1. INTRODUCTION

IoT devices are physically integrated in our living space, and hence, deal with sensitive and private data that could be misused to infer privacy violating information. End-to-End secure communication is a necessary measure for the secure IoT. However, it protects merely the communication against unauthorized entities (e.g., eavesdropping, and modification attacks) and leaves the data unprotected on the Cloud. Storing data in such form leaves it vulnerable to breaches [8], caused by hackers and curious administrators [1].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author. Copyright is held by the owner/author(s).

MobiCom’15, Sep 07-11 2015, Paris, France
ACM ACM 978-1-4503-3619-2/15/09.
http://dx.doi.org/10.1145/2789168.2795172.
in mind and is not suitable for IoT application scenarios, mainly because: (i) it employs cryptographic schemes that are prohibitively expensive for constrained devices and (ii) it relies on a trusted proxy, which implies no protection between proxy and the application server. 

Encrypted Query Processing for the IoT. In this paper, we present our IoT data protection system which securely stores encrypted IoT data on the Cloud database, while allowing for efficient database query processing over encrypted data. In our design, we move away from a mere web application communication paradigm. Instead, we design a secure E2E system that stores encrypted data from personal devices on a Cloud database, and where data protection is executed at the personal device (see Figure 2). Thus, we dispense the role of a trusted proxy on the path that has access to all keying material. This allows us to address a stronger threat model, as the keying material does not leave user’s personal devices.

To motivate the use case of our system, let us consider the application scenario of a health monitoring device similar to Fitbit Tracker (provided as built-in app on modern smartphones) which logs heart rate, location, and timestamps. The heart rate measurements can be used to infer sensitive information about a person, such as stress, depression, and heart-related diseases. Hence, heart rate information should be protected from untrusted parties. To still allow certain computations, e.g., average over the protected heart rate data, we utilize additive homomorphic encryption. The location is potentially also sensitive. Thus, we apply deterministic encryption, allowing encrypted queries correlating heart rate with location. Finally, the timestamps could be encrypted with order-preserving encryption, to allow searches in specified time-frames.

2. RELATED WORK

Related work to our approach can be grouped into two main categories.

Privacy-Preserving Cryptography. There has been a significant amount of work on cryptographic schemes [7, 12, 2, 3] that could be utilized in privacy-preserving computation. Gentry’s work [5] depicts a breakthrough, showing a fully homomorphic encryption (FHE) scheme. Since then, his work has been incrementally enhanced up to 6 orders of magnitudes by the research community [6]. Prior to Gentry’s work, the focus was on partial homomorphic encryption, where only one type of computation such as multiplication or addition is supported [3].

Although FHE provides semantic security and supports at the same time arbitrary computations over encrypted data, it is not yet best suited for encrypted query processing. This is due to both its prohibitive cost and the fact that the Cloud must process all existing data in database for queries such as equality check or comparison.

Computation on Encrypted Data. Perrig et al. [4] introduced an efficient search over encrypted text files. This is achieved by deterministically encrypting metadata of files which are themselves encrypted using strong encryption, i.e., probabilistic. Perrig et al.’s efforts paved the way for more advanced systems enabling encrypted query processing [9, 10] (just to mention a few). CryptDB [9], as detailed in §1, realizes the encrypted query processing for web applications by means of a trusted proxy intercepting and modifying the communication before the database. Mylar [10] introduces a multi-key searchable encryption scheme, exemplified by smartphone apps. Mylar protects the content of the documents and the searched words from the untrusted server.

3. SYSTEM ARCHITECTURE

We consider three main parties in our system: IoT devices, the users, and the Cloud. We embrace the trend in the IoT to store collected data in the Cloud. Hence, our system is suitable for applications which store sensor data in the Cloud for client-side processing. Such applications can either upload the data directly to the Cloud (e.g., smart appliances), or by means of a personal gateway such as the smartphone (e.g., wearables). Our system consists of the following components:

a) Crypto Engine for Constrained Devices. We design and implement the cryptographic algorithms required to protect data on IoT devices with attention to efficient performance, small memory footprint, and reasonable energy consumption. Our targeted constrained device platforms are equipped with similar resources as today’s typical IoT devices, such as FitBit (e.g., System-on-Chip ARM Cortex-M family with Bluetooth Low Energy or IEEE 802.15.4).

b) Client-side Library. We exemplify the client’s personal gateway with a smartphone. Our library enables secure storage and interaction with the Cloud. It provides the data protection services to local applications and external devices, such as health monitoring systems. It manages the keying material and performs en-/decryption operations within a reasonable time, so that the user’s experience remains unchanged while interacting with the Cloud.

c) User Defined Functions for Databases. In our system, the database remains unchanged and is only extended with new functionality. This is possible with User Defined Functions (UDFs). While computing over encrypted data, the corresponding UDF is executed. For instance, the UDF for an ordering-related query (e.g., order by, MAX, MIN) is aware of the order-preserving encryption and computes the result without access to the plaintext values.

Encrypted Data Processing. Utilization of encrypted data processing is dependent on the application scenario and security requirements. One of the main challenges in our system is identifying lightweight cryptographic algorithms. Then in contrast to current approaches [9, 10] which rely on full-fledged machines for cryptographic operations, in our system such operations are performed on constrained IoT or/and end-user devices. Considering the set of encryption schemes to support most of queries over encrypted data, additive homomorphic encryption and order-preserving encryption are the most computation-intensive ones. In our system, we explore alternative lightweight cryptographic
algorithms which provide the same level of security, but exhibit significantly lower computation requirements. With the EC-ElGamal and mutable order-preserving encoding (mOPE) [11], we find alternatives for additive homomorphic encryption and order-preserving encryption, respectively. However, before being able to utilize them in our system we have to address two issues: (i) deterministic mapping of plaintext values to elliptic curve points for EC-ElGamal, and (ii) increased communication overhead of mOPE in favor of lower computations. We address the former with an elliptic curve point multiplication \( \text{EC-ElGamal} \) and a known elliptic curve point \( G \). For mapping back, a bounded Elliptic Curve Discrete Logarithmic Problem (ECDLP) by means of the Baby-Step-Giant-Step algorithm is solved. The latter can be optimized with K-ary trees to reduce the interactions-rounds [11].

**Extendibility.** In our design, we show the potential and feasibility of encrypted query processing for the IoT. Although we focus on a subset of encryption schemes that support queries that are widely used in IoT data processing, our design can be extended to include more advanced, however carefully optimized, encrypted data processing schemes.

4. INITIAL RESULTS

We are prototyping our system consisting of three main components: (i) the IoT component is being implemented for OpenMotes\(^1\), (ii) the gateway component for Google Nexus 5, and (iii) the Cloud database component is an extended implementation of CryptDB [9].

Our early results of additive homomorphic encryption by means of EC-ElGamal as compared to the Paillier cryptosystem indicate a performance improvement by 1 order of magnitude. As shown in Figure 3, Paillier’s encryption takes 1.6 to 3.1 s, depending on the plaintext-size. With EC-ElGamal, we have a constant encryption time of 210 ms, including our plaintext to elliptic curve mapping.

Note the decryption process in EC-ElGamal, specifically due to computation of an ECDLP, is computationally more intensive than the encryption. This is not an issue, since the decryption is performed on the more powerful user device. In our early results of EC-ElGamal, decryption on Google Nexus 5 takes about 190 ms and requires 45 Mbyte of memory. We intend to reduce this value further with an improved implementation of the Baby-Step-Giant-Step algorithm.

\(^1\)OpenMote platform, 32-bit ARM Cortex-M3 microcontroller, 32 MHz CPU, 802.15.4 radio: www.openmote.com

5. FUTURE WORK

We introduced a secure system that provides strong communication and data security for privacy-preserving IoT applications. Our system leverages and tailors cryptographic primitives that allow computation on encrypted data without disclosing decryption keys to the Cloud. In this paper, we show the feasibility of our system and discuss initial results. We will show the practically and performance of our system through an extended prototype implementation and thorough evaluation considering both microbenchmarking and system performance. We quantify the overhead of our system in terms of energy, computation, and latency. Moreover, we show the benefits of our system with a case-study discussion.

6. REFERENCES


