A Digital Asset Inheritance Model to Convey Online Persona Posthumously

Ram Govind Singh 1 · Ananya Shrivastava 2 · Sushmita Ruj 3

Abstract
The astounding growth of the Internet has generated digital asset extensively. Users are concerned about asset management so that the asset can be conveyed successfully to the descendent posthumously. Very few works have addressed designing and modeling of digital asset inheritance (DAI) from a technical design perspective. They have several inherent limitations such as incorrect death confirmation, high participation of nominee, the possibility of failure to obtain recovery key, and are based on many unreasonable assumptions, thus inefficient to design in the real life. In this paper, we have formalized the different categories of digital assets and defined the various security goals, required functionalities, and necessary entities to build an asset inheritance model. We have also proposed a new protocol named digital asset inheritance protocol (DAIP) using certificateless encryption (CLE) and identity-based system (IBS) to convey the user’s online persona efficiently to the descendent after his death. DAIP allows the nominee to successfully retrieve the asset after the user’s demise, even if a nominee is uninformed regarding the asset. We, then, provide rigorous security proofs of various properties using real–ideal worlds paradigm. Finally, we have implemented DAIP model using PBC and pycryptodome library. The simulation results affirm that it can be practically efficient to implement.

Keywords Digital asset management · Digital asset inheritance · Right to posthumously · Certificateless encryption

1 Introduction
An asset is any substance or resource (tangible or intangible) that has an economic value and future benefit. Due to the recent breathtaking development of the Internet, digital communication is now an integral part of people’s lifestyles, and such an expansion has generated digital assets exponentially over cyberspace. The data associated with email, social media platforms (such as Facebook, Instagram, and Twitter), user’s website, blogs, data stored at cloud (such as pictures, photographs, diary, videos, songs, books), online wallets, coupons, and gifts cards are few examples of digital assets. The asset inheritance system provides assistance to the dependent in the absence of an asset holder (a.k.a user). Any person can convey his property in the present or future to one or more living person(s). Currently, organizations authorize inheritance only for financial assets. However, due to digitization, the value and importance of digital data have increased which appeals for asset inheritance and asset management of digital assets as well.

The traditional inheritance model of financial assets has been designed under the assumption that the nominee is aware of the asset and can inherit it later after the user’s demise. However, this model fails to guarantee inheritance if the nominee is uninformed regarding assets. This can be seen in media reports that banks and insurance companies have many dormant accounts with billions of unclaimed assets [26]. Furthermore, the conventional inheritance model is not directly applicable to digital assets because the nature and category of assets are immensely diversified. For instance, an Internet user’s account may not be associated with a real
user name (such as an account created on a cloud platform) or an account holding decentralized cryptocurrencies solely managed by the user. These assets may be lost forever if the transfer process is not robust.

Thus, the critical concern is how the digital asset inheritance (DAI) can be modeled. There have been few works in this direction that have proposed the idea of DAI for user-owned assets like cryptocurrency, passwords, cryptographic private keys, etc. For instance, DigiPulse [10] stores assets in encrypted form and transfer to the descendants using blockchain technology; PassOn [30] stores asset in the form of a token; SafeHaven [35] uses a secret sharing scheme to distribute cryptocurrency private key; and TrustVerse [37] is designed for asset management using the smart contract. However, we observe that all these models do not specify their inheritance design. In particular, it is not known how the asset information will be stored securely, how the user’s death will be confirmed correctly, how many nominees will participate, and how dependents will get to know about the asset after the demise of the user. These models also fail to explain how the asset transfer will work if the dependent does not know about it.

Transfer and inheritance of digital assets are highly overlooked and have not received the research community’s appropriate attention [36]. All the digital assets are worthwhile, and the user would like to transfer them to his family members. Therefore, a simple and robust model is necessary to design so that a nominee could get information regarding the digital assets left after the user’s demise. Nominee can inherit it efficiently without any loss, given that he may or may not know about the assets owned by the user before death.

In this paper, we have solved the above problems. We have formalized the category of digital assets, defined functionalities, and security goals necessary to design a DAI model. We have proposed a digital asset inheritance protocol (DAIP) based on certificateless encryption that stores and manages user’s asset or asset information. After the user’s demise, such information can be conveyed to the nominee efficiently and can be used to inherit the asset.

1.1 Motivation

The research community comprehended the significance of digital asset inheritance and started appraising it from different aspects. To design an effective DAI model, the following needs to be addressed: (i) the creation of well-defined state and federal level digital asset laws and policies [20,21,26,28,38]; (ii) the development of adequate organizational level policies [3,17,22]; and (iii) and the creation of a robust technological model. In this paper, we explore and formalize only the last one, the technical design of DAI, while the first two are out of this paper’s scope.

In the existing system, few organizations practice provisional methods of a legacy contract [4] to grant access to the user’s account after his death. However, this is not a full-fledged inheritance model. Recently, DigiPulse [10], PassOn [30], SafeHaven [35], and TrustVerse [37] have introduced the notion of asset inheritance and asset management system. However, these models have only proposed ideas without straightforward design and have the following issues and challenges: i) Designed for inheriting only user-owned assets (such as cryptocurrency key, password, or any other data) and does not discuss other categories of digital assets such as organization-owned data; ii) Require excessive nominee participation during creating, storing and managing the asset which increases the overall complexity in terms of design; iii) Uses inaccurate ways of death confirmation such as inactivity period [10], voting [30], weighted voting [30], consent of nominee [37]. All these death confirmation methods make the DAI protocol fail to guarantee the robustness property (as well as other security properties), if one nominee denies voting during the death confirmation process; and iv) Does not clearly specify how the secret key required to access the assets will be transferred to the descendants in [10,30,37]. Therefore, current asset management models are at a very early stage and relatively immature.

The above challenges motivated us to design a model that can solve the above limitations. In particular, we ask the following questions:

- What are the different categories of digital assets? How and in which form all categories of assets should be managed so that they can be inherited efficiently?
- How can a user store digital asset information securely so that it can be conveyed to the descendent later after his death?
- What should be the functionalities and properties of the asset inheritance model?
- How can a nominee claim and retrieve all the digital assets successfully after the user’s death?
- Can we force each data fiduciary to hand over all the digital assets to the nominee when they are alive?

The integration of a robust asset inheritance model, particularly with the existing platform, may be of great help for a nation especially during COVID-19 where many people died and left their assets without informing to their nominees.

1.2 Our contribution

Our first contribution is formally modeling the digital asset inheritance (DAI) system. In particular, we have described the different categories of digital assets, entities involved in the system, design functionalities, security goals, and various constraints associated with the DAI model.
Next, we have designed a digital asset inheritance protocol, namely DAIP, that adheres to the required design functionalities and achieves certain security properties. Our DAI protocol uses certificateless encryption and some basic cryptographic primitives like hash function to achieve the desired design functionalities and security goals. Our protocol also solves one of the important challenges, namely the identifiability problem, that exists in the organization, in order to verify the identity of a pseudonymous user correctly after the death.

Thereafter, we have given rigorous security analysis of our protocol to prove that it ensures various security properties (such as asset privacy) using real/ideal simulation paradigm of universal composability (UC) framework. Next, we establish that our DAIP is better than the existing ones both in terms of design functionalities as well as security properties (see Tables 1 and 2 for the details). We have discussed the platform utilization of DAIP model, applications, benefits to stakeholders and service providers, and how it can be integrated with the existing infrastructure (refer Sect. 8.1 for the details). Finally, we have simulated and analyzed the implementation of DAIP and showed that the protocol is efficient for a large scale of users.

1.3 Organization

The rest of the paper is organized as follows. We start with modeling of digital asset inheritance (DAI) system in Sect. 2 that consists of basic description of digital asset, design functionalities, and security goals. In Sect. 3, we have described various related works done in designing DAI system. Thereafter, in Sect. 4, we describe various cryptographic primitives as preliminaries. We present the full description of our DAI protocol along with formal security proof in Sects. 5 and 6. Then, we compare our newly designed protocol with the existing ones, in terms of both design and security, in Sect. 7. We, then, discuss the application and practical implementation of DAIP in Sect. 8. Finally, we conclude in Sect. 9.

2 Digital asset inheritance system

A user must plan to pass his online persona to his successor after his demise. The model that deals with the transfer of such assets is known as the Digital Asset Inheritance (DAI) system. The construction of inheritance models is at a very early stage, and no formal construction is defined yet. This section formalizes the design functionalities, entities required, and security goals necessary to develop an inheritance model.

2.1 Digital asset

We define digital asset as any valuable data, exists over cyberspace, that a user wants to pass-on to his descendants. The asset information is a set of parameters declared by a user during inheritance declaration. It may include user id, account number, registration number, or any other user-specific variables used to identify and claim an asset by the descendants. Digital asset exists in many forms. At a very high level, we can divide the asset into four major categories:

TYPE 1 (ORGANIZATION MANAGED DATA (OMD)) In this category, data is entirely maintained and managed by the organization (a.k.a. data fiduciary) such as web services, cloud data, songs, and videos. The users are allowed to store, update, delete and manage the data with access control.

Based on the nature of the organization’s data and policy, it should register nominee details so that the asset could be transferred to the descendants after his demise. We want to point out that currently most organizations are not concerned about it.

TYPE 2 (ORGANIZATION MANAGED MONETARY ASSET (OMMA)) An organization that manages users financial assets, having monetary value, comes under this category. It can be any local or global currency, bank accounts, insurance, stock market shares, mutual funds, or e-wallets. This category of data mostly belongs to identifiable users.

Users can keep all these account information together in one place so that even if a nominee is not aware, they can know it in the future.

TYPE 3 (USER OWNED DATA (UOD)) These personal data are known to the user’s only and participation of the organization is not needed. User’s sensitive personal data, cryptographic private keys, passwords, cryptocurrency keys, or any non-bankable assets are some of the examples of data that comes under this category. It can also include any data which are managed and stored solely by the user’s itself such as any personal files, books, and songs.

TYPE 4 (MIXED CATEGORY) This category of data consists of two parts. i) a secret key and ii) encrypted data. The key is owned and managed by the user while the organization stores encrypted data (for example, digital will or data stored with client-side encryption).

The user has to add nominee details at the organization and plan the transfer of the secret key to the descendants.

The user needs to convey either the complete asset (e.g., user-owned data, a cryptographic key) or the asset information (e.g., account number) to the descendants based on the category of data.
Our DAIP is designed for type 1 and type 4 categories of assets, nevertheless, it can be easily extended for type 2 and type 3 categories of assets.

### 2.2 Entities involved

In this section, we define the primary entities required to construct DAI model. The details of each entity, its role, and behavior are described below:

**User** ($U$) is the owner of the digital asset who plans to pass its asset to his successors. We divide the user’s into two categories: (i) *pseudonym users*, who have not registered themselves with any real identity at the organization. These users may or may not be identified correctly (for example, email id, Github or Twitter handle); (ii) *identifiable users*, who have registered to the organization using real identity such as bank accounts and mutual funds.

**Nominee** ($N$) is a legal custodian of the asset. Anyone is eligible to become a nominee if and only if he/she fulfills the legal requirements. Generally, preferred nominees are family members and close relatives, but in certain circumstances, anyone could be a nominee if state or federal laws allow it. The user can define single or multiple nominees against an asset.

**Organization** ($O$) stores user’s data for type 1, 2, or 4 category assets. It shall allow users to declare nominee details. A nominee can communicate with the organization upon the user’s death and receive the asset after necessary verification. Based on the nature of the service offered, an organization may register both pseudonyms and identifiable users. Note that the organization has to confirm and verify users’ identity correctly before the transfer of an asset to the nominee for the pseudonym users.

**Certificate Authority Portal** (CAP) is the portal that confirms the user’s death. It can be any trusted entity that shall approve demise correctly. In this model, we have assumed, CAP is a central entity authorized by the government within the state, solely responsible for confirmation of death and generation of death certificate (defined in a later section) upon the request by their belongings.

**Nominee Display Portal** (NDP) is a dedicated system for efficient storage and management of asset inheritance data (AID). NDP implementation can be done either using a dedicated centralized system or a decentralized system such as blockchain.

**Identity Based System** (IBS) stores the identity ($id_U$) of users within the system. IBS could be assumed as any identity-based system such as the Aadhaar system in India or SSIN number in the USA. It will help entities to confirm and verify users identities after demise. IBS has the functionality of authentication service. Any user ($u$, $id_U$) can authenticate himself uniquely and correctly using this service.

### 2.3 Design functionalities

In this section, we define the core functionalities required to design an asset inheritance model. It includes the following:

**AID Creation** The asset inheritance model should implement functionality that allows a user to create Asset Inheritance Data (AID). It may consist of a list of asset or asset information, a list of nominees, a list of secrets, and any other information as per the requirement. The nominee will receive AID after the death of the user. Note that if an organization stores the asset, then communication with the organization is also required during AID creation.

**AID Storage** The DAI model should implement functionality that allows users to store and manage the AID created in the previous step. Storage functionality shall enable the user to edit, update, or delete AID.

**Nominee Participation** The DAI model should define up to which extent and in what manner nominee participation is required. Note that the DAI model should use the minimum participation during asset creation and execution of the protocol.

**Death Confirmation** The DAI model should define a robust and accurate method for death confirmation since security, privacy, and transfer of AID is dependent on the correct implementation of death confirmation.

**AID Retrieval** The DAI model should allow the nominee to retrieve the AID from the storage functionality after a user’s death.

**Asset Transfer** The DAI model should implement functionality that allows the nominee to reconstruct secrets and recover asset or asset information. Note that the asset recovery will also require communication with the organization if stored at the organization.

### 2.4 Security goals

This section defines the various security goals required to design a digital asset inheritance protocol (DAIP) securely.

**Asset Privacy** A DAI model is said to achieve asset privacy if AID stored by the user at the NDP portal does not reveal anything before death to both NDP and any other third party. Note that after the death of a user, an NDP will only learn about the mapping of a pseudonymous user with the
real identity, however, the secrets and the worth of the assets will be revealed only to the nominee.

**Non-repudiation** A DAI model is said to achieve non-repudiation if the organization cannot deny the asset’s holding when the nominee requests it. This definition holds for both pseudonymous and identifiable users and data category type 1, type 2, and type 4.

**Identifiability** A DAI model is said to achieve identifiability if it fulfills the following conditions: i) the user’s identity is not known to the organization before death; and ii) an organization knows the identity of the user only after the death. This definition holds only for the data category type 1 and type 4, where the organization provides service to pseudonymous users.

Currently, in type 1 and type 4 categories of assets, the users are not registered at the organization with any real identity. Thus, his physical identity cannot be known accurately and correctly by the organization. Therefore, there is a need to design a functionality to identify a user correctly after the death and transfer the online persona to his descendant.

**Key Inheritance Privacy** A DAI model is said to achieve key inheritance privacy if secret keys are revealed only to the designated nominee assigned against an asset. This property is important because sometimes a user does not want to disclose all the assets to everyone.

**User Privacy** A DAI model is said to achieve user privacy if the asset inheritance data stored at NDP does not reveal the user’s identity before his death.

**Robustness** The DAI model is said to be robust if the nominee retrieves all the asset or asset information from AID successfully without fail.

**Correctness** A DAI model is said to be correct if it fulfills the following: i) the organization can confirm and validate the user’s identity; and ii) the organization can correctly validate the nominee before transferring the asset.

### 2.5 Concern of correct death confirmation

Digital asset inheritance protocol requires the assumption of a trusted party who can confirm the death correctly.

Death is purely unpredictable. It does not depend on any physical parameters and real-world constraints. The existing models of asset management use the following mechanisms to confirm the death [10,30,35,37]: i) **voting**: In this method, all nominees vote to confirm the death. However, if any nominee denies voting, the asset cannot be recovered; ii) **weighted voting**: It allows confirmation of death by a threshold number of nominees, each having different weights. Although, if some nominees with higher weights deny voting, the threshold cannot be achieved; iii) **notary**: is a legal person who confirms the user’s death and also stores the secret key which will be used later by the nominee to learn about the asset. However, if the notary denies participating in the protocol then the nominee can never retrieve this secret key; iv) **inactivity period**: In this method, the user’s logged-in status is collected from different social media accounts (for example, E-mail, Facebook), and if a user has not logged-in since a pre-defined time, the death is confirmed. However, due to the privacy policy, many websites may not share login details with other organizations. Further, if a user is alive and forgot to update his liveliness status through account login, all his secrets would be revealed to the nominee; and v) **secret sharing**: In this method, the multiple nominees hold the shares of the key. After the user’s death, nominees submit their shares and recover the key. This method does not validate death correctly as all the nominees may collude with each other.

The incorrect execution of death confirmation may produce adverse effects on the model. Therefore, we need a trusted entity who will confirm the user’s demise correctly; otherwise, the complete model will fail.

### 3 Related work

The existing state-of-the-art focuses research on digital asset inheritance from different facets.

The earlier work was concentrated on the issues and challenges of existing laws and policies of digital asset inheritance system. The uniform digital asset law proposed under the Revised Uniform Fiduciary Access to Digital Assets Act (the “RUFADAA”) authorize personal representative of descendant's to access digital assets [38]. However, this accessibility does not imply inheritability (transfer of digital asset after death); hence, an adequate law and policy is needed for effective inheritance [20,28,34]. A successful digital asset transfer should also include other perspectives like legalization by countries, proper practical planning, and the analysis of failure consequences [6,18,23]. It should be handled in a similar way like other assets but using a better defined legal system [12,21]. Many works have analyzed whether the country’s existing laws and regulations are applicable to inherit the digital asset. For instance, a study has been done on Estonian government regulation to know in what context their laws are applicable for an heir to make him eligible to access the account of a deceased to download data [24]. A similar study has also been done on the United Kingdoms’ regulation [15]. Another essential concern for a robust DAI model is the privacy of user’s data. The regulation needs to fix how the assets should be passed to the descendant while maintaining the privacy of the digital asset owner [17]. For instance, to preserve privacy, those data may be transferred that is financially worthwhile, maybe publicly published, or may be beneficial in the future (with the condi-
tion that privacy concerns no longer hold in the future). Any other digital asset which violates the user’s privacy must be destroyed [25]. Also, the concern regarding postmortem data privacy, data ethics, and property rights of the user’s after death have been addressed in [16,29]. In all these studies, we observe that digital asset inheritance is a growing concern among society, and reform in the law and some practical solution is required for this indispensable subject.

The possible steps regarding the practical implementation of DAI are being analyzed in [22,27]. In the current system, the asset’s future is determined by the organization’s public policy. For instance, Facebook uses a postmortem data management policy that preserves the memory of the deceased, allow to post about the user’s death by nominated friends, and grant memorializing practices [4]. Similarly, Google uses an inactive account manager to inform and delete the account of the dead. Twitter allows retrieving the user’s data by their successors from its portal. In the absence of public policy within the organization, the future of the user’s account is determined by the company’s “Terms of Service Agreement” (ToSA) [19]. The asset may still be denied if ToSA is not implemented correctly [33]. However, these are not full-fledged inheritance models. They may allow access to the user’s accounts up to a certain extent but they do not implement a dedicated inheritance model. Thus, an efficient model is necessary to implement DAI effectively.

Designing technical implementation of asset management and asset inheritance is at a very initial stage. Recently, few asset inheritance models have been proposed using Blockchain such as PassOn [30], DigiPulse [10], SafeHaven [35], TrustVerse [37]. In DigiPulse, the user can store sensitive data at the DigiPulse portal in the encrypted form and hash of files on the blockchain to maintain integrity. Thereafter, the encryption key can be distributed among nominees using any secret sharing schemes. PassOn converts various assets in the form of tokens and stores them on the blockchain. The tokens can be transferred to the family member in case of the death of the user. SafeHaven uses a secret sharing scheme to distribute cryptocurrency secret keys among nominees. The original key can be reproduced when each nominee will submit their share, including the share of a trusted party known as Trusted Alliance Network (TAN). TrustVerse is an estate planning and asset management system which allows a user and its family members to create a smart contract using blockchain for an asset. The smart contract will be executed when a threshold number of family members will confirm the death. However, all these have a poor technical description and have many existing challenges, such as being applicable only for type 3 category assets, the possibility of recreation of the incorrect key by a nominee, or weak method of death confirmation.

In this work, we have designed a DAI protocol. Any user wants to transfer, and the nominee can inherit it successfully after the user’s demise (see Sec. 5 for details).

4 Preliminaries

Our construction requires various well-known cryptographic schemes, namely, a symmetric key encryption scheme \( \text{SKS} = (\text{SKS.Setup}, \text{SKS.Enc}, \text{SKS.Dec}) \), a signature scheme \( \Sigma = (\Sigma.\text{Setup}, \Sigma.\text{Sign}, \Sigma.\text{Verify}) \), a pseudo-random function \( \text{PRF} \), and a hash function \( H \). For better readability, the rigorous definitions of these schemes are defined in Appendix A.

In this section, we define the following primitive in detail.

4.1 Certificateless encryption (CLE) scheme

Definition 1 (Certificateless Encryption [2]) A certificateless encryption \( \text{CLE} = \{ \text{CLE.KeyGen}, \text{CLE.Enc}, \text{CLE.Extract}, \text{CLE.Dec} \} \) is a 5-tuple of algorithms over the setup algorithm \( \text{CLE.Setup} \), that works as follows:

- \( \text{CLE.Setup}(1^k) \rightarrow (\text{mpk},\text{msk}) \): It is executed by key generation center (KGC). On input the security parameter \( 1^k \), it returns \( (\text{mpk},\text{msk}) \), where \( \text{mpk} \) is the master public key, \( \text{msk} \) and the master private key. The master public key is distributed and \( \text{msk} \) is kept secret by KGC.
- \( \text{CLE.KeyGen}(\text{mpk},\text{id}) \rightarrow (\text{pk},\text{sk}) \): It is executed by the receiver to generate a public/secret key pair. On input \( (\text{mpk},\text{id}) \), it returns the key pair \( (\text{pk},\text{sk}) \) where \( \text{pk} \) is public key, and \( \text{sk} \) is the secret key. Note that public key generated using this algorithm does not need to be authenticated with a digital certificate.
- \( \text{CLE.Enc}(\text{mpk},\text{pk},\text{id},\text{m}) \rightarrow \text{c} \): It is executed by the sender to send a message \( \text{m} \) to the receiver. On input the parameters \( (\text{mpk},\text{msk},\text{id}) \), and a message \( \text{m} \) drawn from the message space \( M \), it returns either a ciphertext \( c \in C \) or \( \perp \) indicating error that the public key is not valid for the identity \( \text{id} \).
- \( \text{CLE.Extract}(\text{mpk},\text{msk},\text{id}) \rightarrow \text{sk}' \): It is executed by KGC to create partial private key \( \text{sk} \) for the user’s identity \( \text{id} \). It takes master secret key \( \text{msk} \), master public key \( \text{mpk} \) and \( \text{id} \) as parameter, and returns partial private key \( \text{sk}' \).
- \( \text{CLE.Dec}(\text{mpk},\text{sk},\text{sk}',\text{c}) \rightarrow \text{m} \): It is executed by the receiver to decrypt a ciphertext. It takes inputs, the master public key \( \text{mpk} \), the receiver’s secret value \( \text{sk} \), the receiver’s partial private key \( \text{sk}' \), and ciphertext \( \text{c} \), and returns either a message \( \text{m} \in M \) or \( \perp \) indicating that the ciphertext is invalid. The above algorithm satisfies the following property:

The above algorithm satisfies the following property:
– Correctness: For every pair \((pk, sk, sk', id)\) and for every \(m \in \mathcal{M}\), the following holds: \(Pr[CLE.Dec(mpk, sk, sk', CLE.Enc(mpk, pk, id, m)) = m] = 1\).

– Indistinguishability: A CLE scheme is semantically secure against an adaptive chosen ciphertext attack (“IND-CCA secure”), if no polynomially bounded adversary \(A\) of Type I or Type II has a non-negligible advantage against the challenger in the guessing game. For details, see [2].

The concrete instantiation of this scheme is given in Appendix B.

5 Digital asset inheritance protocol (DAIP)

In this section, we design the asset inheritance protocol DAIP, using the CLE primitive, for type 1 and type 4 categories of the asset. We consider six entities, namely: user \(U\), nominee \(N\), NDP, CAP, IBS, and the organization \(O\).

Suppose a user \(U\) has an account \(accU\) in the organization \(O\). Digital assets are managed, modified, and updated by \(U\) and stored at \(O\). \(U\) holds a unique identity number \(idU\) assigned by IBS. He wants to create an asset inheritance data \(AID\) at NDP to convey his persona \(asset^{O}_U\) to his nominee \(N\) after his demise. After the user’s death, the protocol should guarantee that \(N\) can obtain \(AID\) from NDP and retrieve \(asset^{O}_U\) from \(O\).

The protocol \(\Pi\) is a composition of two sub-protocols: (i) Asset Management, denoted \(\Pi_1\), which ensures efficient storage of an asset by the user so that it can be retrieved later by the nominee after the demise of the user; (ii) Asset Inheritance, denoted \(\Pi_2\), which ensures proper delivery of the asset to the nominee post demise of the user.

5.1 Asset management protocol

The Asset Management protocol, denoted \(\Pi_1\), allows a user \(U\) to store the asset inheritance data \(AID\) of \(accU\) so that the nominee can retrieve and inherit it after the demise of the user. In order to design the protocol, we define the following functions:

– \(\text{KeyGen}(\lambda) \rightarrow r\) : On input the security parameter \(\lambda\), it returns a random number \(r \in \{0, 1\}^\lambda\).

– \(\text{PseudoID}(idU, r) \rightarrow \tilde{id}U\) : On input the identity of user \(idU\) and a random number \(r\), it returns the pseudo-identity of the user \(\tilde{id}U = H(idU \| r)\), where \(H\) is a standard hash function. The function maps a user’s unique identity to a pseudo-random identity.

From the standpoint of design, the protocol \(\Pi_1\) consists of 4 stages, all executed by parties \(U, N, O, IBS, CAP,\) and NDP. The pictorial and algorithmic descriptions of \(\Pi_1\) are given in Figs. 1 and 2, respectively.

Stage 0—Setup phase The purpose of this stage is to generate initial parameters—public/private key pairs and pseudo-identity—for the parties for executing the protocol. In this stage, the user also generates an asset key \(k^O\) (which is a function of partial keys: \(k_1\) and \(k^O_2\)) for hiding the asset or asset information. Then, he encrypts the partial key \(k_1\) (along with the user identity \(id_u\)) using public parameters of CAP \((pk_{CAP}, id_{CAP})\). The asset key \(k^O\) has the following features: i) it will be revealed to the nominee only after the user’s demise; ii) and, it will be revealed to the designated nominee only for recovering the \(AID\).

Our model achieves this using key distribution approach. The two partial keys will be stored separately at different entities and will later be revealed to the nominee to retrieve the asset from the organization. Note that in order to retrieve the asset, a nominee has to re-compute the asset key using these partial keys.

Stage 1—Registration This stage aims to register the user’s pseudo-identity, generated in stage 0, at IBS. Note that this mapping is known only to the user and IBS. The IBS will reveal this mapping to NDP only after the death of the user.

Stage 2—Generation of nominee registration certificate \(N_{cert}\) for each organization The purpose of this stage is to generate the nominee registration certificate \(N_{cert}\). It is a set of data generated by the user after registering the nominee details with the organization.

The certificate \(N_{cert}\) confirms that the organization will provide all the assets declared in the certificate to the nominee if he fulfills the conditions, denoted other details, mentioned in the certificate such as verification procedure, nature and category of data to be transferred, and other terms & condition of transfer.

We want to point out that for asset categories type 1 and 4, the user is not identifiable at the organization. Thus, we have stored a pseudo-identity \(\tilde{id}_U\) at the organization, and its pre-image \(id_U\) and nonce \(O\) are stored at IBS and NDP, respectively. By providing these two values by a nominee, the organization can correctly identify the user. Note that these values will be revealed to the nominee after the demise of the user only. Also, nonce \(O\) is defined uniquely for each organization; thus, a malicious nominee cannot use the nonce defined for one organization to retrieve assets stored in another organization.

Stage 3—Storing asset inheritance data \(AID\) at NDP The purpose of this stage is to store the asset inheritance data \(AID\) at NDP so that a nominee can later obtain it after the user’s demise.
In this stage, the user creates an asset inheritance certificate $A_{cert}$ that encrypts the nominee certificate $N_{cert}$ using the (symmetric) asset key $k_O^1$. Then, he encrypts the second partial key $k_O^{2}$ using the public parameter, denoted $(pk_{Ni}, id_{Ni})$, of each nominee in the nominee list. Next, he generates nominee receipt $N_{receipt}$. The information stored in $N_{receipt}$ allows any nominee to know whether he/she has been assigned as the nominee for the asset by the user. This will make the nominee aware of the asset(s) whether he has to apply for the claim after the user’s demise. Note that such disclosure will not breach the asset and user’s privacy since the partial key $k_1$ will not be revealed to the nominee by CAP before the user’s death.

Finally, user stores asset inheritance data $AID = (\tilde{id}_U, A_{cert}, A_{info}, c_1)$ at the NDP. The $AID$ ensures that the descendent can use this data to retrieve the asset from the organization after his demise.

**Stage 4 — Updation/Deletion of nominee details at each organization** The purpose of this stage is to allow a user to update the nominee details at the organization.

In this phase, a user updates (add/delete a nominee or modify the preference level of a nominee) the nominee list, denoted $nlist_O$ and send the tuple $(acc_O, \tilde{id}_U, nlist_O)$ to the organization for updating his record. The protocol then proceeds, as described in Stage 2, with these updated values.

### 5.2 Asset inheritance protocol

The Asset Inheritance protocol, denoted $\Pi_2$, allows a nominee to inherit the asset after user’s death. In order to design the protocol, we define the following functions:

- $DCert(id_U) \rightarrow DC_U$: On input $id_U$, CAP performs multi-level verification process. If verification is successful, it creates a document $m_U$; compute signature $\Sigma.Sig(sk_{CAP}, m_U) \rightarrow \sigma_{dc}$; and returns death certificate $DC_U = (m_U||\sigma_{dc})$. Here, $m_U = (\text{serial no.}, \text{name}, id_U, \text{demise date}, \text{issue date}, \text{other parameters (if any)})$.
- $RCert(id_U, name_n, id_N, rp_{U N}) \rightarrow RC_N^U$: On input $(id_U, id_N)$, CAP performs multi-level verification process. If verification is successful, it creates a document $m_N$; generates signature $\Sigma.Sig \rightarrow \sigma_{rc}$; and returns relationship certificate $RC_N^U = (m_N||\sigma_{rc})$. Here, $m_N = (\text{serial no.}, \text{name}, id_U, id_N, rp_{U N}, \text{issue date}, \text{other parameters (if any)})$.

From the standpoint of design, the protocol $\Pi_2$ consists of 4 stages, all executed by parties $N$, $O$, IBS, CAP, and NDP. The nominee communicates with entities, NDP, CAP, and $O$, to know about the $AID$ and retrieve the asset. The pictorial and algorithmic descriptions of $\Pi_2$ are given in Figs. 3 and 4, respectively.
Stage 0: Setup Phase

1. **Generation of master public-private key by IBS** [Identity-based system IBS invokes \(\text{CLE.Setup}(\lambda) \rightarrow (mpk, msk)\) and sends \(mpk\) to \(U, N\) and \(CAP\).
   
2. **Generation of pseudo-identity by user** User \(U\) invokes the following: \(\text{KeyGen}(\lambda) \rightarrow r, \text{PseudoID}(id_U, r) \rightarrow i'd_U\), \(\text{KeyGen}(\lambda) \rightarrow k_1, \text{KeyGen}(\lambda) \rightarrow k_2^O, \text{PRF}(k_1, k_2^O) \rightarrow k^O\).
   
3. **Generation of key parameter by nominee** Execute the following:
   
   a. Nominee \(N\) invokes the key generation algorithm \(\text{CLE.KeyGen}(mpk, id_N) \rightarrow (pk_N, pr_N)\). It submits \(id_N\) to IBS to generate second secret key.
   
   b. IBS invokes \(\text{CLE.Extract}(mpk, msk, id_N) \rightarrow sk_N'\) and returns \(sk_N'\) to \(N\). [Here \(sk_N'\) denotes the nominee's secret key].
   
   c. Nominee \(N\) shares \((pk_N, id_N)\) with \(U\).

   4. **Stage 1: Registration at IBS**

   a. \(U\) invokes \(\text{CLE.Enc}(mpk, id_CAP, pk_CAP, data_2) \rightarrow c_1\). [Here \(data_2 = (k_1 | id_U)\)].

Stage 2: Generation of Nominee Registration Certificate for Each Organization

5. User \(U\) sends \((id_U, i'd_U)\) to IBS for storage.

6. User \(U\) sends the triplet \((\text{acc}_{OC}, id_U, nlist_{OC})\) to the organization \(O\) and requests to certify the asset.
   
7. Organization \(O\) invokes \(\Sigma\).Sig \((sk_O, M_{0U}^U, \sigma_{OU}^U) \rightarrow \sigma_{OU}^U\) and sends \((M_{0U}^U, \sigma_{OU}^U)\) to \(U\).
   
8. User \(U\) generates \(N_{cert} = (M_{O0}^U, \sigma_{OU}^U, nonce_O)\).

Stage 3: Storing Asset Certificate at NDP

9. User \(U\) executes the following:
   
   a. Invokes \(\text{SKS.Enc}(N_{cert}, k^O) \rightarrow A_{cert}\) [Here \(A_{cert}\) denotes the asset certificate].
   
   b. For all \(i \in [l]\), encrypt the second partial key by invoking \(\text{CLE.Enc}(mpk, pk_Ni, id_Ni, k_2^O) \rightarrow c_N^i\).
   
   c. Compute \(r^N = (c_1^N, c_2^N, ..., c_l^N)\) and \(N_{receipt} = (r^N, nlist, O, other\ details)\).
   
   d. Invokes \(\text{SKS.Enc}(N_{receipt}, k_1) \rightarrow A_{inf}\) [Here \(A_{inf}\) denotes the asset information].
   
   e. Register asset inheritance data (AID) at NDP portal as, \(AID = (id_U, A_{cert}, A_{inf}, c_1)\).

Stage 4: Update/deletion of Nominee Details at Each Organization

10. User \(U\) invokes the following:
   
   a. Update nominee list \(nlist_{OC}'\) by adding, deleting or changing the preference level of nominee.
   
   b. Send the triplet \((\text{acc}_{OC}', id_U', nlist_{OC}')\) to \(O\) and request to certify the asset. [Note that \(id_U'\) is same as generated previously in step 5.]
   
   c. Execute steps 6-9 with modified values.

Fig. 2 Algorithmic description of Asset Management protocol \(\Pi_1\)

Stage A—Death Confirmation by CAP

The purpose of this stage is to generate death and relationship certificates by CAP that confirms the death of a user and the relationship between the nominee and the user.

In India, death confirmation is verified by multiple authorities before issuing a death certificate for a user. Because of such multiple confirmations, the chances of false reporting become negligible. Therefore, we assume that the death certificate is correctly issued by CAP.
Also, CAP generates relationship certificate based on the legal terms and conditions. We leave this at the implementation level and do not go into details of it. The relationship certificate also has the following advantages in our protocol: (i) It ensures that only the nominee will know about the asset, but an eligible nominee can only inherit it; and (ii) It prevents the user’s from designating a malicious person as a nominee. For example, if a user designates an unknown person as the nominee, then the nominee will not get the asset as CAP will not issue a relationship certificate to him/her.

**Stage B—Search Query by Nominee** The purpose of this stage is to allow any individual nominee to query to the NDP to check whether any asset has been assigned against him or not. NDP is a single dedicated entity that stores asset-related information of the user. It removes the problem that arises due to the unawareness of assets by the nominee.

**Stage C—Disclosure of AID to Nominee** The purpose of this stage is to disclose the asset information data AID of the user, which is stored at NDP, to the nominee. In this stage, the nominee decrypts the ciphertexts, \((A_{cert}, A_{info}, c_1)\), which he has retrieved from NDP in stage B. Note that any nominee can obtain the first partial key by submitting \(c_1\) to CAP. Now, the nominee decrypts \(A_{info}\) using \(k_1\). The decrypted values disclose the nominee list along with the preference level and any other information assigned for the nominee. Note that decrypting \(A_{info}\) will only reveal that the user has some asset whose information is stored at NDP; however, an eligible nominee can only claim the asset. Thus, it eradicates the problem of unawareness of the user’s asset by the nominee. Finally, the eligible nominee \(N\) re-compute the asset key \(k^O\) using \(k_1\) and \(k^O_2\).

**Stage D—Asset Inheritance** The purpose of this stage is to inherit the user’s asset to the nominee.

In this stage, the nominee claims \(asset_U\) from the organization \(O\) by submitting \((DC_U, RC^U_N, id_U, nonce_U, acc^O_U)\) along with his identity to the organization for asset inheritance. After verification, the organization transfers all the assets to the nominee. Note that before transferring the asset, the organization verifies the nominee’s preference level from \(nlist^O\) to ensure the order in which the nominee is claiming the asset. An alternative nominee can claim the asset only if the primary nominee has also died. In such a case, the alternative nominee has to present all the nominees’ death certificates whose preference levels are higher than him.

### 6 Proof of security of DAIP

We formally define the security properties using universal composability (UC) framework [5]. In this framework, the security of a protocol \(\Pi\) is analyzed by comparing the “real world” execution of the protocol with the execution of the protocol in the “ideal world.” In the “real world,” the parties execute a real protocol among a set of parties, honest and dishonest. Here, the dishonest parties are under the control of
a "special party" called an adversary $A$, who controls their actions and the internal state. The "real world" protocol is said to be secure if it "closely" mimics the "ideal world." By "close," we mean that the real-world view of the adversary is indistinguishable from its ideal world view. We define the view of a party $P$, denoted $\text{view}_P$, that consists of the following: its input, the value it received during execution, and its internal state. Further, the output of party $P$, denoted $\text{out}_P$, is a function of its view.

In the "ideal world," a trusted third party (TTP) is connected to all the parties—honest and dishonest—using a secure and authenticated channel, and no other communication channel exists among the parties. The adversary, on behalf of all the dishonest parties, interacts with the TTP via a simulator $S$ who simulates the execution. Informally, if the adversary is not able to distinguish whether it is interacting in the ideal world or the real world then the protocol is said to be UC-secure.

We now analyze the DAIP protocol $\Pi = \Pi_2 \circ \Pi_1$ for various security properties, as described in Sect. 2.4, under UC model. Here, $U, N, O, IBS, \text{CAP}, \text{NDP}$, and $A$ denote the user, nominee, organization, identity-based system, certification authority portal, nominee display portal, and adversary, respectively. In addition, there are three more entities: the simulator $S$, the trusted third party $F$ which is also the ideal functionality, and a set of honest parties $H$. The input to $H$ is the same as the input of the honest parties in the real world; the input of $A$ consists of the input of the party it is corrupting, combined with the auxiliary input $z$; the input of $S$ is the
same as the input of $A$; internal random tape of $A$, if any, is accessible to the simulator $S$.

### 6.1 Asset privacy

The asset privacy of DAI protocol $\Pi_1$, as defined in Sect. 2.4, guarantees that no third party, including NDP, can reveal anything from the data stored at NDP portal. Using the simulation of ideal and real worlds it can be formalized as follows. For DAIP $\Pi_1$, the ideal world experiment, denoted as $\text{IDEAL-PRV-Security} (1^\lambda, z, acc_U^0, id_U, id_{CAP}, id_N, asset_U^0)$, is described in Fig. 5. The real-world experiment, denoted as $\Pi_1(1^\lambda, z, acc_U^0, id_U, id_{CAP}, id_N, asset_U^0)$, is described in Fig. 1. The views of $A(NDP)$ are defined as $\text{view}_{A(NDP)}^{\text{real}}$ and $\text{view}_{A(NDP)}^{\text{ideal}}$, in these two worlds.

Here the input to $U$ is $(acc_U^0, id_U)$; the input to CAP is $(id_{CAP})$; the input to IBS is $(id_U, id_{CAP}, id_N)$; the input to $O$ is $(acc_U^0, asset_U^0)$; the input to nominate $N$ is $(id_N)$; the input to $A$ (NDP) is $(1^\lambda, z)$; the input to $S$ is $(1^\lambda, z)$. Here $z$ is the information leaked by the adversary $A$ or additional input actively influencing the execution.

**Definition 2** (ASSET-PRV Security) The protocol $\Pi_1$ as described in Sect. 5.1 is said to be ASSET-PRV secure, if for every non-uniform PPT adversary $A(NDP)$ in the real world $\Pi_1$, there exists a non-uniform PPT simulator $S$ in the ideal world $\text{IDEAL-PRV-Security} (1^\lambda, z, acc_U^0, id_U, id_{CAP}, id_N, asset_U^0)$ such that

$$\text{view}_{A(NDP)}^{\text{ideal}} \equiv \text{view}_{A(NDP)}^{\text{real}}$$

for all $\lambda \in \{0, 1\}^*$.

**Theorem 1** (ASSET-PRV Security) Suppose the hash function $H$ is collision-resistant; the certificateless encryption scheme CLE is IND-CCA secure; the function $f$ is pseudo-random PRF, and the symmetric key encryption scheme SKS is IND-CCA secure. Then the DAI protocol $\Pi_1$ (as described in Fig. 1) satisfies ASSET-PRV security (Definition 2).

**Proof** In order to prove this, we need to show that the execution of protocol $\Pi_1$ in the ideal world in the presence of non-uniform PPT simulator $S$ is indistinguishable from the real-world execution even in the presence of an adversary $A$.

The complete proof consists of following two cases:

**Case 1**—Both NDP and IBS are controlled by $A$: Given the input of IBS $(id_U, id_{CAP}, id_N)$ and NDP $(1^\lambda, z)$, the view of IBS after step 4 of $\Pi_1$ is $(id_U, id_{CAP}, id_N)$, the view of NDP after step 8(e) of $\Pi_1$ is $(id_U, id_N)$. Since both parties are controlled by adversary $A$, the input, the internal states and their views are known to $A$. Thus, the adversary knows the identity of user’s who have stored their DAI data at NDP. It is to be noted that the sharing of such mapping does not break the security of the encrypted values, $(A_{cert}, A_{info}, c_1)$, stored at NDP, therefore, the protocol guarantees asset privacy.

Thus, it is sufficient to prove that if the adversary corrupts only NDP and cannot break the IND-CCA security of CLE then the protocol guarantees asset privacy.

**Case 2**—NDP is controlled by $A$: The NDP has stored the following values: $(id_U, A_{cert}, A_{info}, c_1)$. If NDP is able to decrypt the ciphertext $c_1$ then we can design an adversary to break IND-CCA security of CLE scheme which is not possible. Thus, asset privacy property is guaranteed.

Formally, we need to show that the views of the adversary in the ideal and real worlds are indistinguishable, i.e., $\text{view}_{A(NDP)}^{\text{ideal}} \equiv \text{view}_{A(NDP)}^{\text{real}}$. To prove that we have the following hybrids:

$H_0$: This is $\Pi_1(1^\lambda, z, acc_U^0, id_U, id_{CAP}, id_N, acc_U^0, asset_U^0)$ where $\text{view}_{A(NDP)}^{\text{real}}$ is the view of $A(NDP)$.

$H_1$: Identical to $H_0$ except that we change step 0 of $\Pi_1$: replace $id_U \leftarrow \text{Pseudoid}()$ with $id_U \overset{\$}{\leftarrow} T_1$, where $T_1$ is the distribution of $\text{Pseudoid}()$.

**Lemma 1** If hash function $H$ guarantees CRHF property then $\text{view}_{A(NDP)}^{H_1}$ is computationally indistinguishable from $\text{view}_{A(NDP)}^{H_0}$.

**Proof** Follows directly from CRHF property of hash function $H$.

$H_2$: Identical to $H_1$ except that we change step 3 of $\Pi_1$: replace $c_1 \leftarrow$ with $c_1 \overset{\$}{\leftarrow} T_2$, where $T_2$ is the distribution of CLE.$\text{Enc}(\cdot)$.

**Lemma 2** If CLE is IND-CCA secure then $\text{view}_{A(NDP)}^{H_2}$ is computationally indistinguishable from $\text{view}_{A(NDP)}^{H_1}$.

**Proof** Follows directly from CLE security.

$H_3$: Identical to $H_2$ except that we change step 8(a) of $\Pi_1$: replace $A_{cert} \leftarrow$ with $A_{cert} \overset{\$}{\leftarrow} T_3$, where $T_3$ is the domain of SKS.$\text{Enc}(\cdot)$.

**Lemma 3** If SKS is IND-CCA secure then $\text{view}_{A(NDP)}^{H_1}$ is computationally indistinguishable from $\text{view}_{A(NDP)}^{H_2}$.

**Proof** Follows directly from SKS security.

$H_4$: Identical to $H_3$ except that we change step 8(d) of $\Pi_1$: replace $A_{info} \leftarrow$ with $A_{info} \overset{\$}{\leftarrow} T_3$, where $T_3$ is the domain of SKS.$\text{Enc}(\cdot)$.
If Lemma 5

6.2 Non-repudiation

Non-repudiation ensures that during the execution of protocol $\Pi_2$, organization cannot deny from holding of the asset when nominee request for it (see Sect. 2.4). In order to prove this, consider a nominee having input $\text{ID}_N$. During the execution of the protocol $\Pi_2$, the nominee sends $(\text{DC}_U, \text{RC}_N, \text{id}_U, \text{nonce}, \text{acc}_U)$ to organization for claiming the asset. If the organization is honest then he will transfer the asset to the nominee after performing necessary verification as shown in the steps ix-xi. If the organization is malicious then there are two cases: if organization denies the existence of user’s account then in this case nominee can provide $(\text{MO}_U, \sigma_{(U)}^O)$ which ensures that user’s account does exist in that organization; if organization denies transferring the asset then in this case the parties need to resolve the dispute with the help of trusted legal third party.

6.3 Identifiability

The protocol $\Pi$ is said to achieve identifiability if it fulfills the following conditions (as defined in Sect. 2.4): i) the user’s identity is not known to the organization before death; and ii) After death, an organization knows the identity of the user. Note that the protocol $\Pi_1$ ensures the first condition while $\Pi_2$ ensures the second condition. The user $U$ stores $\tilde{id}_U = H(\text{id}_U || \text{nonce}_O)$ to the organization $O$ during the execution of protocol $\Pi_1$ in step 2. In order to prove the first property, we consider the following cases:

**CASE 1:** IF ORGANIZATION IS CORRUPTED BY ADVERSARY $A$: In this case, $A$ will try to find out the pre-image $(\text{id}_U, \text{nonce}^*)$ such that $\tilde{id}_U = H(\text{id}_U || \text{nonce}^*)$. Since $H$ is secure by the pre-image property of Hash function, therefore, user’s identity cannot be known to the organization. Thus, the protocol $\Pi_1$ guarantees the first property identifiability.

**CASE 2:** BOTH ORGANIZATION AND IBS ARE CORRUPTED BY $A$: We know that $IBS$ has the complete list of identity of users. If $A$ corrupts both organization and IBS, then he knows the mapping $(\text{id}_U, \tilde{\text{id}}_U)$ of protocol $\Pi_1$. However, $\tilde{id}_U$ is computed using $(\text{id}_U, \text{nonce}_O)$ thus due to the pre-image security of hash function, $A$ will not be able to know the mapping between $\text{id}_U$ and $\tilde{id}_U$. This guarantees the first property of identifiability in protocol $\Pi_1$.

Nominee submits $(\text{DC}_U, \text{RC}_N^U, \text{id}_U, \text{nonce}_O, \text{acc}_U)$ to the organization in order to retrieve the asset during the execution of protocol $\Pi_2$. The organization verify the correctness of the commitment value as $H(\text{id}_U || \text{nonce}_O) = \tilde{id}_U$ which reveals identity of user’s to the organization. Thus, ensures the second property of identifiability in protocol $\Pi_2$. 
6.4 Key inheritance privacy

The asset key inheritance privacy guarantees that the key is revealed to the designated nominee only in the protocol. During the execution of protocol $\Pi_2$, nominee sends $(id_U, id_N, name_N, id_N, r_{opt}^{U})$ to obtain $(DC_U, RC_N^{U})$. Using this, nominee can get the first partial key $k^O$ from CAP. He can recover the list of nominee $n_{list}$ from $A_{info}$ designated against the asset $A_{cert}$. Now if $n$ does not belong to the $n_{list}$ then he can not decrypt $c^N$ because of the CLE security definition. Thus, protocol $\Pi_2$ guarantees the key inheritance privacy property.

6.5 User’s privacy

The DAI protocol $\Pi$ guarantees user’s privacy if the data stored at NDP portal does not reveal the identity of the user. The user $U$ stores asset inheritance data using $\tilde{id}_U$. If both NDP and IBS do not collude then the protocol $\Pi$ achieves the user’s privacy due to the pre-image security of the hash function. $A$ will not be able to know the mapping. However, if adversary $A$ corrupts and controls both entities then their input, internal state, and views will be known to $A$. Since $A$ knows the identity of user’s who have stored DAI data at NDP, therefore, user’s privacy is not guaranteed.

6.6 Correctness

The protocol $\Pi$ guarantees the correctness properties if both user and nominee can be identified and verified correctly by the organization before transfer of the asset (as defined in Sect. 2.4). We can prove correctness of both properties using the following cases:

**Correctness 1—User’s identification:** The organization can verify the user correctly before transfer of asset. During the execution of the protocol $\Pi_2$, nominee submits $(DC_U, RC_N^{U}, id_U, nonce_O, acc_U)$. The organization can calculate commitment value as $H(id_U || nonce_O) == \tilde{id}'_U$ which ensure the correctness of user’s identity. An adversary $A$ will not be able to generate a value $id^*_u$ such that $\tilde{id}'_U == H(id^*_U || nonce_O)$ due to the security of hash function.

**Correctness 2—Nominee verification:** The nominee $n$ obtains $(id_U, A_{cert}, A_{info}, c_1)$ from step iv in the execution of protocol $\Pi_2$. Using this only designated nominee can retrieve the value of $nonce_O$. Nominee send $(DC_U, RC_N^{U}, id_U, nonce_O, acc_U)$ to the organization. The organization can verify $(DC_U, RC_N^{U})$ from the CAP to ensure the correctness. In the case of adversarial nominee $A$, he can obtains the value $(DC_U, id_U, nonce_O, acc_U)$ by colluding with other nominee. However, $A$ will not be able to generate $RC_N^{U}$ so would not be able to claim for the asset.

6.7 Robustness

A DAI protocol $\Pi$ is said to be robust if nominee can retrieve the asset or asset information without any loss. The execution of protocol $\Pi_1$ and $\Pi_2$ clearly shows that nominee can obtain data from IBS, CAP, and NDP and will able to know DAI. He can communicate with organization to retrieve the asset successfully. However, asset retrieval will not work if any entity such as IBS or NDP refuses holding of data.

7 Comparison between various asset inheritance models

In this section, we compare our DAIP with the four existing digital asset management models: DigiPulse [10], PassOn [30], SafeHaven [35], and TrustVerse [37]. All these models are proposed as white papers without any detailed technical construction, well-defined definitions, description of properties, and security proofs. Therefore, we have mentioned only those properties which were clearly specified, otherwise, we have specified it as “unknown” for this comparison. We have compared the models in terms of design and security properties.

7.1 Design comparison

We now compare the existing protocols of [10,30,35,37] with our DAIP in terms of design. We have summarized the comparison in Table 1.

Currently, all the existing models are more focused on type 3 category assets and do not discuss about handling other categories of assets whose inheritance is equally essential (see Sect. 2.1 for details on the type of assets). The DAIP supports the inheritance of all four categories of assets. The asset inheritance model should require minimum participation of nominee. However, all four models require high involvement of nominees during the execution of the protocol (see Sect. 3 for details). The DAIP requires nominee participation only at the time of key generation. The protocol will not fail even if any nominee dies or denies the protocol’s execution. In the worst case, if only one nominee survives, he will receive the assets.

The successful implementation of the asset inheritance model relies on the correct confirmation of death. Currently, each model has different approaches to confirm and prove the death such as “inactivity period, voting, weighted voting, or through notary.” All these methods have their own inherent limitations (see Sect. 2.5 for details). Our model uses a robust method of death confirmation—by a legal entity (internally verified and approved by multiple entities)—who maintains the record of birth and death within a territory. Therefore, it will neither deny from issuance of death cer-
Table 1 Design properties comparison

| Properties                  | PassOn | DigiPulse [30] | TrustVerse [37] | SafeHaven [35] | DAIP     |
|-----------------------------|--------|----------------|-----------------|----------------|----------|
| Category of asset           | Type 3 | Type 3         | Type 3          | Type 3         | Type 1, 2, 3, 4 |
| Participation of nominee    | Full   | Full           | Full            | Partial        | Partial  |
| Death confirmation methods  | All Nominee consent | By CAP | By CAP | By CAP | By CAP |
| Storage Mechanism           | Blockchain | Blockchain | Blockchain | Blockchain | Blockchain |
| Verification of nominee     | Yes    | No             | No              | No             | Yes      |

7.2 Security properties comparison

We now compare the existing protocols of [10,30,35,37] with our DAIP in terms of security properties. We have summarized the comparison in Table 2.

We found that our DAIP as well as [10] achieves the asset privacy. Our protocol achieves it by storing the asset information at NDP in the encrypted form. All the four existing models handle only type 3 category of assets, thus, identifiability is not an issue. Since DAIP is applicable for all four category of assets, therefore, it solves the identifiability issue for type 1 and type 4 category of assets also, hence, enabling the pseudonym users to transfer his assets correctly. We critically analyze the key inheritance privacy property in [10,30,35,37] but did not find enough details to support their claim; thus, we assume that these protocols do not guarantee this property. Our protocol achieves this property using CLE scheme that hides the secret key from the nominee, thus, only designated nominee can retrieve it. The non-repudiation property does not have any significance for type 3 category assets, hence, it is not applicable with all the existing models. Our DAIP ensures the non-repudiation property.

Unlike all the existing models, our protocol ensures user's privacy under the assumption that IBS and NDP do not collude and reveal the pseudo-identity, \( \tilde{id}_U \). All the existing models do not guarantee robustness property due to incor-
rect death confirmation methods or fail to recover key. Our protocol achieves robustness property under semi-honest setting.

### 8 Application and implementation of DAIP

#### 8.1 DAIP application

The importance of personal data has increased after the recent bill such as GDPR [11] which impels data fiduciary to transfer significant assets of user’s to descendants posthumously. This necessity enables various stakeholders such as social media platforms, cloud service providers, software developers, system designers, and other intermediaries to design and implement an asset inheritance model. Until now, the stakeholders could not comply with the implementation due to its design challenges (as described in Sect. 2). In this work, we have analyzed the challenges thoroughly and an efficient model DAIP is proposed. The model stores distinct attributes of users securely, reveals the user’s identity correctly, and transfer assets to the right nominee. For stakeholders, the implementation of the DAIP model would enable the transfer of inheritable data to the correct nominee, and allow to deactivate and close the user’s account rightly after demise. It provides direction to the service provider on how the protocol can be implemented as a concrete design with the coordination of unique IBS and CAP. It also works as a guideline for authorities such as IBS to enhance their implementation as per the protocol to empower digital asset inheritance.

The DAIP can be easily integrated with the existing systems. We discuss here how the entities of DAIP can be realized and mapped with the current system along with some expected challenges while incorporating the protocol $\Pi$.

For instance, Aadhaar, a unique biometric-based authentication system of users of India, can be used as IBS for authentication and identity storage [1]. However, a minor modification is required in the existing system. Firstly, it has to provide a service to the users so that they can store pseudo-identity against their actual identity. This service can be integrated easily into the existing system. Secondly, it has to provide additional functionality for generating/storing the master key and user’s partial private keys. The challenging part is the storage of additional attributes for large-scale users as it requires a regulatory effort.

Next, in India, every state has a death certificate issuing portal that has the responsibility to confirm the demise, issuance, and verification of the death certificate. The same entity can be realized as CAP with the integration of additional attributes as discussed in the protocol $\Pi_1$ and $\Pi_2$. Since every state uses independent architecture to issue certificates therefore integration of a common additional attribute among them will require an effort. For instance, one such attribute is the incorporation of decryption modules. Similarly, the entity also has to incorporate appropriate authorization and access control to verify the nominee and transfer the decrypted data.

DigiLocker [9] facilitates Indian citizens to store personal documents through an account associated with Aadhaar. NDP services can be conveniently realized using DigiLocker with some modifications. DigiLocker may implement the functionality of storage and management of asset or asset information. It will enable users to directly link and store the inheritable data with their accounts. DigiLocker account also has to introduce the facility of inclusion of nominees. This would make it convenient for nominees to coordinate with DigiLocker (as NDP) to request and download the inheritable data and keys.

Further, every organization will also need to implement some cryptographic primitives. Generally, organizations are equipped with digital certificates. Therefore, they can conveniently implement the signature schemes and store nominee declaration information as per the protocol $\Pi_1$ and $\Pi_2$.

Finally, a communication mechanism with IBS and CAP can be established to verify the identity and death confirmation of users, respectively. Overall, the integration of the DAIP model with existing infrastructure can work as a single umbrella to store and convey the asset posthumously.

#### 8.2 Implementation

We now describe the implementation results of our DAI protocol $\Pi$ (as described in Sect. 5). We have analyzed the
protocol in terms of storage overhead and the computation cost for various entities involved in the execution of $\Pi$ (see Sect. 2.2). First, we describe the experimental setup for the simulation.

**Experimental setup** The experiments were performed on a machine running 64 bits GNU Linux kernel version 5.8.0-43, Ubuntu 20.04.2 LTS, with an Intel® Core™ i5-8250U CPU @ 1.60 GHz x 8 and 12 GB memory. The prototype simulation is implemented in C/C++ GCC compiler version 9.3.0-17, and Python 3.8.10. We have used the PBC Library version PBC-0.5.14 for implementation of pairing functions in certificateless encryption [31], and PyCryptodome-3.12 for implementation of other low-level cryptographic primitives [32].

We have initialized the pairing parameters for CLE encryption and decryption based on Type 1 symmetric pairing $e : G_1 \times G_1 \rightarrow G_2$. The hash function $H_1$ is defined in the PBC library that maps an arbitrary string to a group element $G_1$. The hash function $H_2$ is constructed based on the SHA256 algorithm. This hash function can be easily replaced with any other cryptographic hash function based on the requirements. The size of message (M) for CLE encryption is selected as 32 bytes (256 bits) to enable elementary XOR operations. In every stage of the protocol $\Pi$, cryptographic operations were called, which are simulated using the following parameters: The function KeyGen() is realized using the function Crypto.Random, and the functions PseudoID() and PRF() are realized using SHA256. The number of nominee selected for asset inheritance creation is varied from $2 \leq l \leq 10$. We have used AES 256 bit encryption in CFB mode for SKS.Enc(), SKS.DEC(), and RSA 1024 bits signature algorithm for the implementation of signature schemes $\Sigma$, KeyGen(), $\Sigma$.Sig(), $\Sigma$.Verify(). Next, based on the experimental setup, the approximate size for different parameters used in $\Pi$ is defined as follows:

- PseudoID(), KeyGen(), Identity ($id_u$, $id_o$), Serial no., Account no., Organization name, Name, $rp_u$,: 32 bytes
- CLE.Enc() (U,V): (128, 32) bytes, pk:=(X,Y)=(128, 128) bytes, $s_k$, $s'_k$: 128 bytes
- Sign():: 128 bytes
- Demise date, Issue date: 8 bytes
- Preference Level: 4 bytes

**Computation overhead** We evaluate the computation overhead of execution of protocol $\Pi$. We have implemented each computational stage. All the intermediate steps outputs are stored in a buffer and processed in the next step as per the protocol. Firstly, we have computed the execution time for each entity. Every entity takes its input data, executes its algorithm(s), and computes the output. The time for executing an operation is measured and summed up to calculate the final computation time for each entity. For instance, the computation time of user $U$ is the total time to compute KeyGen(), Key.Extract(), PseudoID(), SKS.Enc(), and CLE.Enc() operation, which are performed in stage 0 to stage 3 of protocol $\Pi_1$. The results are shown in Table 3. The results depict that each entity has to exercise minimal computation to manage and process the data during asset inheritance modeling.

Secondly, we have computed the execution time for certificateless encryption (CLE) for a fixed input data (the second partial key) of 32 bytes by varying the number of nominees from 2 to 10, refer Fig. 6a. The simulation result affirms that the encryption time grows with the number of nominees. However, the encryption computation time is still minimal and can be efficiently integrated with the existing infrastructure.

Finally, we have analyzed the computation time for creating asset inheritance data (AID) by a user. The AID data consists of $\{A_{cert}, A_{info}, c_1\}$ and uses both symmetric key encryption for $(A_{cert}, A_{info})$ and CLE encryption for $c_1$. The input size for the construction of all three parameters is shown in Table 4. The results of Fig. 6b illustrate that the AID creation cost is very small (in ms), and hence, it would be efficient in practice. The variation of encryption time is also nominal due to the following reasons: i) AID creation uses symmetric key encryption, ii) $c_1$ is independent of the number of nominees, and iii) the size of $(N_{cert}, N_{receipt})$ are small.

**Storage overhead** The various components used by the entities to implement $\Pi$ are shown in Table 3. We can use an existing Aadhaar-based system (in India) or Social Security Number (SSN) system (in the USA) as an IBS to design DAIP. The Aadhaar-based IBS has to integrate the CLE component, which has the functionality to generate a master public/private key pair and a partial private key for a user. The NDP can be implemented using a storage system that allows storing the asset inheritance data. For example, in India, Digilocker [9] is a centralized cloud-based storage service for an individual to store their personal data. CAP is a legal entity that confirms death within the state or territory. Many countries are already using some digital methods to confirm the death and generation of death certificates for a user. The same entity can be amplified to generate a relationship certificate as per the protocol $\Pi$ and execute CLE decryption to disclose the first partial key. We want to point out that such a system would need a small overhead for storing $(DC_U, RC_U^O)$. The organization stores the list of nominees and their preference levels. It should support the signature scheme $\Sigma$ to sign the message $M_U^O$. The remaining two entities, user and nominee, do not need any separate storage space. We observe that the storage overhead in all the above discussed entities is modest and can be easily implemented with the existing technologies.

In summary, the efficient results highlight that DAIP can be easily and effectively implemented in real environments and
Table 3 Performance analysis: storage and computation overhead for type 1 and type 4 category assets (No. of nominee: \( l = 10 \))

| Entity       | Components used                      | Storage overhead                  | Computation cost (in ms) |
|--------------|--------------------------------------|-----------------------------------|--------------------------|
| IBS          | IBS system + CLE + KeyGen            | 32 bytes/user + 384 bytes         | 10.516                   |
| NDP          | Any storage system                   | 3708 bytes/user                   | None                     |
| Organization | Signature Scheme \( \Sigma + \) Hash | 1032 bytes/user                   | 64.678                   |
| CAP          | Signature Scheme \( \Sigma + DCert + RCert + KeyGen \) + CLE | 2486 bytes (a user + l nominee)   | 103.070                  |
| User         | KeyGen + PseudoID + SKS + CLE        | 448 bytes                         | 41.056                   |
| Nominee      | CLE + KeyGen                         | 448 bytes                         | 16.672                   |

Fig. 6 Computation time measurement for CLE encryption and AID creation

(a) Run time of certificateless encryption (CLE) for input data size of 32 bytes vs. the number of nominees.

(b) Run time of creating asset inheritance data, \( AID = \{ A_{cert}, A_{info}, c_1 \} \) by a user vs. the number of nominees.

9 Conclusion

In this paper, we have formalized the DAI system. We have defined its functionalities, security goals to construct DAIP. We proposed a new digital asset inheritance protocol (DAIP) using certificateless encryption. Our newly designed protocol allows a user to create asset information that can be conveyed to the nominee for inheritance. We have designed the protocol with different cryptographic primitives to ensure the correct delivery and inheritance of the asset. With security proofs and performance analysis of the proposed model, we have shown that DAIP can be efficiently integrated with the existing system with a small modification. We plan to further enhance this model by including more security features such as user privacy, tamper-proof in the future.

Funding None.

Code availability Available at https://github.com/smartqw/DAIP-Code.

Declarations

Conflicts of interest Not applicable.

Appendix A: Extended preliminaries

Definition 3 (Symmetric key Encryption Scheme [13]) A symmetric key encryption scheme \( SKS = (SKS.Enc, SKS.Dec) \)
is a 2-tuple of algorithms over the setup algorithm SKS.Setup and the message space \( \mathcal{M} \) that works as follows:

- \( \text{SKS.Setup}(1^k) \rightarrow k \): On input the security parameter \( 1^k \), it returns \( sk \), where \( k \) is the symmetric key to be used in encryption and decryption.
- \( \text{SKS.Enc}(k, m) \rightarrow ct \): On input the message \( m \), it returns ciphertext \( ct \) corresponding to \( m \in \mathcal{M} \) under the symmetric key \( k \).
- \( \text{SKS.Dec}(k, ct) \rightarrow m \): On input the symmetric key \( k \) and the ciphertext \( ct \), it returns the message \( m \).

The above algorithm satisfies the following property:

- Correctness: For every key \( k \in \mathcal{K} \) and for every \( m \in \mathcal{M} \), the following holds:
  \[
  \Pr[ct = \bot \text{ or } \text{SKS.Dec}(k, ct) = m : ct \leftarrow \text{SKS.Enc}(k, m)] = 1
  \]

**Definition 4** (Signature Scheme \([8, 14]\)) A signature scheme \( \Sigma = (\Sigma.\text{Sign}, \Sigma.\text{Verify}) \) is a 2-tuple of algorithms over the setup algorithm \( \Sigma.\text{Setup} \) and the message space \( \mathcal{M} \) that works as follows:

- \( \Sigma.\text{Setup}(1^k) \rightarrow (mpk, msk) \): On input the security parameter \( 1^k \), it returns \((mpk, msk)\), where \( mpk \) is the verification key and \( msk \) is the signing key.
- \( \Sigma.\text{Sign}(msk, m) \rightarrow \sigma \): On input the message \( m \), it returns signature \( \sigma \) corresponding to \( m \in \mathcal{M} \) under the signing key \( msk \).
- \( \Sigma.\text{Verify}(mpk, m, \sigma) \rightarrow b \): On input the message \( m \) and signature \( \sigma \), it returns a bit \( b \in \{0, 1\} \), where: \( b = 1 \) if the pair \((m, \sigma)\) is verified under the verification key \( mpk \).

The above algorithm satisfies the following property:

- Correctness: For every pair \((mpk, msk)\) and for every \( m \in \mathcal{M} \), the following holds:
  \[
  \Pr[\Sigma.\text{Verify}(mpk, m, \sigma) \rightarrow 1 | \Sigma.\text{Sign}(msk, m) \rightarrow \sigma] = 1
  \]
- Unforgeability: A signature scheme is unforgeable under an adaptive chosen message attack, if for any PPT adversary \( \mathcal{A} \), and a negligible function \( \mu \), the following holds:
  \[
  \Pr[\Sigma.\text{Setup}(1^k) \rightarrow (mpk, msk) \\
  \land \mathcal{A}.\Sigma.\text{Sign}(msk, \cdot) \rightarrow (m', \sigma') \\
  \land \Sigma.\text{Ver}(vk, m', \sigma') \rightarrow 1 | m' \notin \mathcal{M}] < \mu(1^k),
  \]

where \( M \) is the set of messages submitted by \( \mathcal{A} \) to the Sign oracle.

**Definition 5** (Pseudo-random function \([13]\)) A family of function \( F_K : \{0, 1\}^n \rightarrow \{0, 1\}^m \), indexed by a key \( K \in \{0, 1\}^\mu \) is said to be a pseudo-random function (PRF) if it satisfies the following:

- Given a key \( K \in \{0, 1\}^\mu \) and an input \( X \in \{0, 1\}^n \) there is an efficient algorithm to compute \( F_K(X) \).

It satisfies the following property:

- Indistinguishability: For all probabilistic polynomial time distinguisher \( D \), there exists a negligible function \( \mu(\cdot) \) such that:
  \[
  \Pr[K \leftarrow \{0, 1\}^\mu | D^{F_K(\cdot)}] - \Pr_{F \in \mathcal{F}}[D^{F(\cdot)}] < \mu(\lambda)
  \]

where \( \mathcal{F} = \{ f : \{0, 1\}^n \rightarrow \{0, 1\}^m \} \).

**Definition 6** (Collision Resistant Hash Function \([7]\)) A collision free hash function family \( \mathcal{H} \) is an infinite family of finite sets \( \{H_m\}_{m=1}^\infty \) and a polynomially bounded function \( t : N \rightarrow N \).

A member \( H_m \) is a function \( h : \{0, 1\}^* \rightarrow \{0, 1\}^{t(m)} \), and is called an instance of \( \mathcal{H} \) of size \( m \).

\( \mathcal{H} \) must satisfy the following:

- Given a value of \( m \), there is a probabilistic polynomial (in \( m \)) time algorithm \( \Theta \) which on input \( m \) selects an instance of \( \mathcal{H} \) of size \( m \) at random.
- For any instance \( h \in H_m \) and \( x, y \in \{0, 1\}^* \), \( h(x) \) is easy to compute, i.e., computable in time polynomial both in \( m \) and \( |x| \).
- Given an instance \( h \in H \) selected randomly as in (1), it is hard to find \( x, y \in \{0, 1\}^* \), such that \( h(x) = h(y) \) and \( x \neq y \).

More formally: For any probabilistic polynomial time algorithm \( \mathcal{A} \), and any polynomial \( P \), consider the subset of instances \( h \) of size \( m \) for which \( \mathcal{A} \), with probability at least \( 1/P(m) \), outputs \( x \neq y \) such that \( h(x) = h(y) \).

Let \( \epsilon(m) \) be the probability with which \( \Theta \) selects one of these instances. Then as a function of \( m \), \( \epsilon(m) \) vanishes faster than any polynomial fraction.

**Appendix B: Construction of certificateless encryption (CLE) scheme**

The concrete instantiation of the CLE scheme (as defined in Def. 1) is as follows \([2]\):
CLE.Setup($1^k$) $\rightarrow$ ($mpk$, $msk$): It works as follows:

1. Generate ($q$, $G_1$, $G_2$, $g$, $e$, $H_1$, $H_2$).
   [Here, $G_1$, $G_2$ are groups of some prime order $q$, $e : G_1 \times G_1 \rightarrow G_2$ is a pairing, generator $g \in G_1$, $H_1 : \{0, 1\}^n \rightarrow G_2$, $H_2 : \{0, 1\}^n$, the message space $M \in \{0, 1\}^n$, and the ciphertext space $C \in G_1 \times \{0, 1\}^n$.]
2. Choose $s \overset{\$}{\leftarrow} \mathbb{Z}_q^*$.
3. Set $p_0 := s \cdot g$.
4. return $mpk := (q, G_1, G_2, e, H_1, H_2, g, n, p_0)$ and $msk := s$.

CLE.KeyGen($mpk$, $id$) $\rightarrow$ ($pk$, $sk$): For an $id \in \{0, 1\}^*$, the algorithm works as follows:

1. Choose $s_k \overset{\$}{\leftarrow} \mathbb{Z}_q^*$.
2. Set $pk := (X, Y)$.
   [Here, $X := sk \cdot g$, $Y := sk \cdot p_0 = sk \cdot s \cdot g$.]
3. return ($pk$, $sk$).

CLE.Enc($mpk$, $pk$, $id$, $M$) $\rightarrow$ $C$: For a message $M \in \mathcal{M}$, the algorithm works as follows:

1. Check $X, Y \in G_1^*$.
   [Here, $pk := (X, Y)$.]
2. Check $e(X, p_0) = e(Y, g)$.
3. Compute $Q = H_1(id)$.
4. Choose a random number $r \overset{\$}{\leftarrow} \mathbb{Z}_q^*$.
5. Compute $C = (r \cdot g, M \oplus H_2(e(Q, Y)^r))$.
6. return $C$.

CLE.Extract($mpk$, $msk$, $id$) $\rightarrow$ $sk'$: It works as follows:

1. Compute $Q = H_1(id)$.
2. Set $sk' := s \cdot Q$.
3. return $sk'$.

CLE.Dec($mpk$, $sk$, $sk'$, $C$) $\rightarrow$ $M$: For a received ciphertext $C = (U, V)$, the algorithm works as follows:

1. Compute the private key $S_{id} = (sk \cdot sk') = sk \cdot s \cdot Q$.
2. Decrypt message as follows:
   
   
   $V \oplus H_2(e(S_{id}, U))$  
   $= V \oplus H_2(e(sk \cdot s \cdot Q, r \cdot g))$  
   $= V \oplus H_2(e(H_1(ID), g)^{sk \cdot r})$  
   $= M \oplus H_2(e(Q, Y)^r) \oplus H_2(e(H_1(ID), g)^{sk \cdot r})$  
   $= M \oplus H_2(e(H_1(ID), g)^{sk \cdot r}) \oplus H_2(e(H_1(ID), g)^{sk \cdot r}) = M$.

References

1. Aadhaar. https://uidai.gov.in/ (last accessed April 2021)
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) Advances in Cryptology—ASIACRYPT 2003. In: 9th International Conference on the Theory and Application of Cryptology and Information Security, volume 2894 of Lecture Notes in Computer Science, pp. 452–473. Springer, Berlin Heidelberg (2003).
3. Banta, N.M.: Inherit the cloud: the role of private contracts in distributing or deleting digital assets at death. Fordham Law Rev. 83, 799–854 (2014)
4. Brubaker, J.R., Callison-Burch, V.: Legacy contact: designing and implementing post-mortem stewardship at Facebook. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pp. 2908–2919. ACM (2016)
5. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. 13(1), 143–202 (2000)
6. Comer, J.: Digital life after death: the issue of planning for a person’s digital assets after death. Estate Plan. Commun. Property Law J. 3, 301–324 (2010)
7. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.). Advances in Cryptology—CRYPTO ’89, 9th Annual International Cryptology Conference, volume 435 of Lecture Notes in Computer Science, pp. 416–427. Springer (1989)
8. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory 22(6), 644–654 (1976)
9. DigiLocker Team.: DigiLocker. https://digilocker.gov.in/ (last Accessed April 2021) (2015)
10. DigiPulse Team.: DigiPulse: digital inheritance service. https://cryptorating.eu/whitepapers/DigiPulse/whitepaper.pdf (Last Accessed April 2021) (2018)
11. GDPR-EU.: General data Protection Regulation-European Union. https://eugdpr.org/ (last Accessed July 2020)
12. Gilden, A.: The social afterlife. Harvard J. Law Technol. (Forthcoming)
13. Goldreich, O.: Foundations of Cryptography: Volume 1, Basic Tools. Cambridge University Press (2007)
14. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 381–398 (1988)
15. Harbinja, E.: Digital inheritance in the United Kingdom. J. Eur. Consum. Market Law (EuCML) (2017)
16. Harbinja, E.: Post-mortem privacy 2.0: theory, law, and technology. Int. Rev. Law Comput. Technol. 31(1), 26–42 (2017)
17. Hopkins, J.P.: Afterlife in the cloud: Managing a digital estate. Hastings Sci. Technol. Law J. 5, 209 (2013)
18. Horton, D.: Tomorrow’s inheritance: the frontiers of estate planning formalism. BCL Rev. 58, 539 (2017)
19. Klasiček, D.: What happens to your Gmail and Facebook account after you die? In: Proceeding of Economics of Digital Transformation Conference, pp. 37–56 (2018)
20. Knapman, H.: What happens to your digital assets when you die? https://www.moneywise.co.uk/work/everyday-life/what-happens-your-digital-assets-when-you-die (Last Accessed April 2021) (2017)
21. Kutler, N.: Protecting your online you: a new approach to handling your online persona after death. Berkeley Technol. Law J. 26, 1641 (2011)
22. McCallig, D.: Facebook after death: an evolving policy in a social network. Int. J. Law Inf. Technol. 22(2), 107–140 (2014)
23. McCarthy, L.: Digital assets and intestacy. BUS Sci. Technol. Law J. 21, 384 (2015)
24. Mikl, T., Sein, K.: Digital inheritance: Heirs’ right to claim access to online accounts under Estonian law. Juridica Int. 27, 117 (2018)
25. Mirshekari, A., Ghasemi, R., Abedi, A.: Inheritance of digital accounts. Natl. J. Cyber Sec. Law 3(1) (2020)
26. MoneyLife Digital Team.: Unclaimed assets with financial regulators. https://www.moneylife.in/article/a-whopping-rs36000-crore-of-peoples-unclaimed-money-lying-with-just-three-financial-regulators/57587.html (Last Accessed April 2021) (2019)
27. Öhman, C., Floridi, L.: The political economy of death in the age of information: a critical approach to the digital afterlife industry. Mind. Mach. 27(4), 639–662 (2017)
28. O’Neill, B.: Document your digital assets. https://njaes.rutgers.edu/sshw/message/message.php?p=Finance&m=338 (last Accessed April 2021) (2016)
29. Park, Y.J., Sang, Y., Lee, H., Jones-Jang, S.M.: The ontology of digital asset after death: Policy complexities, suggestions and critique of digital platforms. In: Digital Policy, Regulation and Governance (2019)
30. PassOn Team. PassOn.: Inventing digital inheritance. https://passon.com/content/home/modules/1-intro-1af8iji/passon-white-paper.pdf (last Accessed April 2021) (2019)
31. PBC Library: The Pairing-Based Cryptography Library. https://crypto.stanford.edu/pbc/, (last Accessed Dec 2021)
32. PyCryptodome. https://pycryptodome.readthedocs.io (last Accessed Dec 2021)
33. Resta, G.: Personal data and digital assets after death: a comparative law perspective on the BGH Facebook ruling. J. Eur. Consum. Market Law 7(5) (2018)
34. Ronderos, J.: Is access enough: addressing inheritability of digital assets using the three-tier system under the revised uniform fiduciary access to digital assets act. Trans. Tennessee J. Bus. Law 18, 1031 (2016)
35. SafeHaven Team.: Safe haven: the solution to digital inheritance. https://safehaven.io/files/SafeHaven_WhitePaper.pdf (last Accessed April 2021) (2019)
36. Scolyer-Gray, P., Shaghaghi, A., Ashenden, D.: Digging your own digital grave: how should you manage the data you leave behind? https://theconversation.com/digging-your-own-digital-grave-how-should-you-manage-the-data-you-leave-behind-143755 (last Accessed April 2021) (2020)
37. TrustVerse Team.: TrustVerse: AI-wealth management & digital estate planning protocol backed by blockchain. https://icosbull.com/eng/ico/trustverse/whitepaper (last Accessed April 2021) (2018)
38. Walker, M.D.: The new uniform digital assets law: estate planning and administration in the information age. Real Property Trust Estate Law J. 52(1):51–78 (2017)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.