Secure Key Management for Multi-Party Computation in MOZAIK

Enzo Marquet*+, Jerico Moeyersons†+, Erik Pohle‡+, Michiel Van Kenhove†+,
Aysajan Abidin‡, Bruno Volckaert†
* *CiTiP, KU Leuven, Belgium,* enzo.marquet@kuleuven.be
† *IDLab-imec, Ghent University, Belgium,* firstname.lastname@ugent.be
‡ *imec-COSIC, KU Leuven, Belgium,* firstname.lastname@esat.kuleuven.be
+ Authors have equal contribution. Author names alphabetical.

***Abstract*—The immense growth of data from the proliferation of Internet of Things (IoT) devices presents opportunities and challenges for privacy engineering. On the one hand, this data can be harnessed for personalized services, cost savings, and environmental benefits. On the other hand, (new) legislation must be complied with and privacy risks arise from collecting and processing of such data. Distributed privacy-preserving analytics offers a promising solution, providing insights while also protecting privacy. However, this approach has new challenges and risks, such as key management and confidentiality. When designing a data marketplace which offers distributed privacy-preserving analytics, the key management comes with different threats, which require a solution adapted to the distributed architecture.**

In this context, the paper presents a comprehensive, end-to-end secure system called MOZAIK for privacy-preserving data collection, analysis, and sharing. The article focuses on the key management aspect of the secure multi-party computation (MPC) component in a distributed privacy-preserving analytics architecture and the specific challenges created by introducing MPC. The proposed solution involves temporary storage of (symmetric) key shares and public-key encryption schemes to ensure secure key management for privacy-preserving computation. Our solution has the potential to be applied in other MPC-based setups, making it a valuable addition to the field of privacy engineering. By addressing key management challenges and risks, MOZAIK enhances data protection while enabling valuable insights from IoT data.

***Index Terms*—Key Management, Privacy-Preserving Computation, Secret Sharing, Secure Multi-Party Computation, GDPR, Data Intermediaries.**

1. Introduction

Recently, versatile Internet of Things (IoT) systems have been widely deployed in daily life, for example, in health care [1] and smart cities [2], [3], which can generate gigabytes of high-definition images, videos, and sensor data every minute. Massive IoT data requires impractically large storage and high-performance computation that an average user or smart object within the IoT hardly supports. Cloud-assisted IoT is popularly applied to leverage a cloud's computation and storage capability for massive IoT data. A cloud is a powerful platform that can provide additional conveniences as a data intermediary.

However, the convenience that the cloud brings to IoT comes at the cost of potential new security risks (e.g., data loss or hacking), which have rarely been the focus of a traditional IoT system. These risks are critical obstacles when building any cloud-assisted IoT system. Moreover, overcoming these security challenges is a big problem due to the versatile functions of cloud-assisted IoT systems and the versatile security requirements of users. Although traditional security mechanisms can protect the security of traditional IoT, they are insufficient to address the risks associated with delegation to the cloud. Normally, a trust-based approach is applied as a solution to those risks. However, trust-based systems cannot provide adequate security. They are susceptible to privacy risks for users since unauthorized user profile creation and user data processing by first or third parties is possible. Likewise, companies face liability and reputational risks e.g. rogue employees or external attackers may misuse their systems. As a data intermediary, companies face additional compliance responsibilities under the Data Governance Act(DGA) as well as the traditional requirements under General Data Protection Regulation (GDPR) for personal data. Moreover, it is impractical in IoT scenarios to apply data anonymization and obfuscation to guarantee privacy for dynamic operations (insertion or deletion). Especially in the context of outsourcing storage and/or computation to the cloud, these applied privacy measures yield no provable security. [4]

The MOZAIK project [5] builds an end-to-end confidential data storage and processing solution for IoT-to-cloud scenarios. The data from an IoT device leaves the user's control encrypted, remains encrypted when stored in the cloud, and is processed without revealing the plaintext to any single entity in the system. Concretely, this architecture supports secure (sensor) data collection and data analysis via privacy-preserving machine learning inference on the collected data.

The MOZAIK architecture comprises three main components to ensure secure, privacy-preserving data collection, analysis, and sharing. The first component is one or more sensors that generate data, which is then encrypted on the device using a suitable secure, possibly lightweight cipher. The second component is Obelisk [6], a secure data store where the encrypted data is stored. The third and most crucial component is a cluster of secure multi-party computation (MPC) engines, which can perform secure distributed calculations on the data without compromising data privacy. To enable processing in MPC, the encrypted

data is first decrypted within the MPC protocol. Since this requires management of the decryption key (shares), key management becomes an important part. We refer to this setup when key management for MPC is mentioned. With these three components, the MOZAIK architecture provides a robust and end-to-end secure solution for handling sensitive data from IoT devices.

The security and privacy issue in big data is an ongoing challenge requiring continued research and development. Particularly on the platform side, security measures have been mainly limited to providing access control on data sets, data analytics jobs, resource scheduling/sharing, or setups [7], [8]. However, there is a pressing need to address key management in a decentralized environment, especially when utilizing MPC. Data privacy is lost if an (internal or external) adversary can access the decryption key. Secure key management is a critical issue in privacy engineering solutions that needs to be addressed securely and reliably.

1.1. Related work

Two ongoing projects, namely, H2020 KRAKEN (BroKeRage and MArKet platform for pErsoNal data)[1] [9], [10] and Agora: A Privacy-Aware Data Marketplace [11] have some relevance to the discussed MOZAIK project. KRAKEN aims at developing a secure and trusted personal data sharing platform with privacy-preserving analytics embedded while Agora aims to create a decentralized data marketplace via smart contracts. Although our goals in the secure data processing aspect are similar to that of KRAKEN and Agora, MOZAIK stands out from those projects in that it takes a holistic approach to secure and scalable IoT data collection, transmission, storage and processing, covering all aspects of a data life cycle. It is in this synergy of the operations at each data cycle point where MOZAIK comes with a unique value proposition.

Similarly, Veeningen et al. [12] also use MPC in three pilot use-cases to perform privacy-preserving computation in the medical domain on collected data. Their work and the two projects include legal assessment and compliance analysis for the GDPR. However, Veeningen et al.'s analysis does not include secure data collection from individual users. Instead the subject's data is collected in an ordinary, non-encrypted way by, e.g., the hospital, a union representative or a family doctor, and then processed such that it remains confidential for the analysis. Note that the secure data collection of MOZAIK provides a stronger, end-to-end confidentiality for the user data without the need to trust the intermediary data collector.

The MyHealthMyData project [13] deploys medical data sharing via distributed ledgers. The proposed architecture focuses on distributed storage and access control. Secure data collection and privacy-preserving use, i.e., computation, on the shared data is not considered since the authorized party obtains the shared data in the clear.

1.2. Contributions

The paper makes three key contributions:

- It analyzes the key management requirements for MPC in a privacy-preserving architecture from a European Union perspective (see Section 2).
- It presents the MOZAIK architecture, which is designed to provide a robust and end-to-end secure solution for handling sensitive data from IoT devices (see Section 3).
- It analyzes and evaluates several key management approaches, highlighting their benefits and trade-offs while ensuring compliance with the identified requirements (see Section 4).

These contributions provide valuable insights and solutions to the challenges of key management in a decentralized environment, making an important contribution to the field of privacy engineering.

2. System model and requirements

In the following sections, we detail the adversarial model and describe the resulting requirements for a secure key management solution in MPC.

2.1. System and adversarial model

The architecture comprises three main components: users with sensors, Obelisk, and a cluster of MPC engines.

We assume that the IoT devices are not compromised and behave honestly. The users behave honestly regarding their own key material but may be dishonest by trying to gain unauthorized access to other data. For this work, we consider Obelisk to be corrupted by a malicious adversary with the goal to access the sensor data. Regarding the MPC parties, we assume the adversarial model of the MPC setup used for the distributed data processing. However, MPC parties may act maliciously when handling key shares. The adversary may corrupt some users, Obelisk and some MPC parties simultaneously to attack, i.e., learn information about other user's sensor data. Note that unlike in federated learning and privacy-preserving machine learning where (distributed) model *training* is the main focus, the setting in this work concerns model *inference* only where the service provider and MPC parties are financially remunerated by the user for their service.

In this paper, we focus on confidentiality and integrity of the sensor data throughout the whole process. We do not consider Denial of Service attacks as they are out of scope of this paper.

2.2. Requirements

A secure key management solution in this setup requires first and foremost that the sensor data cannot be decrypted by unauthorized parties, e.g., other users, Obelisk or corrupted MPC parties, to ensure the confidentiality of data. This entails a requirement for strong keys which are generated in a secure manner in line with data (protection) legislation[2] without increasing overhead significantly and proportionate to the risk. Key backup is out of scope of this paper and left for future work.

However, for the MPC engines to decrypt the encrypted data, they need access to the secret shares of the encryption key. Hence, a secure mechanism is required to enable IoT devices to share their keys with the MPC engines. Further, the user should be empowered to exercise control over the processing of their data i.e., the key

1. https://krakenh2020.eu/

2. As set out in art. 5(1)(f), 25, 32 of the GDPR and art. 12 DGA.

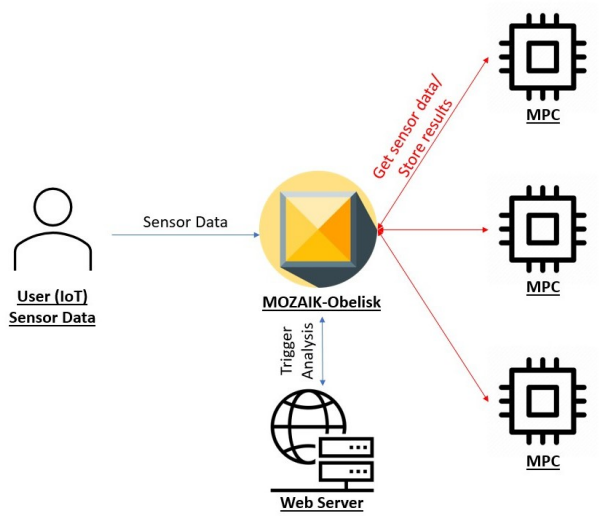

Figure 1. The MOZAIK architecture.

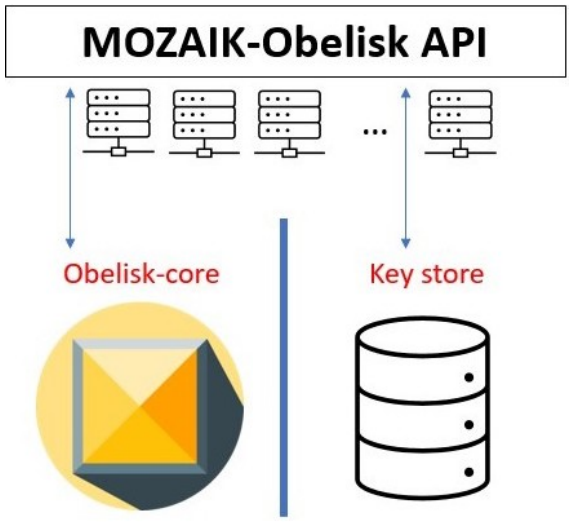

Figure 2. The extended Obelisk version for MOZAIK: MOZAIK-Obelisk.

(share) distribution mechanism requires that only the user can select which parties are involved in the processing and the time when the processing occurs, as well as allowing the user to revoke[3] the encryption keys for further processing.

3. MOZAIK architecture

A general overview of MOZAIK's architecture is illustrated in Figure 1. The main components of MOZAIK are one or more IoT devices under the user's control (full control), MOZAIK-Obelisk – a MOZAIK-specific version of Obelisk –, the cloud-storage entity with an attached web server for client-side user interfaces and a cluster of MPC computing instances for outsourced privacy-preserving computation. An IoT sensor sends its data to Obelisk in a secure way for storage. From all MPC engines participating in the system, the user may select those that match the user's criteria, e.g., availability, cost or their trust in the service, through a web interface. The web interface is accessible by users who pushed IoT data to the platform, or by users who were granted access by another user. A user can only view its own data or the data they received access to. MOZAIK-Obelisk then makes the data available to the MPC engines involved in the calculation request. In Section 3.1 we detail MOZAIK-Obelisk, followed by an overview of the MPC-related functionalities in Section 3.2 and of secure container environments in Section 3.3.

3.1. Obelisk

Obelisk is a cloud-based IoT integration platform that offers interoperability, multi-tenancy and scalability capabilities and is used within the MOZAIK project to capture and store the IoT data [6].

Obelisk allows secured data ingestion, storage, streaming and retrieval through HTTPS REST (REpresentational State Transfer) APIs (Application Programming Interface). With regards to data transport, Obelisk ensures server authentication, data confidentiality and integrity towards its clients (IoT devices, applications, end users) through the use of the Transport Layer Security (TLS) protocol. Obelisk has been designed and implemented by means of an event-based, asynchronous microservices-based software architecture. [14]

In MOZAIK, an extra software layer is built around Obelisk to create a MOZAIK-specific secure version of Obelisk (i.e., MOZAIK-Obelisk), with the focus on privacy-preserving properties. This includes the appropriate encryption key management measures whose requirements were identified in Section 2. Storing (encrypted) keys in the same entity where the data is stored is not advisable, therefore a separate key store entity is introduced, as shown in Figure 2. The key store is physically isolated from Obelisk-core, by the means of node anti-affinity rules, and cannot interact with Obelisk-core or its data because of network policies. The key store can be implemented as a secure database, where an automatic deletion system for keys is preferred, effectively expiring keys after a chosen amount of time.

To provide the user with an easy-to-use experience, an overarching MOZAIK-Obelisk API is created. This newly introduced API provides a stateless, i.e., no data is stored at the API, and access controlled single point of access and is the only publicly exposed service of MOZAIK-Obelisk.

3.2. Secure multi-party computation

Compared to other PETs that offer computation capabilities, MPC has a large communication overhead, but an acceptable computational overhead [15], [16]. Secure multi-party computation protocols are protocols to jointly compute a known function on private input data in distributed systems in a privacy-preserving manner. The protocols guarantee that no subset of adversarial parties that is not in the access structure $\mathcal{A}$ [17] can learn intermediate values or input data from honest parties that are not in the subset, even if the adversarial parties collude and share information. The attacker and trust model in MPC can be

3. The user can retract their consent in line with art. 7(3) GDPR and art. 11(m) DGA.

divided into the semi-honest (or passive security) model and the active security model [16]. In the semi-honest setting, the adversarial parties are allowed to collude and share information but they cannot deviate from the protocol execution. This model is applicable if the MPC parties trust each other not to cheat (e.g., by employing certified software that may be verified by remote attestation) but do not want to handle sensitive cleartext data for, e.g., legal or compliance reasons. In the active security setting, the adversarial parties are allowed to collude and deviate arbitrarily from the protocol, i.e., they may lie about intermediate values or abort the execution preemptively. MPC protocols with active security are usually more expensive in terms of computation and/or communication.

In MOZAIK, MPC is used for the following two steps. First, the MPC parties turn the symmetrically encrypted sensor data into secret-shares of the sensor data via distributed decryption. Then, the sensor data shares are used as input to the inference step in a privacy-preserving machine learning model. Let n be the number of parties, d denote the cleartext sensor data and $[\![d]\!]$ denote the secret shared version. We use secret-sharing (see e.g., [18]–[20]) as a black-box with the following algorithms.

Share The share algorithm secret-shares a data item d into shares $[\![d]\!]_1, \ldots, [\![d]\!]_n$ according to a specific access structure $\mathcal{A}$.

Recon The reconstruction algorithm reconstructs the plaintext data item that is secret-shared if the set of shares $[\![d]\!]_{a_1}, \ldots [\![d]\!]_{a_j}$ is in the access structure $\{a_1, \ldots, a_j\} \in \mathcal{A}$. Thus it returns d.

Therefore, for all d, and all subsets $a \in \mathcal{A}$,

$$\mathsf{Recon}([\![d]\!]_{a_1}, \ldots, [\![d]\!]_{a_j}) = d \ ,$$

with $[\![d]\!]_1, \ldots, [\![d]\!]_n \leftarrow \mathsf{Share}(d)$. And for all other subsets $a' \notin \mathcal{A}$, it is not feasible to recover d.

Further, let k denote the symmetric key. We use a symmetric authenticated encryption with associated data algorithm (AEAD) [21] as a black-box. The concrete algorithm can be selected to suit the targeted IoT environment, e.g., from the NIST Lightweight Cryptography Competition[4]. We omit the nonce and associated data part from the high-level description for conciseness. An AEAD scheme consists of two algorithms.

Enc The encryption algorithm encrypts the sensor data d under key k and produces a ciphertext c and tag t.

DecAndVerify The decryption algorithm decrypts the ciphertext c to the plaintext d and verifies the authenticity of d using the provided tag t using k. It returns d or $\perp$ if the ciphertext or tag have been modified.

For all d and k, we have

$$\mathsf{DecAndVerify}(k, \mathsf{Enc}(k, d)) = d \, .$$

In the MOZAIK architecture, we propose the following algorithms to ensure end-to-end confidentiality of the user's sensor data.

KeyGen The key generation algorithm is run by the user and returns the symmetric key k and an appropriate number of shares $[\![k]\!]_1, \ldots, [\![k]\!]_n \leftarrow \mathsf{Share}(k)$ for n MPC parties.

4. https://csrc.nist.gov/Projects/Lightweight-Cryptography

DistDec The distributed decryption algorithm is run by the n MPC parties with c and t as *public* input and the key shares $[\![k]\!]_i$ as *private* input. The algorithm verifies the tag t and decrypts c into shares of the plaintext $[\![d]\!]_1, \ldots, [\![d]\!]_n$. It returns $[\![d]\!]_1, \ldots, [\![d]\!]_n$ or $\perp$ if the tag did not verify. We instantiate DistDec by running $[\![d]\!] \leftarrow \mathsf{DecAndVerify}([\![k]\!], c, t)$ in MPC[5].

Infer The privacy-preserving inference algorithm computes the data analysis on the secret-shared input data $[\![d]\!]_1, \ldots, [\![d]\!]_n$ and is run by the MPC parties. It produces the inference result $[\![x]\!]_1, \ldots, [\![x]\!]_n$. For example, this can be a forward pass on a trained neural network computed in MPC.

Correctness is defined as

$$\mathsf{Recon}(\mathsf{Infer}(\mathsf{DistDec}(\mathsf{Enc}(k, d), [\![k]\!]_1, \ldots, [\![k]\!]_n))) = \mathsf{Infer}^*(d) \ ,$$

where $k, [\![k]\!]_1, \ldots, [\![k]\!]_n \leftarrow \mathsf{KeyGen}()$ and $\mathsf{Infer}^*(d)$ is the cleartext inference.

KeyGen is run by the user who sets the obtained symmetric key k to be the encryption key in the IoT device(s) and communicates the key share $[\![k]\!]_i$ to MPC party i. The IoT device collects the data d, encrypts it using the symmetric key k and sends ciphertext c and tag t to MOZAIK-Obelisk for storage. The MPC computation parties retrieve c and t from MOZAIK-Obelisk and first run DistDec to obtain shares $[\![d]\!]$. Then, the MPC parties run Infer to compute the domain-specific analysis on the data and obtain shares of the result $[\![x]\!]$. The shares are communicated back to the user who recombines them to obtain the result.

Assuming that the secret sharing scheme $(\mathsf{Share}, \mathsf{Recon})$ and the AEAD mode $(\mathsf{Enc}, \mathsf{DecAndVerify})$ are secure, neither the adversarial MPC parties nor an external observer can learn any information on the IoT sensor data d or the inference result x. However, the process relies on secure key management of k and the shares $[\![k]\!]_i$.

3.3. Secure container environment

The proposed MOZAIK architecture consists of a multitude of microservices. Microservice-based architectures promote application development as a set of distributed small and independent services each one of those focusing on a specific task. An inherent property of microservice-based architectures is the large amount of services that need to be managed in a production environment. To maintain a fully operational state of the application, each service runs in a separate container and the containers are managed by a container orchestration platform, e.g., Kubernetes [22]. Containers bundle all the necessary dependencies and software of an application in a standardized and portable format. This standardized format is laid out by the Open Container Initiative (OCI). In general, the OCI creates open industry standards around container formats and runtimes [23]. Containers offer relatively fast startup times with a smaller computational overhead compared to virtual machines (VM) and allow large applications to scale horizontally in a seamless manner. A container runtime manages the container life cycle

5. In other words, DistDec can be viewed as a thresholdized version of DecAndVerify.

and provides software-based container isolation. Software-based container isolation exposes the risk of container runtime escaping vulnerabilities and system privilege escalation [24]. A recent example of such a vulnerability is CVE-2022-0811 [25].

If an attacker tries to gain unauthorized access to the MOZAIK application or its data, the overarching MOZAIK-Obelisk API can be considered as one of the initial targets, as it is the only publicly reachable access point that can directly give access to the stored data. When the attacker successfully exploits a vulnerability at the API and succeeds to escape the container runtime, the attacker may be able to gain access to other (sensitive) components of the system and access the (encrypted) data and (encrypted) keys. To mitigate these container runtime security vulnerability risks, sensitive components of the MOZAIK architecture will run in a secure container runtime, inherently making the system more secure. A secure container runtime encapsulates the container in a lightweight virtual machine (microVM) or a sandboxed environment and enforces a strict container-host isolation. A VM applies hardware virtualization to create an isolated virtual environment for each VM instance. As a result of such hardware-based isolation, the attacker would be stuck in the isolated environment and cannot access other sensitive components of the system. Using any form of VM automatically entails a certain computational overhead, hence a trade-off between execution performance and added security needs to be made to decide which components should run in a secure container.

Next to using, or in combination with, a secure container runtime, Trusted Execution Environments (TEE) can be used to further enhance the security of the system. A TEE is a tamper-resistant processing environment that runs on a separation kernel. It guarantees the authenticity of the executed code, the integrity of the runtime states (e.g., CPU registers, memory and sensitive I/O), and the confidentiality of its code, data and runtime states stored on a persistent memory. [26]

The guarantees that TEE provides would allow MOZAIK to be more secure and allows MOZAIK to be deployed on third-party multi-tenant cloud service providers (CSP), without the risk that a malicious tenant or CSP can access or alter the data or code. TEE can be used in multiple parts of the MOZAIK architecture, e.g., running the key store in a TEE could provide an extra layer of security and isolation for secure key storage.

4. Key management within MOZAIK

The MOZAIK architecture presented in Section 3 does not cover key management within MOZAIK. This section presents a key management solution that meets the requirements from Section 2. Our key management solution only requires temporary storage of encrypted key shares in the key store of MOZAIK and relies on public-key encryption. Similar to the secret sharing scheme and the AEAD scheme, we use a public-key encryption scheme [27] as a black-box.

PK.KeyGen The public-key scheme key generation outputs a private key k_s and a public key k_p.

PK.Enc Given the public key k_p and message m, the encryption algorithm outputs a ciphertext c.

PK.Dec Given the private key k_s and a ciphertext c, the decryption algorithm outputs the message m.

Figure 3 shows the setup, data collection, computation and result retrieval involving all entities in the MOZAIK architecture. When a MPC computation party P_i joins MOZAIK, it runs $\mathsf{PK.KeyGen}()$ and obtains k_{s_i} and k_{p_i}. It then advertises its public key k_{p_i} to all users in a way preventing impersonation attacks on k_{p_i}, e.g., via a public-key infrastructure. P_i may join anytime but is only available to computation that starts after P_i joined, i.e., we do not consider parties joining the protocol mid-computation like in the YOSO model [28].

Before a computation begins, the user encrypts the key share for P_i using P_i's public key k_{p_i}, i.e., $c_{[\![k]\!]_i} \leftarrow \mathsf{PK.Enc}(k_{p_i}, [\![k]\!]_i)$ and stores all encrypted key shares $c_{[\![k]\!]_1}, \ldots, c_{[\![k]\!]_n}$ in the key store. For added security, the encrypted key shares are only stored temporarily in the key store and thus are removed after their expiry date.

When the MPC parties start the computation, they fetch their respective encrypted key shares $c_{[\![k]\!]_i}$ from the key store and the encrypted user data c from MOZAIK-Obelisk. During the computation, the key shares are decrypted using p_{s_i} and are temporarily stored at the MPC parties. The decrypted $[\![k]\!]_i$ is used as private input to the MPC protocol computing $\mathsf{DistDec}$ and Infer. When the computation has finished, each party obtains a share of the result $[\![x]\!]$. To communicate the result x back to the user securely, two approaches come to mind. First, the MPC parties may run a distributed encryption protocol, $\mathsf{DistEnc}$, to encrypt x and produce a ciphertext c_x. This ciphertext will be stored in Obelisk and the MPC parties erase the temporarily stored key shares. The user retrieves the encrypted result from Obelisk and uses their key k to decrypt the result. Another approach is that each MPC party P_i encrypts their share $[\![x]\!]_i$ using the user's public key, producing ciphertexts $c_{[\![x]\!]_i}$ and storing these ciphertexts in Obelisk. The user fetches the encrypted shares of the result, decrypts them and then reconstructs the inference result x by calling Recon. While the former approach is more storage efficient and does not require the MPC parties to know the public key of the user, it may incur more costly MPC computation compared to the latter approach. However, it is possible that for short inference results, e.g., where x is only a few bits, distributed encryption leads to better overall performance. We leave a detailed study of which approach is more suitable to future work with an implementation of a concrete use-case.

Intuitively, the encrypted key share for MPC party i can only be decrypted by that party and not by others or Obelisk. Similarly, the key store cannot change the encrypted key share without being detected by the party that attempts to decrypt it later. In addition, no further data analysis on the plaintext data can be computed since the honest MPC parties have erased their key share and refuse to commence a new MPC protocol run.

4.1. Trade-offs and other approaches

The presented approach where key shares are temporarily stored at MOZAIK-Obelisk to be relayed to the MPC engines for temporary use is not the only key share distribution that comes to mind. However, we note multiple benefits of our presented approach.

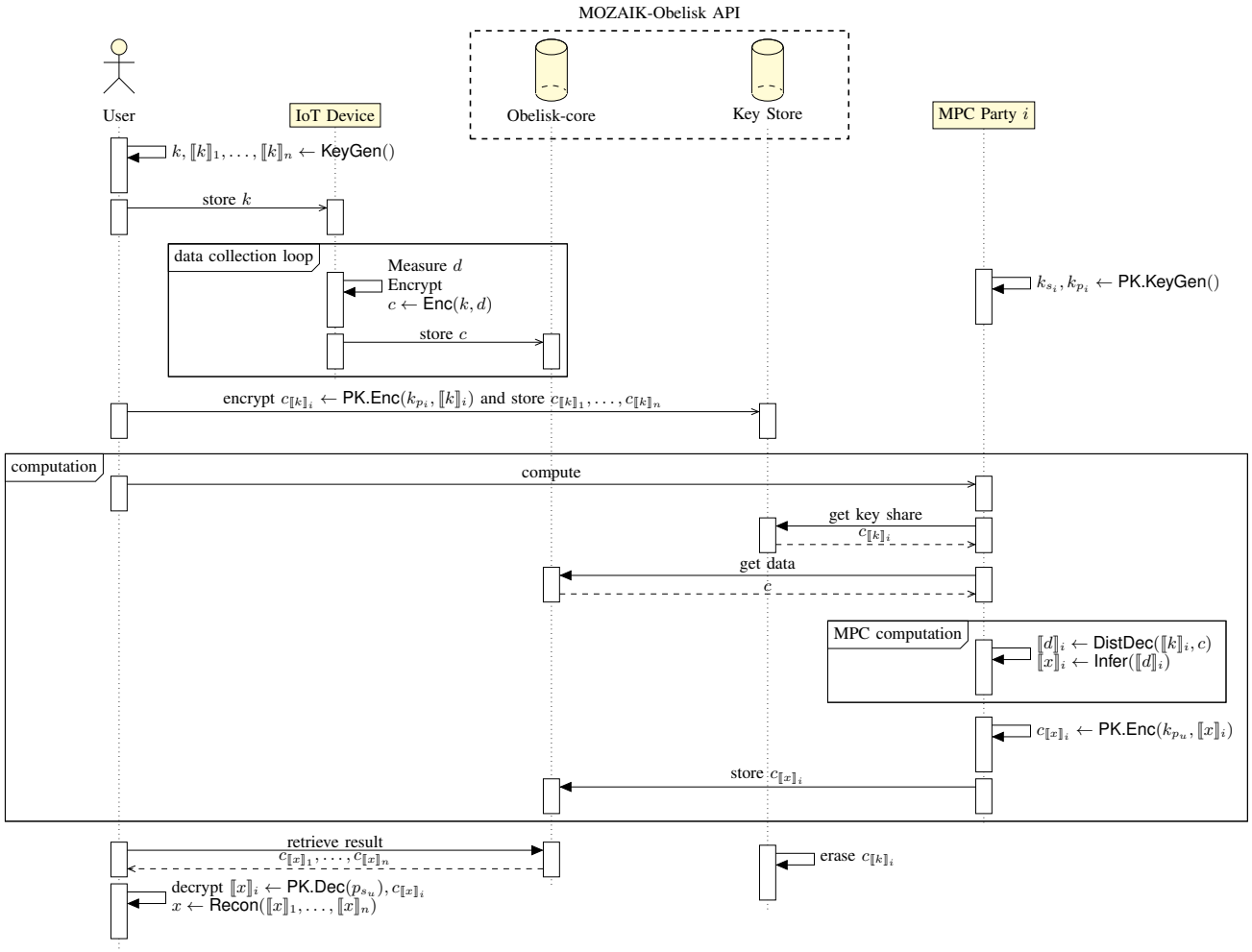

Figure 3. Overview of the setup, data collection, computation and result retrieval in the MOZAIK architecture. We omit calls to the MOZAIK-Obelisk API for clarity and directly show the behaviour of the sub-components (in the dashed box).

First, the MPC parties remain pure computing parties in the architecture. This aligns well with the MPC-as-a-service model of trust [29]–[31], where computation providers offer to take part in MPC computation as a means to outsource computation. Since no data needs to be persisted between two calls of DistDec or Infer, the user is flexible to switch MPC-as-a-service providers easily without overhead. Second, by storing the key shares encrypted in the key store, the user doesn't need to be online to start DistDec since the assigned MPC parties can fetch their shares. Note that the data collection by the IoT device can already start before the user picked the MPC parties, increasing time and operational flexibility. Further, no direct interaction between user and the MPC parties is required, so similarly, the user does not need to be online when Infer returns a result. Still, only those MPC parties receive a share that were picked by the user. The key store cannot chose a different (e.g., less trustworthy) set of MPC parties. The time of processing is in full control of the user since processing is only possible once the key shares are distributed. Third, all information that may be used to decrypt the sensor data, i.e., the encrypted key shares, remain with third-parties, the key store and the assigned MPC parties, only during the time of processing the data. Fourth, if the processed personal data is of a highly sensitive nature, the architecture allows the introduction of more PETs (e.g., obfuscating or de-identifying) on the personal data before it is shared to protect it even further. This ensures GDPR and DGA compliance and flexibility for data marketplaces of any type of data. The marketplace can document this approach as an adequate measure to justify the sharing of (sensitive personal) data in a decentralized setting.

The benefits come at a trade-off. To increase the flexibility of the user and remove the requirement of the user being online, the encrypted sensor data as well as the encrypted key shares are both stored in the centralized entities, Obelisk and key store, which may give rise to single point of failure issues.

Other variants of the key distribution approach include the user sending the key shares directly over a secure channel to each MPC party directly, i.e., not involving other architectural components. This clearly requires the user to be online to start Infer. In another variant, the MPC parties store their key share persistently such that the user only needs to share the key once and subsequent computation tasks do not require the user to be online. However, as noted above, this reduces much flexibility in the MPC-as-a-service model as well as it may introduce additional legal requirements.

4.2. Implementation considerations

Previous sections sketch the solution using the functionalities and cryptographic schemes in a black-box manner. In the following, we briefly describe concrete instantiations and protocols that may be used for deployment.

MOZAIK currently supports multiple AEAD scheme instantiations since the scheme is being computed both on the IoT device as well as in MPC. We support SKINNY [32] and GIFT-based [33] lightweight AEAD schemes from the NIST LWC competition, which are designed to be efficient in resource-constraint devices and are thus a choice favouring the IoT side, i.e., Enc. However, since these algorithms perform orders of magnitude worse than so called MPC-friendly ciphers, we support additionally MiMC [34] and derived AEAD schemes [35]. These constructions allow the evaluation of DistDec to be much more efficient. The components of MiMC, e.g., 256-bit prime field arithmetic, is quite costly on microcontrollers and embedded devices. We found AES-based AEAD schemes, e.g., AES-GCM [36] and AES-GCM-SIV [37] to be a suitable compromise, since AES is much more MPC-friendly than lightweight symmetric primitives due to its largely algebraic structure [38]. Dedicated distributed symmetric encryption (e.g., DiSE [39] and follow-up works) may also be used, however current constructions and security notions do not fit the single encryption entity that we require.

For the public-key encryption, a deployment requires fewer considerations. Any IND-CCA(2) secure scheme may be used, such as standardized constructions, e.g., [40]. For post-quantum security, a suitable candidate may also be chosen after it has been finalized.

To implement the MPC-based algorithms, DistDec and Infer, we first note that the two algorithms do not need to be executed with the same MPC protocol. The shares of the sensor data $[\![d]\!]$ that are output from DistDec in one protocol can be converted into input shares of the protocol suitable for Infer [41]. The protocol for DistDec depends mainly on the choice of the AEAD scheme above. A lightweight scheme will likely perform better in Boolean, constant-round protocols, such as garbled circuits [42], whereas for more than two parties or MPC-friendly ciphers, secret-sharing based protocols [43], [44] are more suitable. Dedicated MPC protocols for AES and SPN-based primitives exist as well, e.g., [45], [46]. Machine-learning model inference has been studied separately, often with dedicated protocols for certain types of models [47], [48].

4.3. Further discussion and future work

Due to space constraints, we cannot formally detail all security and privacy notions and considerations that exist in the architecture, and how they are addressed with various cryptographic, legal and operational means. While sensor data confidentiality and authenticity is guaranteed via the aforementioned use of AEAD encryption, MPC, secret-sharing and secure containers, several other, sometimes subtle, considerations have to be kept in mind.

The current measures tolerate static corruption, i.e., where the set of corrupted components in the architecture does not change during the deployment lifetime. We leave an adaptive corruption where the corrupted parties change, e.g., in between executions of DistDec and Infer, or in between runs of the whole process, for future work. In this setting, information on key shares of previously corrupted parties may allow the adversary to reconstruct the key in certain cases. This may be addressed by re-sharing randomness and proxy re-encrypting the sensor data from time to time. Moreover, several privacy considerations exist due to linkability. Obelisk and key store can build usage and access patterns of (the user's) sensor data and the key shares for the MPC engines. This leads to knowledge regarding the type, frequency and data size of the analysis that is chosen by the user. Mitigations, such as anonymous credentials, are beyond the scope of this paper and left for future work.

Another open problem, from a legal point of view, is classifying the result of the computation [49]. It is unclear whether the result of the computation is personal data at all. If it is not personal data, the GDPR does not apply to the result. If it is personal data, the whole process can be seen as technical and organisation measures taken to protect the data of the data subjects. Further work in this direction is highly relevant for the use of MPC and other PETs in privacy engineering.

5. Conclusion

In conclusion, the convenience of the cloud for IoT devices is enormous. However, data collection and sharing in cloud-assisted IoT systems also introduces new security, privacy and compliance challenges. To address these challenges, we propose MOZAIK, an end-to-end secure and privacy-friendly data sharing architecture, that processes symmetrically encrypted IoT data in a privacy-preserving manner via distributed decryption using secure multi-party computation. In this setting, management of the symmetric encryption key is crucial for the overall security and privacy. In this work, we presented a key management approach of said symmetric key which included temporary storage of the symmetric keys and relied on public-key encryption without significant overhead increase. The paper identified both security and privacy threats for our setting and presented a solution based on the requirements drawn from these threats and overall functionality to ensure that MOZAIK maintains the highest security and privacy standards as well as functionality.

Acknowledgements

This work is supported by the Flemish Government through FWO SBO project MOZAIK S003321N.

References

[1] S. Selvaraj and S. Sundaravaradhan, "Challenges and opportunities in IoT healthcare systems: a systematic review," *SN Applied Sciences*, vol. 2, no. 1, p. 139, 2020.

[2] S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, and P. Demeester, "City of things: An integrated and multi-technology testbed for IoT smart city experiments," in *2016 IEEE international smart cities conference (ISC2)*, pp. 1–8, IEEE, 2016.

[3] J. Santos, T. Vanhove, M. Sebrechts, T. Dupont, W. Kerckhove, B. Braem, G. Van Seghbroeck, T. Wauters, P. Leroux, S. Latre, *et al.*, "City of things: Enabling resource provisioning in smart cities," *IEEE Comm. Magazine*, vol. 56, no. 7, pp. 177–183, 2018.

[4] W. Wang, P. Xu, and L. T. Yang, "Secure data collection, storage and access in cloud-assisted IoT," *IEEE cloud computing*, vol. 5, no. 4, pp. 77–88, 2018.

[5] "Mozaik: Scalable and secure data sharing." https://www.esat.kuleuven.be/cosic/projects/mozaik/.

[6] "Obelisk." https://obelisk.ilabt.imec.be/catalog/home.

[7] S. Flesca, S. Greco, E. Masciari, and D. Saccà, *A comprehensive guide through the italian database research over the last 25 years*, vol. 31. Springer, 2018.

[8] A. Kourid and S. Chikhi, "A comparative study of recent advances in big data for security and privacy," *Networking Communication and Data Knowledge Engineering: Volume 2*, pp. 249–259, 2018.

[9] K. Koch, S. Krenn, D. Pellegrino, and S. Ramacher, "Privacy-Preserving Analytics for Data Markets Using MPC," in *Privacy and Identity Management*, (Cham), pp. 226–246, Springer, 2021.

[10] S. Gabrielli, S. Krenn, D. Pellegrino, J. C. Pérez Baún, P. Pérez Berganza, S. Ramacher, and W. Vandevelde, *KRAKEN: A Secure, Trusted, Regulatory-Compliant, and Privacy-Preserving Data Sharing Platform*, pp. 107–130. Cham: Springer, 2022.

[11] V. Koutsos, D. Papadopoulos, D. Chatzopoulos, S. Tarkoma, and P. Hui, "Agora: A Privacy-aware Data Marketplace," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1211–1212, 2020.

[12] M. Veeningen, S. Chatterjea, A. Z. Horváth, G. Spindler, E. Boersma, P. van der Spek, O. van der Galiën, J. Gutteling, W. Kraaij, and T. Veugen, "Enabling Analytics on Sensitive Medical Data with Secure Multi-Party Computation," in *Studies in health technology and informatics*, vol. 247, pp. 76–80, 2018.

[13] E. Morley-Fletcher, "MHMD: My Health, My Data," in *EDBT/ICDT Workshops*, 2017.

[14] V. Bracke, M. Sebrechts, B. Moons, J. Hoebeke, F. De Turck, and B. Volckaert, "Design and evaluation of a scalable Internet of Things backend for smart ports," *Software: Practice and Experience*, vol. 51, no. 7, pp. 1557–1579, 2021.

[15] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: Secure multi-party computation meets machine learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4961–4973, 2021.

[16] D. Evans, V. Kolesnikov, M. Rosulek, *et al.*, "A pragmatic introduction to secure multi-party computation," *Foundations and Trends® in Privacy and Security*, vol. 2, no. 2-3, pp. 70–246, 2018.

[17] A. Beimel and B. Chor, "Universally Ideal Secret Sharing Schemes," in *Advances in Cryptology — CRYPTO' 92* (E. F. Brickell, ed.), pp. 183–195, Springer, 1993.

[18] G. R. Blakley, "Safeguarding cryptographic keys," in *MARK*, pp. 313–318, 1979.

[19] A. Shamir, "How to Share a Secret," *Commun. ACM*, vol. 22, p. 612–613, nov 1979.

[20] A. Beimel, "Secret-Sharing Schemes: A Survey," in *Proceedings of the Third International Conference on Coding and Cryptology*, IWCC'11, p. 11–46, Springer, 2011.

[21] P. Rogaway, "Authenticated-Encryption with Associated-Data," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, p. 98–107, ACM, 2002.

[22] "Kubernetes." https://kubernetes.io/.

[23] "About the Open Container Initiative - Open Container Initiative." https://opencontainers.org/about/overview/.

[24] M. Souppaya, J. Morello, and K. Scarfone, "NIST Special Publication 800-190 Application Container Security Guide," 2017.

[25] "NVD - CVE-2022-0811." https://nvd.nist.gov/vuln/detail/CVE-2022-0811.

[26] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted Execution Environment: What It is, and What It is Not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, pp. 57–64, 2015.

[27] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes," in *CRYPTO '98*, LNCS, p. 26–45, Springer, 1998.

[28] C. Gentry, S. Halevi, H. Krawczyk, B. Magri, J. B. Nielsen, T. Rabin, and S. Yakoubov, "YOSO: You Only Speak Once," in *Advances in Cryptology – CRYPTO 2021* (T. Malkin and C. Peikert, eds.), (Cham), pp. 64–93, Springer International Publishing, 2021.

[29] A. Barak, M. Hirt, L. Koskas, and Y. Lindell, "An End-to-End System for Large Scale P2P MPC-as-a-Service and Low-Bandwidth MPC for Weak Participants," in *CCS '18*, p. 695–712, ACM, 2018.

[30] N. P. Smart and T. Tanguy, "TaaS: Commodity MPC via Triples-as-a-Service," in *CCSW'19*, p. 105–116, ACM, 2019.

[31] S. Kanjalkar, Y. Zhang, S. Gandlur, and A. Miller, "Publicly Auditable MPC-as-a-Service with succinct verification and universal setup," in *EuroS&PW*, pp. 386–411, 2021.

[32] C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim, "The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS," in *CRYPTO 2016*, pp. 123–153, Springer, 2016.

[33] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "Gift: a small present," in *CHES*, pp. 321–345, Springer, 2017.

[34] M. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, "MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity," in *ASIACRYPT 2016*, pp. 191–219, Springer, 2016.

[35] D. Rotaru, N. P. Smart, and M. Stam, "Modes of Operation Suitable for Computing on Encrypted Data," *IACR Transactions on Symmetric Cryptology*, pp. 294–324, 2017.

[36] M. J. Dworkin, *Sp 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. National Institute of Standards & Technology, 2007.

[37] S. Gueron and Y. Lindell, "GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte," in *CCS '15*, p. 109–119, ACM, 2015.

[38] I. Damgård, M. Keller, E. Larraia, C. Miles, and N. Smart, "Implementing AES via an Actively/Covertly Secure Dishonest-Majority MPC Protocol," in *International Conference on Security and Cryptography for Networks*, pp. 241–263, Springer, 2012.

[39] S. Agrawal, P. Mohassel, P. Mukherjee, and P. Rindal, "DiSE: distributed symmetric-key encryption," in *CCS '18*, pp. 1993–2010, 2018.

[40] ISO/IEC 18033-2:2006, "Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers." ISO/IEC JTC 1/SC 27, 2006.

[41] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2.0: Improved Mixed-Protocol secure Two-Party computation," in *USENIX Security 21*, pp. 2165–2182, USENIX Association, Aug. 2021.

[42] S. Zahur, M. Rosulek, and D. Evans, "Two Halves Make a Whole," in *EUROCRYPT 2015*, pp. 220–250, Springer, 2015.

[43] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical Covertly Secure MPC for Dishonest Majority – or: Breaking the SPDZ Limits," in *European Symposium on Research in Computer Security*, pp. 1–18, Springer, 2013.

[44] V. Goyal, H. Li, R. Ostrovsky, A. Polychroniadou, and Y. Song, "ATLAS: Efficient and Scalable MPC in the Honest Majority Setting," in *CRYPTO 2021*, pp. 244–274, Springer, 2021.

[45] M. Keller, E. Orsini, D. Rotaru, P. Scholl, E. Soria-Vazquez, and S. Vivek, "Faster Secure Multi-party Computation of AES and DES Using Lookup Tables," in *ACNS*, Springer, 2017.

[46] E. Pohle, A. Abidin, and B. Preneel, "Poster: Fast Evaluation of S-boxes in MPC." NDSS 2022, 2022.

[47] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "FALCON: Honest-Majority Maliciously Secure Framework for Private Deep Learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, pp. 188–208, 2020.

[48] N. Koti, M. Pancholi, A. Patra, and A. Suresh, "SWIFT: Super-fast and Robust Privacy-Preserving Machine Learning.," in *USENIX Security*, pp. 2651–2668, 2021.

[49] L. Helminger and C. Rechberger, "Multi-Party Computation in the GDPR," in *Privacy Symposium 2022* (S. Schiffner, S. Ziegler, and A. Quesada Rodriguez, eds.), (Cham), pp. 21–39, Springer, 2022.