Towards Secure and Reliable Distributed Systems with Minimized Trust

Yuan Liang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Haining Wang, Chair
Wenjie Xiong
Angelos Stavrou
Danfeng Yao
Lingjia Liu

August 11, 2025
Blacksburg, Virginia

Keywords: Network Security, Side Channel, Identity-based Encryption

Towards Secure and Reliable Distributed Systems with Minimized Trust

Yuan Liang

(ABSTRACT)

As the most prominent distributed computing platform, the modern Internet infrastructure interconnects various computing resources from data centers to Internet of Things (IoT) devices. Ensuring secure and reliable distributed systems on the Internet is critical for the normal operations of our daily lives. In this dissertation, we conduct three research projects to improve the security and reliability of distributed systems from different aspects.

In the first research project, we investigate the cooling systems of Amazon Web Services (AWS) data centers. We leverage two temperature side channels to capture the information leakage from AWS data centers. These two side channels essentially exploit the temperature effect on FPGAs and DRAMs. After comparing data from both side channels, we believe the information revealed by the data is reliable. This project is a practical application of FPGA-based temperature side channels for the measurement study on data centers. Subsequently, our second and third projects focus on small, resource-constrained devices, like IoT devices, that often provide data to data centers.

Recent research adapts identity-based encryption (IBE) for IoT devices to encrypt messages, and servers are the receivers, but the application inherits the key escrow problem of IBE. In the second project, we propose an interactive protocol among decryptors to tackle it. We assume decryptors like servers have sufficient resources to handle the additional computation and communication costs. Our protocol is based on dhr-IBE (IBE with decentralized setup, homomorphic key derivation, re-encryptable ciphertext), and Boneh-Franklin IBE, Waters

IBE, Boneh-Boyen-Goh IBE are classified as dhr-IBE. The protocol is to build an IBE system as if the master secret key is the sum of all secret keys.

In the third project, we propose an alternative solution that takes a trade-off between security and efficiency into consideration, so that the protocol designer can make the decision. The alternative protocol optimizes linear computation and communication to polylogarithmic complexity, and it can be viewed as a type of registration-based encryption, but it does not protect unregistered users. To develop the new protocol, we extend the dhr-IBE to dhr-HIBE (hierarchical IBE), essentially abstracting properties of Waters HIBE, Boneh-Boyen HIBE, Boneh-Boyen-Goh HIBE. The major technique involves using an O(log n)-size HIBE based tree to minimize the computation and communication, meeting a subset of compactness and efficiency requirements of registration-based encryption. A significant advantage of our protocols is keeping the original encryption algorithm of identity-based encryption for IoT devices. In other words, the sender only needs the constant-size public parameter to encrypt messages. We implement software prototypes to verify the efficiency of our protocols.

Towards Secure and Reliable Distributed Systems with Minimized Trust

Yuan Liang

(GENERAL AUDIENCE ABSTRACT)

The internet has connected various distributed systems from data centers to Internet of Things (IoT) devices. It is needless to say, the reliability and security of distributed systems are critical to our daily lives. Frankly, security could be considered as one part of the reliability. However, the security research emphasizes the existence of a powerful adversary. This assumption is important for security research. It generalizes many practical attack scenarios in applications. Moreover, we value discovering problems in real-world distributed systems and propose practical solutions for applications. In general, we conduct our research with these principles. Our investigation of data center cooling systems expose the temperature information leakage. It reveals that the temperature environments of some Amazon Web Services (AWS) data centers are strongly related to local weather because they have adapted free cooling systems. The adversary may adapt a thermal attack strategy which uncovered by previous research to create a hostile thermal environment for computing equipment. In the rest of this dissertation research, we propose protocol-based solutions for the two-decade-long key escrow problem of IBE, addressing both security and efficiency. It is a single point of failure problem. The key escrow possesses the secret to decrypt all possible ciphertexts. In other words, the adversary may corrupt the key escrow to take control of the whole system. Our protocols distribute the secret of key escrow to every decryptor, so that the adversary

needs to corrupt all decryptors to take over the whole system. As our protocols introduce efficiency problems related to computation and communication, we develop effective solutions to remedy them.

Dedication

Dedication This dissertation is dedicated to my parents, Bing(Robin) Liang Jianjie Zhang, and my sister Xindan Liang for their love and endless support throughout my academic journey.

Acknowledgments

First of all, I would like to express my deepest gratitude to my Ph.D. advisor, Prof. Haining Wang, for his invaluable guidance and support. His dedication to research and academic rigor has profoundly influenced my career and life. Without his help, this dissertation would not have been possible. Next, I would like to thank Prof. Wenjie Xiong, Prof. Angelos Stavrou, Prof. Danfeng (Daphne) Yao, and Prof. Lingjia Liu for serving on my dissertation committee and providing valuable feedback and constructive comments on. Finally, I would like to express my gratitude to my collaborators, including Dr. Giovanni Di Crescenzo, Dr. Xing Gao, and Dr. Kun Sun for their invaluable support and contributions to make my Ph.D. journey a unique experience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today, the security and reliability of distributed systems, including data centers and Internet of Things (IoT) devices, are critical for our daily lives. Data centers are the backbone of computation-intensive applications, and IoT devices provide critical services such as infrastructure and medical monitoring. In this dissertation, we conduct a measurement study on data center cooling systems using temperature side channels. Subsequently, we observe recent research adapting identity-based encryption for IoT devices, which led us to design a protocol to solve the key escrow problem of identity-based encryption. Finally, we improve the efficiency of the previous protocol by sacrificing some of its security. We leverage a tree-based technique with HIBE to tackle the inefficiency.

Amazon Web Services (AWS) data centers are distributed across diverse locations to provide cloud computing services. Their security and reliability are critical for the global-scale services. Each data center has computing resources including CPU, GPU, FPGA, etc. These are abstracted by many layers of software and provide computing power through the Internet. Power and cooling infrastructures are critical to support electronic hardware. Commercial data centers are reluctant to afford redundancy to counter malicious workload increase. An adversary may take advantage of this security weakness.

IoT devices are for small-scale applications e.g. smart home. Their security is essential for users' privacy. Although there are many cryptographic schemes that can protect the message, few of them can achieve efficiency and expressiveness at the same time. Identity-

based encryption is a well-fitted scheme except for its key escrow problem. Currently, IBE is a four-decade-long research topic since it was proposed by Adi Shamir in 1984. There are many IBEs from different assumptions and many variants of IBE. To address this problem, we distribute the key-derivation capability to all users. However, our solutions are less efficient than the original key derivation of IBE. We propose two solutions that allow the protocol designer to make the decision on the trade-off between security and efficiency.

Our techniques involve hardware, software, and math. We first investigate data centers with physical side-channel. The physical side channel is a type of information leakages in the implementation of the system. Nowadays, FPGAs have been deployed in many AWS data centers, so their side channels attract much research attention. Recent research on FPGA-based side channels moves the focus from the lab setting to the cloud setting. We take a step further that leveraging the FPGA-based temperature side channels to investigate the cooling systems of AWS data centers. To ensure the information from side channels is accurate, we leverage two mature temperature side channels. If they produce consistent temperature information, their implied temperature information is believable. In the second project, we are interested in the key escrow problem. To construct our protocols which distribute the key generation to all users, we add three additional algorithms to IBE to capture the homomorphism property of IBE and the re-encryptability of ciphertext for security proof. We classify these IBEs as dhr-IBE and check that Boneh-Franklin IBE, Waters IBE, Boneh-Boyen-Goh IBE satisfy the property. With black-box use of dhr-IBE, we construct the no-server IBE (ns-IBE) and minimial-server-trust IBE (mst-IBE) protocols. In these protocols, each user contributes to the generation of all the key of the other users, and all communication is public. Our protocols have minimal effect on the IBE encryption, so that the IoT sensors still only encrypt messages with IBE. Our third project considers HIBE to construct an variant with better efficiency and degraded security. In the last project, we enhance protocol

efficiency at the expense of security. A efficient communication/computation means it can be done in polylogarithmic time/memory complexity. We define the extension of dhr-IBE called dhr-HIBE. We use dhr-HIBE as the basis to construct a variant of registration-based encryption which achieve polygorarithmic efficiencies but does not protect unregistered users (unregistered users' ciphertexts can be decrypted trivially with the information in the key curator).

1.1 An investigation on data center cooling

As data centers have expanded their scales with more powerful servers to meet the increasing service demands, the amount of heat emitted by those servers also significantly increases and requires the cooling system to prevent overheating[61]. Improving the efficiency of data centers and using most of energy for computing are important[49][66]. The power and cooling systems have played a major role in the total cost of ownership (TCO) of large-scale data centers, which motivates data centers to adopt the power over-subscription and aggressive cooling strategies for cost reduction. Nowadays, there are various technical solutions for data center cooling systems[2][27]. However, although there are significant improvements, they are still far from the ideal due to many factors like the non-linearity of air dynamics. More importantly, warehouse-scale data centers keep their inside information private for security reasons[6][7][49]. Thus, it is difficult to comprehensively investigate their cooling systems.

In this project, we take advantage of the temperature information leakage through side channels to learn the cooling system inside a data center without privileged accesses. We choose physical side channels of the FPGA (Field Programmable Gate Array) to investigate AWS (Amazon Web Services) data centers. As the cloud FPGA becomes popular, covert/side channels research on FPGAs has been intensive. However, when more and more FPGA-

based covert/side channels are identified, some of them are not practical for remote attacks on current data centers[71]. In addition, most of the existing works focus on information leakage among different tenants. Based on previous research, we choose the DRAM and the time-digital converter (TDC) as tools for the temperature estimation. They are used to measure the temperatures of data centers that power the public cloud.

To implement a precise and high-resolution FPGA-based side channel in order to sense a temperature change, we introduce the spatial average technique for TDCs. It recovers power ripple precisely so that the power ripple can be filtered out, and we can observe the temperature effect on the signal propagation in the FPGA. We also use this technique to measure the switching frequency and transient response of the power system of AWS FPGAs, and we demonstrate the process variation of AWS FPGAs. Leveraging the FPGA-based temperature side channel in TDC and DRAM, we measure the daily temperature changes of AWS data centers worldwide (in region-level and zone-level) that provide the FPGA service. We analyze the collected temperature data, and we observe that temperature changes of some AWS data centers are highly related to local weathers, which evidences that these data centers have adopted the free air cooling technique for cost saving and environmentally friendly purposes. Moreover, the behaviors of the cooling system and temperature dynamics caused by computing equipment inside data centers are studied. Finally, we discuss the threats of power/thermal attacks and cloud cartography on the WSC (Warehouse-scale computing) data center.

Operating global-scale commercial data centers is costly, as they require substantial resources to deliver services worldwide. Achieving perfect security is virtually impossible. While reducing operational costs can improve competitiveness, it may compromise reliability, creating a challenge for operators to balance security with profitability. For example, in the context of thermal management, safeguarding the cooling system and maintaining stable temperatures

are critical to preventing permanent data loss and hardware failures. In addition, advanced security measures—whether in software or physical settings—can help prevent information leaks from computing equipment. Data centers also aggregate data from numerous small-scale computing devices; security considerations for these sources will be addressed in the next project.

1.2 Efficient Identity-Based Encryption with Minimal Server Trust

Unlike data centers, small-scale computing devices like Internet of Things (IoT) devices are designed to serve end-user applications rather than provide large-scale computing or storage capabilities. Their business model is fundamentally different, as they are typically marketed and deployed as end-user products. In many applications, these devices function primarily as data sources. A core security requirement is the use of cryptographic techniques to safeguard data from the moment it is generated, ensuring its confidentiality and integrity throughout transmission and storage.

Since IoT devices are resource-constrained and often run as data sources, it is desirable to decouple them from receivers. Identity-based encryption offers an effective solution for securing communications, as its flexibility enables an IoT device to encrypt data for a specific receiver or a defined group of receivers. The key-derivation challenge inherent in IBE, often considered a single point of failure, this can be mitigated by delegating this process to the receiver side. Receivers, such as modern personal computers or data centers, typically possess the computational capacity to perform the more resource-intensive operations required.

Shamir proposed the notion of identity-based encryption (IBE) and identity-based signatures

in [80], as an approach to simplify key management in public-key cryptography. He showed some constructions for identity-based signature schemes but left the construction of an IBE scheme as an open problem. The first constructions of IBE schemes were proposed by Boneh and Franklin [15], and Cocks [25]. Since then, much research has been produced on IBE schemes (and their many extensions and generalized notions), and other uses have been suggested (including various types of one-to-many and label-based encryption). Despite that, however, there is a big gap between the amount of research produced and the number of real-life applications actually using IBE schemes (one exception being the JEDI system [67], which uses some type of IBE schemes to allow sensor data encryption to many recipients in the IoT applications). One often-mentioned obstacle to the real-life adoption of such schemes is the amount of trust in the private-key generator (PKG) server (a.k.a., key derivation server), since the latter has enough key material to decrypt ciphertexts and forge signatures associated with system users (a.k.a., the key escrow or server trust reduction problem). Although some approaches have been proposed to remedy this problem [15, 20, 74, 51, 52, 65, 23, 38, 39, 50], as we further discuss below, none of these seems satisfactory enough, and more proposals are needed.

In this project, we propose to enhance IBE schemes with key derivation distributed across receivers, where the key derivation capability is moved from the key derivation server to the decrypting receivers while targeting no or minimal modification to the encryption and decryption algorithms. This preserves the efficiency of encryption algorithms, as motivated by a large class of IoT applications, where resource-constrained IoT devices encrypt their data and much more computationally powerful data centers and/or application servers can afford somewhat increased costs in the key derivation.

This project demonstrates that its protocol is a complete solution for the single-point-of-failure problem in IBE. However, the protocols still run with polynomial complexity among

receivers. This inefficiency may cause the protocol to be unsuitable for many other applications.

1.3 Sender-Efficient Registration-Based Encryption in the Public-Parameter Model

The third project substantially advances previous protocols by reducing the computational and storage complexities of several performance metrics to polylogarithmic levels. The design is inspired by registration-based encryption(RBE), a recent scheme introduced to address the key escrow problem inherent in identity-based encryption(IBE). While RBE serves as an alternative to IBE, it requires additional interaction between parties. Our protocol retains the encryption algorithm and public parameter model of IBE, enabling it to satisfy a subset of RBE's efficiency properties while preserving the practical and efficient encryption algorithms of IBE, which makes it suitable for resource-constrained IoT devices.

The fundamental notion of IBE was proposed and left as an open problem by Shamir in [80], as an approach to simplify key management in public-key cryptography. The first constructions of IBE schemes were proposed by Boneh and Franklin [15], and Cocks [25]. Since then, much research has been produced on IBE schemes (and their many extensions and generalized notions), including hierarchical IBE (hIBE) [57, 40], which generalizes IBE to a hierarchy tree of key derivation servers, where each node can use its secret keys to derive decryption keys for its children. Despite the much attention to IBE and its extensions, however, there is a big gap between the amount of research produced and the number of real-life applications actually using IBE schemes. One often-mentioned obstacle to the real-life adoption of such schemes is the amount of trust in the private-key generator (PKG) server (a.k.a.,

key derivation server), since the latter has enough key material to decrypt ciphertexts and forge signatures associated with system users (a.k.a., the key escrow or server trust reduction problem). Even if some approaches have been discussed in the literature to mitigate this problem [15, 20, 74, 51, 52, 65, 23, 38, 39, 50], which we also later discuss, none of these seems practical enough, and new approaches are being recently proposed. One such notion, called RBE replaces the key derivation server with a key curator that does not keep secret keys capable of decrypting other users' ciphertexts, but only curates the updates of the shared public parameters and of the decryption keys, as multiple receivers register into the system. The concept of RBE was proposed in [38] precisely with the motivation to solve the key escrow problem of IBE and in the same article, various efficiency requirements were proposed. Early works of RBE [38, 50], put much emphasis on the asymptotic efficiency of the receiver's operations as a function of the number of receivers, and so the encryption algorithms end up heavily relying on (usually runtime-inefficient) non-black-box cryptography. We are only aware of two RBE schemes that address some sender efficiency requirements: the work in [44], but is based on specific pairing-based operations, and requires an $O(\sqrt{n})$-size common reference string; and the work in [31], where however the ciphertext size is quadratic in the identity length. Except for this latter chapter, all known RBE constructions work in the common reference string model, which adds the additional trust assumption of the correct generation of this string. Specifically, private information associated with this string might be subject to subversion attacks or require a separate secure multi-party computation protocol.

In this project, we consider avoiding these shortcomings of past RBE constructions by addressing the problem of constructing an RBE scheme that is sender-efficient (in particular, addressing metrics motivated by Internet-of-Things applications, and making a black-box use of known IBE schemes), and that works in the public-parameter model (in particu-

lar, not requiring a non-transparent public reference string model, or any additional secure multi-party computation protocols needed to avoid subversion attacks).

Our contributions. We propose a new, sender-efficient, RBE scheme to eliminate or reduce the reliance on server trust. Our scheme, called seRBE, makes black-box use of a variant of IBE schemes, which we define and call dhr-hIBE. This variant includes hierarchical IBE (hIBE) schemes that satisfy 3 additional properties, the most important being that key derivation comes with a sum homomorphism. We observe that many known hIBE schemes from the literature are dhr-hIBE scheme, and we show this for the scheme in [13].

Our scheme works in the public-parameter model, and the generation of these public parameters is transparent, in that the key curator does not keep any secret keys about these parameters that might help to decrypt ciphertexts.

Our seRBE scheme is sender-efficient in the sense that it satisfies runtime-efficiency of ciphertext computation (i.e., comparable to the dhr-hIBE scheme), short ciphertext size (i.e., constant in the identity length and number of receivers), and short public parameter size (i.e., constant in the identity length and number of receivers). These performance requirements are motivated by Internet-of-Things applications, where typically resource-constrained clients encrypt their data to receiving application servers. Previous RBE scheme targeted a different set of performance requirements, focusing on receiver efficiency, motivated by applications like the anonymous board communication abstraction in [39]. In this line of work, the closest RBE schemes to satisfying our sender-efficiency requirements are the following two works: (1) in [44], the authors constructed an RBE scheme that makes black-box use of prime-order pairings, but required a trusted common reference string of size $O(\sqrt{n})$, where n is the number of receivers; (2) in [31], the authors propose a weakly-efficient RBE scheme in a public parameter model, with ciphertext size quadratic in the identity length.

We also produced a software implementation of our seRBE scheme, where we show practical runtimes for all of its operations. In particular, encryption time takes 2ms on a commodity laptop, without specific optimizations.

The rest of this dissertation is organized as follows. Chapter 2 presents a measurement study on the data center cooling systems using two temperature side channels. Chapter 3 is about a protocol for identity-based encryption to distribute secret keys to all users and communicate in the public authenticated channel only. Chapter 4 details a construction of registration-based encryption from the enhancement of the previous protocol. Chapter 5 summarizes the dissertation and discusses the future research direction.

Chapter 2

An investigation on data center cooling

2.1 Background and Related Work

2.1.1 Data Center Cooling

The design consideration of a data center cooling system mainly includes effectiveness, cost, and reliability. The critical part of a cooling system is building loops to collect the heat and propel it to the outside. CRACs (computer room air conditioners) are typically used to dissipate the collected heat to the outside[27]. The CRAC is effective, but its operational cost is high.

Since a cooling system once costed a large portion of energy consumption in data centers[59], free cooling is an innovative design to minimize the power consumption of the cooling system by taking advantages of the cool air outside. However, free cooling still requires artificial cooling, and the temperature management should consider the overall condition of the data center[48]. Temperature is an important factor in the power consumption of the cooling system and various computing equipment[30]. Understanding the temperature pattern can help us to deduce the conditions of computing equipment. Even if physical accesses to data centers are limited, FPGA-based temperature side channels allow us to remotely investigate

data centers.

Cool weather is applicable for cooling data centers in many places on the Earth. The cold nights and winter seasons provide excess cold air, and so turning on the air conditioning or chiller under such scenarios is wasteful. But using cool outside air is not entirely free[2]. Note that water is used as a heat medium for some advanced designs. First of all, outside air should be processed to remove dust. Secondly, cooling fans are required to transfer the air, or pumps are used to move the water. Finally, when the outside temperature is high, the chiller operates to cool down the air. In the case of freezing weather, the cold outside air should be mixed with hot outlet air to maintain a warm environment for the computing equipment, and water should be prevented from being frozen. Since water cannot work as the heat medium in extremely low temperatures, radiators can be used as coolers. Radiators rely on coolants that are not frozen below water freezing points to transfer heat.

Figure 2.1 shows the elements considered in this study. The temperature condition of a data center is affected by the local weather, the cooling system, and the heat from the computing equipment. FPGAs in the computing equipment leaks the temperature information to a remote adversary, and the further information can be deduced based on the leakage.

2.1.2 Cloud FPGA

FPGA services have been increasingly provided by data centers recently. Compared to CPUs and GPUs, FPGAs are reconfigurable hardware that can implements digital logic with higher performance and lower power consumption for accelerating certain workloads. We focus on Amazon EC2's F1 instance as an example. Current public available regions that support F1 instances are US East (Northern Virginia), US West (Oregon), Asia(Sydney) and Europe (Ireland, Frankfurt, and London)[3]. The signal propagation in FPGA is affected by process

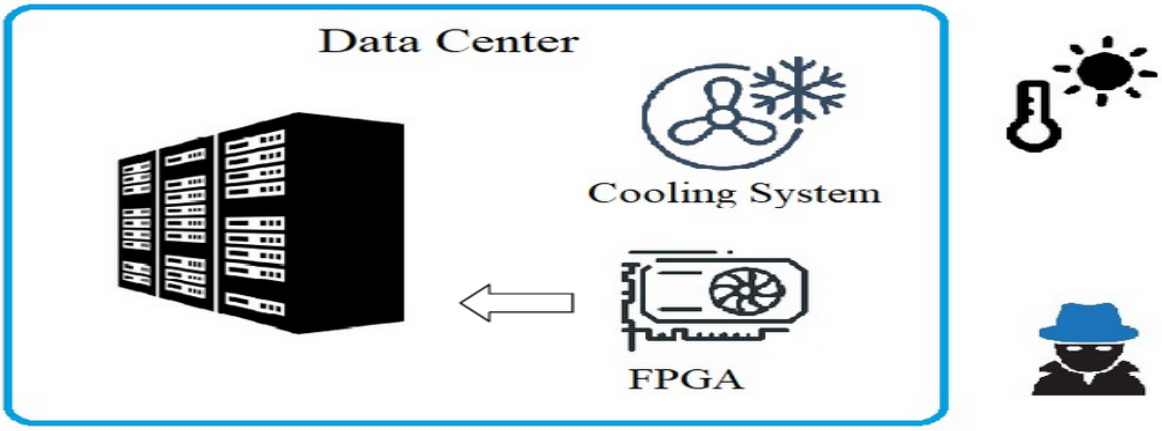


Figure 2.1: A temperature change in the data center can be observed by FPGA side channels.

variation, voltage, and temperature (PVT). Based on the effects of voltage and temperature, many side/covert channels are developed. The research on FPGA-based side/covert channels has progressed rapidly in the past decade. It advances from the lab setting to the cloud setting[71]. Ziener et al.[104] used the power supply pin to communicate with the FPGA. The time-digital converter (TDC) and the ring oscillator implemented with look-up tables (LUT-RO) are tools available to measure the signal propagation in the FPGA. Zick et al.[103] showed the measurement of transient on FPGA with TDC. Because the signal of the power supply correlates with the delay of FPGA, the oscilloscope is not needed. The LUT-RO can be used to develop side/covert channels through a power delivery system[41][100], but it requires restrictive settings and is not generally applicable for the current cloud setting. Because temperature affects the signal propagation of FPGA, the LUT-RO can be used as a temperature sensor to develop covert channels. Iakymchuk et al.[58] built a temperature covert channel between two electrically isolated parts of FPGA. Tian et al.[87] used stressors to heat up multiple cloud FPGAs and measure temperatures of FPGAs to build a covert channel. Generally, the TDC and LUT-RO are similar and interchangeable in many cases, except the TDC outputs higher resolution data and requires more memory.

In this chapter, we use the AWS F1 instance to implement the temperature sensor for the measurement study. Previous studies show that temperature information can be used to estimate power consumption[63], evaluate the vulnerability of cooling system[36], and receive information in the covert channel[55].

2.1.3 Co-Residence and Remote Side Channels

Co-residence is essential for launching many malicious attacks in cloud environments. For example, exploiting co-residence, an adversary can locate victims with fingerprints and then build covert channels. Xu et al.[97] studied co-residence threats inside the AWS data centers and revealed policies and patterns of VM placement. However, AWS often change their policies. Now the `traceroute` command no longer reports any meaningful information for private IPs in AWS. Thus, we have to explore the physical side/covert channels of cloud FPGAs.

Besides the signal propagation of FPGA we discussed before, DRAM and PCIe accessed by the FPGA also can be used to capture the temperature change, fingerprint, and develop a cross-FPGA covert channel[42][43][85][88][95]. Furthermore, all computing equipment in the data center generates heat, and so the airflow design is critical for the cooling system. Since Guri et al. [55] used air as the medium to create a covert channel between two PCs, it is possible that the airflow in the data center is used for the construction of a covert channel.

2.1.4 Power/Thermal Attacks on the Data Center

The threats of power/thermal attacks have been recognized in the past decade[36][81][98], in which adversaries exploit over-subscriptions of power supplies and reduced redundancy of cooling systems to cause power outages or/and heat up machines to disrupt or degrade the

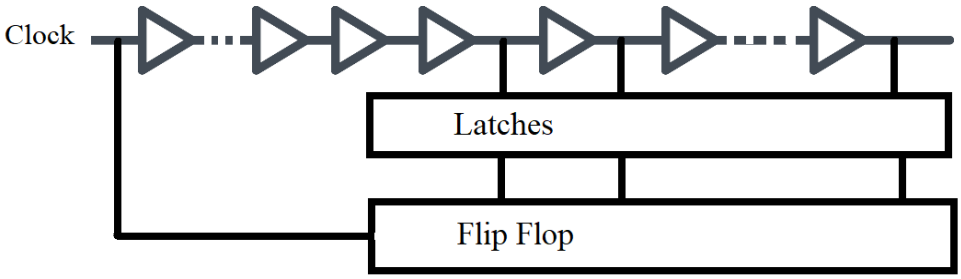


Figure 2.2: Time Digital Converter (Gate-level Illustration)

reliability and performance of data centers. Xu et al.[98] demonstrated that attackers can force victim servers to reach their power peaks at the same time and then trip the circuit breaker, causing power outages. Gao et al.[36] introduced rack-level and data center-level thermal attacks where attackers run thermal-intensive workloads to rapidly generate a large amount of heat, forcing the victim servers into a high temperature, which can potentially cause hardware damage or even server shutdown. Wang et al.[90] analyzed data center hardware failure reports over four years, and found that failure rates are higher in some rack positions. This is because hotspots are unavoidable and difficult to be cooled down, due to the non-linearity of airflow.

In particular, attackers can exploit side/covert channels to further assist mounting their power/thermal attacks. Adversaries can place different types of sensors (e.g., cooling fan sound and some frequency components of the power distribution unit) in the servers to estimate the power consumption[62][63][64], and leveraging benign workloads in the background to amplify their attacks. Adversaries can also utilize covert channels to verify co-residence, which is helpful to improve the effectiveness of both thermal and power attacks[36][97].

Warehouse-scale computing (WSC) refers to a data center in a single organization that owns all equipment. It allows clients to access resources through cloud computing, but its physical access is restrictive. Thus, placing sensors in the WSC data center is not realistic. We focus

on the AWS data centers, which are WSC data centers. AWS has strict rules for physical access to its data centers[6][7]. However, leaking information from operation engineers and other sources is possible. For example, the unreliable source Wikileaks published the internal information about AWS data centers[28][92]. Because FPGAs are close to the physical layer, they can be applied for power/thermal attacks. The power hammer shutting down the individual FPGA is a type of power attack[70]. For the large scale, FPGAs are potential tools to capture information leakage in the physical layer for evaluating the physical condition of a data center.

2.2 Time-Digital Converter

The time-digital converter (TDC) reflects the delay change in FPGA. The TDC layout is shown in Figure 2.2. The input signal does not reach the last output within a clock cycle so that registers can capture how far the signal propagates. The components of TDC include look-up tables (LUT), CARRY8, latches, and flip-flops. It requires manual placement to fix these components to the designed locations. The input source usually is a clock signal. It starts from LUTs, which are buffers, then the signal is passed to CARRY8s and latches, and then its propagation distance is stored in flip-flops (registers)[54].

The TDC can capture the voltage changes and transient responses[46][47][103]. It is useful to test delay changes in various settings and detect voltage attacks from the power system. The previous works demonstrate that TDC has high accuracy for power analysis of the AES algorithm, the BNN accelerator, and versatile tensor accelerator[45][72][78][86].

The TDC is affected by PVT variations, and we use the temperature effect to deduce the temperature change. The process variation and voltage effect in AWS FPGAs are detailed in Section 2.2.4 and Section 2.2.6, respectively. The temperature effect is demonstrated in

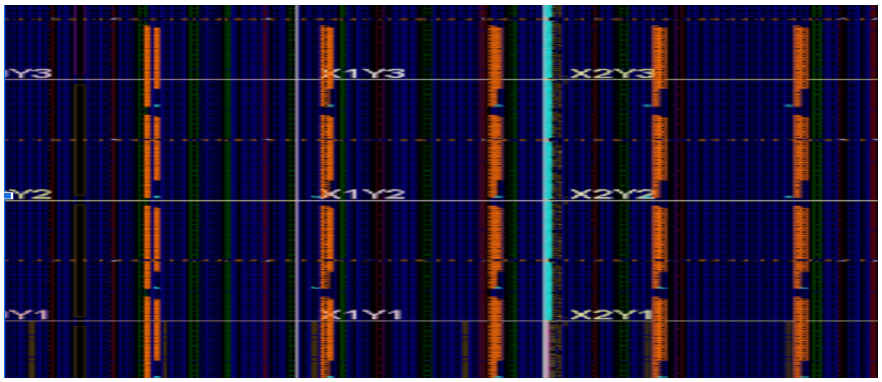


Figure 2.3: An example layout of 20 TDCs (Other FPGA designs are not shown)

Section 2.2.7 and Section 2.3.

2.2.1 Temporal Average and Spatial Average

The resolution and precision of the value in a single run of a TDC is not high enough because it is affected by noise from parasitic elements, process variation, and thermal difference. The previous research uses a temporal averaging technique to improve the resolution and precision[45][72] [78][86]. We introduce the spatial average technique to tackle the noise. The layout of TDCs is shown in Figure 2.3. We define temporal average and spatial average as follows:

- Temporal average is the average of data that are collected in different time instances.
- Spatial average is the average of data that are collected in different positions in the FPGA.

The temporal average technique is useful to observe the FPGA internal activities or power consumption of PDN (power distribution network). Although it improves the precision and resolution, this technique filters out a power ripple with probability. It may be related to the

theory of probability sampling, but sampling timing cannot be controlled. Its uncertainty is difficult to be understood and explained theoretically. We prefer techniques from the field of signal processing to filter out the power ripple.

The power ripple and the oscillation from the feedback loop of the voltage regulator generate noises in certain frequency components. The spatial average technique detects switching frequencies from the power delivery system. It has high-resolution and precise outputs, so that the power ripple can be recovered and removed with a low pass filter.

2.2.2 Amazon EC2 F1 Instance Implementation

AWS does not allow users to upload and run their FPGA images directly. Users have to use AWS FPGA toolkits that are available in GitHub[5]. AWS provides three types of F1 instances. They are F1 2xlarge, F1 4xlarge and F1 16xlarge. The FPGA chip is Xilinx Virtex UltraScale+ VU9P[4]. The CPU of the F1 instance is Intel Xeon E5-2686. We cannot find further information about the FPGA board in the AWS, voltage regulators, and power supply. It seems that AWS designs a custom board for that FPGA chip. Bittware designs the XUP-P3R board for the VU9P FPGA[11], and we believe the F1 instance uses a similar board.

AWS allows synthesizing the FPGA design with constraint files (XDC files)[93]. We use up to 20 TDCs and set up the layout as shown in Figure 2.3. We make TDCs manually using Xilinx Vivado[94], and it generates a XDC file so that we can use it for synthesis with AWS toolkits[5].

We add the TDCs implementation to the `cl_hello_world` example in the AWS toolkit to reuse the interface. The data from TDCs is stored in the BRAM of FPGA then transferred to the software. We debug and develop our design based on the `cl_hello_world` example.

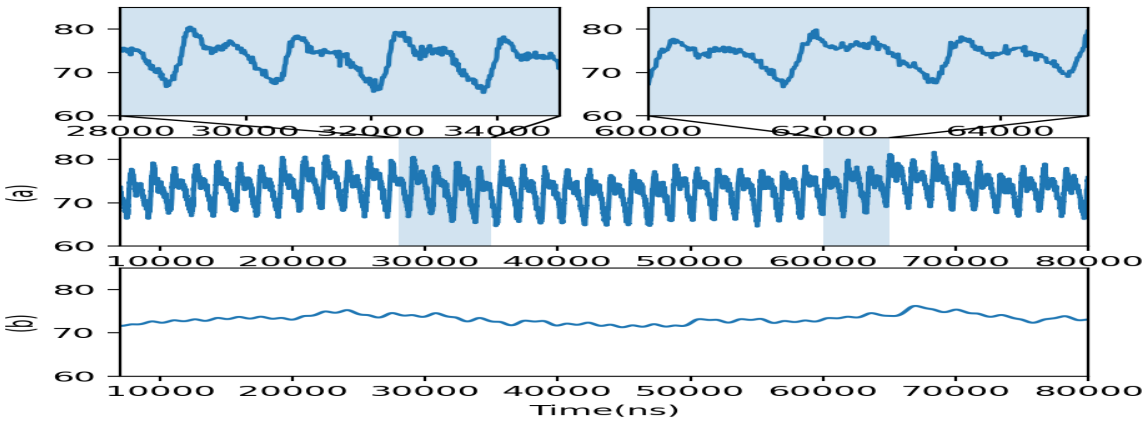


Figure 2.4: (a) An averaged TDC signal (b) The low-pass filter output

We adjust an appropriate number of LUTs and CARRY8s for the TDC to make the signal propagation passing the last LUT but not reaching the last bit output of the last CARRY8. Otherwise, it is not usable. The place and route algorithm may be run multiple times for an acceptable FPGA image.

It is costly to make all TDCs in our design usable, due to the inconsistent clock routing and process variation. The details about the process variation of AWS FPGA are discussed in Section 2.2.4. Suppose we attempt to make all TDCs usable, we need to adjust individual TDCs to synthesize multiple times for an acceptable routing solution, and the solution has to be dedicated to a physical FPGA. After we stop and restart the F1 instance, the physical FPGA can be changed. If some TDCs become unusable, the FPGA design has to be re-synthesized.

Making all TDCs usable is unnecessary because we find 15 usable TDCs are sufficient for this work. Thus, we make sure that at least 15 out of 20 TDCs are usable before starting the data collection. Each TDC output is stored in a separate file, and the average of them is computed with a software program. Because the average of TDC outputs can recover the switching frequency precisely, the low-pass filter can remove the noise caused by the

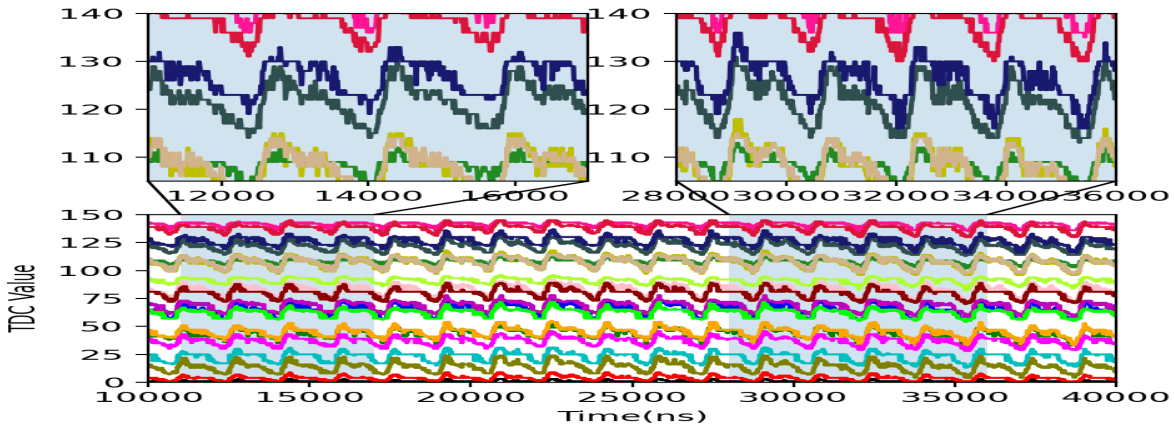


Figure 2.5: Individual TDC signals correspond to the averaged signal in Figure 2.4

switching frequency better. However, if anomalous data is found, we can check individual TDC output for reasoning.

2.2.3 Noise from Power Delivery System

The large scale FPGA in the cloud connects to the power supply directly, and these power supplies and the voltage regulators in the board are likely switch-mode. They generate and regulate power with the switch. Thus, we detect these switching frequencies with the spatial average of TDCs. Figure 2.4 shows the averaged signal of TDCs. It is more precise than signals from individual TDCs shown in Figure 2.5.

2.2.4 Process Variation

Process variation is a vital topic in the semiconductor process. It is the backbone to implement the physical unclonable function (PUF) and is also useful for fingerprinting. Because the AWS FPGA is manufactured with the advanced process, its process variation greatly influences the signal propagation of our TDCs. To show the process variation of FPGAs in

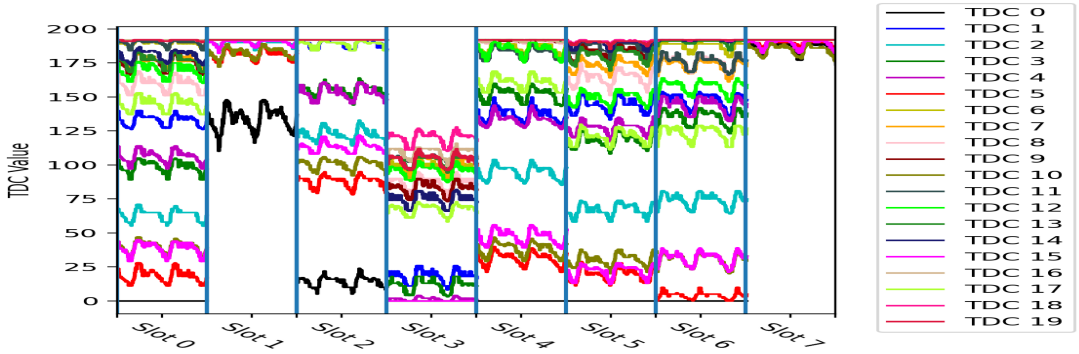


Figure 2.6: Individual TDC signals of a image in different FPGAs of the f1.16xlarge instance

AWS, we use the same image for different FPGAs. Figure 2.6 shows that individual TDCs in the same binary behave differently for FPGAs in different slots of the F1 16xlarge instance.

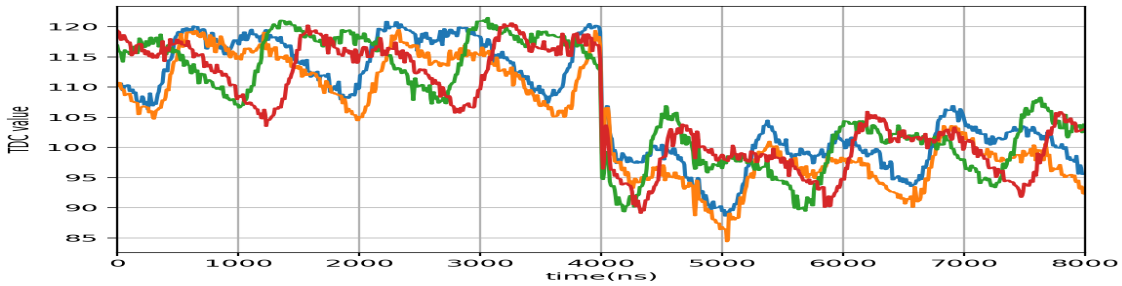


Figure 2.7: Step Response

2.2.5 Analysis of Resolution and Precision

The number of usable TDCs is N, and there are M bits of TDC output. $N \times M$ is the resolution. The improvement of resolution per additional usable TDC is M. The smallest averaged TDC output value difference is $\frac{1}{N}$. Multiple TDCs can sense thermal information uniformly, and the impacts from the process variation, parasitic capacitance, and parasitic inductance are minimized.

2.2.6 Transient Response

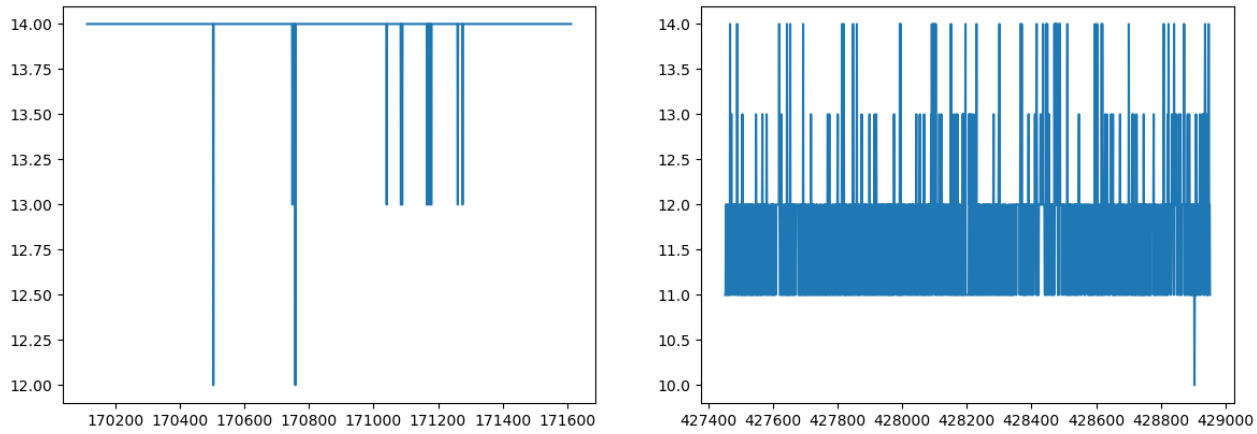
Transient response is an important way to test the power delivery system. Stressors are used to draw a large current to trigger step response. It could be any component of the FPGA as long as it can cause a voltage drop. Previous works show LUT ring oscillators, flip-flops, and programmable interconnect points (PIP)[41][46][47] [103] can be used as stressors. We use chains of LUTs as stressors. The switch of a number of LUTs in the AWS FPGA leads to the step response, and the undershoot and overshoot are visible in Figure 2.7. Temporal average also can recover them, but the information of switching frequency is lost in temporal average.

2.2.7 Local Experiments for Validity

Since we do not have any physical access to measure the temperatures inside AWS data centers as the ground truth, we conduct local experiments for proving the validity of our approach. We implement a TDC in the NEXYS A7 FPGA board and measure the temperature effect on the TDC output. We use the hairdryer to generate hot air and heat up the TDC for 1 minute. The results are shown in Figures 2.8a and 2.8b. The TDC outputs drop because the high temperature increases the delay, showing the effectiveness of our approach, which is applied to the cloud FPGA environment.

2.3 Temperature Side Channel

Our precise FPGA side channel implemented with TDCs enables the collection of temperature information in real-time. We also compare the temperature measured by our TDC side channel with the DRAM side channel[95]. The principle of the DRAM temperature



(a) Before heat up

(b) After heat up

Figure 2.8: TDC local experiment (x-axis:time step (10 ns), y-axis: TDC output)

side channel is that when temperature increases, it accelerates the decays of DRAM cells that hold 1s (The decays of DRAM cells vary due to the manufacture variation). In other words, we can deduce the temperature changes by counting the number of bits that flip from 1 to 0 in a fixed duration. We adopt Tian et al.'s[88] approach to implement the DRAM side channel in the AWS FPGA. We turn off the ECC and scrubber of DRAM by modifying the `ddr4_core_ddr4` module and `cl_dram_dma` example. The `cl_hello_world` example does not use the DRAM, so the DRAM stops refreshing when it is used. We use a modified version of the `cl_hello_world` example to avoid refreshing. The first step to implement the DRAM side channel is writing 1s to the selected DRAM region with the modified `cl_dram_dma` example. The second step is replacing the `cl_dram_dma` image with the modified `cl_hello_world` image so that the selected region will decay. Finally, after a certain time, we load the modified `cl_dram_dma` image back and read the selected region, and we can count the flipped bits to learn the temperature change.

Collecting temperature data in AWS data centers allows us to take a glance at temperature changes inside the data center. Adversaries can also exploit our FPGA side channel as

thermal sensors to obtain temperature information for WSC data centers, and thus find optimal timings to mount power/thermal attacks.

2.3.1 Global-Scale Free Cooling

The free cooling technology has become the trend and supports various climates[73]. However, the local temperature is a physical constraint. In some cities with hot climates, the initial investment of free cooling can be high and its overall saving is small.

Moreover, inaccurate weather forecast and unexpected local high temperature deteriorate the complexity of workload management. In addition to seasonal/daily temperature changes, extreme weathers also should be considered. As global warming heats the earth, a data center should make sure its cooling system can maintain an appropriate temperature. The extremely low temperature should also be paid attention to because water is often used as a heat medium in the cooling system, and some computing equipment cannot run at a low temperature.

Since free cooling increases the predictability of cooling system, previous research on power/thermal attack strategies on a single data center can be extended to multiple data centers. Adversaries can use the FPGA side channel to verify that the data center uses free cooling. Then, they can choose a time when the outside temperature is high to attack.

2.3.2 Locations of Data Centers

Table 2.1 shows approximate locations of data centers in regions that support F1 instance based on the information of baxtel.com[9]. The locations of data centers are important because electricity price, data center market, weather, natural disaster, etc., are factors that

Table 2.1: AWS Data Centers Locations[9]

Region	***Approximate Location***	***Availability Zones***	***Time Offset (Sep. Oct.)***
Virginia (US)	Ashburn	6	UTC-4
Oregon (US)	Umatilla	4	UTC-7
Sydney (Asia)	Sydney	3	UTC+10
London (EU)	Didcot	3	UTC+1
Frankfurt (EU)	Frankfurt	3	UTC+2
Ireland (EU)	Dublin	3	UTC+1

should be considered to choose the location for a data center. Although addresses of AWS data centers can be found, we do not know how each availability zone corresponds to the data center. There are multiple AWS buildings in each region, and they are close to each other.

We need the local weather information to learn whether a data center uses free cooling or not. The local weather information of each region is based on its approximate locations shown in Table 2.1. Furthermore, the locations of availability zones can be better approximated with latency shown in Table 2.2. To know the physical location of each availability zone, one method is to use the FPGA-based temperature side channel as a receiver and the outside air as the transmitter to construct a covert channel because free cooling uses the outside air. Besides the temperature covert channel, the electromagnetic covert channel is also useful to localize the data center[82]. However, the city-level temperature is sufficient for investigating the free cooling scenario.

2.3.3 DRAM Side Channel versus FPGA Side Channel

Each F1 instance in AWS provides FPGA and DRAM that the FPGA image can access. We use them to implement temperature side channels. The DRAM side channel and TDC

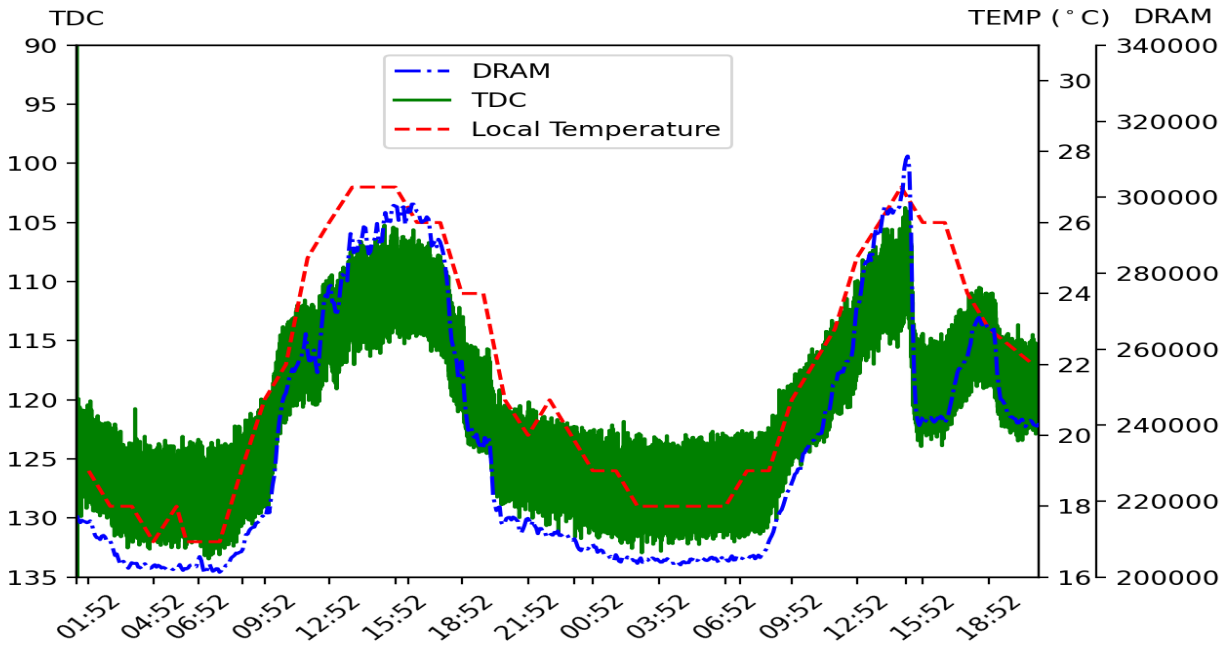


Figure 2.9: The data collected with an F1 2xlarge instance in the subnet *a*, Northern Virginia region (local time: UTC - 4).

side channel are implemented with entirely different mechanisms. After the DRAM refresh is turned off, the time of memory retention for each bit depends on temperature. We write 1s to selected bits in DRAM, and then we read the selected bits after a certain amount of time to count the number of flipped bits[88][95]. If the temperature increases, the number of flipped bits increases. Thus, we can deduce the temperature change. The TDC is implemented to measure the delay of signal propagation. It is well-known that the temperature affects the signal propagation in the digital chip. We observe that when the number of flipped bits increases, the TDC output decreases. Their temperature patterns are mostly consistent, as illustrated in Figures 2.9-2.21. AWS FPGA does not have the temperature inversion phenomenon. This phenomenon happens if the supply voltage in the digital chip is reduced[77][105]. The FPGA chip and DRAM are placed closely to each other, and it is worth noting that a large FPGA is equipped with a cooling fan. Its cooling functionality is slightly better than DRAM.

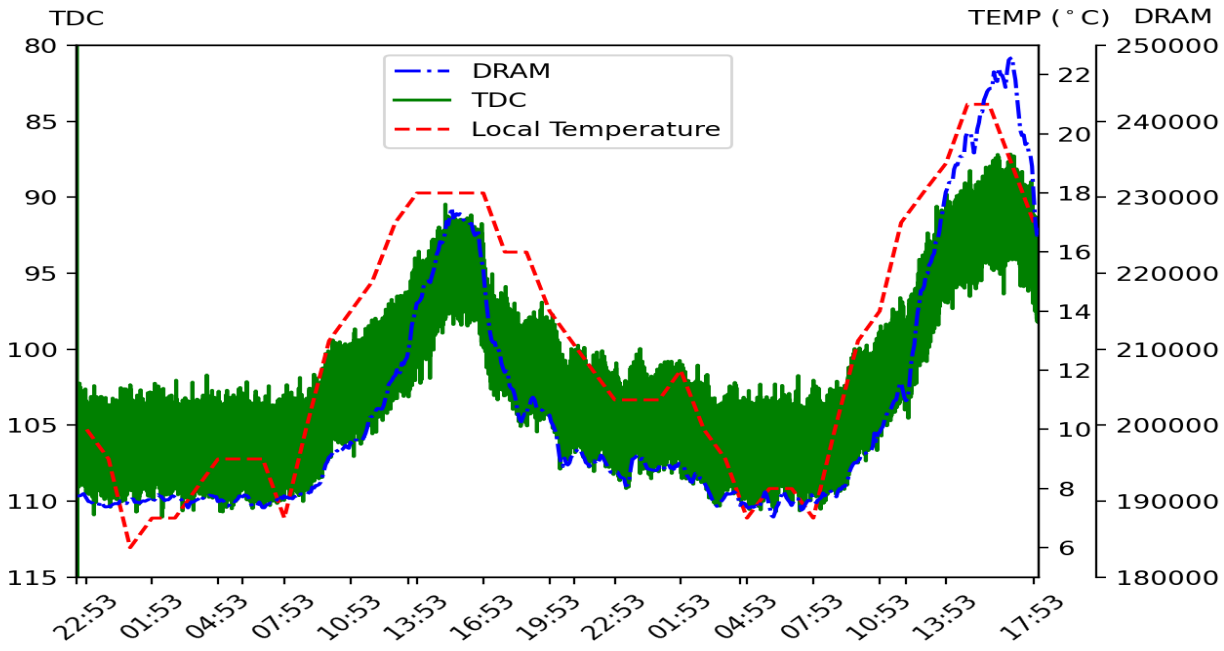


Figure 2.10: The data collected with an F1 2xlarge instance in the subnet a, Oregon region (local time: UTC - 7).

Accuracy Analysis. When the temperature increases, the signal delay in the FPGA increases close to linear, and according to Xiong et al.[95], the number of flipped bits in the DRAM increases superlinearly. Both the TDC side channel and DRAM side channel are non-linear functions on temperature. Because the temperature range is small (about 10 degrees), they can be considered approximately linear. We collect temperature data with both side channels simultaneously for double-checking.

Data Collection. To automate the data collection process, we write a script in Bash. It runs TDCs iteratively, and the DRAM side channel is run once in every ten iterations. The TDC data is stored in RAMs temporarily, and then the data is read with the interface and stored in a file. The spatial average of TDC values is also computed and stored. For the DRAM side channel, the script waits 20 seconds for the decay of selected bits. Then the number of flipped bits is counted and stored. We use the `date` command to record the data

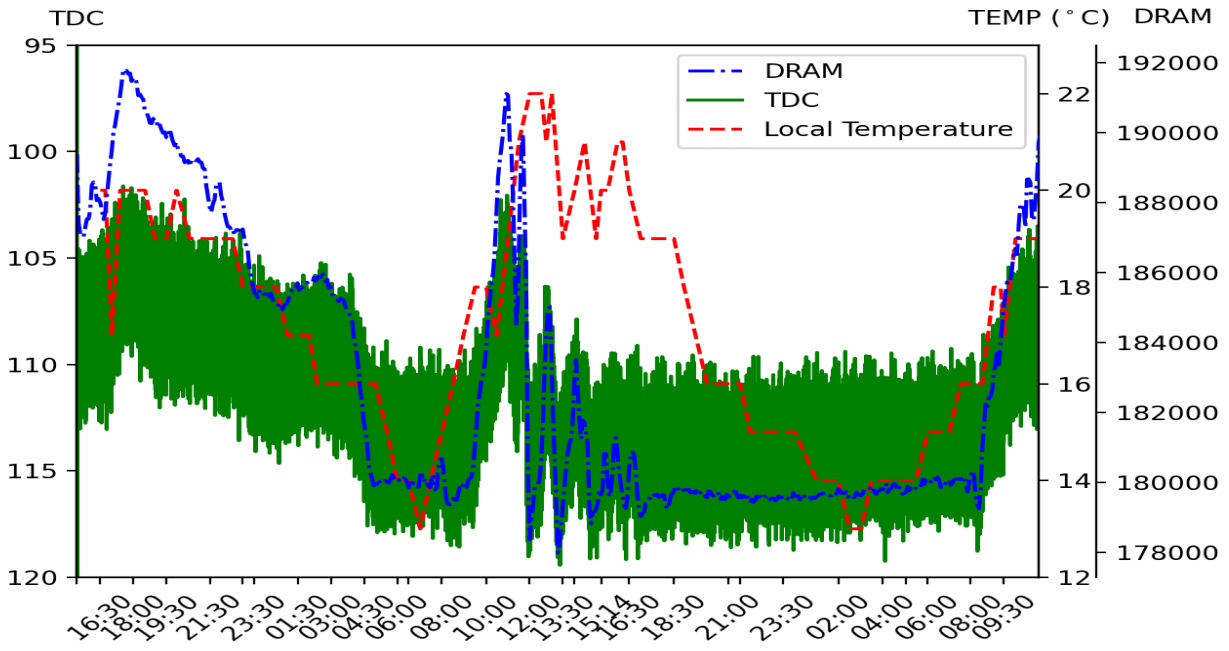


Figure 2.11: The data collected with an F1 2xlarge instance in the subnet a, Sydney region (local time: UTC + 10).

collection time in UTC (coordinated universal time) format. For the analysis of data, the average TDC values are concatenated, and then the high-frequency noise is removed with a low-pass filter for the data plot. The TDC values, numbers of DRAM flipped bits, and local temperatures are aligned and plotted based on the UTC from the `date` command. To show the local weather intuitively, we use the local time in Figures 2.9-2.21.

2.3.4 Temperatures of Regions

In our AWS account, there are 21 regions to choose from. However, only Northern Virginia, Oregon, Ireland, Sydney, Frankfurt, and London regions support F1 instances. We believe the cooling systems of data centers in these regions are applicable for the rest of the regions. To analyze how local weather affects the temperature of a data center, we use the weather data in the website timeanddate.com[89]. It provides hour-by-hour past and forecast weather

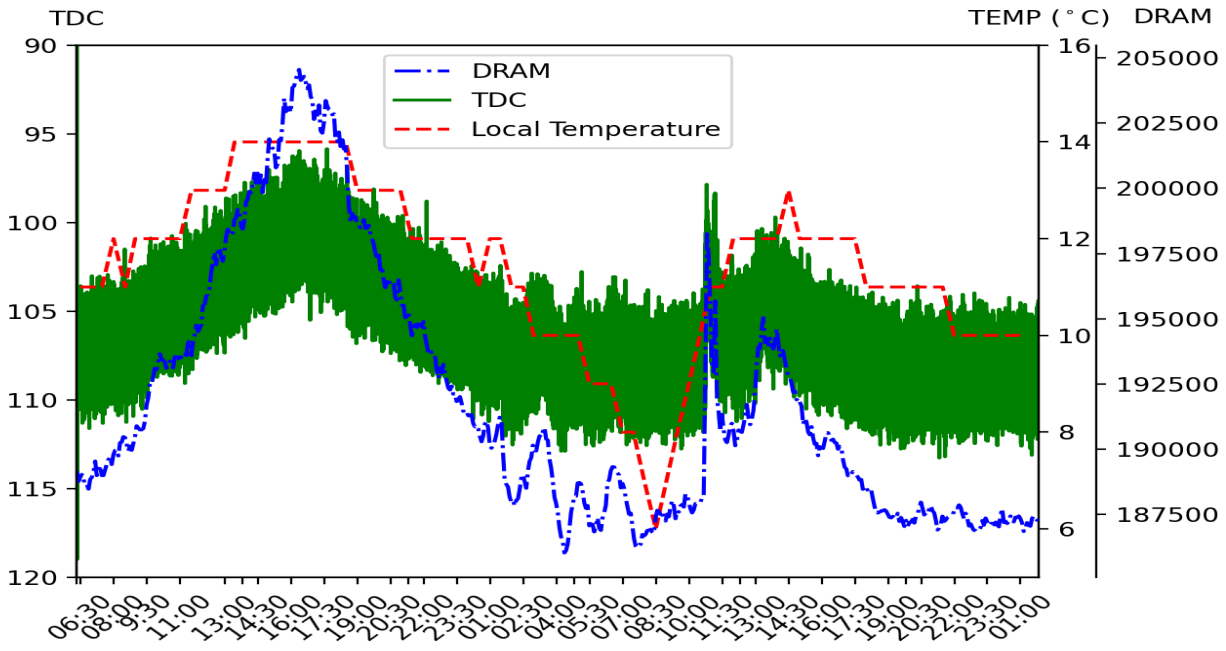


Figure 2.12: The data collected with an F1 2xlarge instance in the subnet a, Ireland region (local time: UTC + 1).

data in all cities around the world. Using the localization methods discussed in Section 2.3.2, we obtain the approximate local weather for data centers in each region. We launch an F1 2xlarge instance in the subnet a of every region that supports the F1 instance for measurement except the Frankfurt region. Only subnet b in the Frankfurt region supports F1 instances, and so we have to launch F1 instances in subnet b. The measurement in all regions starts approximately at 5am on October 14th, 2021 (UTC). The results from different regions are shown in Figures 2.9-2.14.

For F1 instances in Northern Virginia, Oregon, Ireland, and Sydney regions, the patterns of their DRAM side channel and TDC output shown in Figures 2.9-2.12 are consistent. Generally, their temperature patterns match the temperature changes of local weathers, and so we believe data centers in these regions adopt free cooling techniques. When the computing equipment heats up the data center, or/and the outdoor temperature in the

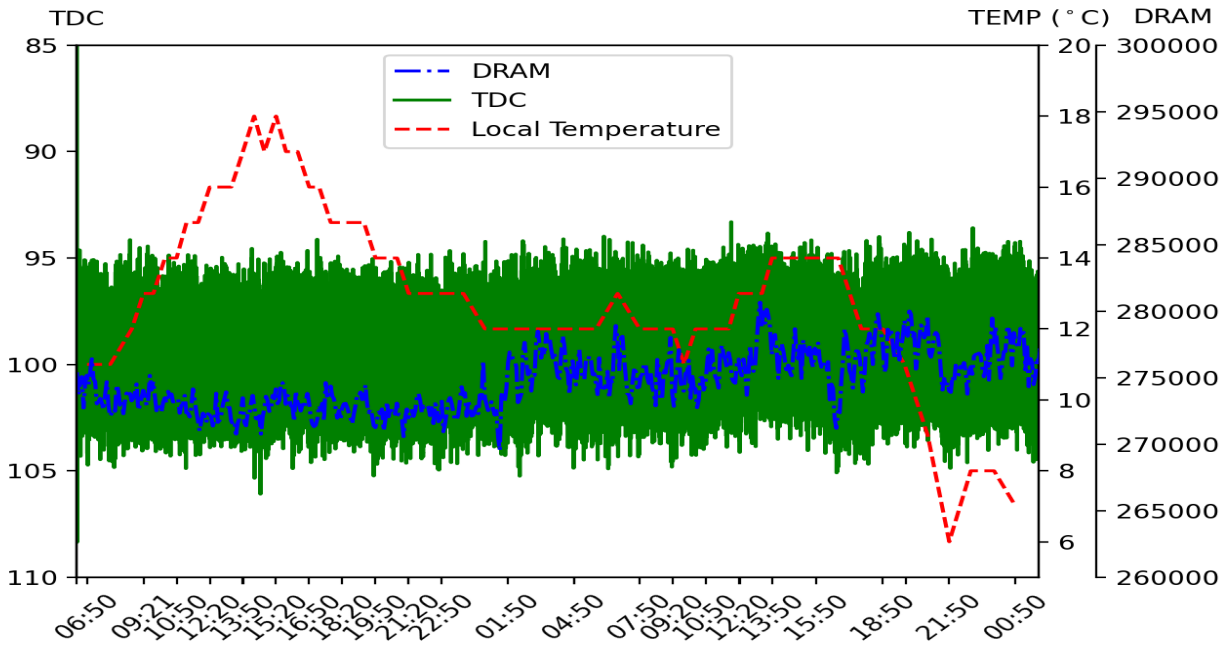


Figure 2.13: The data collected with an F1 2xlarge instance in the subnet *a*, London region (local time: UTC + 1).

daytime is hot, the cooling system cools down the computing equipment. The temperature drop caused by the cooling system is demonstrated in Figure 2.9. The temperature side channel in the Northern Virginia region shows a sudden temperature drop at about local time 3pm on October 15th, 2021. We can see similar phenomena in the Sydney and Ireland regions (Figures 2.11-2.12).

The timing that the cooling system acts is interesting. In Figure 2.12, we can observe a sudden temperature drop at 11am on October 15th, whereas there is no drop on October 14th. It is noticeable that the drop starts when the outside temperature is 11 °C. Similarly, for Figure 2.11, the temperature drops at about 11am on October 15th, when the outside temperature is 19 °C. We can see that the outside temperature is not the only factor for the operation of the cooling system, and there are probably temperature sensors in different locations inside the data center to measure the temperature of computing equipment more

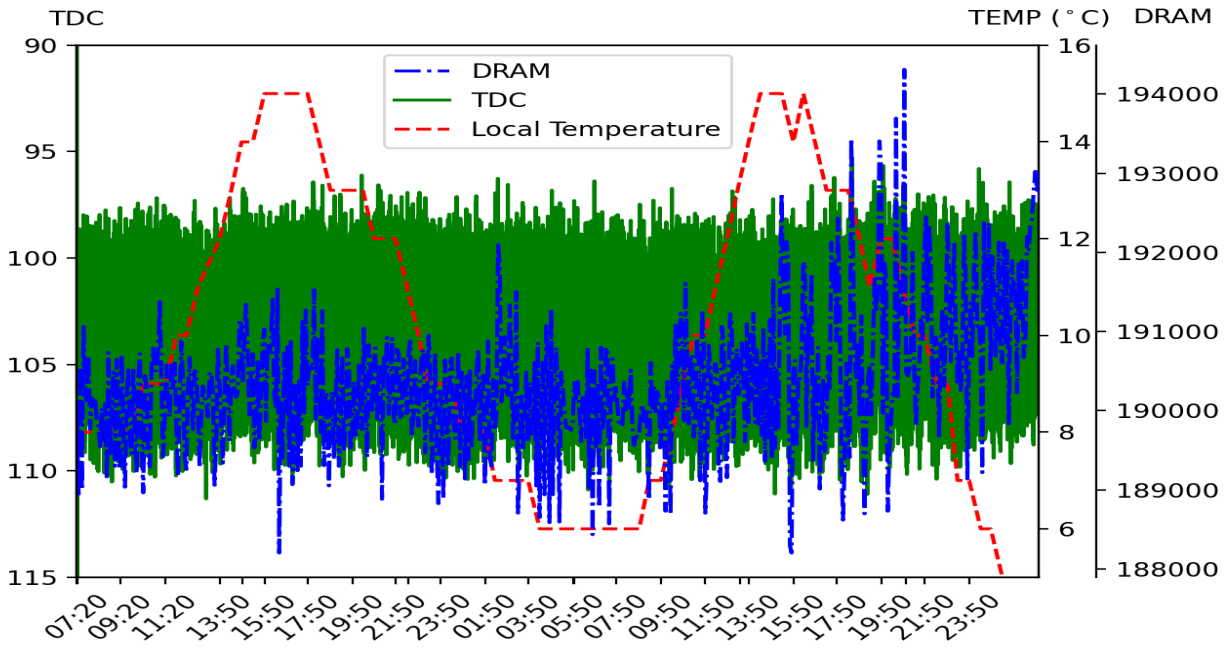


Figure 2.14: The data collected with an F1 2xlarge instance in the subnet *b*, Frankfurt region (local time: UTC + 2).

closely. Thus, the high-temperature spot inside the data center can trigger the cooling system to cool it down. The F1 instance in the Sydney region (Figure 2.11) shows that its temperature is constant during the nighttime on October 15th. We believe that the data center has a mechanism to maintain the constant temperature for the FPGA card for some duration.

For London region and Frankfurt region, temperatures of F1 instances are shown in Figures 2.13-2.14. The patterns of the DRAM side channel and TDC output demonstrate that the temperature changes are minimal. They are independent of the local temperatures, indicating that the data centers of these two regions do not use free cooling during our measurement.

Northern Virginia, Oregon, Ireland, and Sydney are in different time zones, and their local temperatures peak at different times. The computational tasks that require high computa-

tion power and less latency constraint should avoid the peak local temperature or be run in a different data center where the cool outside air and computing resources are available. However, current regions that support F1 instances are insufficient to allow sophisticated global resource allocation. It needs not only the weather forecast but also workload coordination among data centers. Xu et al.[96] formulated this problem and developed an algorithm to address it. AWS or other public clouds do not have much flexibility to distribute workloads, since cloud users usually request a service in a specific area. However, the price can be leveraged by cloud service providers to motivate users to use cloud resources with lower costs.

2.3.5 Temperatures of Availability Zones in Northen Virginia

Table 2.2: Minimum Latencies among Availability Zones in Northern Virginia Region (Unit:ms)

srcdst	a	b	c	d	e
Subnet a	0.237	0.567	0.656	1.278	0.341
Subnet b	0.570	0.072	0.413	0.433	0.630
Subnet c	0.570	0.072	0.413	0.433	0.630
Subnet d	1.272	0.438	0.346	0.139	0.656
Subnet e	0.348	0.620	0.613	0.659	0.109

There is a large data center market in Northern Virginia, and it continues boosting. AWS has invested many resources in this market, and so we study AWS data centers in the Northern Virginia region. AWS makes multiple availability zones in each region, and each availability zone guarantees that it has independent power and networking infrastructures[8]. More flexible choice of availability zone provides better latency service for cloud users. Power outages of data centers are not uncommon, and it happens every year[60]. Having multiple data centers can avoid the single point of failure from the viewpoint of security engineering.

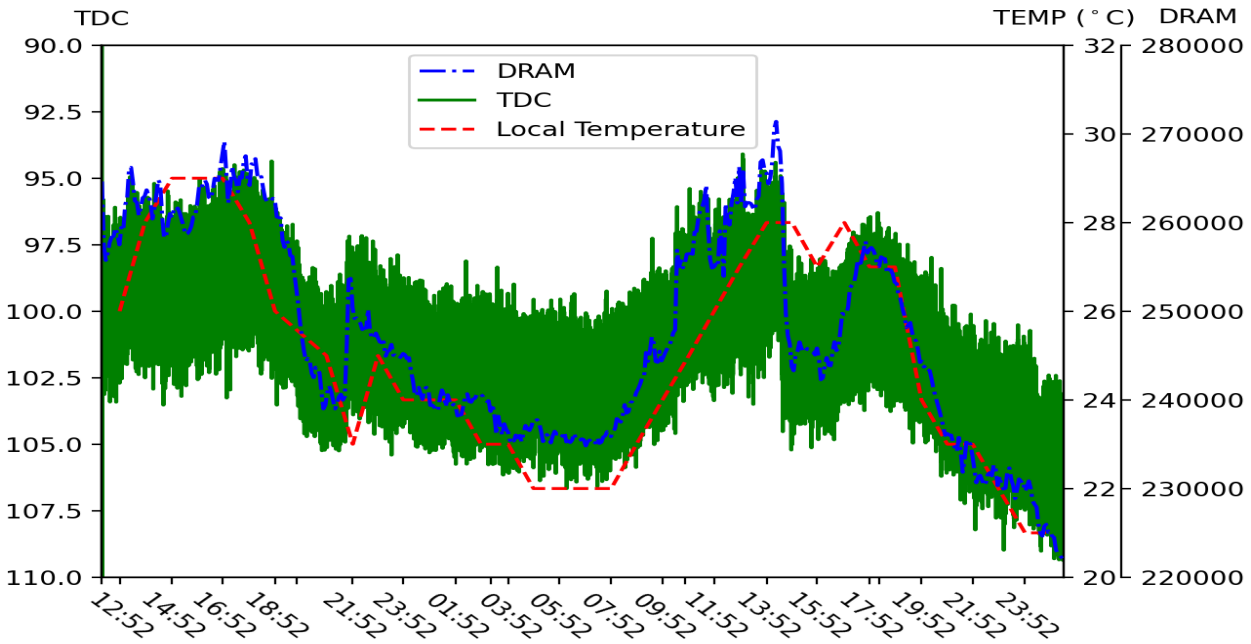


Figure 2.15: The data collection with an F1 2xlarge instance in the subnet a, Northern Virginia region starts at 12pm on September 18th, 2021.

We can find the locations of AWS data centers in baxtel.com[9], but we do not know the physical location of each availability zone. We use latency to estimate the physical distances among data centers. Moreover, latency is a metric to estimate the distances among physical machines[97]. Minimum latency among subnets shown in Table 2.2 are measured with the Linux command `ping`. From Table 2.2, we can approximately know their relative distances among availability zones.

We measure the temperature of all subnets in the Northern Virginia region except subnet f because it does not support F1 instance so far. The measurement starts at around 4pm on September 18th, 2021 (UTC) or local Virginia time 12pm on September 18th. The local temperature is based on the weather in Ashburn, VA. Based on our data (Figures 2.15-2.19, we can verify that temperatures for F1 instances in different availability zones simultaneously synchronize well with the local temperature changes, except that at some

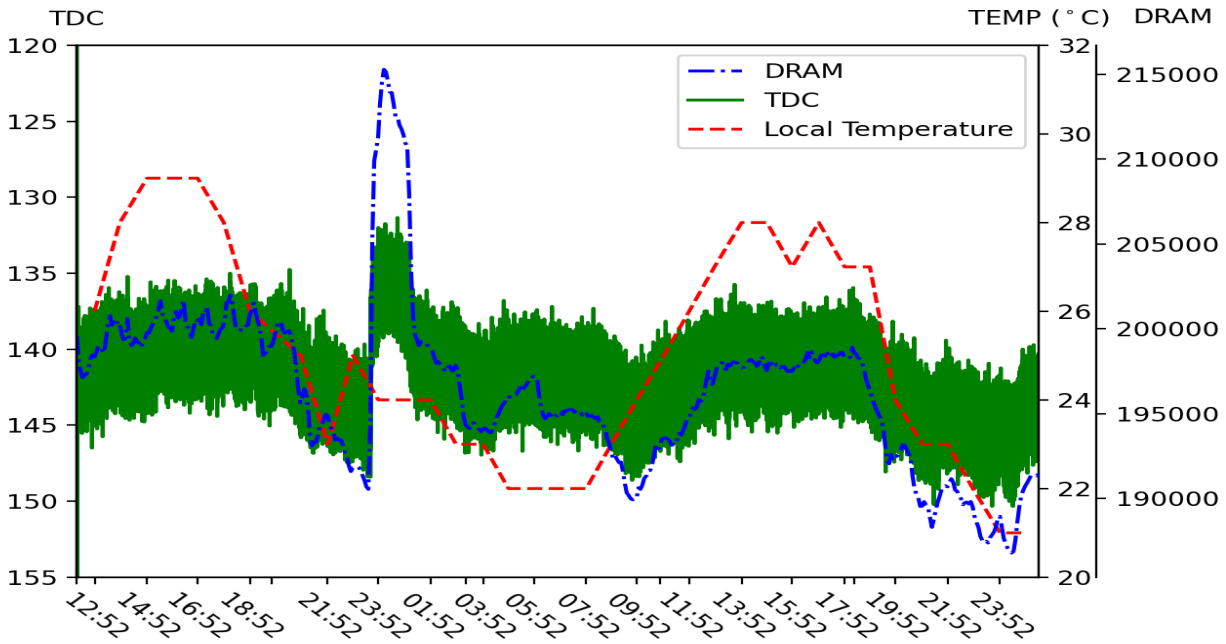


Figure 2.16: The data collection with an F1 2xlarge instance in the subnet *b*, Northern Virginia region.

occasions the temperatures rise due to high workloads. Therefore, we believe that these availability zones supporting F1 instances in Northern Virginia region adopt free cooling.

Since the data of Figures 2.15-2.19 are collected from different data centers, their operations of the cooling system and the heats from computing equipment are independent. From Figures 2.16-2.17, subnets *b* and *c* show high temperature periods during nighttime. It is due to the heat generated by their computing equipment, but we cannot determine if they could apply to the overall utilization of a data center. Each rise and drop of temperature lasts about 30 minutes. Since the efficiency of the data center is important, the data center always tries to improve its utilization and minimize idle time. Moreover, Figures 2.16-2.17 also demonstrate that the FPGA instance can be heated up higher than the peak temperature of the day. Figure 2.15 shows a 30-minute duration temperature drop at about 2:20 pm on September 19th even when the local temperature peaks. It shows that at some scenarios

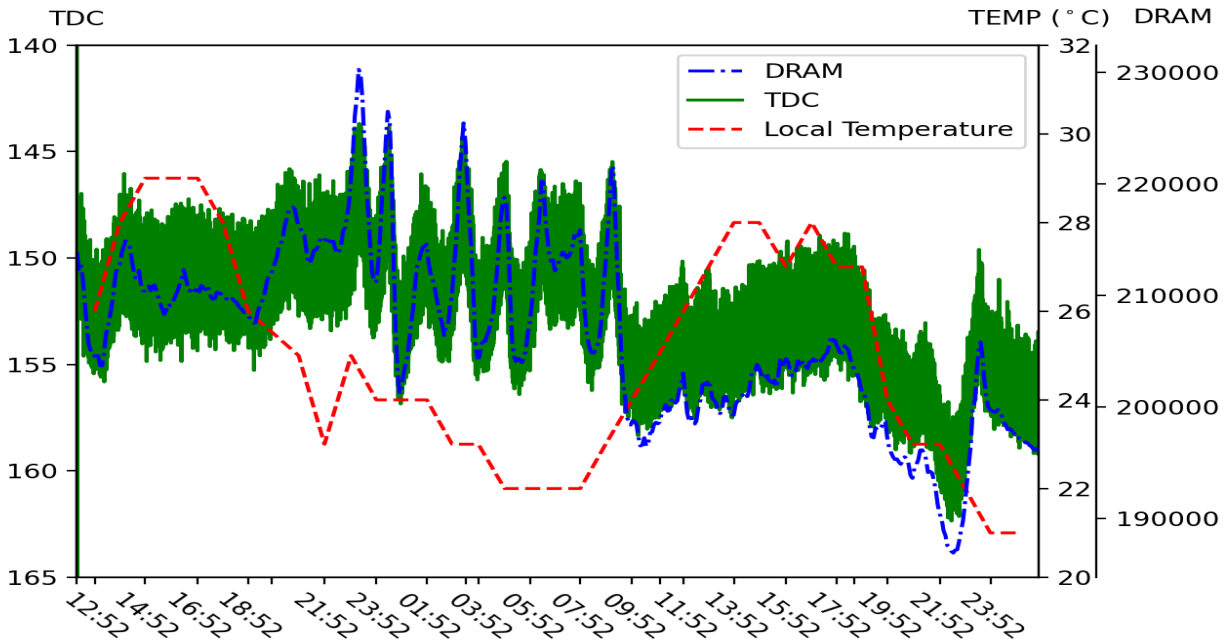


Figure 2.17: The data collected with an F1 2xlarge instance in the subnet *c*, Northern Virginia region.

when too much heat generated by computing equipment, the cooling system has to work without using free air for quickly lowering the temperature. By contrast, Figures 2.16 - 2.19 do not show a sharp temperature drop at that time even the local temperature reaches peak (28 °C).

2.3.6 Temperatures in the Availability Zone A

To observe what types of information are useful for mounting data center level attacks, we collect data from 4 more instances in the subnet *a* of the Northern Virginia region. This measurement starts at local time 1:45am on September 19th, 2021. The F1 instance in Figure 2.15 is the instance 1, and the instance 2 and instance 4 at subnet *a* are shown in Figures 2.20 and 2.21, respectively. The temperature patterns of instance 3 and instance 5 are similar to these two instances, and thus their data are omitted.

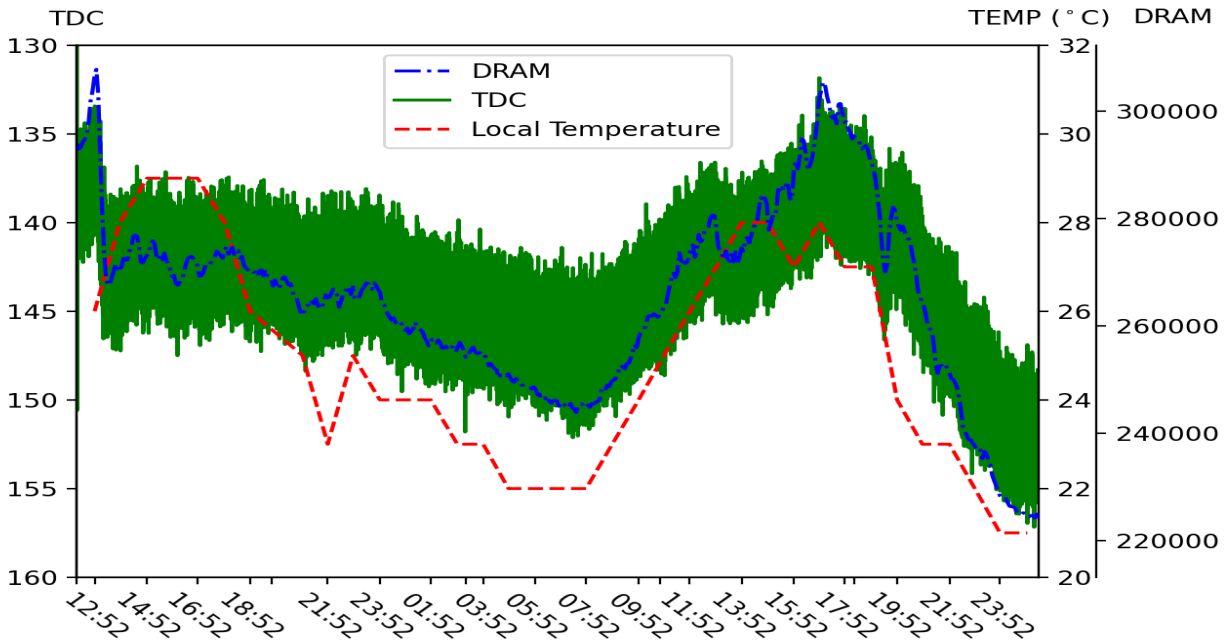


Figure 2.18: The data collected with an F1 2xlarge instance in the subnet *d*, Northern Virginia region.

Although AWS experienced a number of outages and problems, it claimed that outages never caused a loss of the entire data center[29]. AWS also revealed information about the relationship between the data center and the availability zone. It stated that "In every zone, you have at least one data center." In other words, it did not confirm two instances in the same availability zone are in the same data center, and it seems that AWS does a great job of isolating the fault effects.

However, from Figures 2.20 and 2.21, the overall temperature patterns of the two instances are similar. The effects of the local weather and cooling are identical, implying the physical locations of the two instances are possibly close to each other. Both temperatures dropped at about 2:40pm. Figure 2.15 also captures the temperature drop in the subnet *a* at that time. The subtle differences among them are noticeable. We can observe that temperature peaks at about 11:00am in Figure 2.20, which is comparable to the temperature when the

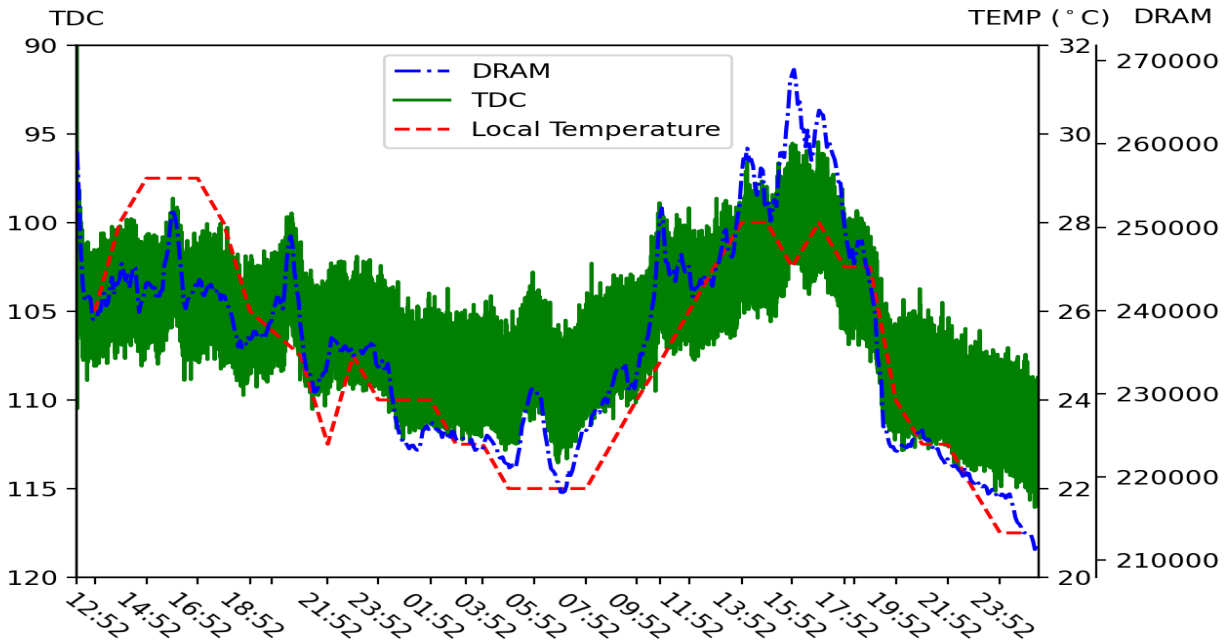


Figure 2.19: The data collected with an F1 2xlarge instance in the subnet *e*, Northern Virginia region.

local temperature peaks. Tian et al.[85] showed the probability that FPGAs share the same NUMA node is decent. Based on their similar temperature patterns, we speculate that these four F1 instances are in the same data center.

2.4 Discussion

In Section 2.3.5, we show all subnets a-e adopt free cooling. However, the climate in Northern Virginia does not always provide cool air. Since our measurements were conducted in the fall, these subnets simultaneously use the outside air of the same area for the cooling purposes. However, the usage of free cooling is highly dependent upon the physical location of a data center and its local weather. If the local weather is hot, all data centers in these subnets should avoid the usage of free cooling. Therefore, it is worth to pay attention to the physical

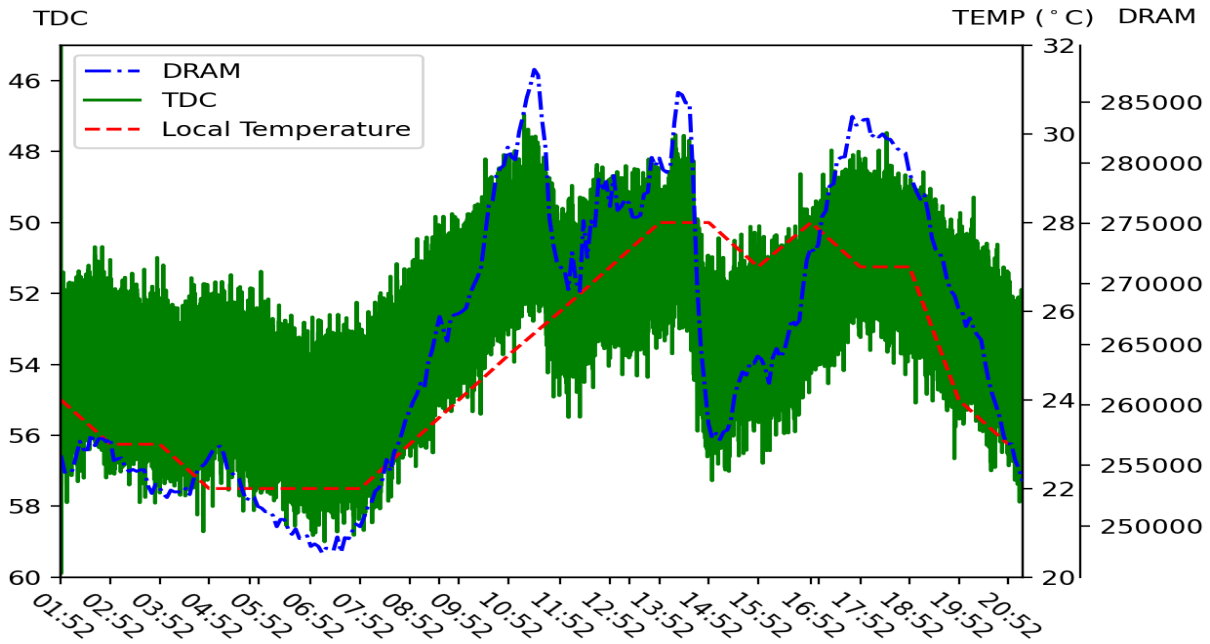


Figure 2.20: The data collected with the additional F1 2xlarge instance 2 in the subnet a, Northern Virginia region.

location and current weather of data centers for studying their cooling systems and daily operations.

Since we do not have physical access to AWS data centers, we cannot know the exact physical locations of FPGAs and their relative relations with other computing equipment. However, from the perspective of an adversary, the information sources do not have to be reliable and accurate. In Section 2.3.6, we show similar temperature patterns of four F1 instances in subnet a, and they are likely in the same data center. The adversary can estimate the physical locations of FPGAs by observing their temperature pattern to launch more effective power/thermal attacks.

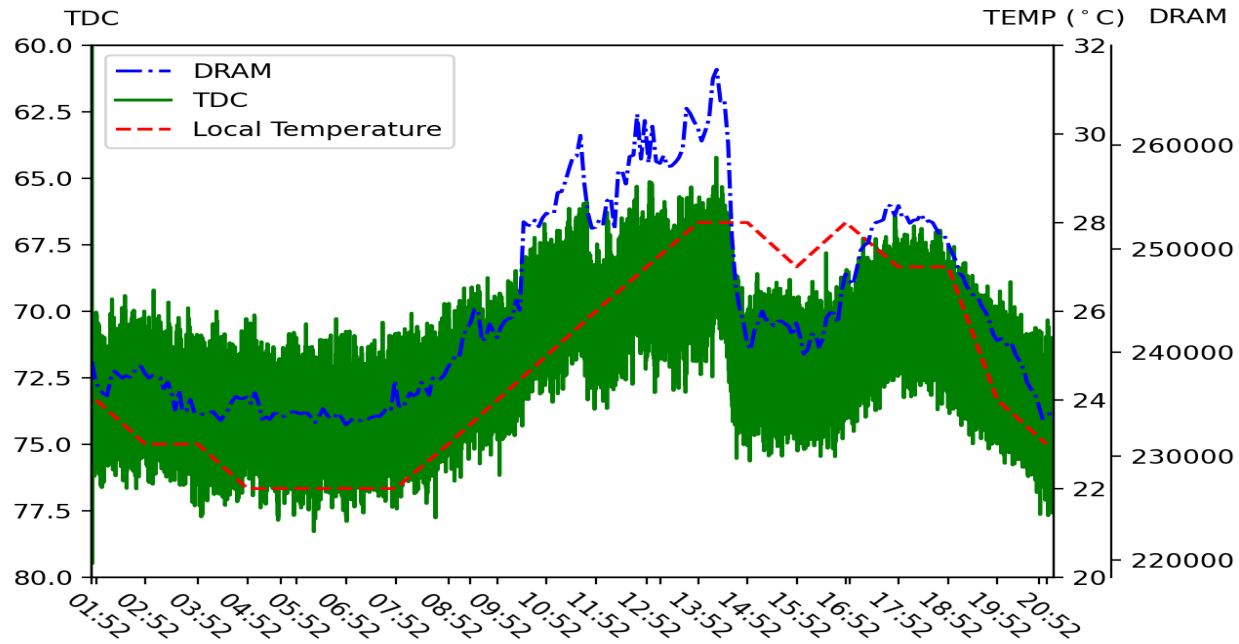


Figure 2.21: The data collected with the additional F1 2xlarge instance 4 in the subnet a, Northern Virginia region.

2.5 Conclusion

In this chapter, we conduct a measurement study on the cooling systems of AWS data centers through FPGA-based temperature side channels. We leverage TDCs to observe the process variation, step response, and temperature effect of AWS FPGAs. We also implement the DRAM temperature side channel. We use the DRAM side channel and TDCs to collect the temperature information leakage. We select 10 availability zones (10 or more AWS data centers) for the measurement study. Based on the collected data, we analyze the cooling system of AWS data centers and identify many of them adopt free cooling for the reduction of cooling cost. Our study not only reveals the temperature information leakage in data centers but also investigates the data center cooling system from security and privacy perspectives.

Chapter 3

Efficienct Identity-based encryption with minimal server trust

Previous work on the IBE key escrow problem. The closest previous work consists of three main approaches: threshold key derivation by splitting one server into many, using an additional authentication server that knows users' identities, and registration-based encryption, which we now discuss.

In [15], Boneh and Franklin already discussed the key escrow problem and suggested using the threshold cryptography techniques (i.e., replicating the PKG server into many servers, of which to trust, say, a majority). Related approaches include [65] and [10]. These techniques resolve the "single point of failure" problem but do not protect the confidentiality of ciphertexts if an adversary corrupts a large enough number of key derivation servers.

Sherman [23] removed the key escrow by separating the knowledge of identities from the key derivation authority, and it has a line of follow-up research [33] [1]. These works do not protect the confidentiality of ciphertexts if an adversary corrupts both the authentication and the key derivation servers.

The recent work of registration-based encryption (RBE) [38] [39] leverages public key encryption to build a scheme very similar to an IBE scheme by replacing a PKG with a key curator. Furthermore, Verifiable RBE (VRBE) [50] improves the auditability of key cura-

tors. Other work in this area includes [56], where Hohenberger et al. constructed an RBE scheme for attribute-based encryption. RBE does address the key escrow problem, but in terms of efficiency requirements, it significantly prioritizes the efficiency of the key derivation protocol over the efficiency of the encryption algorithm. Moreover, all known RBE constructions work in the common reference (or random) string model, which adds the additional trust assumption of the correct generation of this string. As known in the area, private information associated with this string might be subject to subversion attacks or require a preliminary generic secure multi-party generation protocol. Thus, RBE does not protect the confidentiality of ciphertexts if an adversary performs a key subversion attack. Finally, as an alternative to the IBE concept, RBE does not take advantage of many of the several research efforts on IBE schemes. Table 3.1 summarizes the comparison between our results and these previous approaches, with respect to channel assumptions, type of public parameters, and ciphertext/trust assumptions,

Other partial solutions to the key escrow problem include the following: anonymous IBE [14] in systems with large-entropy identities, as observed by [23]; certificateless public key cryptography [76, 21], which, however, does not maintain IBE's identity features; decentralizing attribute-based encryption [69], where however corruption of a few or even one authority server suffice to violate ciphertext confidentiality; and decentralizing functional encryption [22] (of which IBE can be seen as a special case), which, however, does not securely extend to multiple receiving users.

Our contribution. Our proposal to address the IBE key escrow problem consists of enhancing IBE schemes so that the key derivation function is distributed across the decrypting receivers, while targeting no or minimal modification to the encryption and decryption algorithms, so as to preserve the use of such schemes in many real-life IoT applications (as in [67]). Our first solution has no PKG or central server, and transforms anyone in a large

class of IBE schemes, called dhr-IBE schemes, into an IBE scheme where key derivation is fully distributed among the decrypting receivers, called ns-IBE (for no-server-IBE) scheme. In this scheme, security is achieved in the adversary model where the adversary is given secret keys for all identities but the target identity, but the corrupted receivers honestly follow the key derivation protocol. Our novel definition of the class of dhr-IBE schemes uncovers three essential technical properties that allow a transformation of IBE schemes into IBE schemes without the key escrow problem: decentralized server behavior, homomorphism of the key derivation algorithm, and re-encryptability of ciphertexts. We verified that, with no or minimal modifications to the encryption and decryption algorithm, the IBE schemes in [15, 91, 13] belong to the class of dhr-IBE schemes (we also included here the proof for the scheme in [15]).

Our second solution reduces computation and communication and simplifies the work done by decrypting receivers in our ns-IBE scheme by using a repository server, thus obtaining a mst-IBE (for minimum-server-trust-IBE) scheme. This protocol works for another large class of IBE schemes, called vdhr-IBE scheme, which augments dhr-IBE scheme by allowing verifiability of receivers' contributions to the public keys and decryption keys. Again, we verified that, with no or minimal modifications to the encryption and decryption algorithm, the IBE schemes in [15, 91, 13] belong to the class of vdhr-IBE schemes (and included here the proof for the scheme in [15]). In this scheme, security is achieved in the adversary model where the adversary is given secret keys for all identities but the target identity, and no limitation is assumed on how maliciously the corrupted receivers can modify their messages within the key derivation protocol, even possibly when subject to side/physical attacks.

In both of our protocols, we achieve the previously unachieved properties of removing server trust with no or minimal modifications to the original IBE schemes, thus preserving their efficiency. As some IBE schemes (in fact, hierarchical IBE schemes) have been successfully

Table 3.1: Comparisons with previous work: key derivation channel assumptions, type of public parameters, and main application scenario (top), and ciphertext security/trust assumptions (bottom).

Schemes	**Key Derivation Channel**	**Public Parameter**	**Main Application Scenario**
IBE [15, 91, 13]	Encrypted and Authenticated	Fixed	Low/High Ciphertext Rate, Low/High Receiver Join Rate, High Server Trust
RBE [38, 39, 50]	Authenticated	Dynamic	Low Ciphertext Rate, High Receiver Join Rate, Low Server Trust
This Chapter	Authenticated	Dynamic	High Ciphertext Rate, Low Receiver Join Rate, Low Server Trust

Solution Approaches	**Ciphertext Security/Trust Assumptions**
Single Key Derivation Server [80, 15, 25, 91]	Honesty of Key Derivation Server + No Intrusion of User's Key Storage
Multiple Key Derivation Servers [15, 65, 10]	No Collusion of Many Key Derivation Servers + No Intrusion of User's Key Storage
Additional Authentication Server [23, 33, 1]	No Collusion of Key Derivation and Authentication Server + No Intrusion of User's Key Storage
Registration-based Encryption [38, 39, 50, 56, 37]	No Subversion of Server's Random/Reference String + No Intrusion of User's Key Storage
This Chapter	No Intrusion of User's Key Storage

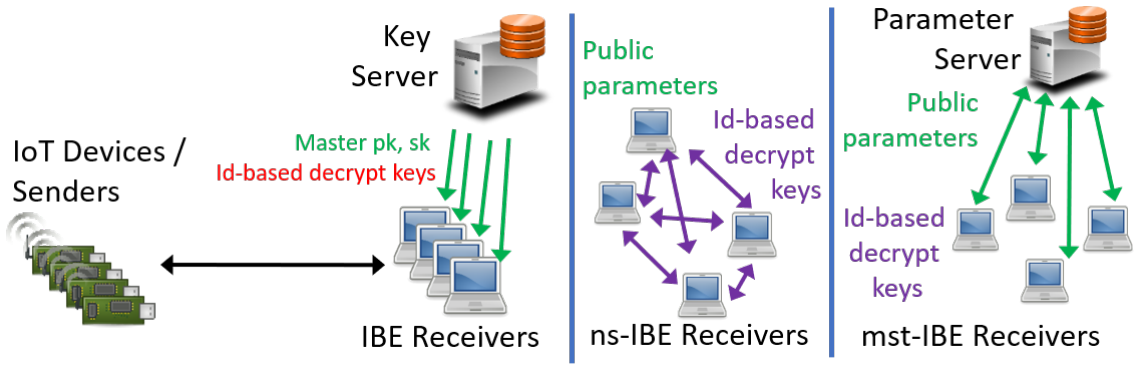


Figure 3.1: Key Derivation in IBE, ns-IBE, and mst-IBE

implemented on IoT devices (see, e.g., [67, 32, 84, 83]), our protocols could be used for distributed key derivation in many IoT and/or cloud computing applications, where resource-constrained IoT devices encrypt their data and much more computationally powerful data centers and/or application servers engage in key derivation and decrypt (see, e.g., [24]). For these applications, low encryption runtime is usually a much more important requirement than key derivation runtime, and the number of IoT devices sending encrypted data is typically much larger than the number of data centers and/or application servers.

We stress that our schemes are the first to not use a common random string, common reference string, or multiple non-colluding servers. In Table 1, we detail a comparison with previous work with respect to trust assumptions.

Our solutions are essentially optimal on our main performance metric (i.e., overhead on IBE encryption) by design, since they require no or minimal changes to the encryption algorithm for a large class of IBE schemes. Still, we produced a software implementation of our two distributed key derivation protocols, and our performance analysis confirms that they perform well on our secondary metric: the protocols' latency scales well with the number of decrypting receivers, due to the protocols' high degree of concurrency.

3.1 Definitions and Two Classes of IBE schemes

In this section, we introduce basic notations, recall the formal definition of identity-based encryption (IBE) schemes, and formally define two new classes of IBE schemes: dhr-IBE, further satisfying newly defined properties of decentralized setup, homomorphic key derivation, and re-encryptable ciphertexts, and vdhr-IBE, further satisfying verifiability of public keys and decryption keys. We also show that a well-known IBE scheme from [91] belongs to both of these classes.

3.1.1 Basic Notations

By 1^λ we denote the security parameter (in unary), and by $\mathsf{negl}(\lambda)$ we denote a function that is negligible in λ (i.e., grows smaller than any inverse polynomial in λ). The expression $y \leftarrow T$ denotes the probabilistic process of randomly and independently choosing y from set T. The expression $y \leftarrow A(x_1, x_2, \ldots)$ denotes the probabilistic process of running algorithm A on input $x_1, x_2, \ldots$ and any necessary random coins, and obtaining y as output. The notation $A^{B(z,\cdot)}(x_1, x_2 \ldots)$ denotes an algorithm A that, during its execution, can also make calls to oracle algorithm $B(z, \cdot)$, as follows: A issues query w_i, and the output returned by execution of algorithm B on input (z, w_i) is made available to A. The expression $(y, tr) \leftarrow P(x_1, x_2, \ldots)$ denotes the probabilistic process of running a multi-party interactive protocol P, taking as input $x_1, x_2, \ldots$ and any necessary random coins, where y denotes the protocol parties' outputs at the end of this protocol's execution, and tr is the transcript of exchanged messages. By aux we denote an algorithm's auxiliary input, often used to keep state or history among different executions of a party's code. The expression $\mathrm{Prob}\,[\, y \leftarrow T; (y, tr) \leftarrow P(x_1, x_2, \ldots) : E\,]$ denotes the probability of event E after the sequential execution of the random processes $y \leftarrow T$ and $(y, tr) \leftarrow P(x_1, x_2, \ldots)$.

3.1.2 Identity-based Encryption

An IBE scheme [15] is a 4-tuple of randomized algorithms ($Setup$, $KeyDer$, Enc, Dec), satisfying the following syntax:

- on input security parameter 1^λ, the *setup* algorithm $Setup$ returns public parameters pp and master secret key msk
- on input pp, msk and an identity string ID, the *key-derivation* algorithm $KeyDer$ returns the decryption key dk corresponding to ID
- on input pp, ID and message M, the *encryption* algorithm Enc returns ciphertext C
- on input pp, dk and ciphertext C, the *decryption* algorithm Dec returns message M' or failure symbol $\perp$.

Informally, the IBE scheme's *decryption correctness requirement* says that for any message M in the message space and identity ID in the identity space, after honestly executing $Setup$, $KeyDer$, Enc and Dec, the event $M = M'$, denoting correctness of decryption, holds with probability 1. Moreover, the IBE scheme's *IND-ID-CPA-security* requirement (i.e., the security of IBE ciphertexts in the presence of a chosen plaintext attack) states that no efficient adversary can tell which of two chosen messages was encrypted as a given challenge ciphertext with respect to a given identity string, even if the adversary can corrupt a polynomial number of different users and obtain their decryption keys.

3.1.3 Two IBE classes: dhr-IBE and vdhr-IBE

Dhr-IBE schemes. Our first class of schemes, called dhr-IBE schemes, consists of IBE schemes ($Setup$, $KeyDer$, Enc, Dec) with the following additional properties: (a) the set

of master secret keys is a commutative algebraic group, where we denote as + the group operation; and (b) there exist 5 additional algorithms (*CommonSetup*, *KeySetup*, *pCombine*, *kCombine* and *cCombine*), satisfying 3 natural properties (decentralized setup, (+)-homomorphic key derivation, and re-encryptable ciphertexts), which we now formally define.

The first additional algorithm, called *CommonSetup* and generating public parameters and/or group values, and the second additional algorithm, called *KeySetup* and generating a secret key which is only known by the party running this algorithm, have the following syntax:

- On input security parameter 1^λ, and a message length 1^{ℓ_m} and an identity length 1^{ℓ_i}, *CommonSetup* returns public parameters pp_0.
- On input public parameters pp_0, *KeySetup* returns additional public parameters pp_1 and secret key sk.

We note that the output of a run of these two additional algorithms is intended to be equivalent to a run of *Setup*. Formally, we define the following property:

(1) (Decentralized Setup) The following two distributions are equal:
- $\{(pp, msk) \leftarrow Setup(1^\lambda, 1^{\ell_m}, 1^{\ell_i}) : (pp, msk)\}$
- $\{\ pp_0 \leftarrow CommonSetup(1^\lambda, 1^{\ell_m}, 1^{\ell_i});$
  $(pp_1, sk) \leftarrow KeySetup(pp_0) : ((pp_0, pp_1), sk)\}$

The third, fourth, and fifth additional algorithms are meant to homomorphically combine two public parameter tuples and two decryption keys so as to realize a homomorphism between any two underlying secret keys, and have the following syntax:

- on input pp_0 and public parameters pp_1 and pp_2, algorithm *pCombine* returns combined public parameters *cpp*;
- on input pp_0 and decryption keys dk_1, dk_2 derived from identity ID and secret keys sk_1, sk_2, algorithm *kCombine* returns a combined decryption key dk;

- on input pp_0, ciphertext c for public parameters pp_1, and secret key sk_2, algorithm $cCombine$ returns ciphertext c'.

Based on these algorithms, we define the properties:

(2) ((+)-homomorphic key derivation) For any identity string ID, and any pp_0 returned as output by $CommonSetup$, these two distributions are equal:

- $\{(pp_i, sk_i) \leftarrow KeySetup(pp_0), i = 1, 2$

  $sk \leftarrow sk_1 + sk_2;$

  $dk \leftarrow KeyDer(pp_0, sk, ID);$

  $pp \leftarrow pCombine(pp_0, pp_1, pp_2) : (pp, dk)\}$

- $\{(pp_i, sk_i) \leftarrow KeySetup(pp_0), i = 1, 2$

  $dk_i \leftarrow KeyDer(pp_0, sk_i, ID), i = 1, 2$

  $dk \leftarrow kCombine(pp_0, dk_1, dk_2);$

  $pp \leftarrow pCombine(pp_0, pp_1, pp_2) : (pp, dk)\}$

(3) (Re-encryptable ciphertexts). For any identity ID, any message M, and any pp_0 output by $CommonSetup$, the following two distributions are equal:

- $\{(pp_i, sk_i) \leftarrow KeySetup(pp_0), i = 1, 2$

  $cpp \leftarrow pCombine(pp_0, pp_1, pp_2)$

  $C \leftarrow Enc(pp_0, cpp, ID, m) : C\}$

- $\{(pp_i, sk_i) \leftarrow KeySetup(pp_0), i = 1, 2$

  $C \leftarrow Enc(pp_0, pp_1, ID, m)$

  $C' \leftarrow cCombine(pp_0, sk_2, C) : C'\}$

Note that in the above definition of dhr-IBE schemes: property (1) formalizes the requirement that the $CommonSetup$ and $KeySetup$ algorithms behave exactly as the $Setup$ algo-

rithm; property (2) formalizes the (+)-homomorphism requirement for the keys produced by the *KeyDer* algorithm; and property (3) formalizes the requirement that ciphertexts produced by the *Enc* algorithm with respect to public parameters pp_1 can be re-encrypted, using secret key sk_2, with respect to public parameters obtained as the combination of pp_1 and pp_2.

Vdhr-IBE schemes. Our second class of schemes called vdhr-IBE schemes, consists of dhr-IBE schemes additionally satisfying 2 natural properties (public key verifiability and decryption key verifiability), implying that each receiver's contributions to the master public key and to another receiver's decryption key are verifiable. More formally, vdhr-IBE schemes are defined as dhr-IBE schemes for which there exist 2 additional algorithms (*pkVerify* and *dkVerify*), which have the following syntax:

- on input security parameter 1^λ, public parameters pp, and contribution public key pk_i, algorithm *pkVerify* returns a bit $b \in \{0,1\}$, where $b = 1$ (resp., $b = 0$) denotes that pk_i was (resp., was not) correctly generated;
- on input security parameter 1^λ, public parameters pp, and contribution decryption key dk_i, algorithm *dkVerify* returns a bit $b \in \{0,1\}$, where $b = 1$ (resp., $b = 0$) denotes that dk_i was (resp., was not) correctly generated;

and the following properties:

1. (Public key verifiability): Algorithm *pkVerify* returns 1 with probability 1 if pk_i is a valid output of algorithm *KeySetup* and with negligible probability otherwise.
2. (Decryption key verifiability): Algorithm *dkVerify* returns 1 with probability 1 if dk_i is a valid output of algorithm *KeyDer* and with negligible probability otherwise.

3.1.4 An Example of a dhr-IBE and vdhr-IBE scheme

We show that the scheme in [91], an IBE scheme secure under the bilinear decision Diffie-Hellman assumption, is both a dhr-IBE scheme and a vdhr-IBE scheme. Towards that goal, we start by briefly recalling the *Setup*, *KeyDer*, *Enc*, *Dec* algorithms from Section 4 of [91].

Setup$(1^\lambda, 1^{\ell_m}, 1^{\ell_i})$: Randomly choose a prime q, and generate a description of q-order additive group $\mathbb{G}$ with scalar multiplication $*$, a description of q-order multiplicative group $\mathbb{G}_T$ with multiplication $\cdot$, and an admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Uniformly choose $g, g_2, u', u_1, ..., u_\ell \leftarrow \mathbb{G}$ and $sk \xleftarrow{\$} \mathbb{Z}_q^*$, and set $g_1 \leftarrow (sk) * g$. Output public parameters $pp = (q, desc(\mathbb{G}, \mathbb{G}_T, e), g, g_2, g_1, u', U)$, where $U = (u_1, \ldots, u_\ell)$, and secret key sk. Here, λ denotes the security parameter, ℓ_m denotes the message length and ℓ_i denotes the ID length.

KeyDer(pp_0, sk, ID): let V denote the set of identity ID bits equal to 1; do:

1. uniformly choose $r \xleftarrow{\$} \mathbb{Z}_q^*$
2. set $d_1 \leftarrow (sk) * g_2 + r * (u' + \sum_{i \in V} u_i)$ and $d_2 \leftarrow r * g$
3. output $d = (d_1, d_2)$.

Enc(pp, ID, M): On input public parameters pp, identity ID and a message M,

1. uniformly choose $t \xleftarrow{\$} \mathbb{Z}_q^*$
2. set $C_1 \leftarrow e(g_1, g_2)^t \cdot M$, $C_2 \leftarrow t * g$ and $C_3 \leftarrow t * (u' + \sum_{i \in V} u_i))$
3. output $CT = (C_1, C_2, C_3)$.

Dec(pp, CT, d): On input public parameters pp, ciphertext $CT = (C_1, C_2, C_3)$ and decryption key $d = (d_1, d_2)$,

1. set $M \leftarrow C_1 \cdot e(d_2, C_3) \cdot e(d_1, C_2)^{-1}$
2. output M.

The scheme in [91] is a dhr-IBE scheme. Recall that Section 5 of [91] proves that this scheme is a secure IBE assuming the bilinear Decisional Diffie-Hellman assumption. To further show that it is a dhr-IBE scheme, following the definition in Section 3.1.3, we show 5 additional algorithms (*CommonSetup*, *KeySetup*, *pCombine*, *kCombine* and *cCombine*) for this scheme, satisfying the decentralized setup, (+)-homomorphic key derivation, and re-encryptable ciphertexts properties.

CommonSetup$(1^\lambda, 1^{\ell_m}, 1^{\ell_i})$: Randomly generate a prime q, a description of q-order additive group $\mathbb{G}$, q-order multiplicative group $\mathbb{G}_T$ and an admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Uniformly choose $g, g_2, u', u_1, ..., u_\ell \leftarrow \mathbb{G}$, and output: $pp_0 = (q, desc(\mathbb{G}, \mathbb{G}_T, e), g, g_2, u', U)$, where $U = (u_1, \ldots, u_\ell)$, and ℓ denotes the ID length.

KeySetup(pp_0): On input above public parameters pp_0,

1. uniformly choose $sk \xleftarrow{\$} \mathbb{Z}_q^*$
2. set $g_1 \leftarrow (sk) * g$
3. output: public parameter $pp = (g_1)$ and secret key sk.

kCombine(pp_0, dk, dk'): rewriting decryption keys dk, dk' as $dk = (d_1, d_2)$ and $dk' = (d_1', d_2')$,

1. set $d_1'' = d_1 + d_1'$ and $d_2'' = d_2 + d_2'$
2. output $dk'' = (d_1'', d_2'')$.

pCombine(pp_0, pp_1, pp_1'): rewriting public parameters pp_1, pp_1' as $pp_1 = (g_1)$ and $pp_1' = (g_1')$,

1. set $g_1'' = g_1 + g_1'$
2. output $cpp = (g_1'')$.

cCombine(pp_0, sk_2, CT): rewriting ciphertext CT $CT = (C_1, C_2, C_3)$,

1. Read g_2 from pp_0 and set $C_1' \leftarrow C_1 \cdot e((sk_2) * g_2, C_2)$.

2. Output $CT' = (C_1', C_2, C_3)$.

Finally, we show that the above algorithms satisfy the 3 properties from the definition of dhr-IBE in Section 3.1.3: decentralized setup, (+)-homomorphic key derivation, and re-encryptable ciphertexts.

Decentralized Setup. This property directly follows by observing that algorithms *Common-Setup* and *KeySetup* defined above contain the same instructions as algorithm *Setup*.

Homomorphic Key Derivation. First, we observe that the first component pp is generated in the same way in both distributions involved in the definition of this property. Then, we consider how the decryption key is generated in these two distributions. Let s denote $u' + \sum_{i \in V} u_i$. In the first distribution, the decryption key is generated as

$$dk = ((sk_1 + sk_2) * g_2 + r * s, r * g);$$

i.e., by adding the secret keys sk_1, sk_2 and then using the result as input to the execution of *KeyDer*. In the second distribution, the decryption key dk is generated as

$$\begin{aligned} &= (sk_1 * g_2 + sk_2 * g_2 + r_1 * s + r_2 * s, r_1 * g + r_2 * g) \\ &= ((sk_1 + sk_2) * g_2 + (r_1 + r_2) * s, (r_1 + r_2) * g); \end{aligned}$$

i.e., by running *KeyDer* twice and then using *kCombine* to combine those two outputs. Thus, since r, r_1, r_2 are uniformly and independently chosen from $\mathbb{Z}_q^*$, the property follows by observing that these two expressions are equally distributed, even conditioned on pp.

Re-encryptable ciphertexts. We consider how the ciphertext is generated in both distributions involved in the definition of this property. As for the previous property, let s denote $u' + \sum_{i \in V} u_i$. In the first distribution, the ciphertext is generated by running *pCombine* to

combine two tuples of public parameters and then *Enc* on the resulting public parameters, thus returning

$$\begin{aligned}(C_1, C_2, C_3) &= (e(g_1 + g_1', g_2)^t \cdot M, t * g, t * s) \\ &= (e((sk_1 + sk_2) * g, g_2)^t \cdot M, t * g, t * s).\end{aligned}$$

In the second distribution, the ciphertext is generated by running *Enc* on the first tuple of public parameters and then modifying the ciphertext using *cCombine* and the second tuple of public parameters, thus resulting in (C_1, C_2, C_3)

$$\begin{aligned}&= (e(sk_1 * g, g_2)^t \cdot M \cdot e(sk_2 * g_2, C_2), t * g, t * s) \\ &= (e(sk_1 * g, g_2)^t \cdot e(sk_2 * g_2, g)^t \cdot M, t * g, t * s) \\ &= (e(sk_1 * g, g_2)^t \cdot e(sk_2 * g, g_2)^t \cdot M, t * g, t * s) \\ &= (e((sk_1 + sk_2) * g, g_2)^t \cdot M, t * g, t * s).\end{aligned}$$

Thus, the property follows by observing that these two expressions are equally distributed.

The scheme in [91] is a vdhr-IBE scheme. Following the definition of vdhr-IBE schemes in Section 3.1.3, we need to show 2 additional algorithms (*pkVerify*, and *dkVerify*) for the IBE scheme in [91], satisfying the public key verifiability, and decryption key verifiability properties.

Towards that goal, we start with the above dhr-IBE scheme just constructed from the IBE scheme in [91], and augment the output of algorithm *KeySetup* with a zero-knowledge proof of knowledge of a scalar $sk \in \mathbb{Z}_q$ such that $g_1 = (sk) * g$, where g is a generator of group $\mathbb{G}$. Well-known techniques for this include a natural adaptation to group $\mathbb{G}$ of Schnorr's zero-knowledge proof of knowledge of discrete logarithms in $\mathbb{Z}_p^*$ [79], which can be made

non-interactive, and thus publicly verifiable by all receivers, using the Fiat-Shamir heuristic [34].

Then, algorithm *pkVerify* consists of the verification algorithm associated with this non-interactive zero-knowledge proof of knowledge. Then the *public-key verifiability property* follows from the properties of this proof system.

Finally, algorithm *dkVerify* consists of verifying that a receiver's contributions d to the decryption keys of another receiver are correctly computed. This has already been addressed in somewhat related contexts; for instance, Boneh and Franklin [15] mention the use of a pairing-based equality to test the honesty of PKG for their scheme, and similar methods are used for checking decryption keys in [51, 65]. In the case we are considering here (i.e., the IBE scheme in [91]), the pairing-based equality "$e(d_1, g) == e(s, d_2) \cdot e(g_2, g_1)$" suffices, where s denotes $u' + \sum_{i \in V} u_i$. Hence the *decryption key verifiability* property follows.

We also observe that using the simulatability of the d values, as shown in Section 3.2.2, these values can be published in case of a conflict (i.e., receiver i sending an incorrect $d_{i,j}$ value to receiver j, or receiver j incorrectly claiming that receiver i did so). After that, the correctness of these values can be publicly verified by all receivers.

3.2 Ns-IBE schemes from any dhr-IBE scheme

In this section, we show our results on constructing no-server IBE schemes from any dhr-IBE scheme, by keeping the same encryption and decryption algorithms of the original IBE scheme. Our main result is the following

Theorem 1. If there exists a dhr-IBE scheme, as defined in Section 3.1.3, then there exists an ns-IBE scheme, as defined in Section 3.2.1.

As we show that the IBE schemes in [15, 12, 91], after minor changes, belong to the class of dhr-IBE schemes (in the case of [91], we gave a sketch of proof in Section 3.1.4), we obtain the possibility of ns-IBE schemes under any of various hardness assumptions related to Diffie-Hellman-type problems on bilinear maps. In the rest of this section, we prove Theorem 1 by first formally defining ns-IBE schemes in Section 3.2.1 and then show our construction of an ns-IBE scheme and prove its properties in Section 3.2.2.

3.2.1 Ns-IBE schemes: formal definition

We define no-server IBE schemes as IBE schemes where the key derivation algorithm is replaced by a distributed key derivation protocol, to be run between the currently joining receiver and the receivers who previously joined, and the adversary's power, in addition to exfiltrating decryption keys corresponding to chosen identities, is augmented by eavesdropping (i.e., semi-honest behavior) on all messages sent or received in this protocol run. More formally, a no-server IBE scheme (briefly, ns-IBE scheme) is a 4-tuple (*CommonSetup*, *dKeyDer*, *Enc*, *Dec*), where algorithms *Enc*, *Dec* are defined as in conventional IBE schemes, and algorithm *CommonSetup* and protocol *dKeyDer* satisfy the following syntax:

- on input security parameter 1^λ, and length parameters $1^{\ell_m}, 1^{\ell_i}$, *CommonSetup* returns public parameters pp_0
- the *distributed key-derivation* protocol $dKeyDer$ is a protocol between the joining (say, i-th) receiver and the current $i-1$ receivers, with communication transcript tr_i, such that at the end of the protocol, the i-th receiver obtains a decryption key dk_i, the j-th receiver, for $j \in [1, i-1]$, updates its decryption key dk_j, and an update (pp_0, cpp_i) of combined public parameters is shared by all i receivers.

An ns-IBE scheme is also required to satisfy the following decryption correctness and security

requirements.

The ns-IBE scheme's *decryption correctness requirement*, informally, says that after one execution of $CommonSetup$, n executions of $dKeyDer$, and one execution of Enc and Dec, the event $M = M'$, denoting correctness of decryption by any of the n receivers, holds with probability 1. More formally, for any number n of users, any message M in the message space $\{0,1\}^{\ell_m}$, any distinct identities $ID_1, \ldots, ID_n$ in the identity space $\{0,1\}^{\ell_i}$, and any $h, h' \in [1, n]$,

$$\Pr\left[\begin{array}{l} pp_0 \leftarrow CommonSetup(1^\lambda, 1^{\ell_m}, 1^{\ell_i}) \\ \text{for } i = 1, \ldots, n: \\ \quad y_i \leftarrow dKeyDer(x_i, y_{i-1}), \text{where } y_0 = \emptyset, \\ \qquad y_i = (tr_i, \{cpp_j, dk_j \mid j \in [1, i]\}), \\ \qquad x_i = (pp_0, ID_1, \ldots, ID_i). \\ M \leftarrow \{0,1\}^{\ell_m}; C \leftarrow Enc((pp_0, cpp_h), ID_{h'}, M); \\ M' \leftarrow Dec((pp_0, cpp_h), dk_{h'}, C) : M' = M \end{array}\right] = 1.$$

The ns-IBE scheme's *IND-ID-ECPA-security* requirement (i.e., the security of ns-IBE ciphertexts in the presence of eavesdropping and chosen plaintext attacks) extends the IND-ID-CPA-security requirement for a conventional IBE scheme in that the adversary is also given access to the message transcripts that are exchanged during the executions of the distributed key derivation protocol. Specifically, it states that no efficient adversary can tell which of two chosen messages was encrypted as a given challenge ciphertext with respect to a target identity string ID, even if the adversary can exfiltrate decryption keys corresponding to non-target identity strings, and eavesdrop on any messages sent and received during executions of the distributed key derivation protocol, run for each new receiver. More formally, for any efficient adversary $\mathcal{A}$, we define the ns-Atk-Game$_\mathcal{A}$ as the game, where, on input pp

and auxiliary information aux, $\mathcal{A}$ may do a polynomial sequence of steps, each steps being any of these (followed by an arbitrary update of aux):

1. trigger a new receiver in the system, and thus an execution of protocol *dKeyDer* between the new receiver and previous ones, and obtain a transcript tr of all messages sent or received in this protocol run;
2. return a non-target identity $ID_i \in \{0,1\}^{\ell_i} \setminus \{\text{target } ID\}$ and obtain decryption key dk_i;
3. (once) return a target identity $ID \in \{0,1\}^{\ell_i} \setminus \{\text{any previous non-target } ID_i\}$, two same-length messages $m_0, m_1 \in \{0,1\}^{\ell_m}$, obtain challenge ciphertext c_b, encrypting m_b with respect to ID and public parameters, for some random bit b;
4. (once) return a bit d attempting to guess bit b.

The requirement then says that $|p_{atk} - 1/2| \leq \mathsf{negl}(\lambda)$, where p_{atk} is the probability that $d = b$, after the following steps:

1. $pp_0 \leftarrow CommonSetup(1^\lambda, 1^{\ell_m}, 1^{\ell_i})$
2. $(b, d) \leftarrow \text{ns-Atk-Game}_{\mathcal{A}}(pp_0)$

3.2.2 Ns-IBE schemes: construction

In this section, we describe our construction of an ns-IBE scheme from any dhr-IBE scheme. Because the *CommonSetup*, *Enc*, and *Dec* algorithms are defined as the same algorithms as in the dhr-IBE scheme, we only need to describe the distributed key derivation protocol *dKeyDer*, the latter also using the algorithm *KeySetup*, *pCombine* and *kCombine*, from the dhr-IBE scheme. We assume the distributed key derivation protocol is run over a synchronous and authenticated network, with no denial of service attacks.

Protocol description. The main goal of this protocol is to distributively derive new decryption keys for both the currently joining, say, i-th receiver, and the j-th receivers, for $j < i$, who previously joined. Recall that in a conventional IBE scheme, there is a single pair of a master public key and secret key, which is used by a single server to derive one decryption key for each receiver in the system. Instead, in our distributed key derivation protocol *dKeyDer*, the i-th receiver generates its own public and secret keys exactly as a master public key and secret key are generated in the original IBE scheme. From then on, the i-th receiver and the j-th receiver, for any $j < i$, can uses their public and secret keys to derive a contribution to each other's decryption key based on their ID's. Specifically, the i-th receiver:

- runs $KeySetup(pp_0)$ to generate its own public parameters pp_i and secret key sk_i
- runs $KeyDer(pp_0, sk_i, ID_j)$ to derive a contribution $dk_{i,j}$ to the decryption key for receiver with identity ID_j, for $j \leq i$,
- sends pp_i and $dk_{i,j}$ to the j-th receiver, for $j < i$,
- receives similarly generated pp_j and $dk_{j,i}$ from the j-th receivers, for $j < i$.

The assumed homomorphism property will allow combining the contributions to the public keys and to the decryption keys through algorithms $pCombine$ and $kCombine$. Specifically, each receiver:

- using algorithm $pCombine$, combines public parameter contributions pp_j or pp_i into a single set of combined public parameters cpp_i, which will be the same for all i receivers,
- using algorithm $kCombine$, combines decryption key contributions $dk_{j,i}$ or $dk_{i,j}$ into a single decryption key for itself.

As described, the distributed key derivation protocol, when run to admit the i-th receiver in the system, effectively builds an IBE system for which value $msk = sk_1 + ... + sk_i$ is the master secret key, where sk_j is the j-th receiver's secret key generated using the *KeySetup*

algorithm, and the corresponding set of public parameters is obtained after using $pCombine$ on all contributions pp_j, for $j \leq i$.

We observe that the resulting set of public parameters cpp_i is computed by each receiver using communication received by all other receivers. Later, in our construction of an mst-IBE scheme in Section 3.3, these redundant computations and messages are eliminated with the help of a minimum-trust repository server, which does not store any secret key.

More formally, protocol *dKeyDer* goes as follows, where we assume that a designated entity (e.g., the first receiver) has run $pp_0 \leftarrow CommonSetup(1^\lambda, 1^{\ell_m}, 1^{\ell_i})$:

1. The i-th receiver obtains pp_0 from the j-th receiver, for some $j \in [1, i-1]$.
2. set $tr_i = pp_0$
3. i-th receiver sets

   $(pp_i, sk_i) \leftarrow KeySetup(pp_0)$

   $dk_i \leftarrow KeyDer(pp_0, sk_i, ID_i)$

   $cpp_i \leftarrow pp_i$
4. for $j = 1, \ldots, i-1$,

   i-th receiver

   sets $dk_{i,j} \leftarrow KeyDer(pp_0, sk_i, ID_j)$

   sends $pp_i, dk_{i,j}$ to j-th receiver

   j-th receiver

   obtains $pp_i, dk_{i,j}$ from i-th receiver

   sets $dk_{j,i} \leftarrow KeyDer(pp_0, sk_j, ID_i)$

   sends $pp_j, dk_{j,i}$ to i-th receiver

   sets $dk_j \leftarrow kCombine(pp_0, dk_j, dk_{i,j})$

   sets $cpp_j \leftarrow pCombine(pp_0, cpp_j, pp_i)$

   i-th receiver

obtains $pp_j, dk_{j,i}$ from j-th receiver

sets $dk_i \leftarrow kCombine(pp_0, dk_i, dk_{j,i})$

sets $cpp_i \leftarrow pCombine(pp_0, cpp_i, pp_j)$

$tr_i = tr_i \cup (pp_i, dk_{i,j}, dk_{j,i})$

5. **return:** $(tr_i, \{cpp_j, dk_j \mid j \in [1, i]\})$

Note that, by construction, our ns-IBE scheme satisfies our main desired efficiency requirements: i.e., it does not use any server, and uses the same encryption and decryption algorithm as the assumed dhr-IBE scheme. In the rest of this section, we discuss the remaining properties of the described ns-IBE scheme: decryption correctness and IND-ID-ECPA-security, as defined in Section 3.2.1.

Decryption Correctness. Let $n > 0$ be an integer, let $ID_1, \ldots, ID_n$ be distinct identities, and consider an execution of $pp_0 \leftarrow CommonSetup(1^\lambda, 1^{\ell_m}, 1^{\ell_i})$, followed by an execution of $y_i \leftarrow dKeyDer(x_i, y_{i-1})$, where $y_i = (tr_i, \{cpp_j, dk_j \mid j \in [1, i]\})$, and $x_i = (pp_0, ID_1, \ldots, ID_i)$, for $i = 1, \ldots, n$.

Based on the description of the protocol *dKeyDer*, we note that at the end of the protocol execution, for each $j = 1, \ldots, n$, the decryption key dk_j and the combined public key cpp_j are obtained as follows:

- dk_j is first computed as $KeyDer(pp_0, sk_j, ID_j)$, and then obtained by running $dk_j \leftarrow kCombine(pp_0, dk_j, dk_{h,j})$, for $h = 1, \ldots, j-1$;
- cpp_j is first computed as the value pp_j returned by one execution of *KeySetup*, and then obtained by running $cpp_j \leftarrow pCombine(pp_0, cpp_j, pp_h)$, for $h = 1, \ldots, j-1$.

By the decentralized setup property of dhr-IBE schemes, each pair (pp_j, sk_j) returned after running first *CommonSetup* and then *KeySetup* has the same distribution as if it was output by *Setup*. By the (+)-homomorphic key derivation property of dhr-IBE schemes, for each

$j = 1, \ldots, n$, the pair (cpp_j, dk_j) computed by after the j-th receiver's execution of *kCombine* and *pCombine* in the 'for' loop, has the same distribution as if dk_j was output by *KeyDer* on input $sk_1 + \cdots + sk_j$ and identity ID_j. Finally, the decryption correctness of the ns-IBE scheme follows from the same property of the dhr-IBE scheme with $sk_1 + \cdots + sk_n$ as master secret key.

Security. We now sketch the proof that if the dhr-IBE scheme is IND-ID-CPA-secure, then the described ns-IBE scheme is IND-ID-ECPA-secure.

We assume an efficient adversary $\mathcal{A}$ achieving probability p_{atk} in the IND-ID-ECPA security experiment for the ns-IBE scheme, and show how to construct an efficient adversary $\mathcal{A}'$ achieving a related probability p'_{atk} in the IND-ID-CPA security experiment for the dhr-IBE scheme. We note that the two attack games are almost identical, with some important differences with respect to what information the adversary has access to. Specifically, in the experiment for the dhr-IBE scheme, the adversary has access to decryption keys relative to any (non-target) input identities and a challenge ciphertext, while in the experiment for the ns-IBE scheme, the adversary again has access to the decryption keys and challenge ciphertexts (although specific to the ns-IBE scheme), and additionally to any messages sent and received during executions of the distributed protocol *dKeyDer*, which is run once for each new receiver. Thus, algorithm $\mathcal{A}'$, in addition to running algorithm $\mathcal{A}$, also needs to simulate: (a) any messages sent and received during the execution of protocol *dKeyDer*; (b) correct answers to each $\mathcal{A}$'s identity query ID_j (i.e., decryption key dk_j, which is the sum of decryption key contributions $dk_{i,j}$ sent to ID_j by party ID_i, for all $i \in \{1, \ldots, n\} \setminus \{j\}$); and (c) a challenge ciphertext for the ns-IBE scheme.

Consider the worst-case scenario where adversary $\mathcal{A}$ exfiltrates decryption keys from all receivers but the target receiver. First, we observe that the simulation of (c) can be performed by applying the re-encryptable ciphertext property. Then, to show the simulation of (b), we

observe that each decryption key $dk_{i,j}$ can be simulated in one of two ways, depending on whether party ID_i is the target receiver or not, as follows.

- In the former case (i.e., party ID_i *is* the target receiver), the decryption key $dk_{i,j}$ can be simulated by an execution of algorithm *KeyDer* on input sk_i and ID_j, where sk_i can be obtained by $\mathcal{A}'$ by running *KeySetup* on input pp_0.
- In the latter case (i.e., party ID_i *is not* the target receiver), the decryption key $dk_{i,j}$ can be simulated by a call to oracle *KeyDer* on input identity ID_j.

Finally, we observe that the simulation of (a) follows from that of (b) since all messages sent during executions of the distributed protocol *dKeyDer* can be computed using secret keys of non-target receivers or call to *KeyDer* for the target receiver.

Now, how does algorithm $\mathcal{A}'$ know the index j^* of the target receiver? In general, $\mathcal{A}'$ may actually not know or find out too late, so $\mathcal{A}'$ will randomly choose $h \in \{1, \ldots, n\}$, behave as if ID_h is the target receiver, and hope that $h = j^*$. We can prove that algorithm $\mathcal{A}'$ correctly guesses $h = j^*$ with (large enough) probability $1/n$, by observing that h is randomly chosen in $\{1, \ldots, n\}$, and that all past simulated messages of protocol *dKeyDer* and answers from the *KeyDer* oracle do not leak any information about the value $j^* \in \{1, \ldots, n\}$. Because whenever $\mathcal{A}'$ correctly guesses which party was the target receiver, its simulations of the values in (a), (b), and (c) are successful, and $\mathcal{A}'$ achieves probability $p'_{atk} \geq p_{atk}/n$, as desired.

3.3 Mst-IBE schemes from any vdhr-IBE scheme

In this section, we show our results on constructing mst-IBE schemes (i.e., IBE schemes with minimum server trust) from any vdhr-IBE scheme, by keeping the same encryption and

decryption algorithms of the original scheme. Our main result is the following

Theorem 2. If there exists a vdhr-IBE scheme, as defined in Section 3.1.3, then there exists an mst-IBE scheme, as defined in Section 3.3.1.

As we show that the IBE schemes in [15, 91, 13], after minor changes, belong to the class of vdhr-IBE schemes (in the case of [91], we gave a sketch of proof in Section 3.1.4), we obtain the possibility of mst-IBE schemes under any of the various hardness assumptions related to Diffie-Hellman-type problems on bilinear maps. In the rest of this section, we prove Theorem 2 by first formally defining mst-IBE schemes in Section 3.3.1 and then describe our construction of an mst-IBE scheme and its properties in Section 3.3.

3.3.1 IBE with Minimal Server Trust: definition

We define IBE schemes with minimal server trust by modifying ns-IBE schemes as follows: first, the distributed key derivation protocol (previously run among all receivers) is replaced with a server-based key derivation protocol (where the receivers only interact with the server, which maintains a repository with public parameters and key contributions); then, analogously as before, the adversary's power includes the capability of eavesdropping on all messages exchanged between receivers and the server; finally, the adversary's power additionally includes the capability of forcing malicious behavior (i.e., arbitrary protocol deviation) on any corrupted receivers in their interaction with the server.

More formally, an IBE scheme with minimal server trust (briefly, mst-IBE scheme) is a 4-tuple (*CommonSetup*, *sKeyDer*, *Enc*, *Dec*), where algorithms *Enc* and *Dec* are defined exactly as in conventional IBE schemes, and algorithm *CommonSetup* and protocol *sKeyDer* satisfy the following syntax:

- on input security parameter 1^λ, and and a message length 1^{ℓ_m} and an identity length 1^{ℓ_i}, the *CommonSetup* algorithm returns public parameters pp_0.
- the *server-based key-derivation* protocol $sKeyDer$ is a protocol between n receivers and one server, where the server is trusted to maintain a public repository including public parameters pp; each receiver has an identity string ID_i, and exchanges messages only with the server (and not with other receivers), forming a transcript tr_i.

An mst-IBE scheme is further required to satisfy the following decryption correctness and security requirements, which are obtained similarly as for ns-IBE schemes, with the important differences that we now use protocol $sKeyDer$ instead of $dKeyDer$, and that in protocol $sKeyDer$ we admit malicious party behavior in trying to violate the security property.

3.3.2 IBE with Minimal Server Trust: construction

In this section, we describe our construction of an mst-IBE scheme from any vdhr-IBE scheme. Because the *CommonSetup*, *Enc*, and *Dec* algorithms are defined as the same algorithms as in the vdhr-IBE scheme, we only need to describe the distributed key derivation protocol *sKeyDer*, the latter also using the algorithms *KeySetup*, *pCombine*, *kCombine*, *pkVerify* and *dkVerify* from the vdhr-IBE scheme. We first give an informal description of the main ideas in the protocol, then describe the protocol, and then observe that the resulting scheme satisfies the decryption correctness and IND-ID-MECPA security requirements from Section 3.3.1, thus showing that it is an mst-IBE scheme. We assume the distributed protocol is run over a synchronous network with authenticated channels between server and receivers, and with no denial of service attacks.

Protocol description. Our protocol $sKeyDer$ contains two important updates with respect to protocol $dKeyDer$.

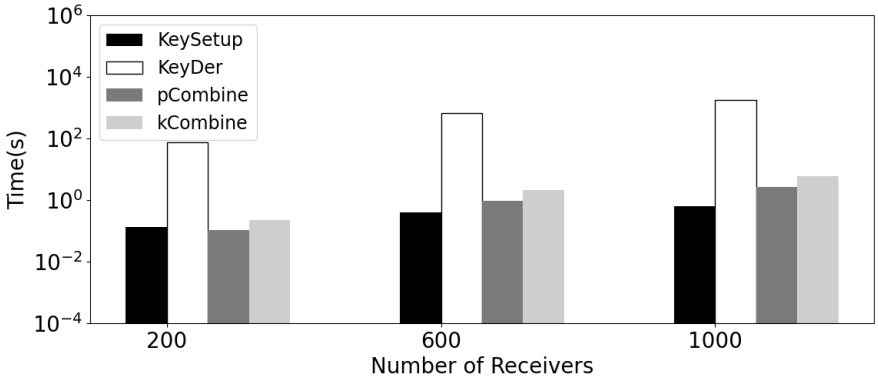
The first update consists of each receiver only exchanging messages with the server, and not with all other receivers. With this modification, the public parameters can be posted on the server's public repository and any updates to the public parameters can be computed only once by the server, instead of having to be computed independently by each receiver.

The second update consists of the server verifying that the receiver's messages are honestly computed. This can be achieved by demanding that the receivers append a zero-knowledge proof of knowledge of secret keys corresponding to their public key contributions, and that the server runs algorithm *pkVerify* to check the honest computation of each receiver's public key contributions, and algorithm *dkVerify* to check the honest computation of each receiver's decryption key contributions.

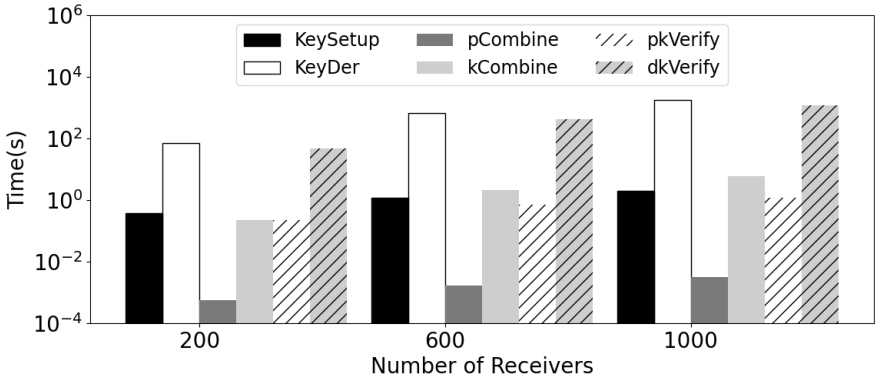
Note that, by construction, our mst-IBE scheme satisfies our main desired efficiency requirement: i.e., it uses the same encryption and decryption algorithm as the assumed vdhr-IBE scheme. In the rest of this section, we briefly discuss the remaining properties of the described mst-IBE scheme: decryption correctness and IND-ID-EMCPA-security.

Decryption Correctness. By observing that when both the server and all receivers honestly follow their instructions in protocol *sKeyDer*, all public parameters and decryption keys are computed exactly as in protocol *dKeyDer*, we obtain that the proof for the decryption correctness property of protocol *sKeyDer* follows from the proof for the same property of protocol *dKeyDer*.

Security. We briefly discuss the additional ideas behind the proof that if the vdhr-IBE scheme is IND-ID-CPA-secure, then the described mst-IBE scheme is IND-ID-EMCPA-secure. We observe that in protocol *sKeyDer*, there are two types of messages sent by receivers: contributions to the public parameters, and contributions to decryption keys, and we note that the correctness of both types of messages are publicly verifiable using the assumed algorithms



(a) Benchmarks for Protocol *dKeyDer*, implemented with Waters IBE



(b) Benchmarks for Protocol *sKeyDer*, implemented with Waters IBE.

Figure 3.2: Protocol Benchmarks

pkVerify and *dkVerify* of the assumed vdhr-IBE scheme. This nullifies the increased power of the adversary in the IND-ID-EMCPA-security attack game associated with our mst-IBE scheme with respect to the IND-ID-ECPA-security attack game.

3.4 Performance Evaluation

We evaluate the performance of our protocols from both theoretical and practical points of view. In the theoretical analysis, we compare the runtime increase factor with respect to standard IBE, of the encryption and user join operations, for both our protocols and those in

Table 3.2: Runtime increase factor, with respect to standard IBE, for encryption and user join in our and previous work

Solution approaches	t(encryption)	t(user join)
multiple key derivation servers	O(# of servers)	O(# of servers)
additional authentication server	small polynomial of security parameter	O(1)
registration-based encryption	large polynomial of security parameter	O(log(# of users))
this chapter	essentially 0	O(# of users)

previous work. As noted in Table 3.2, our protocol targets and manages to keep encryption essentially as efficient as in the original IBE schemes (as motivated by the required efficiency of software to be run on IoT devices). This does come at the cost of having a larger runtime than essentially all of the previous work, for the user join protocol. This trade-off well accommodates the needs of our application use cases of interest; i.e., encryption performed by resource-constrained IoT devices, and receivers running conventional machines.

Recall that in our intended application resource-constrained IoT devices encrypt their data and much more computationally powerful data centers and/or application servers engage in key derivation and decryption (see, e.g., [24]). The feasibility of IBE on some IoT devices has already been demonstrated in previous papers (see, e.g., [32, 67, 83, 84]). We then evaluate the practical performance of our distributed key derivation protocols on a commodity laptop: a MacBook Air (2022) with the following specifications: CPU: Apple M2, Memory: 8 GB, and OS: macOS Sonoma.

We use the PBC library [75] to prototype and simulate our ns-IBE and mst-IBE protocols, using Waters' IBE scheme [91] (briefly, Waters IBE) as the underlying dhr-IBE scheme. We leverage the compilation tools in the "example" folder of the PBC library source code. Our code runs like the given examples. In the experiment, we use a single thread to simulate protocols in order to evaluate the computation time of algorithms that are part of protocol

dKeyDer and protocol *sKeyDer*, and they are implemented using Waters IBE, using numbers as receivers' IDs. We simulate communication by simply copying. We repeat the steps above for 200, 600, and 1000 receivers. We measure total computation times of algorithms $KeySetup$, $pCombine$, $kCombine$, and $KeyDer$ in Protocol *dKeyDer* and Protocol *sKeyDer*.

Figures 4.1c and 4.1b show that the computation times of $KeySetup$, $kCombine$, and $KeyDer$ in Protocol *dKeyDer* are similar to those in Protocol *sKeyDer*. The computation times of zero-knowledge proof and key verification in *sKeyDer* are also shown in Figure 4.1b.

The runtime of the $pCombine$ algorithm is much shorter than in *sKeyDer* due to the elimination of repetitive computation. Excluding verification algorithms (the generation of the zero-knowledge proof and the *pkVerify* and *dkVerify* algorithms), most of the computation time is spent on the $KeyDer$ algorithm (more than 100 times larger than the computation times of $KeySetup$, $pCombine$, and $kCombine$ algorithms).

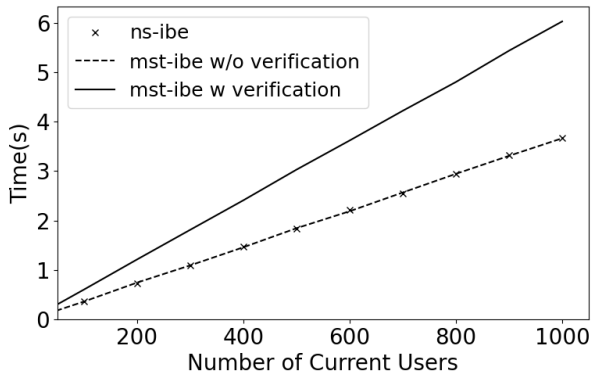


Figure 3.3: User join benchmarks, implemented using Waters IBE

In Figure 3.3, we evaluate the computation time of adding one receiver for different numbers of current receivers in ns-IBE, mst-IBE without or with verification. In all three cases, the computation time t of adding a single receiver increases linearly in the number n of current receivers, with the following measured trendlines, all with $R^2 \geq 0.9999$:

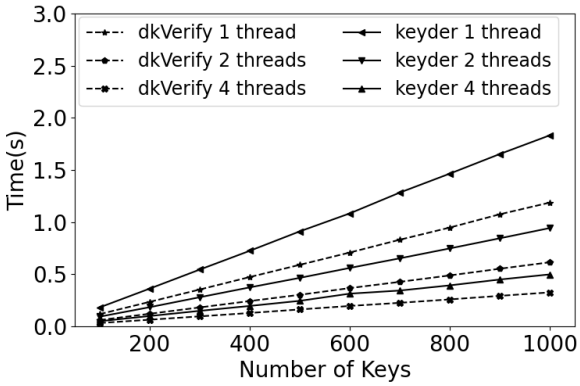


Figure 3.4: Benchmarks for *KeyDer* and *dkVerify* (Waters IBE)

1. $t = 0.0037 \cdot n - 0.0034$ for ns-IBE;

2. $t = 0.0037 \cdot n - 0.0007$ for mst-IBE w/o verification;

3. $t = 0.006 \cdot n - 0.0057$ for mst-IBE with verification.

In Figure 3.4, we evaluate multi-thread executions of the most expensive algorithms used in our protocols, and show that the runtime improvement factor is essentially equal to the number of threads. When coupled with our observation that in our protocols, almost all of the receivers' computations can be run concurrently, this suggests that for both our protocols, concurrency and parallelism can be very effective in reducing runtime and achieving desirable runtime scalability properties.

3.5 Conclusion

We propose and realize two notions of IBE schemes with receiver-assisted key derivation to eliminate or reduce the reliance on server trust. In our first notion of no-server IBE schemes, there is no server, and key derivation is performed through a fully distributed protocol among the decrypting receivers. In our second notion of minimum-server-trust IBE schemes,

there is a single server, which only manages public parameters (not including any common random or reference string), and key derivation is performed by the decrypting receivers, with some help by the server, which is only trusted to keep available and update public parameters. In this latter protocol, a receiver's ciphertext remains secure even if all other receivers collude and act maliciously. Both of our protocols realize these notions through the transformation of any in a large class of IBE schemes with a novel set of natural properties, like key homomorphism, decentralized setup, re-encryptable ciphertexts, and verifiability of contributions to public keys and decryption keys, which we show to be satisfied by well-known schemes in the literature. Moreover, in both of our protocols, the encryption and decryption algorithms are essentially identical to the algorithms in the original IBE schemes. This is our main efficiency requirement and is motivated by using our schemes in IoT applications, where resource-constrained IoT devices encrypt their data and much more powerful data centers engage in key derivation and decryption. On the other hand, our schemes require help from every receiver at every receiver join, and would need further interaction in the presence of receiver crashes. We believe that improving our protocols in these scenarios would be an important direction for further studies.

Chapter 4

Sender-Efficient Registration-Based Encryption in the Public-Parameter Model

Our contributions. We propose a new, sender-efficient, RBE scheme to eliminate or reduce the reliance on server trust. Our scheme, called seRBE, makes black-box use of a variant of IBE schemes, which we define and call dhr-hIBE. This variant includes hierarchical IBE (hIBE) schemes that satisfy 3 additional properties, the most important being that key derivation comes with a sum homomorphism. We observe that many known hIBE schemes from the literature are dhr-hIBE scheme, and we show this for the scheme in [13].

Our scheme works in the public-parameter model, and the generation of these public parameter is transparent, in that the key curator does not keep any secret keys about these parameters that might help to decrypt ciphertexts.

Our seRBE scheme is sender-efficient in the sense that it satisfies runtime-efficiency of ciphertext computation (i.e., comparable to the dhr-hIBE scheme), short ciphertext size (i.e., constant in the identity length and number of receivers), and short public parameter size (i.e., constant in the identity length and number of receivers). These performance requirements are motivated by Internet-of-Things applications, where typically resource-constrained

clients encrypt their data to receiving application servers. Previous RBE scheme targeted a different set of performance requirements, focusing on receiver efficiency, motivated by applications like the anonymous board communication abstraction in [39]. In this line of work, the closest RBE schemes to satisfying our sender-efficiency requirements are the following two works: (1) in [44], the authors constructed an RBE scheme that makes black-box use of prime-order pairings, but required a trusted common reference string of size $O(\sqrt{n})$, where n is the number of receivers; (2) in [31], the authors propose a weakly-efficient RBE scheme in a public parameter model, with ciphertext size quadratic in the identity length.

We also produced a software implementation of our seRBE scheme, where we show practical runtimes for all of its operations. In particular, encryption time takes 2ms on a commodity laptop, without specific optimizations.

Previous work on RBE schemes. In the original papers on RBE [38, 39], Garg et al. introduced the notion of RBE and proposed various constructions with different receiver-efficiency properties and intractability assumptions. Verifying the key curator's actions in RBE schemes was studied in [50]. All of the above constructions make non-black box use of various primitives (e.g., encryption algorithms, hash functions), and are therefore quite far from practical, even after optimizations (see, e.g., [26]). Much further work focused on extending these approaches to attribute-based encryption and functional encryption [56, 37, 102, 101, 17, 35, 101].

Previous work on the IBE key escrow problem. There is also much other work in the literature, beyond registration-based encryption, on this problem. Some of the main approaches include: threshold key derivation by splitting one server into many, using an additional authentication server that knows users' identities, and the use of large-entropy identities.

4.1 Definitions of RBE and hIBE Schemes

In this section, we introduce basic notations, show our formal definitions of sender-efficient registration-based encryption (RBE) schemes, and recall the definition of hierarchical identity-based encryption (hIBE) schemes.

4.1.1 Basic Notations

By 1^λ we denote the security parameter (in unary), and by $\mathsf{negl}(\lambda)$ we denote a function that is negligible in λ (i.e., grows smaller than any inverse polynomial in λ). The expression $y \leftarrow T$ denotes the probabilistic process of randomly and independently choosing y from set T. The expression $y \leftarrow A(x_1, x_2, \ldots)$ denotes the probabilistic process of running algorithm A on input $x_1, x_2, \ldots$ and any necessary random coins, and obtaining y as output. The notation $A^{B(z,\cdot)}(x_1, x_2 \ldots)$ denotes an algorithm A that, during its execution, can also make calls to oracle algorithm $B(z, \cdot)$, as follows: A issues query w_i, and the output returned by execution of algorithm B on input (z, w_i) is made available to A. By aux we denote an algorithm's auxiliary input, often used to keep state or history among different executions of a party's code. The expression $\mathrm{Prob}\,[\, y \leftarrow T; (y, tr) \leftarrow P(x_1, x_2, \ldots) : E\,]$ denotes the probability of event E after the sequential execution of the random processes $y \leftarrow T$ and $(y, tr) \leftarrow P(x_1, x_2, \ldots)$.

4.1.2 Registration-based Encryption

Our main formal definition of RBE schemes is almost identical to the original definition from [38], except for the following differences.

First, in our definition, we do not use any common reference string, while the definition in

[38] does.

Second, in our definition the encryption and decryption algorithms can make a query to the Key Curator for the following reasons: the encryption algorithm needs confirmation of the latest public parameters as well as the receiver's identity registration status; and the decryption algorithm may need to update its decryption key. The definition from [38] does not model these aspects, although its use does require a sender to confirm the latest public parameters and a receiver to occasionally receive a decryption key update, both via the Key Curator.

Finally, in the definition from [38], each receiver generates a pair of public and secret keys using a conventional (i.e., not necessarily identity-based) public-key encryption scheme, while this is not embedded in our definition, mainly to keep the notion closer to identity-based encryption.

We define an RBE scheme in the *public-parameter model* as a 6-tuple of (possibly) probabilistic polynomial-time algorithms ($Setup$, Gen, Reg, $Update$, Enc^{KC}, Dec^{KC}), satisfying the following syntax:

- $pp \leftarrow Setup(1^\lambda, 1^\ell)$: on input a security parameter 1^λ and an identity parameter 1^ℓ, the *setup* algorithm $Setup$ outputs public parameters pp.
- $(dk, ups) \leftarrow Gen(pp, id)$: on input public parameters pp, the *key generation* algorithm Gen outputs a decryption key dk and a server update string ups
- $pp \leftarrow Reg^{[aux]}(pp, id, ups)$: on input public parameters pp, identity id, server update ups, and read/write access to auxiliary information aux, the *registration* algorithm returns updated public parameters pp and updates auxiliary information aux.
- $upr \leftarrow Update^{aux}(pp, id)$: on input public parameters pp, identity id, and read-only

access to auxiliary information aux, the *update* algorithm *Update* returns a receiver update string upr.

- $C \leftarrow Enc^{\mathrm{KC}}(id, M)$: on input a receiver identity id and message M, the *encryption* algorithm *Enc* outputs ciphertext C or a failure symbol $\perp$, possibly after a query to Key Curator KC for the most recent public parameters pp as well as information about receiver identity id.

- $M \leftarrow Dec^{\mathrm{KC}}(id, dk, upr, C)$: this algorithm takes on input receiver identity id, decryption key dk, receiver update string upr, and ciphertext C, the *decryption* algorithm *Dec* outputs message M or a failure symbol $\perp$, after possibly a query to Key Curator KC for updated decryption key for receiver identity id.

The decryption correctness requirement for RBE schemes is defined using an experiment where an adversary can trigger the registration of receivers and choose a target receiver among these, for which it hopes to enforce a decryption failure. More formally, for any efficient adversary $\mathcal{A}$, we define the $\mathrm{CorrExp}_{\mathcal{A}}$ as the experiment, where, after system setup steps $pp = Setup(1^{\lambda}, 1^{\ell}, 1^{d})$ and $i = 0$, on input pp and state information st, $\mathcal{A}$ may do a polynomial sequence of steps, each steps being any of these (followed by an arbitrary update of st):

1. $\mathcal{A}$ issues a new (non-target) identity id, after which receiver id is registered by running $(dk, ups) \leftarrow Gen(pp, id)$, the public parameters are updated as $pp = Reg^{aux}(pp, id, ups)$, and $\mathcal{A}$ obtains pp, dk and ups.

2. (once) $\mathcal{A}$ issues a (target) identity id^* different than all non-target identities, after which receiver id^* is registered by running $(dk, ups) \leftarrow Gen(pp, id^*)$, the public parameters are updated as $pp = Reg^{aux}(pp, id^*, ups)$, index i is incremented, $\mathcal{A}$ issues message $m_i \in \{0,1\}^*$ and obtains pp, dk, ups and ciphertext $c_i = Enc^{\mathrm{KC}}(id^*, m_i)$.

3. $\mathcal{A}$ issues $j \leq i$, and we set $m'_j = \perp$ if $c_j = \perp$, or $m'_j = Dec^{\mathrm{KC}}(id^*, dk, upr, c_j)$ otherwise, where $upr = Update(pp, id^*)$.

We say that an RBE scheme satisfies *decryption correctness* if $p_{atk} \geq 1 - \mathsf{negl}(\lambda)$, where p_{atk} is the probability that $m'_j = m_j$, for all values j issued by $\mathcal{A}$ during a random execution of experiment $\mathrm{CorrExp}_{\mathcal{A}}(1^\sigma, 1^\ell)$.

The security requirement for RBE schemes is defined using an experiment where an adversary can trigger the registration of receivers and choose a target receiver, for which it hopes to guess which, out of two chosen messages, was encrypted as a given challenge ciphertext. More formally, for any efficient adversary $\mathcal{A}$, we define $\mathrm{rSecExp}_{\mathcal{A}}$ as the experiment, where, after system setup step $pp = Setup(1^\lambda, 1^\ell)$ on input pp, and state information st, $\mathcal{A}$ may do a polynomial sequence of steps, each step being any of these (followed by an arbitrary update of st):

1. $\mathcal{A}$ issues a new (non-target) identity id, after which receiver id is registered by running $(dk, ups) \leftarrow Gen(pp, id)$, the public parameters are updated as $pp = Reg^{aux}(pp, id, ups)$ and $\mathcal{A}$ obtains pp, dk and ups.

2. (once) $\mathcal{A}$ issues a (target) identity id^* different than all non-target identities, after which receiver id^* is registered by running $(dk, ups) \leftarrow Gen(pp, id^*)$, the public parameters are updated as $pp = Reg^{aux}(pp, id^*, ups)$ and $\mathcal{A}$ obtains pp and ups.

3. (once) $\mathcal{A}$ issues messages m_0 and m_1, such that $|m_0| = |m_1|$, and obtains $c = Enc^{\mathrm{KC}}(id^*, m_b)$, for some random bit b.

4. (once) $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

We say that an RBE scheme, as above defined, satisfies *IND-id-CPA message security* if

$|p_{atk,r} - 1/2| \leq \mathsf{negl}(\lambda)$, where $p_{atk,r}$ is the probability that $b' = b$ after a random execution of experiment $\mathrm{rSecExp}_{\mathcal{A}}(1^{\sigma}, 1^{\ell})$.

We say that an RBE scheme, as above defined, satisfies (t_{enc}, s_{pp}, s_c)-*sender efficiency* if in a random execution of experiment $\mathrm{CorrExp}_{\mathcal{A}}$ where $i \leq n$, the following requirements hold: (a) the encryption algorithm *Enc* runs in time t_{enc}; (b) the size of the message to update public parameters pp is s_{pp}; and (c) the size of the ciphertext c is s_c. Here, t_{enc}, s_{pp}, s_c are seen as functions of the security parameter λ, the identity length ℓ and the number n of receivers, where we note that it is desirable to achieve t_{enc} polynomial in λ, linear in ℓ but constant with respect to n, and s_{pp}, s_c polynomial in λ, but constant with respect to ℓ, n.

4.1.3 Hierarchical Identity-Based Encryption

An IBE scheme [15, 25] is an encryption scheme where a sender can encrypt using a receiver's identity as a public key, and any authorized receiver can obtain a decryption key associated with its identity.

Hierarchical IBE (briefly, hIBE) schemes generalize this concept by using identity vectors and allowing a holder of a decryption key associated with a prefix vector at a given level to derive decryption keys associated with an identity vector at the next level. The concept of HIBE was introduced by Horwitz and Lynn [57]. Then, Gentry and Silverberg naturally extended the Boneh-Franklin IBE scheme into an hIBE scheme [40]. Other well-known schemes are Waters' hIBE [91], Boneh-Boyen's hIBE [12],and Boneh-Boyen-Goh's hIBE [13].

Many cryptographic schemes with various security goals [19, 16, 18, 99, 53] have been built using hIBE. For instance, Canetti, Halevi, and Katz build a forward secure encryption from hIBE [18]. Moreover, Yao et al. present a scalable forward-secure hIBE [99].

More formally, an hIBE scheme is a 4-tuple of randomized algorithms ($Setup$, $KeyDer$, Enc, Dec), satisfying the following syntax:

- on input security parameter 1^λ, and depth parameter 1^d, the *setup* algorithm $Setup$ returns public parameters pp and master secret key msk, also called $dk_{ID|0}$
- on input pp, an integer $h \in [1, d]$, a decryption key $dk_{ID|h-1}$ for $(h-1)$-depth identity vector $(ID_1, \ldots, ID_{h-1})$, and h-level identity vector $(ID_1, \ldots, ID_h)$, the *key-derivation* algorithm $KeyDer$ returns the decryption key $dk_{ID|h}$ for h-level identity vector $(ID_1, \ldots, ID_h)$.
- on input pp, identity vector $ID = (ID_1, \ldots, ID_h)$ and message m, the *encryption* algorithm Enc returns ciphertext c
- on input pp, decryption key dk for identity vector $ID' = (ID'_1, \ldots, ID'_h)$ and ciphertext c, the *decryption* algorithm Dec returns message m' or failure symbol $\perp$.

Informally, the hIBE scheme's *decryption correctness requirement* says that for any message m in the message space and identity vectors ID, ID' in the identity vector space, after honestly executing $Setup$, $KeyDer$, Enc and Dec, if $ID = ID'$ then the event $m = m'$, denoting correctness of decryption, holds with probability 1.

Moreover, the hIBE scheme's *IND-id-CPA message security* requirement (i.e., the security of hIBE ciphertexts in the presence of a chosen plaintext and chosen identity attacks) states that no efficient adversary can tell which of two chosen messages was encrypted as a given challenge ciphertext with respect to a target identity vector, even if the adversary can obtain decryption keys corresponding to a polynomial number of identity vectors (not a prefix of the target identity vector used in the challenge ciphertext).

With respect to efficiency requirements, we define and use the same notion of (t_{enc}, s_{pp}, s_c)-sender efficiency as already defined for RBE schemes.

More formal definitions can be found in Appendix **??**.

4.2 Dhr-hIBE schemes: a Class of IBE Schemes

In this section we formally define a new class of hIBE schemes: dhr-hIBE, including hIBE scheme that further satisfy properties of decentralized setup, homomorphic key derivation, and re-encryptable ciphertexts. We also show that a well-known hIBE scheme (i.e., the scheme in [13]) belongs to this class.

4.2.1 Definition of Dhr-hIBE schemes

We define a class of schemes, called dhr-hIBE schemes, consisting of hIBE schemes, as defined in Section 4.1.3, with the following additional properties: (a) the set of master secret keys is a commutative algebraic group, where we denote as + the group operation; and (b) there exist 5 additional algorithms (*CommonSetup*, *KeySetup*, *pCombine*, *kCombine*, *cCombine*), satisfying 3 natural properties, called decentralized setup, (+)-homomorphic key derivation, and re-encryptable ciphertexts, which we now formally define.

The first two algorithms split the *Setup* algorithm into two parts (a first part, called *CommonSetup*, generating public parameters and keys that are common to all parties, and a second part, called *KeySetup*, generating a secret key to be only known by the party running this algorithm), with the following syntax:

- On input security parameter 1^λ and depth parameter 1^d, *CommonSetup* returns public

parameters pp_0.

- On input public parameters pp_0, algorithm *KeySetup* returns updated public parameters pp and secret key sk.

We note that the output of a run of these two algorithms is intended to be equivalent to a run of *Setup*. Formally, we define the following property:

(1) (Decentralized Setup) The following two distributions are equal:

- $\{(pp, msk) \leftarrow Setup(1^\lambda, 1^d) : (pp, msk)\}$
- $\{pp_0 \leftarrow CommonSetup(1^\lambda, 1^d);\ (pp, sk) \leftarrow KeySetup(pp_0) : (pp, sk)\}$

The third, fourth, and fifth algorithms are meant to homomorphically combine two public parameter tuples and two decryption keys so as to realize a homomorphism between any two underlying secret keys, and have the following syntax and two properties:

- on input public parameters pp_1, pp_2, algorithm *pCombine*, returns public parameters pp;
- on input decryption keys dk_1, dk_2 derived with respect to the same identity vector and different secret keys sk_1, sk_2, algorithm *kCombine*, returns a decryption key dk;
- on input public parameters pp_0, ciphertext c for public parameters pp_1, and secret key sk_2, algorithm *cCombine* returns ciphertext c'.

(2) ((+)-homomorphic key derivation) For any depth parameter d, any $0 \leq j \leq d$, any $(j-1)$-depth derived key $dk_{ID|j-1}$, any identity vector $(ID_1, \ldots, ID_j)$, and any pp_0 returned as output by *CommonSetup*, these two distributions are equal:

- $\{(pp_i, sk_i) \leftarrow KeySetup(pp_0),\ i = 1, 2;\ sk = sk_1 + sk_2, dk_{ID|0} \leftarrow sk$

  for $h = 1 \ldots, j$,

  $dk_{ID|h} \leftarrow KeyDer(pp_0, dk_{ID|h-1}, ID_1, \ldots, ID_h)$

  $pp \leftarrow pp_0, pp \leftarrow pCombine(pp, pp_i), i = 1, 2\ :\ (pp, dk_{ID|j})\}$

- $\{(pp_i, sk_i) \leftarrow KeySetup(pp_0), dk_0^{(i)} \leftarrow sk_i,\ i = 1, 2$

  for $h = 1 \ldots, j$, and $i = 1, 2$,

  $dk_{ID|h}^{(i)} \leftarrow KeyDer(pp_0, dk_{ID|h-1}^{(i)}, ID_1, \ldots, ID_h)$

  $dk_{ID|j} \leftarrow kCombine(dk_{ID|j}^{(1)}, dk_{ID|j}^{(2)})$

  $pp \leftarrow pp_0, pp \leftarrow pCombine(pp, pp_i), i = 1, 2\ :\ (pp, dk_{ID|j})\}$

(3) (Re-encryptable ciphertexts). For any identity ID, any message M, and any pp_0 output by *CommonSetup*, the following two distributions are equal:

- $\{(pp_i, sk_i) \leftarrow KeySetup(pp_0), i = 1, 2;\ pp \leftarrow pp_0;$

  $pp \leftarrow pCombine(pp, pp_i), i = 1, 2; c \leftarrow Enc(pp, ID_1, \ldots, ID_h, m) : c\}$

- $\{(pp_i, sk_i) \leftarrow KeySetup(pp_0), i = 1, 2$

  $c \leftarrow Enc(pp_1, ID_1, \ldots, ID_h, m);\ c' \leftarrow cCombine(pp_0, c, sk_2) : c'\}$

Note that in the above definition of dhr-IBE schemes: property (1) formalizes the requirement that the output obtained after running the *CommonSetup* and *KeySetup* algorithms is distributed exactly as the output of the *Setup* algorithm; property (2) formalizes the (+)-homomorphism requirement for the keys produced by the *KeyDer* algorithm; and property (3) formalizes the requirement that ciphertexts produced by the *Enc* algorithm with respect to public parameters pp_1 can be re-encrypted with respect to the secret key sk_2 and result in ciphertexts with respect to the combination of pp_1 and pp_2.

4.2.2 Dhr-hIBE from the hIBE scheme in [13]

We noticed that several hIBE schemes in the literature (e.g., those in [91, 12, 13, 68]) are dhr-IBE schemes. As an example, we show this for the hIBE scheme in [13]. Towards that goal, we start by briefly recalling the *Setup*, *KeyDer*, *Enc*, *Dec* algorithms from the scheme in [13]. For simplicity, we only discuss the IND-selective-id-CPA secure scheme, as our discussion continues to work after applying the transformations in [91, 12, 13] to achieve full ID security and even those in [19] to achieve chosen ciphertext security to the recalled scheme.

$pp_0 \leftarrow Setup(1^\lambda, 1^d)$: this algorithm takes input as the security, identity and depth parameters and returns the common public parameters.

1. sample $g, g_2, g_3, u_1, \ldots, u_d \leftarrow \mathbf{G}$, $\alpha \leftarrow \mathbf{Z}_p$ and set $g_1 = g^\alpha$
2. return $pp = (g, g_1, g_2, g_3, u_1, \ldots, u_d)$ and $msk = g_2^\alpha$

$dk_{ID|h} \leftarrow KeyDer(pp, dk_{ID|h-1}, (ID_1, \ldots, ID_h))$, for $h \in [1, d]$:

1. parse $dk_{ID|h-1}$ as $(k_1, k_2, k_{3,h}, \ldots, k_{3,d})$, which is $= (msk, 1, \ldots, 1)$ if $h = 1$
2. sample $r \leftarrow \mathbf{Z}_p$ and set $k_1' \leftarrow k_1 \cdot (k_{3,h})^{ID_h} \cdot (g_3 \cdot \prod_{i=1}^{h-1} u_i^{ID_i})^r$
3. set $k_2' \leftarrow k_2 \cdot g^r$ and $k_{3,j}' \leftarrow k_{3,j} \cdot u_j^r$ for $j = h+1, \ldots, d$
4. return $(k_1', k_2', k_{3,h+1}', \ldots, k_{3,d}')$

$CT \leftarrow Enc(pp, ID_1, \ldots, ID_h, M)$

1. sample $r \leftarrow \mathbf{Z}_p$, set $C_1 \leftarrow e(g_1, g_2)^r$, $C_2 \leftarrow g^r$ and $C_3 \leftarrow (g_3 \cdot \prod_{i=1^h} u_i^{ID_i})^r$
2. return ciphertext $CT = (C_1, C_2, C_3)$

$M \leftarrow Dec(pp, CT, dk)$

1. parse CT as (C_1, C_2, C_3) and dk as $(k_1, k_2, k_{3,h+1}, \ldots, k_{3,d})$

2. set $M \leftarrow C_1 \cdot e(k_2, C_3)/e(C_2, k_1)$ and return message M.

The scheme in [13] is a dhr-hIBE scheme. To show that the above scheme is a dhr-hIBE scheme, following the definition in Section 4.2, we show 5 additional algorithms (*CommonSetup*, *KeySetup*, *pCombine*, *kCombine*, *cCombine*) for this scheme, and prove that the resulting 9 algorithms satisfy the decentralized setup, (+)-homomorphic key derivation, and re-encryptable ciphertexts properties.

$pp_0 \leftarrow CommonSetup(1^\lambda, 1^d)$: this algorithm takes input as the security, identity and depth parameters and returns the common public parameters.

1. sample $g, g_2, g_3, u_1, \ldots, u_d \leftarrow \mathbf{G}$ and return $pp_0 = (g, g_2, g_3, u_1, \ldots, u_d)$

$pp, sk \leftarrow KeySetup(pp_0)$

1. sample $sk \leftarrow \mathbf{Z}_p$, set $g_1 \leftarrow g^{sk}$ and return $(pp = (pp_0, g_1), sk)$

$pp \leftarrow pCombine(pp_1, pp_2)$

1. parse pp_1 as $(pp_0, g_1^{(1)})$ and pp_2 as $(pp_0, g_1^{(2)})$, and set $g_1 \leftarrow g_1^{(1)} \cdot g_1^{(2)}$
2. return $pp = (pp_0, g_1)$

$dk \leftarrow kCombine(dk^{(1)}, dk^{(2)})$

1. parse $dk^{(1)}$ as $(x_1, x_2, x_{3,h+1}, \ldots, x_{3,d})$ and $dk^{(2)}$ as $(y_1, y_2, y_{3,h+1}, \ldots, y_{3,d})$
2. set $k_1 \leftarrow x_1 \cdot y_1$, $k_2 \leftarrow x_2 \cdot y_2$, and $k_{3,j} \leftarrow x_{3,j} \cdot y_{3,j}$ for $j = h+1, \ldots, d$
3. return $dk = (k_1, k_2, k_{3,h+1}, \ldots, k_{3,d})$

$c' \leftarrow cCombine(pp_0, c, sk_2)$

1. parse c as (C_1, C_2, C_3) and pp_0 as $(g, g_2, g_3, u_1, \ldots, u_d)$
2. set $C_1' \leftarrow C_1 \cdot e(sk_2 \cdot C_2, g_2)$ and return $c' = (C_1', C_2, C_3)$

Decentralized Setup. This property directly follows by observing that algorithms *Common-Setup* and *KeySetup* defined above contain the same instructions as algorithm *Setup* in the hIBE scheme from [13].

Homomorphic Key Derivation. First, we observe that the first component pp is generated in the same way in both distributions involved in the definition of this property. Then, we consider how the decryption key is generated in these two distributions. Let y denote $g_3 \cdot \prod_{i=1}^{h} u_i^{ID_i}$. In the first distribution, the decryption key is generated as

$$dk_{ID|h} = (g_2^{(sk_1+sk_2)} \cdot y^r, g^r, u_{h+1}^r, \ldots, u_d^r);$$

i.e., by using the sum of secret keys sk_1, sk_2 as input to an execution of *KeyDer*. In the second distribution, the decryption key is generated as $dk_{ID|h} =$

$$\begin{aligned} &= (g_2^{sk_1} \cdot g_2^{sk_2} \cdot y^{r_1} \cdot y^{r_2}, g^{r_1} \cdot g^{r_2}, u_{h+1}^{r_1+r_2}, \ldots, u_d^{r_1+r_2}) \\ &= (g_2^{sk_1+sk_2} \cdot y^{r_1+r_2}, g^{r_1+r_2}, u_{h+1}^{r_1+r_2}, \ldots, u_d^{r_1+r_2}); \end{aligned}$$

i.e., by running *KeyDer* twice and then using *kCombine* to combine those two outputs. Thus, since r, r_1, r_2 are uniformly and independently chosen from $\mathbb{Z}_q^*$, the property follows by observing that these two expressions are equally distributed.

Re-encryptable ciphertexts. We consider how the ciphertext is generated in both distributions involved in the definition of this property. As for the previous property, let y denote $g_3 \cdot \prod_{i=1}^{h} u_i^{ID_i}$. In the first distribution, the ciphertext is generated by running *pCombine* to combine two tuples of public parameters and then *Enc* on the resulting public parameters, thus returning

$$(C_1, C_2, C_3) = (e((sk_1 + sk_2)g, g_2)^r \cdot M, g^r, y^r)$$

In the second distribution, the ciphertext is generated by running *Enc* on the first tuple of public parameters and then modifying the ciphertext using *cCombine* and the second tuple of public parameters, thus resulting in $(C_1, C_2, C_3) =$

$$\begin{aligned} &= (e(sk_1 \cdot g, g_2)^r \cdot M \cdot e(sk_2 \cdot C_2, g_2), g^r, y^r) \\ &= (e(sk_1 \cdot g, g_2)^r \cdot M \cdot e(sk_2 \cdot g, g_2)^r, g^r, y^r) \\ &= (e((sk_1 + sk_2) \cdot g, g_2)^r \cdot M, g^r, y^r). \end{aligned}$$

The property follows since these two expressions are equally distributed.

4.3 Our RBE scheme in the public-parameter model

In this section, we show our main result on constructing a sender-efficient RBE scheme in the public-parameter model. This result is achieved via a black-box construction starting from any dhr-hIBE scheme, and, in fact, keeps essentially the same encryption and decryption algorithms of the original dhr-hIBE scheme. Formally, our result is the following

Theorem 4.1. Assume there exists a dhr-hIBE scheme, as defined in Section 4.2, satisfying decryption correctness, IND-id-CPA message security, and $(t'_{enc}, s'_{pp}, s'_c)$-sender efficiency. Then there exists (via a black-box construction) an RBE scheme in the public-parameter model, as defined in Section 4.1.2. This scheme, also called seRBE, satisfies:

1. black-box use of the dhr-hIBE scheme's algorithms
2. decryption correctness,
3. IND-id-CPA message security, and
4. (t_{enc}, s_{pp}, s_c)-sender efficiency,

where $t_{enc} = O(t'_{enc})$, $s_{pp} = O(s'_{pp})$, $s_c = O(s'_c)$.

We note that the scheme seRBE constructed via Theorem 4.1 works in the public parameter model (i.e., no need for non-transparent common reference strings).

As we can show that the hIBE schemes in [91, 12, 13, 68] belong to the class of dhr-hIBE schemes (in the case of [13], we gave a sketch of proof in Section 4.2), we obtain the feasibility of public-parameter sender-efficient RBE schemes under hardness assumptions related to Diffie-Hellman-type problems on bilinear maps.

Our seRBE scheme achieves the same sender efficiency than the dhr-hIBE scheme used; in particular, since the scheme in [13] satisfies $s_c = O(1)$ (i.e., ciphertexts of size constant in hierarchy depth d), and seRBE uses hierarchy depth d equal to the identity length ℓ, we obtain that seRBE has ciphertexts of size constant in the identity length. Moreover, since the encryption algorithm for the scheme in [13] has practically efficient runtime, so does the encryption algorithm in seRBE, as showed in our software implementation, described in Section 4.3.5.

In the rest of this section, we prove Theorem 1 by first formally defining seRBE and then proving its properties.

4.3.1 Informal description

Our approach to simplify the trust assumptions on the IBE key derivation server is to avoid centralization of secret key material at this server. Specifically, our seRBE scheme will be a combination of n dhr-hIBE schemes, each one created by a receiver at joining time, so that the seRBE scheme is equivalent to a single hIBE scheme, whose decentralized master secret key $dmsk$, not known to anyone, is the sum of secret keys $dmsk = sk_1 + \cdots + sk_n$, where each

sk_j is contributed by the j-th joining receiver, for $j = 1, \ldots, n$. Using dhr-hIBE will be a critical component for this approach to succeed, since it will help distributing parameter and key setup operations (using the decentralized setup property), generating a system master key as a sum of individual secret key contributions (using the homomorphic key derivation property) and, for the security proof, relating single-user to multi-user ciphertexts (using the re-encryptable ciphertext property).

At system's beginning, the key curator generates some common setup information (with no secret key). From then on, every joining receiver contributes to both the public parameters and the decryption keys for all receivers in the system. A decryption key will be defined as an hIBE decryption key for the receiver's id and using $dmsk$ as a secret key. Thus, the j-th receiver id_j generates a j-th hIBE system with secret key sk_j, uses it to derive a secret partial contribution to the decryption key dk_j associated with id_j, and shares to the key curator the contribution, through sk_j, to decryption keys for all other identities. Because the number of possible identities is exponential in the identity length, the j-th user does not send a separate contribution for all possible id values. Instead, we use the following tree-based contributory key-derivation method, based on known properties of hIBE: first, a complete, ℓ-depth, binary hierarchy tree is used to represent all possible ℓ-bit identities; then, the j-th receiver's secret partial contribution to dk_j is defined to be the value at the leaf of the path labeled as id_j, computed by ℓ applications of $hKeyDer(pp, sk_j, \cdot)$ on input the bits in the tree path labels; finally, the j-th receiver's public partial contributions to decryption keys for all other identities are defined to be the values at the internal nodes of the sibling nodes in the path labeled as id_j, also computed by applications of $hKeyDer(pp, sk_j, \cdot)$. The j-th receiver sends these latter contributions to the key curator who can use them to derive, using the properties of hIBE, a key contribution for any tree leaves corresponding to all other identities of receivers in the system. Note that the j-th user always keeps secret exactly one

contribution to decryption key dk_j from the key curator as well as any coalition of other receivers, and the key curator is merely trusted to pass the decryption key contribution from the next receiver to the previous receivers, and viceversa.

In our scheme the sender, in addition to obtaining the updated public parameters by KC (as in all RBE schemes), also obtains some group membership information about the receiver identity (i.e., whether receiver with queried identity id has registered or not). This can be seen as the analogue of the step in Public-Key Infrastructures, where the sender checks if a public key has been revoked or not with the Certification Authority as part of the encryption process.

4.3.2 Formal description of seRBE

The scheme seRBE uses a dhr-hIBE scheme, as defined in Section 4.2 and here denoted as $(l.Setup, l.KeyDer, l.Enc, l.Dec)$, for $l = dhr.hIBE$, and maintains an ℓ-depth complete binary tree, called hT, where every node is either empty or contains an hIBE decryption key. In hT, for each node, the branch to its left child is labeled as 0 and the branch to its right child is labeled as 1. Then, each ℓ-bit string id uniquely identifies the *id-path*, defined as the nodes reached traversing hT from the root to a leaf, following branches labeled as bits $id[1] \,|\, \ldots \,|\, id[\ell]$. We also define the *id-sibling path*, defined as the set of nodes that are siblings to the nodes in the id-path. Moreover, we define the *i-th sibling forking path labels*, called $s[i]$, as the sequence $id[1] \,|\, \ldots \,|\, id[i-1] \,|\, 1 - id[i]$, namely, the sequence of labels on $i-1$ nodes on the id-path and on the next node on the id-sibling path.

$pp \leftarrow Setup(1^\lambda, 1^\ell)$. // run by KC at startup

1. Initialize auxiliary information $aux = hT$ and group size counter $ct = 0$
2. set $pp^0 \leftarrow dhr.hIBE.CommonSetup(1^\lambda, 1^\ell)$ and output $pp = (ct, pp^0)$.

$(dk, ups) \leftarrow Gen(pp, id)$. // Receiver generating decryption and help keys

1. run $pp_{id}, sk_{id} \leftarrow dhr.hIBE.KeySetup(pp)$.
2. run $hdk^{id} \leftarrow dhr.hIBE.KeyDer(pp_{id}, sk_{id}, id[1, \ell])$.
3. $hdk^{id}_{s[1]} \leftarrow dhr.hIBE.KeyDer(pp_{id}, sk_{id}, 1 - id[1])$
4. for $i \in [2, \ell]$,
   $hdk^{id}_{s[i]} \leftarrow dhr.hIBE.KeyDer(pp_{id}, sk_{id}, s[i])$.
5. output: $(dk = hdk^{id}, ups = (pp_{id}, \{hdk^{id}_{s[i]} : i \in [\ell]\}))$.

$pp \leftarrow Reg^{[aux]}(pp, id, ups)$. // KC registers Receiver id and help keys ups

1. parse ups as $(pp_{id}, \{hdk^{id}_{s[i]} : i \in [\ell]\})$
2. for $i \in [\ell]$: query $aux = hT$ to read the key, if any, stored at node labeled as $s[i]$; if node contains a non-empty key, obtain its key $hdk_{s[i]}$, and update it as $hdk_{s[i]} \leftarrow kCombine(hdk_{s[i]}, hdk^{id}_{s[i]})$; else (i.e., node contains an empty key) set key $hdk_{s[i]} = hdk^{id}_{s[i]}$ into hT.
3. parse pp as (ct, pp^{ct}), set $pp^{ct+1} \leftarrow pCombine(pp_{id}, pp^{ct})$ and $ct = ct + 1$
4. output: $pp = (ct, pp^{ct})$.

$u \leftarrow Update^{aux}(pp, id)$. // KC helps Receiver id update decryption keys

1. read from $aux = hT$ the sequence $hdkSeq_{id} = (hdk_{id[1,i]} : i \in [\ell]$ of keys (some possibly empty) associated with all nodes on the path labeled as $id[1, \ell]$; if $hdkSeq_{id} = \emptyset$, output $\perp$ and halt
2. for all non-empty $hdk_{id[1,i]} \in hdkSeq_{id}$:
   $uhdk^{i}_{id[1,\ell]} \leftarrow dhr.hIBE.KeyDer(pp, hdk_{id[1,i]}, id[1, \ell])$.
3. $uhdk \leftarrow hdk_{id[1,1]}$
4. for $i \in [2, \ell]$: $uhdk \leftarrow kCombine(uhdk, uhdk^{i}_{id[1,\ell]})$
5. parse pp as (ct, pp^{ct}), and output: $u = (ct, uhdk)$.

$c \leftarrow Enc^{\mathrm{KC}}(id, M)$. // Sender encryption

1. ask query id to Key Curator; here, KC replies with latest public parameters pp if identity id was registered or $\perp$ otherwise
2. if KC's reply is $\perp$ then output: $\perp$ and halt
3. parse pp as (ct, pp^{ct}), set $C \leftarrow dhr.hIBE.Enc(pp^{ct}, id[1, \ell], M)$
4. output the ciphertext (ct, C).

$m \leftarrow Dec^{\mathrm{KC}}(id, c, dk, u)$. // Receiver decryption

1. parse c as $c = (ct_s, C)$ and u as $u = (ct_r, uhdk)$
2. if $ct_s > ct_r$ then ask Key Curator for a decryption key update for identity id; here, KC's reply is string u obtained after running $u \leftarrow Update^{aux}(pp, id)$
3. parse u as $u = (ct_r, uhdk)$ and set $dk \leftarrow kCombine(dk, uhdk)$.
4. Output: $M \leftarrow dhr.hIBE.Dec(C, dk)$.

4.3.3 Proof for seRBE's Decryption Correctness

Let $n > 0$ be an integer, let $ID_1, \ldots, ID_n$ be distinct identities, and consider a random execution of experiment $\mathrm{CorrExp}_{\mathcal{A}}(1^{\lambda}, 1^{\ell})$ relative to protocol seRBE. Note that at the very beginning, the experiment starts with an execution of the *CommonSetup* algorithm, and then the adversary $\mathcal{A}$ makes a polynomial number of registration queries, including one target id registration, which create public parameters pp and auxiliary tree hT.

Based on the description of seRBE, we note that during a random execution of experiment $\mathrm{CorrExp}_{\mathcal{A}}$, for each $j = 1, \ldots, n$, after the registration of the j-th receiver, the public parameters pp and the decryption key of the j'-th registered receiver, called $dk_{j'}$, for $j' \leq j$, are obtained as follows:

(a) pp is (1) first computed as the output of *dhr.hibe.CommonSetup* in *Setup*; (2) then, updated as the output of *pCombine* in step 3 of *Reg*, in turn based on the output of *dhr.hibe.KeySetup* in *Gen*, step 1;

(b) $dk_{j'}$ is (1) first computed as the output of *dhr.hibe.KeyDer* in *Gen*, step 2; (2) then, updated as the output of *kCombine* in step 3 of *Dec*, in turn based on the output of *kCombine* in steps 3 and 4 of *Update*.

By the decentralized setup property of dhr-hIBE schemes, for each $j = 1, \ldots, n$, each pair (pp, sk_j) returned after running first *dhr.hIBE.CommonSetup* and then *dhr.hIBE.KeySetup*, as in item (a) above, has the same distribution as when output by *dhr.hIBE.Setup*. By the (+)-homomorphic key derivation property of dhr-hIBE schemes, for each $j = 1, \ldots, n$, and each $j' \leq j$, the pairs $(pp, dk_{j'})$ computed after the j-th receiver's registration, and after updates of the j'-th receiver's decryption keys, through algorithms *Update* and *Dec*, as in item (b) above, have the same distribution as if $dk_{j'}$ was output by *KeyDer* on input $sk_1 + \cdots + sk_j$ and the j'-th receiver's identity. Thus, the decryption correctness of the seRBE scheme follows from the same property of the dhr-hIBE scheme with $sk_1 + \cdots + sk_n$ as master secret key.

Remark. As in previous work, we made the assumption the adversary does not trigger new recever joins (and thus no parameter updates) between the time a ciphertext c is generated by a sender and the time c is received for decryption by its intended receiver. To prevent these types of attacks, the key curator would keep records of previous system states and facilitate a receiver's decryption of an old ciphertext.

4.3.4 Proof for seRBE's IND-id-CPA security

We now prove that if the dhr-hIBE scheme satisfies IND-id-CPA message security, then seRBE satisfies IND-id-CPA message security.

We assume an efficient adversary $\mathcal{A}$ achieving probability $p_{atk,r}$ in the IND-id-CPA message security experiment rSecExp for RBE schemes, and show how to construct an efficient adversary $\mathcal{A}'$ achieving a related probability p'_{atk} in the IND-id-CPA security experiment hibe-SecExp for the dhr-hIBE scheme. We note that the two attack experiments are almost identical, with some important differences with respect to what information the adversary has access to, as we now clarify. In experiment hibe-SecExp, $\mathcal{A}'$ obtains decryption keys relative to any (non-target) input identities and a challenge ciphertext relative to the hIBE scheme. In experiment rSecExp, $\mathcal{A}$ again has access to the decryption keys and challenge ciphertext (although specific to seRBE), updated public parameters pp, and server update string ups for each non-target identity query id.

Thus, algorithm $\mathcal{A}'$, in addition to running algorithm $\mathcal{A}$, produces a simulation of all these quantities, as follows.

When $\mathcal{A}$ produces a non-target identity query id, $\mathcal{A}'$ obtains $(pp_{id}, sk_{id}) \leftarrow KeySetup(pp)$, runs $KeyDer(pp_{id}, sk_{id}, \cdot)$, on the following inputs: the same identity string id, thus obtaining answer ans_{id}, and additional input strings $id[1] \,|\, \ldots \,|\, id[i-1] \,|\, 1 - id[i]$, for all $i \in [2, \ell]$, thus obtaining answers ans_{ups}. Then $\mathcal{A}'$ obtains (dk, ups), after setting $dk = ans_{id}$ and $ups = ans_{ups}$, and the updated public parameters pp after receiver id is registered in the system by the experiment execution $pp = Reg^{aux}(pp, id, ups)$. Here, $\mathcal{A}'$ also saves sk_{id} as part of its state information, for later use.

When $\mathcal{A}$ issues a target identity query id^*, $\mathcal{A}'$ performs to its own oracle $KeyDer$, the queries $id^*[1] \,|\, \ldots \,|\, id^*[j-1] \,|\, 1 - id^*[j]$, for all $j \in [2, \ell]$, thus obtaining answers ans_{ups}. Then $\mathcal{A}'$

obtains ups, after setting $ups = ans_{ups}$ and the updated public parameters pp after receiver id is registered in the system by the experiment execution $pp = Reg^{aux}(pp, id^*, ups)$.

When $\mathcal{A}'$ obtains a challenge ciphertext c for the dhr-hIBE scheme, $\mathcal{A}'$ transforms it into a challenge ciphertext c' for scheme seRBE, as follows. $\mathcal{A}'$ sets $c_0 = c$ and $c_i = cCombine(pp_{id(i)}, sk_{id(i)}, c_{i-1})$, for $i = 1, \ldots, m-1$, where $id(1), \ldots, id(m)$ are the non-target identity queries made by $\mathcal{A}$ before generating the challenge ciphertext. Then, $\mathcal{A}'$ sets challenge ciphertext for $\mathcal{A}$ as $c' = c_m$.

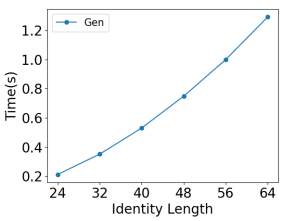
Finally, when $\mathcal{A}$ returns b', $\mathcal{A}'$ returns the same bit b'.

We now observe that adversary $\mathcal{A}'$ perfectly simulates the view of adversary $\mathcal{A}$. In the case of the two steps relative to $\mathcal{A}$ producing a non-target identity string id and $\mathcal{A}$ producing a non-target identity string id^*, this follows since $\mathcal{A}'$ uses the same exact algorithm executions as in the experiment of $\mathcal{A}$'s attack. In the step where a challenge ciphertext is produced, this is followed by the re-encryptable ciphertext property of the dhr-hIBE scheme used.

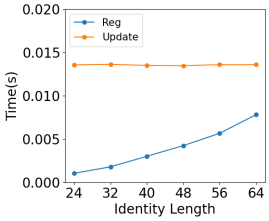
Since $\mathcal{A}'$ perfectly simulates the view of adversary $\mathcal{A}$, we obtain that $p'_{atk} = p_{atk,r}$, as desired.

4.3.5 Results on seRBE's Efficiency Properties

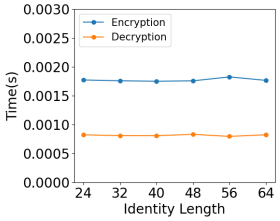
We now show that seRBE satisfies (t_{enc}, s_{pp}, s_c)-sender efficiency for $t_{enc} = O(t'_{enc})$, $s_{pp} = O(s'_{pp})$, $s_c = O(s'_c)$, as claimed in Theorem 1. The claims on t_{enc} and s_c follow by observing that seRBE uses essentially the same encryption algorithm as in the dhr-hIBE scheme, which is constant in the number of receivers n and linear in the identity length ℓ when using [13] as a dhr-hIBE scheme. The claim on s_{pp} follows by observing that seRBE uses the same public parameters as in dhr-hIBE for encryption, which have constant (with respect to ℓ, n) size when using [13] as a dhr-hIBE scheme.



(a) Key Generation



(b) Registration & Update



(c) Encryption & Decryption

We also produced a 'proof-of-concept' software implementation of our seRBE scheme, using the PBC library [75], and measured benchmarks for it. Our code runs like the other examples in PBC library. The computer used for our measurements was a commodity Macbook Air laptop, with operating system macOS Sonoma, chip Apple M2, and 8GB of Memory.

We benchmark our seRBE construction by showing the average run-time of algorithms *Gen*, *Reg*, *Update*, *Enc*, and *Dec*, across 1000 independent invocations of each algorithm. We set the identity length in the range $[24, 64]$.

Our performance results demonstrate that our scheme is lightweight for senders (and, in fact, also for receivers). Both encryption and decryption take less than 2ms and increasing the identity length has no or minimal effect on this number, and thus they are even appealing for resource-constrained devices (e.g., IoT devices). Moreover, the key generation runtime is about 1s, which is quite acceptable because it is run only once by each receiver. The registration runtime is about 10ms, and the update algorithm's runtime is about 15ms. Accordingly, we also estimate that a small-scale server can run the key curator quite efficiently.

4.4 Conclusions

To eliminate or reduce the reliance on server trust on IBE schemes and their potential applications in the Internet of Things, we consider the recently introduced notion of RBE schemes

and accompany it with sender-efficiency requirements that are suitable for these applications. Our main result is the first RBE scheme that satisfies sender efficiency requirements, such as: (1) practical ciphertext computation runtime, using, as a black box, an IBE scheme that has been implemented on resource-constrained IoT devices [67]; (2) short ciphertext (i.e., constant with respect to the identity length and number of receivers); and (3) short public parameters (i.e., constant with respect to the number of receivers). Our scheme makes black-box use of a class of known IBE schemes, which we define and call dhr-hIBE. Here, we also show that the scheme in [13] belongs to this class.

Chapter 5

Summary and Future Works

This dissertation addresses critical issues by both exposing problems and proposing solutions. We expose that information leakage from commercial data centers poses a potential threat to their reliability.

Although data centers with free cooling systems are distributed geographically to mitigate the single point of failure problem, an adversary can still find their vulnerable periods by monitoring local weather. The long-standing key escrow problem of IBE is also a type of single-point-of-failure problem. It prevents many practical usages of IBE. We acknowledge that there are limitations in our studies. For example, we do not have physical access to the data center while investigating data centers. Moreover, we do not achieve polylogarithmic efficiencies in the initial IBE protocol. In the last project, we achieve polylogarithmic efficiency, but it degrades security. In short, we cannot achieve the best efficiency and security at the same time. The findings and results of our projects are summarized as follows.

- We used FPGA-based side channels to investigate the cooling systems of AWS data centers. We set up experiments to reveal PVT variations of FPGAs (process, voltage, temperature). We leverage the temperature effect on the FPGA and DRAM side channels to exploit temperature information leakage.

- The key finding regarding data centers is that most of them utilize free cooling. Free cooling depends on the local weather because it uses outside air. An adversary may

take advantage of it to launch more dangerous attacks and degrade the reliability of data centers.

- For IBE's key escrow problem, we design a key-derivation protocol for IBE to distribute the master secret key to all receivers. It results in a protocol that requires linear-time computation/communication among all receivers while keeping the IBE encryption. However, it is acceptable for receivers with sufficient resources. Note that only senders use the IBE encryption.

- To improve the efficiency of the prior protocol, we construct a registration-based encryption from HIBE, which is a relaxed variant of IBE. It satisfies a subset of efficiency metrics of RBE, but it does not protect unregistered users. It can also be viewed as the protocol allows receivers to revoke the key escrow's decryption capability.

To address future work on reliable and secure distributed systems, we must focus on eliminating single points of failure, improving efficiency, and lowering costs. For data centers, this means continuing to minimize information leakage and providing sufficient redundancy. A recent challenge is the significantly increased power consumption due to intense AI-related computation. FPGA is a hardware candidate for AI workloads. FPGAs are often placed in close proximity to memory, allowing them to avoid the memory access bottleneck that CPUs frequently encounter. This makes FPGAs highly suitable for bypassing the memory wall, a term that describes the growing gap between processor speeds and memory access speeds. However, more FPGA deployments may lead to larger-scale information leakage.

To further improve our registration-based encryption, there are a few potential research directions. We will consider the behavior of malicious users who may attempt to disrupt the protocol computation. An improvement is to require all receivers to provide a zero-knowledge proof of their secrets. Its associated knowledge extractor is useful to complete

a security proof for the dishonest case. The meta-reduction technique in the impossibility result of witness encryption may be used to prove that our registration-based encryption cannot protect unregistered users due to its certain properties. We observe that the update times of public parameters in registration-based encryption are linear in the number of users, and we may use a batch method to amortize it.

There are more advanced topics in applied cryptography. Functional encryption is useful for AI because its receiver decrypts and obtains the result of the function on the encryptor's message input. It can protect the privacy of machine learning algorithms. We believe functional encryption will have a strong impact on the security and privacy of AI. Functional encryption generalizes attribute-based encryption, which generalizes the HIBE. Another challenge is the development of quantum computing, and encryption protocols should be secure against the quantum adversary. How to improve our protocols to be quantum resistant is an interesting topic to explore in the future.

References

[1] Carlisle Adams. "Security Analysis of a Privacy-Preserving Identity-Based Encryption Architecture". In: *Journal of Information Security* (2022).

[2] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. *Data Center Basics: Building, Power, and Cooling*.

[3] AWS. *Amazon EC2 F1 Instance Expands to More Regions, Adds New Features, and Improves Development Tools*. https://aws.amazon.com/about-aws/whats-new/2018/10/.

[4] AWS. *Amazon EC2 Instance Types*. https://aws.amazon.com/ec2/instance-types/.

[5] AWS. *AWS/AWS-FPGA: Official Repository of the AWS EC2 FPGA hardware and software development kit*. https://github.com/aws/aws-fpga.

[6] AWS. *Our Controls*. https://aws.amazon.com/compliance/data-center/controls/.

[7] AWS. *Our Data Centers*. https://aws.amazon.com/compliance/data-center/data-centers/.

[8] AWS. *Regions and Availability Zones*. https://aws.amazon.com/about-aws/global-infrastructure/regions_az/.

[9] *Baxtel*. https://baxtel.com.

[10] Rikke Bendlin, Sara Krehbiel, and Chris Peikert. "How to Share a Lattice Trapdoor: Threshold Protocols for Signatures and (H)IBE". In: *Applied Cryptography and Network Security*. 2013.

[11] BittWare. https://www.bittware.com/fpga/xup-p3r/.

[12] Dan Boneh and Xavier Boyen. "Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles". In: *EUROCRYPT*. 2004.

[13] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. "Hierarchical Identity Based Encryption with Constant Size Ciphertext". In: *EUROCRYPT*. 2005.

[14] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. "Public Key Encryption with Keyword Search". In: *EUROCRYPT*. 2004.

[15] Dan Boneh and Matt Franklin. "Identity-Based Encryption from the Weil Pairing". In: *CRYPTO*. 2001.

[16] Dan Boneh and Jonathan Katz. "Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption". In: *CT-RSA*. 2005.

[17] Pedro Branco, Russell W. F. Lai, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Ivy K. Y. Woo. "Traitor Tracing Without Trusted Authority from Registered Functional Encryption". In: *Advances in Cryptology – ASIACRYPT 2024*. Ed. by Kai-Min Chung and Yu Sasaki. Singapore: Springer Nature Singapore, 2025, pp. 33–66. ISBN: 978-981-96-0891-1.

[18] Ran Canetti, Shai Halevi, and Jonathan Katz. "A Forward-Secure Public-Key Encryption Scheme". In: *EUROCRYPT*. 2003.

[19] Ran Canetti, Shai Halevi, and Jonathan Katz. "Chosen-ciphertext security from identity-based encryption". In: *EUROCRYPT*. 2004.

[20] Liqun Chen, Keith A. Harrison, David Soldera, and Nigel P. Smart. "Applications of Multiple Trust Authorities in Pairing Based Cryptosystems". In: *Infrastructure Security*. 2002.

[21] Zhaohui Cheng, Richard Comley, and Luminita Vasiu. "Remove key Escrow from the Identity-Based Encryption System". In: *Exploring New Frontiers of Theoretical Informatics*. 2004.

[22] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. "Decentralized Multi-Client Functional Encryption for Inner Product". In: *ASIACRYPT*. 2018.

[23] Sherman S. M. Chow. "Removing Escrow from Identity-Based Encryption". In: *PKC*. 2009.

[24] Cisco. *The Internet of things reference model*. Technical report. 2014.

[25] Clifford C. Cocks. "An Identity Based Encryption Scheme Based on Quadratic Residues". In: *IMA Conference on Cryptography and Coding*. 2001.

[26] Kelong Cong, Karim Eldefrawy, and Nigel P. Smart. "Optimizing Registration Based Encryption". In: *Cryptography and Coding*. Ed. by Maura B. Paterson. 2021.

[27] Jun Dai, Michael M. Ohadi, Diganta Das, and Michael G. Pecht. *Optimum Cooling of Data Centers*. 2014.

[28] Datacenterdynamics. *WikiLeaks publishes list of AWS data center locations, colo providers*. https://www.datacenterdynamics.com/en/news/wikileaks-publishes-list-aws-data-center-locations-colo-providers/.

[29] Datacenterknowledge. *AWS Says It's Never Seen a Whole Data Center Go Down*. https://www.datacenterknowledge.com/amazon/aws-says-it-s-never-seen-whole-data-center-go-down.

[30] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. "Data Center Energy Consumption Modeling: A Survey". In: *IEEE Communications Surveys Tutorials* (2016).

[31] Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. “Efficient Laconic Cryptography from Learning with Errors”. In: *EUROCRYPT 2023*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14006. LNCS. Springer, 2023, pp. 417–446.

[32] Sayon Duttagupta, Dave Singelée, and Bart Preneel. “T-HIBE: A Novel Key Establishment Solution for Decentralized, Multi-Tenant IoT Systems”. In: *19th Annual IEEE Consumer Communications & Networking Conference (CCNC)*. IEEE Press, 2022.

[33] Keita Emura, Shuichi Katsumata, and Yohei Watanabe. “Identity-Based Encryption with Security Against the KGC: A Formal Model and Its Instantiation from Lattices”. In: *ESORICS*. 2019.

[34] Amos Fiat and Adi Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *CRYPTO*. 1986.

[35] Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. “Registered (Inner-Product) Functional Encryption”. In: *Advances in Cryptology – ASIACRYPT 2023*. Ed. by Jian Guo and Ron Steinfeld. Springer Nature Singapore, 2023.

[36] Xing Gao, Zhang Xu, Haining Wang, Li Li, and Xiaorui Wang. “Reduced Cooling Redundancy: A New Security Vulnerability in a Hot Data Center”. In: *NDSS*. 2018.

[37] Rachit Garg, George Lu, Brent Waters, and David J. Wu. “Reducing the CRS Size in Registered ABE Systems”. In: *CRYPTO*. 2024.

[38] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. “Registration-Based Encryption: Removing Private-Key Generator from IBE”. In: *TCC*. 2018.

[39] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. "Registration-Based Encryption from Standard Assumptions". In: *PKC*. 2019.

[40] Craig Gentry and Alice Silverberg. "Hierarchical ID-Based Cryptography". In: *ASIACRYPT*. Ed. by Yuliang Zheng. Springer Berlin Heidelberg, 2002.

[41] Ilias Giechaskiel, Kasper Bonne Rasmussen, and Jakub Szefer. "C3APSULe: Cross-FPGA Covert-Channel Attacks through Power Supply Unit Leakage". In: *IEEE Symposium on Security and Privacy*. 2020.

[42] Ilias Giechaskiel, Shanquan Tian, and Jakub Szefer. "Cross-VM Covert-and Side-Channel Attacks in Cloud FPGAs". In: *ACM TRETS* (2022).

[43] Ilias Giechaskiel, Shanquan Tian, and Jakub Szefer. "Cross-VM Information Leaks in FPGA-Accelerated Cloud Environments". In: *IEEE HOST*. 2021.

[44] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. "Efficient registration-based encryption". In: *CCS*. 2023.

[45] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. "Are Cloud FPGAs Really Vulnerable to Power Analysis Attacks?" In: *DATE*. 2020.

[46] Dennis R. E. Gnad, Fabian Oboril, Saman Kiamehr, and Mehdi B. Tahoori. "An Experimental Evaluation and Analysis of Transient Voltage Fluctuations in FPGAs". In: *IEEE VLSI* (2018).

[47] Dennis R.E. Gnad, Fabian Oboril, Saman Kiamehr, and Mehdi B. Tahoori. "Analysis of transient voltage fluctuations in FPGAs". In: *FPT*. 2016.

[48] Íñigo Goiri, Thu D. Nguyen, and Ricardo Bianchini. "CoolAir: Temperature- and Variation-Aware Management for Free-Cooled Datacenters". In: *ACM ASPLOS*. 2015.

[49] Google. *Google Data Centers*. https://www.google.com/about/datacenters/.

[50] Rishab Goyal and Satyanarayana Vusirikala. “Verifiable Registration-Based Encryption”. In: *CRYPTO*. 2020.

[51] Vipul Goyal. “Reducing Trust in the PKG in Identity Based Cryptosystems”. In: *CRYPTO*. 2007.

[52] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. “Black-Box Accountable Authority Identity-Based Encryption”. In: *CCS*. 2008.

[53] Matthew D. Green and Ian Miers. “Forward Secure Asynchronous Messaging from Puncturable Encryption”. In: *IEEE Symposium on Security and Privacy*. 2015.

[54] 7 Series FPGALibraries Guide. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug953-vivado-7series-libraries.pdf.

[55] Mordechai Guri, Matan Monitz, Yisroel Mirski, and Yuval Elovici. “BitWhisper: Covert Signaling Channel between Air-Gapped Computers Using Thermal Manipulations”. In: *IEEE CSF*. 2015.

[56] Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. “Registered Attribute-Based Encryption”. In: *EUROCRYPT*. 2023.

[57] Jeremy Horwitz and Ben Lynn. “Toward Hierarchical Identity-Based Encryption”. In: *EUROCRYPT*. 2002.

[58] Taras Iakymchuk, Maciej Nikodem, and Krzysztof Kępa. “Temperature-based covert channel in FPGA systems”. In: *ReCoSoC*. 2011.

[59] Infotech. *Top 10 Energy-Saving Tips for a Greener Data Center*. http://static.infotech.com/downloads/samples/070411_premium_oo_greendc_top_10.pdf.

[60] Uptime Institute. *Uptime Institute data shows outages are common, costly, and preventable*. https://uptimeinstitute.com/data-center-outages-are-common-costly-and-preventable.

[61] Intel. *The State of Data Center Cooling*. http://www.ceclimited.com/sites/all/themes/creative/state-of-date-center-cooling.pdf.

[62] Mohammad A. Islam and Shaolei Ren. "Ohm's Law in Data Centers: A Voltage Side Channel for Timing Power Attacks". In: *ACM CCS*. 2018.

[63] Mohammad A. Islam, Shaolei Ren, and Adam Wierman. "Exploiting a Thermal Side Channel for Power Attacks in Multi-Tenant Data Centers". In: *ACM CCS*. 2017.

[64] Mohammad A. Islam, Luting Yang, Kiran Ranganath, and Shaolei Ren. "Why Some Like It Loud: Timing Power Attacks in Multi-Tenant Data Centers Using an Acoustic Side Channel". In: *Proc. ACM Meas. Anal. Comput. Syst.* (2018).

[65] Aniket Kate and Ian Goldberg. "Distributed Private-Key Generators for Identity-Based Cryptography". In: *Security and Cryptography for Networks*. 2010.

[66] Christos Kozyrakis. "Resource efficient computing for warehouse-scale datacenters". In: *DATE*. 2013.

[67] Sam Kumar, Yuncong Hu, Michael P. Andersen, Raluca Ada Popa, and David E. Culler. "JEDI: Many-to-Many End-to-End Encryption and Key Delegation for IoT". In: *USENIX Security Symposium*. 2019.

[68] Allison Lewko and Brent Waters. "New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts". In: *Theory of Cryptography*. Ed. by Daniele Micciancio. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 455–479. ISBN: 978-3-642-11799-2.

[69] Allison B. Lewko and Brent Waters. "Decentralizing Attribute-Based Encryption". In: *EUROCRYPT*. 2011.

[70] Kaspar Matas, Tuan Minh La, Khoa Dang Pham, and Dirk Koch. "Power-hammering through Glitch Amplification – Attacks and Mitigation". In: *IEEE FCCM*. 2020.

[71] Seyedeh Sharareh Mirzargar and Mirjana Stojilović. "Physical Side-Channel Attacks and Covert Communication on FPGAs: A Survey". In: *ACM FPL*. 2019.

[72] Shayan Moini, Shanquan Tian, Daniel Holcomb, Jakub Szefer, and Russell Tessier. "Remote Power Side-Channel Attacks on BNN Accelerators in FPGAs". In: *DATE*. 2021.

[73] NORTEK. *Free Cooling Concepts for Data Centers*. https://www.nortekair.com/wp-content/uploads/2017/01/Free-Cooling-Concepts-for-Data-Centers.pdf.

[74] Kenneth G. Paterson and Sriramkrishnan Srinivasan. "Security and Anonymity of Identity-Based Encryption with Multiple Trusted Authorities". In: *Pairing*. 2008.

[75] PBC. *The Pairing-Based Cryptography Library*. https://crypto.stanford.edu/pbc/.

[76] Sattam S. Al-Riyami and Kenneth G. Paterson. "Certificateless Public Key Cryptography". In: *ASIACRYPT*. 2003.

[77] Behzad Salami, Erhan Baturay Onural, Ismail Emir Yuksel, Fahrettin Koc, Oguz Ergin, Adrian Cristal Kestelman, Osman Unsal, Hamid Sarbazi-Azad, and Onur Mutlu. "An Experimental Study of Reduced-Voltage Operation in Modern FPGAs for Neural Network Acceleration". In: *IEEE/IFIP DSN*. 2020.

[78] Falk Schellenberg, Dennis R.E. Gnad, Amir Moradi, and Mehdi B. Tahoori. "An inside job: Remote power analysis attacks on FPGAs". In: *DATE*. 2018.

[79] Claus-Peter Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *CRYPTO*. 1989.

[80] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *CRYPTO*. 1984.

[81] Zhihui Shao, Mohammad A. Islam, and Shaolei Ren. "A First Look at Thermal Attacks in Multi-Tenant Data Centers". In: *SIGMETRICS Perform. Eval. Rev.* (2019).

[82] Cheng Shen, Tian Liu, Jun Huang, and Rui Tan. "When LoRa Meets EMR: Electromagnetic Covert Channels Can Be Super Resilient". In: *2021 IEEE Symposium on Security and Privacy*. 2021.

[83] David Shur, Giovanni Di Crescenzo, Ta Chen, Zahir Patni, Yow-Jian Lin, Scott Alexander, Benjamin Flin, and Rob Levonas. "Energy-efficient Hardening of the SEDIMENT Methodology for Scalable IoT Network Security". In: *IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE Press, 2024.

[84] David Shur, Giovanni Di Crescenzo, Qinqing Zhang, Ta Chen, Rajesh Krishnan, Yow-Jian Lin, Zahir Patni, Scott Alexander, and Gene Tsudik. "SEDIMENT: An IoT-device-centric Methodology for Scalable 5G Network Security". In: *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE Press, 2022.

[85] Shanquan Tian, Ilias Giechaskiel, Wenjie Xiong, and Jakub Szefer. "Cloud FPGA Cartography using PCIe Contention". In: *IEEE FCCM*. 2021.

[86] Shanquan Tian, Shayan Moini, Adam Wolnikowski, Daniel Holcomb, Russell Tessier, and Jakub Szefer. "Remote Power Attacks on the Versatile Tensor Accelerator in Multi-Tenant FPGAs". In: *IEEE FCCM*. 2021.

[87] Shanquan Tian and Jakub Szefer. "Temporal Thermal Covert Channels in Cloud FPGAs". In: *ACM FPGA*. 2019.

[88] Shanquan Tian, Wenjie Xiong, Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. "Fingerprinting Cloud FPGA Infrastructures". In: *ACM FPGA*. 2020.

[89] Timeanddate. https://www.timeanddate.com/.

[90] Guosai Wang, Lifei Zhang, and Wei Xu. "What Can We Learn from Four Years of Data Center Hardware Failures?" In: *IEEE/IFIP DSN*. 2017.

[91] Brent Waters. "Efficient Identity-Based Encryption without Random Oracles". In: *EUROCRYPT*. 2005.

[92] Wikileak. *Amazon Atlas*. https://wikileaks.org/amazon-atlas/.

[93] Xilinx. *Using Constraints*. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug903-vivado-using-constraints.pdf.

[94] Xilinx. *Vivado*. https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2020-2.html.

[95] Wenjie Xiong, Nikolaos Athanasios Anagnostopoulos, André Schaller, Stefan Katzenbeisser, and Jakub Szefer. "Spying on Temperature using DRAM". In: *DATE*. 2019.

[96] Hong Xu, Chen Feng, and Baochun Li. "Temperature Aware Workload Management in Geo-distributed Datacenters". In: *ICAC 13*. USENIX Association, 2013.

[97] Zhang Xu, Haining Wang, and Zhenyu Wu. "A Measurement Study on Co-residence Threat inside the Cloud". In: *USENIX Security*. 2015.

[98] Zhang Xu, Haining Wang, Zichen Xu, and Xiaorui Wang. "Power Attack: An Increasing Threat to Data Centers". In: *NDSS*. 2014.

[99] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. "ID-Based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption". In: *CCS*. 2004.

[100] Mark Zhao and G. Edward Suh. "FPGA-Based Remote Power Side-Channel Attacks". In: *IEEE Symposium on Security and Privacy*. 2018.

[101] Ziqi Zhu, Jiangtao Li, Kai Zhang, Junqing Gong, and Haifeng Qian. "Registered Functional Encryptions from Pairings". In: *Advances in Cryptology – EUROCRYPT 2024*. Ed. by Marc Joye and Gregor Leander. Springer Nature Switzerland, 2024.

[102] Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. "Registered ABE via Predicate Encodings". In: *Advances in Cryptology – ASIACRYPT 2023*. Ed. by Jian Guo and Ron Steinfeld. Singapore: Springer Nature Singapore, 2023.

[103] Kenneth M. Zick, Meeta Srivastav, Wei Zhang, and Matthew French. "Sensing Nanosecond-Scale Voltage Attacks and Natural Transients in FPGAs". In: *ACM FPGA*. 2013.

[104] Daniel Ziener, Florian Baueregger, and Jürgen Teich. "Using the Power Side Channel of FPGAs for Communication". In: *IEEE FCCM*. 2010.

[105] Yazhou Zu, Wei Huang, Indrani Paul, and Vijay Janapa Reddi. "Ti-states: Processor power management in the temperature inversion region". In: *IEEE/ACM MICRO*. 2016.