Armored Twins: Flexible Privacy Protection for Digital Twins through Conditional Proxy Re-Encryption and Multi-Party Computation

Felix Hörandner[a] and Bernd Prünster[b]
Graz University of Technology, Graz, Austria
felix.hoerandner@iaik.tugraz.at, bernd.pruenster@iaik.tugraz.at

Keywords: Digital Twin, Proxy Re-Encryption, Fine-Grained Access Control, Multi-Party Computation, Recovery

Abstract: Digital twins, i.e., up-to-date digital copies of a physical object maintained in the cloud, make it possible to conveniently review a physical object's state, indirectly interact with the physical object, or perform computations on the object's state and history – also in combination with data from other digital twins. The concept of digital twins has seen wide uptake in Internet of Things use cases, e.g., in manufacturing to monitor a product's lifecycle, or precision medicine to provide personalized treatment. Besides these benefits, challenges arise, especially if the involved data producers, clouds and data consumers are not in the same trusted domain: Who owns and controls the data? Are the parties (e.g., cloud) sufficiently trusted to handle privacy-sensitive data? In this work, we propose ARMOREDTWINS, i.e., a system for digital twins that protects the confidentiality of digital twin data while providing flexible and fine-grained sharing by employing key-policy conditional proxy re-encryption to enable processing on subsets of the data. Alternatively, to support computation on very sensitive data, our system integrates secure multi-party computation, which does not reveal the data items to the individual nodes performing the computation. Benchmarks of our implementation highlight the system's feasibility and practical performance.

1 INTRODUCTION

The concept of a *digital twin* (Barricelli et al., 2019) has gained significant traction over recent years, also being listed in Gartner's strategic technology trends for 2019[1]. Such a digital twin is the constantly updated, electronic representation of a physical object, e.g., a product in a manufacturing process, personal and identity data of a citizen, or medical measurements of a patient. The object's properties are recorded by one or more devices (e.g., phones, wearables, medical devices, or industry sensors) and synchronized to a cloud service. Instead of interacting with the physical objects directly, users and devices engage with the digital twin on the cloud service to review the twin's state and history, as well as to trigger state changes and actions at the physical object.

The collected data of one or more digital twins enables computations for various use cases, as surveyed by Barricelli et al. (2019) and Fuller et al. (2020). Digital twins of products or a production line can be used to monitor the manufacturing process, detect failures, and compute optimized solutions (Qi and Tao, 2018). In an open domain such as a smart city environment, digital twins of cars allow managing traffic, connecting cars, and providing real-time feedback for drivers with the aim to improve safety (Chen et al., 2018). Medical digital twins of patients can be built from health data by wearables and in-house sensors to monitor the health status and predict problems (Liu et al., 2019). Also, digital twins of citizens could include location data, which is highly sensitive, but would be very valuable in contact-tracing scenarios, as vigorously discussed in many countries during the COVID-19 pandemic.

Challenge: Privacy. Privacy, security and trust are central challenges for the deployment of digital twins (Fuller et al., 2020), as sensitive digital twin data are valuable targets for (insider) attackers, e.g., to spy on manufacturing details, health status, and behavior. Previous research has focused on closed domains, where the objects and devices as well as their data and the cloud are under the control of the same entity (e.g., a manufacturing company) or assumed to be fully trusted. However, challenges arise in open domains, where separate entities own the digital twins

[a] https://orcid.org/0000-0001-8591-3463
[b] https://orcid.org/0000-0001-7902-0087

[1] https://www.gartner.com/en/documents/3904569/top-10-strategic-technology-trends-for-2019-digital-twin

(i.e., the physical object's sensitive data), operate the storage service in the cloud, and perform computations on that data. For sensitive data, there is a need for a multi-user system that (a) protects the data of digital twins, so that the cloud does not learn them, and (b) enables owners to control which entities are authorized to access specified subsets of their data.

Challenge: Flexibility. Flexibility in the digital twin system and the data protection mechanisms is also essential, as trust in data receivers and processing services change over time and new services emerge that would benefit from re-using previously collected data. Unfortunately, designing a system for digital twins that both protects the digital twins' data as well as provides flexibility is not a trivial task. Of course, it is possible to naively use symmetric or asymmetric cryptographic mechanisms to encrypt the digital twins' data before uploading them to the cloud. However, this naive approach is inflexible, as it requires configuring each device to encrypt its data or various subsets for multiple receivers, e.g., for services that should process the data. These sharing configurations have to be updated on each device every time data should be shared with new receivers or the extend of shared subsets changes. Thus, such approaches require significant maintenance effort and hinder the opportunistic reuse of data for new services.

Our Contribution. We propose ARMOREDTWINS: a flexible protection-layer for digital twins, which (1) protects the digital twins' data, (2) gives owners control over their digital twins, (3) retains the functionality and benefits of digital twins, i.e., enables interaction with other devices, and supports processing of digital twin data, and (4) provides dynamic maintenance of sharing permissions, as well as a convenient recovery and replacement strategies after device loss.

Protecting Digital Twins. Our ARMOREDTWINS system integrates key-policy conditional proxy re-encryption (KP-CPRE) (Zhao et al., 2010) to achieve flexible end-to-end secure data-sharing. Devices collect data about physical objects to build digital twins and encrypt it with KP-CPRE for the owner according to a set of attributes that describes the data (e.g., location data, blood-pressure data). With KP-CPRE, the owner can then authorize sharing of specified data subsets with others by generating a re-encryption key with a fine-grained policy over the ciphertext attributes. This re-encryption key enables the cloud to transform the digital twin data for the receiver without learning the underlying plaintext. Additionally, the sharing mechanism can also be applied to encrypt requests by external entities to protect the request's confidentiality while enabling to route these requests to a designated device where they can be decrypted. Our system protects the confidentiality of the digital twin data but remains flexible and gives owners control over which data subsets should be made available to processing services for their computations.

Maintenance & Recovery. Our approach also proves useful when reacting to changed trust relationships or new services. With KP-CPRE, owners can revoke access by removing re-encryption keys from the cloud or establish new sharing permissions for previously collected data by generating new re-encryption keys, which enable the opportunistic reuse of data as new data processing services emerge. Additionally, our system offers a convenient recovery strategy if a defective device needs to be replaced: Even through the protection layers, the system can seamlessly shift the old device's information (stored as digital twin in the cloud) as well as incoming requests to the replacement device.

Processing without Revealing Data. While the selective sharing presented before enables processing for data subsets the owners feel comfortable to disclose, owners may refrain from exposing data of very sensitive nature to processing services. To tackle this challenge, our system also integrates secure multi-party computation (MPC) (Yao, 1982) as an alternative to also support processing on privacy-sensitive data. In this approach, the devices split the digital twin data into multiple shares, which are encrypted with KP-CPRE and shared with separate processing nodes. These processing nodes use MPC to compute a function on the shares of multiple digital twins from various owners without ever learning the plain data or result. Only the final receiver obtains the result.

Implementation & Evaluation. Finally, we implemented our proposed ARMOREDTWINS system to highlight its feasibility and practical performance. Our benchmarks of the involved cryptographic mechanisms make it possible to derive the computational effort of our system's individual phases. Further, we implemented privacy-preserving contact tracing on privacy-sensitive location data as an example use case to evaluate the effort of computing a function without revealing the involved data. Our results show a low performance overhead for KP-CPRE-based data sharing in general and linear scalability of the MPC computation in the contact-tracing use case.

2 RELATED WORK

Early work on digital twins focused on their features, e.g., convenient, up-to-date access to data and simulations, mainly in closed domains with static trust relationships. While security, privacy and trust have been

neglected, they were recently identified as main challenges for the use of digital twins (Fuller et al., 2020) and have started to gain attention in the research community.

Twins to Enhance Security. The digital twin concept has the power to enhance security. Communication exclusively through the cloud enables protection against denial-of-sleep attacks, while the data collected in digital twins allows performing security and safety analyses as well as intrusion detection without disrupting live systems (Eckhart and Ekelhart, 2018).

Security for Twins. In open domains, where multiple actors are involved, who are not fully trusted, the confidentiality of sensitive digital twin data has to be ensured, e.g., to protect company secrets or the privacy of involved people. In a first step, it is possible to enforce access control at the cloud before sharing data with others (Kern and Anderl, 2020). Gehrmann and Gunnarsson (2020) propose a security architecture for digital twins that integrates security testing and intrusion detection based on digital twins, enforces access control, and uses secure communication channels.

Concrete security mechanisms for digital twin systems were introduced by Dietz et al. (2019); Huang et al. (2020); Putz et al. (2021). They propose to build upon distributed ledger technology to protect against modifications and integrate symmetric as well as public key encryption to ensure the confidentiality of digital twin data. These approaches are best suited for scenarios with a limited amount of actors, as sharing and maintaining decryption keys in the face of many devices, owners and receivers becomes a complex task. Generally, encryption layers remain an obstacle for digital twin functionality, as reviewing and processing digital twin data requires to share decryption keys, and strategies to recover after loss of devices and keys need to be developed.

Our contributions complement the related work by introducing flexible and fine-grained data sharing into the digital twin ecosystem without sacrificing digital twin functionality, while enabling dynamic maintenance, replacement of devices, and processing on subsets. Furthermore, we introduce processing on highly-sensitive digital twin data even without revealing that data in the computation process.

3 CRYPTOGRAPHIC BUILDING BLOCKS

This section explains two fundamental building blocks of our system: key-policy conditional proxy re-encryption and secure multi-party computation.

Key-Policy Conditional Proxy Re-Encryption. Proxy Re-Encryption (PRE) (Ateniese et al., 2006; Blaze et al., 1998) extends asymmetric encryption by enabling a semi-trusted proxy to transform ciphertext encrypted for one user into ciphertext then encrypted for another user, without learning the underlying plaintext in any intermediate step. For this transformation, the proxy requires a re-encryption key, which is generated based on the private key of the data owner and the public key of the intended recipient (in non-interactive schemes).

Key-policy conditional proxy re-encryption (KP-CPRE) (Zhao et al., 2010) enables more fine-grained access control. Senders attach a set of attributes $\mathbb{A}$ during encryption. The resulting ciphertext can only be re-encrypted successfully if the used re-encryption key has been generated for a sufficiently strong policy $\mathbb{P}$ of AND/OR logic gates, i.e., where $\mathbb{P}(\mathbb{A}) = 1$. We focus on schemes that are unidirectional, single-hop, and non-interactive. The algorithm definition below has been adapted from Zhao et al., where we made the ability to specify attributes public, i.e., removed AKeyGen.

Definition 1 (KP-CPRE) *A fine-grained key-policy conditional proxy re-encryption (*KP-CPRE*) scheme consists of the following algorithms:*

$\mathsf{KeyGen}() \to (\mathsf{sk}, \mathsf{pk})$**:** *This algorithm generates a secret and public key* $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{RKGen}(\mathsf{sk}_A, \mathbb{P}, \mathsf{pk}_B) \to \mathsf{rk}_{A \to B;\mathbb{P}}$**:** *Given a secret key* sk_A *of user A, the public key of user B, and a policy* $\mathbb{P}$*, this algorithm creates a re-encryption key* $\mathsf{rk}_{A \to B;\mathbb{P}}$.

$\mathsf{Enc}^l(\mathsf{pk}, \mathbb{A}, M) \to C_A$**:** *Given a public key* pk*, a message M, and an attribute set* $\mathbb{A}$*, this algorithm outputs a level-l ciphertext* C^l*. A first-level ciphertext* C^1 *can be re-encrypted, while a second-level ciphertext* C^2 *cannot be (further) re-encrypted.*

$\mathsf{ReEnc}(\mathsf{rk}_{A \to B;\mathbb{P}}, C_A^1) \to C_B^2$**:** *Given a first-level ciphertext* C_A^1 *with attribute set* $\mathbb{A}$ *and a re-encryption key* $\mathsf{rk}_{A \to B;\mathbb{P}}$ *for policy* $\mathbb{P}$*, this algorithm translates the ciphertext for the other user, iff the attributes satisfy the policy, i.e.,* $\mathbb{P}(\mathbb{A}) = 1$.

$\mathsf{Dec}^l(\mathsf{sk}, C^l) \to M$**:** *Given the secret key* sk *of a user and a ciphertext* C^l *for the same user, the algorithm returns the underlying message M or fails.*

Secure Multi-Party Computation (MPC) (Yao, 1982) enables a set of nodes to compute a function without revealing their input data to the other nodes. Two main directions have evolved: (1) schemes based on garbled circuits (Yao, 1982) and (2) schemes based on secret sharing techniques (Bogdanov et al., 2012). We focus on secret sharing-based MPC. Given shares

of the input data set, n processing nodes are able to jointly compute a function f on the input data set, without learning their plaintext, and create output shares that can be combined to the function's result. During this process, the processing nodes neither learn the plain input data nor the result, as long as not more than a threshold t of the nodes collude.

Definition 2 (MPC) *A secure multi-party computation (*MPC*) protocol based on secret sharing for a function f on a set of inputs $(M_i)_{i\in I}$, where not more that t of n processing nodes may be corrupted, consists of the following algorithms:*

$\mathsf{Split}(M,t,n) \to (\mathsf{is}_j)_{j\in[n]}$**:** *The algorithm applies a secret sharing technique to split the message M into n input shares $(\mathsf{is}_j)_{j\in[n]}$ with a threshold t.*

$\mathsf{Compute}((\mathsf{is}_{i,j})_{i\in I},(\mathsf{n}_k)_{\forall k\neq j}) \to \mathsf{os}_j$**:** *The processing node n_j engages in an interactive protocol with the other nodes $(\mathsf{n}_k)_{\forall k\neq j}$. Each n_j applies its input shares $(\mathsf{is}_{i,j})_{i\in I}$ from multiple messages M_i, so that they jointly compute a function f. Each node outputs an output share os_j of the result.*

$\mathsf{Combine}((\mathsf{os}_j)_{\forall j}) \to \mathsf{res}$**:** *This algorithm combines the output shares os_j from each node n_j to the function's result $\mathsf{res} \leftarrow f((M_i)_{i\in I})$ or fails.*

4 SYSTEM MODEL

This section gives a high-level introduction of our ARMOREDTWINS system, its actors, as well as their interactions and trust assumptions (cf. Figure 1).

Setup and Access Control. The *owner* has one or more *devices* that create a *digital twin* of a *physical object* at their *cloud* account. This owner uses their *management device* to initially set up the device (c.f. ①) and to intermittently manage access to the digital twin data collected at the cloud (c.f. ②). The owner is only sporadically online, at a time of their choosing.

Synchronization. One or more devices gather information about the physical object, which they frequently synchronize to digital twin in the cloud (c.f. ③). For example, a smart watch might contribute to building the digital twin of a patient, while manufacturing equipment might create a digital twin about itself (as physical object). Changes to the digital twin in the cloud can also propagate to devices (c.f. ④). Highly constrained devices may also rely on a *gateway* for connectivity and computation resources.

External Interaction. *Requesters* may wish to interact with the physical object (c.f. ⑤). This interaction is directed at the digital twin in the cloud, which forwards the requests at appropriate times to the devices that are associated with the physical object. The

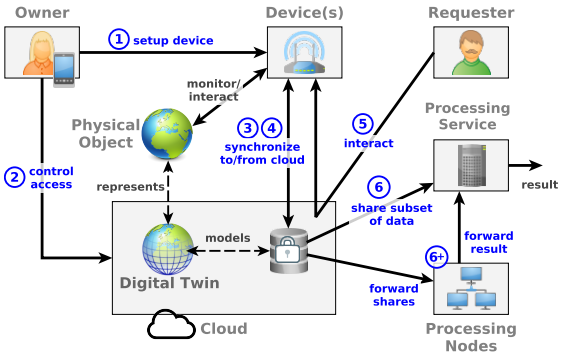

Figure 1: ARMOREDTWINS System Model

cloud filters requests (e.g., based on plain metadata or results of a processing service) and negotiates with the device on the delivery of the requests (e.g., deliver immediately or periodically in bulk, or let device fetch requests). The cloud routes (and re-encrypts) the requests to the appropriate set of devices. These requests may trigger the device to interact with the physical object, change its state, and/or respond.

Processing of Data. The digital representation of physical objects as up-to-date digital twins enables powerful computation, such as simulations of behavior (Chen et al., 2018; Qi and Tao, 2018), maintenance predictions (Kraft, 2016), and precision medicine (Liu et al., 2019). Given the owner's permission, the cloud forwards a subset of relevant digital twin data to a *processing service*, upon which computations can be performed (c.f. ⑥). For more sensitive data, the data may also be split across a network of *processing nodes* that engage in a secure multi-party computation protocol to compute a result for a processing service without ever learning the digital twin's data or the function's result (c.f. ⑥+). Finally, the processing service is able to act upon the result, e.g., block an external request, or notify the owner of predicted maintenance needs.

Trust. The owner trusts the other actors as follows: (1) The owner's device is trusted to correctly handle the data of the physical object and interact with it. (2) The cloud is honest-but-curious, i.e., tries to learn about the physical object's data, while correctly following the protocols, without acting actively malicious, e.g., by withholding data or requests, or by trying to inject malicious data. (3) The owner's management device is fully trusted, as it holds the main private keys. (4) Sufficiently trusted processing services are selected by the owner so that they may learn user-specified subsets of the digital twins' data. (5) The processing nodes are trusted to not collude with each other and to compute the function correctly, while refusing privacy-invasive functions.

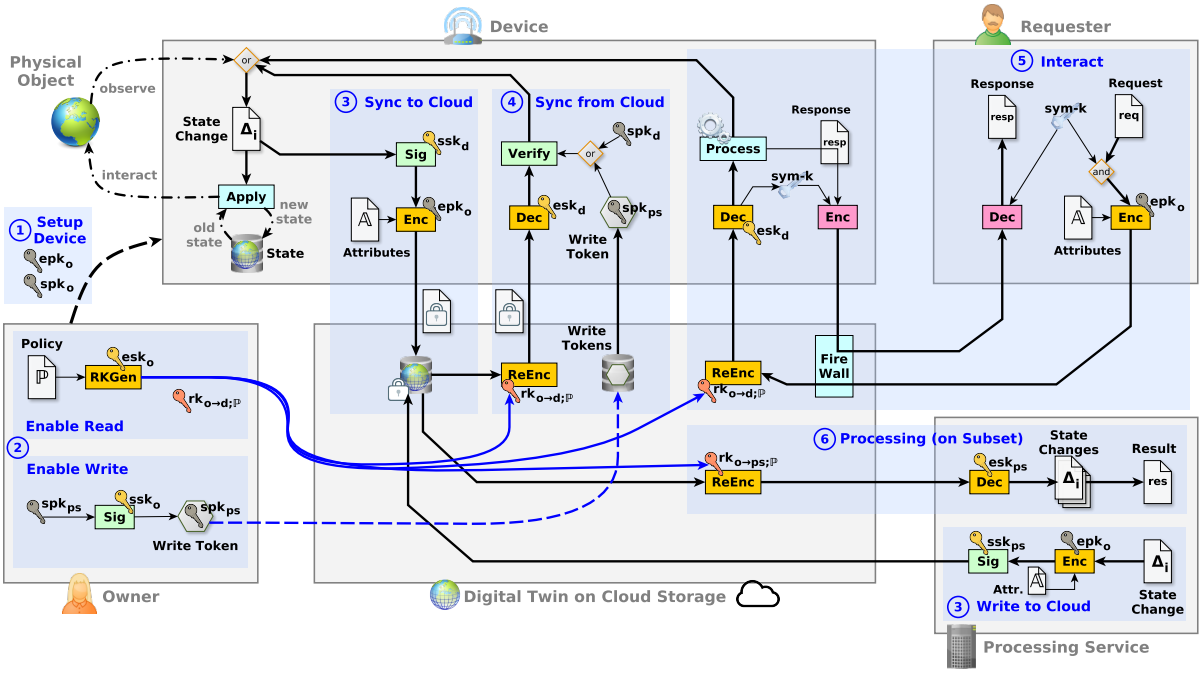

Figure 2: Setup, Access Control, Synchronization, Interaction, and Processing (on Subsets) in our ARMOREDTWINS System

5 CONCEPT

This section introduces our ARMOREDTWINS security architecture for digital twins as well as the main interactions between the involved actors. Initially, we present the main interactions, as illustrated in Figure 2 and specified in Protocol 1: ①-④ the general setup to securely synchronize the characteristics of a physical object to a digital twin in the cloud, ⑤ interactions triggered by external requesters towards the physical object via the digital twin and attached devices, and ⑥ processing of data from digital twins, which may belong to various owners. Next, as specified in Protocol 2, we elaborate on performing secure computations on the encrypted data of multiple digital twins without revealing the involved data (⑥+) and describe required extensions to previous processes (②+ and ③+). Finally, we review maintenance processes to adjust access permissions and recover in case of device loss (c.f. M1-M3).

5.1 Synchronization

As detailed in phases ① to ④ of Protocol 1, our ARMOREDTWINS employ key-policy conditional proxy re-encryption (KP-CPRE) to protect the synchronized data with end-to-end encryption, enforce fine-grained access control, and simplify the maintenance of access control rules. Generally, the devices record the physical object's characteristics and encrypt them for the owner before uploading them to the cloud. To later use the digital twin's data, a network of end-to-end encryption links between multiple data sources (device and requester) and data receivers (processing service and device) needs to be set up and maintained over time. With public key encryption, owners would need to explicitly configure each sender to encrypt the same data with various keys of multiple receivers and re-configure these senders upon future changes. Instead, our system uses KP-CPRE, where the device attaches a set of attributes $\mathbb{A}$ to the data in the encryption process, e.g., that describe the content. To enable read access for others, the owner generates re-encryption keys from their private key, the receiver's public key, and an access control policy $\mathbb{P}$. These re-encryption keys enable the cloud to transform ciphertext of the digital twin (encrypted for the owner) into ciphertext that can be decrypted by the selected recipient, given the key's policy matches the ciphertext's attributes, i.e., $\mathbb{P}(\mathbb{A}) = 1$. Therefore, owners can define sharing rules at a later point in time without having to re-configure the devices and requesters.

Synchronized Data. Data on the digital twin is popularly defined as a state model in related work (Eckhart and Ekelhart, 2018), where the final state is constructed by successively applying state changes. Such a state model can be applied rigidly in the form of a finite state machine, or interpreted loosely, where state changes are, e.g., attaching documents to an entity. Given this flexibility, we use a state

model for the digital twins' data in our system. As our system enables to share subsets of the state changes, it is beneficial if the state changes are also meaningful on their own, i.e., if a state change sufficiently describes an aspect of the state. Further, we assume that the state information carries metadata that identifies the digital twin to facilitate processing of data from multiple twins.

Authenticity of State Data. While some external entities may be authorized to write state changes, e.g., owners or processing services, it is necessary to prevent unauthorized actors from inject state data. Our system employs digital signatures to tackle this challenge. Each device, owner, and processing service generates their own signature key pairs. The device obviously accepts state data that is signed either by the device itself or by its owner, whose signature key was marked as trustworthy during setup. For authorized processing services, our system uses a dynamic delegation mechanism, where the owner issues *write tokens*. Such a write token states that the owner sufficiently trusts the processing service identified by its signature verification key to write to the digital twin's state, possibly along with further restrictions, e.g., a time period. A write token has to be presented alongside new state data that is signed by the processing service. Finally, the device verifies the signature on the state data and the write tokens against its trusted keys before accepting a state change. Alternatively or in addition, the database storing the state data could be modeled with distributed ledger technology (Dietz et al., 2019).

Attributes and Policies. Our system distinguishes between state data and request data, so that re-encryption keys cannot be reused for unintended purposes. For state data, the ciphertext's attributes contain `sync` as well as further attributes that the device derives from the content. For requests, the requester attaches `req` as well as attributes identifying the digital twin $\mathsf{id}_{\mathsf{dt}}$. Owners use these attributes in their re-encryption key policies, i.e., trees of AND/OR logic gates, to specify which receivers are authorized to decrypt state and/or request data.

Reduce Computation Effort. Computational efficiency is an important factor for low-power and battery-powered devices, as often found in IoT environments. Symmetric cryptography (e.g., AES) is generally considered to be more efficient than asymmetric encryption (e.g., KP-CPRE schemes). Additionally, many processor manufacturers have integrated hardware-support for AES, enabling very efficient use of this symmetric mechanism. Naturally, KP-CPRE is employed in a hybrid encryption setting, where KP-CPRE encrypts a newly generated symmetric key, which is then used to encrypt the actual content. Our system's computation effort for state synchronization can be reduced by re-using the symmetric content encryption key for a limited time period on data that would be encrypted for the same set of attributes. Consequently, the more expensive KP-CPRE operations to encrypt and decrypt the symmetric key only have to be performed once per time period (also called epoch).

5.2 External Interaction

Instead of communicating with a device directly, the requester interacts with the cloud-based digital twin, which forwards the requests to the appropriate devices at a suitable time, as detailed in ⑤ of Protocol 1. Initially, the requester needs to obtain connection information to reach the digital twin and public key material out-of-band, e.g., by scanning a QR code, receiving a wireless announcement, or querying discovery services. The requester uses KP-CPRE to encrypt the request for the owner of the digital twin, and includes the digital twin's identity $\mathsf{id}_{\mathsf{dt}}$ as well as the marker `req` as attributes. During setup or maintenance, the owner decided which device handles the digital twin by generating a re-encryption key. The cloud uses such re-encryption keys to route the request to the appropriate device. As multiple devices could handle different aspects of one digital twin, re-encryption keys with policies that also consider additional attributes may be used for further fine-grained routing. The cloud makes the re-encrypted request available to the device, either by pushing it directly or offering it to be pulled by the device. Finally, the device is able to decrypt the request with its private key and process the contents.

Request Filtering. Before re-encrypting requests, the cloud may apply additional filtering logic to protect the device and its resources, e.g., to minimize the impact of denial-of-sleep attacks by enforcing rate- and time-limits. Some filtering logic can be implemented based on metadata that is available in plain (e.g., number and time of requests per digital twin), while more elaborate logic depending on the request's content requires further processing (see Section 5.3).

Additional Request Content. Requests may also include information to protect further communication, e.g., symmetric session keys that allow for more efficient cryptographic mechanisms in subsequent interactions. Also, the requester may need to provide authentication credentials or go through a subsequent authentication process so that the device processes the requests (out of scope).

① Setup Device:

On device d, after receiving initial confiuration conf: [1] Extract and store from conf: initial state st, owner's public encryption key $\mathsf{epk_o}$, owner's signature verification key $\mathsf{spk_o}$, id of the linked digital twin $\mathsf{id_{dt}}$, and connectivity information for the owner's cloud account. [2] Generate an encryption key pair $(\mathsf{esk_d}, \mathsf{epk_d}) \leftarrow \mathsf{PRE.KeyGen}(1^\kappa)$ and [3] a signature key pair $(\mathsf{ssk_d}, \mathsf{spk_d}) \leftarrow \mathsf{SIG.KeyGen}(1^\kappa)$.

On owner o: [4] Run ② Control Access for device d with policy $\mathbb{P} \leftarrow \mathtt{sync} \vee (\mathtt{req} \wedge \mathsf{id_{dt}} \wedge \mathbb{P}_{\mathsf{req}})$, where $\mathbb{P}_{\mathsf{req}}$ may define further limitations.

② Control Access (toward entity x with policy $\mathbb{P}$)**:**

On owner o: [1] To enable sharing of data items that satisfy policy $\mathbb{P}$, generate a re-encryption key $\mathsf{rk}_{\mathsf{o}\to x;\mathbb{P}} \leftarrow \mathsf{PRE.RKGen}(\mathsf{esk_o}, \mathsf{epk}_x, \mathbb{P})$ and send it to cloud. [2] To grant write permissions for processing service, issue a write token $\mathsf{wt} \leftarrow (\mathsf{spk}_x, \mathsf{id_{dt}})$, $\sigma_{\mathsf{wt}} \leftarrow \mathsf{SIG.Sign}(\mathsf{ssk_o}, \mathsf{wt})$.

③ Synchronize to Cloud:

On device d, upon observing change Δ_{i+1} in the physical object: [1] Change the internal state $\mathsf{st}_{i+1} \leftarrow \mathsf{Apply}(\mathsf{st}_i, \Delta_{i+1})$, and [2] sign it as $\sigma_{\Delta_{i+1}} \leftarrow \mathsf{SIG.Sign}(\mathsf{ssk_d}, \Delta_{i+1})$, where $\mathsf{ssk_d}$ are replaced with $\mathsf{ssk_o}$ or $\mathsf{ssk_{ps}}$ if run by the owner o or an authorized processing service ps, respectively. [3] Set attributes $\mathbb{A} \leftarrow \mathtt{sync} \wedge \mathbb{A}_\Delta$, where $\mathbb{A}_\Delta$ is derived from the content of Δ_{i+1}, and [4] encrypt the state change as $C \leftarrow \mathsf{PRE.Enc}(\mathsf{epk_o}, \mathbb{A}, (\Delta_{i+1}, \sigma_{\Delta_{i+1}}))$. [5] Remember and re-use the SYM key and its PRE ciphertext for the current timeframe and $\mathbb{A}$, so that subsequently only SYM encryption is necessary.

On cloud, receiving C: [6] Store encrypted state change C.

④ Synchronize from Cloud:

Assumption: Owner o has performed ② Control Access for device d with a policy $\mathbb{P}$ on type sync.

On cloud: [1] Re-encrypt encrypted state change C for device d as $C' \leftarrow \mathsf{PRE.ReEnc}(\mathsf{rk}_{\mathsf{o}\to\mathsf{d}}, C)$, which applies only for SYM keys of timeframes, if not yet available.

On device d, receiving C': [2] Decrypt change $(\Delta_{i+1}, \sigma_{\Delta_{i+1}}) \leftarrow \mathsf{PRE.Dec}(\mathsf{esk_d}, C')$. If the SYM key for the timeframe is already available, only perform SYM decryption. [3] Verify that state can be accepted with $\mathsf{SIG.Verify}(\mathsf{spk}_x, \Delta_{i+1}, \sigma_{\Delta_{i+1}}) = 1$, where spk_x has to be in the device's trust store or is included in a valid write token $\mathsf{wt} = (\mathsf{spk}_x, \mathsf{id_{dt}})$ with $\mathsf{SIG.Verify}(\mathsf{spk_o}, \mathsf{wt}, \sigma_{\mathsf{wt}}) = 1$. [4] Change the internal state to $\mathsf{st}_{i+1} \leftarrow \mathsf{Apply}(\mathsf{st}_i, \Delta_{i+1})$.

⑤ Interact:

Assumption: Owner o has performed ② Control Access for the device d with a policy $\mathbb{P}$ on type req and $\mathsf{id_{dt}}$.

On requester: [1] Obtain $\mathsf{id_{dt}}$ of the twin and key of its owner $\mathsf{epk_o}$. [2] Encrypt and send request req as $C_{\mathsf{req}} \leftarrow \mathsf{PRE.Enc}(\mathsf{epk_o}, \mathbb{A}, \mathsf{req})$, where $\mathbb{A}$ includes req, $\mathsf{id_{dt}}$ and possibly further attributes, and req or PRE defines the encryption key for the response.

On cloud, receiving C_{req}: [3] Optionally, analyze if the request should be forwarded (e.g., via processing service or rate limiting). [4] Route the request to the digital twin's devices by $C'_{\mathsf{req}} \leftarrow \mathsf{PRE.ReEnc}(\mathsf{rk}_{\mathsf{o}\to\mathsf{d};\mathbb{P}}, C_{\mathsf{req}})$, where the policy $\mathbb{P}$ contains req and $\mathsf{id_{dt}}$.

On device d, receiving C'_{req}: [5] Decrypt the request as $\mathsf{req} \leftarrow \mathsf{PRE.Dec}(\mathsf{esk_d}, C'_{\mathsf{req}})$ and [6] process it $(\mathsf{resp}, \Delta_{i+1}) \leftarrow \mathsf{Process}(\mathsf{req}, \mathsf{st})$. [7] If this resulted in a state change Δ_{i+1}, run ③ Synchronize to Cloud. [8] Encrypt and return the response resp with key material from the request, e.g., obtain the SYM key from the request or re-use the key obtained via PRE.

⑥ Processing (on Subset):

Assumption: Owner o has perfurmed ② Control Access for processing service ps with a policy $\mathbb{P}$ on type sync for data with relevant attributes.

On cloud: [1] Identify the set of relevant state changes I as $(C_i)_{\forall i \in I}$ and [2] re-encrypt them for the processing service yielding $\forall i \in I : C'_i \leftarrow \mathsf{PRE.ReEnc}(\mathsf{rk}_{\mathsf{o}\to\mathsf{ps}}, C_i)$. Re-encryption applies only to SYM keys of related timeframes.

On processing service ps, receiving $(C'_i)_{\forall i \in I}$ from one or multiple owners: [3] Decrypt the set of state changes $\forall i \in I : (\Delta_{i+1}, \sigma_{\Delta_{i+1}}) \leftarrow \mathsf{PRE.Dec}(\mathsf{esk_{ps}}, C'_i)$, where decrypted SYM keys of same timeframes can be re-used. [4] Optionally, verify their signatures by $\mathsf{SIG.Verify}(\mathsf{spk_o}, \Delta_{i+1}, \sigma_{\Delta_{i+1}}) = 1$. [5] Process the data.

M1 Change Access of Processing Service:

On owner: [1] To remove access rights of processing service, instruct the cloud to delete $\mathsf{rk}_{\mathsf{o}\to\mathsf{ps};\mathbb{P}}$. [2] To grant (reduced/extended) rights, generate and upload a new $\mathsf{rk}_{\mathsf{o}\to\mathsf{ps};\mathbb{P}}$ with policy $\mathbb{P}$. [3] To revoke write access, instruct the cloud to remove/revoke write the token wt.

M2 Replace Processing Node:

On owner: [1] Remove access of old node n_j by instructing cloud to delete $\mathsf{rk}_{\mathsf{o}\to\mathsf{n}_j;\mathbb{P}}$ and [2] grant access to new node n'_j by generating and uploading a new $\mathsf{rk}_{\mathsf{o}\to\mathsf{n}'_j;\mathbb{P}}$ with policy $\mathbb{P}$.

M3 Replace Device:

On owner and new device d′: [1] Remove access of the old device d by instructing cloud to delete $\mathsf{rk}_{\mathsf{o}\to\mathsf{d};\mathbb{P}}$. [2] Run ① Setup Device on new device d′ and same $\mathsf{id_{dt}}$. [3] To load previous state, mark old device's signature keys $\mathsf{spk_d}$ as trusted or issue a write token for it.

On cloud and new device d′: [4] Run ④ Synchronize from Cloud to load state of the old device.

Protocol 1: Main Phases of our ARMOREDTWINS System

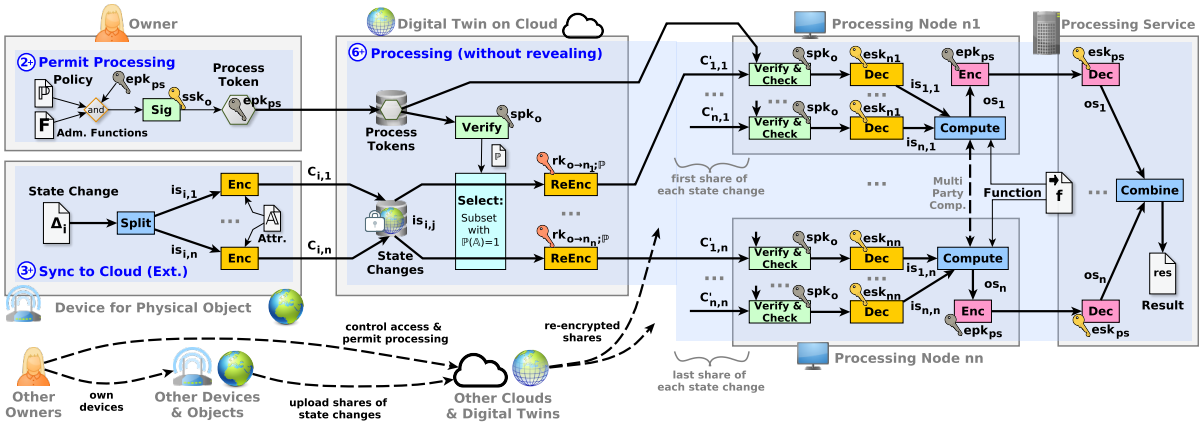

Figure 3: Processing of Digital Twin Data with Secure Multi-Party Computation

5.3 Data Processing

In the concept of digital twins, up-to-date data about multiple digital twins of various owners is accumulated in the cloud, which is a key enabler to perform powerful computation. These computations include running simulations, makeing predictions, developing optimizations, finding errors or defects (e.g., deviation from predicted behavior), integrating further services (e.g., request filtering), etc. Unfortunately, by encrypting the digital twins' data to prevent the cloud from learning the plaintext, the cloud is also no longer able to perform these processing tasks. In a naive approach, owners download and decrypt the data of their digital twins and subsequently performs the processing locally, e.g., on their device or a more powerful computing system. However, this approach falls flat once data from other owners is involved. Our ARMOREDTWINS system supports two approaches that differ in the level of provided privacy, which has an impact on the required computation cost.

⑥ **Processing on Subsets.** In the first approach, owners share subsets of digital twin data with processing services that are sufficiently trusted or where the owner made a conscious decision to exchange privacy for other benefits (e.g., convenience or money). For example, a car owner sells a subset of the car's digital twin data that has little privacy impact to the manufacturer, who uses it to optimize future models. Such processing services could also act as intermediaries that may learn the subset to compute a function before only forwarding the result to a less-trusted consumer.

In this approach, the owners use KP-CPRE to share subsets of their digital twins' data with processing services. Based on the ciphertexts' attributes and the re-encryption key's policy, the owners are able to selectively disclose only a subset of relevant state information. The processing service aggregates data about digital twins from multiple owners, decrypts them, and processes the subsets in plain. To ensure authenticity, the processing service could also verify the state's signatures. This approach hands control to the owners and enables them to make conscious sharing decisions, but requires that the shared subset is not considered too sensitive.

⑥+ **Processing with MPC.** The second approach can be applied for sensitive data that owners would contribute towards the result of a computation, but where the input data must not be directly exposed. Examples include participating in a health-related simulation without fearing a negative impact on insurance rates or estimating whether a request is malicious without revealing the actual content of the request to a party to keep honest requests confidential.

This approach combines multi-party computation (MPC) with end-to-end data sharing via KP-CPRE, as illustrated in Figure 3 and further specified in Protocol 2. Initially, devices prepare their state data for MPC by splitting it into multiple shares, one for each processing node. The number of the share is then assigned as an attribute while encrypting the share (e.g., `node-`j). As in the first approach, the owner generates re-encryption keys, now also defining which share number may be re-encrypted for which processing node. After the shares are re-encrypted and decrypted, the processing nodes engage with each other in an MPC protocol to collaboratively compute a function. To also ensure the shares' authenticity, they could be signed by the devices and verified on the node before computation. Each processing node generates an output share, which is securely forwarded to the final receiver, which combines the output share and alone learns the function's result. The processing nodes learn neither the state data from their input shares nor the result from the output shares, as long as not more than a threshold of nodes collude.

Processing Authorization. Of course, our system needs to ensure that the computation function does not violate the owners' privacy and that only authorized receivers learn the computation result. The re-encryption keys explicitly specify which processing node may obtain which shares of which data subset per owner. Additionally, owners need to be able to define which type of function may be computed and with which receiver the processing nodes may share the result. In our system, owners issue signed *process tokens* that specify the data set, the admissible functions, and the public key of the authorized receiver. Each processing node verifies such process tokens before participating in the computation.

Selecting Processing Nodes. KP-CPRE enables owners to change the set of processing nodes at a later point, e.g., if a processing node is not sufficiently trusted anymore, without having an impact on the previously created cipertexts or the way devices encrypt data. However, two limitations of MPC have to be accommodated: Firstly, a processing node must not learn different shares of the same state data item, as the node could then learn about the state. Therefore, each owner has to ensure that no two re-encryption keys map from different share numbers to the same processing node. Alternatively, an interesting open question would be to transform a set of encrypted shares into a different set for the same secret, e.g., using homomorphic properties of PRE schemes. Secondly, per computation, the shares have to be processed by the same set of processing nodes. While this is easy to achieve for data from one owner, if the data is aggregated from multiple owners, they have to agree on one set of processing nodes. The receiver or a marketplace may facilitate the negotiation process.

5.4 Maintenance and Recovery

A digital twin environment is rarely static, as e.g., actors interacting with the system change and new processing service may offer valuable services, while other actors lose the owner's trust and devices break. Therefore, our system needs to be dynamic as well, i.e., it has to be able to accommodate changes in the involved actors and trust relationships without extensive effort for the owner, as specified in Protocol 1.

(M1) **Change Access of Processing Service.** Owners can revoke access by instructing the cloud to delete re-encryption keys towards a receiver. Without the cloud's involvement (i.e., re-encryption), corrupted receivers cannot learn data from stolen ciphertext. By combining removal and generation of re-encryption keys, owners can extend or reduce existing access rights by adjusting the strength of the policy.

(2+) Control Access (extends (2)):

On owner o: [1] For each processing node n_j, run (2) Control Access for processing node n_j with a policy $\mathbb{P} \leftarrow \texttt{sync} \wedge \texttt{node-}j \wedge \mathbb{P}_r$, where $\mathbb{P}_r$ selects relevant items. [2] Issue a process token for processing service ps as $\mathsf{pt} \leftarrow (\mathbb{P}, \mathsf{epk}_{\mathsf{ps}}, \mathcal{F})$, $\sigma_{\mathsf{pt}} \leftarrow \mathsf{SIG.Sign}(\mathsf{ssk}_{\mathsf{o}}, \mathsf{pt})$, where $\mathcal{F}$ defines admissible functions.

(3+) Synchronize to Cloud (extends (3)):

On device d: [1] Split state changes into shares $(\mathsf{is}_{i,j})_{j\in[n]} \leftarrow \mathsf{MPC.Split}(\Delta_{i+1}, t, n)$, where n is the number of processing nodes n_j and t is the threshold. [2] Encrypt the input shares for the respective n_j by $C_{i,j} \leftarrow \mathsf{PRE.Enc}(\mathsf{epk}_{\mathsf{o}}, \mathbb{A}, \mathsf{is}_{i,j})$, where $\mathbb{A}$ contains $\texttt{sync}$, $\texttt{node-}j$, and further attributes derived from the content. [3] Upload encrypted shares $(C_{i,j})_{j\in[n]}$ to the cloud.

(6+) Processing (with MPC):

On cloud, for participating owners o_k: [1] Identify set of relevant changes I yielding shares $(\mathsf{is}_{i,j})_{\forall j\in[n], i\in I}$, which satisfy $\mathbb{P}$ of owner's pt. [2] For each state change $i \in I$, re-encrypt the shares for the respective n_j by $C'_{i,j} \leftarrow \mathsf{PRE.ReEnc}(\mathsf{rk}_{\mathsf{o}_k \to \mathsf{n}_j; \mathbb{P}}, C_{i,j})$. [3] For each node n_j, forward the respective set $(C'_{i,j})_{i\in I}$ with matching j.

On each processing node n_j, receiving $(C'_{i,j})_{\forall i\in I}$ from various owners: [4] For each data set from different digital twins and owners, check there is a process token $\mathsf{pt} = (\mathbb{P}, \mathsf{epk}_{\mathsf{ps}}, \mathcal{F})$, $\mathsf{SIG.Verify}(\mathsf{spk}_{\mathsf{o}}, \mathsf{pt}, \sigma_{\mathsf{pt}}) = 1$ for receiver ps and that the function f is admissible by $\mathcal{F}$. [5] Decrypt the shares $\mathsf{is}_{i,j} \leftarrow \mathsf{PRE.Dec}(\mathsf{esk}_{\mathsf{n}_j}, C'_{i,j})$ and [6] compute $\mathsf{os}_j \leftarrow \mathsf{MPC.Compute}((\mathsf{is}_{i,j})_{i\in I}, (\mathsf{n}_k)_{\forall k\neq j})$ with the other nodes. [7] Encrypt the output share os_j as $C_{\mathsf{os}_j} \leftarrow \mathsf{PKE.Enc}(\mathsf{epk}_{\mathsf{ps}}, \mathsf{os}_j)$ and send it to receiver ps.

On processing service, given $(C_{\mathsf{os}_j})_{\forall j}$: [8] Decrypt the output shares $\mathsf{os}_j \leftarrow \mathsf{PKE.Dec}(\mathsf{esk}_{\mathsf{ps}}, C_{\mathsf{os}_j})$ and [9] combine them to the result $\mathsf{res} \leftarrow \mathsf{MPC.Combine}((\mathsf{os}_j)_{\forall j})$.

Protocol 2: Processing with Sec. Multi-Party Computation

(M2) **Replace Processing Node.** The owner may apply the above approach to change the set of processing nodes, i.e., by removing the re-encryption key towards an old processing node and generate a new key for the replacement processing node. However, two conditions need to be observed when selecting the new processing node: Firstly, no processing node must get access to different shares of the same state data item. Secondly, to process data from multiple owners, they need to agree on the same set of nodes.

(M3) **Replace Device.** Due to various circumstances, devices that maintain a digital twin of a physical object may need to be replaced, e.g., if the device breaks, gets vandalized, or is stolen. Initially, the owner revokes old device's access rights by instructing the cloud to remove the corresponding re-encryption key. Next, the owner sets up a replacement

device with minimal configuration effort, e.g., without the need to perform complex key management and distribution to configure the device for various data sharing connections. Instead, the owner generates a re-encryption key towards the replacement device with an appropriate policy (including the digital twin's id $\mathsf{id}_{\mathsf{dt}}$), so that the replacement device is able to recover the state of the old device via our synchronization mechanism. Also, such a re-encryption key enables to seamlessly route external requests to the new device. As our system has been designed so that the old device's keys do not have to be extracted, backed up and re-used, these keys can be provisioned in a hardware-protected module, which protects them even if the device was stolen.

Recovery of Management Device. While the management device is a key enabler for the owner's control, it may also become unavailable for various reasons (e.g., it breaks or is stolen). The only crucial data on the management device is the owner's private key, for which a recovery strategy has to be integrated. Building on related research various approaches can be integrated: (A) The private key is stored on a flash drive or printed as QR code that is kept in a secure location. (B) Password-based encryption is applied to the key before uploading it to a storage server (e.g., our cloud service). (C) (Password-protected) secret sharing techniques split the owner's key into shares, which are distributed to a set of the owner's device, trusted users, to a hierarchy of servers (Abdalla et al., 2016; Hörandner and Rabensteiner, 2019).

6 EVALUATION

We implemented and evaluated our concept of ARMOREDTWINS to underline its practicability. Section 6.1 focuses on the general system with processing on data subsets shared via KP-CPRE. Section 6.2 introduces privacy-preserving contact tracing as an example use case for MPC, where highly sensitive data is processed without revealing them to the nodes.

6.1 General System

This section focuses on KP-CPRE, as AES (used as SYM), ECDSA (used as SIG), and ECIES (used as PKE) are well-known mechanisms with well-established performance characteristics. The collected times allow to derive the overall compuation effort for the individual phases of our system.

PRE **Implementation.** We instantiate the CCA-secure KP-CPRE scheme of Zhao et al. (2010) with SHA-256 as hash function. For all used schemes, we have selected parameters to achieve 128bit security according to recommendations from National Institute of Standards & Technology (2016). Groups for pairing-based curves are chosen following recent recommendations by Menezes et al. (2016) and Barbulescu and Duquesne (2019). Our implementation builds upon the RELIC toolkit (Aranha and Gouvêa, 2021) and currently only uses a single thread, which leaves room for future performance improvements.

Table 1: Execution Times (in ms) of our Implementation for KP-CPRE. The attribute set $\mathbb{A}$ consists of the leaf values $L(\mathbb{P})$ of the policy $\mathbb{P}$, which is a binary tree of AND gates.

$\lvert\mathbb{P}\rvert$	$\lvert\mathbb{A}\rvert$	KeyGen	RKGen	Enc[1]	ReEnc	Dec[2]
PC (AMD Ryzen 5600X)						
3	2	0.39	2.04	2.62	4.22	2.36
7	4	0.39	3.54	3.00	4.86	2.36
15	8	0.39	6.52	3.75	6.13	2.36
Phone (OnePlus 6T)						
3	2	6.93	37.35	46.96	76.85	41.32
7	4	6.93	65.34	54.79	90.22	41.29
15	8	6.93	121.46	70.11	117.06	41.38
IoT Device (Raspberry Pi 4B)						
3	2	12.81	70.27	88.41	143.72	78.50
7	4	12.81	123.03	102.48	167.70	78.36
15	8	12.75	228.59	131.16	215.82	78.18

Results. Table 1 shows average execution times of the PRE algorithms (in milliseconds) for various sizes of policies and attribute sets on three platforms: a PC, a mobile phone, and an IoT device. The presented numbers are an average of 100 runs, while the standard deviation was <1.8% of each measurement. Initially, we generate key pairs for two participants, as well as a re-encryption key between them, with a policy $\mathbb{P}$. Next, the benchmark takes a 128bit AES key as payload, encrypts the payload with an attribute set $\mathbb{A}$ for the first participant, re-encrypts it with the re-encryption key for the second participant, and finally decrypts the payload with the second participant's private key. The policy $\mathbb{P}$ is shaped as a binary tree of AND gates. OR gates only have a negligible impact on the execution time, as our implementation finds a minimal but sufficient match between ciphertext attributes and policy nodes. The attribute set $\mathbb{A}$ consists of the leaves $L(\mathbb{P})$ of the policy $\mathbb{P}$, so that $\mathbb{P}(\mathbb{A}) = 1$.

In our system, KP-CPRE encryption and decryption is performed once per epoch for the same attribute set and requires <132*ms* and <42*ms* even for large policies on a Raspberry PI 4B. Re-encryption is performed on a powerful cloud to share items, once per data set with the same attributes of an owner also per epoch, which takes <7*ms* on our benchmark PC.

6.2 Privacy-Preserving Processing

As the performance of our privacy-preserving processing is dictated by the function that should be evaluated, we cannot present generic numbers. Instead, we use privacy-preserving contact tracing as an example, which is based on location data that is collected by a device as part of a person's digital twin. Such location data (over time) is sensitive information, as it not only discloses the living and work address but can also reveal movement patterns, behavior, relationships (proximity to others), or even health status (stay at a hospital). However, such sensitive information can be very valuable. Our use case traces the contacts of one reference person (e.g., infected patient) with n other people by comparing the proximity of their paths and calculating how often they were too close. Building on MPC, this computation is possible without the processing nodes learning the location of involved people, which people were close to others, or the number of contacts. Thus, users may contribute data to computations without sacrificing their privacy.

MPC **Instantiation.** We build upon SCALE-MAMBA as MPC framework (Aly et al., 2021), due to its accessibility, which enables to define general-purpose functions in a python-like programming language. The benchmarks use SCALE-MAMBA's default configuration (statistical security parameter of 40) and Shamir's secret sharing with three nodes (#nodes $= 3$) where one may be corrupted ($t = 1$).

Use Case Parameters. Our benchmark aims to capture the computational effort in relation to the number of users (#users). Therefore, we fix the other parameters as follows: Each user uses their phone to collect location data over time and synchronize it with the user's digital twin at the cloud (#devices/user $= 1$ and #devices $=$ #users). They each record a series of 50 location points that consist of an x and y coordinate (#items/device $= 50 \cdot 2 = 100$). The individual coordinates are split into one share per processing node (#shares/device $=$ #items/device $\cdot$ #nodes $=$ 300). We further assume that each location point of a user is recorded in a different timeframe (#epochs $=$ 50). Each PRE ciphertexts is associated with three attributes $\mathbb{A} = (\texttt{sync}, \texttt{node-}j, \texttt{location})$, which are used in a conjunction to form the policy $\mathbb{P}$.

Use Case Process. The user's phones (as devices) periodically collect the current location and synchronizes it with the user's digital twin at a cloud (3+). Users also use their phones (as management devices) to authorize the processing nodes to perform computations on their location information (2+). The cloud forwards the designated shares to the individual processing nodes, which compute the function in an MPC protocol and output the shares of the result to the receiver (6+). The cloud, processing node, and receiver are simulated on a desktop PC. This PC executes all three processing nodes in parallel as Docker containers, with 30ms round-trip time between them.

Table 2: Execution Times for Use Case (in Milliseconds)

2+ Control Access: per user, on phone (OnePlus 6T)		
PRE.RKGen	53.99	× #devices/user × #nodes
SIG.Sign	1.44	× 1
UseCase-Σ =	**163.39**	*(per user)*
3+ Sync. to Cloud: per device, on phone (OnePlus 6T)		
MPC.Split	0.02	× #items/device
AES.Enc	<0.01	× #shares/device
PRE.Enc	51.13	× #epochs × #nodes
UseCase-Σ =	**7672.41**	*(per user, over epochs)*
6+ Processing: cumulated, on PC (AMD Ryzen 5600X)		
PRE.ReEnc	4.52	× #devs. × #epochs × #nodes
SIG.Verify	0.19	× #users (on each node)
PRE.Dec	2.35	× #devices × #epochs (on each node)
AES.Dec	<0.01	× #shares (on each node)
MPC.Compute	6530.58	× #users +12433.61 (on each node)
PKE.Enc	0.18	× 1 (on each node)
PKE.Dec	0.12	× #nodes
MPC.Combine	0.08	× 1
UseCase-Σ =	**8242.24**	*(per user) + 12433.61 (const.)*

Results. Table 2 presents time measurements for the individual phases and their cryptographic operations. While these measurements of the cryptographic operations are generic, i.e., can be applied for various parameters, we also present sums according to the parameters chosen for our use case. Our results show that 2+ Control Access and 3+ Synchronize to Cloud require little computational effort on the phone, $<200ms$ (once) and $<8s$ (over a longer timespan), respectively, and therefore also draw little power. The privacy benefits of 6+ Processing (with MPC) obviously have a computational cost but enables processing on sensitive data that might not be available otherwise. We have measured the MPC computation on a PC for various numbers of users (18 samples up to 100 users). The computation time grows linearly with $6.53s \cdot$ #users $+ 12.43s$ and 95% confidence intervals of $[6.40, 6.65]$ and $[8.39, 16.48]$. Adding the systems' other cryptographic mechanisms, we have $8.24s$ per user with a constant initialization offset of $12.43s$. To obtain the computational cost, the benchmarks measured single-threaded implementations. Performance improvements could be achieved with parallelization as well as by replacing the general-purpose and accessible SCALE-MAMBA framework with other MPC protocols, e.g., that are optimized for a given use case.

7 CONCLUSION

In this work, we proposed ARMOREDTWINS: a secure digital twin system that protects the data of digital twins, so that it can also be employed for sensitive information. We build on KP-CPRE for flexible, fine-grained, end-to-end encryption, which enables sharing of data with its owner and other parties to review the data or process it, e.g., in simulations. Furthermore, we integrated MPC to enable processing of sensitive data that owners would not expose even partially. Our system allows to compute functions on shares of digital twin data so that neither the cloud nor the processing nodes learn the input or result, but only the designated receiver gets the outcome. Also, the system offers protected interaction with external requesters, processes to manage changing trust relationships, and strategies to recover from device and key loss. Benchmarks show the feasibility and practicability of our ARMOREDTWINS system. We evaluated the MPC integration with privacy-preserving contact tracing as an example use case, which scales linearly with about 8.2*s* per user and 50 uploaded location points each.

REFERENCES

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.

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*. https://homes.esat.kuleuven.be/~nsmart/SCALE/Documentation-SCALE.pdf. Accessed: 2021-02-12.

Aranha, D. F. and Gouvêa, C. P. L. (2021). *RELIC is an Efficient LIbrary for Cryptography*. https://github.com/relic-toolkit/relic. Accessed: 2021-02-12.

Ateniese, G., Fu, K., Green, M., and Hohenberger, S. (2006). "Improved proxy re-encryption schemes with applications to secure distributed storage". *ACM Trans. Inf. Syst. Secur.*, 9(1), pp. 1–30.

Barbulescu, R. and Duquesne, S. (2019). "Updating Key Size Estimations for Pairings". *J. Cryptol.*, 32(4), pp. 1298–1336.

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.

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

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.

Chen, X., Kang, E., Shiraishi, S., Preciado, V. M., and Jiang, Z. (2018). "Digital Behavioral Twins for Safe Connected Cars". In: *MoDELS*. ACM, pp. 144–153.

Dietz, M., Putz, B., and Pernul, G. (2019). "A Distributed Ledger Approach to Digital Twin Secure Data Sharing". In: *DBSec*. Vol. 11559. LNCS. Springer, pp. 281–300.

Eckhart, M. and Ekelhart, A. (2018). "A Specification-based State Replication Approach for Digital Twins". In: *CPS-SPC@CCS*. ACM, pp. 36–47.

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

Gehrmann, C. and Gunnarsson, M. (2020). "A Digital Twin Based Industrial Automation and Control System Security Architecture". *IEEE Trans. Ind. Informatics*, 16(1), pp. 669–680.

Hörandner, F. and Rabensteiner, C. (2019). "Horcruxes for Everyone - A Framework for Key-Loss Recovery by Splitting Trust". In: *TrustCom/BigDataSE*. IEEE, pp. 50–57.

Huang, S., Wang, G., Yan, Y., and Fang, X. (2020). "Blockchain-based data management for digital twin of product". *Journal of Manufacturing Systems*, 54, pp. 361–371.

Kern, A. and Anderl, R. (2020). "Using Digital Twin Data for the Attribute-Based Usage Control of Value-Added Networks". In: *SDS*. IEEE, pp. 29–36.

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*.

Liu, Y., Zhang, L., Yang, Y., Zhou, L., Ren, L., Wang, F., Liu, R., Pang, Z., and Deen, M. J. (2019). "A Novel Cloud-Based Framework for the Elderly Healthcare Services Using Digital Twin". *IEEE Access*, 7, pp. 49088–49101.

Menezes, A., Sarkar, P., and Singh, S. (2016). "Challenges with Assessing the Impact of NFS Advances on the Security of Pairing-Based Cryptography". In: *Mycrypt*. Vol. 10311. LNCS. Springer, pp. 83–108.

National Institute of Standards & Technology (2016). *SP 800-57. Recommendation for Key Management, Part 1: General (Rev 4)*. Tech. rep. NIST.

Putz, B., Dietz, M., Empl, P., and Pernul, G. (2021). "EtherTwin: Blockchain-based Secure Digital Twin Information Management". *Inf. Process. Manag.*, 58(1), p. 102425.

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.

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

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