Evaluating and Optimizing Cryptographic Offloading for IoT Devices: Attribute-Based Encryption Case Study

Sunanda Roy, Thomas Crowley, Brian L. Mark, Kai Zeng, and Khaled N. Khasawneh
George Mason University, Fairfax, Virginia
{sroy9, tcrowle, bmark, kzeng2, kkhasawn}@gmu.edu

***Abstract*—Computational offloading can provide significant latency and energy savings. However, most computational offloading protocols have been theoretically proposed, with some empirical evidence supporting their effectiveness. In this paper, we implement and critically evaluate a reference computational offloading protocol for attribute-based encryption on small-scale IoT devices at the network's edge. Our analysis led to the development of an enhanced version of the protocol that further reduces latency and energy usage without compromising the original protocol's security standards. Our results show significant savings of up to 30$\times$ for latency and 32$\times$ for energy consumption using the original protocol, and up to 40$\times$ for latency and 42$\times$ for energy consumption using our proposed enhancement on the end device.**

I. Introduction

A. Research Question

The number of Internet of Things (IoT) devices deployed at the beginning of 2020 is estimated to have been 8.74 billion and is expected to be greater than 25.4 billion by the start of 2030. These IoT devices have low computational power, and offloading complex operations to more powerful devices has become common. Gartner [1] estimates that by 2025, 75% of data will be processed outside the traditional data center or cloud. The move away from the traditional data center is driven by the need to have computing resources closer to where the data is generated, which is referred to as Edge Computing (EC).

EC reduces bandwidth requirements and most importantly reduces latency. The reduced latency of EC enables more complicated real-time to near real-time applications to be fielded utilizing low-power devices. One such application is intelligent transport infrastructure, like self-driving cars [2], [3]. In the example of intelligent transport, the risks are multifaceted and include sensitive data leaks and incorrect offloaded computations to determine future actions. These complex systems require data and computations to traverse and be processed on multiple organizations' infrastructures, further complicating the problem of how to secure the data and computations. Such complex systems would benefit from having a security protocol based on zero-trust architectures.

Attribute-based encryption (ABE) is an important component in zero-trust architectures and is also a complex crypto operation that may be expensive for small-scale IoT devices to perform. Protocols have previously been proposed to enable secure offloading of computations, but most of the existing research is theoretical or solves a very specific problem. There is also a lack of a general framework for implementing the computational offloading solutions for evaluation or real-world usage. An ideal framework would work across a host of platforms and provide an easy way to add primitive calculations and string together existing primitives to form new protocols. This paper presents a Python-based secure offloading framework, universally adaptable across different CPU architectures and operating systems, provided they support Python. It features a flexible design that simplifies the integration of new processing primitives and facilitates their seamless interconnection. The framework is applied to secure computational offloading for ABE.

B. Overview of 5G and EC

5G is the most recent version of the mobile communication standard [4]. Compared to its predecessors, the 5G standard offers much higher data transfer rates and provides seamless communication between devices of varying computational power. However, most small-scale IoT devices suffer from performance issues due to the increased bandwidth of data transfer and the limited hardware resources. Additionally, more data is now exposed to the outside world with weaker security techniques guarding its usage. This has necessitated the development of technologies to increase the performance and resource utilization of IoT devices during the secure exchange of data in a heterogeneous landscape.

The advent of 5G has been accompanied by the generation of large amounts of data as a result of the increased number of IoT devices being connected to the Internet. This has made it increasingly difficult to transport this data to the cloud, given the low latency and resource utilization requirements of IoT devices connected to 5G. The result has been a drop in the Quality of Service (QoS) and response time for associated user applications [5]. EC has been developed to move the burden of processing aggregated data away from the distant cloud infrastructure to nearby devices acting as gateways to the cloud and having greater computational power than the IoT devices capturing these data. These recipient devices are termed as the *edge* devices and the sender devices are termed as the *end* devices. The key factors driving the incorporation of EC into an IoT framework are the increased scalability, security, speed, and efficiency of computations.

C. Edge-based Reconfigurable Cryptography

Reconfigurable cryptography is a well-known technique that offloads the high-overhead functions from a resource-constrained device to an edge device to provide gains in performance and energy efficiency. Prior reconfigurable crypto methods mostly assume that the edge device is trusted. If the edge device is untrusted, one can plausibly apply homomorphic encryption (HE) [6], [7] to enable computation over ciphertext. However, [8] shows that HE incurs an almost four orders of magnitude slow-down even after software optimization. In [9], fully homomorphic encryption (FHE) has been combined with secure multiparty computation (MPC) and a trusted execution environment (TEE) to leverage computations inside multiple untrusted servers when the client is a resource-constrained device. As per their experimental results on Intel's SGX platform which is their hardware-based TEE, the performance suffers from three main drawbacks: latency delay due to encryption and decryption between the enclave and the host machine, the small memory size of the enclave (the area where computations are to be performed) and lack of support for operating system (OS) based libraries or system calls inside the enclave.

We infer that the performance of this application is hindered due to two layers of crypto operations. The first delay occurs during MPC due to the encryption/decryption of chunks of data between one or more clients and multiple untrusted servers and the subsequent combining of multiple decrypted results to get the final result. The second delay comes from the bottlenecks of computing inside the SGX enclave. In this paper, we propose alternative approaches to overcome these appreciable latency delays which can prove to be very costly in terms of energy consumption on a small-scale IoT device.

D. Main Contributions

The main contributions of our work are listed below.

- With the assumption that key management between IoT devices has already been established, we propose a secure data exchange protocol, which aims to minimize the latency and energy consumption of an end device while requesting data from another end device in a shared network.
- We explore the usage of a TEE to offload crypto operations from end devices to edge servers and derive recommendations based on our obtained results.
- We standardize our methods of offloading crypto operations using two different crypto libraries supporting ABE.
- We plan to open source the code for reproducibility.

E. Paper Organization

The remainder of the paper is organized as follows. Section II provides an overview of the key technologies and algorithms that form the backbone of our work. Section III describes our proposed solution for improving the theoretical data exchange protocol described in [10] including the software and hardware details of our experimental setup. The results of our experiments are evaluated in Section IV. Finally, Section V concludes the paper and discusses future research directions.

II. Background

A. Types of Computing: Cloud, Fog and Edge

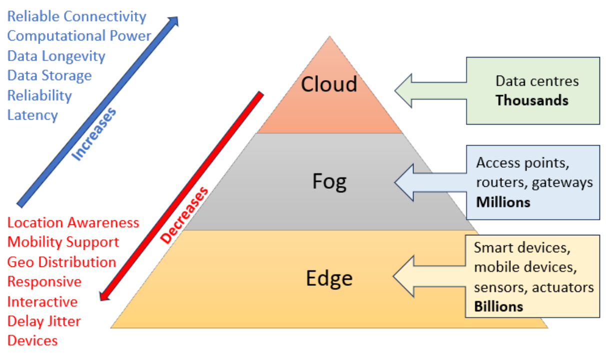

Fig. 1. Three-Layer Architecture of Cloud, Fog, and Edge

This section describes three important computing paradigms with different performance attributes as shown in Fig. 1. While Cloud Computing [11], [12], [13] allows rapid provisioning and release of computing resources to its client applications, Fog Computing [14] minimizes the request-response time from/to its client applications on end devices. Edge Computing [15] pushes this computational layer more towards the edge of the network allowing relatively powerful edge devices to process upstream/downstream IoT data.

B. EC-assisted IoT: Attacks on Data Security

An EC-based framework is vulnerable to numerous types of attacks as shown in [16]. We describe here those attacks that impact the data security of the EC nodes.

1) *Malicious Hardware/Software Injection*: Attackers can inject unauthorized software/hardware components into the EC nodes, to corrupt upstream servers.
2) *Distributed Denial of Service (DDoS)*: This type of attack resorts to the jamming of transmission messages to and from the EC nodes.
3) *Tampering*: The EC nodes are accessed physically to steal sensitive crypto information.
4) *Eavesdropping*: Remote adversaries can covertly listen to EC node traffic and extract their vital details like their configuration and passwords on a shared network.
5) *Side-Channel*: The signals emitted by the different hardware components of the EC nodes can be sniffed by remote adversaries and interpreted to extract sensitive crypto information.

6) *Replay*: Attackers capture data traffic of EC nodes and replace them with malicious data, leading to adverse effects on EC nodes including their battery depletion.
7) *Privacy Leakage*: Attackers can detect the location of EC nodes and use it to extract sensitive information from them, including establishing communication with other EC nodes in that network.

The vulnerability of EC nodes to such attacks is exacerbated by their limited software and hardware capabilities compared to more powerful cloud servers. Thus, it is necessary for security countermeasures to incorporate techniques that minimize the resource usage of IoT devices during their operation.

C. Attribute Based Encryption

ABE [17] overcomes the limitations of traditional public-key cryptography which can provide security permissions at a coarse-grained level only, wherein a user has no way to selectively provide access to its encrypted data to multiple users. In ABE, both the users' keys and ciphertexts are labeled with sets of descriptive attributes so that a particular key can decrypt a particular ciphertext only if there is a match between the attributes of the ciphertext and the user's key. Two sub-types of ABE are mentioned below:

1) Key-Policy (KP-ABE): Each ciphertext is associated with a set of descriptive attributes. Each private key is associated with an access structure (i.e., policy) that specifies which type of ciphertexts the key can decrypt [18].

2) Ciphertext-Policy (CP-ABE): Each user's private key is associated with a set of attributes. Each ciphertext is associated with an access structure that needs to be satisfied by matching users' private keys [19].

D. Trusted Execution Environment

A TEE is an isolated and secure execution environment that guarantees code and data integrity, and data confidentiality. These guarantees are designed to be upheld even when the main operating system has been compromised. The protections are created from a mix of software and hardware components. The idea of secure computing has been around for a long time. In June of 1978, the U.S. Department of Defence established the DOD Computer Security Technical Consortium which had the goal of stimulating the widespread commercial availability of trusted computer systems [20]. In the early 2000s, a virtual machine-based platform for trusted computing, Terra, was proposed. Terra included all of the components of today's TEEs including isolated execution, protected storage, and remote attestation [21]. Remote attestation is the process where a verifier ascertains the current state or behavior of the prover. In TEEs, this action means the prospective user of the TEE's isolated execution environment is obtaining proof in the form of a cryptologic signature that the application running in the TEE is unmodified.

In modern TEEs storage is protected using *data sealing*, whereby any data that needs to be re-used in successive calls is encrypted before it is stored outside of the secure enclave. For example, while an application is running in the TEE, the application's data is secured by the TEE, but when the application exits, the data needs to be destroyed or protected when it leaves the secure enclave [22]. Each of the major hardware manufacturers has their own TEE implementations. The most prominent is Intel's Software Guard Extensions (SGX), which spawned the ancillary, open-source project, Gramine.

III. Proposed Offloaded ABE Protocol

In this section, we first describe the reference offloaded ABE protocol [10] that we implemented. Next, we describe our proposed enhancement of this protocol, while keeping the security level unchanged. Finally, we present a security analysis of our proposed solution to prove the correctness and validity of those steps that differ from the original protocol.

A. Reference Protocol

We considered the reconfigurable ABE protocol described in [10] as our reference for implementing an edge-based offloading framework. In this work, the authors have devised protocols that can help two untrusted small-scale IoT devices mutually authenticate each other followed by securely exchanging data between them. To mutually authenticate each other, they use a group signature scheme based on [23]. For data exchange, they use Diffie Hellman Key Exchange (DHKE) to obtain a shared secret key that can subsequently be used multiple times for encryption/decryption of the shared data. Next, they add an extra layer of fine-grained security by using ABE [18].

The reconfigurable aspect is introduced by incorporating a security agent (SA) near each end device. These SAs are computationally more powerful than the end IoT devices and act as platforms for executing complex crypto operations. Each SA maintains a public key and a unique secret key obtained from the Global Key Management System (GKMS). This allows only those SAs to successfully decrypt the received data, whose secret key satisfies the attribute policy of the encrypted data. The GKMS allows for the allocation and revocation of keys to an SA as per security requirements. During the data exchange process, ABE computations are offloaded to the corresponding SAs, and their results are transmitted back to end devices. With this offloading approach, they report a reduction of the processing time on the end IoT device of approximately 80% for ABE.

B. Enhanced Protocol

Initially, we evaluate the energy savings on the end IoT device while offloading ABE to a powerful edge device. In the first step, we perform ABE on the end device only. We measure the mean execution time and energy consumption and use that as our baseline. The second step consists of establishing the authenticity of the edge device where ABE computations are to be offloaded and provision a shared symmetric key to it for future exchange of secure data. The authenticity of the edge device is performed by doing Remote Attestation (RA) using

Gramine. The overhead due to RA is a one-time overhead only, where the end device verifies the enclave attributes of the edge device with the Intel Attestation Server (IAS) for correctness before offloading computations to the edge device. For simulation only, we assume the shared symmetric key to be stored in advance on both end and edge devices. The final step is to perform offloading of ABE from end device to edge device with any original or generated plaintext being encrypted using AES-256-CBC cipher before transmitting from one device to another. Our enhancement is achieved by removing the second layer of symmetric key encryption after the ABE encryption. Specifically, the ABE output is sent *directly* from SA_i to SA_j, thus removing the three-step path from SA_i to D_i, D_i to D_j, and finally D_j to SA_j.

C. Security Analysis

The original protocol ensured confidentiality between the two end IoT devices by using a layer of symmetric encryption on the data exchanged between them. We notice an anomaly here, wherein the requested data, which has already been encrypted using ABE (to ensure authentication [24], [25] between the end IoT devices), are encrypted a second time by the symmetric encryption algorithm. This overlap of confidentiality and authenticity is redundant and leads to unwanted energy consumption on the end IoT devices. Our justification for the proposed enhancement is that when SA_i sends the ABE ciphertext to SA_j, this step also ensures the confidentiality of communication between the end devices D_i and D_j, the reason being SA_j can successfully decrypt the ABE ciphertext only if its secret key possesses the same attributes with which SA_i encrypted the requested data. Thus our proposed enhancement maintains confidentiality between end IoT devices as well as authenticity between a pair of end IoT devices and SA while reducing the performance overhead on the end IoT devices.

IV. Evaluation of the Proposed Solution

A. Experimental Setup

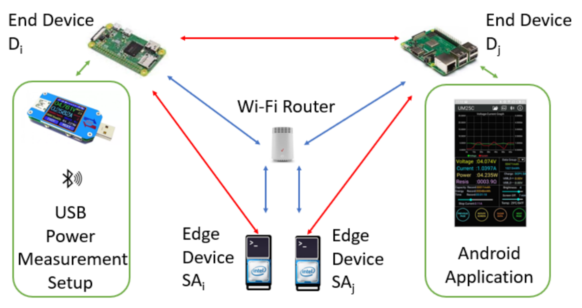
End Device D_i
End Device D_j
Wi-Fi Router
USB Power Measurement Setup
Edge Device SA_i
Edge Device SA_j
Android Application

Fig. 2. Edge-based Crypto Offloading

The experimental setup shown in Fig. 2 consists of hardware, software, and measurement equipment.

TABLE I
End Device Configuration

Device (Pi)	RAM (GB)	Frequency (GHz)	Bus Width (Bits)	Hardware	Number of Cores	Architecture	OS Version (raspberrypi)
Zero W	0.44	1	32	BCM 2835	1	armv6l	6.1.19+
3B+	0.93	1.4	64	BCM 2835	4	aarch64	6.1.21-v8+

TABLE II
Edge Device Configuration

Processor (Intel)	RAM (GB)	Frequency (GHz)	Bus Width (Bits)	Number of Cores	Architecture	OS Version (Ubuntu)
i7-6700	16.12	3.4	64	8	x86_64	20.04.6 LTS

1) Hardware: The details of the two end devices used in our experiment are shown in Table I. The edge devices are two independent processes running inside a single desktop machine whose configuration is shown in Table II. The Wi-Fi Router is a Verizon/Fios Home Router G3100 running Verizon's Open Source Software.

2) Software: We have used two ABE libraries to demonstrate our work. CPABE [19] is the collection of executables of the software implementation of ciphertext policy attribute-based encryption (CPABE) specification. OpenABE [26] is an open-source package of C++ language-based ABE implementation which can be integrated with a parent software package to perform ABE-related operations. To encrypt plaintext using the AES-256-CBC cipher, we have used the PyCryptodome crypto library [27], [28]. We used Gramine [29] to interface our client code (end device) with the SGX instruction set of the server (edge device). Our implementation is written in Python and can be executed on any Linux-based operating system. We have used the TCP protocol to transfer data packets from one device to another and our implementation is single-threaded.

3) Measurement Equipment: We used a UM25C USB Power Meter to measure the mean energy consumed during a crypto operation. The relevant parameters of this device can be found in Table III. We installed the UM25C USB Power Meter Android application to capture the instantaneous power readings and export them to an Excel sheet for calculating the mean energy consumed by a crypto operation.

B. Results

The mean execution time of our implementation for doing RA was 0.53 seconds on the end device. Our results for executing the offloaded ABE and ABD computations inside an SGX enclave showed a huge latency of about 3.5 seconds before receiving the final results from the edge device. Since the end device was idle during this time frame, we excluded this idle energy consumption from our analysis.

TABLE III
USB Power Measurement Configuration

Model	Range		Resolution		Accuracy	
	Voltage (V)	Current (A)	Voltage (V)	Current (A)	Voltage (V)	Current (A)
UM25C	4 - 24	0 - 5	10^{-3}	10^{-4}	± (0.5% + 2 digits)	± (1% + 4 digits)

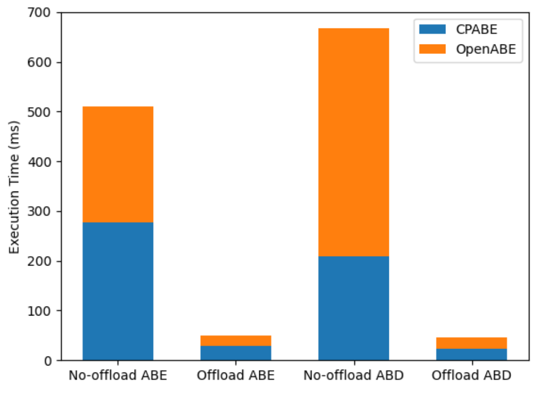

Fig. 3. Execution time of end device for no offloading and offloading of ABE and ABD for plaintext of size 32 B using both libraries.

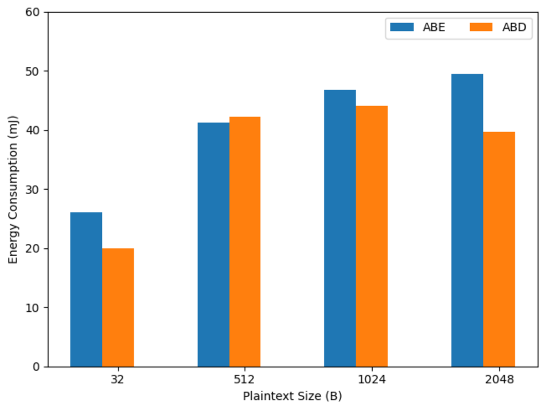

Fig. 5. Energy consumption of end device for different plaintext sizes for offloading using CPABE library.

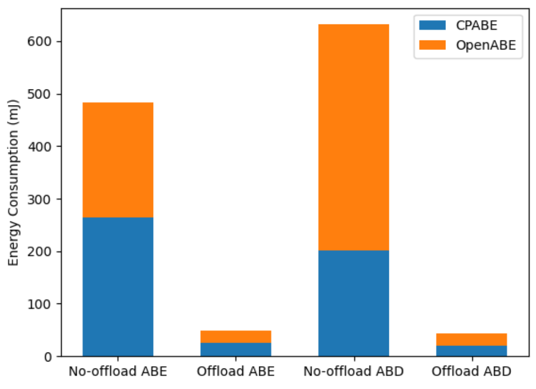

Fig. 4. Energy consumption of end device for no offloading and offloading of ABE and ABD for plaintext of size 32 B using both libraries.

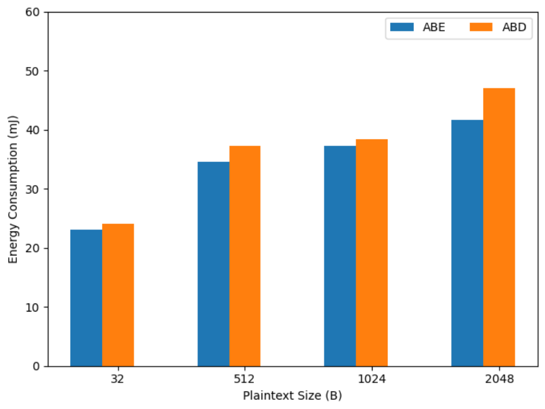

Fig. 6. Energy consumption of end device for different plaintext sizes for offloading using OpenABE library.

From Figs. 3 and 4, we can see the savings in both execution time and energy consumption of about 90% for both ABE and ABD using the CPABE library for a plaintext of size 32 B. Using similar parameters, we can see the savings in both execution time and energy consumption of about 90% for ABE and 94% for ABD using the OpenABE library. Thus, we did not find any major difference in the libraries' energy consumption on the end device. Our results show that OpenABE consumes less energy than CPABE for ABE, while the reverse holds for ABD. When we varied the plaintext sizes while offloading ABE or ABD using both the libraries, we found a steady increase in the energy consumption of the end device as shown in Figs. 5 and 6.

From Figs. 7 and 8, using our proposed solution for data exchange on D_j, we find that overall we can achieve 2.1× savings in both latency and energy consumption during both the crypto operations (ABD and AES-D) and 1.1× savings during the wait time of receiving data from D_i. These savings can be categorized into three areas:

1) ABD – D_j does not have to send AES decrypted ABE ciphertext to SA_j.
2) AES – D_j has to perform only one round of AES-D to retrieve the requested data.
3) WAIT – D_j does not have to wait to receive the ABE encrypted data from D_i before offloading the ABD computations to SA_j.

V. Conclusion

Navigating the complex landscape of security in constrained environments is paramount, especially in the context of the ever-expanding IoT and emerging edge Internet systems. While current cryptographic solutions are designed for robust systems like desktops and servers, they are not feasible for low-power devices, particularly those operating on limited battery resources, such as IoT devices. In this paper, we have made several contributions to address this challenge through an end-to-end implementation, evaluation, and enhancement of crypto offloading using ABE as a case study. Our results show

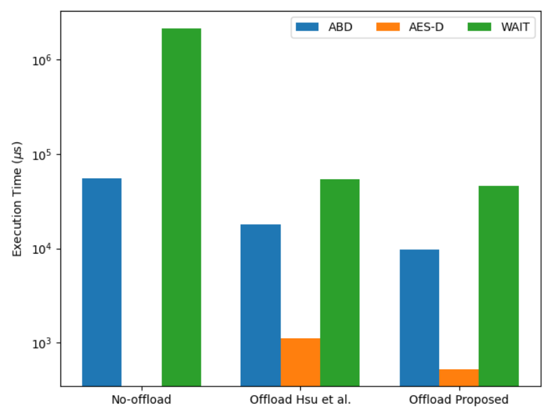

Fig. 7. Execution time of end device D_j for no offloading, offloading using Hsu et al. [10] and our proposed offloading protocol for plaintext of size 32 B using CPABE library.

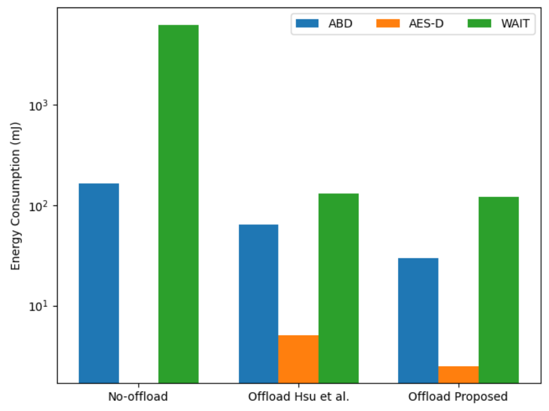

Fig. 8. Energy consumption of end device D_j for no offloading, offloading using Hsu et al. [10] and our proposed offloading protocol for plaintext of size 32 B using CPABE library.

that our proposed enhancement of a reference reconfigurable ABE protocol was able to achieve significant gains in both latency and energy consumption.

VI. Acknowledgement

This material is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) under agreement number HR001120C0154.

References

[1] "Edge computing promises near real-time insights and facilitates localized actions," 2018. [Online]. Available: https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders

[2] A. S. Kabanov, V. N. Azarov, and V. P. Mayboroda, "An analysis of the use and difficulties in introducing information technology and information systems in transport and the transport infrastructure," in *IEEE IT&QM&IS*, 2019, pp. 192–196.

[3] J. Gao, C. Cao, W. Li, and X. Li, "Research and Verification of Risk Prevention Methods for Commercial Vehicles Based on Intelligent Vehicle and Infrastructure System," in *IEEE ICDSBA*, 2020, pp. 191–194.

[4] M. Wazid, A. K. Das, S. Shetty, P. Gope, and J. J. Rodrigues, "Security in 5G-enabled Internet of Things Communication: Issues, challenges, and future research roadmap," *IEEE Access*, vol. 9, pp. 4466–4489, 2020.

[5] S. A. Bhat, I. B. Sofi, and C.-Y. Chi, "Edge computing and its convergence with blockchain in 5G and beyond: Security, challenges, and opportunities," *IEEE Access*, vol. 8, pp. 205 340–205 373, 2020.

[6] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *Cryptology ePrint Archive*, 2012.

[7] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1999, pp. 223–238.

[8] B. Reagen, W.-S. Choi, Y. Ko, V. T. Lee, H.-H. S. Lee, G.-Y. Wei, and D. Brooks, "Cheetah: Optimizing and accelerating homomorphic encryption for private inference," in *HPCA*. IEEE, 2021, pp. 26–39.

[9] C. Yakupoglu and K. Rohloff, "PREFHE, PREFHE-AES and PREFHE-SGX: Secure multiparty computation protocols from fully homomorphic encryption and proxy reencryption with AES and Intel SGX," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2022, pp. 819–837.

[10] R.-H. Hsu, J. Lee, T. Q. Quek, and J.-C. Chen, "Reconfigurable security: Edge-computing-based framework for IoT," *IEEE Network*, vol. 32, no. 5, pp. 92–99, 2018.

[11] P. Mell, T. Grance *et al.*, "The NIST definition of cloud computing," 2011.

[12] H. M. Makrani, H. Sayadi, N. Nazari, K. N. Khasawneh, A. Sasan, S. Rafatirad, and H. Homayoun, "Cloak & co-locate: Adversarial railroading of resource sharing-based attacks on the cloud," in *SEED*. IEEE, 2021, pp. 1–13.

[13] C. Fang, H. Wang, N. Nazari, B. Omidi, A. Sasan, K. N. Khasawneh, S. Rafatirad, and H. Homayoun, "Repttack: Exploiting cloud schedulers to guide co-location attacks," *arXiv preprint arXiv:2110.00846*, 2021.

[14] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. S. Goren, and C. Mahmoudi, "Fog computing conceptual model," 2018.

[15] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[16] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge-computing-assisted internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4004–4022, 2020.

[17] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*. Springer, 2005, pp. 457–473.

[18] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *13th ACM Conf. on Computer and Communications Security*, 2006, pp. 89–98.

[19] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symp. Security and Privacy*, 2007.

[20] P. S. Tasker, "Trusted computer systems," in *IEEE Symp. Security and Privacy*, 1981.

[21] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: A virtual machine-based platform for trusted computing," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003.

[22] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing," in *2nd ACM Int. Workshop on Hardware and Architectural Support for Security and Privacy*, 2013.

[23] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Annual international cryptology conference*. Springer, 2004.

[24] Y. He, K. Zeng, B. L. Mark, and K. N. Khasawneh, "Secure and energy-efficient proximity-based pairing for iot devices," in *IEEE Globecom 2022 7th on 5G and Beyond Wireless Security*, 2022, pp. 1359–1364.

[25] Y. He, K. Zeng, L. Jiao, B. L. Mark, and K. N. Khasawneh, "Swipe2pair: Secure and fast in-band wireless device pairing," in *ACM WiSec*, 2024.

[26] "Openabe," https://github.com/zeutro/openabe.

[27] "Pycryptodome: Python package of low-level cryptographic primitives," https://github.com/Legrandin/pycryptodome.

[28] S. Roy, A. Stavrou, B. L. Mark, K. Zeng, S. M. PD, and K. N. Khasawneh, "Characterization of AES Implementations on Microprocessor-based IoT Devices," in *IEEE WCNC Workshop on Securing and Operating Through 5G*, 2022, pp. 55–60.

[29] "Gramine," https://gramine.readthedocs.io/en/stable/index.html.