Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

A-SIT

Seidlgasse 22 / 9, 1030 Wien
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

Inffeldgasse 16a, 8010 Graz
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461
ZVR: 948166612
UID: ATU60778947

CHALLENGES OF ENCRYPTION: RECOVERY AND COMPUTATION

Version 1.0 vom 30.09.2021

Felix Hörandner – felix.hoerandner@iaik.tugraz.at

Encryption represents a fundamental tool to protect the confidentiality of sensitive data, such as identity attributes, medical data, and digital twins of physical objects. By employing encryption, sensitive data can be securely stored in not-fully-trusted cloud systems, thereby enabling various use cases.

However, encryption also introduces challenges: If decryption keys are lost (e.g., because the device holding them breaks), users face the threat of losing access to their encrypted data, as they are no longer able to decrypt these data. Furthermore, while encryption prevents the cloud from learning the users' data, the cloud is also greatly hindered in processing the encrypted data even if the user would benefit from the results.

We investigate the two challenges caused by encryption and propose solutions in the context of a digital twin system. A digital twin is the continuously updated digital representation of a physical object that is maintained in the cloud. As digital twin data can be highly sensitive, we have previously proposed a security architecture for digital twins that introduces an encryption layer to protect the data. For such an encrypted digital twin system, recovery and processing are also core requirements. Firstly, we make use of the flexibility of proxy re-encryption to change access permissions dynamically (e.g., as trust relationships change) and recover digital twin data on a replacement device after the original device is no longer functional. Furthermore, we integrate secure multi-party computation to process protected data according to the users' permissions without revealing the inputs or outputs to the processing nodes. An evaluation highlights the feasibility and practicability of our approach based on an example use case, namely privacy-preserving contact tracing.

© A-SIT, Partner für Sicherheitsfragen

Table of Contents

1. Introduction

Encryption serves as a key enabler to outsource sensitive data to not fully trusted cloud services. By encrypting the users' sensitive data with private key material before uploading them to the cloud, the cloud operator or potential attackers cannot learn the users' plain data. Users can still download and decrypt their data or even share them with others by sharing relevant decryption keys.

Challenges. Unfortunately, such protection mechanisms also introduce new challenges: processing of protected data and recovery after key loss. Firstly, while encryption prevents the cloud from learning the encrypted data, the same property also presents a significant obstacle for the cloud to process this data. Secondly, if devices break and their decryption keys are lost, a strategy is needed to replace these devices and recover their data.

Context: Digital Twins. We investigate the possibilities and challenges of encryption in the context of digital twins [Barricelli, Fuller]. A digital twin is the digital representation of a physical object. Changes in the physical object's characteristics are continuously synchronized at the digital twin in the cloud, while interaction with the digital twin can trigger reactions that influence the physical object. Digital twins can, for example, be established for machinery in a production line, vehicles in transportation, or also humans in governmental or medical use cases. The accumulated data of digital twins enables powerful computations and simulations. For example, Qi and Tao [Qi] apply digital twins to monitor a manufacturing process, such that failures are detected, and the system can compute an optimized solution to address the problem. Kraft [Kraft] uses digital twins for aircraft components, where the collected sensor data enables simulations and prediction to reduce both the development and maintenance effort.

Encryption for Digital Twins. Digital twin systems may collect sensitive data, e.g., health data to build a digital twin of a human or company secrets such as the layout of a manufacturing plant. As the digital twin data is stored in a not-fully-trusted cloud service, they need to be protected. In previous work [Hörandner-1], we have applied encryption to ensure the confidentiality of the digital twin data while preserving the functionality and benefits of digital twin systems. Unfortunately, the challenges of encryption also apply to protected digital twin systems.

Extending an Encrypted Digital Twin System. This work aims to address two challenges caused by adding an encryption layer in the context of digital twins: (1) computation on protected data and (2) recovery/maintenance as devices break and trust changes. In this work, we build and extend upon our security architecture for digital twins [Hörandner-1]. The results of this work served as one contribution to our accepted research paper at SECRYPT 2021 [Hörandner-2].

Approach to Recovery and Maintenance. Digital twin systems may consist of multiple stakeholders with various devices that maintain different cryptographic keys. If such devices break and their keys are lost, the owners still need to be able to access their digital twin data and replace broken devices to recover the functionality of the overall system. Our security architecture for digital twins [Hörandner-1] employs a flexible cryptographic mechanism to protect the digital twin's data, namely key-policy conditional proxy re-encryption [Zhao]. We are able to use proxy re-encryption to change sharing permissions and recover digital twin data on replacement devices to react to changes in the overall system as devices break or trust relationships to data receivers change.

Approach to Privacy-Preserving Processing. Encrypted data of digital twins needs to be processed to achieve the full benefits promised by a digital twin system. Therefore, we integrate multi-party computation [Bogdanov, Yao] to enable processing on the protected digital twin data. The digital twin data items are split into shares, which are stored in encrypted form in the cloud service. To perform computations on these encrypted data items, their shares are distributed to a set of processing nodes via key-policy conditional proxy re-encryption. These processing nodes engage in a multi-party computation protocol with each other to compute a function on (the shares of) the input data items without actually learning the plain inputs or their computed output. Only a user-specified receiver learns the result. Our evaluation underlines the feasibility and performance of this approach. We evaluate the example use case of privacy-preserving contact tracing, where the paths of multiple users are compared to one (infected) reference user. The results show a linear growth in execution time relative to the number of users.

Outline: Initially, Section 2 explains the concept of digital twins. Section 3 summarizes how the data of digital twins can be protected via encryption, as presented in [Hörandner-1]. Section 4 tackles the challenge of recovery and maintenance in a system with encrypted data. Section 5 addresses our second challenge, privacy-preserving computation on protected data, and gives a performance evaluation based on our example use case of privacy-preserving contact tracing. Finally, Section 6 concludes this work.

2. Digital Twins

Over recent years, a new paradigm has gained popularity in the Internet of Things (IoT): the so-called *digital twins*. Applying the concept of digital twins enables to give more structure to an IoT system by defining data storage and communication patterns. This section first explains the general parts of a digital twin system. Then, this section describes functionality offered by a digital twin system to various stakeholders

A **digital twin** (c.f. Figure 1) is the digital representation of a **physical object**. Changes in the physical object's characteristics are continuously synchronized at the digital twin, while interaction with the digital twin can trigger reactions that influence the physical object. The **cloud** (or a service in the cloud) serves as a central location to collect and share the digital twins' data. **Devices** form the technical link between physical objects and digital twins. They monitor the physical object and transmit changes to the digital twin in the cloud but also interact with the physical object upon changes applied to the digital twin in the cloud. Digital twins can, for example, be established for machinery in a production line, vehicles in transportation, or also humans in governmental or medical use cases. The accumulated data of digital twins enables powerful computations and simulations.

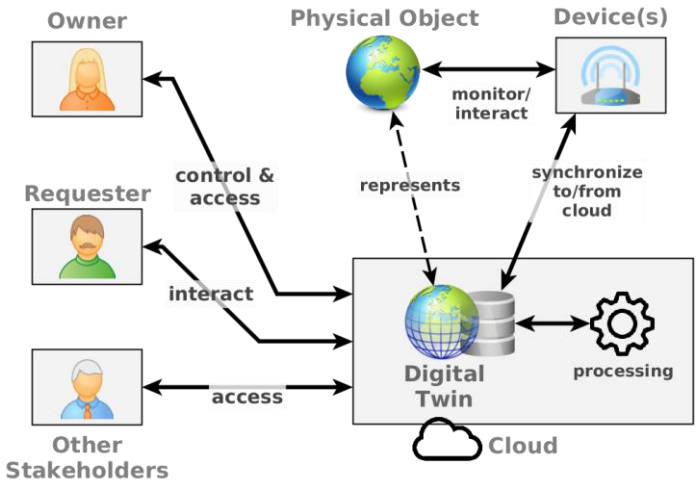

Figure 1: Digital Twins

The following functionalities and benefits become possible by creating and maintaining an always-synchronized digital twin in the cloud.

Control & Access. The owners and other authorized users can interface with the digital twin in the cloud to access an up-to-date description of the physical objects in the real world. With write access, they are also able to control and influence the physical objects, e.g., by changing the operational parameters of these objects. Compared to a less-structured IoT system, users do not have to establish a connection to the (potentially large number of) actual objects or their devices to read and change their state.

Interaction. Requesters who wish to interact with the physical objects can perform their interaction with the digital twin in the cloud. The cloud answers their requests if possible or forwards them at an appropriate time to the relevant devices. By having the cloud as an intermediary, the cloud may also filter malicious requests (e.g., denial-of-service or denial-of-sleep attacks). Furthermore, the cloud serves as a single endpoint that simplifies communication with various devices that no longer have to be approached directly. The requesters may be other physical objects (or rather their digital twins) or other external entities.

Processing. Finally, the digital twins in the cloud collect an extensive amount of data, not only of the current state of a single physical object but also of all historical states of various objects. This accumulated data serves as a basis to perform complex computations (e.g., simulations and predictions). For example, the data can be used to make predictions on the future expected state, based on the current highly detailed state as well as previous developments derived from historical states. Such predictions allow searching for optimal decisions.

Application. Digital twins have been applied to various domains. For example, Qi and Tao [Qi] apply digital twins to monitor a manufacturing process, such that failures are detected, and the system can compute an optimized solution to address the problem. Kraft [Kraft] uses digital twins for aircraft components, where the collected sensor data enables simulations and prediction to reduce both the development and maintenance effort. The challenges of encryption also apply to protected digital twin systems.

3. Encryption to Protect Digital Twins (Recall)

Digital twin systems may also be appealing to be applied in use cases that handle sensitive data. An example would be a system that builds a digital twin of a human based on a person's medical data. However, privacy concerns arise if the sensitive data is stored in plain in a not-fully-trusted cloud service. The operator of the cloud or an insider attacker would be able to learn all sensitive digital twin data stored there.

Fortunately, encryption can also be applied to protect the confidentiality of data in a digital twin system. Our previously defined security architecture for digital twins [Hörandner-1] introduces key-policy conditional proxy re-encryption for end-to-end confidential yet flexible storage and sharing of digital twin data. This section recalls our previous architecture, which we extend in the subsequent section to elaborate on the challenges of recovery and maintenance, as well as privacy-preserving processing.

3.1. Building Block: Key-Policy Conditional Proxy Re-Encryption

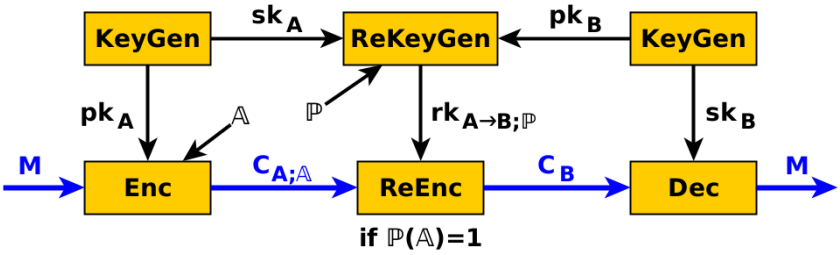

Figure 2: Key-Policy Conditional Proxy Re-Encryption

Key-policy conditional proxy re-encryption [Zhao] (KP-CPRE) extends upon classical proxy re-encryption. Classical proxy re-encryption [Blaze] enables a proxy to transform ciphertext encrypted for one entity into ciphertext for another entity. This re-encryption requires a re-encryption key that is generated with the private key of the first entity and the public key of the second entity. Key-policy conditional proxy re-encryption additionally controls with re-encryption operations are permitted based on attributes and policies. A set of attributes is attached to the ciphertext during encryption, while a policy is associated with the re-encryption. Re-encryption only succeeds if the ciphertext's attributes satisfy the re-encryption key's policy.

3.2. General Idea

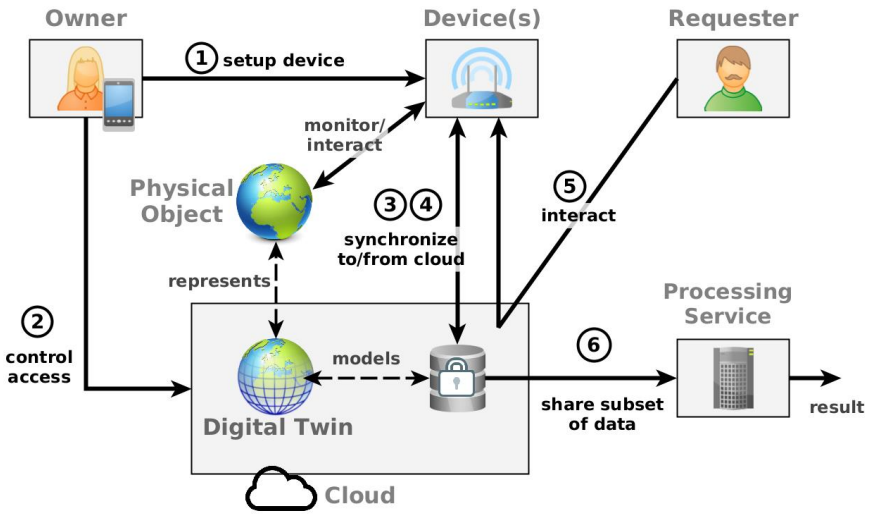

Figure 3: Security Architecture for Digital Twins

Basic Concept. Basically, devices encrypt their digital twin data for their owner before uploading it to the cloud. At first, only the owner can access these data by downloading and decrypting them, e.g., to monitor the state of the digital twins and their associated physical objects. By generating re-encryption keys, owners can also enable access to the digital twins for others. More precisely, the owner generates a re-encryption key with their private key for the public key of a selected receiver, such that the cloud may use this re-encryption key to transform the digital twin's ciphertext for others.

With this access control mechanism, the owner can enable the device to read its own digital twin data and may also share subsets of the digital twin data with sufficiently trusted recipients, e.g., to process this data.

Fine-grained data sharing is enforced according to attributes and policies. Devices derive a set of attributes, e.g., based on the data's content or type, which they attach to ciphertexts during encryption. Relying on these attributes, owners may define hierarchical access policies in the form of logic formulas (AND, OR). Such access policies are attached to the re-encryption keys and enable the cloud to only re-encrypt a subset of the owner's ciphertexts with attributes that satisfy the policy.

Processes. The following processes (c.f. Figure 3) integrate the described encryption layer to provide the functionality expected of a digital twin system while protecting the data.

1. **Setup Device**: Initially, the owner has to set up their devices (1) to enable the devices to monitor and interact with their physical objects and (2) to establish the connection with a digital twin in the associated cloud service.
2. **Control Access**: Next, the owner defines access control rules, which determine who is able to read and write the digital twins' data.
3. **Synchronize to Cloud**: After their configuration, the devices monitor their physical objects. Once they observe a change in the objects, they communicate this change to the digital twin in the cloud service.
4. **Synchronize from Cloud**: If the digital twin's data in the cloud is modified, this change is also forwarded to the respective device. The device may then use this new data to interact with the physical object.
5. **Interact**: Requesters may wish to interact with a physical object. Therefore, they direct their request to the digital twin in the cloud. This digital twin then forwards the request to the device associated with the physical object at an appropriate time.
6. **Processing / Share Subset**: To enable computation on the digital twins' data, the cloud may forward a suitable subset to a new actor, the so-called processing service. For example, these processing services may offer computations such as simulations or predictions. Of course, the data is shared according to the users' policies established in Process 2 (Control Access).

3.3. Processes

This section describes three related processes of our previous work [Hörandner-1], which set the stage for our subsequent extensions. For more formal information, we refer the interested reader to our publication [Hörandner-2].

Process 1: Setup Device. When setting up a new device, the device initially generates its own key pairs for encryption and signing. Next, the owner imports their public keys for encryption and signature verification. Additionally, the device obtains an initial state, the digitial twin's id, and connectivity information to reach the cloud service. Finally, the owner generates re-encryption keys to enable the device to read its own data, which are encrypted for the owner (c.f. Process 2 – Control Access).

Process 2: Control Access. Permission to read subsets of the owner's digital twin data is given by generating a re-encryption key. In particular, the owner generates a re-encryption key with their private key for a designated receiver identified by their public key. In this key generation process, the owner also supplies an access control policy that defines which data can be transformed. This re-encryption key (along with write tokens for write access described in [Hörandner-1]) is then installed at the cloud service.

Process 3: Synchronization to the Cloud. Once the device observes new data (e.g., a state change) on the linked physical object, the device aims to update the digital twin in the cloud. Therefore, the device signs the new data to ensure authenticity before encrypting the singed data for the device's owner and a suitable set of attributes. These attributes may be derived by the type

or content of the data (e.g., a location update or a temperature change). For efficiency, we build upon hybrid encryption such that the bulk of data is encrypted with symmetric schemes, and only the symmetric key is protected by the more heavy-weight key-policy conditional proxy re-encryption. The same symmetric key may be re-used for a suitable timeframe, such that PRE operations can be skipped for some subsequent data items. These signed and encrypted data are uploaded to the digital twin in the cloud. The device and other authorized receivers may access these data by having the cloud re-encrypt the items with an owner-generated re-encryption key.

4. Recovery and Maintenance

Recovery. Encryption has the goal to ensure that no-one gains access to the plain data unless they have knowledge of the associated decryption key. If the key or the encrypted data is lost, the plain data cannot be recovered. By storing the encrypted data in the cloud, which replicates its data, it is unlikely that the encrypted data will be lost. However, the decryption key needs to be kept confidential. Typically, such decryption keys are stored on a small number of devices that need to operate on the encrypted data.

Recovery is also a challenge in our security architecture for digital twins, as IoT devices and the owner's mobile phone store their own respective decryption keys. However, devices may break over time. If they hold the only copy of the decryption keys, access to the plain contents of their ciphertexts would become impossible. Therefore, a strategy to recover from device and key loss is required.

Flexibility for Maintenance. In digital twin systems, a large number of parties may interact in various roles: Various devices record characteristics of physical objects and synchronize them with their digital twins in different clouds. These devices belong to different owners. The digital twin data may be used in numerous computations that are of interest for various processing services. However, such complex systems are likely to change over time. Devices may break and therefore become an obstacle to continuously updating their digital twins. New processing services may emerge that are of interest to the owner but require data of the owner's digital twins. Authorized processing services may require additional data as their features evolve. In contrast, processing services and nodes might also lose the owner's trust, and their access permissions should therefore be revoked.

4.1. General Idea

Basic Concept. To regain access to encrypted data, we need to ensure that the ciphertext is not lost and that a key remains to decrypt this ciphertext. Therefore, we store an encrypted backup of the ciphertext in the cloud. This backup needs to be continuously synchronized, which perfectly fits the properties of our security architecture for digital twins. The data is not encrypted for the IoT device but for the device's owner. We use key-policy conditional proxy re-encryption to dynamically give access to encrypted data, i.e., by generating new re-encryption keys to transform the data for someone else. Therefore, even if the original IoT device is broken, it is possible to generate a new re-encryption key to recover the data on a replacement device.

Another challenge remains: This basic concept relies on the private key stored on the owner's device to generate re-encryption keys. The owner's device may also break. Therefore, it is necessary to create a backup of the owner's private key and keep it confidential, e.g., on a flash drive or QR-coded letter in a secure physical location, password-encrypted in the cloud, or distributed as fragments to several IoT devices.

Processes for Recovery and Maintenance. The following processes accommodate the dynamic nature of a digital twin system with regard to recovery in case of device malfunction and maintenance of access policies while supporting encrypted data.

M1. **Recover/Replace Device**: After a device is no longer available (e.g., broken), an owner wishes to recover the functionality of the device/system and the device's data. Therefore, the owner sets up a replacement device and triggers the synchronization of the old device's digital twin data to the new device.

M2. **Recover Management Device**: A owner may also lose their management device, which holds the owner's master key material. In this case, these vital keys need to be recovered on a replacement management device.
M3. **Change Access of Processing Service/Node**: Over time, the trustworthiness or data requirements of actors involved in the processing may change. The owner may commodate these changing circumstances by revoking access or adjusting access rights to larger/smaller data sets.
M4. **Replace Processing Node**: In Section 5, processing nodes are introduced to enable privacy-preserving processing. The privacy guarantees of such processing rely on the assumption that the processing nodes do not collude. If the trust in the operators of these processing nodes erodes over time, the owners may replace an insufficiently trusted processing node with another one.

4.2. Processes

This section gives details on the recovery and maintenance processes within the extended security architecture for digital twins.

Process M1: Recover/Replace a Device. If a device breaks, the functionality of the overall system (i.e., continuous synchronization of the physical object with its digital twin) can be recovered by installing a replacement device with the same data. Therefore, the owner runs process 2 (Control Access) to generate a re-encryption key for the owner's private key to the new device's public key, which makes it possible to re-encrypt all data of the old device for the new device. This new device is consequently able to take over all responsibilities of the old device seamlessly. Access rights of the old device should be revoked by removing the respective re-encryption keys from the cloud service.

Process M2: Recover the Management Device. The owner's management device holds their keys, which are crucial in our system as they enable controlling access to the digital twin data. Without this device and key material, owners lose the ability to access their digital twin data or change access permissions. Therefore, our system needs to foresee strategies to recover for such circumstances on a new management device the owner obtained as a replacement. Related research proposes various approaches that can be integrated into our system: (A) Of course, owners may create backups of their essential keys on a flash drive or printed as a QR code on a sheet of paper, which they need to store in a secure location. (B) Owners may also derive a backup key from their password and encrypt their essential keys with this backup key before uploading them to the cloud holding their digital twins. (C) Alternatively, password-protected secret-sharing [Abdalla] may be used to split the owner's essential keys into multiple shares, distribute these shares to a set of the owner's devices, and, once necessary, use the password to recover the essential key from the shares of these devices.

Process M3: Change Access of Processing Service / Node. Key-policy conditional proxy re-encryption enables changing access permissions flexibly. Owners can adjust permissions by generating new re-encryption keys for the same recipient with a policy that grants more or less access rights. Such adjusted re-encryption keys then replace the respective old re-encryption keys at the cloud service.

Process M4: Replace Processing Node. The privacy of the users' data within multi-party computation builds upon the assumption that the individual processing nodes do not collude. If a processing node loses the users' trust, the users may replace this node with another sufficiently trusted node. Therefore, users remove the old re-encryption key towards the untrusted node and generate a new re-encryption key towards the new node. These re-encryption keys are associated with an access policy, which defines the share index for each data item the processing node may receive. Multi-party computation dictates two conditions that must be observed when selecting a replacement node: No node must learn different shares (i.e., shares with a different index) for the same data item. Data from multiple owners can only be processed by the same agreed-upon set of nodes.

5. Privacy-Preserving Processing

Encryption is a useful tool to protect the confidentiality of data in an attempt to preserve the users' privacy. If users encrypt their data before uploading them to the cloud, the cloud cannot learn the plaintext. Consequently, the cloud also cannot process that encrypted data (if traditional encryption mechanisms are used). Generally, it is hard to conceive how to process something you cannot "see". However, some computations would also be in the users' interest, as long as their privacy is not violated in the processing.

Processing based on Subsets. A simple approach would be to identify relevant subsets of the users' data (e.g., stored in the cloud) that are not too sensitive and reveal these subsets with another entity, i.e., a processing service, that is sufficiently trusted to learn these data subsets. The processing service could decrypt the data and perform its computation on the plain data. This approach may be sufficient for several use cases where the subsets by themselves are not too sensitive.

Goal. When handling highly sensitive data, these data might be too sensitive to even reveal in part. We strive for an approach that enables processing where only the data owners learn the input data and only the receiver learns the output, but any cloud services or other entities do not learn the inputs and outputs. Such privacy-preserving processing is also relevant for digital twins, as computation on the vast gathered data set is a core benefit, which is unfortunately hampered by encryption mechanisms.

Outline. This section describes our approach to privacy-preserving processing based on secret-sharing-based multi-party computation in the context of protected digital-twin data. Initially, we give an overview of approaches to computation on protected data before focusing on multi-party computation. We describe the general idea of applying multi-party computation to extend our security architecture for digital twins and then go into detail on the extended processes. Finally, we evaluated the feasibility and performance of this approach in an example use case of privacy-preserving contact tracing.

5.1. Overview of Related Work on Computation on Protected Data

In research efforts over the last decades, approaches have been introduced to achieve a contradictory goal: To process data without ever learning it. While we build upon multi-party computation, this section gives an overview of other possibilities.

Functional Encryption. In functional encryption [Boneh], a ciphertext can be decrypted into the result of a function on the underlying plaintext, instead of decryption into the plaintext itself. An evaluation key is required to perform such decryption. An authority for a specific function generates such evaluation keys. The supported class of functions depends on the various proposed schemes. For example, functional encryption schemes have been introduced for access control, search on encrypted data, or computation, e.g., of inner production functions. Unfortunately, efficient functional encryption for general-purpose functions remains elusive.

Homomorphic Encryption. Homomorphic encryption [Gentry] enables computation in the encrypted domain. These schemes offer operations that can be performed on the ciphertexts that cause a related operation on the underlying plaintexts. For example, there may be an operation on two encrypted numbers that outputs a new ciphertext, which can be decrypted into the sum or product of the two numbers. If a scheme supports both an unlimited number of additions and multiplications in the encrypted domain, we refer to it as fully homomorphic encryption. Such a scheme can compute arbitrary functions, as any Boolean circuit can be evaluated by those operations. The execution times also of current schemes are still prohibitively high for many practical applications.

Trusted Execution Environments. Hardware manufacturers have extended their devices to introduce trusted execution environments, e.g., Intel's SGX [Costan]. These environments are isolated from the rest of the system (e.g., protect their memory and program execution) and offer remote attestation, which enables to convince external parties of which code is running in the trusted execution environment. These properties may convince users that the execution environment can be sufficiently trusted to handle and process the users' sensitive data, even if the execution environments are running on hardware operated by not-fully-trusted cloud providers. By securely sharing the users' data with the program in the trusted execution environment, the users' data can be processed there in plain. While such an approach has performance benefits, the user needs to trust the hardware manufacturers and the strength of the isolation mechanisms, which have been broken multiple times.

5.2. Building Block: Multi-Party Computation

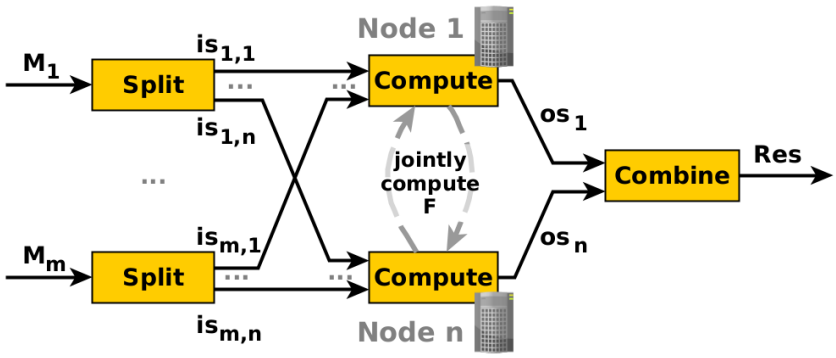

Figure 4: Secret Sharing-based Multi-Party Computation

Secure multi-party computation [Bogdanov, Yao] (MPC) enables a set of multiple parties to engage in an interactive protocol to jointly compute a function on the parties' inputs without revealing these inputs to the other parties. In this work, we focus on multi-party computation based on secret sharing [Shamir]. Initially, each input data is split into multiple input shares. These input shares are handed to the respective processing nodes, which engage in a protocol to jointly compute a function on the input shares. Each node computes an output share. Finally, these output shares can be combined into the result of the function. As long as not too many nodes collude, the individual nodes learn neither the plain input data nor the function's result.

5.3. General Idea

Basic Concept. To achieve privacy-preserving processing in a digital twin context, we apply secret-sharing-based multi-party computation. The devices split the digital twin data into shares before uploading them to the cloud. We introduce processing nodes as new actors that will receive the shares from the cloud and use them to compute a function. The set of processing nodes engages in a multi-party protocol with each other, where each node holds a different share of the individual data items. These processing nodes are assumed to not collude with each other.

Further, we apply key-policy conditional proxy re-encryption to protect and distribute the shares of digital twin data. The encrypted shares can be stored in the cloud without the cloud learning any plain data. Additionally, proxy re-encryption allows to re-encrypt and send the individual shares to the correct processing nodes.

After the processing nodes have finished their computation, each holds a share of the output. The nodes encrypt their output shares for the final receiver of the output. This receiver is able to decrypt the shares and combine them into the output. Only this receiver learns the output.

Control by Data Owners. Data owners control the sharing of their data by generating re-encryption keys with fine-grained access policies (e.g., only state data on the phone's locations for a specific timeframe). These re-encryption keys enable the cloud to make individual shares available to the respective nodes. Furthermore, data owners define a policy on the class of permitted functions and authorized receivers of the result. These policies are verified at each individual processing node before they start their computations.

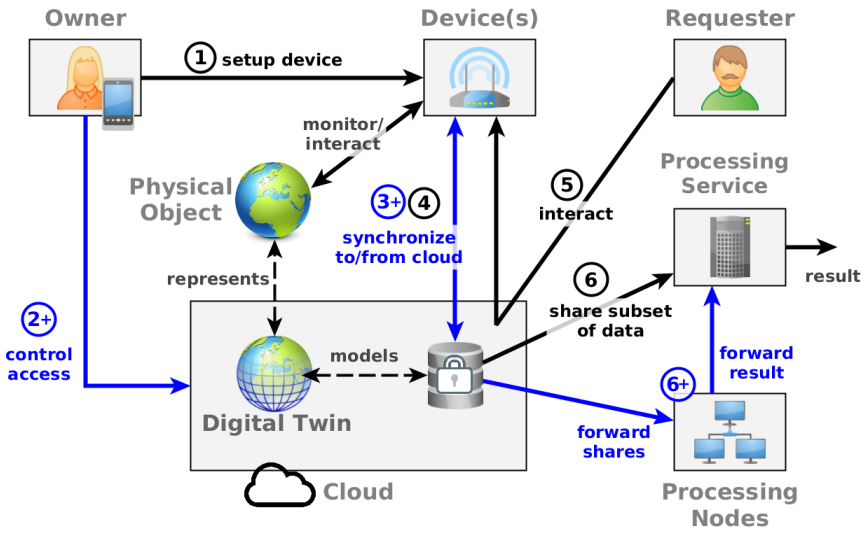

Figure 5: Overview of Privacy-Preserving Processing

Extended Processes. Figure 5 illustrates the extensions to our security architecture for digital twins [Hörandner-1]. Previously introduced processes are shown in black, while processes related and extended for computation on protected data are highlighted in blue.

2+. **Control Access – Permit Processing**: The owner not only grants read access to processing nodes as in Process 2 but also specifies (a) which data may be used to compute, (b) which classes of functions are permitted, and (c) which parties may learn the result.
3+. **Synchronize to Cloud**: Before encrypting and uploading the data to the digital twin in the cloud, the device prepares these data for computation in a multi-party computation protocol. That is, the data is split into a share for each processing node.
6+. **Processing with Multi-Party Computation**: To process data of multiple digital twins, the cloud forwards the shares of relevant data to the respective processing nodes. These nodes jointly compute a function on the data without learning their inputs or outputs. Finally, the nodes forward the shares of the result to a receiver (e.g., processing service), who is able to combine them into the function's result.

5.4. Processes

Process 2+: Control Access – Permit Processing. Owners follow two steps to grant permission to process their data. Firstly, owners grant read access for specific subsets of digital twin data to processing nodes by generating re-encryption keys (c.f. Process 2). Secondly, owners generate a so-called process token to define which classes of functions are admissible for their data and which party may receive the result. Such a process token is signed by the owner and verified at the processing nodes before engaging in the multi-party computation protocol.

Process 3+: Synchronize to Cloud. Enabling processing for secret-sharing-based multi-party computation requires preparing the data. Therefore, the device first splits each input data into exactly one input share per processing node. The individual input shares are then encrypted with key-policy conditional proxy re-encryption for the owner with a set of attributes that denotes the number of the processing node. These encrypted input shares are then uploaded to the digital twin in the cloud.

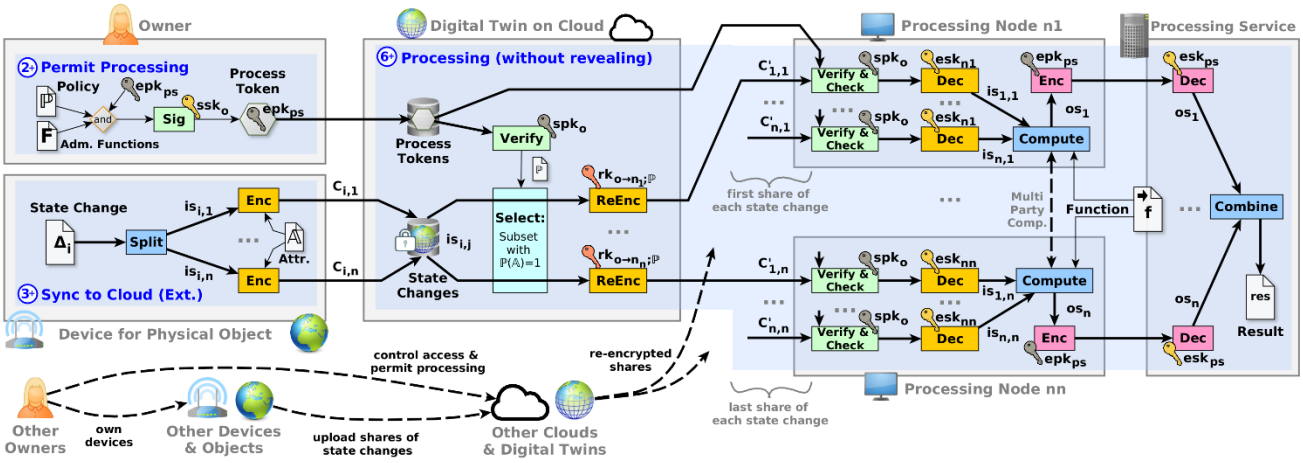

Figure 6: Details on Privacy-Preserving Processing in the Context of Digital Twins

Process 6+: Processing with Multi-Party Computation. Our approach to privacy-preserving processing based on multi-party computation can be split into three phases that involve different actors.

Firstly, the cloud services gather the encrypted input shares of relevant data items from their digital twin databases. Given the owners' consent and appropriate re-encryption keys (c.f. Process 2+), the cloud transforms the encrypted input shares for the respective process nodes.

Secondly, the individual processing nodes initially verify that the involved data owners consent to the planned processing. That is, the processing node verifies the process token of each involved data owner with regard to the involved data subset, the admissibility of the function, and the authorized receiver of the function's result. In case of success, the processing node decrypts the re-encrypted input shares and applies them in a multi-party computation protocol with the other processing nodes. Each processing node obtains one share of the function's result, which they encrypt for the authorized receiver.

Finally, the authorized receiver (e.g., a processing service) decrypts the output shares of the individual processing nodes. These output shares can then be combined into the function's result.

5.5. Evaluation: Privacy-Preserving Contact Tracing

This section presents an evaluation of our approach to privacy-preserving processing. As the execution time depends heavily on the complexity of the function being computed, it can only be evaluated for a concrete function. We have chosen privacy-preserving contact tracing as an example use case to illustrate the efficiency of our approach and give a general indication of the magnitude of computing resources needed.

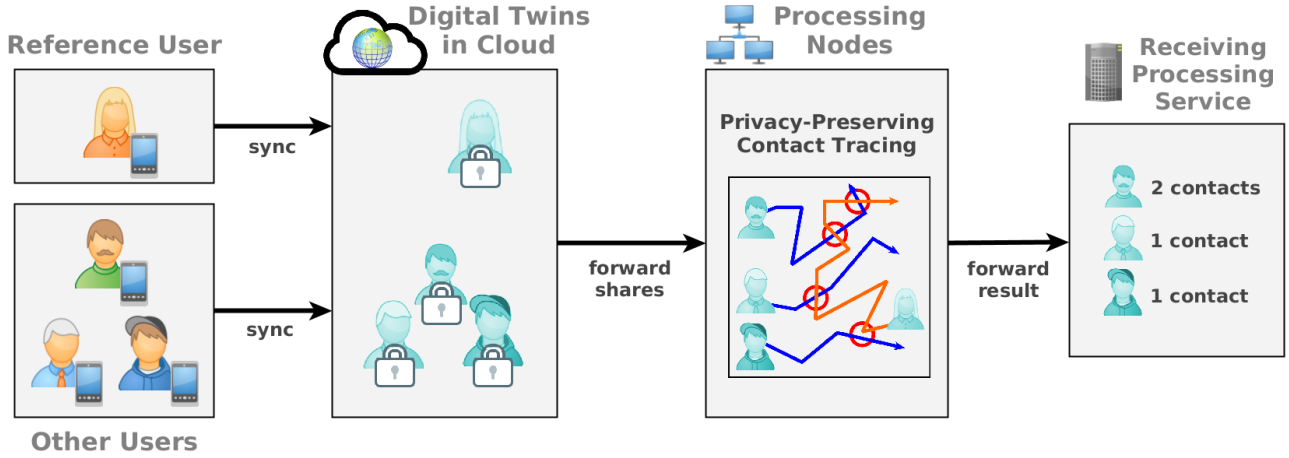

Figure 7: Privacy-Preserving Contact Tracing

Example Use Case. Privacy-preserving contact tracing based on location information serves as our example use case. Location data that is collected over time can reveal quite sensitive information, e.g., relationships between people, movement patterns, or even the health status if the person frequently visits a hospital. Therefore, these sensitive location data must not be revealed to others, while computation on that data is nevertheless desired. In our use case, the paths of a number of people are compared to one reference person's path (e.g., an infected patient). The function should eventually output how often each of these people has been too close to the reference person, thereby indicating who is likely to have contracted the disease. The process is illustrated in Figure 7. With multi-party computation, we compute this function without revealing the actual locations or the function's outputs to the processing nodes.

Table 1: Variables and Their Values in Our Example Use Case

Variable	Use Case Value
#devices	#users
#devices/user	1
#locations/device	50
#items/device	100
#shares/device	300
#epochs	50
#nodes	3
#results	#users

Parameters. The parameters for our example use case are shown in Table 1. We assume that each person uses their mobile phone to build a digital twin of themselves (#devices/user = 1). These phones each collect 50 location points (#locations/device = 50), which consist of an x and y coordinate (#items/device = 100) and are split into three shares (#shares/device = 300) for the three processing nodes (#nodes = 3). A users' locations are collected in different epochs (#epochs = 50), such that no symmetric keys can be re-used to highlight the worst-case scenario. The use case's function returns one output per user (#results = #users), i.e., the number of path intersections of that person with the reference person.

MPC Instantiation. Our evaluation builds upon the SCALE-MAMBA framework [SCALE-MAMBA], which exposes the functionalities of multi-party computation via a high-level interface. This framework enables to write functions in a Python-like syntax but compile these functions such that they can be executed in parallel on multiple parties and their respective shares of the input data. In our setup, we use a network of three nodes with 30ms round-trip time between them.

Table 2: Execution Times of Privacy-Preservice Computation and Our Example Use Case (in Milliseconds)

	Generic		Use Case	
	Time [ms]	**Multiplier**	**Time [ms]**	**Multiplier**
2+ Control Access (on Phone)		**per user**		**per user**
PRE.RKGen	53.98	x #devices/user x #nodes	161.95	
SIG.Sign	1.44	x 1	1.44	
			SUM= 163.39	(per user)
3+ Sync to Cloud (on Phone)		**per device**		**per user**
MPC.Split	0.02	x #items/device	2.00	
AES.Enc	<0.01	x #shares/device	0.23	
PRE.Enc	51.13	x #epochs x #nodes	7670.18	
			SUM= 7672.41	(per user)
6+ Processing (on PC)		**cumulated**		**cumulated**
PRE.ReEnc	4.52	x #devs. x #epochs x #nodes	678.54	x #users
SIG.Verify	0.19	x #users	0.57	x #users
PRE.Dec	2.35	x #devices x #epochs	352.48	x #users
AES.Dec	<0.01	x #shares	0.54	x #users
MPC.Compute	(depends on function)		6530.58	x #users
			12433.61	(constant)
PKE.Enc	0.18	x #results	0.55	x #users
PKE.Dec	0.12	x #results x #nodes	0.36	x #users
MPC.Combine	0.08	x #results	0.08	x #users
			SUM= 7563.70	(per user)
			12433.61	(constant)

in parallel on each node

in parallel on each node

Results. Table 2 presents two types of performance numbers. Firstly, generic execution times are shown that can be applied for various parameters. Secondly, these use-case-specific times have been collected, where some parameters have been applied according to Table 1. Also, for these use-case specific times, we have collected execution times of the multi-party computation for different numbers of users, showing linear growth. Both types of performance numbers have been gathered for the three relevant processes: 2+ Control Access, 3+ Synchronize to Cloud, and 6+ Processing. Process 2+ Control Access is performed on the users' mobile phones (i.e., OnePlus 6T) and takes ~163ms per user. Process 3+ Synchronize to Cloud runs continuously on the same phones. The measured location points are split, encrypted, and uploaded to the digital twin in the cloud once each epoch, taking an accumulated ~7.7s per user over all epochs. For process 6+ Processing, the cloud distributes the relevant shares to the processing nodes, which engage in the multi-party computation and hand the results to the final receiver. This process requires about 7.6s per user, with ~12.4s constant overhead. This calculation takes the multi-party computation as a parallel process on each node. The digital twin cloud, processing nodes, and final receiver are simulated on a PC with an AMD Ryzen 5600X CPU.

6. Conclusion

This work has tackled challenges caused by encryption to protect data, namely privacy-preserving processing of that data and recovery in case of key loss. These challenges were investigated in the context of our security architecture for digital twin [Hörandner-1], where the digital twin data is encrypted. Key-policy conditional proxy re-encryption not only enables to securely store digital twin data in the cloud and share it with others. We can use the flexibility of this cryptographic mechanism to also change share permissions at any point in time. Furthermore, broken devices can be replaced with new devices that gain access to the same data, in re-encrypted form. Consequently, owners are able to maintain and recover the overall system in the face of changing trust relationships and unexpected circumstances. To tackle the challenge of computation on protected data, our concept also integrates multi-party computation, where the protected data can be processed without revealing the plain data or the results to the individual nodes involved in the computation. The performance evaluation shows the feasibility of the concept. Our example use case on privacy-preserving contact tracing scales linearly with a computation time of about 7.6s per user for 50 uploaded location points each.

7. Bibliography

[Hörandner-1] Hörandner, Felix (2021). “Security Architecture for Digital Twins”. A-SIT Report, https://technology.a-sit.at/en/security-architecture-for-dititал-twins/

[Hörandner-2] Hörandner, Felix and Bernd Prünster (2021). "Armored Twins: Flexible Privacy Protection for Digital Twins through Conditional Proxy Re-Encryption and Multi-Party Computation". SECRYPT, 2021, https://www.scitepress.org/Link.aspx?doi=10.5220/0010543301490160

[Yao] Yao, A. C. (1982). “Protocols for Secure Computations (Extended Abstract)”. In: FOCS. IEEE Computer Society, pp. 160–164.

[Bogdanov] Bogdanov, D., Niitsoo, M., Toft, T., and Willemson, J. (2012). “High-performance secure multi-party computation for data mining applications”. Int. J. Inf. Sec., 11(6), pp. 403–418.

[Shamir] Shamir, A. (1984). “Identity-Based Cryptosystems and Signature Schemes”. Advances in Cryptology, Proceedings of CRYPTO ’84, Lecture Notes in Computer Science. Springer, 1984, pages 47–53.

[Blaze] Blaze, M., Bleumer, G., and Strauss, M. (1998). "Divertible Protocols and Atomic Proxy Cryptography". In: EUROCRYPT. Vol. 1403. LNCS. Springer, pp. 127–144.

[Zhao] Zhao, J., Feng, D., and Zhang, Z. (2010). "Attribute-Based Conditional Proxy Re-Encryption with Chosen-Ciphertext Security". In: GLOBECOM. IEEE, pp. 1–6.

[SCALE-MAMBA] Aly, A, Cong, K, Cozzo, D, Keller, M, Orsini, E, Rotaru, D, Scherer, O, Scholl, P, Smart, N., Tanguy, T, and Wood, T (2021). SCALE–MAMBA v1.11: Documentation. Accessed: 2021-02-12 https://homes.esat.kuleuven.be/~nsmart/SCALE/Documentation-SCALE.pdf

[Abdalla] Abdalla, M., Cornejo, M., Nitulescu, A., and Pointcheval, D. (2016). “Robust Password-Protected Secret Sharing”. In: ESORICS (2). Vol. 9879. LNCS. Springer, pp. 61–79.

[Barricelli] Barricelli, B. R., Casiraghi, E., and Fogli, D. (2019). "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications". IEEE Access, 7, pp. 167653–167671.

[Fuller] Fuller, A., Fan, Z., Day, C., and Barlow, C. (2020). "Digital Twin: Enabling Technologies, Challenges and Open Research". IEEE Access, 8, pp. 108952–108971.

[Qi] Qi, Q. and Tao, F. (2018). “Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison”. IEEE Access, 6, pp. 3585–3593.

[Kraft] Kraft, E. M. (2016). “The Air Force Digital Thread/Digital Twin - Life Cycle Integration and Use of Computational and Experimental Knowledge”. In: 54th AIAA Aerospace Sciences Meeting.

[Costan] Costan, V. and Devadas, S. (2016). “Intel SGX Explained”. IACR Cryptological ePrint Archive. http://eprint.iacr.org/2016/086

[Boneh] Boneh, D., Sahai, A., and Waters, B. (2011). “Functional Encryption: Definitions and Challenges”. Theory of Cryptography Conference. Vol. 6597. LNCS. Springer. pp. 253-273.

[Gentry] Gentry, C. (2009). “Fully homomorphic encryption using ideal lattices”. Symposium on Theory of Computing, STOC 2009. ACM. pp. 169-178.