



Deposited via The University of Sheffield.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/210384/>

Version: Published Version

---

**Article:**

Patel, C., Pasikhani, A., Gope, P. et al. (2024) User-empowered secure privacy-preserving authentication scheme for Digital Twin. *Computers & Security*, 140. 103793. ISSN: 0167-4048

<https://doi.org/10.1016/j.cose.2024.103793>

---

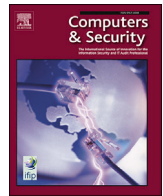
**Reuse**

This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here:

<https://creativecommons.org/licenses/>

**Takedown**

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing [eprints@whiterose.ac.uk](mailto:eprints@whiterose.ac.uk) including the URL of the record and the reason for the withdrawal request.



# User-empowered secure privacy-preserving authentication scheme for Digital Twin

Chintan Patel, Aryan Pasikhani, Prosanta Gope, John Clark

University of Sheffield, Sheffield, S14DP, South Yorkshire, United Kingdom

## ARTICLE INFO

### Keywords:

Auditability  
Digital Twin  
Verifiable credentials  
User revocation  
Usability

## ABSTRACT

Digital Twin (DT) is a revolutionary technology changing how a smart manufacturing industry carries out its day-to-day activities. DT can provide numerous advantages such as real-time synchronised functioning, monitoring and data analysis. However, security and privacy issues in DT have not been thoroughly investigated. This article proposes a user-empowerment-based privacy-preserving authentication protocol for a cloud-based Digital Twin using a Decentralised Identifier (DID) and Verifiable Credential (VC). Here, user empowerment provides full control to users over their identities, and with the help of VC, users can prove their authenticity and preserve their privacy. Although DID has emerged as a promising technology for introducing user empowerment, it suffers from some fundamental problems such as *usability* and *auditability*. Here we address all these issues and propose a user-revocation-enabled security solution for the DT. A security analysis of the proposed scheme shows that it is secured against significant security threats. With the help of performance analysis, we prove that the proposed work effectively ensures security and privacy in DT.

## 1. Introduction

Digital Twin (DT) is a revolutionary technology that is enhancing the way the manufacturing industry operates. In 2012, the National Aeronautics and Space Administration (NASA) introduced the DT [1]. It is a digital model of its physical manufacturing system counterpart running over the manufacturing plant and assures synchronised functioning of the Product Life Cycle (PLC) [2]. The DT plays a pivotal role in Industry 4.0, enabling a shift from real-time physical monitoring to analytical digital monitoring. DT plays a key role in cost reduction and enhancing supply chain operations in the manufacturing industry [3]. A DT can significantly impact the manufacturing process, asset management, performance analysis, real-time configuration management, and run-time simulation modelling. There are numerous applications of DT, such as product failure prediction, real-time monitoring, energy consumption monitoring, raw material usage and wastage analysis, real-time feedback and so on. Global leaders like Amazon, IBM, Microsoft, etc., have invested significantly in DT and related solutions. The market size of DT is predicted to grow from USD 6.9 bn in 2022 to USD 73.5 bn by 2027 [4].

Digital Twin is considered a completely digitised aspect of the Physical Twin. Everything related to the synchronisation process is done automatically [3]. In this paper, we have considered the same and

provided a solution to the highly challenging security problem related to data access and command execution. In DT, data is collected from the field sensors, cyber-physical system life cycles, and domain-related knowledge and uploaded to the cloud. At the same time, real-time feedback is provided based on generated knowledge from the received data. In DT, real-time synchronisation between the physical and virtual entities is very important, creating an opportunity for the attacker [2]. An attack on the virtual part may directly affect the “outcome” of a related physical part and may lead to the Garbage In Garbage Out (GIGO) problem, i.e. a general degradation of operating integrity. So, securing the DT data access and command execution is a key requirement for a successful implementation of the DT. Any large DT-based application (such as a manufacturing system) involves many users with several critical roles. In this regard, a user needs to prove his/her identity. Existing identifier solutions have multiple problems, such as a different identifier for each service, lack of interoperability and user empowerment (allowing users to manage their identities). In this context, a Decentralised Identifier (DID) can play a vital role in ensuring a privacy-preserving user-empowered solution [5].

DID provides self-sovereignty to users over their identity. It allows users to create, register and manage their identities by themselves. Here, the user registers their DID with their public key and other meta-data over a distributed ledger, and anyone with the DID of the user can

E-mail address: [p.gope@sheffield.ac.uk](mailto:p.gope@sheffield.ac.uk) (P. Gope).

<https://doi.org/10.1016/j.cose.2024.103793>

Received 11 April 2023; Received in revised form 10 January 2024; Accepted 27 February 2024

Available online 4 March 2024

0167-4048/© 2024 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

access user metadata. For example, with the help of Bob's DID, Alice can verify Bob's signature and also encrypt a message to be sent to Bob. Any trusted entity of the system provides verifiable credentials to the user and uses those verifiable credentials to prove their membership in the system. The DID technology can be key in designing security solutions for DT and IoT-based industrial applications. DID technology provides both *usability*, ensuring a user can access the system even when its private key is lost, and *auditability*, allowing the user to track misuse of its DID.

On the other hand, an immutable distributed ledger-based blockchain technology plays a key role in developing different security solutions for almost all industries. Blockchain provides numerous advantages such as transparency, decentralisation, security and privacy, and individual control of data [6]. Blockchain can improve security and trust in the industrial life cycle. According to the National Institute of Standards and Technology (NIST), a blockchain network can be either (1) a permissioned blockchain; or (2) a permissionless blockchain. In a permissioned blockchain, only authorised users can be part of it and generate, register and update the blocks. In a permissioned blockchain, an organisation's authorised entity can restrict access to individuals based on roles and attributes. In a permissionless blockchain, any individual can create and read the block over the distributed ledger and does not require permission from any authorised entity [7].

Overall, this paper proposes a permissioned blockchain-enabled DID-based solution for secure data access and command execution. We provide a solution for some of the unsolved fundamental problems of DID (i.e. *usability*, *auditability*). We consider a system model with *user space*, *cloud space*, *edge space* and *physical space*. The user space comprises all system users associated with the organisation. The edge space performs data collection and initial data processing and forwards those data to the cloud, where advanced data processing and service management entities run. This paper presents a solution for secure data access from the physical twin and command execution over the physical twin. In our system, we identified five fundamental entities: *authentication server* and *authorisation server* running over a cloud space, together with *command engine*, *system admin*, and *database server(s)*, running over edge spaces. As a result, the user can securely access the data from the database server and execute commands over twin through the command engine with some advanced properties such as enhanced *usability*, *auditability* and *user revocation*.

### 1.1. Related work

Since 2018, numerous system models have been presented for DT. In 2019, Tao et al. [8] presented a five dimension system model for the DT with the *physical entity*, *virtual entity*, *connection entity*, *data entity* and *service entity*. They highlighted three types of communications in DT: physical to physical, virtual to physical and virtual to virtual. However, we observed that in their work, Tao et al. completely neglected the *user entity*. The user entity plays a vital role in DT operation, from requiring real-time data access to run time command executions over DT. Unfortunately, there is negligible discussion about how these interactions will happen and what are the related security challenges. The DT provides real-time data access from the physical space to the virtual space and run-time command execution (a.k.a. run-time feedback) over the physical space from the virtual space. However, negligible work addresses appropriate problems such as how the system user will access twin data securely or how the system user can securely execute commands over the DT.

The Decentralised Identifier (DID) provides self-sovereignty to a user over their identity. It is a digital identity that is decentralised and verifiable among entities without any central authority. Since the inception of DID, numerous researchers have explored its usability in different applications. In 2019, Kortessniemi et al. [9] presented a solution for enhancing the privacy of IoT using DID and highlighted that DID can play a key role in the privacy-preserving of IoT devices. In 2020, Xinxin et al.

[10] provided a DID-based identity and access management framework for IoT that generates a tamper-proof registry for the IoT devices on top of the blockchain. In 2022, Myeonghyun et al. [11] presented DID-based solutions for privacy-preserving blockchain-based energy trading for vehicle-to-vehicle systems and proved using formal and informal security analysis that their scheme is secured against traditional security attacks. In 2022, Rohini et al. [12] presented a privacy-preserving DID-based authentication protocol for secure electric vehicle charging. It would seem that ensuring privacy has been a goal for many researchers, but some challenges, such as *usability* and *auditability*, remain unaddressed.

The DT plays a critical part in Industry 4.0. and is at the centre of the next-generation industrial revolution. Unfortunately, DTs security and privacy challenges are not thoroughly investigated [13,14]. The user plays a key role in the based industrial eco-system, and there is a critical need to empower the user with self-sovereignty about their identity. The user empowerment part of DT security remains completely unaddressed. To our knowledge, this is the first paper presenting a security solution for DT considering DID-based user empowerment. We are unaware of any solutions for *usability* and *auditability* problems of DID for DT. The work in this paper provides a solution for these problems. We have used GAN-based bio-metric DID to achieve the *usability* and an authentication counter to achieve the *auditability*.

#### 1.1.1. Motivation and contributions

Numerous applications and studies have discussed DT and its role in industrial development. However, only a very few studies have considered the security and privacy aspects of DT. The DID technology can be key in designing security solutions for DT and IoT-based industrial applications. In this article, we identified three fundamental challenges/issues associated with DID technology: *usability*, *auditability* and the *user revocation*. The loss of the user's private key leads to a *usability* problem. An *auditability* problem arises when a system user loses his/her DID and an adversary tries to access the system. In this case, with *auditability*, the user can verify and/or audit such attempts. The *user revocation* need arises when an authenticated system user engages in malicious activities and the system admin needs to remove that user from the system. To the best of our knowledge, this is the first article which has considered and provided an efficient solution to all these issues. Therefore, the major contributions of this article can be summarised as follows:

- A *privacy-preserving user-empowered security framework* for DT using DID and VC, which can also support an efficient way of user revocation.
- To resolve fundamental problems in (*usability* and *auditability*) of the DID technology. In this regard, we use the *Biometric-DID* to achieve effective *usability* and *authentication counter* to achieve efficient *auditability*.
- An *irreversible CyclicGAN-based bio-template* for preserving user privacy. Existing biometric feature extraction technologies, e.g. a fuzzy extractor, suffer from several critical problems such as lack of uniformity and imperfect reproducibility [15]. To overcome all these problems, we propose a *novel bio-template generation system using CyclicGAN* that generates an irreversible bio-template from the user's biometric features and is verifiable later in time.
- A *user revocation system* for efficient user management in the DT environment. To the best of our knowledge, this is the *first decentralised modifiable accumulator* enabled user revocation solution for the DT environment. Based on allocated *dynamic membership witness*, an *accumulator manager* validates the user's membership during authentication.
- A *rigorous security analysis* of the proposed scheme using formal security games and analysis based on fundamental security properties proves that the proposed security framework is secured against most well-known attacks. A *performance analysis* of the proposed

**Table 1**

Symbols.

| Symbols           | Description                                                 |
|-------------------|-------------------------------------------------------------|
| $(VC_x)K_{Pr}^y$  | Verifiable Credential of entity x signed by entity y        |
| $MW_X$            | Membership witness of user X signed by AM                   |
| $DID_x$           | Decentralised Identifier for entity x                       |
| $Enc(m)K_{Pub}^y$ | Encryption of message m by using the public key of entity y |
| $K_{Pr}^y$        | Private key of entity y                                     |
| $K_{Pub}^y$       | Public key of entity y                                      |
| $K_{em}$          | Emergency key                                               |
| PRF               | Pseudo Random Function                                      |
| AM                | Accumulator Manager                                         |
| AZ                | Authorisation Server                                        |
| AS                | Authentication Server                                       |
| CE                | Command Engine                                              |
| U                 | User                                                        |
| $\Delta$          | Unique Bio-template of the User                             |
| KDF               | Key Derivation Function                                     |
| Hash              | Hash Function                                               |
| $n_x, r_x$        | Nonce and Random number generated by entity x               |
| Hash              | Hash Function                                               |
| $T_k$             | Token                                                       |
| $T_S$             | Time Stamp                                                  |
| $M_x$             | xth Message                                                 |

framework shows that it is efficient and effective for ensuring *security and privacy* in DTs.

The remainder of the article is organised as follows: Basic preliminaries such as *DID*, *ML* and *GAN*, *ECDSA*, *accumulator*, *ZKP*, *system model*, *adversary model*, and *security properties* are defined in section 2. In section 3, we present the proposed scheme, which consists of several phases: *user bio-template generation phase*, *initial setup phase*, *verifiable credential obtaining phase*, *token validation and key derivation phase* and *user usability maintaining phase*. The formal security analysis using *random oracles* and informal analysis based on *security properties* is discussed in section 4. In section 5, we present the performance analysis for the proposed scheme, followed by the conclusion and future work in section 6. Table 1 presents basic symbols and notations used in the proposed scheme.

## 2. Preliminaries

This section discusses the basic preliminaries used for designing the proposed scheme.

### 2.1. Zero Knowledge Proof (ZKP)

A method known as the zero-knowledge protocol allows one person, known as the prover, to convince another, known as the verifier, that a particular assertion is true while the prover refrains from providing any information other than the fact that the claim is true. The ZKP with fewer interactions (i.e. single) is considered a Non-interactive ZKP (i.e. zk-SNARK) [16]. Any ZKP must satisfy three properties. The first property is **completeness**, the second property is **soundness** and the third property is **zero-knowledge**. The **completeness** assures that for any true statement, an honest verifier will successfully verify an honest prover. The **soundness** assures that a dishonest prover can never prove any false statement to the honest verifier (or at least can do so only with a negligibly small probability). The **zero-knowledge** assures that the verifier will not get any information related to the statement except the truthfulness of the statement.

Zk-SNARK (Zero-knowledge succinct non-interactive argument of knowledge) is a highly effective non-interactive ZKP method that is used to prove the validity of something without revealing information about it. This protocol consists of the following phases:

- **Key Generation:** In this phase, the proving key  $K_{pk}$  and verification key  $K_{vk}$  is generated as:  $\{K_{pk}, K_{vk}\} \leftarrow KeyGen(C, \tau)$ . Here,  $C$  is an arbitrary circuit, and  $\tau$  is a secret parameter.
- **Proof Generation:** In this phase, the integrity proof  $\phi$  is generated as:  $\{\phi\} \leftarrow ProofGen(Pub_{in}, Pr_{wt})$ . Here  $Pub_{in}$  and  $Pr_{wt}$  are public input and private witness, respectively.
- **Proof Verification:** In this phase, the function  $ProofVrf$  returns *True* if the valid proof is presented by the prover else, it returns *False*. It works as:  $\{True, False\} \leftarrow ProofVrf(Pub_{in}, Pr_{wt}, K_{vk})$ .

The zk-SNARK is best suited for the system that does not require a separate trusted setup for proof verification.

### 2.2. Decentralised Identifier (DID) and permissioned blockchain

The DID is a globally unique identifier that provides self-sovereignty to the users over their identity in terms of generation and control of it. With the help of DID, the user identity presentation system achieves global resolvability, decentralisation, cryptographic verifiability and persistence. There are two essential terms in the DID mechanism. The first one is the **DID document** with which the DID is associated, and it stores public key parameters and other associated metadata. The global DID resolver can resolve the DID into the DID document from the public or private decentralised distributed ledger such as a blockchain. The resolved DID document can be used for signature verification and authentication purposes. The second important term is **Verifiable Credentials (VC)**. These are delegated tamper-resistant alternates of the physical credentials. The VCs are acquired from the trusted entity (i.e. authentication server) and shown by the user to prove their identity over the manufacturing system. With the help of VC, users can prove that they are legitimate players of the system because they have VCs signed by the trusted party of the system.

The DID mechanism involves three major players. The **VC issuer** (i.e. trusted party) signs VCs using its private key and issues those VCs to the **VC holder** (i.e. user), which are unique for each holder. The holder presents those VCs with their DID to the **VC verifier** (i.e. other servers of the manufacturing system) in Verifiable Presentation (VP) format. The verifier resolves the holder's DID and verifies the presented VP. Though the VP is verified over the DID document generated from the central registry, the holder can also prove them with the Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (Zn-SNARK). With the help of Zn-SNARK, the prover (i.e. user) can prove the possession of VCs without interaction with the verifier (i.e. other servers of the manufacturing system) and without revealing any information about VCs.

### 2.3. Machine learning and Generative Adversarial Network

A Generative Adversarial Network (GAN) is a category of machine learning techniques that adopts two simultaneously trained models, called generator and discriminator. The generator generates fake data, and the discriminator distinguishes the fake instances from the real ones. The word generative indicates creating new data from the given data. GAN generates the data that is learned from the training set's choice. The term adversarial points to maintaining the dynamic between the two models: the generator and the discriminator. Here two networks are continually trying to deceive one another as the generator renders reasonably fake data to acquire convincing data. The discriminator attempts to indicate the genuine instances from the fake generated ones. Both the generator and the discriminator employ a neural network. In this article, we adopt a CycleGAN [17] to implement style transfer of fingerprint robustly; it balances the privacy and usability of a fingerprint through visual style transfer techniques, and it is safely reversible for authorised personnel; The core idea of CycleGAN is built on the assumption of cycle consistency, which means that if we have two generative models,  $G$  and  $F$ , that translate between two sets of images,

$X$  and  $Y$ , in which  $Y = G(X)$  and  $X = F(Y)$ , we can naturally assume that  $F(G(X))$  should be very similar to  $X$  and  $G(F(Y))$  should be very similar to  $Y$ . This means that we can train two sets of generative models simultaneously that can freely translate between two sets of images.

## 2.4. Elliptic Curve Digital Signature Algorithm (ECDSA)

Elliptic Curve Cryptography (ECC) is a lightweight public key cryptography solution for securing resource-constrained devices [18]. The ECC involves two major algorithms that make ECC difficult to break. The first one is the *Elliptic Curve Decisional Diffie-Hellman Problem (ECDHP)* and *Elliptic Curve Discrete Logarithm Problem (ECDLP)*. The ECDSA is a widely used signature generation and signature verification solution with asymmetric ECC operations as a base. For the entity  $X$ , if  $K_{Pr}^X$  is a private key and  $G_p$  is the generator point of the elliptic curve  $E_q$  defined over a finite field  $F_q$ , then the public key  $K_{Pub}^X = K_{Pr}^X * G_p$ . As per the ECDLP, it is computationally infeasible for any polynomial-time  $Adv_p$  to compute the  $K_{Pr}^X$  from the given  $\{K_{Pub}^X, G_p\}$  pair. We have considered signature generation in ECDSA as discussed by Doerner et al. [19]. An ECDSA signature is a four-tuples algorithm,  $ECDSA = \{ECDSA.Setup, ECDSA.KeyGen, ECDSA.SignGen, ECDSA.SignVrf\}$  works as follows:

- $\{Pub_{Par}\} \leftarrow ECDSA.Setup(1^\lambda)$ : An ECDSA setup algorithm takes secret  $\lambda$  as an input and provides a public parameter  $Pub_{Par}$  as an output.
- $\{K_{Pr}^X, K_{Pub}^X\} \leftarrow ECDSA.KeyGen(Pub_{Par})$ : An ECDSA  $KeyGen$  algorithm takes a public parameter  $Pub_{Par}$  as an input and returns secret key / signature key  $K_{Pr}^X$  and public key / verification key  $K_{Pub}^X$  as an output.
- $(msg_{sgn}) \leftarrow ECDSA.SignGen(K_{Pr}^X, msg)$ : An ECDSA  $SignGen$  algorithm signs message  $msg$  using secret key / signature key  $K_{Pr}^X$  and generates signature  $msg_{sgn}$ .
- $(1 \text{ or } 0) \leftarrow ECDSA.SignVrf(msg_{sgn}, K_{Pub}^X, msg)$ : An ECDSA  $SignVrf$  algorithm takes signature  $msg_{sgn}$ , message  $msg$  and public key/verification key  $K_{Pub}^X$  as an input and returns "1" if the signature is a valid signature and returns "0" if the signature is invalid.

An ECDSA satisfies two properties:

- **Correctness:** This property assures that all valid signatures will be verified. For all messages  $msg$  in message space, and  $\{K_{Pr}^X, K_{Pub}^X\} \leftarrow ECDSA.KeyGen(Pub_{Par})$ , we can define this property as,

$$\Pr_{K_{Pr}^X, K_{Pub}^X, msg} [[ECDSA.SignVrf((msg_{sgn}, K_{Pub}^X, msg))] = 1] > 1 - negl(\lambda)$$

- **Existential Unforgeability** Any polynomial time adversary  $\mathcal{A}$  must not be able to forge the signature with higher than *negligible property*. Even if  $\mathcal{A}$  receives message-signature pair  $(M_k^*, S_k^*)$  from the valid signer, for any message  $msg \notin M_k^*$ :

$$\Pr_{K_{Pr}^X, K_{Pub}^X} [ECDSA.SignVrf((msg_{sgn}, K_{Pub}^X, msg))] = 1^{msg} \notin M_k^* : (msg_{sgn}, K_{Pub}^X, msg) \leftarrow \mathcal{A}^{ECDSA.SignGen(K_{Pr}^X, \cdot)}(K_{Pub}^X)] < negl(\lambda)$$

## 2.5. Zero-knowledge dynamic accumulator

A trusted entity cryptographic Accumulator Manager (AM) validates the membership and non-membership properties of the system participants in the user space. A membership accumulator assures correctness, and a non-membership accumulator assures soundness property. The

AM publishes a fixed-length digest by aggregating multiple different elements as an accumulator. The Accumulator manager also provides a membership witness ( $MW_x$ ) to the user  $x$ . The user  $x$  presents  $MW_x$  to the verifier to prove that they are still a system member and are not revoked by the system admin (i.e. authentication server). The user in the proposed system uses a non-interactive ZKP to prove membership over the ECC-based dynamic accumulator implemented at the authentication server.

- **Accumulator Parameters:** Considering security parameter  $\Lambda$ , with elliptic curve  $E(F_p)$  defined over prime field  $F_p$  with equation  $Y^2 = X^3 + aX + b \text{ mod } P$  where  $P$  is a large prime number. A parameter  $G_p$  presents a based point, and  $\mathcal{O}$  presents a number of points over the prime field.
- **Accumulator Key Generation:** A key generation function  $KeyGen$  generates an accumulator secret key  $K_{Acc}^{Pr}$  as  $K_{Acc}^{Pr} \leftarrow KeyGen(1^\Lambda)$ . Here  $K_{Acc}^{Pr} \in \mathbb{Z}_p$ . The AM also computes public key as  $K_{Acc}^{Pub} = K_{Acc}^{Pr} * G_p$ . The AM shares  $K_{Acc}^{Pub}$  with all entities of the system and any new users added.
- **Accumulator Initialisation:** The accumulator is initialised with the base point as  $a_0 = G_p$ .
- **Accumulator Update:** For any new user  $X$ ,  $DID_x$  is considered as a data element  $D_x$ , and to protect the privacy, the related value to the  $D_x$  is computed as  $Y_x = \text{Hash}(D_x)$ . The accumulator value for  $D_x$  is computed as  $a_x = (Y_x + K_{Acc}^{Pr}) * a_{(x-1)}$ . For any revoked user with data element  $D_x$ , the next updated accumulator value is  $a_x = (1 / Y_x) * a_{(x+1)}$ .
- **Membership Witness Generation:** The AM issues membership witness  $MW_x$  to the user associated with the data element  $D_x$ . The AM computes  $MW_x$  as  $MW_x = (1 / Y_x + K_{Acc}^{Pr}) * a_x$ . User  $X$  uses  $MW_x$  to prove that  $D_x$  is accumulated into the value  $a_x$ .
- **Membership Witness Update:** Based on the addition of a new user or revocation of the old user, the membership witness for existing users will be constantly updated to prove themselves as a member of the system. Based on the addition of new user  $X_j$ , the membership witness ( $MW_x$ ) of the existing user  $X$  is updated as per the following: Suppose the accumulator state changes from  $a_x$  to  $a'_x$ . Then the membership  $MW_x$  for user  $X$  is updated as  $MW'_x = (Y'_x - Y_x) * MW_x + a_x$  and  $MW'_x$  is sent to user  $X$  as an updated membership witness.
- **Zero Knowledge Membership Witness Verification:** A membership witness  $MW_x$  for user  $x$  is valid for the accumulator state  $a_x$  iff  $e(MW_x, y_x * G_p + K_{Acc}^{Pub}) = e(a_x, G_p)$ . The verifier uses ZKP to verify  $MW_x$  and gets **accept** or **reject** as an outcome of the  $MemVtVrf$  function implemented by the accumulator manager at the authentication server. This proof is considered a Non-interactive ZKP because the verifier will never know the exact value of  $D_x$  but will be able to verify that the user with data element  $D_x$  (i.e. DID) is still a member of the system with only single one-way interactions.

An accumulator manager over the authentication server assures that every non-revoked user  $X$  has updated  $MW_x$  signed by a private key. Thus, the user can present it during the authentication phase to prove their membership in the system.

The system model shown in Fig. 1 presents seven major entities over four major ecosystem spaces. The first space is *physical space* where *physical twins* are deployed. These twins send data over the secure channel to the **database server** deployed over *edge space*. The *cloud space* consists of **command engine**, **authentication server**, **authorisation server**, and the **system admin** with divided functionalities discussed further. The system **users** are part of the **user space**, and based on their authorisation, they try to access data as well as perform command execution over physical twins in the **physical space**.

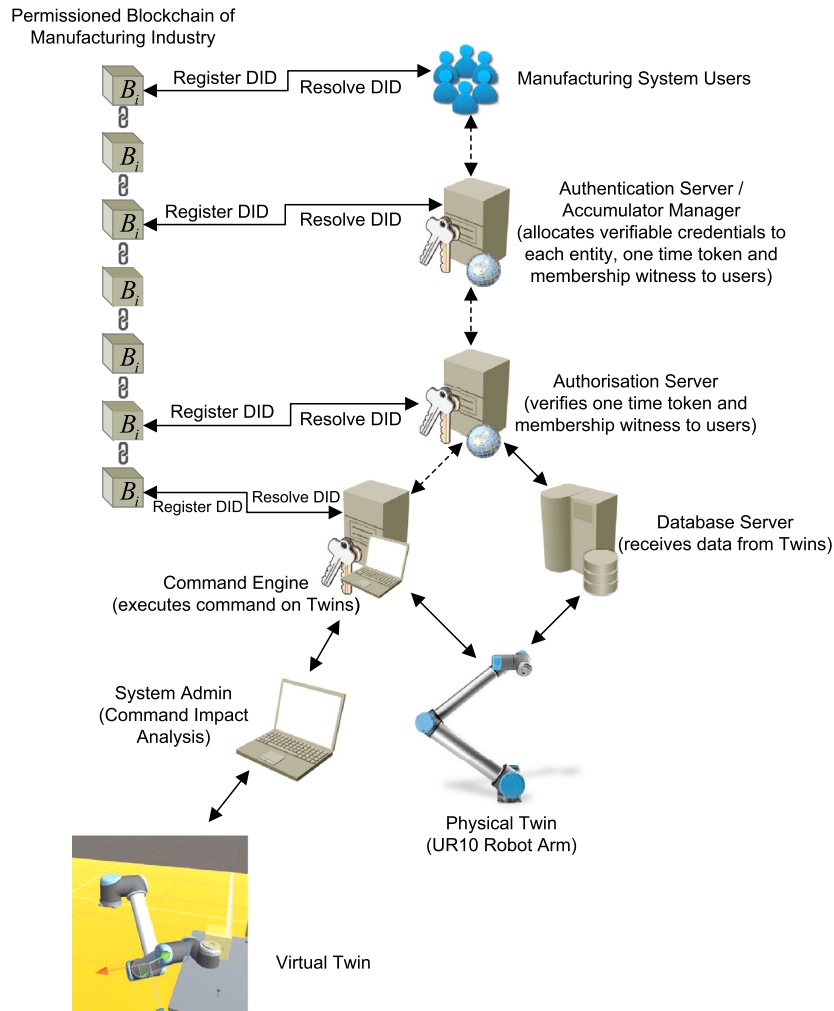


Fig. 1. System Model.

- The system *user* has mainly two functions called *authorised data access* and *secured command executions*.
- The *authentication server* who is a trusted entity of the system and is responsible for providing an authorisation token to the *user*. The *user* presents that token to the *authorisation server* for access verification. The *authentication server* with a dynamic *accumulator manager* is also responsible for providing a membership witness to the *user*.
- The *user* uses the membership witness to prove that they are still a member of the system and not revoked from the system. The *command engine* is responsible for executing a command on the twin that is received from the authorised system *user* through an open channel.
- The *system admin* is responsible for command impact validation on the twin before executing the command on the actual twin and instructs the command engine accordingly.
- The *database server* receives the data from the twin over the secure channel, and any system *user* that wants to get those data has to establish a key with the *authorisation server*. The system *user* doesn't have any direct access to *database server* to avoid attacks related to the database, such as data poisoning.
- The physical twin *UR10* (a product of Universal Robots available with us in university) interacts with the *system admin* and *database server* over the secured TCP interface.
- The *permissioned private blockchain* is managed by a single organisation, and access to this blockchain is restricted to the authorised users and system components. The permissioned private blockchain

allows an organisation to define their rules regarding joining, data access and block validation process. In our case, we are considering permissioned blockchain, however, our system can be integrated with any ledger system.

In the presented system model, there are two open digital threads. The first digital thread is between user - authentication server - and authorisation server, and the second is between the user - user-authentication server - authorisation server - command engine. The first thread is used for data access, and the second is used for the command execution by the user of the manufacturing system. Any attacker over this open thread can easily steal the data and log the command performed over the twin for further attacks like SQL injections.

## 2.6. Adversary model

In this section, we define the capabilities of the adversary  $\mathcal{A}$  for our system model. In this regard, we consider four types of adversaries  $\mathcal{A}$  based on their abilities to monitor and interrupt the digital twin-based manufacturing system.

**Type 1: Adversary with access to open network channels:** We allow *type 1 adversary* to intercept and monitor, change, and delete the messages exchanged between the entities (say, user and authorisation server).

**Type 2: Adversary having access to bio-template:** First, the aim of the *type 2 adversary* is to obtain the encoded bio-template of

the user stored at the authentication server and then perform reverse engineering to obtain the original bio-template.

**Type 3: Adversary having user access DID:** The aim of *type 3 adversary* is to misuse the user DID to establish a key with an authorisation server and command engine.

**Type 4: Adversary having access to the mobile user agent:** In this regard, we consider a *type 4 adversary* have access to the mobile agent of a user for a short duration and tries to delete the private key before being noticed by the user.

## 2.7. Security objectives

Now, we consider the abilities of the above adversaries and identify the following security objectives:

- **User Empowered Authentication:** The user-empowered authentication aims to provide self-sovereignty to the user and manage their identity. No central system provides an identity to the user in the user-empowered authentication [12].
- **Auditability:** Auditability aims to allow a compromised user to detect misuse of their credentials by an adversary at an honest server. The proposed work supports auditability using the authentication counter maintained by the user and the authentication server.
- **Bio-template Confidentiality:** The compromising of a user bio-template should not allow an adversary to perform reverse engineering and must not reveal confidential information about the user, such as user gender, age, etc.
- **Privacy:** The system user identity is not linked to any specific service and is not logged into the permissioned blockchain. The honest but curious authentication server also can not track whether the user is trying to access the data or perform the command execution as well as any outside adversary (eavesdropper) also can not identify and track the user.
- **User Revocation:** The system user revocation is very important for any manufacturing-based organisation when a user leaves or is observed as a malicious insider. In the proposed system, the dynamic accumulator manager issues a dynamic membership witness to the user. The accumulator manager can revoke the membership witness whenever a user leaves the organisation or a user's role changes in the organisation.
- **Usability using Bio-Metric DID:** Any friendly enemy (a trusted person with access to the user's mobile) can delete the private key (associated with his/her DID) of the user from the user's mobile agent to stop the user from accessing services. The *usability* assures that after the loss of the private key also, the system user can securely continue accessing services with the help of bio-template-based bio-metric DID.

## 3. Construction of proposed scheme

In this section, we first discuss our idea and then present the proposed scheme for secure key establishment between the user - authorisation server and user-command engine by solving the existing fundamental problems (*usability*, *auditability*, and *user revocation*) of DID. The proposed scheme consists of *initial setup phase*, *verifiable credential obtaining phase*, *Protocol 1: token validation and key derivation phase for secure data access*, *Protocol 2: token validation and key derivation phase for secure command execution* and *user usability maintaining phase*.

The integration of the blockchain-based DID system with the UR10 robot arm's digital twin is detailed in this section. Following robust user authentication using the Decentralised Identifier (DID), users can interact with the digital twin, which offers two key functionalities: data access and command execution. Data access includes real-time monitoring of operational parameters, providing predictive maintenance insights, and analysing energy consumption. Command execution allows for precise control and automation of the robot arm's tasks, including

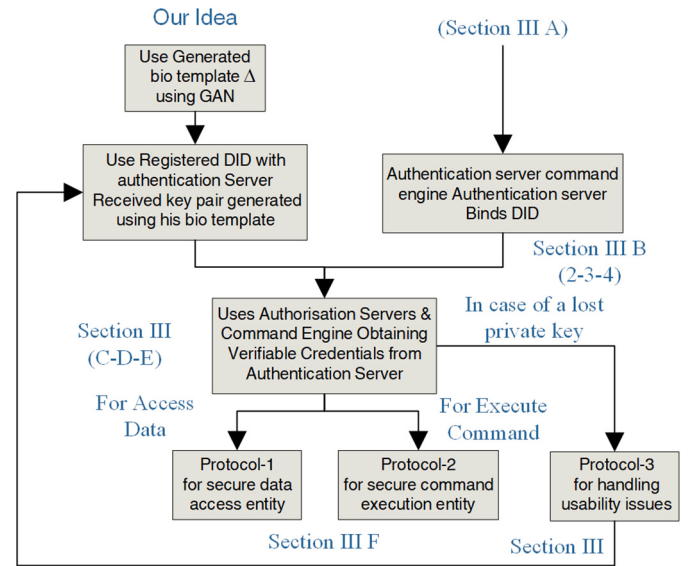


Fig. 2. Our Idea.

remote operations in hazardous environments. This dual-path approach enhances operational efficiency, safety, and adaptability in smart manufacturing settings.

### 3.1. Our idea

In this section, we discuss our basic idea that makes the foundation for articulating the proposed authentication scheme for secure data access and command execution over Digital Twin for the manufacturing industry. Fig. 2 presents an overall idea of the proposed work. In our proposed system, first, the DT user generates bio-template  $\Delta$  using GAN (Discussed in Section 3.2). After a successful generation of  $\Delta$ , the initial setup phase starts in which the user binds a Decentralised Identity (DID) and other parameters over a private blockchain. Upon successful completion of this phase, the user will have a key pair generated using  $\Delta$  and other parameters required for further data access and command execution (Discussed in section 3.3.1). Furthermore, the authorisation server, authentication server and command also bind their decentralised identity over a private blockchain (Discussed in section 3.3.2, 3.3.3, 3.3.4). After completion of the setup phase, the verifiable credential obtaining phase is performed by the user, authorisation server and command engine (Discussed in section 3.4, 3.5, 3.6). Now, the system enters the proposed protocol. If the user wants secure access to data generated by the manufacturing plant, the user calls for **Protocol-1** (Discussed in 3.7), else, if the user wants to perform command execution over the physical counterpart of the digital twin, the user goes for **Protocol-2** (Discussed in 3.7). In case the user loses his/her private key and is not able to access the system agent, it will call for **Protocol-3** (Discussed in 3.8) to get back the usability of the system.

### 3.2. User bio-template generation phase

To preserve users' privacy, we propose a GAN-based fingerprint verification method that learns the latent features of fingerprints through adversarial training of generators  $G_{\lambda\theta}$  and  $G_{\theta\lambda}$ , and their corresponding discriminators  $D_\theta$  and  $D_\lambda$ . These paired networks each handle a direction of translation—encoding and decoding. The privacy-enhancing CycleGAN processes fingerprints to produce complex and noisy images ( $\Delta$ ), as depicted in Fig. 3, which are sent to the server for verification purposes. The discriminators play a vital role in adversarial training, with  $D_\theta$  ensuring the generators produce indistinguishable privacy-enhanced images, and  $D_\lambda$  verifying the authenticity of the Reconstructed Fingerprints (RF).

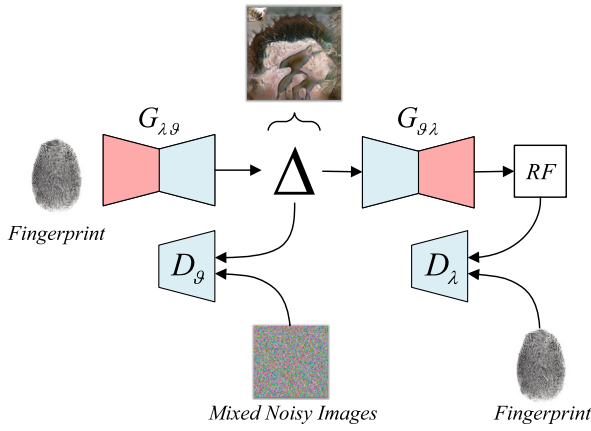


Fig. 3. Proposed GAN-based Privacy-preserving User Bio-template Generation; (RF: Reconstructed Fingerprint).

In the GAN-based privacy-preserving framework, the user's thumbprint serves as the input to the function  $G_{\lambda, \theta}$  in Fig. 4. This function employs a generative approach to transform the fingerprint into a non-invertible, complex representation, denoted as  $\Delta$ , for privacy enhancement. Despite potential unauthorised access to  $\Delta$  and knowledge of  $G_{\lambda, \theta}$ , the reverse engineering to retrieve fingerprint is prohibitively complex. Additionally,  $\Delta$  is securely stored on the server for subsequent authentication processes. The figures depict  $G_{\lambda, \theta}$  as the generator that creates the privacy-preserving template, while  $G_{\theta, \lambda}$  represents the reconstruction mechanism used for authentication, designed to be a one-way process that verifies the authenticity of a fingerprint without reproducing the original, thus maintaining privacy.

### 3.3. Initial setup phase

In this phase, user ( $U$ ), authorisation server ( $AZ_i$ ), command engine ( $CE_i$ ) and authentication server ( $AS_i$ ) each generates a public key, the corresponding private key and a DID for themselves. Later on, these four entities bind DID and Public key in the permissioned blockchain of the manufacturing industry.

#### 3.3.1. A new user DID registration phase

- User presents  $(VC_U)K_{Pr}^{Gov}$  to the authentication server  $AS_i$ . After that,  $AS_i$  verifies verification credential of the user ( $VC_U$ ) signed using the  $K_{Pr}^{Gov}$  with the help of  $K_{Pub}^{Gov}$ .
- After successful verification,  $AS_i$  provides a  $DID_{AS}$  to the user.
- Next, user resolves  $DID_{AS}$  and gets the public key of the  $AS_i$  as  $K_{Pub}^{AS}$ .
- Now, user generates the bio template  $\Delta$  using the CycleGAN (as discussed in section 3.2). After generating the bio-template, the user device extracts the features from  $\Delta$  and converts them into a stable binary string. Now, this binary string is given as an input to generate the private and public key pair as  $\{K_{Pr}^U, K_{Pub}^U\} \leftarrow \text{KDF}(\Delta)$  [20].
- Now, user computes emergency key  $K_{em} = \text{PRF}(\Delta || \text{pwd})$  where  $\text{pwd}$  is a password used by user to regenerate  $K_{em}$  during any emergency time like private key lost.
- Next, user computes  $DID_U = \text{Hash}(K_{Pub}^U)$  and sets authentication counter as  $0 \leftarrow \text{Auth}_{cnt}$ .
- Now, user computes  $Z_U = \text{Enc}(\Delta, K_{em})K_{Pub}^{AS}$ .
- Next, user binds  $\{DID_U, K_{Pub}^U, Z_U, \text{Auth}_{cnt}\}$  over the permissioned blockchain and stores  $\text{Auth}_{cnt}$  in own mobile agent.

#### 3.3.2. Authorisation server DID binding phase

The authorisation server ( $AZ_i$ ) generates a public key  $K_{Pub}^{AZ}$ , and private key  $K_{Pr}^{AZ}$  pair using ECC parameters. Now,  $AZ_i$  computes its

decentralised identifier as  $DID_{AZ} = \text{Hash}(K_{Pub}^{AZ})$  and binds  $DID_{AZ}$  and  $K_{Pr}^{AZ}$  over the permissioned blockchain using the key method of the DID. Next,  $AZ_i$  stores the private key  $K_{Pr}^{AZ}$  into non-erasable secure memory.  $AZ_i$  also has verifiable credentials  $(VC_{AZ})K_{Pr}^{Mfg}$  issued by an authorised manufacturer of the  $AZ_i$ .

#### 3.3.3. Command engine DID binding phase

The command engine ( $CE_i$ ) generates a public key  $K_{Pub}^{CE}$ , and private key  $K_{Pr}^{CE}$  using ECC parameters. Now,  $CE_i$  computes its decentralised identifier as  $DID_{CE} = \text{Hash}(K_{Pub}^{CE})$  and binds  $DID_{CE}$  and  $K_{Pr}^{CE}$  over the permissioned blockchain using the key method of DID. Next,  $CE_i$  stores the private key  $K_{Pr}^{CE}$  into non-erasable secure memory.  $CE_i$  also has verifiable credentials  $(VC_{CE})K_{Pr}^{Mfg}$  issued by the authorised manufacturer of the  $CE_i$ .

#### 3.3.4. Authentication server DID binding phase

The authentication server ( $AS_i$ ) generates a public key  $K_{Pub}^{AS}$ , and private key  $K_{Pr}^{AS}$  using ECC parameters. Now,  $AS_i$  computes its decentralised identifier as  $DID_{AS} = \text{Hash}(K_{Pub}^{AS})$  and binds  $DID_{AS}$  and  $K_{Pr}^{AS}$  over the permissioned blockchain using the key method of the DID. Now,  $AS_i$  stores the private key  $K_{Pr}^{AS}$  into non-erasable secure memory.  $AS_i$  also has verifiable credentials  $(VC_{AS})K_{Pr}^{Mfg}$  issued by the authorised manufacturer of the server.

### 3.4. User verifiable credential obtaining phase

In this phase, user  $U$  receives a verifiable credential  $(VC_U)K_{Pr}^{AS}$  from the authentication server  $AS_i$  to establish a session with other servers of the DT-enabled manufacturing system. As shown in Table 2, the user generates a nonce  $n_U$  and constructs a message  $M_1$ . Next, the user sends  $M_1$  to the AS/AM. Upon receiving  $M_1$ , the AS/AM resolves the user DID and verifies the signature over  $n_U$  to validate that the user is a registered system user. Now, the AS/AM also verifies signature  $(VC_U)K_{Pr}^{Gov}$  followed by generation of the message  $M_2$  and sending it to the user. After receiving the message  $M_2$  from the AS/AM, the user verifies a signature on  $n_{AS}$  by resolving the DID of the AS/AM. Now, the user confirms the message sender by computing  $X_1^*$  followed by a comparison with the received  $X_1$ . After this successful verification, the user stores  $(VC_U)K_{Pr}^{AS}$  signed by the AS/AM.

### 3.5. Authorisation server verifiable credential obtaining phase

In this phase, the authorisation server ( $AZ$ ) receives a verifiable credential  $(VC_{AZ})K_{Pr}^{AS}$  from the authentication server ( $AS_i$ ) used to establish a session with the other parties of the DT enabled manufacturing system. As shown in Table 3, an authorisation server generates a nonce  $n_{AZ}$  followed by generating message  $M_1$ . The authorisation server sends  $M_1$  to the AS/AM. Upon receiving  $M_1$ , AS/AM resolves authorisation server DID and verifies the signature over  $n_{AZ}$  to validate that the authorisation server is a registered system server. Next, AS/AM also verifies signature over  $(VC_{AZ})K_{Pr}^{Mfg}$  followed by message  $M_2$  generation. AS/AM forwards message  $M_2$  to  $AZ$ . After receiving message  $M_2$  from the AS/AM,  $AZ$  verifies the signature on  $n_{AS}$  by resolving the DID of the AS/AM. Next,  $AZ$  verifies the message sender by computing  $X_1^*$  followed by comparison with the received  $X_1$ . Upon successful verification,  $AZ$  stores  $(VC_{AZ})K_{Pr}^{AS}$  signed by the AS/AM.

### 3.6. Command engine verifiable credential obtaining phase

In this phase, the command engine ( $CE$ ) receives a verifiable credential  $(VC_{CE})K_{Pr}^{AS}$  from the  $AS_i$  used to establish a session with the other parties of the DT-enabled manufacturing system. As shown in Table 4,  $CE$  generates a nonce  $n_{CE}$  followed by generating message  $M_1$ .

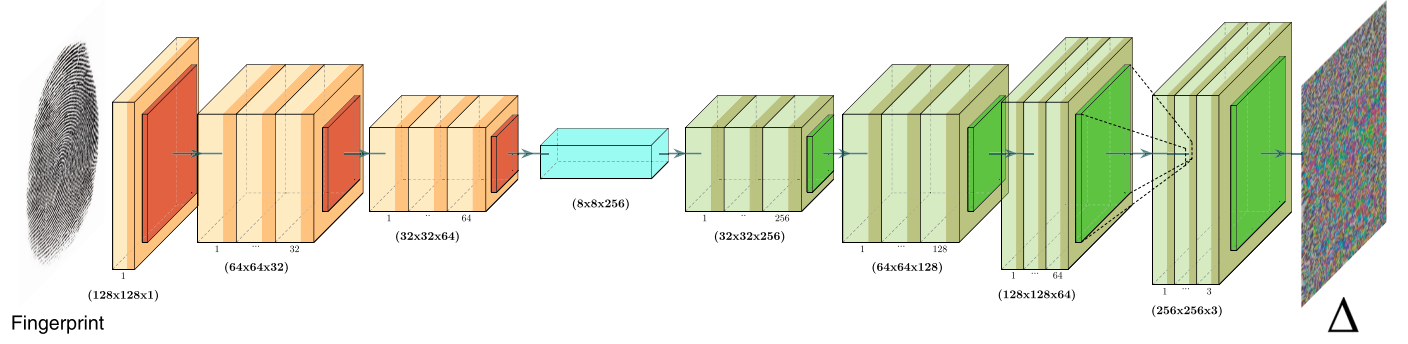
Fig. 4. GAN Generator,  $G_{\lambda\theta}$ .

Table 2

User Verifiable Credential Obtaining Phase.

| User                                                                                                                                                          | Authentication Server/ Accumulator Manager                                                                                                                                                                                                                                                                                                                          |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Generate $n_U$<br>$M_1 = \{DID_U, (VC_U)K_{Pr}^{Gov}, Sign_{Gen}(n_U)K_{Pr}^U\}$                                                                              |                                                                                                                                                                                                                                                                                                                                                                     |
|                                                                                                                                                               | $M_1$                                                                                                                                                                                                                                                                                                                                                               |
|                                                                                                                                                               | Resolves $DID_U$ and $Sign_{Verf}(n_U)K_{Pr}^U$ ,<br>$Sign_{Verf}(VC_U)K_{Pr}^{Gov}$ ,<br>Generate $n_{AS}$ , $Sign_{Gen}(n_{AS})K_{Pr}^{AS}$<br>Generate $VC_U = Hash(n_U    n_{AS}    (VC_U)^{Gov})$ ,<br>$Sign_{Gen}(VC_U)K_{Pr}^{AS}$ ,<br>$X_1 = Hash(salt    (VC_U)K_{Pr}^{AS})$ ,<br>$M_2 = \{X_1, (VC_U)K_{Pr}^{AS}, salt, (n_{AS})K_{Pr}^{AS}, DID_{AS}\}$ |
|                                                                                                                                                               | $M_2$                                                                                                                                                                                                                                                                                                                                                               |
| Resolve $DID_{AS}$ and $Sign_{Verf}(n_{AS})K_{Pr}^{AS}$ ,<br>$X_1^* = Hash(salt    n_U    n_{AS}    (VC_U)K_{Pr}^{Gov}) = X_1$ ,<br>Store $(VC_U)K_{Pr}^{AS}$ |                                                                                                                                                                                                                                                                                                                                                                     |

Table 3

Authorisation Server Verifiable Credential Obtaining Phase.

| Authorisation Server                                                                                                                                                   | Authentication Server                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Generate $n_{AZ}$<br>$M_1 = \{DID_{AZ}, (VC_{AZ})K_{Pr}^{Mfg}, Sign_{Gen}(n_{AZ})K_{Pr}^{AZ}\}$                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                   |
|                                                                                                                                                                        | $M_1$                                                                                                                                                                                                                                                                                                                                                                                             |
|                                                                                                                                                                        | Resolves $DID_{AZ}$ and $Sign_{Verf}(n_{AZ})K_{Pr}^{AZ}$ ,<br>$Sign_{Verf}(VC_{AZ})K_{Pr}^{Mfg}$ ,<br>Generate $n_{AS}$ , $Sign_{Gen}(n_{AS})K_{Pr}^{AS}$<br>Generate $VC_{AZ} = Hash(n_{AZ}    n_{AS}    (VC_{AZ})^{Mfg})$ ,<br>$Sign_{Gen}(VC_{AZ})K_{Pr}^{AS}$ ,<br>$X_1 = Hash(salt    (VC_{AZ})K_{Pr}^{AS})$ ,<br>$M_2 = \{X_1, (VC_{AZ})K_{Pr}^{AS}, salt, (n_{AS})K_{Pr}^{AS}, DID_{AS}\}$ |
|                                                                                                                                                                        | $M_2$                                                                                                                                                                                                                                                                                                                                                                                             |
| Resolve $DID_{AS}$ and $Sign_{Verf}(n_{AS})K_{Pr}^{AS}$ ,<br>$X_1^* = Hash(salt    n_{AZ}    n_{AS}    (VC_{AZ})K_{Pr}^{Mfg}) = X_1$ ,<br>Store $(VC_{AZ})K_{Pr}^{AS}$ |                                                                                                                                                                                                                                                                                                                                                                                                   |

Next,  $CE$  forwards  $M_1$  to the AS/AM. Next, AS/AM resolves command engine DID verifies the signature over  $n_{CE}$  and validates that the  $CE$  is a registered system entity. Furthermore, AS/AM also verifies signature over  $(VC_{CE})K_{Pr}^{Mfg}$  followed by message  $M_2$  generation. Now, AS/AM forwards  $M_2$  to the  $CE$ . After receiving message  $M_2$  from the AS/AM,  $CE$  verifies the signature on  $n_{AS}$  by resolving the DID of the AS/AM. Now,  $CE$  verifies the message sender by computing  $X_1^*$  followed by comparison with the received  $X_1$ . If the verification is successful,  $CE$  stores  $(VC_{CE})K_{Pr}^{AS}$  signed by the AS/AM.

### 3.7. Token validation and key derivation phase

#### Protocol 1: Token Validation and Key Derivation Phase for Secure Data Access:

After execution of above phases, user holds  $\{DID_U, (VC_U)K_{Pr}^{AS}, K_{Pr}^U, DID_{AS}\}$ , authorisation server holds  $\{DID_{AZ}, (VC_{AZ})K_{Pr}^{AS}, K_{Pr}^{AZ}\}$ , authentication server holds  $\{DID_{AS}, DID_U, DID_{AZ}, DID_{CE}, K_{Pr}^{AS}\}$  and command engine holds  $\{DID_{CE}, (VC_{CE})K_{Pr}^{AS}, K_{Pr}^{CE}, DID_{AS}\}$ . In this phase, the user receives a one-time authorisation to-

**Table 4**  
Command Engine Verifiable Credential Obtaining Phase.

|                                                                            |                                                                                                                                                                                                                                                                                                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Command Engine                                                                                                                                              | Authentication Server                                                                                                                                                                                                                                                                                                                                                              |
| Generate $n_{CE}$<br>$M_1 = \{DID_{CE}, (VC_{CE})K_{Pr}^{Mfg}, Sign_{Gen}(n_{CE})K_{Pr}^{CE}\}$                                                             | Resolves $DID_{CE}$ and $Sign_{Vrf}(n_{CE})K_{Pr}^{CE}$<br>$Sign_{Vrf}(VC_{CE})K_{Pr}^{Mfg}$<br>Generate $n_{AS}, Sign_{Gen}(n_{AS})K_{Pr}^{AS}$<br>Generate $VC_{CE} = Hash(n_{CE}  n_{AS}  (VC_{CE})K_{Pr}^{Mfg})$<br>$Sign_{Gen}(VC_{CE})K_{Pr}^{AS}$<br>$X_1 = Hash(salt  (VC_{CE})K_{Pr}^{AS})$<br>$M_2 = \{X_1, (VC_{CE})K_{Pr}^{AS}, salt, (n_{AS})K_{Pr}^{AS}, DID_{AS}\}$ |
| $M_1$                                                                                                                                                       | $M_2$                                                                                                                                                                                                                                                                                                                                                                              |
| Resolve $DID_{AS}$ and $Sign_{Vrf}(n_{AS})K_{Pr}^{AS}$<br>$X_1^* = Hash(salt  n_{CE}  n_{AS}  (VC_{CE})K_{Pr}^{Mfg}) = X_1$<br>Store $(VC_{CE})K_{Pr}^{AS}$ |                                                                                                                                                                                                                                                                                                                                                                                    |

ken from the authentication server and establishes a session key with the authorisation server upon successful token validation. During this phase, the user presents membership witness  $MW_U$  to the accumulator manager running over the authentication server. The accumulator manager verifies  $MW_U$  for the user and instructs the authentication server to continue or not. Table 5 shows the working of this phase.

**Step 1.**  $U \rightarrow AS/AM$ : The user  $U$  generates a nonce  $n_U$  and generates a signature using his/her own private key. Now,  $U$  obtains  $K_{Pub}^{AS}$  by resolving the DID of the AS/AM. Next,  $U$  generates a message  $M_1 = Enc(DID_U, (n_U)K_{Pr}^U, MW_U, TS_1)K_{Pub}^{AS}$  and sends  $\{M_1\}$  to AS/AM.

**Step 2.**  $AS/AM \rightarrow U$  and  $AS/AM \rightarrow AZ$ : Upon receiving  $\{M_1\}$ , AS verifies  $TS_1^* - TS_1 = \Delta T$  and AM verifies  $MW_U$  after decryption of the message  $M_1$ . Now, AS/AM resolves  $DID_U$  and verifies sign-on  $n_U$ . Furthermore, AS/AM retrieves current  $Auth_{cnt}$  from  $U$ 's DIDDoc and increments it by one. Next, AS/AM also generates one one-time token  $T_k$  and random nonce  $n_{AS}$ . After that, AS/AM computes  $X_1 = Hash(T_k||n_{AS}||n_U)$ , generate signature on  $X_1$  using  $K_{Pr}^{AS}$  and generates  $M_2 = Enc(S_1, n_{AS}, T_k, DID_{AZ}, Auth_{cnt}, TS_2)K_{Pub}^U$  for  $U$ . AS/AM also generates  $M_3 = Enc(S_1, T_k, DID_U, n_{AS}, TS_3)K_{Pub}^{AZ}$  for  $AZ$ . Now, AS/AM sends  $\{M_2\}$  to  $U$  and  $\{M_3\}$  to  $AZ$ .

**Step 3.**  $U \rightarrow AS \rightarrow AZ$ : Upon receiving  $\{M_2\}$ ,  $U$  verifies  $TS_2^* - TS_2 = \Delta T$  and decrypts message  $M_2$  using own private key. Next,  $U$  verifies  $Auth_{cnt}$  received from AS/AM, and it should be exactly one value lesser than the available  $Auth_{cnt}$  value with him/her. If the  $Auth_{cnt}$  value is higher, a user can audit that any adversary tried to use his/her DID to authenticate with AS/AM. Upon successful audit,  $U$  verifies signature  $S_1$ , further generates random  $r_U$  and computes  $X_2 = Hash(r_U||S_1||T_k)$  using one time token  $T_k$ . Now,  $U$  resolves  $DID_{AZ}$  received from the AS, computes  $M_4 = Enc(X_2, r_U, TS_4)K_{Pub}^{AZ}$  and sends  $\{M_4\}$  to  $AZ$ .

**Step 4.**  $AZ \rightarrow AS \rightarrow U$ : Upon receiving  $\{M_3\}$  and  $\{M_4\}$ ,  $AZ$  verifies  $TS_3^* - TS_3 = \Delta T$ , verifies  $TS_4^* - TS_4 = \Delta T$ , and decrypts  $M_3$  and  $M_4$ . Now,  $AZ$  verifies signature  $S_1$  and validates  $X_2^* = Hash(r_U||S_1||T_k') = X_2$ . Upon successful validation,  $AZ$  discards  $T_k'$  received from the AS. Hence,  $T_k$  received from AS/AM will be used only once by  $AZ$  for  $U$ 's authorisation purpose. Now,  $AZ$  generates a random  $r_{AZ}$ , resolves  $DID_U$  and computes  $M_5 = Enc(sign((VC_{AZ})K_{Pr}^{AS}), r_{AZ}, TS_5)K_{Pub}^U$ . At last,  $AZ$  sends  $\{M_5\}$  to  $U$ .

**Step 5.**  $U \rightarrow AS \rightarrow AZ$ : Upon receiving  $\{M_5\}$  from  $AZ$ ,  $U$  verifies  $TS_5^* - TS_5 = \Delta T$ , decrypts  $M_5$ , and verifies both the signatures on  $VC_{AZ}$ . Now,  $U$  computes  $M_6 = Enc(Sign((VC_U)K_{Pr}^{AS}, TS_6)K_{Pr}^U)$  and sends  $\{M_6\}$  to  $AZ$ .  $U$  computes key as:  $K_{U-AZ} \leftarrow KDF(VC_U||VC_{AZ}||r_U||r'_{AZ}||n_{AS})$ .

**Step 6.**  $AZ$ : Upon receiving  $\{M_6\}$ ,  $AZ$  verifies  $TS_6^* - TS_6 = \Delta T$  and decrypts  $M_6$ . Now,  $AZ$  verifies both signatures on  $VC_U$  by  $U$  itself and by issuer AS/AM. Upon successful verification,  $AZ$  computes key as:  $K_{AZ-U} \leftarrow KDF(VC_U||VC_{AZ}||r'_U||r'_{AZ}||n_{AS})$

**Protocol 2: Token Validation and Key Derivation Phase for Secure Command Execution:** In this phase, the user ( $U$ ) receives a one-time authorisation token from the authentication server (AS/AM) and establishes a session key with the command engine ( $CE$ ) for secure command execution upon successful token validation by the authorisation server ( $AZ$ ). The user communicates with a command engine through the authentication and authorisation servers before key derivation. In this phase, each participating entity verifies verifiable credentials issued to each other by the authentication server. Table 6 presents the working of this phase.

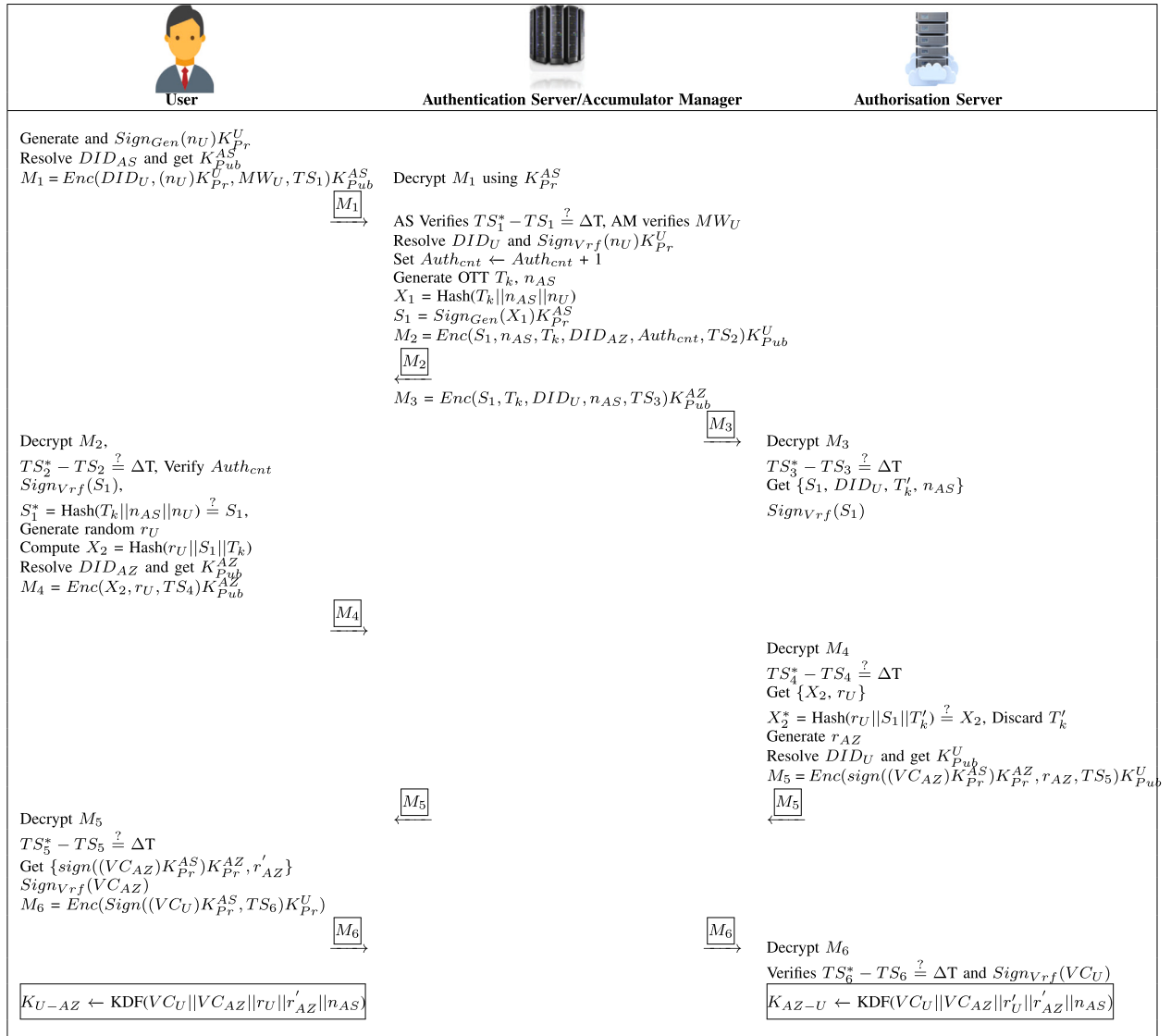
**Step 1.**  $U \rightarrow AS/AM$ :  $U$  generates  $n_U$  and signs it using own private key. Now,  $U$  obtains  $K_{Pub}^{AS}$  by resolving the DID of the authentication server. Next,  $U$  generates  $M_1 = Enc(DID_U, (n_U)K_{Pr}^U, MW_U, TS_1)K_{Pub}^{AS}$  and sends  $\{M_1\}$  to AS/AM.

**Step 2.**  $AS/AM \rightarrow U$  and  $AS/AM \rightarrow AZ$ : Upon receiving  $\{M_1\}$ , AS Verifies  $TS_1^* - TS_1 = \Delta T$  and AM verifies  $MW_U$  after decryption of  $M_1$ . Now, AS/AM resolves  $DID_U$  and verifies the signature on  $n_U$ . Next, AS/AM retrieves current  $Auth_{cnt}$  from the user  $U$ 's DID binding and increments it by one. Furthermore, AS/AM also generates one one-time token  $T_k$  and random nonce  $n_{AS}$ . Next, AS/AM computes  $X_1 = Hash(T_k||n_{AS}||n_U)$ , signs  $X_1$  using  $K_{Pr}^{AS}$  and generates  $M_2 = Enc(S_1, n_{AS}, T_k, DID_{AZ}, Auth_{cnt}, TS_2)K_{Pub}^U$  for  $U$ . Next, AS/AM also generates  $M_3 = Enc(S_1, T_k, DID_U, DID_{CE}, TS_3)K_{Pub}^{AZ}$  for  $AZ$ . After that, AS/AM sends  $\{M_2\}$  to  $U$  and  $\{M_3\}$  to  $AZ$ .

**Step 3.**  $U \rightarrow AS/AM \rightarrow AZ$ : Upon receiving  $\{M_2\}$ ,  $U$  verifies  $TS_2^* - TS_2 = \Delta T$  and decrypts message  $M_2$  using own private key. Next,  $U$  verifies  $Auth_{cnt}$  received from AS/AM, and it should be exactly one value lesser than the available  $Auth_{cnt}$  value with him/her. If the  $Auth_{cnt}$  value is higher it means the author

**Table 5**

Token Validation and Key Derivation Phase for Secure Data Access.



audit that any adversary tried to use his/her DID to perform authentication with AS/AM. Upon successful audit,  $U$  verifies the signature  $S_1$ , retrieves  $X_1$  and verifies  $X_1' = Hash(T_k || n_{AS} || n_U) \stackrel{?}{=} x_1$ . Further,  $U$  generates random  $r_U$ , computes  $X_2 = Hash(r_U || S_1 || T_k)$  using a one time token  $T_k$  and  $S_2 = Sign_{Gen}(X_2)K_{Pr}^U$ . Now,  $U$  resolves  $DID_{AZ}$  and gets  $K_{Pub}^{AZ}$ , computes  $M_4 = Enc(S_2, r_U, TS_4)K_{Pub}^{AZ}$  and sends  $\{M_4\}$  to AZ.

**Step 4. AZ  $\rightarrow$  AS/AM  $\rightarrow$  U:** Upon receiving  $\{M_3\}$  and  $\{M_4\}$ , AZ verifies  $TS_3^* - TS_3 \stackrel{?}{=} \Delta T$  and verifies  $TS_4^* - TS_4 \stackrel{?}{=} \Delta T$ . Next, AZ decrypts  $M_3$ , gets  $T_k'$ , verifies signature  $S_1$  and retrieves  $X_1$  as  $X_1'$ . Further, AZ resolves  $DID_U$  and  $DID_{CE}$  to receive  $K_{Pub}^U$  and  $K_{Pub}^{CE}$  respectively. After decryption of message  $M_4$ , AZ verifies signature  $S_2$ , and validates  $X_2^* = Hash(r_U || X_1' || T_k') \stackrel{?}{=} X_2$ . Upon successful validation, AZ discards  $T_k'$ . Now, AZ generates a random number  $r_{AZ}$  and generates the message  $M_5 = Enc(Sign_{Gen}(VC_{AZ}, TS_5)K_{Pr}^{AS})K_{Pr}^{AZ}, r_{AZ})K_{Pub}^U$ . Finally, AZ sends  $\{M_5\}$  to U.

**Step 5.  $U \rightarrow$  AS/AM  $\rightarrow$  AZ:** Upon receiving  $\{M_5\}$ ,  $U$  verifies  $TS_5^* - TS_5 \stackrel{?}{=} \Delta T$ , decrypts  $M_5$  and verifies both signature on  $VC_{AZ}$ . Fur-

ther,  $U$   $M_6 = Enc(Sign_{Gen}(VC_U)K_{Pr}^{AS}, TS_6)K_{Pr}^U)K_{Pub}^{AZ}$  and sends  $\{M_6\}$  to AZ.

**Step 6. AZ  $\rightarrow$  CE:** Upon receiving  $\{M_6\}$ , AZ decrypts  $M_6$  and verifies signature on  $VC_U$  and verifies  $TS_6^* - TS_6 \stackrel{?}{=} \Delta T$ . Next, AZ generates  $S_3 = Sign_{Gen}((VC_U)K_{Pr}^{AS})K_{Pr}^U)K_{Pr}^{AZ}$  and  $S_4 = Sign_{Gen}((VC_{AZ})K_{Pr}^{AS})K_{Pr}^{AZ}$  followed by  $M_7 = Enc(S_3, S_4, DID_U, DID_{AZ}, r_U, r_{AZ}, TS_7)K_{Pub}^{CE}$ . last, AZ sends  $\{M_7\}$  to CE.

**Step 7. CE  $\rightarrow$  AZ:** After receiving  $\{M_7\}$ , CE decrypts  $M_7$ , resolves  $DID_U$ ,  $DID_{AZ}$  and  $DID_{AS}$ , and verifies  $TS_7^* - TS_7 \stackrel{?}{=} \Delta T$ . Now, CE verifies signatures  $S_3$  and  $S_4$  and generates a random number  $r_{CE}$ . Now, CE generates  $S_5 = Sign_{Gen}((VC_{CE})K_{Pr}^{AS})K_{Pr}^{CE}$ , and  $M_8 = Enc(S_5, r_{CE}, TS_8)K_{Pub}^{AZ}$  and sends  $\{M_8\}$  to AZ.

**Step 8. AZ  $\rightarrow$  AS/AM  $\rightarrow$  U:** Upon receiving  $\{M_8\}$ , AZ decrypts  $M_8$ , verifies  $TS_8^* - TS_8 \stackrel{?}{=} \Delta T$  and verifies  $S_5$ . Now, AZ computes  $S_6 = Sign_{Gen}((VC_{CE})K_{Pr}^{AS})K_{Pr}^{AZ}$  and  $M_9 = Enc(S_6, r_{CE}, TS_9)K_{Pub}^U$ . Further, AZ sends  $\{M_9\}$  to U.

Table 6

Token Validation and Key Derivation Phase for Secure Command Execution.

| <br>User                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | <br>AS/AM | <br>Authorisation Server                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <br>Command Engine                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Generate and <math>Sign_{Gen}(n_U)K_{Pr}^U</math><br/>           Resolve <math>DID_{AS}</math> and get <math>K_{Pub}^{AS}</math><br/> <math>M_1 = Enc(DID_U, (n_U)K_{Pr}^U, MW_U, TS_1)K_{Pub}^{AS}</math><br/> <math>\xrightarrow{M_1}</math></p> <p>Decrypt <math>M_1</math> using <math>K_{Pr}^{AS}</math>,<br/>           Verifies <math>TS_1^* - TS_1 \stackrel{?}{=} \Delta T</math>, AM<br/>           verifies <math>MW_U</math>, AS sets <math>Auth_{cnt}</math><br/> <math>\leftarrow Auth_{cnt} + 1</math><br/>           Resolve <math>DID_U</math> and<br/> <math>Sign_{Vrf}(n_U)K_{Pr}^U</math><br/>           Generate OTT <math>T_k, n_{AS}</math><br/> <math>X_1 = Hash(T_k    n_{AS}    n_U)</math><br/> <math>S_1 = Sign_{Gen}(X_1)K_{Pr}^{AS}</math><br/> <math>M_2 = Enc(S_1, n_{AS}, T_k, DID_{AZ}, Auth_{cnt}, TS_2)K_{Pub}^U</math><br/> <math>\xrightarrow{M_2}</math></p> <p><math>M_3 = Enc(S_1, T_k, DID_U, DID_{CE}, TS_3)K_{Pub}^{AZ}</math><br/> <math>\xrightarrow{M_3}</math></p> <p>Decrypt <math>M_2</math>,<br/>           Verifies <math>TS_2^* - TS_2 \stackrel{?}{=} \Delta T</math>, Verify<br/> <math>Auth_{cnt}</math><br/> <math>Sign_{Vrf}(S_1)</math> and retrieve <math>X_1</math>,<br/> <math>X_1' = Hash(T_k    n_{AS}    n_U) \stackrel{?}{=} x_1</math>,<br/>           Generate random <math>r_U</math><br/>           Compute <math>X_2 = Hash(r_U    X_1    T_k)</math><br/> <math>S_2 = Sign_{Gen}(X_2)K_{Pr}^U</math><br/>           Resolve <math>DID_{AZ}</math> and get <math>K_{Pub}^{AZ}</math><br/> <math>M_4 = Enc(S_2, r_U, TS_4)K_{Pub}^{AZ}</math><br/> <math>\xrightarrow{M_4}</math></p> <p>Decrypt <math>M_5</math><br/>           Verifies <math>TS_5^* - TS_5 \stackrel{?}{=} \Delta T</math> and<br/> <math>Sign_{Vrf}(VC_{AZ})</math><br/>           Compute <math>M_6 = Enc(Sign_{Gen}(VC_U)K_{Pr}^{AS}, TS_6)K_{Pr}^U)K_{Pub}^{AZ}</math><br/> <math>\xrightarrow{M_6}</math></p> <p>Decrypt <math>M_9</math>, Verifies <math>TS_9^* - TS_9</math><br/> <math>\stackrel{?}{=} \Delta T</math> and <math>Sign_{Vrf}(S_6)</math><br/> <math>K_{U-CE} \leftarrow KDF(VC_{CE}    VC_U    r_{CE}    r_{AZ}'    r_U')</math></p> |                                                                                            | <p>Decrypt <math>M_3</math> and get <math>T_k'</math><br/>           Verifies <math>TS_3^* - TS_3 \stackrel{?}{=} \Delta T</math><br/> <math>Sign_{Vrf}(S_1)</math> and retrieve <math>X_1</math> as<br/> <math>X_1'</math><br/>           Resolve <math>DID_U</math> and get <math>K_{Pub}^U</math><br/>           Resolve <math>DID_{CE}</math> and get <math>K_{Pub}^{CE}</math></p> <p>Decrypt <math>M_4</math><br/>           Verifies <math>TS_4^* - TS_4 \stackrel{?}{=} \Delta T</math>,<br/> <math>Sign_{Vrf}(S_2)</math> and retrieve <math>X_2</math><br/> <math>X_2^* = Hash(r_U    X_1'    T_k') \stackrel{?}{=} X_2</math><br/>           Discard <math>T_k'</math><br/>           Generate <math>r_{AZ}</math><br/> <math>M_5 = Enc(Sign_{Gen}(VC_{AZ}, TS_5)K_{Pr}^{AS})K_{Pr}^{AZ}, r_{AZ})K_{Pub}^U</math><br/> <math>\xrightarrow{M_5}</math></p> <p>Decrypt <math>M_6</math><br/>           Verifies <math>TS_6^* - TS_6 \stackrel{?}{=} \Delta T</math> and<br/> <math>Sign_{Vrf}(VC_U)</math><br/> <math>S_3 = Sign_{Gen}((VC_U)K_{Pr}^{AS})K_{Pr}^U)K_{Pr}^{AZ}</math><br/> <math>S_4 = Sign_{Gen}((VC_{AZ})K_{Pr}^{AS})K_{Pr}^{AZ})</math><br/> <math>M_7 = Enc(S_3, S_4, DID_U, DID_{AZ}, r_U, r_{AZ}, TS_7)K_{Pub}^{CE}</math><br/> <math>\xrightarrow{M_7}</math></p> <p>Decrypt <math>M_8</math>, Verifies <math>TS_8^* - TS_8</math><br/> <math>\stackrel{?}{=} \Delta T</math>, and <math>Sign_{Vrf}(S_5)</math><br/> <math>S_6 = Sign_{Gen}((VC_{CE})K_{Pr}^{AS})K_{Pr}^{CE})</math><br/> <math>M_8 = Enc(S_5, r_{CE}, TS_8)K_{Pub}^{AZ}</math><br/> <math>\xrightarrow{M_8}</math></p> <p><math>M_9 = Enc(S_6, r_{CE}, TS_9)K_{Pub}^U</math><br/> <math>\xrightarrow{M_9}</math></p> <p><math>K_{CE-U} \leftarrow KDF(VC_{CE}    VC_U    r_{CE}    r_{AZ}'    r_U')</math></p> | <p>Decrypt <math>M_7</math>, Verifies <math>TS_7^* - TS_7 \stackrel{?}{=} \Delta T</math> and resolve<br/> <math>DID_U, DID_{AZ}</math> and <math>DID_{AS}</math><br/> <math>Sign_{Vrf}(S_3)</math><br/> <math>Sign_{Vrf}(S_4)</math><br/>           Generate <math>r_{CE}</math><br/> <math>S_5 = Sign_{Gen}((VC_{CE})K_{Pr}^{AS})K_{Pr}^{CE})</math><br/> <math>M_8 = Enc(S_5, r_{CE}, TS_8)K_{Pub}^{AZ}</math><br/> <math>\xrightarrow{M_8}</math></p> <p><math>K_{CE-U} \leftarrow KDF(VC_{CE}    VC_U    r_{CE}    r_{AZ}'    r_U')</math></p> |

**Table 7**  
User Usability Maintaining Phase.

| User                                                                                                                                                                                                                                                                                                                      | Authentication Server                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Generates $n_U$ ,<br>Generate bio-template $\Delta$ ,<br>Computes $X_1 = \text{Hash}(\Delta    n_U    K_{Pub_{old}}^U)$<br><br>Generates $M_1 = \text{Enc}(DID_U, n_U, X_1, \Delta, TS_1) K_{Pub}^{AS}$                                                                                                                   | Decrypt $M_1$ , Resolve $DID_U$ ,<br><br>Verifies $TS_1^* - TS_1 \stackrel{?}{=} \Delta T$ ,<br>Untie $Z_U$ and retrieves $\{\Delta', K_{em}, K_{Pub_{old}}^U\}$ ,<br>Verify $\Delta - \Delta' \leq \text{Threshold}$ ,<br>$X_1^* = \text{Hash}(\Delta    n_U    K_{Pub_{old}}^U) \stackrel{?}{=} X_1$ ,<br>Compute $\{K_{Pr_{old}}^U, K_{Pub_{old}}^U\} \leftarrow \text{KDF}(\Delta')$ ,<br>Generate $n_{AS}$ and $\text{Sign}_{Gen}(S_1) = (n_{AS}) K_{Pr}^{AS}$ , |
| Computes $K_{em} = \text{PRF}(\Delta    PW D)$<br>Verifies $TS_2^* - TS_2 \stackrel{?}{=} \Delta T$ , Decrypt $M_2$ using $K_{em}$<br>Verifies $S_1$ using $K_{Pub}^{AS}$<br>$\{K_{Pr_{new}}^U, K_{Pub}^{U_{new}}\} \leftarrow \text{KDF}(\Delta'    K_{Pr_{old}}^U)$<br>$DID_{U_{new}} = \text{Hash}(K_{Pub}^{U_{new}})$ | $M_2 = \text{Enc}(K_{Pr_{old}}^U, S_1, TS_2) K_{em}$                                                                                                                                                                                                                                                                                                                                                                                                                  |
| $M_3 = \text{Enc}(DID_U, DID_{U_{new}}, K_{Pub}^{U_{new}}, Auth_{cnt}, TS_3) K_{Pub}^{AS}$                                                                                                                                                                                                                                | Decrypt $M_3$ , Verifies $TS_3^* - TS_3 \stackrel{?}{=} \Delta T$<br>Binds $\{DID_U, DID_{U_{new}}, K_{Pub}^{U_{new}}, Auth_{cnt}\}$                                                                                                                                                                                                                                                                                                                                  |

**Step 9.**  $U$ : Upon receiving  $\{M_9\}$ ,  $U$  verifies  $TS_9^* - TS_9 \stackrel{?}{=} \Delta T$ , decrypts  $M_9$  and verifies both signature on  $S_6$ . Last,  $U$  generates a key as:  $K_{U-CE} \leftarrow \text{KDF}(VC_{CE} || VC_U || r_{CE}' || r_{AZ}' || r_U')$ .

**Step 10.**  $CE$ :  $CE$  computes a key as:  $K_{CE-U} \leftarrow \text{KDF}(VC_{CE} || VC_U || r_{CE}' || r_{AZ}' || r_U')$ .

### 3.8. User usability maintaining phase

#### Protocol 3: For Handling usability issues in case of lose of private key by user:

This phase provides a solution to address *usability* problem that arises when any friendly enemy (or any insider) deletes the user's private key from the mobile device, or the user loses their private key in any other way. In that situation, the user's system usability is compromised, and they need to recover their private key to continue operating on the system. Table 7 shows the working of this phase. The user Usability Maintaining Phase performs over the public channel as follows:

- User generates  $n_U$ , bio-template  $\Delta$  and computes  $X_1 = \text{Hash}(\Delta || n_U || K_{Pub_{old}}^U)$ , and generates  $M_1 = \text{Enc}(DID_U, n_U, X_1, \Delta, TS_1) K_{Pub}^{AS}$ . Sends  $M_1$  to AS.
- Upon receiving  $M_1$  from user, AS verifies  $TS_1^* - TS_1 \stackrel{?}{=} \Delta T$ , decrypts  $M_1$ , resolves  $DID_U$ , untie  $Z_U$  and retrieves  $\{\Delta', K_{em}, K_{Pub_{old}}^U\}$ . Verifies  $\Delta - \Delta' \leq \text{Threshold}$ ,  $X_1^* = \text{Hash}(\Delta || n_U || K_{Pub_{old}}^U) \stackrel{?}{=} X_1$ , Compute  $\{K_{Pr_{old}}^U, K_{Pub_{old}}^U\} \leftarrow \text{KDF}(\Delta')$ , generate  $n_{AS}$  and  $\text{Sign}_{Gen}(S_1) = (n_{AS}) K_{Pr}^{AS}$ ,  $M_2 = \text{Enc}(K_{Pr_{old}}^U, S_1, TS_2) K_{em}$ . Sends  $M_2$  to U.
- After receiving  $M_2$  from AS, user decrypts  $M_2$  using  $K_{em}$ , verifies  $TS_2^* - TS_2 \stackrel{?}{=} \Delta T$   $S_1$  using  $K_{Pub}^{AS}$ , computes  $\{K_{Pr_{new}}^U, K_{Pub}^{U_{new}}\} \leftarrow \text{KDF}(\Delta' || K_{Pr_{old}}^U)$ ,  $DID_{U_{new}} = \text{Hash}(K_{Pub}^{U_{new}})$ . Generates  $M_3 = \text{Enc}(DID_U, DID_{U_{new}}, K_{Pub}^{U_{new}}, Auth_{cnt}, TS_3) K_{Pub}^{AS}$ . Sends  $M_3$  to AS.
- Upon receiving  $M_3$  from user, AS decrypts  $M_3$ , verifies  $TS_3^* - TS_3 \stackrel{?}{=} \Delta T$ , binds  $\{DID_U, DID_{U_{new}}, K_{Pub}^{U_{new}}, Auth_{cnt}\}$  for user  $U_i$ .

Upon receiving the old DID, new DID, new public key, and current authentication counter from the user, the authentication server replaces

the old DID binding with the new DID binding using these parameters through the blockchain administrator.

## 4. Security analysis of proposed scheme

In this section, we present the formal security analysis of our proposed scheme. In this regard, we use Random Oracle Model (ROR) [21]. In addition, here we also provide a brief description to show how our proposed scheme can satisfy the security properties relevant to the digital twin-based manufacturing system.

### 4.1. Formal security analysis

Now, to prove that the proposed scheme achieves *Secure-AKE* with the help of ROR, here we consider a polynomial-time adversary  $\mathcal{A}$  who interacts with the  $j^{\text{th}}$  participant instance  $\delta_X^j$  of any system model (ref. Fig. 1) entity  $X$ . Using ROR, we can prove that  $\mathcal{A}$  with the oracle capabilities (i.e. *Reveal*, *Send*, *Eavesdrop*, *CorruptUserAgent*, *CorruptUserIdentity*, *Sign*, *CorruptUserBio* and *Test*) can not distinguish between the retrieved values (c) and the original key computed between the parties. Following are the essential preliminaries used to discuss the formal security analysis:

- **Random Oracles and Complexity Assumptions:** We use a pseudo-random one-way hash function  $\mathcal{H}(M)$  to achieve the integrity and use public key encryption  $\mathcal{E}(k_{pub}, M)$  with private key decryption  $\mathcal{D}(k_{pr}, M)$  to achieve confidentiality and user privacy in the proposed work. Assume that a polynomial time adversary  $\mathcal{A}$  captures message  $M_i$  then the oracle computes  $R_i = \mathcal{H}(M_i)$  where  $R_i$  is the fixed size irreversible value related to message  $M_i$ . It is stored in the list  $L$  as the  $(M_i, R_i)$  pair with oracles. Through the following definition for the  $\mathcal{H}$  function, we prove that for any polynomial-time adversary  $\mathcal{A}$ , it is computationally infeasible to win the related game in non-negligible time as defined.

**Definition 1.** Let  $Adv_H$  present the success of an adversary  $\mathcal{A}$  in differentiating two functions  $f_n$  and  $f_n^*$  such that  $Adv_H = |Pr[f_n = 1] - Pr[f_n^* = 1]|$  presents the distinguishing capacity of  $\mathcal{A}$  for  $f_n$  and  $f_n^*$  in polynomial time. For any  $\mathcal{A}$  who performs  $i$  oracle queries for  $j$  times then the  $f_n(i, j, \delta)$  is secure if  $\mathcal{A}$  can distinguish an output of  $f_n$  and  $f_n^*$  with  $Adv_H \geq \delta$ .

**Initialisation:** Let challenge  $C$  define functions  $f_n^0$  and  $f_n^1$  and selects randomly among them to interact with  $\mathcal{A}$  where  $f_n^0$  is a pseudo random function  $\mathcal{H}$  and  $f_n^1$  is truly random function  $\mathcal{R}$ .

**Adversary Training:** In this step,  $\mathcal{A}$  sends  $i$  oracle queries  $q_1 \dots q_i$  to  $C$ . The  $C$  responds to these queries by sending  $f_n^b(q_1) \in \{0,1\}^l$ , where  $f_n^b$  can be either  $f_n^0$  or  $f_n^1$ .

**Adversary Guess** In this step, an adversary  $\mathcal{A}$  tries to guess the value of  $b$  in  $f_n^b$  as  $b^*$  and if  $\mathcal{A}$  can correctly guess that  $C$  has used either  $f_n^0$  or  $f_n^1$  then it wins the game. Hence the winning probability of  $\mathcal{A}$  in guessing correct  $f_n^b$  is  $Adv_{A,f_n} = |Pr[b = b^*] - 1/2|$ .

As per the assumption of a pseudo-random function, we can argue that it is impossible for any adversary  $\mathcal{A}$  to win this game within a polynomial time with a non-negligible advantage.

- **Oracle Participants:** In the proposed scheme, we propose two protocols, The *Protocol 1* consists of user  $U$ , authentication server  $AS$ , and authorisation server  $AZ$  while *Protocol 2* consists of user  $U$ , authentication server  $AS$ , authorisation server  $AZ$ , and the command engine  $CE$ . Let  $\delta_U^i$ ,  $\delta_{AS}^j$ ,  $\delta_{AZ}^k$  and  $\delta_{CE}^l$  represent the oracles of  $U$ ,  $AS$ ,  $AZ$ , and  $CE$  with instances  $i$ ,  $j$ ,  $k$ , and  $l$  respectively.
- **Freshness of the Oracles:** We can say that  $\delta_U^i$ ,  $\delta_{AS}^j$ ,  $\delta_{AZ}^k$  and  $\delta_{CE}^l$  are fresh oracles if the reveal oracle query  $R$  by an adversary  $\mathcal{A}$  does not generate the correct key between  $\delta_U^i$  and  $\delta_{AZ}^k$  or between  $\delta_U^i$  and  $\delta_{CE}^l$ .
- **Oracle Partnering:** Oracle instances  $\delta_x^m$  and  $\delta_y^n$  are partners if:
  - They share a common session identification ( $sid$ ) with complete mutual authentication. The  $sid$  represents the log for the communicated messages before the acceptance state.
  - They are in the same acceptance state.
  - They satisfy partner identification.
- **Accepted States:** An oracle instance  $\delta_x^m$  communicates the last message with  $\delta_y^n$  and reaches to the acceptance state and generates a common fresh session identifier  $sid$  as a log of all the communicated messages among them.
- **Polynomial Time Adversary:** Let us assume that the adversary  $\mathcal{A}$  have all the capabilities defined in 2.6 and be able to perform queries discussed below:
- **Adversary Model and Adversary Capabilities:** The modelling of  $\mathcal{A}$  is based on capabilities of the adversary discussed in section 2.6. Through the following oracle queries, adversary  $\mathcal{A}$  tries to perform either an active attack or a passive attack.

**Reveal  $\mathcal{R}(\delta_x^m)$ :** This query provides a session key to the  $\mathcal{A}$  shared between the oracle instance  $\delta_x^m$  and its partner.

**Send  $\mathcal{S}(\delta_x^m, msg)$ :** With the help of this query, an adversary  $\mathcal{A}$  receives a response from the  $\delta_x^m$  and tries to perform an active attack.

**Eavesdrop  $\mathcal{E}(\delta_x^m, \delta_y^n)$ :** With the help of this query, an adversary  $\mathcal{A}$  monitors traffic between  $\delta_x^m$  and its partner  $\delta_y^n$ .

**CorruptUserAgent  $\mathcal{C}_R(\delta_x^m)$ :** With the help of this query, an adversary  $\mathcal{A}$  is able to get the values stored with the user agent. Also, an adversary can delete the private key.

**CorruptUserIdentity  $\mathcal{C}_R(\delta_x^m)$ :** With the help of this query, an adversary  $\mathcal{A}$  receives the DID for the  $\delta_x^m$  and tries to behave as a trusted participant of the system.

**Sign  $\mathcal{S}(M_k, \delta_x^m)$ :** With the help of this oracle query, an adversary  $\mathcal{A}$  receives the valid signature  $S_k$  for any message  $M_k$  signed using private key of the entity  $\delta_x^m$ .

**CorruptUserBio  $\mathcal{C}_R(\delta_x^m)$ :** With the help of this query, an adversary  $\mathcal{A}$  receives bio-template  $\Delta$  of the  $\delta_x^m$  and tries to perform reverse

engineering to extract more information (i.e. role, age, etc.) related to the user.

**Test  $\mathcal{T}(\delta_x^m)$ :** With the help of this query, an adversary  $\mathcal{A}$  tries to guess the output of unbiased coin  $c$ . Based on the output of  $c = 0$  or  $c = 1$ , an  $\mathcal{A}$  receives either a random value or the original key, respectively. Whenever  $\delta_x^m$  reaches the acceptance state, the  $\mathcal{A}$  sends this query and tries to differentiate between the random value and the real key. Except for the above two cases, an instance  $\delta_x^m$  returns NULL for any other case.

- **Session Key Semantic Security for Protocol with DID:** The session key semantic security of the protocol depends on the capacity of the  $\mathcal{A}$  to distinguish between any random output and actual session key output.

**Theorem 1.** Let  $Adv_A$  define the advantage of adversary  $\mathcal{A}$  in receiving the correct session key by guessing the correct value of the coin as  $c'$  during the  $\mathcal{T}$  query. If we can prove that the  $Adv_A$  is negligible for the proposed work then we can say that the proposed work is secured from the random oracle-enabled adversary  $\mathcal{A}$ . We can define  $Adv_A$  as follows,

$$Adv_A(\mathcal{A}(t)) = 2 * Pr[SC] - 1 \quad (1)$$

OR

$$Adv_A(\mathcal{A}(t)) = 2 * Pr[c' = c] - 1 \quad (2)$$

**Theorem 2.** Let  $Adv_A^{ECDSA}(t)$ ,  $Adv_A^{EUF-CMA}(t)$  show an advantage of the polynomial time adversary  $\mathcal{A}(t)$  in breaking the digital signature and performing the existential unforgeability against chosen message attack (EUF-CMA) [22] over the Verifiable Credentials signed by the genuine signer. For any polynomial time adversary  $\mathcal{A}$  and challenge  $C$ , if challenger  $C$  generates a valid key pair as  $(K_{Pr}^C, K_{Pub}^C) \leftarrow Gen(1^\lambda)$  where  $\lambda$  is a secret parameter. A challenger  $C$  sends  $K_{Pub}^C$  to the  $\mathcal{A}$  and now  $\mathcal{A}$  obtains signatures  $\{s_1, s_2, \dots, s_n\}$  for message set  $\mathcal{M} = Sgn\{m_1, m_2, \dots, m_n\}$  from the  $C$ . Now, adversary  $\mathcal{A}$  generates a pair  $(m_k^*, s_k^*)$  where  $m_k^* \notin \mathcal{M}$ , so we can say the digital signature of the message is secured against EUF-CMA if for any polynomial-time adversary  $\mathcal{A}$ ,

$$Adv_A^{EUF-CMA}(t)(\mathcal{A}) = Pr[Vrf(K_{Pub}^C, m_k^*, s_k^*)] = 1 \quad (3)$$

is negligible in  $\lambda$ .

**Theorem 3.** Let  $Adv_A^{ECDLP}(t)$ ,  $Adv_A^{EUF-CMA}(t)$ ,  $Adv_A^{ECDSA}(t)$ ,  $Adv_A^{GAN}(t)$  show an advantage of the polynomial time adversary  $\mathcal{A}(t)$  in breaking the session key, digital signature and performing reverse engineering on bio-template generated using GAN for the proposed protocol then we can define the session key breaking probability  $Adv_A^{SK}(t)$  in polynomial time  $t$  as:

$$Adv_A^{SK}(t)(\mathcal{A}) \leq \frac{q_h^2}{l_h} + 2 * Adv_A^{ECDLP}(t) + 2 * Adv_A^{ECDSA}(t) + 2Adv_A^{EUF-CMA}(t) + Adv_A^{GAN}(t) \quad (4)$$

**Proof.** With the help of Theorem 1, Theorem 2, and Theorem 3 we have defined seven security games  $Game_j$  ( $j = 0, 1, 2, 3, 4, 5, 6$ ) to prove that the proposed protocol is secured against a polynomial-time adversary  $\mathcal{A}$ . The game starts with the  $Game_0$  and finishes at  $Game_6$ . We also define that  $SC_j$  presents the correct guess for the coin  $c$  in each game  $Game_j$  through the  $\mathcal{T}$  query by  $\mathcal{A}$ .

- **Game<sub>0</sub>:** The  $Game_0$  presents an identical and attacks by adversary  $\mathcal{A}$  against proposed protocol and  $\mathcal{A}$  tries to predict the value of  $c$ .

$$Adv_A^{SK}(t)(\mathcal{A}) \leq 2 * Pr[SC_0] - 1 \quad (5)$$

- **Game<sub>1</sub>**: In this game, an adversary  $\mathcal{A}$  performs **Eavesdrop**  $\mathcal{E}(\delta_U^i, \delta_{AS}^j, \delta_{AZ}^k)$  and **Eavesdrop**  $\mathcal{E}(\delta_U^i, \delta_{AS}^j, \delta_{AZ}^k, \delta_{CE}^l)$  queries and performs passive attack to trace the communication between  $\delta_x^m$  and  $\delta_y^n$ . In the proposed protocol, **Protocol 2** involves communication between four entities and **Protocol 1** involves communication between three entities. An  $\mathcal{A}$  traces all these communication and tries to compute or guess  $K_{U-CE}$  or  $K_{U-AZ}$ . Since the computation of these two keys involves secret parameter  $VC_x$  (presents verifiable credential of entity  $x$ ) and random parameters  $r_x$  (presents random number generated by entity  $x$ ) these parameters are not communicated in plain text. Since it is nearly impossible for  $\mathcal{A}$  to guess all these parameters in a polynomial time,  $\mathcal{A}$  can not compute any of the keys. Hence, both games ( $Game_0$ , and  $Game_0$ ) are not possible to distinguish, and we can say that:

$$Adv_A^{SK}(Game_1) = Adv_A^{SK}(Game_0) \quad (6)$$

OR

$$Pr[SC_1] = Pr[SC_0] \quad (7)$$

- **Game<sub>2</sub>**: In this game, an adversary  $\mathcal{A}$  performs active attack using queries  $H(M)$  and **Send**  $S(\delta_x^m, msg)$ . With the help of  $H(M)$ , adversary  $\mathcal{A}$  receives hash of message  $M$  and with the help of **Send**  $S(\delta_x^m, msg)$ ,  $\mathcal{A}$  communicates entity associated with oracle  $\delta_x^m$  through message  $msg$ . With the help of computed hash values and received communication, an adversary  $\mathcal{A}$  tries to compute the keys  $K_{U-CE}$  and  $K_{U-AZ}$ . Since we use a random salt-based key derivation function, the adversary cannot compute the correct key based on computed hash values. An adversary  $\mathcal{A}$  can never distinguish between the value of computed hashes and  $K_{U-CE}$  and  $K_{U-AZ}$ . An adversary  $\mathcal{A}$  validates collision, and as per the definition of the birthday paradox, the collision probability for the oracle  $H$  is at most  $\frac{q_h^2}{l_h}$ . There are three other challenges that  $\mathcal{A}$  needs to solve. The first challenge is  $\mathcal{A}$  need to guess the DID value of entities. Now let us assume that  $\mathcal{A}$  somehow got the DID value, then needs access to the permissioned blockchain to resolve it and get the entity's public key, so it's a second challenge. And even if  $\mathcal{A}$  resolves DID and somehow got the entity's public key, needs to solve the ECLDP problem, which is computationally infeasible to solve in polynomial time. Hence, the proposed authentication scheme is secured from Type 3 adversary (sec. 2.6). So:

$$Adv_A^{SK}(Game_2) - Adv_A^{SK}(Game_1) \leq \frac{q_h^2}{l_h} + 2 * Adv_A^{ECDLP}(t) \quad (8)$$

- **Game<sub>3</sub>**: In this game, an adversary performs **CorruptUserBio**  $C_R(\delta_x^m)$ . An adversary  $\mathcal{A}$  receives bio-template  $\Delta$  of any user associated with oracle  $\delta_x^m$ . This is just an assumption that a trusted authentication server is hacked and  $\mathcal{A}$  somehow got the private key of the authentication server to get the  $\Delta$ . Upon receiving a  $\Delta$ , an adversary tries to perform reverse engineering on it and tries to get the old private key of the user (that was computed using a random salt-based key derivation function). As shown in section 3.2, we have used a noise-based generative adversarial network to generate the bio-template. Hence, it is impossible to break the user's privacy and get more information about the user from an adversary. Upon receiving  $\Delta$ ,  $\mathcal{A}$  can not compute any previous session keys because of the involvement of random numbers and verifiable credentials in their computations. Hence, the proposed authentication scheme is secured from Type 2 adversary (sec. 2.6). Thus,

$$Adv_A^{SK}(Game_3) - Adv_A^{SK}(Game_2) \leq Adv_A^{GAN}(t) \quad (9)$$

- **Game<sub>4</sub>**: In this game, an  $\mathcal{A}$  performs **CorruptUserIdentity**  $C_R(\delta_x^m)$  through which  $\mathcal{A}$  gets DID of the user associated with oracle  $\delta_x^m$ . Based on received DID,  $\mathcal{A}$  can not be communicated with

the authentication server because  $\mathcal{A}$  have neither  $(VC_U)K_{Pr}^{AS}$  nor  $(VC_U)K_{Pr}^{Gov}$ . Based on DID, an adversary  $\mathcal{A}$  can neither perform any signature verification because to solve signature verification  $\mathcal{A}$  needs to solve polynomial time computationally infeasible ECDSA problem nor compute any of the session keys. Hence, the proposed authentication scheme is secured from Type 1 adversary (sec. 2.6). Thus:

$$Adv_A^{SK}(Game_4) - Adv_A^{SK}(Game_3) \leq 2 * Adv_A^{ECDSA}(t) \quad (10)$$

- **Game<sub>5</sub>**: In this game, an  $\mathcal{A}$  performs **Sign**  $S(M_k, \delta_x^m)$  through which  $\mathcal{A}$  receives a valid signature  $S_k$  for any message  $M_k$  communicated with any system entity associated with oracle  $\delta_x^m$ . Through this  $\mathcal{A}$  can try to forge the  $(VC_x)K_{Pr}^y$  and can perform **EUFCMA** over it. In the proposed scheme, a private key  $K_{Pr}^y$  is not shared with any entity and it is securely stored over the device. Since  $VC_x$  is highly random in nature and protected by one way hash function. As per Theorem 3, for any polynomial time adversary, it is not possible to generate a valid pair  $(VC_x, (VC_x)K_{Pr}^y)$  based on received message-signature set  $(M_k^*, S_k^*)$  and hence:

$$Adv_A^{SK}(Game_5) - Adv_A^{SK}(Game_4) \leq 2 * Adv_A^{EUFCMA}(t) \quad (11)$$

- **Game<sub>6</sub>**: In this game, an  $\mathcal{A}$  performs **CorruptUserAgent**  $C_R(\delta_x^m)$  through which  $\mathcal{A}$  receives all the values stored in the mobile device including user private key, verifiable credentials, DID and so on. An  $\mathcal{A}$  also intercepted the messages communicated between the entities to compute the session key. Here, we consider that based on this query,  $\mathcal{A}$  tries to compute the session key, not try to establish a session, and this is a valid consideration. The computed session keys are  $K_{AZ-U} \leftarrow KDF(VC_U || VC_{AZ} || r'_U || r'_{AZ} || n_{AS})$  and  $KDF(VC_{CE} || VC_U || r_{CE} || r'_{AZ} || r'_U)$ . The key computations involve random numbers, the authentication server's nonce, and other entities' verifiable credentials. Since it is not possible for the  $\mathcal{A}$  to guess all the correct values in the polynomial time with the correct seed value used by the random key derivation function, we can say that  $\mathcal{A}$  can not distinguish between the random string and original session key thus it only remains to guess the correct value of  $c$  to win the game. So winning probability of the  $Game_6$  is:

$$Adv_A^{SK}(Game_6) = \frac{1}{2} \quad (12)$$

From the equations (4), (5), (10), and (11),

$$\begin{aligned} \frac{1}{2} Adv_A^{SK} &= \frac{1}{2} Adv_A^{SK}(Game_0) = Adv_A^{SK}(Game_1) = \\ &Adv_A^{SK}(Game_1) - Adv_A^{SK}(Game_6) \end{aligned} \quad (13)$$

After applying triangle equality with multiplying both the sides by 2 on equation 12 and with the help of equations (5) - (9), we can derive that:

$$\begin{aligned} Adv_A^{SK}(t)(\mathcal{A}) &\leq \frac{q_h^2}{l_h} + 2 * Adv_A^{ECDLP}(t) + 2 * Adv_A^{ECDSA}(t) \\ &+ 2 * Adv_A^{EUFCMA}(t) + Adv_A^{GAN}(t) \end{aligned} \quad (14)$$

Following this formal proof, the proposed protocol achieves authenticated key exchange with session key security.

#### 4.2. Informal security analysis

In this section, we discuss the analytical discussion for the proposed scheme based on security properties achieved through it.

- **User Empowered Authentication**: This property ensures that the user is not dependent on any central authority for identity generation and registration. In the proposed protocol, we use a decentralised identifier (DIDs) in which users generate their identity

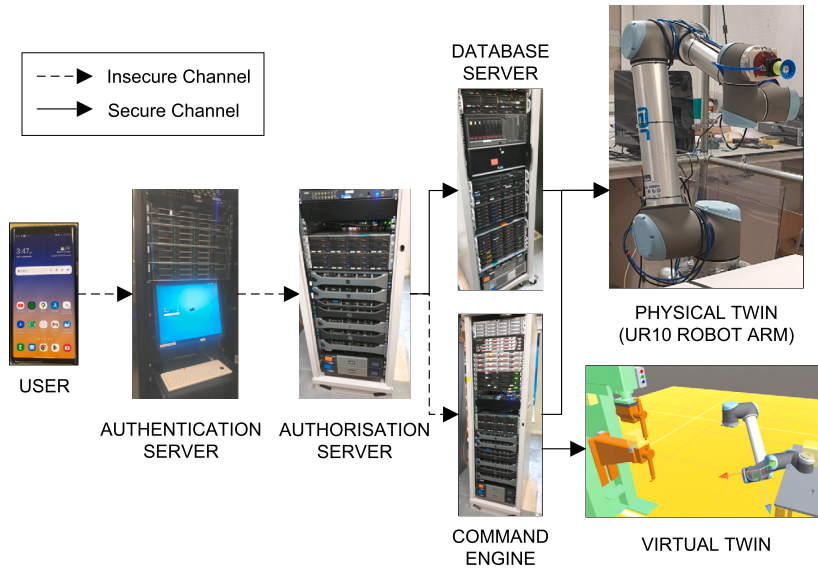


Fig. 5. Experimental Setup of the Proposed Framework.

and key pair by themselves and can register in a permissioned blockchain. Upon generation of DID, users can communicate with other system entities by verifying verifiable credentials and token authorisation.

- **Auditability:** This property ensures that only an authenticated system user can audit the authentication attempts by an adversary who has stolen a user's DID. In the proposed scheme, during new user DID registration phase in section 3.3.1, the User sets the authentication counter as  $0 \leftarrow Auth_{cnt}$  and binds it to the permissioned blockchain. During key generation, upon successful signature verification, the authentication server increments this counter as  $Auth_{cnt} \leftarrow Auth_{cnt} + 1$  and sends it to the user for verification. The system user compares the received  $Auth_{cnt}$  with the  $Auth_{cnt}$  available with the user agent and ensures that it is precisely one value lesser than the  $Auth_{cnt}$  which is received from the authentication server. If the validation is successful then the user updates its counter with the received counter and continues key derivation. If it has a difference more than once, then the user detects that some adversary has tried to perform authentication using their DID. Hence, auditability property is achieved in the proposed work.
- **User Revocation:** This security property ensures that if an authenticated user is revoked by the system, then he/she should not be able to derive the session key with any system entity or/and should not have access to the services. In the proposed scheme, an accumulator manager operating by the authentication server provides  $MW_U$  to each valid system user. The  $MW_U$  is dynamic and based on any updates (i.e. new user added or any user revoked); existing (i.e. not revoked) system user receives the latest  $MW_U$  from the accumulator manager to prove their membership. Hence, the proposed scheme achieves the accumulator manager's user revocation and run time dynamicity, making a more accurate revocation system.
- **Security from Replay Attack:** In a replay attack, an adversary uses old messages to reestablish a session with the system entities and tries to establish a session key. The session keys computed in the proposed scheme as  $K_{CE-U} \leftarrow KDF(VC_{CE} || VC_U || r_{CE}' || r_{AZ}' || r_U')$  and  $K_{AZ-U} \leftarrow KDF(VC_U || VC_{AZ} || r_U' || r_{AZ}' || n_{AS})$ . The session keys contain random numbers generated during each session, adding randomness to each new session key. In the proposed scheme, each message has a time stamp verified by the receiver with the time-stamp threshold. Hence, the proposed work is secured against replay attacks.

- **Security from User Impersonation:** In the user impersonation attack, an adversary  $\mathcal{A}$  tries to pretend as a valid system user and establish a valid session key. To establish a key, user needs to generate  $M_1 = Enc(DID_U, (n_U)K_{pr}^U, MW_U)K_{pub}^{AS}$ . To generate valid  $M_1$ , the user must know the user's DID, the user's private key, the random nonce generated by the user, and the membership witness of the user. The user can only generate any random nonce out of these parameters. In continuation to session key generation, the user also needs to provide  $((VC_U)K_{pr}^{AS})K_{pr}^U$  signed using the private key to other system entities. Hence, without knowledge of all these parameters, it is impossible for  $\mathcal{A}$  to impersonate a valid system user. Hence, the proposed scheme achieves security from impersonation attacks.
- **User Privacy using Irreversible Bio-template:** In the proposed system, the user generates bio-template  $\Delta$  and binds it with the permissioned blockchain to achieve usability. The bio-template  $\Delta$  here is irreversible, and it is generated using CycleGAN, which uses an unpredictable random noise mechanism inside the generator. Hence, it will be difficult for any adversary to guess user role, user age or any other parameter associated with user privacy who somehow receives or regenerates bio-template  $\Delta$ . As discussed and proved in section 3.2, user privacy is achieved in the proposed protocol.
- **Usability using Bio-Metric DID:** This security property assures that if any frenemy deletes the user's private key or the user's private key is deleted in any other way, then also the user can prove himself as a legitimate entity, in the proposed scheme, we use biometric DID where the system user generates a bio-template (as discussed in section 3.2) and binds it into the permissioned blockchain after encrypting it using the public key of the authentication server. In the case of key loss, the system user provides a bio-template to the authentication server. The authentication server validates this bio-template, generates the old private key and public key pair using it, and provides it to the user for generating a new public key and private key and to register new DID. As per detailed discussion in section 3.8, the proposed system achieves user usability and hence, the proposed authentication scheme is secured from Type 4 adversary (sec. 2.6).
- **Secure Session Key Generation:** In the proposed work, session keys are generated as  $K_{CE-U} \leftarrow KDF(VC_{CE} || VC_U || r_{CE}' || r_{AZ}' || r_U')$  and  $K_{AZ-U} \leftarrow KDF(VC_U || VC_{AZ} || r_U' || r_{AZ}' || n_{AS})$ . We use a random salt-based key derivation function for the key generation that takes random numbers and secures verifiable credentials as input. For

**Table 8**  
Comparison of Security Properties Achieved.

| Security Properties                          | Parameswarath et al. [12] | Li et al. [24] | Proposed Solution |
|----------------------------------------------|---------------------------|----------------|-------------------|
| User Empowered Authentication                | YES                       | YES            | YES               |
| Auditability                                 | NO                        | NO             | YES               |
| Revocation                                   | NO                        | Yes            | YES               |
| Security from Replay Attack                  | YES                       | Yes            | YES               |
| Security from User Impersonation             | YES                       | NO             | YES               |
| User privacy using Irreversible Bio-template | NO                        | NO             | YES               |
| Usability using Bio-Metric DID               | NO                        | NO             | YES               |
| Secure Key Generation                        | YES                       | YES            | YES               |

any polynomial time adversary, it is computationally infeasible to guess the random numbers, predict the correct salt used by the function, or generate the valid, verifiable credentials of the system user and other entities to generate the session key. Hence, the proposed system achieves secure key generation.

- **Complexity of Reverse Engineering:** Analysing the complexity and feasibility of reverse engineering our transformation process, particularly for  $\delta_1$  and  $f^G$ , reveals that the inherent properties of techniques like CycleGAN significantly elevate the difficulty of mapping transformed data back to its original form. The non-linear and complex nature of these transformations presents substantial computational challenges, suggesting a reduced risk of successful reverse engineering attempts.
- **Future Directions and Security Enhancements:** In response to potential security concerns, we propose advanced transformation algorithms and additional security layers as potential modifications to enhance our model's robustness. The survey of literature indicates a lack of extensive exploration in reversing transformations in similar contexts, denoting it as a critical area for future research. This highlights the strength of our authentication system, especially under the Game3 scenario with an assumed compromised authentication server, establishing a significant barrier against adversarial attacks.

## 5. Discussion

In this section, we first compare our proposed scheme with respect to the other existing DID-based works in terms of the security properties. Next, we measure the performance of the proposed GAN-based privacy preservation scheme using Sokoto Coventry Fingerprint Dataset (SOCOFing). Subsequently, we present the computation cost of the proposed scheme using publicly available DECO protocol on our system for integration with blockchain [23].

Furthermore, we present practical case studies involving the UR10 robot arm's digital twin. The digital twin of the UR10 robot arm plays a crucial role in automated quality control within manufacturing environments. Leveraging our blockchain-based DID system, the scenario demonstrates how precise measurements and inspections are securely managed. This ensures that only authorised personnel can make critical adjustments or access sensitive data, thereby enhancing the security and accuracy of quality control processes.

Another case study involves the application of the digital twin in customised production runs. Here, our authentication system allows for secure and rapid adjustments to the UR10 robot m, catering to varying product specifications. This showcases the system's adaptability and efficiency in handling diverse manufacturing demands. Additionally, a unique scenario in a training environment is presented, where the digital twin simulates emergencies. The secure access provided by the DID system ensures that sensitive training data for emergency response protocols is developed and tested in a controlled and secure manner.

The digital twin architecture offers several advantages over direct robot interaction. It enhances predictive analytics, remote monitoring and operation and integrates seamlessly with broader digital systems.

**Table 9**  
Operation Wise Execution Time.

| Entity                   | $T_h$ | $T_{sgn}$ | $T_{verf}$ | $T_{enc}$ | $T_{dec}$ |
|--------------------------|-------|-----------|------------|-----------|-----------|
| User                     | 1.075 | 1.117     | 1.134      | 1.098     | 1.084     |
| Servers + Command Engine | 0.012 | 0.039     | 0.080      | 0.028     | 0.032     |

**Notations:**  $T_h$ : Hash Time,  $T_{sgn}$ : Signature Generation Time,  $T_{verf}$ : Signature Verification Time,  $T_{enc}$ : Encryption Time,  $T_{dec}$ : Decryption Time.

These capabilities significantly elevate its strategic utility in modern manufacturing.

### 5.1. Performance comparison

In this section, we compare the proposed scheme with other related schemes based on DID technology, such as the schemes of [12] [24]. In order to analyse the performance of the proposed scheme, particularly on the security front, our scheme has been compared with [12], and [24], by considering the major security properties such as *user empowered authentication*, *auditability*, *user revocation*, *security from replay attack*, *security from user impersonation*, *user privacy using irreversible bio-template*, *usability* and *secure key generation* (shown in Table 8). Table 8 shows that existing solutions cannot achieve certain properties such as *auditability*, *user revocation*, *User Privacy using Irreversible Bio-template* and *Usability using Bio-Metric DID*. Parameswarath et al. [12] presented an authentication protocol for electric vehicle charging using DID but doesn't provide any solution for *auditability* and *user revocation*. Similarly, Li et al. [24] presented a double-layer blockchain and decentralised identifiers that assisted secure registration and authentication for vehicular ad-hoc networks. However, Li et al. [24] does not provide any solution for the aforementioned fundamental problems related to DID technology.

### 5.2. Bio-template uniqueness and accuracy

Now we evaluate our proposed GAN-based privacy preservation scheme using Sokoto Coventry Fingerprint Dataset (SOCOFing) [25]. The SOCOFing contains 6000 fingerprints of 600 individuals. In this experiment, we aim to determine up to what extent our proposed scheme can generate stable  $\Delta$  considering divination on fingerprints (our model input). Hence, we apply different Gaussian noises 2 ~ 10% to raw fingerprints to generate our testing dataset (e.g. fingerprint and sensor ageing challenge). Next, we use testing instances (noisy fingerprint) and collect  $\Delta'$ . To ensure the generation of users' primary key pairs, we measure to what extent  $\Delta$  and  $\Delta'$  are similar As shown in Fig. 6. In this regard, we use the cosine similarity to measure the similarity of  $\Delta$  and  $\Delta'$ , and decide the relative threshold value. Our experiment shows that our system can ensure 100% similarities of 6000 fingerprints.

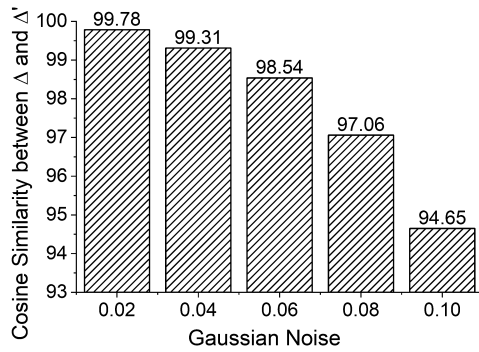
### 5.3. Computation cost

To assess the performance of the proposed work in terms of computation cost, we consider a digital twin system available at the Advance

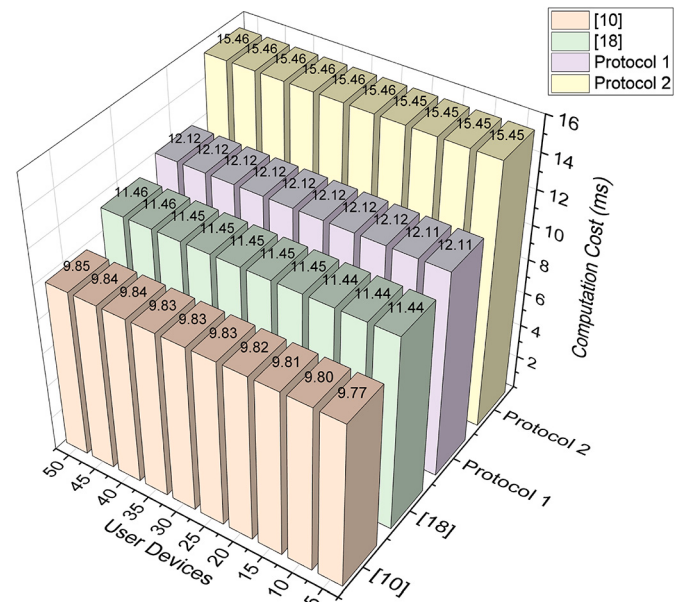
**Table 10**  
Overall Computation Cost.

| Entity                                       | #Hash | #Sign_Gen | #Sign_Verify | #Enc | #Dec | Computation Cost |
|----------------------------------------------|-------|-----------|--------------|------|------|------------------|
| <b>Protocol 1 : Secure Data Access</b>       |       |           |              |      |      |                  |
| User                                         | 2     | 2         | 2            | 3    | 2    | 12.114 ms        |
| Authentication Server                        | 1     | 1         | 1            | 2    | 1    | 0.219 ms         |
| Authorisation Server                         | 1     | 1         | 2            | 1    | 3    | 0.335 ms         |
| <b>Total Computation Cost</b>                |       |           |              |      |      | <b>12.668 ms</b> |
| <b>Protocol 2 : Secure Command Execution</b> |       |           |              |      |      |                  |
| User                                         | 2     | 3         | 3            | 3    | 3    | 15.449 ms        |
| Authentication Server                        | 1     | 1         | 1            | 2    | 1    | 0.219 ms         |
| Authorisation Server                         | 1     | 4         | 4            | 3    | 4    | 0.7 ms           |
| Command Engine                               | -     | 1         | 2            | 1    | 1    | 0.259 ms         |
| <b>Total Computation Cost</b>                |       |           |              |      |      | <b>16.627 ms</b> |

**Notations:** # Number of, **Hash:** Hash Operation, **Sign\_Gen:** Signature Generation Operation, **Sign\_Verify:** Signature Verification Operation, **Enc:** Encryption Operation, **Dec:** Decryption Operation.

**Fig. 6.** Cosine Similarity between  $\Delta$  and  $\Delta'$  considering different Gaussian Noises on fingerprints.

Manufacturing Research Center (AMRC) of the University of Sheffield [26]. In this regard, Universal Robots 10e (UR10e) with the 500 Hz frequency has been considered as a physical twin. The robot communicates with the servers through Modbus TCP and Ethernet/IP. UR10 robot interfaces are based on the Robot Operating System (ROS) that sends data with the sampling time of  $8 \times 10^{-3}$  sec [27] to the database server. On the other hand, we conducted simulations of the cryptographic operations used in the proposed scheme and [12], [24] on a OnePlus 9 Pro with Octa-core (1x2.84 GHz Cortex-X1 & 3x2.42 GHz Cortex-A78 & 4x1.80 GHz Cortex-A55 CPU and Adreno 660 GPU over Android 11, OxygenOS 12 with Qualcomm SM8350 Snapdragon 888 5G (5 nm) (operating as a user device), and an Intel(R) Core(TM) i9-6500 CPU @ 3.20 GHz with 128 GigaBytes of RAM servers (operating as an authentication server, authorisation server, and command engine) available with the university. Fig. 5 presents the experimental setup of the proposed work. Implementation of the fundamental cryptographic operations included publicly available Python libraries such as *starkbank-ecdsa* (for signature generation and verification) [28], *hashlib* (for hashing operation) [29], and *tinyec* (for ECC point multiplication, point summation and point subtraction operations) [30] for the evaluation of execution time of each cryptographic operation involved in the proposed work. There are two key derivation phases in the proposed work. The first key derivation phase inculcates the user, authentication server and authorisation server, while the second key derivation phase inculcates the user, authentication server, authorisation server and command engine. Let us consider  $T_h$ ,  $T_{sgn}$ ,  $T_{vrf}$ ,  $T_{enc}$ ,  $T_{dec}$  represents time (in ms) required for one-way hash computation, ECDSA signature generation, ECDSA signature verification, ECC Encryption, and ECC decryption operation. For ECC Encryption and decryption, we have used Elliptic Curve Cryptography with ElGamal, where each ECC

**Fig. 7.** Average Computation Cost. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

Encryption ( $T_{enc}$ ) operation requires one point multiplication and one point addition. In comparison, each ECC Decryption ( $T_{dec}$ ) requires one point multiplication and one point subtraction operation. The following Table 9 shows the time required by each participating entity for computing each operation. Table 10 shows the computation cost required by each entity during the secure data access (protocol-1) and secure command execution (protocol-2). Fig. 7 presents the average computation cost for different numbers of user devices.

## 6. Conclusions and future work

User empowerment ensures secure data access and efficient command execution in the digital twin. In this article, we have resolved three fundamental problems of DID technology. With the help of bi-template, we presented the solution for the usability problem, while with the help of the authentication counter, we solved the auditability problem. With the help of a zero-knowledge dynamic accumulator, this paper presents a very efficient way to revoke the user for any critical infrastructure. We quantified the security analysis using formal and informal methods while introducing the performance analysis based on

security objectives, bio-template uniqueness and accuracy, and computation cost. This paper presents the first user-empowered security solution for the digital twin using the decentralised identifier and verifiable credentials. In the future, we look forward to solving some of the critical challenges, such as recoverability in the case of lost key and implementing the proposed work over public blockchain technology. In this work, we have considered a single physical twin for the experiment; however, we are also looking forward to deploying the proposed work for a large number of physical twins in the future, considering the presence of a white box adversary.

### CRedit authorship contribution statement

**Chintan Patel:** Methodology. **Aryan Pasikhani:** Methodology. **Prosanta Gope:** Conceptualization, Formal analysis, Writing – review & editing. **John Clark:** Supervision, Writing – review & editing.

### Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Prosanta Gope reports was provided by The University of Sheffield. Prosanta Gope reports a relationship with The University of Sheffield that includes: employment. Prosanta Gope has patent NA pending to NA.

### Data availability

No data was used for the research described in the article.

### Acknowledgement

This work was supported in part by the Engineering and Physical Sciences Research Council (EPSRC) under Award EP/V039156/1.

### References

- [1] M. Shafiq, M. Conroy, R. Doyle, E. Glaesgen, C. Kemp, J. LeMoigne, L. Wang, Draft modeling, simulation, information technology & processing roadmap, *Technol. Area* 11 (2010) 1–32.
- [2] C. Gehrmann, M. Gunnarsson, A digital twin based industrial automation and control system security architecture, *IEEE Trans. Ind. Inform.* 16 (1) (2020) 669–680, <https://doi.org/10.1109/TII.2019.2938885>.
- [3] J. Leng, D. Yan, Q. Liu, K. Xu, J.L. Zhao, R. Shi, L. Wei, D. Zhang, X. Chen Manuchain, Combining permissioned blockchain with a holistic optimization model as bi-level intelligence for smart manufacturing, *IEEE Trans. Syst. Man Cybern. Syst.* 50 (1) (2020) 182–192, <https://doi.org/10.1109/TSMC.2019.2930418>.
- [4] S. Suhail, S. Zeadally, R. Jurdak, R. Hussain, R. Matulevičius, D. Svinovic, Security attacks and solutions for digital twins, Working paper, 8 pages, 6 figures, <https://doi.org/10.48550/arXiv.2202.12501>, Feb. 2022.
- [5] S. Huh, M. Shim, J. Lee, S.S. Woo, H. Kim, H. Lee, Did we miss anything?: Towards privacy-preserving decentralized id architecture, *IEEE Trans. Dependable Secure Comput.* 20 (6) (2023) 4881–4898, <https://doi.org/10.1109/TDSC.2023.3235951>.
- [6] P. Ramanan, D. Li, N. Gebräel, Blockchain-based decentralized replay attack detection for large-scale power systems, *IEEE Trans. Syst. Man Cybern. Syst.* 52 (8) (2022) 4727–4739, <https://doi.org/10.1109/TSMC.2021.3104087>.
- [7] D. Yaga, P. Mell, N. Roby, K. Scarfone, Blockchain technology overview, *arXiv preprint*, arXiv:1906.11078, 2019.
- [8] F. Tao, F. Sui, A. Liu, Q. Qi, M. Zhang, B. Song, Z. Guo, S.C.-Y. Lu, A.Y.C. Nee, Digital twin-driven product design framework, *Int. J. Prod. Res.* 57 (12) (2019) 3935–3953, <https://doi.org/10.1080/00207543.2018.1443229>.
- [9] Y. Kortessniemi, D. Lagutin, T. Elo, N. Fotiou, Improving the privacy of iot with decentralised identifiers (dids), *J. Comput. Netw. Commun.* (2019) 2019.
- [10] X. Fan, Q. Chai, L. Xu, D. Guo, Diam-iot: a decentralized identity and access management framework for Internet of things, in: *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, 2020, pp. 186–191.
- [11] M. Kim, J. Lee, J. Oh, K. Park, Y. Park, K. Park, Blockchain based energy trading scheme for vehicle-to-vehicle using decentralized identifiers, *Appl. Energy* 322 (2022) 119445, <https://doi.org/10.1016/j.apenergy.2022.119445>, <https://www.sciencedirect.com/science/article/pii/S0306261922007723>.
- [12] R.P. Parameswarath, P. Gope, B. Sikdar, User-empowered privacy-preserving authentication protocol for electric vehicle charging based on decentralized identity and verifiable credential, *ACM Trans. Manag. Inf. Syst.* (Apr. 2022), <https://doi.org/10.1145/3532869>.
- [13] G. Thakur, P. Kumar, S. Jangirala, A.K. Das, Y. Park, et al., An effective privacy-preserving blockchain-assisted security protocol for cloud-based digital twin environment, *IEEE Access* 11 (2023) 26877–26892.
- [14] V. Damjanovic-Behrendt, A digital twin-based privacy enhancement mechanism for the automotive industry, in: *2018 International Conference on Intelligent Systems (IS)*, IEEE, 2018, pp. 272–279.
- [15] X. Boyen, *Robust and Reusable Fuzzy Extractors*, Springer London, London, 2007, pp. 101–112.
- [16] Z. Guan, Z. Wan, Y. Yang, Y. Zhou, B. Huang, Blockmaze: an efficient privacy-preserving account-model blockchain based on zk-snarks, *IEEE Trans. Dependable Secure Comput.* 19 (3) (2022) 1446–1463, <https://doi.org/10.1109/TDSC.2020.3025129>.
- [17] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.
- [18] N. Kobitz, A. Menezes, S. Vanstone, The state of elliptic curve cryptography, *Des. Codes Cryptogr.* 19 (2) (2000) 173–193.
- [19] J. Doerner, Y. Kondi, E. Lee, A. Shelat, Secure two-party threshold ecDSA from ecDSA assumptions, in: *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 980–997.
- [20] K. Suresh, R. Pal, S. Balasundaram, Two-factor-based rsa key generation from fingerprint biometrics and password for secure communication, *Complex Intell. Syst.* (2022) 1–15.
- [21] C. Patel, N. Doshi, Secure lightweight key exchange using ecc for user-gateway paradigm, *IEEE Trans. Comput.* 70 (11) (2021) 1789–1803, <https://doi.org/10.1109/TC.2020.3026027>.
- [22] M. Tezuka, Y. Yoshida, K. Tanaka, Weakened random oracle models with target prefix, in: *International Conference on Security for Information Technology and Communications*, Springer, 2018, pp. 344–357.
- [23] F. Zhang, D. Maram, H. Malvai, S. Goldfeder, A. Juels, Deco: liberating web data using decentralized oracles for tls, in: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1919–1938.
- [24] X. Li, T. Jing, R. Li, H. Li, X. Wang, D. Shen, Bdca: blockchain and decentralized identifiers assisted secure registration and authentication for vanets, *IEEE Int. Things J.* (2022) 1, <https://doi.org/10.1109/JIOT.2022.3164147>.
- [25] Y.I. Shehu, A. Ruiz-Garcia, V. Palade, A. James, Sokoto coventry fingerprint dataset, *arXiv preprint*, arXiv:1807.10609, 2018.
- [26] A.M.R. Center, Helping healthcare firm de-risk automation investment, <https://www.amrc.co.uk/news/helping-healthcare-firm-de-risk-automation-investment>, 2019, 2022-09-16.
- [27] A. Dalla Libera, R. Carli, A data-efficient geometrically inspired polynomial kernel for robot inverse dynamic, *IEEE Robot. Autom. Lett.* 5 (1) (2019) 24–31.
- [28] starkbank ecDSA, A lightweight and fast ecDSA implementation, <https://github.com/starkbank/ecdsa-dotnet>, 2022.
- [29] hashlib, Python secure hash and message digest module, <https://docs.python.org/3/library/hashlib.html>, 2022.
- [30] alexmgr, tinyec - a tiny elliptic curve library, <https://github.com/alexmrg/tinyec>, 2022.



Dr Chintan Patel is an Assistant Professor in Kaushalya – The Skill University, India. Previously he was working as a Post-doctoral Research Associate with Security of Advance Systems Research Group In University of Sheffield, UK and Adjunct Faculty at Rashtriya Raksha University, Gandhinagar. Dr. Patel completed his Ph.D. from School of Technology at Pandit Deendayal Energy University, Gandhinagar, India in 2021. With the active academician, he is an active researcher also. He has published papers in various reputed journals like IEEE TC, Info. Sec., Wireless Personal Communications and many others. He has also authored a book titled “Internet of Things Security Challenges, Advances, and Analytics” with CRC Press, Taylor and Francis group. He is also an active member of the IEEE, ACM and CSI.



Dr Aryan Pasikhani is currently working as an Academic Fellow in Cybersecurity at the University of Sheffield. Previously, he served as a Post-Doctoral Researcher with the Security of Advanced Systems Group at the same institution. His focus is on publishing research that makes a high impact in the fields of Computer and Network Security. His primary research interests encompass intrusion detection and prevention systems, reinforcement learning, machine learning, quantum computing, and the security of embedded systems. Furthermore, he has been an esteemed Technical Program Committee (TPC) Member and has undertaken peer review responsibilities for several prestigious international journals and conferences, including the IEEE Internet of Things Journal, IEEE Transactions on Industrial Informatics, and the IEEE Sensors Journal.



**Dr Prosanta Gope** is currently working as an Associate Professor in the Department of Computer Science (Cyber Security) at the University of Sheffield, UK. Dr Gope served as a Research Fellow in the Department of Computer Science at the National University of Singapore (NUS). Primarily driven by tackling challenging real-world security problems, he has expertise in lightweight authentication, authenticated encryption, access control, security of mobile communications, healthcare, Internet of Things, Cloud, RFIDs, WSNs, Smart-Grid and IoT Hardware. He has authored more than 100 peer-reviewed articles in several reputable international journals and conferences and has four filed patents. He received the Distinguished PhD Scholar Award in 2014 from the National Cheng Kung University (Taiwan). Several of his papers have been published in high-impact journals (such as IEEE TIFS, IEEE TDSC, IEEE TIE, IEEE TSG, and IEEE/ACM TON), and prominent security conferences (such as IEEE Computer Security Foundations Symposium (CSF), Euro S&P, IEEE TrustCom, IEEE HoST, etc.) Dr Gope has served as a TPC member/Co-Chair in several

reputable international conferences such as IEEE TrustCom, IEEE GLOBECOM(Security-Track), ARES, ESORICS, etc. He currently serves as an Associate Editor of the IEEE Internet of Things Journal, IEEE Systems Journal, IEEE Sensors Journal, and the Journal of Information Security and Applications (Elsevier). His research has been funded by EPSRC, Innovate UK, and the Royal Society.



**John Clark** is a Professor of Computer and Information Security at the University of Sheffield since April 2017 and leads the Security of Advanced Systems Research Group. Previously he was Professor of Critical Systems at the University of York, having joined academia in 1992 as a Lecturer in Safety Critical Systems. He studied Maths and then Applied Statistics at Oxford, before joining the security division of the software and systems house Logica (where he worked on security evaluation and security R&D).