IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Digital Object Identifier

UReum: A Universally Composable Blockchain-enabled Model for Secure and Privacy-Preserving Data Awareness in Energy Internet

ABUBAKAR SADIQ SANI[1], (Member, IEEE), DONG YUAN[2], (Member, IEEE), George Loukas[1], (Fellow, IEEE), and Zhao Yang Dong[3], (Fellow, IEEE)

[1]School of Computing and Mathematical Sciences, University of Greenwich, London, SE10 9LS, United Kingdom (email: [s.sani, g.loukas]@gre.ac.uk)
[2]School of Electrical and Information Engineering, The University of Sydney, Sydney, NSW 2006, Australia (email: dong.yuan@sydney.edu.au)
[3]Department of Electrical Engineering, City University of Hong Kong, Hong Kong (email: zydong@ieee.org)

Corresponding author: Abubakar Sadiq Sani (e-mail: s.sani@gre.ac.uk).

ABSTRACT Recent findings show that many energy nodes rely on energy operators for inputs on energy operations. In this paper, we introduce UReum, a universally composable blockchain-enabled secure and privacy-preserving data awareness solution by which energy nodes can provide visibility into energy operations without involving or relying on energy operators and leaking sensitive information in the Energy Internet. In the energy ecosystem, Energy Internet represents an advanced internet-based energy system that aims to enable complex interconnection and interaction amongst energy nodes. UReum consists of a registration protocol ($RPro$) for assigning a cryptographic identity to an energy node and a data-aware protocol ($DAPro$) for executing data awareness with the support of a shared secret session key and UReum smart contracts, which facilitate data awareness consensus amongst energy nodes. UReum satisfies Energy Internet's data awareness security and privacy requirements. The security requirements include correctness of data, assurance of energy node identity, and fairness of data awareness transactions such as energy node registration, while the privacy requirements include anonymity of energy node identity, the confidentiality of data, and unlinkability of data awareness transactions. We evaluate our model with respect to security, privacy, and performance, and the results show that our model is suitable for the Energy Internet. As a proof of concept, we apply our model to mitigate the loss of State Estimator (SE) and loss of Energy Management System (EMS) issues in real-world energy grids.

INDEX TERMS Data awareness, energy internet, privacy, security, universal composability.

I. INTRODUCTION

THE Energy Internet [1] is the next-generation energy system that aims to provide better interconnection and improve energy efficiency in the energy ecosystem and further support multi-directional communications of data, which enables energy nodes or devices to share energy resources such as renewable energy. The Energy Internet architecture supports ethernet interfaces, information routers that route data, and open-standard operating systems capabilities [1]. Data awareness is an essential feature in Energy Internet. It enables energy nodes to provide insights into energy operations such as energy storage and trading. It can be achieved by the support of energy operators – such as service providers – that are responsible for providing means of data storage via their distributed databases. Energy data such as energy dispatch and load information can be included in the data awareness. Security and privacy of data awareness can be relatively easy to compromise due to the heavy reliance on the energy operators. Without understanding the behaviour of adversaries that can carry out such a compromise, it is very difficult for the Energy Internet to provide appropriate mitigation measures. This paper takes a step in this direction by analysing adversarial behaviour and providing security and privacy during data awareness in the Energy Internet.

Security situational awareness [2] is a widely adopted means for data awareness on security attacks in the energy grids. Satisfying data awareness security and privacy requirements such as correctness and confidentiality, respectively, make it possible for energy nodes to support security and privacy during data awareness. However, the constant involvement of energy operators and critically of communications amongst energy nodes in providing energy operations insights make data awareness a high potential target for compromise by adversaries. Moreover, the use of preshared keys, which

are issued to energy nodes by the energy operators, for secure communications introduces huge security and privacy concerns during data awareness in the Energy Internet.

Limited progress has been made regarding the development of distributed solutions for mitigating compromised attacks and detecting the presence of adversaries during data awareness in the Energy Internet. Recent studies on Energy Internet have shown that data awareness solutions are needed to support its rapid advancement in the real world. Additionally, Energy Internet security solutions like [3] lack integrated secure and privacy-preserving data awareness functionality and their shortcomings could be exploited by adversaries to compromise the energy grid. Thus, adequate security and privacy measures are required to enhance data awareness security and privacy, respectively in the Energy Internet. A simple data awareness architecture of the Energy Internet is presented in Figure 1. This figure shows data awareness via collecting and reporting of data amongst energy operators. Based on the multi-directional communications amongst the energy nodes, data can be sufficiently exchanged. Then, data awareness can be realized based on the insight of the exchanged data.

We stress that the lack of solutions for detecting the presence of adversaries during data awareness as well as lack of secure, privacy-preserving, and efficient multi-directional communications are major hindrances towards the wide integration of energy nodes as well as energy operators for distributed data awareness in the Energy Internet. To address these challenges, this paper proposes UReum, a secure and privacy-preserving data awareness system for Energy Internet based on universal composability and blockchain, which is a distributed technology that offers fascinating possibilities for managing the security and privacy of data awareness in Energy Internet. UReum uses a universal composability model [4], [5], which allows modular design and analysis of cryptographic protocols, to provide security in arbitrary adversarial environments and detect adversaries in these environments. In the universal composability model, smart grid protocols can use ideal functionalities as subroutines to perform their cryptographic operations. Furthermore, UReum is supported by an Elliptic Curve Cryptography (ECC) [6] algorithm, which we referred to as a privacy-preserving Elliptic Curve Diffie-Hellman key exchange (P-ECDH), which enforces anonymity, confidentiality, and unlinkability, to mitigate privacy attacks during data awareness. More specifically, our main contributions are as follows.

- We propose UReum transactions and smart contracts for data awareness support and enforcement, respectively. The transactions and smart contracts use lightweight cryptographic algorithms to support and maintain consensus, respectively, and enhance the security and privacy of data awareness in an efficient manner.
- We propose an ideal functionality F_{CR} for cryptographic operations related to energy node registration and data awareness. F_{CR} supports many cryptographic primitives such as our P-ECDH, which is based on the properties of Zero-Knowledge Proof of Knowledge (ZKPK) [7] and standard ECDH that exhibits very low computational and communication costs. We show that F_{CR} can be realized under standard cryptographic assumptions. Furthermore, we propose and prove a realization P_{CR} of F_{CR} based on standard cryptographic assumptions. The proof consists of several hybrid arguments as F_{CR} supports a wide range of operations.
- We propose an ideal functionality F_{DA}^{RP} for secure and privacy-preserving data awareness with perfect forward secrecy and efficient transition from energy node registration to data awareness.
- We provide UReum, which consists of the registration protocol $(RPro)$ for securely enrolling a new energy node and a data-aware protocol $(DAPro)$ for creating secure and privacy-preserving data awareness in the Energy Internet. UReum utilises UReum transactions and smart contracts, uses F_{CR}, and realizes F_{DA}^{RP} to provide secure and privacy-preserving data awareness and mitigate privacy and active man-in-the-middle attacks in the Energy Internet.
- We perform some experiments to analyse the performance of UReum and then show that it meets the latency requirements of energy system applications and technologies such as wide-area situational awareness and state estimation.
- We analyse the issues on the loss of State Estimator (SE) and loss of Energy Management System (EMS) in real-world energy grids and then use UReum to mitigate the issues.

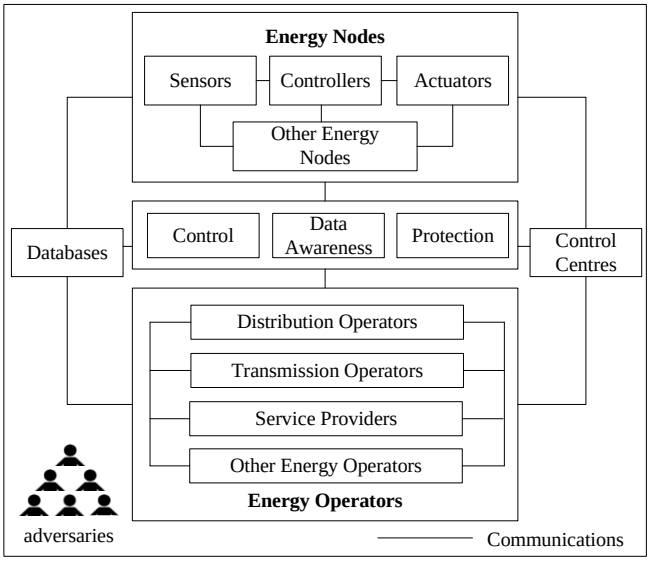

FIGURE 1. A simple data awareness architecture of the Energy Internet.

The rest of this paper is organised as follows: Section II provides an overview of the related works. Section III provides the cryptographic techniques we use in this work and presents a data and communication model and a threat

model. In Section IV, we present the essential blockchain components for secure and privacy-preserving data awareness. In Section V, we present our ideal functionality F_{CR} and its realization P_{CR}. The proof of P_{CR} is also presented in this section. In Section VI, we present our ideal functionality F_{DA}^{RP} for secure and privacy-preserving data awareness. In Section VII, we present UReum. In Section VIII, we present the security analysis of UReum. In Section IX, we present the implementation and experimental results for our model. In Section X, we present the case studies. Finally, we conclude and outline future work in Section XI.

II. RELATED WORKS

Many data awareness solutions have been proposed for the energy grid, mainly situational awareness, status awareness, computational awareness, and social awareness approaches. These approaches have different advantages (such as providing real-time information for supporting energy grid management [8], [9]) and limitations (such as lack of data privacy [8], [10] as well as lack of providing security in arbitrary adversarial environments [9], [11]). Liu et al. [8] proposed a battery status-aware authentication scheme for collecting information regarding the states of batteries used in the energy grid. However, the scheme relies on a central authority and preshared secret keys for battery status awareness thereby presenting a single point of failure and privacy concerns, respectively, in the energy grid. Furthermore, the scheme incurs extra communication and computational costs because of the preshared keys and thus is not practical for resource-constrained energy nodes. We argue that it is very important for data awareness solutions for smart grid as well as Energy Internet to meet the latency requirements of energy system applications and technologies which range from 20 msec to 2 sec [12], [13], [14].

Lin et al. [9] presented a situation awareness technology and its application in a practical Active Distribution Management System (ADMS), which is used to monitor and control distribution networks in the energy grid. While the technology supports the perception and synthesisation of information, and projection of new future behaviours of Active Distribution Network (ADN), which aggregates and manages distributed resources in the energy grid, it lacks the capability of detecting the presence of adversaries in the ADN. Additionally, the security evaluation issue is identified as one of the bottlenecks of implementing such a technology for the ADN.

Ghosh et al. [10] proposed a situational awareness mechanism for observing the energy grid based on the Markov approach, which efficiently describes dependencies amongst energy nodes for increased reliability and performance analyses and decision-making. However, the mechanism does not offer security and privacy during situational awareness. Besides, the increasing integration of internet technology into the energy grid shows that situational awareness mechanisms with embedded security and privacy are required to enhance detection of adversaries and further mitigate cyber-attacks such as privacy and active man-in-the-middle attacks.

Wu et al. [11] proposed a big data-enabled security situational awareness mechanism to mitigate threats that can disrupt normal operations in the energy grid. However, the mechanism is vulnerable to active man-in-the-middle attacks during data collection from the communication network of the energy grid. Additionally, realizing security situational awareness using game theory provides expressivity and flexibility in defining security properties, however, in general, game-based models do not enjoy built-in modularity. By contrast, our solution provides protection against active man-in-the-middle attacks during data collection and has built-in modularity via universal composability by simplifying an analytic process and providing more expressive security.

Tsai et al. [15] provided a detailed discussion about smart grid computational awareness, which focuses on providing more intelligent and reliable smart grid services. The authors identified security challenges as open issues that need to be solved to realize a high-end computational awareness securely and sufficiently in the energy grid. Chawla and Kowalska-Pyzalska [16] discussed public awareness of smart meters in the energy grid and presented some recommendations such as addressing ensuring security and privacy of personal data. Singh et al. [17] highlighted the need for social awareness to be integrated into energy grid operation – with privacy and security protections – to enhance demand-side management and emergency management. Nambi and Prasad [18] proposed a Techno-Social Smart Grid (TSSG) framework to enhance consumer awareness and active participation in the energy grid with considerations on addressing key challenges such as data security and privacy. This paper takes a step in the directions presented in [15], [16], [17], and [18] via providing security and privacy during data awareness. Zhu et al. [19] presented an overview of a situational awareness tool, FNET/GridEye [20], deployed at the distribution level of the energy grid. The tool is used to take measurements from energy outlets and transfer them to a data centre via the Internet. Some of the major limitations of the tool are the lack of data immutability, security, and privacy during situational awareness.

Sani et al. [21] proposed SDAG, blockchain-enabled model for secure data awareness in smart grids. This paper builds up on SDAG by providing universal composability design and analysis, introducing and satisfying privacy requirements (such as confidentiality, anonymity, and unlinkability), introducing ideal functionalities for essential cryptographic operations and secure and privacy-preserving data awareness, providing a practical implementation of secure and privacy-preserving data awareness, and demonstrating a case study on the loss of energy management system in a real-world energy grid.

Although the proposed solutions, overviews, and recommendations succeeded in providing data awareness for the energy grid, they still suffer from one or more of the following drawbacks: i) solution analysis based on game-based models do not offer cryptographic soundness and security in

arbitrary adversarial environments (see, e.g., [11]); ii) lack of mitigating privacy and active man-in-the-middle attacks (see, e.g., [8], [10]); iii) solution does not provide data immutability (see, e.g., [9], [20]); iv) reliance on a central authority and/or preshared keys leads to high communication and computational complexities (see, e.g., [8]); and v) lack of evidence of meeting the latency requirements of energy grid applications and technologies (see, e.g., [8], [9], [10], [11], [21]). In this paper, we propose a solution that addresses those drawbacks, and more importantly, provide data immutability and efficiently secure and privacy-preserving data awareness in the Energy Internet. Furthermore, we present case studies of the loss of SE and loss of EMS issues in which the use of our model allows us to mitigate the issues.

III. PRELIMINARIES

In this section, we introduce the main cryptographic techniques, which are used as building blocks in our model, present a data and communication model for data awareness in the Energy Internet, and describe a threat model for data awareness security and privacy. The notations used in this paper are listed in Table 1.

A. ELLIPTIC CURVE PEDERSEN COMMITMENT SCHEME

The Elliptic Curve Pedersen Commitment Scheme (EC-PCS) is an efficient implementation of the Pedersen Commitment Scheme (PCS) [22]. The EC-PCS uses ECC based on the elliptic curve discrete logarithm assumption. In this scheme, a committer commits to secrets such that it is hard for a verifier to open the commitment. The description of the EC-PCS is presented in the following steps:

1) Setup

Let F_p be an elliptic curve, where p is a large prime of 128 bits. Let Z_p be an integer group of order p. Let $G \in F_p$ be a random generator point of order n and $H \in F_p$ be a chosen generator point of order n such that it is computationally hard to find $H = u.G$, where $u \in Z_p$ is a random secret. Then, F_p's domain parameters (p, ai, bi, G, H, n, h) are published by a master energy node M_1 in the Energy Internet, where ai and bi are curve parameters and h is a cofactor. Note that: (I) M_1 is a registered energy node with adequate computational resources in the Energy Internet. (II) A registered energy node is an energy node with a computed cryptographic identity (see Section IV for a description of a registered energy node).

2) Commit

The committer creates a commitment C of $u \in Z_p$ by randomly choosing $v \in Z_p$ and computing $C(u, v) = u.G + v.H$, where v is a random secret.

3) Reveal

To confirm the authenticity of C, the committer reveals u and v and the verifier checks if $C = u.G + v.H$.

The EC-PCS and PCS have similar properties as follows: i) Perfectly hiding, i.e., every possible random secret u is equally committed in C; and ii) Computationally binding, i.e., a random secret u cannot open C since $u' \neq u$ unless one can solve the Elliptic Curve Discrete Logarithm Problem (ECDLP), which represents that it is difficult to guess u from $H = u.G$, where u is an integer. In this paper, we adopt the

TABLE 1. Notations used in this paper

Symbol	Description
p	Large prime
F_p	Elliptic Curve
Z_p	Integer group of order p
G	Random generator point
H	Chosen generator point
ai and bi	Curve parameters
n	Order of G
h	Cofactor
$u; v; s; ti; o$	Random secrets
C	Commitment
CC	Cryptograhic Commitment
g	Generator of G_p
G_p	Subgroup of Z_p
q	Element of G_p
e	Random challenge
$d; a; b$	Computed values
nn	Large random selected integer
$pv; pb$	Private key (pv); Public key (pb)
$k; ty$	Shared secret key (k); Key type (ty)
$Sy; Sz; M$	System of IITM (Sy, Sz); Machine (M)
$S; E$	Adversarial system (S); Envi. system (E)
$F; P$	Ideal protocol (F); Real protocol (P)
$s1; s2; En$	Site $(s1)$; Site $(s2)$; Entity (En)
ZM_i	Message payload size
$extra$	Additional security feature
RT	Register transaction
DT	Data-aware initiation transaction
ST	Store transaction
DRT	Data-request transaction
RVT	Revoke transaction
SSC	Store smart contract
$DRSC$	Data-reply smart sontract
$RVSC$	Revoke smart contract
$name$	Name of an energy node
ID	Cryptographic identity of an energy node
pe	P-ECDH value
t	Transaction
th	Transaction header
td	Transaction information
nn	Large randomly selected integer
rv	Random value
$ptr; nptr$	Pointer (ptr); Notification pointer $(nptr)$
$sid; r$	Session identifier (sid); Role/tape (r)
η	Security parameter
$Gen(1^\eta)$	Elliptic curve domain parameters algorithm
$Hash(.)$	Secure Hash Algorithm (SHA-256)
$msg; emsg; rmsg; \alpha$ $imsg; dmsg; vmsg$	Message
$\beta; \phi$	Signed/MACed value (β); Hashed value (ϕ)
m	Random bit string
$Enc(.)$	128 bits AES algorithm encryption part
$Dec(.)$	128 bits AES algorithm decryption part
$MAC(.)$	MACing part of a 256-bit MAC algorithm
$VMAC(.)$	MAC verficiation algorithm
$Sig(.)$	Digital signature part of a 160 bits ECDSA
$VSig(.)$	Verification part of $Sig(.)$'s ECDSA
$S(.)$	Set of energy nodes
$F'; F''; F'''$	Pseudorandom functions

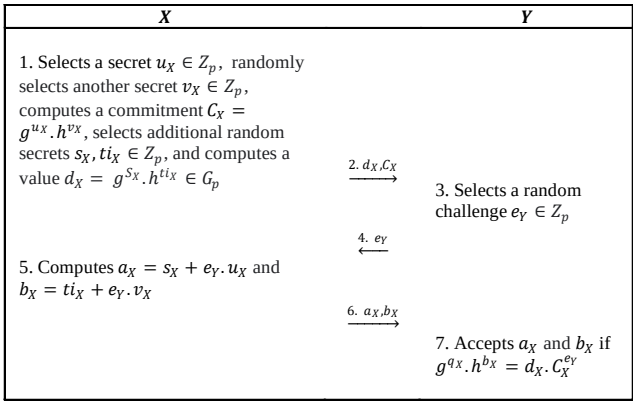

FIGURE 2. ZKPK between a prover, say *X*, and a verifier, say *Y*.

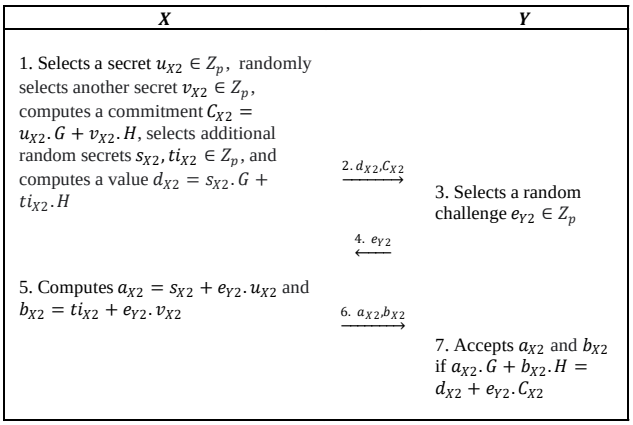

FIGURE 3. EC-ZKPK between a prover, say *X*, and a verifier, say *Y*.

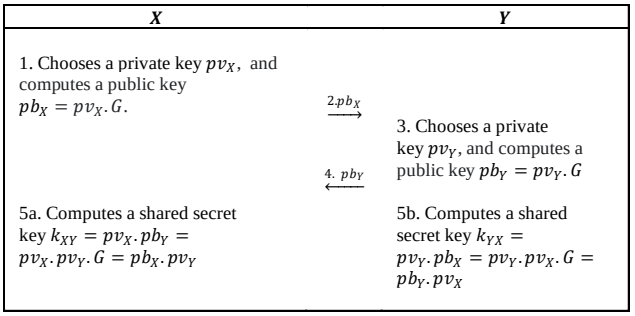

FIGURE 4. ECDH between energy nodes *X* and *Y*.

EC-PCS to support the derivation of a cryptographic identity for an energy node in our *RPro*.

B. ZERO KNOWLEDGE PROOF OF KNOWLEDGE

The ZKPK [7] is a cryptographic protocol by which the owner of a secret proves to a verifier his/her knowledge about the secret without revealing the secret or making it easier for the verifier to obtain the actual secret. To execute the ZKPK, an additional setup is required by M_1 based on F_p as follows: M_1 chooses and publishes the additional parameters (g, q), where g is a generator of G_p subgroup of Z_p and $q = g^o \ mod \ p$ is an element of G_p and $'o'$ is a secret. The ZKPK operations between a prover, say X, and a verifier, say Y, are provided in Figure 2. The properties of ZKPK are as follows: i) Completeness, i.e., the protocol succeeds if X and Y are honest; ii) Soundness, i.e., the protocol does not allow X to prove a false statement; and iii) Zero-knowledge, i.e., no information about the secrets are leaked by the proof.

In this paper, we utilise an efficient ZKPK, i.e., the Elliptic-Curve ZKPK (EC-ZKPK), to enhance the efficiency of the ZKPK and make it easier to use by energy nodes in the Energy Internet. We also use the EC-ZKPK to support privacy preservation in our *DAPro*. The EC-ZKPK is a protocol based on the elliptic curve F_p, which also takes into consideration additional parameters. The description of the EC-ZKPK is presented in Figure 3.

C. ELLIPTIC CURVE DIFFIE-HELLMAN KEY EXCHANGE

The ECDH is a key exchange protocol based on the elliptic curve F_p as presented above. In the ECDH, energy nodes X and Y can generate their public and private key pair. The private key of X is a randomly chosen value pv_X from $\{1, ..., n-1\}$ and the public key is computed by $pb_X = pv_X.G$. Similarly, Y has a private key pv_Y and a public key pb_Y. The ECDH operations are provided in Figure 4. At the end of the ECDH, X and Y can now use $k_{XY} = k_{YX}$ as a shared secret key for securing subsequent communications between them during data awareness. The key property of the ECDH is completeness, i.e., the protocol succeeds if X and Y are honest. In this paper, we use ECDH to support shared secret key derivation in our *DAPro*.

D. UNIVERSAL COMPOSABILITY

In this section, we briefly describe the general notion of universal composability and the universal composability model we use in this paper.

1) General Notion of Universal Composability

We have real and ideal protocols/functionalities in universal composability models. An ideal protocol, also known as ideal functionality, represents the desired behaviour and intended security properties of a protocol. The real protocol, also known as real functionality, represents the protocol to design and analyse and should be at least as secure as the ideal protocol, i.e., it realizes the ideal protocol. For the ideal protocol, there exists an ideal adversary or a simulator while a real adversary exists for the real protocol, such that there is no environment that can distinguish between real and ideal settings. Thus, by definition of universal composability, attacks on the real protocol cannot be successful since there exists no successful attack on the ideal protocol.

2) Inexhaustible Interactive Turing Machine Model

In this paper, the universal composability model we use is the Inexhaustible Turning Machine (IITM) model [4], [5] with responsive environments from [23]. It consists of a general computational model and provides several composition theorems. The general computational model is defined by systems of interactive Turing machines. An interactive Turing machine (or a machine) is defined as a probabilistic polynomial-time Turing machine with named bidirectional tapes. In a system $Sy = M_1|...|M_k|!M_1^{'}|...|!M_k^{'}$ of IITM model, where M_1 and M_k^i are machines and ! indicates that an unbounded number of fresh copies of machines may be generated in a run of Sy. A machine M_1 can be triggered by another machine $M_1^{'}$ if $M_1^{'}$ receives a message on a tape that connects them. Sy will always have polynomial runtime in the security parameter η. Two systems Sy and Sz are called indistinguishable, i.e., $Sy \equiv Sz$, if the difference between the probability that Sy outputs 1 and the probability that Sz outputs 1 is negligible in the security parameter η.

There are three different types of systems in the IITM model: i) real and ideal protocols/ functionalities; ii) adversaries and simulators; iii) and environments. The real and ideal protocols and the environmental systems have input/output (I/O) and network interfaces or tapes, while the adversarial systems have only network tapes. We say that the environmental and adversarial systems are responsive if they immediately respond to so-called *restricting messages* on the network. Restricting messages are represented in the form $(\text{Respond}, id, m)$, where id and m are random bit strings. They are used for enforcing a natural execution of a protocol by preventing the adversary from disrupting or interfering with the protocol execution. In general, restricting messages improves the expressivity of universal composability models. Furthermore, we define strong simulatability in the IITM model which allows for omitting the adversary and showing that the real protocol realizes or emulates the ideal protocol.

Definition 1 (Strong simulatability) [24]. *Let P and F be protocol systems with similar I/O interface, i.e., the real and ideal protocol, respectively. Then, P realizes $F (P \leq_R F)$ if there exists an adversarial system S (an ideal adversary or a simulator) such that P and $S|F$ have similar external interfaces and for all environmental systems E, connecting to the external interface of P (and thus, $S|F$), it holds true that $E|P \equiv E|S|F$, where S is subsumed by E in P.*

Furthermore, the IITM model provides several composition theorems. We present Kusters's composition theorem that handles the concurrent composition of a fixed number of protocol systems.

Theorem 1 [24]. *Let P_1, P_2, and P_3 be real protocol systems and let F_1, F_2, and F_3 be ideal protocol systems such that P_1, P_2, and P_3, as well as F_1, F_2 and F_3 only connect with each other via their I/O interfaces and $P_i \leq F_i$, for $i \in \{1, 2, 3\}$. Then, it holds true that $P_1|P_2|P_3 \leq F_1|F_2|F_3$.*

To construct more complex systems using the IITM model, other composition theorems in [4] and [25] can be combined and applied in an iterative manner.

E. DATA AND MULTIDIRECTIONAL COMMUNICATION MODEL

Real-time communications amongst energy nodes are required to provide insights into the energy operations. In this case, transport layer protocols are utilised to support end-to-end data exchange for data awareness in the Energy Internet. User Datagram Protocol/Internet Protocol (UDP/IP) paradigm is considered for communication amongst the energy nodes due to its latency requirements, congestion control mechanism, and protection against data integrity errors [26]. A simple data and multidirectional communication model in the Energy Internet is presented in Figure 5. As shown in Figure 5, energy nodes (i.e., a sender and receivers) send and receive data using a variety of channels (such as Point-To-Point (P2P) and Carrier Sense Multiple Access (CSMA)). The sender (i.e., energy node X) transmits data to the intended receiver (i.e., energy node Y) and/or the intended receivers (i.e., energy nodes Ns). We note that: (I) The data and multidirectional communication model for data awareness shown in Figure 5 does not include any security and privacy features – for example, all the energy nodes send data without any security feature (such as Message Authentication Code (MAC) for authentication and integrity) and their real identities are transferred in plain text. (II) We use the model to analyse the challenges in satisfying data awareness security requirements such as correctness (i.e., prevent counterfeit data creation and active man-in-the-middle attacks), assurance (i.e., prevent impersonation and identity compromise), and fairness (i.e., prevent skipping of appropriate data awareness transactions (including unavailability of data) and further ensure that energy nodes have access to real-time data) as well as privacy requirements such as anonymity, confidentiality, and unlinkability. In this paper, the size of a data packet from an energy node can be calculated as $ZM_i + extra$, where ZM_i is the size of the message payload in the energy grid and $extra$ represents additional features added to the data packet (say, 256-bit MAC for security and 128-bit Advanced Encryption Standard (AES) for privacy). We note that a minimum ZM_i

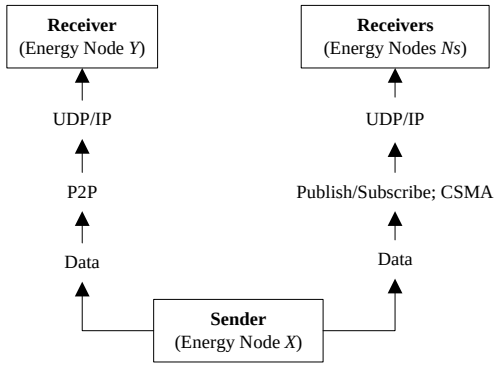

FIGURE 5. Data and multidirectional communication model.

of 33 bytes for UDP can be used for low power wireless communication in the energy grid [27], [28]. In this paper, we use 33 bytes for ZM_i without loss of generality (see Section IV for more details).

F. THREAT MODEL

We identify the different ways in which an adversary or malicious energy node can compromise the security of the data and multidirectional communication model as follows: (I) Correctness compromise: An adversary can compromise data from an energy node (i.e., the sender) during data exchange by replacing the data with a different/counterfeit one thereby leading to an active man-in-the-middle attack. (II) Assurance compromise: For data exchange between energy nodes, one of the nodes (say X) can be impersonated by the adversary or malicious energy node thereby affecting and compromising the assurance of X's identity. Furthermore, fake information, which does not reflect data from X, can be provided to mislead other energy nodes. (III) Fairness compromise: The malicious energy node can skip a data awareness thereby compromising the provision and availability of real-time data which can affect other energy nodes that are depending on such data for their operations.

To guarantee the security of data awareness, correctness compromise, assurance compromise, and fairness compromise are key challenges to be addressed. We briefly describe some simple mechanisms that should be incorporated into data awareness to address the challenges accordingly as follows: (I) Correctness: A shared secret key can be established and applied for data MACing or signing to provide data integrity and authenticity. (II) Assurance: A verification of every energy node identity along with some (identity) artefacts such as random values can be carried out to provide identity and data assurance. (III) Fairness: Energy nodes can verify the execution of appropriate data awareness transactions (including the availability of data) and their outcomes to detect malicious behaviours. Also, the outcomes should be made available on a distributed and transparent system such as a blockchain to support real-time data availability and access.

We now discuss the goals of the adversary or malicious energy node in compromising the privacy of the model as follows: (I) Anonymity compromise: When anonymity of identities of energy nodes who exchange data are not preserved, the anonymity of the identities can be compromised by an adversary or a malicious energy node. (II) Confidentiality compromise: The adversary or malicious energy node learns about the plaintext data exchanged amongst energy nodes. (III) Unlinkability compromise: The adversary or malicious energy node learns about the data awareness transactions and energy nodes involved in such transactions. In some cases, the energy nodes usually apply information such as secrets issued by a trusted authority to provide a form of security for data exchange. In this case, such information can still be exposed to the adversary or malicious energy node before or during data exchange amongst the energy nodes.

The above privacy challenges, i.e., anonymity compromise, confidentiality compromise, and unlinkability compromise, are to be addressed to guarantee the privacy of data awareness. We provide some mechanisms to support the incorporation of privacy features into data awareness as follows: (I) Anonymity: A cryptographic hash-supported energy node identity can be used during data awareness to preserve the identity of the energy nodes. (II) Confidentiality: An encryption algorithm can be used to encrypt sensitive plaintext data during data awareness to prevent the adversary or malicious energy node from learning about the data. (III) Unlinkability: Shared secret keys can be derived by energy nodes to prevent the adversary or malicious energy node from learning about the transactions and associated energy nodes.

IV. ESSENTIAL BLOCKCHAIN COMPONENTS FOR SECURE AND PRIVACY-PRESERVING DATA AWARENESS

In this section, we present the main components of UReum which consist of energy nodes, UReum blockchain, UReum transactions (i.e., data awareness transactions), and UReum smart contracts.

A. ENERGY NODES

There are three different types of energy nodes: (I) Basic energy nodes, which send and receive transactions, but neither manage nor store the transactions. (II) Master energy nodes, which act as managers that manage and store transactions and smart contracts. (III) Edge server, which only provides system bootstrapping that initializes all transactions and smart contracts. The basic energy nodes can be considered as energy devices while the master energy nodes are distributed systems or servers with identical information and are equipped with adequate computational and storage resources to enhance data access and availability. For example, an energy sensor and a meter data management system can be considered as a basic energy node and a master energy node, respectively. We note that since the edge server is only responsible for registering the first master energy node, the main energy nodes in this paper are the basic energy nodes and master energy nodes.

B. ENERGY OPERATORS

The energy operators represent all the operators that support energy operations in the energy grid. These operators include but are not limited to, generator operators, transmission operators, distribution operators, market operators, and service providers. Every energy operator manages dedicated energy operation(s) and/or the energy nodes involved in the operations. For example, a market operator manages energy markets and electricity systems and usually has a database of Distributed Energy Resources (DER) devices installed at end users' locations. For brevity, we say that energy operators are part of the Energy Internet as shown in Figure 1.

C. UREUM BLOCKCHAIN

UReum blockchain is used for data storage and verification as well as keeping track and record and querying of UReum

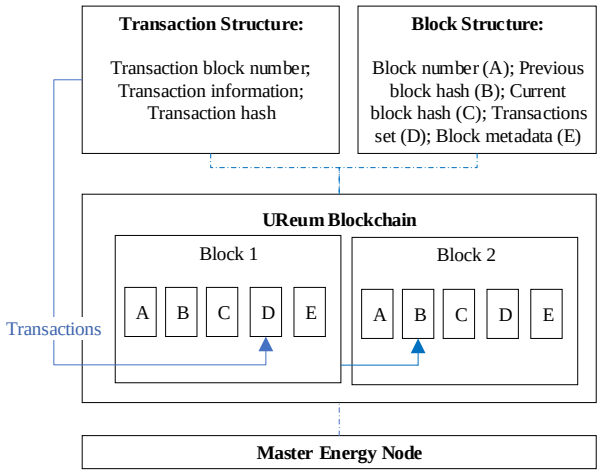

FIGURE 6. UReum Blockchain.

transactions, which are chained together and stored in blocks. It is managed by master energy nodes and can be accessed by all energy nodes for the above purposes. Without loss of generality, every UReum blockchain at each master energy automatically synchronises a copy of its new data with other UReum blockchains at other master energy nodes – to maintain identical information in real-time. Each block in the UReum blockchain as illustrated in Figure 6 consists of the following sections: i) Block number, denoted as A, represents the position of the block on the UReum blockchain; ii) Previous block hash, denoted as B, represents the cryptographic hash from the previous block; iii) Current block hash, denoted as C, represents the cryptographic hash of all transactions in the current block; iv) Transactions set, denoted as D, represents all the transactions in the current block; and v) Block metadata, denoted as E, represents the block's basic information (such as the signature of the creator of the current block which is used to verify the block by the energy nodes) and additional data (such as a notification pointer that points to a transaction in the block (see below)) that are not input to the block hash computation. Each transaction consists of the following sections: i) Transaction block number representing the block number associated with the transaction; ii) Transaction information representing the essential transaction information that is only stored in the UReum blockchain; and iii) Transaction hash representing the cryptographic hash of the transaction information. As shown in Figure 6, the master energy node collects transactions into Block 1. The hash of the block is appended to Block 2 once the block is full.

D. UREUM TRANSACTIONS

UReum transactions are a set of activities that support secure and privacy-preserving data awareness. For example, a new energy node can obtain a cryptographic identity from a master energy node without disclosing any associated secret keys to other energy nodes. These transactions include a register transaction, a data-aware initiation transaction, a store transaction, a data-request transaction, and a revoke transaction. The details about the transactions are as follows.

1) Register Transaction

A register transaction RT is generated by an (unregistered) energy node to obtain a cryptographic identity from a registered master energy node. Upon obtaining the cryptographic identity, the energy node becomes a registered energy node. RT is expressed as $RT = (name, pb_1, pb_2)$, where $name$ is the energy node's name of 32 bits and pb_1 and pb_2 are the energy node's public keys of 160 bits each. The protocol implementing RT is given in Section VII-A.

2) Data-aware Initiation Transaction

A data-aware initiation transaction DT is generated by a registered energy node to initiate a data awareness with another registered energy node. DT is expressed as $DT = (ID, CC, pe, pb)$, where ID is the registered energy node's identity of 160 bits, CC is the registered energy node's cryptographic commitment of 160 bits, pe is the energy node's P-ECDH value of 256 bits, and pb is the registered energy node's public key of 160 bits (see Section V for more details of P-ECDH value). The protocol implementing DT is given in Section VII-B.

3) Store Transaction

A store transaction ST is generated by a registered energy node to store a transaction at registered master energy nodes, which store the transaction in their respective UReum blockchain. ST is expressed in the form $ST = (Hash(t), t)$, where $t = (th_2, td_2, Hash(td_2))$ is a transaction to be stored in UReum blockchain, th_2 is a transaction header of 32 bits, td_2 is transaction information, say a message payload of 264 bits or 33 bytes (see Section III-E for more details), and $Hash(td_2)$ is td_2's cryptographic secure hash of 256 bits which is computed via a cryptographic secure hash algorithm (SHA-256) denoted as $Hash(.)$.

4) Data-request Transaction

A data-request transaction DRT is generated by a registered energy node to ask for data. DRT is expressed as $DRT = (MAC_{k_4}(Enc_{k_4}(xmsg)), Enc_{k_4}(xmsg))$, where $xmsg$ is an energy grid message of 64 bits, k_4 is a shared secret session key of 256 bits derived by ID_X and ID_Y, and $MAC(.)$ is a MACing part of a 256-bit MAC algorithm, and $Enc(.)$ is an encryption part of a 128 bits AES algorithm.

5) Revoke Transaction

A revoke transaction RVT is generated by an energy node to revoke a transaction or an identity of any energy node that provides false information about a transaction or misuse any cryptographic algorithm. RVT is expressed as $RVT = Sig_{pv_1}(t_0, name, ID)$, where t_0 is a violated transaction of an existing transaction t, name is the energy node's name of 32

bits (say, X) that violates t, ID is the energy node's identity of 160 bits (say, ID_X), pv_1 is a private key of 80 bits of a master energy node (say, ID_{M_1}) that generated RVT, and $Sig(.)$ is a digital signature part of a 160 bits Elliptic Curve Digital Signature Algorithm (ECDSA).

E. UREUM SMART CONTRACTS

UReum smart contracts are a set of agreements that facilitate and enforce data awareness. They are supported by the UReum transactions amongst the energy nodes and stored in the UReum blockchain. For example, a master energy node can store a new transaction in the UReum blockchain by using a UReum smart contract designed for storing transactions in the UReum blockchain. Cryptographic algorithms, such as SHA-256 $Hash(.)$ and 160 bits ECDSA, represent the consensus algorithms used by the energy nodes for verifying transactions. All data required to execute the smart contracts are obtained directly from the UReum blockchain to provide a high degree of assurance, enhanced data availability, and simplified execution of the smart contracts. These smart contracts include store smart contract (SSC), data-reply smart contract $(DRSC)$, and revoke smart contract $(RVSC)$. The details about the smart contracts are as follows.

1) Store Smart Contract

A store smart contract SSC is executed by a registered master energy node to store a new transaction in a UReum blockchain upon receiving a store transaction and its verifier from a registered basic energy node. We use $Hash(.)$ and $Sig_{pv}(.)$ as the cryptographic algorithms of SSC, where pv is a public key of a registered master energy node.

Let ID_X and ID_{M_1} be a registered basic energy node and a maser energy node, respectively, with a shared secret session key k_4. Let $t_2 = (th_2, td_2, Hash(td_2))$ be a new transaction, where th_2 is a transaction header of 32 bits, td_2 is a transaction information (say of 264 bits), and $Hash(td_2)$ is td_2's secure hash of 256 bits. Upon receiving a store transaction $ST = (t_2, Hash(t_2))$ and $ST_V = Hash(k_4, ST)$ of 256 bits from ID_X, where ST_V is the verifier for ST, ID_{M_1} first checks whether ID_X exists in the UReum blockchain and initiates a revoke transaction RT and executes a revoke smart contract $RVSC$ to revoke ID_X if the check fails (see below for more details of $RVSC$). Secondly, it checks whether $ID_X(revoked)$ does not exist and returns failed to ID_X if the check fails. Lastly, it checks whether ST does not exist and if the check fails, it returns a notification pointer $nptr1$ (of ST), i.e., $nptr1 = Hash(ST)$, to ID_X, where $nptr1$ points to ST in the UReum blockchain. If all the checks succeed, it computes $ST_{V2} = Hash(k_4, ST)$ and checks whether $ST_{V2} = ST_V$. If the check fails, it returns fails to ID_X. Otherwise, it inserts ST, pv_M, ID_{M_1}, $S(ID_{M_i})$, and $S(ID_{B_i})$ into SSC, where pv_M is a private key of ID_{M_1}, $S(ID_{M_i})$ is a set of all registered master energy nodes, and $S(ID_{B_i})$ is a set of all registered basic energy nodes. SSC takes into account ST format above and executes as follows: SSC generates a notification pointer $nptr1$ that points to ST, stores $ST = (Hash(t_2), t_2)$ and $nptr1$ in ID_{M_1}'s UReum blockchain, computes and broadcasts $Sig_{pv_M}(ST, nptr1)$ to $S(ID_{M_i})$, and further broadcasts $nptr1$ to ID_B to provide awareness on ST – with the support of multidirectional communication. Note that: (I) Since all the master energy nodes have adequate computational and storage resources, we introduce $Sig(.)$ in SSC for only the master energy nodes to support data authenticity without impacting the performance of UReum (see Section IX). (II) We use $Hash(.)$ as the only cryptographic algorithm in ST to prevent data modification since all data are disclosed and available to all nodes in UReum.

2) Data-reply Smart Contract

A data-reply smart contract $DRSC$ is executed by a registered energy node (say, ID_Y) in response to a received DRT from another registered energy node. We use $MAC_k(.)$, and $Enc_k(.)$ as the cryptographic algorithms of $DRSC$, where k is a shared secret session key between two nodes.

Let ID_X and ID_Y be registered basic energy nodes with a shared secret session key k_4. Let $DRT = (MAC_{k_4}(Enc_{k_4}(emsg)), Enc_{k_4}(emsg))$ be a data-request transaction, where $emsg$ is an energy grid message of 64 bits. Upon receiving DRT from ID_X, ID_Y verifies that $VMAC_{k_4}(MAC_{k_4}(Enc_{k_4}(data))) = 1$? and computes $Dec_{k_4}Enc_{k_4}(data) = emsg, ID_X, X, ID_Y$, where $VMAC(.)$ is the verification algorithm of $MAC(.)$, $Dec(.)$ is the decryption part of the 128 bits AES algorithm, prepares a response $rmsg$ to $emsg$, and inserts $rmsg$, $emsg$, ID_Y, $S(ID_{M_i})$, $S(ID_{B_i})$, and ID_X into $DRSC$, where $S(ID_{M_i})$ is a set of all registered master energy nodes and $S(ID_{B_i})$ is a set of all registered basic energy nodes. $DRSC$ is available in the UReum blockchain and it takes into account the DRT format above and executes as follows: $DRSC$ computes a message $msg = Enc_k(rmsg, emsg, ID_Y, ID_X)$ and sends it to ID_X, prepares a store transaction $ST = (Hash(msg), msg)$, generates a notification pointer $nptr2$ that points to ST, stores ST and $nptr2$ in the UReum blockchain, and broadcasts $(ST, nptr_2)$ and $nptr_2$ to $S(ID_{M_i})$ and $S(ID_{B_i})$, respectively, to provide awareness on ST. Note that: (I) $Hash(.)$ is used for reaching consensus amongst the master energy nodes since it prevents unauthorised changes to data. (II) $nptr2$ is used by the energy mode to verify ST in the UReum blockchain.

3) Revoke Smart Contract

A revoke smart contract $RVSC$ is executed by a registered master energy node ID_{M_1} upon receiving an RVT from another registered master energy node ID_{M_2}. We use $Hash(.)$, and $VSig_{pb_1}(.)$ as the cryptographic algorithms of $RVSC$, where pb_1 is a public key of ID_{M_1}.

Let ID_{M_1} and ID_{M_2} be registered master energy nodes with a shared secret session key k_4. Let $RVT = (Sig_{pv_1}(t_0, X, ID_X)$ be a revoke transaction, where t_0 is a violated transaction of an existing transaction t (say of 264 bits), ID_X is an identity of a registered basic energy node that violates t, and pv_1 is a private key of ID_{M_1}. Let be a revoke confirmation message $rmsg = (t, ID_X, pb_1, RVT_V)$ and $RVT_V =$

$Hash(k_4, Sig_{pv_1}(t_0, X, ID_X))$ is a verifier for RVT. Upon receiving RVT and $rmsg$ from ID_{M_1}, ID_{M_2} checks whether (ID_X, X) and t exist in the UReum blockchain. If the checks succeed, it computes $RVT_{V2} = Hash(k_4, Sig_{pv_1}(t_0, X, ID_X))$ and further checks whether $RVT_{V2} = RVT_V$. If the RVT_{V2} check fails, it returns failed to ID_{M_1}. Otherwise it inserts RVT, t, t_0, ID_X, $S(ID_{M_i})$, and pb_1 and pv_1 into $RVSC$, which takes into account RVT format above and executes as follows: $RVSC$ verifies whether $t_0 = t$ and $VSig_{pb_1}(RVT)$ is valid. If the verifications succeed, it computes $ID_X(revoked) = Sig_{pv_1}(Hash(ID_X))$ to revoke ID_X, prepares a store transaction $ST = (Hash(RVT, ID_X(revoked)), (RVT, ID_X(revoked)))$, generates a notification pointer $nptr3$ that points to ST, stores ST and $nptr3$ in ID_{M_1}'s UReum blockchain, and broadcasts $(ST, nptr3)$ to $S(ID_{M_i})$ to provide awareness on ST. Note that if the execution of $RVSC$ succeeds at ID_{M_1}, the executions of $RVSC$ at other master energy nodes in ID_{M_i} also succeed.

Remarks: (I) A data-reply transaction is not required since the $DRSC$ automatically responds to the nodes for enhanced data awareness based on the requested data. (II) Master energy nodes are the only nodes that can execute SSC and $RVSC$ to ensure that all transactions are stored in the UReum blockchain and the transactions/identities can be revoked, respectively. (III) Data awareness is provided via the smart contracts, i.e., SSC, $DRSC$, and $RVSC$.

V. IDEAL FUNCTIONALITY FOR ESSENTIAL CRYPTOGRAPHIC OPERATIONS IN UREUM

In this section, we present our ideal functionality F_{CR} that supports our P-ECDH, which provides privacy during data awareness between two registered energy nodes. In P-ECDH, every registered energy node verifies the name, identity, and attributes of another registered energy node without the support and knowledge of any other energy node before deriving a P-ECDH key to prevent privacy attacks. A data-aware protocol P can use F_{CR} for energy node registration and data-aware cryptographic operations that support secure and privacy-preserving data awareness (see below). We argue that a P using F_{CR} realizes some ideal data-aware functionality F, i.e., $P|F_{CR} \leq_R F$. Using the composition theorems of the IITM model (see Section II), we can replace F_{CR} with its realization P_{CR} after $P|F_{CR} \leq_R F$ has been proven, where all ideal operations by F_{CR} are replaced by real counterparts. F_{CR} guarantees that only the registered owners of a P-ECDH key can have access to a shared secret session key derived from the P-ECDH key for securing data awareness (see below for more details). F_{CR} allows all energy nodes to perform the following cryptographic operations: i) generate a random value; ii) generate a private key; iii) verify a cryptographic identity; iv) compute a cryptographic commitment; v) accept a computed cryptographic identity; vi) compute an ECDH key and a preshared key; vii) select a random challenge; viii) encrypt and decrypt messages; and ix) MAC and VMAC messages. Furthermore, it allows only the master energy nodes to perform the following additional cryptographic operations: i) compute a cryptographic identity; and ii) compute P-ECDH and shared secret session keys.

Formally, F_{CR} is a machine with n I/O tapes and a network tape. The I/O tapes represent different roles in a protocol and the network tape is used for communicating with the adversary. In every protocol run, there is always one instance of F_{CR} which handles all requests for data-aware cryptographic operations. An unregistered energy node using F_{CR} is recognised by a tuple $(name, sid, r)$, where $name$ is the name of the energy node, sid is a session identifier, and r is the role/tape that connects the node to F_{CR}, $name$, and sid, which is always selected and managed by the protocol. On the other hand, a registered energy node using F_{CR} is recognised by a tuple (ID, sid, r), where ID is the energy node identity and r connects the energy node to F_{CR}, ID, and sid. Note that sid is always selected and managed by the protocol. To ensure that F_{CR} can identify the node who sent/receives a message, all messages on I/O tapes from unregistered and registered energy nodes are prefixed with $(name, sid)$ and (ID, sid), respectively. We use restricting messages (cf. Section III-D) to assure that an adversary responds to F_{CR}'s requests and cannot interfere with any run of F_{CR} and all operations of F_{CR} always succeed.

In F_{CR}, we model secret keys such that energy nodes can only access (or get their hands on) pointers to such secret keys. Then, the energy nodes can use the pointers to perform several cryptographic operations that support data awareness. F_{CR} maintains the actual values of the secret keys in the UReum blockchain. Without loss of generality, data added to the UReum blockchain of a master energy node by F_{CR} are replicated across the respective UReum blockchain of all other master energy nodes using SSC, which is executed by the master energy node (see Section IV-E for a description of the store smart contract). Additionally, F_{CR} also maintains the following in the UReum blockchain: i) a set $Names$ of energy nodes' names; ii) a set $RandValues$ of random values; iii) a set $RNIDs$ of random numbers and names/identities; iv) a set $Private$ of private keys; v) sets $BlockedPbs$ and $BlockedCCs$ of public keys and cryptographic commitments, respectively, which cannot be generated again; vi) a set $CCset$ of cryptographic commitments; vii) a set NCC of names of energy nodes and cryptographic commitments; viii) sets BID and MID of basic energy nodes identities and master energy nodes identities, respectively; ix) a set $IDCC$ of energy nodes identities, cryptographic commitments, and public key; x) a set $BlockedPEs$ of P-ECDH values that cannot be computed again and their associated public keys; xi) a set $Rand$ of random challenges; xii) a set $RandPEs$ of random challenges and cryptographic commitments; and xiii) sets $PresharedKeys$, $P-ECDHKeys$, and $SKeys$ of preshared keys, P-ECDH keys, and shared secret session keys, respectively. Note that F_{CR} uses the UReum blockchain to prevent guessing and collisions of keys, values, and identities (see below). Any key, value, or identity that cannot be verified by F_{CR} is treated as an unidentified key, value, or identity, respectively. The unidentified data are also maintained by F_{CR} in the UReum

blockchain. Furthermore, F_{CR} has access to all data such as transactions and smart contracts stored in the UReum blockchain.

We parameterize F_{CR} with a $Gen(1^{\eta})$ algorithm that is used to generate the elliptic curve domain parameters. $Gen(1^{\eta})$ takes a security parameter η as input and outputs (p, ai, bi, G, H, n, h). F_{CR} executes $Gen(1^{\eta})$ and stores the generated parameters (p, ai, bi, G, H, n, h) in the UReum blockchain upon its initialisation for the first time. Then F_{CR} sends (p, ai, bi, G, H, n, h) and a request for cryptographic algorithms to the adversary via a restricting message. Upon the completion of this initialisation, if a response is received on the I/O tape, F_{CR} continues to process the original message. Otherwise, if the response is received on a network tape, F_{CR} returns control to the adversary. The commands provided by F_{CR} on the I/O interface are listed in Tables 2 and 3.

Note that ‘‘fairness" is supported in F_{CR} via the verification of available data like random values, random challenges, or public keys to detect any malicious behaviour by any node.

We construct a realization P_{CR} of F_{CR} which uses well-known cryptographic schemes to implements all operations of F_{CR}. Formally, P_{CR} is a machine with the same I/O and network interfaces as F_{CR}. It is parameterized with $Gen(1^{\eta})$ algorithm with the same properties as for F_{CR}, three encryption schemes $\sum_{authenc}$, $\sum_{unauthenc}$, and $\sum_{pubenc}$ for authenticated symmetric encryption, unauthenticated symmetric encryption, and public key encryption, respectively, a MAC scheme $\sum_{MAC}$, a signature scheme $\sum_{Sig}$ be a signature scheme, a cryptographic hash function $hf = Hash(.)$, and three families of Pseudorandom Functions (PRFs) $F^{'} = (F^{'}_{\eta})_{\eta \in N}$, $F^{''} = (F^{''}_{\eta})_{\eta \in N}$, and $F^{'''} = (F^{'''}_{\eta})_{\eta \in N}$ that take key(s) and salt as input, and output a key (see [29] for formal definitions of these cryptographic primitives). Upon the activation of P_{CR} for the first time by some message msg, P_{CR} activates itself by executing the $Gen(1^{\eta})$ algorithm and storing results of the execution before processing msg. Like F_{CR}, P_{CR} maintains keys, values, and identities in the UReum blockchain and use them in deciding the type of cryptographic primitive to execute. For example, $F^{'}$ is used for deriving preshared keys from ECDH keys, $F^{''}$ is used for deriving P-ECDH keys from ECDH keys, and $F^{'''}$ is used for deriving shared secret session keys from P-ECDH keys. However, P_{CR} does not maintain any set of F_{CR}. The implementation of the F_{CR}'s commands in P_{CR} is provided in Table 4.

Using standard cryptographic assumptions, we state and prove that $P_{CR} \leq_R F_{CR}$. We use the Decisional Diffie-Hellman (DDH) assumption [30], [31] to prove that $P_{CR} \leq_R F_{CR}$ such that the simulator in this proof provides k_3 for uncorrupted P-ECDH key and $k_3^{'}$ for unvalidated (or corrupted) P-ECDH key. We restrict the environment in this proof not to cause a commitment problem [32] and produce key cycles [33], [34] in order to capture the expected use of P_{CR}/F_{CR} and say that the environment is used-order respecting and well-behaved. Furthermore, we introduce a machine M^* that is located between the environment and I/O interface of P_{CR}/F_{CR} to ensure that the restrictions and conditions are satisfied by the environment. If at any point one of the conditions is violated, M^* stops and blocks all current and future communications. We now obtain the following theorem:

Theorem 2. *Let Gen be an algorithm as above, $\sum_{authenc}$, $\sum_{unauthenc}$, $\sum_{pub}$ be encryption schemes, $\sum_{Sig}$ be a signature scheme, $\sum_{MAC}$ be a MAC scheme, $F^{'}$ and $F^{''}$ be families of pseudorandom functions for Gen, $F^{'''}$ be a family of pseudorandom functions, and hf be a cryptographic hash function. Let P_{CR} be parameterized with these algorithms. Let F_{CR} be parameterized with Gen. Then, the following holds true:*

$$M^*|P_{CR} \leq_R M^*|F_{CR}$$

if *Gen* always outputs groups with $n \geq 2$ to fulfil the DDH assumption, which holds true for *Gen*, $\sum_{unauthenc}$ and $\sum_{pub}$ are IND-CCA2 secure, $\sum_{authenc}$ is IND-CPA and INT-CTXT secure, and $\sum_{Sig}$ and $\sum_{MAC}$ are UF-CMA secure.

Proof Sketch. This proof consists of several hybrid systems where we replace the ideal protocol is replaced with several parts of its realization and no environment can distinguish the replacements. The hybrid systems are as follows: In step one, we define a hybrid system P^1_{CR} where all asymmetric operations and random challenge selection are replaced with their ideal versions. In step two, we define a hybrid system P^2_{CR} where the handling of private keys is replaced with its ideal version. In step three, we define a hybrid system P^3_{CR} where generations of P-ECDH keys are replaced with their ideal versions, however, guessing or collision of keys are not prevented (i.e., the simulator in this proof provides the P-ECDH keys). In step four, we define a hybrid system P^4_{CR} where real symmetric operations and key derivations are replaced with their ideal versions and the guessing and collision of keys are prevented. In step five, we replace MACs with their ideal versions. In the final step, we combine the results from steps 1 to 5 and use the [14]'s Lemma 4.4, which says that two environments are indistinguishable for all responsive environments, to conclude that for all responsive environments $E \in Env_R(M^*|P_{CR})$, $E|M^*|P_{CR} \equiv E|S|M^*|F_{CR}$ such that S is a responsive simulator that immediately responds with an expected answer.

VI. IDEAL FUNCTIONALITY FOR SECURE AND PRIVACY-PRESERVING DATA AWARENESS

In this section, we present our ideal functionality for secure and privacy-preserving data awareness, denoted by F^{RP}_{DA}. The functionality F^{RP}_{DA} is used by our *RPro* and *DAPro* (see below). The main idea of F^{RP}_{DA} is that energy nodes can request identities, exchange shared secret session keys, and further use the keys to execute UReum smart contracts for secure and privacy-preserving data awareness. F^{RP}_{DA} allows the modelling of validity and expiration of data awareness sessions (via the session keys and smart contracts), which are important to the Energy Internet for enhanced security and privacy.

Formally, F^{RP}_{DA} is a machine with two I/O tapes, network tape, and additional two tapes that connect connects

TABLE 2. The commands of the ideal functionality F_{CR}

Cryptographic Commands
Get generated elliptic curve domain parameters [(GetECDP)]. An energy node (say, (X, sid, r)) can request the elliptic curve domain parameters (p, ai, bi, G, H, n, h) that were generated by F_{CR} during initialisation. Upon receiving this request, F_{CR} sends (ECDP, (p, ai, bi, G, H, n, h)) to the node.
Generate a random value [(GenRV, X)]. An energy node (say, (X, sid, r)) can request F_{CR} to generate a random value rv_X. Upon receiving this request, F_{CR} checks whether X exists in the UReum blockchain (i.e., $X \in Names$) and returns (GenRV, *Restricted*) to the node if the check fails. Otherwise, F_{CR} forwards the request to the adversary via a restricting message to provide $rv_X \in \{1, ..., nn-1\}$, where nn is a large random selected integer in the UReum blockchain. F_{CR} checks whether rv_X is new (i.e., $rv_X \notin RandValues$). If this check fails, F_{CR} asks the adversary again for another rv_X until the check succeeds. Then, F_{CR} adds rv_X to *RandValues*, adds (rv_X, X) to *RNIDs*, and returns (RV, rv_X) to the node.
Generate a private key [(GenPV, X)]. An energy node (say, (X, sid, r)) can request F_{CR} to generate a private key pv_X. Upon receiving this request, F_{CR} first checks whether X exists in the UReum blockchain (i.e., $X \in Names$) and returns (GenPV, *Restricted*) to the node if the check fails. Otherwise, it forwards the request to the adversary via a restricting message to provide $pv_X \in \{1, ..., n-1\}$. Then, F_{CR} checks whether pv_X is new (i.e., $pv_X \notin Private$) and a public key $pb_X = pv_X.G$ has not been blocked from being generated (i.e., $pv_X \notin BlockedPbs$). If any of the checks fails, F_{CR} asks the adversary again for another pv_X until the checks succeed. Then, F_{CR} adds pv_X to *Private*, add pb_X to *BlockedPbs*, stores a pointer ptr_1 that points to pv_X for the node, and returns (PVPointer, ptr_1, pb_X) to the node.
Sign a message [(Sig, ptr_1, α)]. An energy node (say, (X, sid, r)) can request F_{CR} to sign a message α using its private key pv_X that the pointer ptr_1 points to. Upon receiving this request, F_{CR} verifies that ptr_1 belongs to the node. If the verification is successful, F_{CR} signs α using the digital signature part $Sig(.)$ of the ECDSA provided by the adversary (i.e., $Sig_{pv_X}(\alpha) = \beta$), and the resulting signature β is computed. Then, F_{CR} stores α for X (for subsequent verification) and returns β to the node.
Verify a signature [(VSig, pb_X, α, β)]. An energy node (say, (X, sid, r)) can request F_{CR} to verify a signature β for some message α using its public key pb_X. Upon receiving this request, F_{CR} verifies that pb_X is recorded for the node. If the verification succeeds, F_{CR} further verifies β using the verification part $VSig(.)$ of the $Sig(.)$'s ECDSA provided by the adversary (i.e., $VSig_{pb_X}(\beta) = \alpha$) and returns α to the node.
Generate a cryptographic hash of a message [(GenHash, α)]. An energy node (say, (X, sid, r)) can request F_{CR} to generate a cryptographic hash of a message. Upon receiving this request, F_{CR} hashes α using the SHA-256 algorithm $Hash(.)$ provided by the adversary (i.e., $Hash(\alpha) = \phi$) and returns (Hash, ϕ, α) to the node. *We note that a pointer ptr_2 to a key k_2 could also be included as a part of the request (say, $(Hash, ptr_2, \alpha)$). In this case, F_{CR} will compute $Hash(k_2, \alpha) = \phi$ using the same algorithm after the successful verification of ptr_2 and returns $(Hash, (ptr_2, \alpha), \phi)$ to the node.*
Compute a cryptographic commitment [(ComCC, X, pb_X, pb_{X2})]. An energy node (say, (X, sid, r)) can request F_{CR} to compute a cryptographic commitment CC_X from two public keys pb_X and pb_{X2}. Upon receiving this request, F_{CR} checks whether $X \in Names$, $pb_X \in BlockedPbs$, and $pb_{X2} \in BlockedPbs$, and returns (ComCC, *restricted*) to the node if any of these checks fail. Otherwise, it forwards the request to the adversary via a restricting message to provide a cryptographic commitment $CC_X = pb_X + pb_{X2}$. F_{CR} ensures that CC_X is new (i.e., $CC_X \notin BlockedCCs$) and it is computed as $pb_X + pb_{X2}$. Then, F_{CR} adds CC_X to *CCset*, adds (X, CC_X) to *NCC*, adds CC_X to *BlockedCCs*, and returns (CC, CC_X) to the node.
Compute a cryptographic identity [(ComID, X, CC_X, ptr_M)]. A registered master energy node (say, (ID_M, sid, r)) can request F_{CR} to compute a cryptographic identity ID_X from the basic energy node name X, a cryptographic commitment CC_X, and a private key pv_M to which the pointer ptr_M points. Upon receiving this request, F_{CR} checks whether (i) X is a genuine basic energy node, i.e., $X \in Names$, (ii) CC_X belongs to X, i.e., $(X, CC_X) \in NCC$, (iii) ID_M is a registered master energy node, i.e., $ID_M \in MID$, and (iv) ID_M has not been revoked, i.e., $ID_M(revoked)$ does not exist in the UReum blockchain, and returns (ComID, *restricted*) to the node if any of these checks fail. Otherwise, it forwards the request to the adversary via a restricting message to provide $ID_X = Hash(Sig_{pv_M}(X, CC_X))$ using cryptographic algorithms $Hash(.)$ and $Sig(.)$ provided by the adversary. F_{CR} ensures that ID_X is new and it is derived as $Hash(Sig_{pv_M}(X, CC_X))$. Then, F_{CR} adds ID_X to *BID*, adds (X, CC_X, pb_M) to *IDCC*, and returns (ID, ID_X) to the node. The ComID command supports "anonymity" by computing ID_X.
Verify a cryptographic identity [(VerID, ID_X, pb_M, X, CC_X)]. An energy node (say, (Y, sid, r)) can request F_{CR} to verify a cryptographic identity ID_X. Upon receiving this request, F_{CR} first checks whether pb_M is a valid public key (i.e., $pb_M \notin BlockedPbs$) and returns (AccID, *restricted*) to the node if the check fails. Otherwise, it forwards the request to the adversary via a restricting message to compute $VSig_{pb_M}(ID_X)$ using the cryptographic algorithms $Hash(.)$ and $VSig(.)$ provided by the adversary. F_{CR} ensures that $VSig_{pb_M}(ID_X) = Hash(X, CC_X)$. Then, F_{CR} verifies that ID_X exists and $ID_X(revoked)$ does not exist in the UReum blockchain and returns (VerID, *restricted*) to the energy node if any of the verifications fail. Otherwise, it returns *Okay* to the node. The VerID command supports "assurance" by verifying ID_X.
Validate and accept a cryptographic identity [(AccID, ID_X, pb_M, X, CC_X)]. An energy node (say, (X, sid, r)) can request F_{CR} to validate and accept ID_X as its identity. To validate and accept ID_X, F_{CR} first checks $pb_M \notin BlockedPbs$ and returns (AccID, *restricted*) to the node if the check fails. Otherwise, it forwards the request to the adversary via a restricting message to compute $VSig_{pb_M}(ID_X)$. F_{CR} ensures that $VSig_{pb_M}(ID_X) = Hash(X, CC_X)$. Then, F_{CR} adds ID_X to *BID* and returns *Okay* to the node.
Compute a P-ECDH value [(ComPE, CC_X, ID_X, ID_Y)]. A registered energy node (say, a registered basic energy node (ID_X, sid, r)) can request F_{CR} to compute a P-ECDH value pe_X from its cryptographic commitment CC_X and identity ID_X, and an identity ID_Y of another registered energy node. Upon receiving this request, F_{CR} first checks whether $CC_X \in BlockedCCs$, $ID_X \in BID$, $ID_Y \in BID$, and $ID_Y(revoked)$ and $ID_X(revoked)$ do not exist in the UReum blockchain, and returns (ComPE, *restricted*) to the node if any of these checks fail. Otherwise, it forwards the request to the adversary via a restricting message to provide $pe_X = Hash(CC_X.ID_X.ID_Y)$. F_{CR} ensures that $pe_X = Hash(CC_X, ID_X, ID_Y)$. Then, F_{CR} adds (pe_X, CC_X) to *BlockedPEs*, and returns (PE, pe_X) to the node.
Verify a P-ECDH value [(VerPE, pe_X, pb_X, ID_X, ID_Y)]. A registered energy node (say, a registered basic energy node (ID_Y, sid, r)) can request F_{CR} to verify a P-ECDH value pe_X from its identity ID_Y and a cryptographic commitment CC_X and another identity ID_X. Upon receiving this request, F_{CR} checks whether $(pe_X, CC_X) \in BlockedPEs$, $ID_X \in BID$, ID_Y exists, and $ID_X(revoked)$ and $ID_Y(revoked)$ do not exist in the UReum blockchain, and returns (VerPE, *restricted*) to the node if any of these checks fail. Otherwise, it returns *Okay* to the node.

TABLE 3. The remaining commands of the ideal functionality F_{CR}

Cryptographic Commands
Generate a random challenge $[(\text{GenRC}, pe_X)]$. A registered energy node (say, a registered basic energy node (ID_Y, sid, r)) can request F_{CR} to generate random challenge e_Y. Upon receiving this request, F_{CR} checks whether $pe_X \in BlockedPbs$ and pe_X does not have a selected random challenge (i.e., $(e_Y, pe_X) \notin RandPEs$), and returns $(\text{GenRC}, restricted)$ to the node if any of the checks fail. Otherwise, it forwards the request to the adversary via a restricting message to provide a random challenge $e_Y \in Z_p$. F_{CR} ensures that e_Y is new (i.e., $e_Y \notin Rand$). Then, F_{CR} adds e_Y to $Rand$, adds (e_Y, pe_X) to $RandPEs$ and returns (RC, e_Y) to the node.
Compute a preshared key $[(\text{ComPS}, ptr_1, pb_M, rv_M, ID_{M_1})]$. An energy node (say, (X, sid, r)) that is acting as an initiator during energy node registration can request F_{CR} to compute a preshared key k_2 of type ty_1 from its private key pv_X to which the pointer ptr_1 points, a public key pb_M, and random value rv_M of a registered master energy node ID_{M_1}. Upon receiving this request, F_{CR} checks whether $pb_M \in BlockedPbs$, $rv_X \in RandValues$, and $(rv_M, ID_{M_1}) \in RNIDs$. If the checks succeed, it further checks whether a preshared key k_1 has already been computed from pb_M/pb_X and pv_X that ptr_1 points to. If this check succeeds, it returns the pointer ptr_2 pointing to this same key k_2 to the node. Otherwise, it asks the adversary via a restricting message to provide $k_2 = F^{'}(k_1.rv_M.mod\ p)$, where $k_1 = pv_X.pb_M$ is an ECDH key, p is the large prime (in the domain parameters), and $F^{'}$ is a PRF. F_{CR} ensures that k_2 is new (i.e., $k_2 \notin PresharedKeys$) and is derived as $F^{'}(k_1.rv_M.mod\ p)$. Then, F_{CR} adds k_2 to $PresharedKeys$, stores a new pointer ptr_2 pointing to k_2 for the owners of pb_M/pb_X, and returns (ComPS, ptr_2) to the node. The ComPS command supports "correctness" and "unlinkability" by computing k_3.
Encrypt using a secret key (say, a preshared key) $[(\text{Enc}, ptr_2, msg)]$. An energy node (say, (X, sid, r)) can request F_{CR} to encrypt a message msg using a preshared key k_2 to which the pointer ptr_2 points. Upon receiving this request, F_{CR} verifies that ptr_2 belongs to the node. If the verification succeeds, F_{CR} encrypts msg using the encryption part $Enc(.)$ of the advanced encryption algorithm provided by the adversary and the resulting ciphertext $cmsg = Enc_{k_2}(msg)$ of msg is computed. Then, F_{CR} checks whether the decryption of $cmsg$ under k_2 yields msg. If the check succeeds, F_CR stores $(msg, cmsg)$ for k_2 in the UReum blockchain and returns $cmsg$ to the node. The Enc command supports "confidentiality" by computing $Enc_{k_2}(msg)$.
Decrypt using a secret key (say, a P-ECDH key) $[(\text{Dec}, ptr_2, cmsg)]$. An energy node (say, (X, sid, r)) can request F_{CR} to decrypt a ciphertext $cmsg$ using a preshared key k_2 to which the pointer ptr_2 points to. Upon receiving this request, F_{CR} first verifies whether ptr_2 belongs to the node and if there exists $cmsg$ such that $(msg, cmsg)$ is stored for k_2 in the UReum blockchain. Then, it uses the decryption part $Dec(.)$ of the advanced encryption algorithm provided by the adversary to compute $Dec_{k_2}(cmsg) = msg$. If the verifications and computation succeed, F_{CR} returns msg to the node.
MAC using a secret key (say, a P-ECDH key) $[(\text{Mac}, ptr_2, \alpha)]$. An energy node (say, (X, sid, r)) can request F_{CR} to MAC a message α using a preshared key k_2 to which the pointer ptr_2 points to. Upon receiving this request, F_{CR} verifies that ptr_2 belongs to the node. If the verification succeeds, F_{CR} computes a MAC of α using the MAC algorithm $MAC(.)$ provided by the adversary (i.e., $MAC_{k_2}(\alpha) = \beta$), and the resulting MAC β is computed. Then, F_{CR} stores α for k_2 (for subsequent verifications) and returns β to the node. The MAC command supports "correctness" by computing $MAC_{k_2}(\alpha)$.
VMAC using a secret key (say, a P-ECDH key) $[(\text{VMac}, ptr_2, \alpha, \beta)]$. An energy node (say, (X, sid, r)) can request F_{CR} to verify a MAC β for some message α using a preshared key k_2 to which the pointer ptr_2 points to. Upon receiving this request, F_{CR} first verifies whether ptr_2 belongs to the node. If the verification succeeds, F_{CR} verifies β using the MAC verification algorithm $VMAC(.)$ provided by the adversary (i.e., $VMAC_{k_2}(\alpha, \beta) = 1$). Then, F_{CR} returns $Okay$ to the node.
Compute a P-ECDH key $[(\text{ComPK}, ptr_1, pb_X 2, e_Y, pe_X, CC_X)]$. A registered energy node (say, (ID_Y, sid, r)) acting as a responder during data awareness can request F_{CR} to compute a P-ECDH key k_3 of type ty_2 from the private key pv_{Y2} to which the pointer ptr_1 points to, public key pb_{X2} of some registered energy node ID_X, random challenge e_Y, P-ECDH value pe_X, and cryptographic commitment CC_X. Upon receiving this request, F_{CR} first checks whether pb_{x2} is a valid public key (i.e., $pb_{x2} \in BlockedPbs$), e_Y is a valid random challenge (i.e., $e_Y \in Rand$), pe_X is a valid P-ECDH value and computed via CC_X (i.e., $(pe_X, CC_X) \in BlockedPEs$), and CC_X is a valid cryptographic commitment (i.e., $CC_X \in BlockedCCs$), and returns $(\text{ComPK}, restricted)$ to the node if any of these checks fails. Then, F_{CR} checks whether a P-ECDH key k_3 has already been computed using pv_Y, pb_{X2}, e_Y, pe_X, and CC_X, and returns the pointer ptr_3 pointing to the already computed key k_3. If k_3 does not exist, F_{CR} computes a P-ECDH key as follows: • If pb_{X2} can be validated (i.e., $pb_{X2} \in BlockedPbs$), F_{CR} forwards the request to the adversary via a restricting message $(\text{ComPK}, uncorrupted, pv_{Y2}, pb_{X2}, e_Y, pe_X, CC_X)$ to provide a new P-ECDH $k_3 = Hash(F^{''}(k_1.CC_X.e_Y.pe_X))$, where $k_1 = pv_{Y2}.pb_{X2}$ is an ECDH key and $F^{''}$ is another PRF. F_{CR} ensures that k_3 is new, (i.e., $k_3 \notin P - ECDHKeys$). Then, F_{CR} adds k_3 to $P - ECDHKeys$, stores a new pointer ptr_3 to k_3 for the owners of k_1/k_3, and returns $(\text{ComPKPointer}, ptr_3)$ to the node. The ComPK command supports "correctness" and "unlinkability" by computing k_3. • If pb_{X2} cannot be validated, F_{CR} forwards the request to the adversary via a restricting message $(\text{ComPK}, unvalidated, pv_{Y2}, pb_{X2}, e_Y, pe_X, CC_X)$ to provide a new P-ECDH key $k_3^{'}$. F_{CR} ensures that $k_3^{'}$ is new, (i.e., $k_3^{'} \notin Unvalidated$). Then, F_{CR} adds $k_3^{'}$ to $Unvalidated$, stores a new pointer ptr_3 to $k_3^{'}$ for the owners of $k_1/k_3^{'}$, and returns $(\text{ComPKPointer}, ptr_3)$ to the node.
Derive a shared secret session key $[(\text{DerSKey}, ptr_3, e_Y, pe_X, rv_Y, ID_Y)]$. A registered energy node (say, (ID_Y, sid, r)) can request F_{CR} to derive a shared secret session key k_4 of type ty_3 from the P-ECDH key k_3 to which the pointer ptr_3 points to, random challenge e_Y, P-ECDH value pe_X, random value rv_Y, and identity ID_Y. Upon receiving this request, F_{CR} first checks whether $e_Y \in Rand$, $(e_Y, pe_X) \in RandPEs$, rv_Y is a valid random value (i.e., $rv_Y \in RandValues$), and $(rv_Y, ID_Y) \in RNIDs$, and returns $(\text{DerSKey}, restricted)$ to the node if any of the checks fails. Otherwise, it checks whether a shared secret session key k_4 has already been derived using k_3,e_Y,pe_X,rv_Y, and ID_Y, and returns the pointer ptr_4 pointing to the same key k_4. If no k_4 already exists, F_{CR} forwards this request to the adversary via a restricting message to provide $k_4 = Hash(F^{'''}(k_3.e_Y.pe_X.rv_Y.ID_Y))$, where $F^{'''}$ is another PRF. F_{CR} ensures that k_4 is new and it is derived as $Hash(F^{'''}(k_3.e_Y.pe_X.rv_Y.ID_Y))$. Then, F_{CR} adds k_4 to $SKeys$, stores a new pointer ptr_4 pointing to k_4 for the owners of k_3, and returns $(\text{SKeyPointer}, ptr_4)$ to the node. The $DerSKey$ command also supports "correctness" and "unlinkability" by computing k_4. *Note that in a situation where the node ID_Y is an initiator during a data awareness, such a node will provide the random value and identity of its responder node to compute a new P-ECDH key. Thus, the random values and identities of energy nodes acting as responders during data awareness are used for computing P-ECDH keys.*

to F_{CR}. F_{DA}^{RP} uses F_{CR} as a subroutine for cryptographic operations. We parameterized F_{DA}^{RP} with a symmetric key type $ty \in \{preshared\ key, unauthenc\ key, authenc\ key, MAC\ key\}$ which helps to determine the key generated before performing

TABLE 4. The implementation of all F_{CR} commands in P_{CR}

Cryptographic Commands
Get generated elliptic curve domain parameters [(GetECDP)]. P_{CR} outputs the elliptic curve domain parameters (p, ai, bi, G, H, n, h) generated during its activation.
Generate a random value [(GetRV, X)]. P_{CR} checks whether X is a valid name of an energy node (i.e., exists in the UReum blockchain), selects a random value $rv_X \in \{1, ..., nn-1\}$, and outputs rv_X to the node.
Generate a private key [(GenPV, X)]. P_{CR} checks whether X is valid and selects a private key $pv_X \in \{1, ..., n-1\}$, computes a public key $pb_X = pv_X.G$, records a pointer ptr_1 to pv_X for the node, and returns (ptr_1, pb_X) to the node.
Sign a message [(Sig, ptr_1, α)]. P_{CR} signs a message α using pv_X (as $Sig_{pv_X}(\alpha) = \beta$) and the resulting signature β is returned to the node.
Verify a signature [(VSig, pb_X, α, β)]. P_{CR} verifies a signature β using pb_X (as $VSig_{pb_X}(\beta) = \alpha$) and returns the message α to the node if the verification succeeds.
Generate a cryptographic hash of a message [(Hash, α)]. P_{CR} generates $Hash(\alpha) = \phi$ and returns (ϕ, α) to the node.
Compute a cryptographic commitment [(ComCC, X, pb_X, pb_{X2})]. P_{CR} checks whether X and private keys pb_X and pb_{X2} are valid, and returns (ComCC, *restricted*) to the node if any of these checks fail. Otherwise, it computes $CC_X = pb_X + pb_{X2}$ and returns CC_X to the node.
Compute a cryptographic identity [(ComID, X, CC_X, ptr_M)]. P_{CR} checks whether X and an identity ID_M are valid and returns (ComID, *restricted*) to the node if any of these checks fails. Otherwise, it computes $ID_X = Sig_{pv_M}(Hash(X, CC_X))$, records ID_X in the UReum blockchain, and returns ID_X to the node.
Verify a cryptographic identity [(VerID, ID_X, pb_M, X, CC_X)]. P_{CR} computes $VSig_{pb_M}(ID_X) = Hash(X, CC_X)$. If the computation succeeds, it verifies whether ID_X exists and ID_X(*revoked*) does not exist in the UReum blockchain, and then return *Okay* to the node.
Validate and accept a cryptographic identity [(AccID, ID_X, pb_M, X, CC_X)]. P_{CR} checks whether pb_M is valid public key (i.e., $pb_M \in G$), computes $VSig_{pb_M}(ID_X) = Hash(X, CC_X)$, and returns *Okay* to the node if the check and computation succeed.
Compute a P-ECDH value [(ComPE, CC_X, ID_X, ID_Y)]. P_{CR} checks whether CC_X is a valid cryptographic commitment and ID_X and ID_Y are valid identities, and returns (ComPE, *restricted*) to the node if any of these checks fails. Otherwise, it computes $pe_X = Hash(CC_X, ID_X, ID_Y)$, records pe_X in the UReum blockchain, and returns pe_X to the node.
Verify a P-ECDH value [(VerPE, CC_X, pb_X, ID_X, ID_Y)]. P_{CR} checks whether CC_X and pb_X are valid cryptographic commitment and public key, respectively, and ID_X and ID_Y are valid identities, and returns (VerPE, *restricted*) to the node if any of these checks fail. Otherwise, it returns *Okay* to the node if the checks succeed.
Generate a random challenge [(GenRC, pe_X)]. P_{CR} checks whether pe_X is a valid P-ECDH value, selects $e_Y \in Z_p$, records e_Y and (e_Y, pe_X) in the UReum blockchain, and returns e_Y to the node if the check succeeds.
Compute a preshared key [(ComPS, $ptr_1, pb_M, rv_M, ID_{M_1}$)]. P_{CR} checks whether pb_M is a valid public key, rv_M is a valid random value, and (rv_M, ID_{M_1}) is a valid combination, and then computes the preshared key $k_2 = F'(k_1.rv_M.mod\ p)$ if the check succeeds, where $k_1 = pv_X.pb_M$ is an ECDH key and p is the large prime (in the domain parameters), and F' is a PRF. Then, it records a new pointer ptr_2 pointing to k_2 for the node, and returns ptr_2 to the node. Note that ptr_2 is recorded in the UReum blockchain.
Encrypt using a secret key (say, a preshared key) [(Enc, ptr_2, msg)]. P_{CR} encrypts msg using k_2 (as $Enc_{k_2}(msg) = cmsg$) and the resulting ciphertext $cmsg$ is returned to the node.
Decrypt using a secret key (say, a P-ECDH key) [(Dec, $ptr_2, cmsg$)]. P_{CR} decrypts cmsg using k_2 (as $Dec_{k_2}(cmsg) = msg$) and returns the message msg to the node if the decryption succeeds.
MAC using a secret key (say, a P-ECDH key) [(Mac, ptr_2, α)]. P_{CR} MACs α using k_2 (as $MAC_{k_2}(\alpha) = \beta$) and the resulting MAC β is returned to the node.
VMAC using a secret key (say, a P-ECDH key) [(VMac, ptr_2, α, β)]. P_{CR} verifies β using k_2 (as $VMAC_{k_2}(\alpha, \beta) = 1$) and returns *Okay* to the node if the verification succeeds.
Compute a P-ECDH key [(ComPK, $ptr_1, pb_{X2}, e_Y, pe_X, CC_X$)]. P_{CR} checks whether pb_{x2} is a valid public key (i.e., $pb_{X2} \in H$), e_Y is a valid random challenge, pe_X is a valid P-ECDH value, (pe_X, CC_X) is a valid combination, and CC_X is a valid cryptographic commitment, and returns (ComPK, *restricted*) to the node if any of these checks fails. Otherwise, it computes the new P-ECDH $k_3 = Hash(F''(k_1.CC_X.e_Y.pe_X))$, where $k_1 = pv_{Y2}.pb_{X2}$, records a new pointer ptr_3 to k_3 for the node, and returns ptr_3 to the node. Note that ptr_3 is recorded in the UReum blockchain.
Derive a shared secret session key [(DerSKey, $ptr_3, e_Y, pe_X, pe_Y, rv_Y, ID_Y$)]. P_{CR} checks whether e_Y is a valid random challenge and pe_X and pe_Y are valid P-ECDH values, rv_Y is a valid random value, and (rv_Y, ID_Y) is a valid combination, and returns (DerSKey, *restricted*) to the node if any of the checks fails. Otherwise, it derives the shared secret session key $k_4 = Hash(F'''(k_3.e_Y.pe_X.pe_Y.rv_Y.ID_Y))$, records a pointer ptr_4 to k_4 for the node, and returns ptr_4 to the node. Note that ptr_4 is recorded in the UReum blockchain.

data awareness. Similar to F_{CR}, an unregistered energy node using F_{DA}^{RP} for data awareness is identified by $(name, sid, r)$. Note that r can either be an initiator or responder of an energy node registration protocol as well as a data awareness protocol. Additionally, a registered energy node is identified by (ID, sid, r). In F_{DA}^{RP}, every message from/to every I/O tape is prefixed with $(name, sid)$ and (ID, sid) for unregistered energy node and registered energy node, respectively. F_{DA}^{RP} maintains several states such as *restricted*, *startedRPro*, *startedDAPro*, *closed*, and *corrupted* with the initial state

for every energy node is set to *restricted*. The operations provided by F_{DA}^{RP} to energy nodes for data awareness are as follows:

- An energy node, say (X, sid, r), with state *restricted* can start an energy node registration by sending an initialisation message $imsg = (\text{InitialiseRPro}, ID_M, m_1)$, where ID_M is the identity of a master energy node, which represents the intended responder, and m_1 is a random big string that might be used by RPro. Upon receiving this request, F_{DA}^{RP} uses the UReum blockchain to verify X and ID_M. If the verifications succeed, F_{DA}^{RP} sets $state(X, sid, r) = startedRPro$, sets $Responder(ID, sid, r) = ID_M$, and forwards a message $(imsg, (X, sid, r))$ to the adversary. In this case, the states of the energy nodes become *startedRPro*. We note that as *imsg* is not digitally signed since the node (X, sid, r) in the state *restricted* does not have access to any cryptographic operation, prefixing every message with (X, sid) plays an important role in preventing any other node from claiming to be X, or X claiming to be another node in the Energy Internet.
- An energy node, say (X, sid, r), with state *startedRPro* can use F_{DA}^{RP} to access the following commands of F_{CR} for registration/cryptographic identity request: i) GetECDP; ii) GenRV; iii) GenPV; iv) VSig; v) Hash; vi) ComCC; vii) VerID; viii) AccID; ix) VSig; x) ComPS; xi) Enc; xii) Dec; xiii) MAC; and xiv) VMAC. On the other hand, a master energy node (ID, sid, r) with state *startedRPro* can use F_{DA}^{RP} to access the following command of F_{CR} for computing the cryptographic identity: i) GetECDP; ii) GenRV; iii) GenPV; iv) Sig; v) VSig; vi) Hash; vii) ComCC; viii) ComID; ix) VerID; x) AccID; xi) VSig; xii) ComPS; xiii) Enc; xiv) Dec; xv) MAC; and xvi) VMAC. In each of the above cases, F_{DA}^{RP} forwards all responses received from F_{CR} to the associated node and keeps track of keys and pointers available to the nodes. Once a cryptographic identity ID_X is validated and accepted by the node (X, sid, r) via the execution of the command AccID, the node sends a close session message to F_{DA}^{RP}. In this case, F_{DA}^{RP} sets the state of the node as $state(X, sid, r) := closed$, notifies the adversary via a restricting message $(\text{CloseSesion}, (X, sid, r))$, and returns *Okay* to the node after receiving a response from the adversary. Thus, the node losses access to all its computed keys, pointers, and cryptographic operations.
- An energy node, say (ID_X, sid, r), with state *restricted* can start a data awareness by sending a data-aware message $dmsg = (\text{InitialiseDAPro}, ID_Y, m_2)$, where ID_Y is the identity of its data-aware responder and m_2 is a random bit string that might be used by the DAPro. F_{DA}^{RP} uses the UReum blockchain to verify ID_X and ID_Y upon receiving *dmsg*. If the verifications succeed, F_{DA}^{RP} sets $state(ID_X, sid, r) := startedDA$, sets $Responder(ID, sid, r) := ID_Y$, and forwards a message $(dmsg, (ID_X, sid, r))$ to the adversary. Then, the states of the energy nodes become *startedDA*.
- An energy node, say (ID_X, sid, r), with state *startedDA* can perform the following: (I) Use F_{DA}^{RP} to access the following commands of F_{CR} to support data awareness: i) GetECDP; ii) GenRV; iii) GenPV; iv) Hash; v) ComCC; vi) VerID; vii) ComPE; viii) VerPE; ix) GenRC; x) Enc; xi) Dec; xii) MAC; xiii) VMAC; xiv) ComPK; and xv) DerSKey. Note that F_{DA}^{RP} use the above commands of F_{CR} to generate any of the transactions *ST*, *DRT*, and *RVT*. (II) Execute *SSC*, *DRSC*, and *RVSC* (see Section IV-E for a detailed description of the smart contracts). Note that F_{DA}^{RP} forwards all responses it received from F_{CR} to the node and continue to keep track of the session keys, pointers, and cryptographic operations.
- An energy node, say (ID_X, sid, r), with state *startedDA* can request F_{DA}^{RP} to close her session. F_{DA}^{RP} forwards the request to the adversary upon receipt and then returns *Okay* to the node after receiving a response from the adversary. The state of the node becomes *closed* and thus the node losses access to all keys, pointers, and cryptographic operations.

In F_{DA}^{RP}, corruption is modelled in such a way that the adversary can only corrupt the energy nodes in the states *restricted* and *closed* (for example, $state(X, sid, r) \in restricted, closed$). If this is the case, F_{DA}^{RP} updates the energy node's state to *corrupted* after receiving and processing a corruption request (for example, $(\text{Corrupt}, (X, sid, r))$) from the adversary to corrupt the energy node (X, sid, r). Note that: (I) Since the energy nodes do not have access to any keys, pointers, or cryptographic operations in the states *restricted* and *closed*, this shows that F_{DA}^{RP} does not give the adversary access to any keys, pointers, or cryptographic operations thereby modelling perfect forward secrecy. (II) The adversary may request F_{DA}^{RP} to pair a corrupted energy node with an uncorrupted energy node if the energy nodes are not yet in a session. F_{DA}^{RP} checks whether the energy nodes are not yet in a session. If the check succeeds, F_{DA}^{RP} pairs the energy nodes accordingly. However, the corrupted energy node will not get access to any keys in F_{CR}. (III) An energy node can ask F_{DA}^{RP} for its corruption status. F_{DA}^{RP} returns the corruption status to the energy node. However, if the energy node is in the state *restricted*, F_{DA}^{RP} asks the adversary via a restricting message whether it wants to corrupt the energy node. If F_{DA}^{RP} receives *Yes* as a response, it updates the state of the energy node to *corrupted* and returns the corruption status to the energy node. Furthermore, the environment can ask F_{DA}^{RP} whether a public key, private key, secret key, or pointer of the energy code is recorded as corrupted for the energy node. (IV) We further restrict our corruption modelling to make F_{DA}^{RP} easier to use for data awareness. In this case, we can say that a session cannot be corrupted if it has returned a shared secret session key and the corruption status of instances in F_{DA}^{RP} remains unchanged during energy node registration or data awareness.

Remarks:

- The transactions *ST*, *DRT*, and *RVT* are generated as

follows: (I) ST: The energy node (ID_X, sid, r) with a message msg can request F_{CR} to generate the hash of msg. In this case, F_{CR} uses the Hash command to generate $Hash(msg)$. Upon the successful execution of the command, the node receives $(Hash(msg), msg)$, which represents ST (see Section IV-D for a detailed description of ST). (II) DRT: The energy node (ID_X, sid, r) with a message $emsg$ and shared secret session key k_4 can request F_{CR} to encrypt and MAC and message. In this case, F_{CR} uses the Enc and Mac commands to generate $Enc_{k_4}(emsg)$ and $MAC_{k_4}(Enc_{k_4}(emsg))$, respectively. Upon the successful execution of the commands, the node receives $(Enc_{k_4}(emsg), MAC_{k_4}(Enc_{k_4}(emsg)))$, which represents DRT (see Section IV-D for a detailed description of DRT). (III) RVT: The energy node (ID_X, sid, r) with a violated message $vmsg$, name of a node Y, identity of a node ID_Y, and private key pv_X can request F_{CR} to sign a set of data. In this case, F_{CR} uses the Sig command to generate $Sig_{pv_1}(vmsg, name, ID)$. Upon the successful execution of the commands, the node receives the generated data as $RVT = Sig_{pv_1}(vmsg, name, ID)$ (see Section IV-D for a detailed description of RVT).

- In the state *startedRPro*, F_{DA}^{RP} models anonymity, assurance, confidentiality, correctness, and unlinkability after the execution of the commands ComID, VerID, Enc/Dec, MAC/VMAC/ComPS, and ComPS, respectively.
- In the state *startedDAPro*, F_{DA}^{RP} models assurance, confidentiality, correctness, and unlinkability after the execution of the commands VerID, Enc/Dec, MAC/VMAC/ComPK/DerSKey, and ComPK/DerSkey, respectively.
- In the state *startedDAPro*, F_{DA}^{RP} also models fairness after the execution of SSC, $DRSC$, or $RVSC$.

VII. UREUM

In this section, we present our UReum, which consists of *RPro* and *DAPro*, for providing secure and privacy-preserving data awareness in the Energy Internet. *RPro* and *DAPro* use F_{CR} to perform all their cryptographic operations and realize F_{DA}^{RP}.

A. RPRO

We present our *RPro*, which is based on EC-PCS and cryptographic algorithms such as $Hash(.)$ and $Sig_{pv}(.)$ and is executed between an unregistered energy node X and a registered master energy node ID_{M_1} is depicted in Figure 7). To model the initiator X and responder ID_{M_1} of our protocol, we use two machines M_I and M_R for the initiator and responder, respectively. Each of these machines has a network tape and the same I/O interface as F_{DA}^{RP} and the machines use F_{CR} as a subroutine to perform all their cryptographic operations. In every run of *RPro*, there is one instance of M_I/M_R per energy node executing the protocol according to Figure 7. When ID_{M_1} receives a register transaction $RT = X, pb_X, pb_{X2}$, it

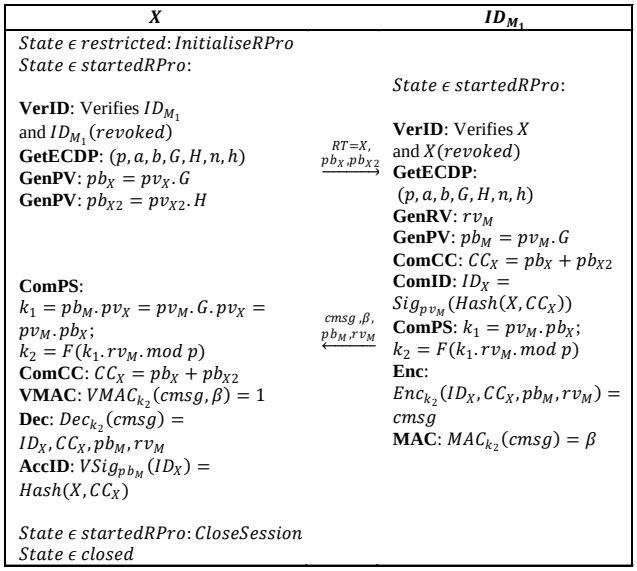

FIGURE 7. Registration Protocol (*RPro*).

leverages the properties of EC-PCS and ECDH to create a cryptographic commitment and generate a preshared key, respectively, to support the creation of a cryptographic identity, where pb_X and pb_{X2} are public keys of X. At the end of the protocol, the instance of ID_{M_1} computes a cryptographic identity $ID_X = Hash(Sig_{pv_M}(X, CC_X))$ for X while the instance of the initiator accepts ID_X upon the successful validation of $VSig_{pb_M}(ID_X) = X, CC_X$, where $CC_X = pb_X + pb_{X2}$. Note that private keys (such as pv_M, pv_X, and pv_{X2}) are successfully delivered by the respective instances, which further allow the energy nodes to use F_{CR} to perform cryptographic operations with the keys.

B. DAPRO

Our *DAPro* with security properties such as data correctness, identity assurance, and transaction fairness as well as privacy properties such as identity anonymity, data confidentiality, and transaction unlinkability is depicted in Figure 8. The protocol is executed between two registered energy nodes ID_X and ID_Y to derive a shared secret session key for creating a data awareness in the Energy Internet. More precisely, the shared secret session key is used by ID_X and ID_Y to support the execution of data awareness smart contracts between them and at the same time satisfy the security and privacy requirements of the Energy Internet. Upon the successful execution of any smart contracts, the energy nodes provide distributed insights into the activities of the Energy Internet to all energy nodes.

Similar to *RPro*, we model the initiator ID_X and responder ID_Y of *DAPro* using two machines M_I and M_R for the initiator and responder, respectively. Each of the machines has a network tape, uses F_{CR} as a subroutine for cryptographic operations, and provide the same I/O interface as F_{DA}^{RP}. In

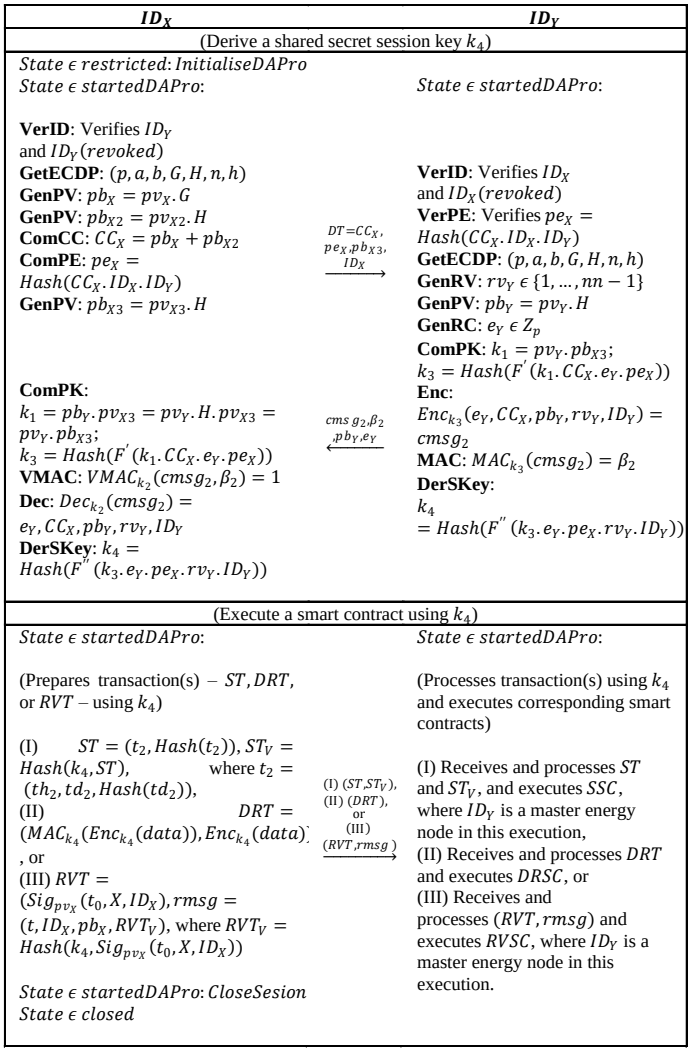

FIGURE 8. Data-aware protocol (*DAPro*).

the run of DAPro, there is only one instance of M_I/M_R for every energy node which executes *DAPro* from Figure 8. When ID_Y receives a data-aware initiation transaction $DT = CC_X, pe_X, pb_{X3}, ID_X$, it leverages the properties of EC-ZKPK to support derivation of a shared secret session key, where CC_X is a cryptographic commitment, *pe* is a P-ECDH value, and *pb* is a public key. In other words, the instances of ID_X and ID_Y derive the shared secret session key k_4. Upon the derivation of k_4, ID_X uses it (i.e., k_4) to prepare transaction(s) for their corresponding smart contracts' executions. If ID_Y receives $ST = (t_2, Hash(t_2))$ and $ST_V = Hash(k_4, ST)$ from ID_X, it executes *SSC* as described in Section IV-E, where $t_2 = (th_2, td_2, Hash(td_2))$ is a new transaction, th_2 is a transaction header of 32 bits, td_2 is transaction information, and $Hash(td_2)$ is a secure hash of td_2. ID_Y executes *DRSC* and *RVSC* if it receives $DRT = (MAC_{k_4}(Enc_{k_4}(emsg)), Enc_{k_4}(emsg))$ and $RVT = (Sig_{pv_X}(t_0, X, ID_W), rmsg = (t, ID_X, pb_X, RVT_V)$, respectively, from ID_X, where *emsg* is an energy grid message, t_0 is a violated transaction of an existing transaction t, ID_W is the identity of the energy node that violates t, pv_X is a private key of ID_X, pb_X is a public key of ID_X, and $RVT_V = Hash(k_4, Sig_{pv_X}(t_0, X, ID_X))$ is a verifier for RVT. Note that: (I) ID_Y is a master energy node in the execution of *SSC* or *RVT*. The description of *SSC* or *RVT* is provided in Section IV. (II) Every smart contract is executed to provide secure and privacy-preserving data awareness in the Energy Internet.

VIII. SECURITY ANALYSIS OF UREUM

In this section, we analyse UReum, i.e., *RPro* and *DAPro*, using our ideal functionalities F_{CR} and F_{DA}^{RP} and prove that *RPro* and *DAPro* are secure universally composable protocols. We use information-theoretic arguments for the proofs of these protocols due to the use of F_{CR}, which simplifies the proofs.

A. SECURITY ANALYSIS OF RPRO

We provide a theorem that states that *RPro* is a secure registration protocol. A cryptographic identity returned by *RPro* can be used in an ideal manner to support the execution of data awareness.

Theorem 3. *Let M_I and M_R be machines that model RPro as described above. Let F_{CR} and $F_{CR}^{'}$ be two versions of the ideal functionality for cryptographic operations with the same parameter and let F_{DA}^{RP} be the ideal functionality for security and privacy-preserving data-awareness with parameters ty_1 = preshared key. Then, the following holds true:*

$$M_I|M_R|F_CR \leq_R F_{DA}^{RP}|F_{CR}^{'}$$

Proof Sketch. An energy node X is corrupted if its preshared key is corrupted. Furthermore, an instance is corrupted if it returns *Yes* when asked for its corruption status by the environment. We define a simulator S and show that $E|M_I|M_R|F_{CR} \equiv E|S|F_{DA}^{RP}|F_{CR}^{'}$ for all responsive environments $E \in Env(M_I|M_R|F_{CR})$. S fulfils the runtime conditions and immediately responds to restricting messages as long as E does the same with overwhelming probability. $M_I|M_R|F_{CR}$ is simulated by S, which keeps track of corruption statuses of instances belonging to the energy nodes in F_{DA}^{RP} and synchronises the simulated instances of M_I/M_R. S initialises F_{CR} by sending a message to $F_{CR}^{'}$ and receiving Okay for simulating F_{CR}. Then, S requests for the cryptographic algorithms associated with F_{CR} from the environment and forwards the received algorithms to $F_{CR}^{'}$. If F_{DA}^{RP} specifies that an energy node $(name, sid, r)$ has started an energy node registration, S updates its internal simulation accordingly. If a cryptographic commitment CC_X, public key pb_M, and random value rv_M are accepted by an uncorrupted instance (X, sid_X, r_X), S instructs F_{DA}^{RP} to create an energy node registration session from (X, sid_X, r_X) and instance $(ID_{M_1}, sid_{M_1}, r_{M_1})$ that computed the signature and MAC in the second message of the *RPro*. Then, $F_{CR}^{'}$ of F_{DA}^{RP} requests S to provide the value of the cryptographic identity. The same value is also provided by S as cryptographic identity in its simulation. Lastly, S instructs F_{DA}^{RP} to provide the cryptographic identity

for $(ID_{M_1}, sid_{M_1}, r_{M_1})$. If S receives a notification that an instance has closed its session, S updates its internal simulation to reflect this and responds with *Okay*. All operations performed by F_{CR} are successful and have no side effects on M_I and M_R due to the use of the restricting message as presented in Section III-D.

We present an argument that the simulation is perfect using a case of uncorrupted initiator instances during energy node registration. Let (X, sid_X, r_X) be an uncorrupted instance of M_X that wants to perform an energy node registration with the master energy node $ID^{'}$. (X, sid_X, r_X) will use F_{CR} for cryptographic operations. We need to show that S finds a responder instance that can be paired with (X, sid_X, r_X) for the energy node registration. If (X, sid_X, r_X) validates and accepts a cryptographic identity, this shows that it has already accepted the second message of the *RPro* and the preshared key of the energy node $ID^{'}$ is still uncorrupted. This shows that a message $cmsg = Enc_{k_2}(ID_X, CC_X, pb_M, rv_M)$ is encrypted and MACed by some instance of $ID^{'}$, say $(ID^{'}, sid^{'}, r^{'})$, where $CC_X = pb_X + pb_{X2}$ and $pb_M = pv_M.G$. Thus, this instance $(ID^{'}, sid^{'}, r^{'})$ is uncorrupted and all the cryptographic operations were successful. Additionally, since $(ID^{'}, sid^{'}, r^{'})$ considers X to be its energy node registration partner (or energy node registration initiator) as recognised in the signature and MAC (i.e., in *cmsg*), this shows that $(ID^{'}, sid^{'}, r^{'})$ is not corrupted and thus $(ID^{'}, sid^{'}, r^{'})$ is a responder instance. We now argue that $r^{'} = r_{M_1}$. Suppose by contradiction $r^{'} = r_X$, this means that the second message of *RPro* was accepted by (X, sid_X, r_X). However, since pv_X/pb_X is ideally computed, this means that there is only one instance that would encrypt and MAC *cmsg* and (X, sid_X, r_X) does out output any encryptions or MACs before accepting the second message of *RPro*. Thus, this indicates that $r^{'} = r_{M_1}$.

By Theorem 2, we can now replace F_{CR} by P_{CR} which yields that the *RPro* is a universally composable registration protocol when using the actual cryptographic operations.

Proposition 1. *Let M_I, M_R be as defined above. Let F_{CR}, P_{CR}, and M^* be as in Theorem 2, where $P_{CR} \leq_R F_{CR}$ and M^* enforces well-behaved environments. Then, the following holds true:*

$$M^*|M_I|M_R|P_{CR} \leq_R M^*|F_{DA}^{RP}|F_{CR}$$

Proof. This proposition follows easily from Theorem 1, Theorem 2, and Theorem 3, and the fact that combination of $M^*|M_I|M_R$ and E creates a well-behaved environment for energy node registration. Note that: i) corrupted instances cannot violate the well-behaved property since they do not have access to uncorrupted keys; and ii) uncorrupted instances do not violate the well-behaved property since private keys are never accessed or reused after they have been used to create a secret key.

B. SECURITY ANALYSIS OF DAPRO

We mode the initiator ID_X and responder ID_Y of the *DAPro* as well as the responsive simulator in a similar way to the *RPro*, except that in this case, we use the machines for data awareness. At the end of *DAPro*, the instances provide data awareness in an ideal manner. The following theorem states that the *DAPro* is a secure and privacy-preserving universally composable data-aware protocol.

Theorem 4. *Let M_I and M_R be machines that model the DAPro as described above. Let F_{CR} and $F_{CR}^{'}$ be two versions of the ideal functionality for cryptographic operations with the same parameters, and let F_{DA}^{RP} be the ideal functionality for data awareness with parameters $ty_2 = P - ECDH$ key and $ty_3 =$ secret session key. Then, the following holds true:*

$$M_I|M_R|F_{CR} \leq_R F_{DA}^{RP}|F_{CR}^{'}$$

Proof Sketch. In this proof, if F_{DA}^{RP} specifies that a user (ID, sid, r) has started a data awareness, S updates its internal simulation accordingly. If an uncorrupted instance (ID_X, sid_X, r_X) accepts a random challenge e_Y, public key pb_Y, and random value rv_Y, then S instructs F_{DA}^{RP} to create a data awareness session from (ID_X, sid_X, r_X) and instance (ID_Y, sid_Y, r_Y) that computed the encryption and MAC in the second message of *DAPro*. $F_{CR}^{'}$ of F_{DA}^{RP} will request S provide a session key for data awareness. Then, S instructs F_{DA}^{RP} to output the pointer to the secret session key for (ID_X, sid_X, r_X) which is used to support the execution of smart contracts. Just as the *RPro*, by Theorem 2, we can replace F_{CR} by P_{CR} which yields that *DAPro* is a secure and privacy-preserving universally composable data awareness protocol.

Proposition 2. *Let M_I, M_R be as defined above. Let F_{CR}, P_{CR}, and M^* be as in Theorem 2. Then, the following holds true:*

$$M^*|M_I|M_R|P_{CR} \leq_R M^*|F_{DA}^{RP}|F_{CR}$$

Proof. It can be easily seen from Theorem 1, Theorem 2, and Theorem 4 that a well-behaved environment for data awareness is created by $M^*|M_I|M_R$ when combined with E.

In UReum, the data awareness security and privacy requirements are satisfied via integration of the cryptographic operations and providing effective transitions during data awareness. Furthermore, privacy and active man-in-the-middle attacks in the Energy Internet are mitigated via the successful execution of the operations amongst the actual energy nodes (using their anonymous identities). Thus, we observed that UReum has more security and privacy features (such as correctness and unlinkability, respectively) when compared to other related solutions [8], [9], [10], [11], [15], [16], [17], [18] in the related work section of this paper.

IX. IMPLEMENTATION AND EXPERIMENTS

In this section, we present the implementation of UReum and our experiments.

A. IMPLEMENTATION

UReum implementation consists of components, communication, and data awareness amongst the components. In the paper, we use Tmote Sky sensor nodes [35] as components

of the UReum implementation. These sensor nodes, such as CM3000 and CM5000 sensors, represent the basic energy nodes and master energy nodes described in Section IV-A and are widely used wireless sensor modules for collecting sensory data in energy grids and offer interoperability amongst IEEE 802.15.4 devices. The sensor nodes without storage resources are the basic energy nodes, while the sensor nodes with storage resources are the master energy nodes. We note that: (I) Every master energy node is equipped with a secure solid-state drive (SSD) for storing its UReum blockchain. (II) The implementation of UReum on Tmote Sky sensor nodes, which provide increased automation in the energy grids, provides insight into the actual deployment of *RPro* and *DAPro*.

The communication amongst the basic energy nodes and master energy nodes is based on the UReum transactions, while data awareness amongst the components is based on UReum smart contracts. Every energy node is registered via *RPro* as per Figure 7. Once registered, such a node can establish data awareness via *DAPro* as per Figure 8.

B. EXPERIMENTS

The goals of our experiments are as follows: (I) Evaluating the performance of UReum via observing the number of computational resources required to execute *RPro* and *DAPro* and then comparing the performance of UReum with existing blockchain schemes and data awareness solutions based on transactions and data awareness, respectively. (II) Measuring the End-to-End Delay (EED) of the protocols to show their impact on Energy Internet. To set up our experiments, we connect the Tmote Sky sensors to a MacBook Pro Machine with Ubuntu 18.04 TLS, 2.3Hz Dual-Core processor, 128GB SDD, and 8GB RAM size. All cryptographic operations including UReum transactions and UReum smart contracts are compiled under a TinyOS operating system [36] and executed on the sensors with their programs written in nesC language [37]. Note that we consider the possibility of using open-source blockchain-enabled platforms such as Ethereum [38] for our experiments. However, these platforms are presently unsuitable for Energy Internet due to differences in performance requirements and operations.

1) Performance Evaluation

We calculated the number of bits required for *RPro* and *DAPro* and then measured their processing times. The number of bits represents the size of the cryptographic operations, while the process times represent the amount of time required to execute the operations. We measure the processing times of the steps related to computing a register transaction RT and identity ID_X and accepting ID_X on the Tmote Sky sensors by executing cryptographic algorithms (such as 32 bits random number and 160 bits ECC point multiplication) since the main purposes of *RPro* is to request, derive, and accept and identity. Additionally, since the main purpose of *DAPro* is to request and derive a shared secret session key and use the shared secret session key to execute any smart contracts, we measure the processing time of the steps related to computing a data-aware transaction DT, deriving a shared secret session key, and executing smart contracts with the support of the shared secret session key and cryptographic algorithms such as SHA-256. We present the breakdown of the number of bits and processing times of the key steps in Table 5. We observe that the step of computing RT in *RPro* has the highest processing time due to the public keys pb_X and pb_{X2} computations via ECC point multiplication. Furthermore, the step of deriving a shared secret key has the least processing time in *DAPro* due to the use of the SHA-256 cryptographic algorithm.

TABLE 5. Breakdown of number of bits and processing times of key steps of *RPro* and *DAPro*

Registration Protocol (*RPro*)		
Steps	Number of Bits (in bits)	Processing Time (in sec)
Compute a register transaction RT	352	$\approx$ 2.080007
Compute an identity ID_X	160	$\approx$ 1.581
Accept ID_X	160	$\approx$ 2.017
Data-aware Protocol (*DAPro*)		
Steps	Number of Bits (in bits)	Processing Time (in sec)
Compute a data-aware initiation transaction DT	736	$\approx$ 3.67016
Derive a shared secret session key k_4	256	$\approx$ 0.00916
Execute ST and ST_V with k_4	1,064	$\approx$ 0.027545
Execute DRT with k_4	448	$\approx$ 0.020014
Execute RVT and $rmsg$ with k_4	1,000	$\approx$ 4.21122

Remarks: We compare the performance of UReum with popularly used blockchains in the energy grid, i.e., Ethereum and Hyperledger Fabric [39], in terms of average transaction size, and the results show that UReum with 74 bytes (across RT with 44 bytes, DT with 92 bytes, ST with 101 bytes, DRT with 56 bytes, and RVT with 77 bytes) offers better performance than Ethereum with 271 bytes and Hyperledger Fabric with 3,000 bytes. Thus, UReum requires the least number of bits for every transaction compared to Ethereum and Hyperledger Fabric. Due to the difference in hardware requirements, we leave the performance comparison UReum with the above blockchains in terms of average processing time for future work.

2) End-to-End Delay (EED)

We simulated the EED of *RPro* and *DAPro* using the widely accepted Peer-to-Peer (P2) network simulation tool, Network Simulator 3 (NS-3) [40]. The EED represents the combination of the time taken for a sender energy node to prepare a data packet, the time taken for the network to deliver the packet, and the time taken for a receiver energy node to process the packet or depacketization. Details of the simulation parameters used in the NS-3 simulation are provided in Table 6. Other standard parameters, such as flow monitor for network protocols performance measurements, are used in the NS-3. We note that: i) the P2P channel model is used for communication between two energy nodes and the CSMA

TABLE 6. Simulation Parameters

Parameter	Description
Platform	Ubuntu 18.04 TLS
Network Scenarios	Scenario 1 for *RPro*; Scenario 2 for *DAPro*
Number of basic energy nodes	2 for Scenario 1; 10 for Scenario 2
Number of master energy nodes	50 for all scenarios
Communication Medium	Wi-Fi
Channel Model	P2P for 2 energy nodes; CSMA for many energy nodes
Transport Protocol	UDP

channel model is used for communication between an energy node and a group of other energy nodes; and ii) the Wi-Fi communication medium is widely used in the energy grid due to its reliability, ease of connectivity, and data rate flexibility for improved grid connection and communication. Note that the bit length of various binary sequences in UReum are as follows: i) 32 bits for energy node name; ii) 160 bits for energy node identity; iii) 32 bits for random values; iv) 256 bits for hash function; v) 160 bits for MAC; vi) 128 bits for symmetric encryption/decryption; and vii) 160 bits for public-key encryption/decryption.

In *RPro*, two different types of messages are used, namely (X, pb_X, pb_{X2}) and $(cmsg, \beta, pb_M, rv_M)$ which are of sizes 352 bits and 1,088 bits, respectively. In *DAPro*, three different types of messages are used as follows: i) the messages used for *ST* are $(CC_X, pe_X, pb_{X3}, ID_X)$, $(cmsg_2, \beta_2, pb_Y, e_Y)$, and $((t_2, Hash(t_2)), Hash(k_4, (t_2, Hash(t_2))))$ which are of sizes 736 bits, 672 bits, and 808 bits, respectively; ii) the messages used for *DRT* are $(CC_X, pe_X, pb_{X3}, ID_X)$, $(cmsg_2, \beta_2, pb_Y, e_Y)$, and $(MAC_{k_4}(Enc_k(data)), Enc_{k_4}(data))$ which are of 736 bits, 672 bits, and 448 bits, respectively; and iii) the messages used for *RVT* are $(CC_X, pe_X, pb_{X3}, ID_X)$, $(cmsg_2, \beta_2, pb_Y, e_Y)$, and $(Sig_{pv_X}(t_0, X, ID_X), t, ID_X, pb_X, Hash(k_4, Sig_{pv_X}(t_0, X, ID_X)))$ which are of 736 bits, 672 bits, and 904 bits, respectively. The EED values are approx. 6.0384 msec for Scenario 1 and approx. 8.6952 msec, approx. 7.0103 msec, and approx. 9.2768 msec for *ST*, *DRT*, and *RVT*, respectively, for Scenario 2. Hence, we can see that the EED values for scenarios 1 and 2 are below our maximum 2 sec target of some of the latency requirements of energy system applications and technologies such as wide-area situational awareness, state estimation, transient stability, small signal stability, and distribution grid management with latencies 20 msec to 200 msec [12], 1 sec [14], 100 msec [14], 1 sec [14], and 10 msec to 2 sec [12], respectively. We note that the number of energy nodes does not impact the latency of UReum since no energy node acts as a route to other energy nodes in the network. Thus, (i) an increase in the number of energy nodes in the network does not affect UReum's performance, thereby supporting UReum's scalability in the Energy Internet, and (ii) UReum suffices the time and delay sensitivities nature of energy system data exchange. Additionally, as there is no continuous communication amongst energy nodes after a data awareness is completed, this shows that UReum supports the time and delay sensitivities nature of the energy system data exchange. Compared with the existing related solutions [8], [9], [10], [11], [15], [16], [17], [18] in the related work section, we showed that UReum is fit and suitable for energy systems via meeting the 20 msec to 2 sec latency target.

X. CASE STUDIES

In this section, we carry out two case studies to demonstrate the usefulness of UReum.

A. CASE STUDY I

We analyse an issue encountered by an energy grid entity regarding loss of SE due to contradicting information as a result of a corrupted database across energy operators such as Transmission Owners (TOPs), Reliability Coordinators (RCs), and Balancing Authorities (BAs) in a real-world energy grid [41]. The SE, which provides an estimate of the operational state of energy systems or devices, received information on a device status from the primary Inter-Control Center Communications Protocol (ICCP) cluster (i.e., Site 1) and a backup ICCP cluster (i.e., Site 2). The database on the backup ICCP cluster had incorrect indexes associated with the statuses and values of the device thereby affecting the availability of the SE and Real-Time Contingency Analysis (RTCA). The entity notified the RC of the unavailability of the SE and RCTA and requested the RC to monitor contingencies until the SE and RCTA are restored. While it was not determined how the issue occurred, a reboot of the backup ICCP cluster fixed the issue. However, the correctness, assurance, fairness, and confidentiality of data as well as providing real-time insight into the issue to all associated energy operators (like TOPs and BAs) remain major concerns. These show that the measure deployed in fixing the issue is not capable of preventing the reoccurrence of the issue as well as providing a secure and privacy-preserving data awareness, which supports correctness, assurance, fairness, and confidentiality of data. A simple description of the issue is illustrated in Figure 9. This figure shows that the primary ICCP cluster, entity, backup ICCP cluster, and RC are not capable of preventing the issue and they do not satisfy the security and privacy requirements of UReum. To see this, consider the following SE setting: The sites $s1$ and $s2$ (i.e., the primary ICCP and backup ICCP clusters) send a new device status ds_2 to *En* (i.e., the entity). *En* might have received either a correct or incorrect ds_2 from $s2$ with a (corrupted) database. Thus, we have no correctness, assurance, fairness, and confidentiality guarantees for ds_2 and the corrupted database can cause a loss of SE due to contradicting ds_2. Furthermore, an attacker can let *En* accept a compromised ds_2 since there is no authentication between $s1$ and *En* as well as between $s2$ and *En* and further cause privacy and active man-in-the-middle attacks. In this setting, RC is not aware of the above issue since ds_2 was only sent to *En*. Thus, the current capabilities of *En* and the database are not adequate to prevent the reoccurrence of the issue as

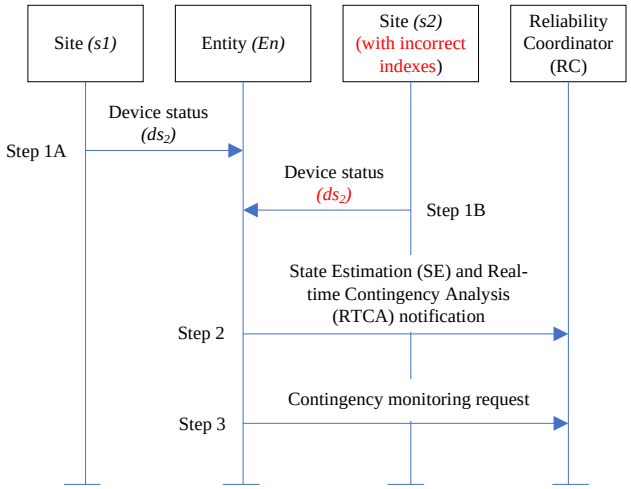

FIGURE 9. A simple description of the loss of State Estimator (SE) issue in an energy grid.

well as security attacks (i.e., privacy and active man-in-the-middle attacks) and to provide insight on ds_2 to RC without an initiated communication from *En* to RC.

To fix these problems, we first equip $s1$, $s2$, and RC with a UReum blockchain each to provide data immutability and avoid database corruption, and then present an enhancement as follows: (I) Execute *RPro* for $s1$, $s2$, and *En* in the setting before sending any status data to mitigate privacy attacks. More precisely, via generating a unique cryptographic identity for $s1$, $s2$, and *En* as well as utilising ComPS and Enc/Dec commands in *RPro* (see Section VI for details of the commands). (II) Execute *DAPro* with *DRT* for ds_2 between $s1$ and *En* as well as between $s2$ and *En* in the setting to establish data awareness via *DT* and provide (universally composable) guarantees for correctness (via ComPK/DerSKey/MAC/VMAC commands), assurance (via VerID command), fairness (via *DRSC*), and confidentiality (via Enc/Dec commands) (see Sections IV-D and VI for details of *DT* and the commands, respectively). Note that: i) the utilisation of *RPro*, as well as ComPK/DerSKey/Enc/Dec commands, in *DAPro* further mitigates privacy attacks; ii) the utilisation of the ComPK/DerSKey commands in *DAPro* mitigates active man-in-the-middle attacks; and iii) the utilisation of *DRSC* ensures that RC and other entities are aware of ds_2 (in real-time). Hence, using *RPro* and *DAPro* with the support of the UReum blockchain, transactions, and smart contracts provide data awareness enhancement in the SE.

B. CASE STUDY II

We analyse an issue where an EMS becomes unavailable to operators in a real-world energy grid entity [42]. A flaw in an antivirus software engine installed on the EMS production servers led to the unresponsiveness of the servers. Due to the difference in input/output (I/O) workload across the production and test servers, the flaw was not recognised in the test server environments. The issue, which occurred continuously for over two weeks, prevented the entity from Reporting Area Control Error (ACE), Implementing Automatic Generation Control (IAGC), Real-Time Contingency Analysis (RTCA), SE, and Real-Time Monitoring and Alarming (RTMA). Disabling selected services, uninstallation of flawed malware engine, and service restoration resolved the issue. However, data correctness, assurance, and fairness remain major concerns. A simple description of the issue is illustrated in Figure 10. This figure shows that the EMS is not capable of mitigating the abovementioned issue. To see this, consider the following EMS setting: The energy grid entity *En* installed an antivirus software engine on the EMS. As EMS does not check whether such installation contains counterfeit data, it might have received a flawed antivirus software engine. Thus, there are no correctness, assurance, and fairness guarantees for the antivirus software engine, and the installation can cause a loss of EMS at any time in the energy grid.

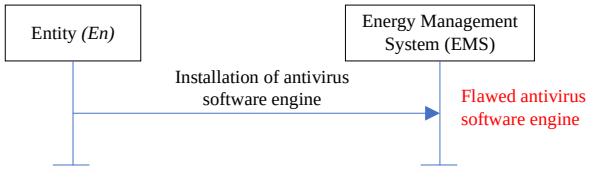

FIGURE 10. A simple description of the loss of Energy Management System (EMS) in an energy grid.

To fix the abovementioned problems, we equip *En* and EMS with a UReum blockchain and the executions of *RPro* and *DAPro* to establish secure data awareness and provide guarantees for data correctness, assurance, and fairness (see Section VII for details of *RPro* and *DAPro*). Hence, using *RPro* and *DAPro* enhances data awareness on antivirus software engine across *En* and EMS.

XI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed UReum, a secure and privacy-preserving data awareness model using universal composability and blockchain. In Energy Internet, energy nodes can provide data awareness or insights into energy operations without involving energy operators and leaking sensitive information using UReum. We have proposed an ideal functionality F_{CR} that supports cryptographic primitives such as authentication and our P-ECDH and models the data awareness security requirements (such as data correctness, energy node identity assurance, and transaction fairness) and privacy requirements such as energy node identity anonymity, data confidentiality, and transaction unlinkability. We also proposed an ideal functionality F_{DA}^{RP} for secure and privacy-preserving data awareness. F_{DA}^{RP} uses F_{CR} as a subroutine for cryptographic operations and enforces the data awareness security and privacy requirements in the Energy Internet. UReum uses an *RPro* based on EC-PCS and ECDH to register energy nodes and assign them a cryptographic identity, a *DAPro* based on EC-

ZKPK and P-ECDH to create data awareness and mitigates privacy and active man-in-the-middle attacks. Compared with existing data awareness solutions in the energy grid, UReum provides security in arbitrary adversarial environments and offers strong universally composable guarantees on satisfying the data awareness security and privacy requirements. We implemented UReum and the results show that it meets the latency target of the Energy Internet. Furthermore, we demonstrated the usefulness of UReum in two real-world case studies, which show that energy grid entities are either unable to prevent the loss of SE or mitigate the loss of EMS, and we used UReum to enhance the capabilities of the entities. In future work, we plan to enhance our model with distributed key management, introduce new transactions such as penalize transactions, and implement our scheme atop an open-source blockchain platform.

REFERENCES

[1] A. Q. Huang, M. L. Crow, G. T. Heydt, J. P. Zheng, and S. J. Dale, "The Future Renewable Electric Energy Delivery and Management (FREEDM) System: The Energy Internet," Proceedings of the IEEE, vol. 99, no. 1, pp. 133-148, 2011.

[2] H. Tianfield, "Cyber security situational awareness," in 2016 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData), 2016, pp. 782-787: IEEE.

[3] A. S. Sani, D. Yuan, W. Bao, and Z. Y. Dong, "Towards secure energy internet communication scheme: An identity-based key bootstrapping protocol supporting unicast and multicast," in 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, 2017, pp. 1-5: IEEE.

[4] R. Kusters, "Simulation-Based Security with Inexhaustible Interactive Turing Machines," presented at the Proceedings of the 19th IEEE workshop on Computer Security Foundations, 2006.

[5] R. Küsters and M. Tuengerthal, "The IITM Model: a Simple and Expressive Model for Universal Composability," IACR Cryptology EPrint Archive, vol. 2013, p. 25, 2013.

[6] V. S. Miller, "Use of elliptic curves in cryptography," in Conference on the Theory and Application of Cryptographic Techniques, 1985, pp. 417-426: Springer.

[7] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in Advances in Cryptology—CRYPTO'86, 1986, pp. 186-194: Springer.

[8] H. Liu, H. Ning, Y. Zhang, and M. Guizani, "Battery status-aware authentication scheme for V2G networks in smart grid," IEEE Transactions on Smart Grid, vol. 4, no. 1, pp. 99-110, 2013.

[9] J. Lin et al., "Situation awareness of active distribution network: Roadmap, technologies, and bottlenecks," CSEE Journal of Power and Energy Systems, vol. 2, no. 3, pp. 35-42, 2016.

[10] S. Ghosh, D. Ghosh, and D. K. Mohanta, "Situational awareness enhancement of smart grids using intelligent maintenance scheduling of phasor measurement sensors," IEEE Sensors Journal, vol. 17, no. 23, pp. 7685-7693, 2017.

[11] J. Wu, K. Ota, M. Dong, J. Li, and H. Wang, "Big data analysis-based security situational awareness for smart grid," IEEE Transactions on Big Data, vol. 4, no. 3, pp. 408-417, 2016.

[12] U. DoE, "Communications requirements of Smart Grid technologies," US Department of Energy, Tech. Rep, pp. 1-69, 2010.

[13] K. C. Budka, J. G. Deshpande, T. L. Doumi, M. Madden, and T. Mew, "Communication network architecture and design principles for smart grids," Bell Labs Technical Journal, vol. 15, no. 2, pp. 205-227, 2010.

[14] P. Kansal and A. Bose, "Bandwidth and latency requirements for smart transmission grid applications," IEEE Transactions on Smart Grid, vol. 3, no. 3, pp. 1344-1352, 2012.

[15] C.-W. Tsai, A. Pelov, M.-C. Chiang, C.-S. Yang, and T.-P. Hong, "Computational awareness for smart grid: a review," International journal of machine learning and cybernetics, vol. 5, no. 1, pp. 151-163, 2014.

[16] Y. Chawla and A. Kowalska-Pyzalska, "Public awareness and consumer acceptance of smart meters among Polish social media users," Energies, vol. 12, no. 14, p. 2759, 2019.

[17] A. Singh, A. Pooransingh, C. J. Ramlal, and S. Rocke, "Toward Social Awareness in the Smart Grid," in 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN), 2016, pp. 537-543: IEEE.

[18] S. A. U. Nambi and R. V. Prasad, "Toward the development of a techno-social smart grid," IEEE Communications Magazine, vol. 54, no. 11, pp. 202-209, 2016.

[19] L. Zhu et al., "FNET/GridEye: A tool for situational awareness of large power interconnetion grids," in 2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe), 2020, pp. 379-383: IEEE.

[20] Y. Liu et al., "A distribution level wide area monitoring system for the electric power grid–FNET/GridEye," IEEE Access, vol. 5, pp. 2329-2338, 2017.

[21] A. S. Sani, D. Yuan, and Z. Y. Dong. "Sdag: blockchain-enabled model for secure data awareness in smart grids." In 2023 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), pp. 1-5. IEEE, 2023

[22] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in Annual International Cryptology Conference, 1991, pp. 129-140: Springer.

[23] J. Camenisch et al., "Universal Composition with Responsive Environments," presented at the Proceedings, Part II, of the 22nd International Conference on Advances in Cryptology — ASIACRYPT 2016 - Volume 10032, 2016.

[24] R. Küsters and M. Tuengerthal, "Ideal Key Derivation and Encryption in Simulation-based Security," presented at the Proceedings of the 11th international conference on Topics in cryptology: CT-RSA 2011, San Francisco, CA, USA, 2011.

[25] R. Küsters and M. Tuengerthal, "Composition theorems without pre-established session identifiers," presented at the Proceedings of the 18th ACM conference on Computer and communications security, Chicago, Illinois, 2011.

[26] Y. Wang, T. T. Gamage, and C. H. Hauser, "Security Implications of Transport Layer Protocols in Power Grid Synchrophasor Data Communication," IEEE Transactions on Smart Grid, vol. 7, no. 2, pp. 807-816, 2016.

[27] A. Al-Ali and R. Aburukba, "Role of internet of things in the smart grid technology," Journal of Computer and Communications, vol. 3, no. 05, p. 229, 2015.

[28] T. Gonnot and J. Saniie, "User defined interactions between devices on a 6LoWPAN network for home automation," in 2014 IEEE International Technology Management Conference, 2014, pp. 1-4: IEEE.

[29] R. Kuesters and D. Rausch, "A Framework for Universally Composable Diffie-Hellman Key Exchange," in "Cryptology ePrint Archive, Report 2017/256," 2017, Available: https://eprint.iacr.org/2017/256.

[30] D. Boneh, "The decision diffie-hellman problem," in International Algorithmic Number Theory Symposium, 1998, pp. 48-63: Springer.

[31] M. Abdalla, M. Bellare, and P. Rogaway, "The oracle Diffie-Hellman assumptions and an analysis of DHIES," in Cryptographers' Track at the RSA Conference, 2001, pp. 143-158: Springer.

[32] R. Canetti and M. Fischlin, "Universally composable commitments," in Annual International Cryptology Conference, 2001, pp. 19-40: Springer.

[33] J. Black, P. Rogaway, and T. Shrimpton, "Encryption-scheme security in the presence of key-dependent messages," in International Workshop on Selected Areas in Cryptography, 2002, pp. 62-75: Springer.

[34] M. Backes, B. Pfitzmann, and A. Scedrov, "Key-dependent message security under active attacks - BRSIM/UC-soundness of Dolev-Yao-style encryption with key cycles," J. Comput. Secur., vol. 16, no. 5, pp. 497-530, 2008.

[35] Tmote Sky, "Ultra low power IEEE 802.15. 4 compliant wireless sensor module," Moteiv Corporation, 2006.

[36] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," ACM SIGOPS operating systems review, vol. 34, no. 5, pp. 93-104, 2000.

[37] D. Gay, P. Levis, R. v. Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," presented at the Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, San Diego, California, USA, 2003.

[38] G. Wood, "Ethereum: A secure decentralized transaction ledger," ed, 2014.

[39] E. Androulaki et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in Proceedings of the thirteenth EuroSys conference, 2018, pp. 1-15.
[40] NS-3 Consortium. The Network Simulator 3. [Online]. Available: https://www.nsnam.org/
[41] NERC. Lesson Learned: Loss of State Estimator due to Contradicting Information from Dual ICCP Clusters. [Online]. Available: https://www.nerc.com/pa/rrm/ea/Lessons%20Learned%20Document%20Library/LL20201102_Loss_of_SE_due_to_Contradicting_Information_from_Dual_ICCP_Clusters.pdf
[42] NERC. Lesson Learned: Loss of Energy Management System Functionality due to Server Resource Deadlock. [Online]. Available: https://www.nerc.com/pa/rrm/ea/Lessons%20Learned%20Document%20Library/LL20220901_EMS_Server_Resource_Deadlock.pdf