in terms of algebraic group multiplications) needed for summation adjustment at each data session, where the amortization is carried out across multiple sessions. We achieved the following results:

- If precomputation can store a single fixed summation, the average number of multiplications per encryption performed during adjustment reduces from $n+3$ to $n+2.59$.
- If precomputation can store a single variable summation, adjustment requires an average of about 4.04 multiplications per encryption.
- If precomputation dynamically updates its storage of 4 (variable) summations, the average number of multiplications per encryption in adjustment reduces to about 4.
- Finally, we consider a precomputation algorithm where we precompute and store a larger number of multiplication results (requiring less than 5KB for some small yet practical values of constant n). Here, the average number of multiplications per encryption reduces to about 3.02.

We have implemented some of these optimizations on a Raspberry Pi0 and obtained the performance described in Figure 4. In particular, the runtime dependency on the number of attribute fields becomes essentially constant (it is actually still linear, but the increase is so slow that it is not noticeable).

Support of zero-trust principles. In SEDIMENT, support of zero-trust policies [4] is addressed in these main directions: the security (in terms of confidentiality, integrity protection and authenticity) of publication data issued by publishing IoT devices, and the authorization and revocation of decryption access for subscribing application servers. The security of publication data is addressed by end-to-end encryption and authentication. The authorization and revocation of decryption access for subscribing application servers is addressed by resource-owner decryption authorization and revocation, and hierarchy access structures. Both are achieved through our optimized wkd-ibe encryption and signing.

V. Conclusions

SEDIMENT supports the Operate Through concept by means of a zero trust architecture, with end-to-end encryption, signing, and end-point attestation protocols that assume no trust relationships with the underlying 5G network. SEDIMENT focuses on IoT devices, operating across the entire scale of devices on a 5G network, with special emphasis on resource-constrained endpoints, and takes advantage of Multi-access Edge Computing (MEC) resources available to reduce the remote attestation and cryptographic computation burden on the IoT devices. SEDIMENT leverages the concept of decoupled senders and receivers to scale up many-to-many communications security from the JEDI [1] research effort and enhances its data encryption, data signing and key revocation mechanisms to improve the battery lifetime of the IoT devices.

SEDIMENT establishes trust over a zero-trust network, where most IoT devices either have no Trusted Platform Module or available/enabled TrustZone, by means of a minimalist software root of trust, able to easily operate within significantly resource constrained IETF class 1 (10kB RAM, 100kB Flash) IoT devices [5]. SEDIMENT utilizes the formally verified Security MicroVisor (SμV) [6], a lightweight, open-source software-based memory isolation technique. SμV turns part of the physical non-volatile memory into a virtual ROM, isolated and not accessible from untrusted application software. In particular, all secrets, cryptographic, and security-related functions are stored in the virtual ROM, and cannot be tampered with by any loaded untrusted user application. SμV is installed prior to the deployment of the IoT device using a physical programming device, e.g., JTAG [14], before loading any user application.

SEDIMENT employs a software attestation scheme for resource constrained IoT devices with guaranteed security properties. In contrast to the prevailing state-of-the-art in remote attestation, SEDIMENT verifies an IoT device's software integrity without relying on hardware support while protecting against verifier impersonation, denial-of-service (DoS) and replay attacks. SEDIMENT accomplishes this by deploying its attestation protocol as statically allocated code built atop SμV, with both the attestation protocol and secret keys residing inside the secure memory area of SμV. It is extremely efficient in terms of storage, computation (and battery power), and thus is an excellent choice as a first-line defense in detecting and blocking malware implants.

SEDIMENT includes specialized resource efficient cryptography techniques targeted to the most power, memory and computation constrained IoT devices. It extends state-of-the-art cryptographic key revocation mechanisms, which tend to place substantial computational burden on the sending constrained IoT device. It does so by building upon the wkd-ibe scheme [13], and providing wider decryption rights expressiveness derived from the wibe scheme [15]. Further, SEDIMENT's repartitioning of the key revocation functionality, transfers the most expensive computational parts of the key revocation algorithm to a power and computation-rich MEC node, thereby reducing the constrained IoT device sender's computational load. This reduces the energy required by the constrained device in supporting volatile group additions and deletions. Our modeling and analysis results indicate that as the group structures grow larger and join-and-leave dynamics more volatile, the improvements from our approach grow more pronounced. In a penetration scenario requiring frequent key revocations with a group size of 1024 and 8% group-membership change rate, we achieved a factor of five improvement in battery lifetime.

We have implemented SEDIMENT on a variety of representative IoT devices including Silicon Labs Giant and Pearl Gecko, Nordic NRF9160, and Raspberry Pi devices, with RAM as small as 128 Kbytes, and under a variety of IOT OS's including Zephyr, Micrium, and Raspian, and shown less than 10% impact on energy consumption in most cases due the presence and use of SEDIMENT software.

Acknowledgment

This work was supported by the Defense Advanced Research Projects Agency (DARPA), contract number HR001120C0156. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, or the U.S. Government. (Approved for Public Release, Distribution Unlimited).

References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[2] C. Kolias, G. Kambourakis, A. Stavrou, and J. M. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[3] S. Kumar, Y. Hu, M. P. Andersen, R. A. Popa, and D. E. Culler, "JEDI: Many-to-Many End-to-End encryption and key delegation for IoT," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019.

[4] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," *NIST Special Publication 800-207*, 2020.

[5] R. Ross, V. Pillitteri, G. Guissanie, R. Wagner, R. Graubart, and D. Bodeau, "Enhanced security requirements for protecting controlled unclassified information: A supplement to nist special publication 800:171," *NIST Special Publication 800-172*, 2021.

[6] W. Daniels, D. Hughes, M. Ammar, B. Crispo, N. Matthys, and W. Joosen, "Sμv - the security microvisor: A virtualisation-based security middleware for the internet of things," in *Proc. of 18th ACM/IFIP/USENIX Middleware Conf.: Industrial Track*, 2017, p. 36–42.

[7] M. Ammar, B. Crispo, and G. Tsudik, "Simple: A remote attestation approach for resource-constrained iot devices," in *2020 ACM/IEEE 11th Internat. Conf. on Cyber-Physical Systems (ICCPS)*, 2020, pp. 247–258.

[8] B. Jacobs, J. Smans, P. Philippaerts, F. Vogels, W. Penninckx, and F. Piessens, "Verifast: A powerful, sound, predictable, fast verifier for C and java," in *Proc. of NASA Formal Methods - 3rd Internat. Symposium, NFM 2011*, ser. LNCS, vol. 6617. Springer, pp. 41–55.

[9] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote attestation procedures architecture," *Draft-IETF-rats-architecture-14*, 2021.

[10] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[11] D. Boneh, G. DiCrescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc, of EUROCRYPT 2004*, ser. LNCS, vol. 3027. Springer, pp. 506–522.

[12] A. Sahai, B. Waters, and S. Lu, "Attribute-based encryption," in *Identity-Based Cryptography*, ser. Cryptology and Information Security Series. IOS Press, 2009, vol. 2, pp. 156–168.

[13] M. Abdalla, E. Kiltz, and G. Neven, "Generalised key delegation for hierarchical identity-based encryption," *IET Inf. Secur.*, vol. 2, no. 3, pp. 67–78, 2008.

[14] R. Buskey and B. Frosik, "Protected jtag," in *Proc. of 2006 Internat. Conf. on Parallel Processing Workshops (ICPPW'06).*

[15] M. Abdalla, D. Catalano, A. Dent, J. Malone-Lee, and N. Smart, "Identity-based encryption gone wild," in *Proc. of ICALP 2006*, ser. LNCS, vol. 4052. Springer Berlin Heidelberg.