Privacy-Preserving Exchange Mechanism and its Application to Energy Market

Victor Langguille
Télécom Paris, EDF R&D
Palaiseau, France
victor.langguille@telecom-paris.fr

Hamza Zarfaoui
Télécom Paris, EDF R&D
Palaiseau, France
hamza.zarfaoui@telecom-paris.fr

Gerard Memmi
LTCI, Télécom Paris
Palaiseau, France
gerard.memmi@telecom-paris.fr

David Menga
EDF R&D
Palaiseau, France
david.menga@edf.fr

***Abstract*—Considering the context of energy market, we propose a blockchain-based privacy-preserving protocol supporting very general kinds of auctions, keeping private not only the bids and participant identities but also auction result, all in a completely verifiable way. We use our scheme described through four algorithms, to instantiate multi-units double auctions which are used in our application to protect privacy in microgrids. A prototype is implemented and performances are presented.**

***Index Terms*—blockchain, DLT, auction, privacy, microgrid, zero-knowledge proofs**

I. Introduction

Since the paradigm of microgrids is based on a decentralized approach, Distributed Ledger Technologies (DLT), which enable mutually distrustful agents to maintain a common register in a robust and transparent way, have been proposed as the basis for the architecture of their information system. In this context they serve as a support for exchange platform for the local energy market [1] [2] [3] [4].

Operating on the basis of a publicly accessible register generally comes at the price of privacy, which poses a major problem given the sensitivity of energy data, both from the individual and commercial point of view.

Fortunately, a number of techniques for reconciling blockchain and privacy, notably for building anonymous cryptocurrencies, have been developed [5].

In our work, we chose Zerocash [5] (one of the anonymous cryptocurrency protocol) with some modifications to build a privacy-preserving transactive platform. We built upon it a general purpose auction protocol allowing several participants and an auctioneer to take part in all kind of auction mechanisms. We formally define it section II, while keeping private not only the bids of participants but also the auction result, which contrasts with most of the concurrent privacy-preserving proposals working on public ledger.

We use this new privacy-preserving platform to implement the strategy-proof double auction mechanism between energy token and monetary units, prototyped for microgrids management introduced in [1].

In Section II, we precise the kinds of exchange allowed by our protocol as well as the supported privacy properties. We briefly compare our work to related works section II-C, particularly in terms of levels of privacy.

In section III, we present the Zerocash-like infrastructure we developed, including its modifications specially adapted to exchange mechanisms (section III-C), and upon which we built our application to the energy market section IV. In section V, we apply it to the context of energy market trading and show our implementation performance before concluding.

II. Preliminaries

We consider a microgrid operated by a Distribution System Operator (DSO) managing smart meters measuring production and consumption of distinct prosumers .

We aim at building an architecture for energy exchanges, operating on public ledgers, such that anyone reading the ledger is convinced that no unauthorized behavior has taken place, while learning nothing about either the energy load profile or the commercial behavior of prosumers. In addition, we want to put as little trust as possible in any party. Namely, we don't rely on any DSO services save for smart meters assuming honesty in measuring energy production and consumption, and compliance to their specification. However, we do rely on an auctioneer $\mathcal{T}$, whose role can typically be held by the DSO or by successively sorted participants, considered as an "Almost-not" Trusted Third Party with the following meaning:

"Almost-not" Trusted Third Party: participants are not a priori convinced that $\mathcal{T}$ strictly follows the specified protocol. Participants are a priori entitled to consider that $\mathcal{T}$ could deviate from executing actions in conformity with their specifications either by not or partially executing them or by executing actions not requested. participants could even suspect collusion via extra-communication between $\mathcal{T}$ and other participants. It is strongly desirable that such a third party learns the less participant private data as possible.

First, we describe a Sealed-bid Exchange Mechanism (SBExM), the extremely general kind of market mechanism our protocol will allow supporting in a privacy-preserving and verifiable manner, and upon which our application will be build in section V. Classical first-price auction, VCG auction, multi-unit double auction are all particular cases of SBExM.

Then, we describe the security and privacy properties we consider and briefly position our contribution to related works section in table I sectionII-C.

A. Sealed-Bid Exchange Mechanism (SBExM)

SBExM involves a list of n participants $[\mathsf{i}]_{\mathsf{i}=1}^{\mathsf{n}}$ and an auctioneer $\mathcal{T}$. A finite set T of asset's type t represents the different kinds of goods that can be exchanged. Typically, one of them corresponds to monetary units, and in the context of local energy market, one of them will represent energy tokens.

SBExM is specified by an exchange function $\mathcal{F}$ $(\Gamma_{\mathcal{T}}^{\mathsf{in}}, [(\Gamma_{\mathsf{i}}^{\mathsf{in}}, \mathsf{b_i})]_{\mathsf{i}=1}^{\mathsf{n}}) \rightarrow (\Gamma_{\mathcal{T}}^{\mathsf{out}}, [\Gamma_{\mathsf{i}}^{\mathsf{out}}]_{\mathsf{i}=1}^{\mathsf{n}}, \mathsf{info})$. It takes as inputs $[\Gamma_{\mathsf{i}}^{\mathsf{in}}]_{\mathsf{i}=1}^{\mathsf{n}}$ (resp $\Gamma_{\mathcal{T}}^{\mathsf{in}}$), the provided funds of participants $[\mathsf{i}]_{\mathsf{i}=1}^{\mathsf{n}}$ (resp the auctioneer) and $[\mathsf{b_i}]_{\mathsf{i}=1}^{\mathsf{n}}$, the bids of participants. It outputs $[\Gamma_{\mathsf{i}}^{\mathsf{out}}]_{\mathsf{i}=1}^{\mathsf{n}}$ (resp $\Gamma_{\mathcal{T}}^{\mathsf{out}}$), the received funds of participants (resp the auctioneer) and info, a bitstring representing any information about inputs/outputs of $\mathcal{F}$ intended to be public [1]. In case no information about the exchange are revealed, the bitstring info is void.

$\mathcal{F}$ is also required to satisfy the balance property, namely, it neither creates nor destroys any goods.

Concretely, the provided and received funds Γ are tuples $[\mathbb{q}_{\mathsf{t}}]_{\mathsf{t}\in\mathsf{T}}$ of positives reals were each $\mathbb{q}_{\mathsf{t}}$ represents the quantity of assets of type t. In particular, if type $\mathsf{t_0}$ corresponds to a non-fungible good, $\mathbb{q}_{\mathsf{t_0}}$ can only take value 0 or 1.

Each participant bid $\mathsf{b_i}$ is a function $\mathsf{b_i}(\Gamma) \rightarrow \mathbb{v}_{\mathsf{i}}$ representing the value $\mathbb{v}_{\mathsf{i}}$ that i gives to the potential outcome Γ. We obviously need the additional assumption that $\mathsf{b_i}$ is finitely representable, e.g. being entirely specified by its values on 1 unit of each type and extended by linearity[2].

A function $\mathsf{Leak}(\Gamma^{\mathsf{in}}, \mathsf{b}) \rightarrow \mathsf{info_{bid}}$ specifies a (potentially void) bitstring $\mathsf{info_{bid}}$ representing information about participant's provided funds and bids intended to be public.

B. Security and Privacy

1) Security: The security guarantees that we provide, are those usually required in exchange mechanisms.

Integrity : Unspecified movement of funds cannot happen.
Safety : Honest participants cannot lose funds.

Non-Repudiation : Participants cannot refuse the auction result after they accepted to participate.

2) Privacy: We consider several privacy guarantees that one could expect from a privacy-preserving exchange mechanism, and in particular an auction protocol on blockchain.

- **Bidders' Anonymity**: bidders' identity is kept secret.
- **Bidders' Confidentiality**: bidders' bids are kept secret.
- **Winners' Anonymity**: winners' identity is kept secret.
- **Winners' Confidentiality**: winners' receiving is kept secret.

Each of these privacy guarantee will come in three flavors: against the other bidders, against the auctioneer, and against the Ledger, depending upon the kind of entity considered as Adversary. Note that the Ledger being public, any privacy property that does not hold against the Ledger cannot hold against any other entity.

Remarks: Our protocol supports an auction exchange function $\mathcal{F}$ that outputs a public string info, and the function Leak also outputs a public string $\mathsf{info_{bid}}$. The privacy guarantees provided by our protocol directly involve at it doesn't leak any information beyond the information voluntarily revealed by info and $\mathsf{info_{bid}}$. In particular, when the functions $\mathcal{F}$ and Leak are chosen such that info and $\mathsf{info_{bid}}$ are void, our protocol provides complete confidentiality against the ledger.

Privacy against other participants is necessarily a little bit weaker than against an outside agent who would simply observe the ledger: since they know their own inputs and outputs from the exchange function $\mathcal{F}$ participants can infer partial information about the inputs of other participants[3].

So again, the privacy guarantee against the other participants is that they know nothing more than the information they can extract by knowing their own inputs and outputs only [4].

C. Comparison with related works

Table I below informally benchmarks our protocol with others blockchain-based private auction scheme. While not directly comparable, we added to the benchmark some non blockchain-based but publicly verifiable privacy-preserving auction schemes proposed in the microgrid context. The amount of trust put in third party in our protocol is less than or equal to the amount of trust other protocols put in third parties (except for [6]). Notably, most of the others blockchain-based protocols such as Ethereum, fail to achieve winner anonymity because they do not rely on privacy-preserving payment scheme, but rather on ordinary payment scheme. Unlike some other propositions, our protocol is quantum resistant, not relying on specific constructions built on the discrete logarithm problem. On another note, our protocol requires noticeable computational power compared to concurrent work.

III. Zerocash-like infrastructure

To obtain an anonymous exchange mechanism, we rely on an anonymous payment infrastructure.

Cryptocurrencies such as Bitcoin or Ethereum are pseudonymous but not anonymous [14]. Zerocash [5] is a protocol introduced circa 2014 to remedy this situation by proposing an anonymous and confidential cryptocurrency, where transactions reveal neither their senders and receivers nor the amount of coins exchanged.

This Zerocash-like infrastructure is streamlined compared to the original Zerocash and contains slight but key modifications (described at the end of this section) in order to better fit with the needs required by the SBExM protocol presented section IV. We refer to the original paper for a more in depht and pedagogical presentation [5].

[1]As an example, such a bitstring can correspond to the value of the highest bid, a final exchange price, a number of bids less than a given threshold etc.

[2]Non linear function b are still allowed and potentially useful in practice. E.g. a manufacturer needing at least 100kWatt to keep the machines running can value zero any amount less than 100kWatt, and value positively any amount of kWatt greater than or equal to 100.

[3]As an example, imagine a function $\mathcal{F}$ executing a first-price private bid auction of a unique good a between two participants. Necessarily, the winner will know that the loser's bid was less than their own bid.

[4]The issue and guarantee is the same as in Multi-Party Computation protocols.

	Our Protocol	[7]	[6]	[8]	[9]†	[10]†	[11]	[12]	[13]†
Bidders' Privacy vs Ledger	$\checkmark$	$\checkmark$	$\checkmark$	$\sim+$	$\checkmark$	$\emptyset$	$\sim+$	$\sim+$	$\emptyset$
Bidders' Privacy vs Third Party	$\sim+$	$\sim-$	$\checkmark$	$\sim+$	$\sim+$	$\sim+$	$\times$	$\sim-$	$\sim$
Winners' Privacy vs Ledger	$\checkmark$	$\sim+$	$\times$	$\sim+$	$\sim+$	$\emptyset$	$\sim+$	$\sim+$	$\emptyset$
Winners' Privacy vs Third Party	$\sim+$	$\times$	$\times$	$\times$	$\sim+$	$\sim+$	$\times$	$\sim-$	$\sim$
Generality/Auction type	all	first-price	DA	VCG	DA	first-price	simple transfer	DA	DA
Post-Quantum Transposable	$\checkmark$	$\checkmark$	$\times$	$\times$	$\times$	$\times$	$\checkmark$	$\checkmark$	$\times$

TABLE I: **Comparison with others privacy-preserving auction mechanism.**
$\checkmark$ means that the property is supported, $\times$ not supported, $\sim \pm$ partially supported (e.g. when anonymity is supported but not confidentiality) and $\emptyset$ that the property is not relevant. DA stands for "Double Auction". References marked by † are not public-ledger based.

A. Cryptographic building blocks

Our infrastructure offers different cryptographic building blocks to achieve our privacy guarantees described section II-B. We only give informal definitions, referring to classic literature for rigorous and standard ones, altogether with corresponding security definitions. We omit the public parameters taken as input by algorithms.

A Public-Private Key Encryption Scheme [15] [16]: $\Pi_{\mathsf{enc}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined by the algorithms:

- $\mathsf{KeyGen}(\mathsf{sk}) \rightarrow \mathsf{pk}$: Creates a public key pk from a private (or secret) key sk.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \rightarrow \mathfrak{C}$: Takes as inputs a public key pk and a message m. Outputs a cipher text $\mathfrak{C}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathfrak{C}) \rightarrow \mathsf{m}$: Takes as inputs a private key and a cipher $\mathfrak{C}$ encrypted for pk. Outputs the original clear message m.

We require the public-key encryption scheme to be *IND-CCA1 Secure* and *Key-Privacy*[5].

A Pseudorandom Function Family [17]: $\mathsf{PRF} = \{\mathsf{PRF}_{\mathsf{x}} : \{0,1\}^* \rightarrow \{0,1\}^{O(|\mathsf{x}|)}\}_{\mathsf{x}}$ where x denotes a seed computationally indistinguishable from random function family.

A Collision-Resistant Hash function: $\mathsf{H}(\mathsf{m}) \rightarrow \mathsf{h}$

A Commitment Scheme [17]: *computationally binding* and *hiding* $\mathsf{Com}(\mathsf{m}, \mathsf{r}) \rightarrow \mathsf{cm}$ taking as inputs a message m and a nonce r and returning the corresponding commitment.

A Non-Interactive Zero-Knowledge Poof (NIZK) [17]: $\Pi_{\mathsf{proof}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ allowing a prover to prove to a verifier that, given a statement defined by an NP relation $\mathcal{R}(a, b)$ and an instance (i.e public inputs) $\mathbb{x}$, they knows a witness (i.e private inputs) $\mathbb{w}$ such that $\mathcal{R}(\mathbb{x}, \mathbb{w})$.

- $\mathsf{Prove}(\mathbb{x}, \mathbb{w}) \rightarrow \pi$: Takes as inputs instance and witness. Outputs a proof π that $\mathcal{R}(\mathbb{x}, \mathbb{w})$.
- $\mathsf{Verify}(\mathbb{x}, \pi) \rightarrow \mathsf{b}$: Takes as input instance $\mathbb{x}$ and proof π. Outputs 1 if the proof is valid and 0 otherwise.

B. Infrastructure functioning

As a first approximation, it could be said that unlike Bitcoin, which operates on the model of bank accounts being credited and debited, Zerocash's operations are similar to that of fiat money, where banknotes are exchanged from hand to hand; a user's wealth being the sum of the values of the banknotes they hold. However, the analogy is not perfect: during the transaction from a user A to a user B one or several old notes owned by A are destroyed while one or several new notes are created to be owned by B.

The underlying Zerocash-like infrastructure is specified by the following data structure and algorithms:

Infrastructure setup:

Ledger: The participants share a common *append-only* and *public* ledger. Such ledger can typically be implemented as distributed ledger such as a blockchain. The ledger maintains three distinct lists CmList, SnList and TxList, the contents of which will be specified below.

Public-Private Keys: Each participant has a pair of public-private keys $(\mathsf{pk}, \mathsf{sk})$ where pk is created from sk, and distributed to any other participant susceptible to send them funds [6].

Notes: Each participant owns a wallet containing *notes* which constitute their wealth, i.e the totality of their goods. A *note* $\mathbf{n}$ is a set of data $\mathbf{n} = (\Gamma, \mathsf{pk}_{\mathsf{u}}, \rho, \mathsf{r}, \mathsf{cm})$ where:

- $\mathbf{n}.\mathsf{value}() = \Gamma$ is a tuple $[\mathbb{q}_{\mathsf{t}}]_{\mathsf{t} \in \mathsf{T}}$ with $\forall \mathsf{t} \in \mathsf{T},\ \mathbb{q}_{\mathsf{t}} \in \mathbb{R}+$, corresponding to the values of the different good embodied by the *note*.
- $\mathbf{n}.\mathsf{addr}() = \mathsf{pk}_{\mathsf{u}}$ is the *public address* of the *note* owner u.
- $\mathbf{n}.\mathsf{seed}() = \rho$ is a secret nonce used to generate the *note*.
- $\mathbf{n}.\mathsf{rand}() = \mathsf{r}$ a random number used to generate the commitment.
- $\mathbf{n}.\mathsf{com}() = \mathsf{cm}$ is the *commitment* of the *note* $\mathsf{cm} = \mathsf{Com}(\Gamma || \mathsf{pk}_{\mathsf{u}} || \rho, \mathsf{r})$, computed and appended to the ledger at the creation of the *note*.

Amongst these data, only the *commitment* cm will appears on the ledger. The data $(\Gamma, \mathsf{pk}_{\mathsf{u}}, \rho, \mathsf{r})$ are kept secret by the owner u, so we refer to them as the private data of the note. A *note* $\mathbf{n}$ is attached a *serial number* sn which is computed from the private data of the note and the private address of

[5]It means that an adversary knowing a cipher text $\mathfrak{C}$ and a list of public keys $[\mathsf{pk}_{\mathsf{i}}]_{\mathsf{i}=1}^{\mathsf{n}}$, such that $\mathfrak{C}$ is the encryption of a certain message m from one of these keys pk_{j}, cannot distinguish it amongst the other pk_{i}.

[6]Actually, a participant may use several public-private keys in the same session to appear under different identities.

the participant u as $\mathsf{sn} = \mathsf{PRF}_{\mathsf{sk_u}}(\rho)$. This serial number is computed and appended the ledger at the spending/destruction of $\mathbf{n}$. Because this serial number is derived from a random nonce ρ by a PRF, it is unique to a note. We will see that it prevents double-spending.

The ledger maintains two lists SnList and CmList respectively listing the serial numbers sn and the commitments cm of all notes $\mathbf{n}$ respectively consumed (or spent) and created.

Transactions

A *transaction* from n (non necessary distinct) senders $[\mathsf{i}]_{\mathsf{i}=1}^{\mathsf{n}}$ to m (non necessary distinct) receivers $[\mathsf{j}]_{\mathsf{j}=1}^{\mathsf{m}}$ corresponds to a tuple of data $\mathsf{tx} = ([\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [(\mathsf{cm}_\mathsf{j}^{\mathsf{new}}, \mathfrak{C}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}}, \pi)$ computed and appended to the ledger by the senders, consuming old notes $[\mathbf{n}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}$ respectively owned by $[\mathsf{i}]_{\mathsf{i}=1}^{\mathsf{n}}$ and creating new notes $[\mathbf{n}_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}}$ respectively owned by $[\mathsf{j}]_{\mathsf{j}=1}^{\mathsf{m}}$. More precisely, the computation of a transaction proceeds as follows:

Algorithm 1: $\mathsf{Transaction}([\mathbf{n}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [\mathsf{sk}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [\Gamma_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}}, [\mathsf{pk}_\mathsf{j}]_{\mathsf{j}=1}^{\mathsf{m}}) \rightarrow (\mathsf{tx})$

```
Input: Notes $[\mathbf{n}_\mathsf{i}]_{i=1}^{n} = [(\Gamma_\mathsf{i}, \mathsf{pk_i}, \rho_\mathsf{i}, \mathsf{r_i})]_{\mathsf{i}=1}^{\mathsf{n}}$ to be consumed.
Corresponding private addresses $[\mathsf{sk_i}]_{\mathsf{i}=1}^{\mathsf{n}}$
Public addresses $[\mathsf{pk_j}]_{\mathsf{j}=1}^{\mathsf{m}}$
Output: $\mathsf{tx} = ([\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [(\mathsf{cm}_\mathsf{j}^{\mathsf{new}}, \mathfrak{C}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}}, \pi)$
Run subroutine
  $\mathsf{Transfert}([\mathbf{n}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [\mathsf{sk}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [\Gamma_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}}, [\mathsf{pk_j}]_{\mathsf{j}=1}^{\mathsf{m}})$ :
    Compute $[\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}} = [\mathsf{PRF}_{\mathsf{sk_i}}(\rho_\mathsf{i}^{\mathsf{old}})]_{\mathsf{i}=1}^{\mathsf{n}}$
    Compute $[\rho_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}} = [\mathsf{H}(\mathsf{j}||\mathsf{sn}_1^{\mathsf{old}}||...||\mathsf{sn}_\mathsf{i}^{\mathsf{old}})]_{\mathsf{j}=1}^{\mathsf{m}}$
    Sample randoms $[\mathsf{r}_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}}$
    Compute $[\mathsf{cm}_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}} = [\mathsf{Com}(\Gamma_\mathsf{j}^{\mathsf{new}}||\mathsf{pk_j}||\rho_\mathsf{j}^{\mathsf{new}}, \mathsf{r}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}}$
    Let $[\mathbf{n}_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}} = [(\Gamma_\mathsf{j}^{\mathsf{new}}, \mathsf{pk_j}, \rho_\mathsf{j}^{\mathsf{new}}, \mathsf{r}_\mathsf{j}^{\mathsf{new}}, \mathsf{cm}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}}$
    Sample randoms $[\mathsf{r_j}]_{\mathsf{j}=1}^{\mathsf{m}}$
    Compute $[\mathfrak{C}_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}} = [\mathsf{Enc}_{\mathsf{r_j}}(\mathsf{pk_j}, \mathbf{n}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}}$
    return $([\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [(\mathsf{cm}_\mathsf{j}^{\mathsf{new}}, \mathfrak{C}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}})$
Compute $\mathsf{Prove}(\mathbb{x}, \mathbb{w}) \rightarrow \pi$ with
  Instance: $\mathbb{x} = (\mathsf{CmList}, [\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [(\mathsf{cm}_\mathsf{j}^{\mathsf{new}}, \mathfrak{C}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}})$
  Witness: $\mathbb{w} = ([\mathsf{sk}_\mathsf{i}^{\mathsf{old}}]_{i=1}^{n}, [\mathbf{n}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [\mathbf{n}_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}}, [(\mathsf{r_j}, \mathsf{pk_j})]_{\mathsf{j}=1}^{\mathsf{m}})$
  Statement:
  $\mathsf{tx} = \mathsf{Transfert}([\mathbf{n}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [\mathsf{sk}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [\Gamma_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}}, [\mathsf{pk_j}]_{\mathsf{j}=1}^{\mathsf{m}})$
  $([\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [(\mathsf{cm}_\mathsf{j}^{\mathsf{new}}, \mathfrak{C}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}}) = \mathsf{tx}$
  $\sum_{\mathsf{i}=1}^{\mathsf{n}} \Gamma_\mathsf{i}^{\mathsf{old}} = \sum_{\mathsf{j}=1}^{\mathsf{m}} \Gamma_\mathsf{j}^{\mathsf{new}}$: Balance is preserved (i.e no values are created nor destroyed)
  $[\mathsf{cm}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}} \subseteq \mathsf{CmList}$: Consumed notes have previously been appended to the ledger.
  $[\mathbf{n}_\mathsf{i}^{\mathsf{old}}.\mathsf{addr}()]_{\mathsf{i}=1}^{\mathsf{n}} = [\mathsf{KeyGen}(\mathsf{sk}_\mathsf{i}^{\mathsf{old}})]_{\mathsf{i}=1}^{\mathsf{n}}$: Private addresses correspond to public addresses of consumed notes.
Let $\mathsf{tx} = ([\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [(\mathsf{cm}_\mathsf{j}^{\mathsf{new}}, \mathfrak{C}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}}, \pi)$
return tx
```

Run the algorithm 1 Transaction, obtaining $\mathsf{tx} = ([\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}, [(\mathsf{cm}_\mathsf{j}^{\mathsf{new}}, \mathfrak{C}_\mathsf{j}^{\mathsf{new}})]_{\mathsf{j}=1}^{\mathsf{m}}, \pi)$, and send it to the ledger.

The Ledger checks two conditions:

- the proof π is valid,
- none of the serial numbers $[\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}$ appears in the list SnList (thus avoiding double-spending).

If they are verified, the ledger validates the transaction, adding the *serial numbers* $[\mathsf{sn}_\mathsf{i}^{\mathsf{old}}]_{\mathsf{i}=1}^{\mathsf{n}}$ to SnList, the *commitments* $[\mathsf{cm}_\mathsf{j}^{\mathsf{new}}]_{\mathsf{j}=1}^{\mathsf{m}}$ to CmList and the *transaction* tx to TxList.

Remark that *commitments* are computed from public addresses, and *serial numbers* are computed from private addresses. This is what allows a user A to create notes for a user B without being able to spend it.

C. Main differences with the original Zerocash

First, notes embodied tuple of values Γ corresponding to several goods quantities instead of a unique value in Zerocash and that the sender proof the balance is verified for each good. The second, more subtle difference lies in the use of a public-private key pair as public-private addresses instead of a 256-bit random number and its hash, and in the fact that we require the sender to prove that he has encrypted the note for the recipient with his public key. After all, Zerocash is simply a payment system. So, it does not matter whether a sender did or did not transmit well the notes created during a transaction to the receiver; the receiver can simply refuse the payment. By contrast, at several points of our protocol, it is mandatory for the ledger to be sure that the funds have actually been transferred from senders to receivers preventing malicious behaviors.

IV. Protocol

Our protocol requires the intervention of an auctioneer $\mathcal{T}$ who is considered as an "almost-not" trusted third party (see section II).

We consider the list of participants $\mathcal{P} = [\mathsf{i}]_{\mathsf{i}=1}^{\mathsf{n}}$ with respective provided funds and bids $[\Gamma_\mathsf{i}^{\mathsf{in}}, \mathsf{b_i}]_{\mathsf{i}=1}^{\mathsf{n}}$. We assume that each participant i owns a note $\mathbf{n}_\mathsf{i}^{\mathsf{base}}$ with embodied goods equal to $\Gamma_\mathsf{i}^{\mathsf{in}}$. A participant i can always create such a note for themself by merging some of its other notes containing fractions of the total amount of goods via a standard transaction.

The auctioneer $\mathcal{T}$ has a pair $(\mathsf{pk}_\mathcal{T}, \mathsf{sk}_\mathcal{T})$ of public-private keys. $\mathsf{pk}_\mathcal{T}$ is known by all the participants. The auctioneer $\mathcal{T}$ provides the goods $\Gamma_\mathcal{T}^{\mathsf{in}}$.

To store the public revealed bitstrings info and $\mathsf{info_{bid}}$ output by the exchange function $\mathcal{F}$ and the Leak function at each iteration, a list of public information $\mathsf{InfList}$ is maintained on the ledger.

After a setup step, our protocol progresses in three phases. It starts with a *registration phase* where the participants $[\mathsf{i}]_{\mathsf{i}=1}^{\mathsf{n}}$ have a time delay Δ_1 to manifest their intent to participate in the auction iteration, communicating their provided funds $\Gamma_\mathsf{i}^{\mathsf{in}}$ and bids $\mathsf{b_i}$ to the third party following the modality we will specify. Then, follows the *auction phase* where the auctioneer has a time Δ_2 to operate the auction. Finally, the participants receive their funds in a last *receiving phase*.

A. Protocol

Protocol setup:

In addition to the permanent lists CmList and SnList which are maintained on the ledger, temporary lists $\mathsf{TxList}_{\mathsf{temp}}$, $\mathsf{CmList}_{\mathsf{temp}}$ and $\mathsf{SnList}_{\mathsf{temp}}$ are created and used only during this iteration of auction protocol. To these temporary lists will corresponds special kinds of transactions: the transactions "in" that consume coins of CmList to create new coins in the temporary list $\mathsf{CmList}_{\mathsf{temp}}$ (in which case spending numbers are added to SnList), and the transactions "out" that consume coins of $\mathsf{CmList}_{\mathsf{temp}}$ to create new coins in the ordinary list CmList (in which case spending numbers are added to $\mathsf{SnList}_{\mathsf{temp}}$). A temporary auxiliary list $\mathsf{AuxList}$ is also created.

Registration Phase:

The ledger opens auction at time t_1, allocating special memory emplacement to the auction iteration and lets participants register during a delay Δ_1 specified by the protocol.

Each participant $\mathsf{i} \in \mathcal{P}$ runs the algorithm 2 $\mathsf{Register}$ obtaining $(\mathfrak{C}_{\mathsf{i}}^{\mathsf{Aux}}, \mathsf{tx}_{\mathsf{i}}^{\mathsf{in}}, \mathsf{info}_{\mathsf{bid,i}}, \pi_{\mathsf{reg,i}})$ and sends it to the ledger.

Algorithm 2: Register($\mathsf{n}^{\mathsf{base}}, \Gamma^{\mathsf{in}}, \mathsf{b}) \to (\mathfrak{C}^{\mathsf{Aux}}, \mathsf{tx}^{\mathsf{in}}, \mathsf{info}_{\mathsf{bid}}, \pi_{\mathsf{reg}})$

```
Input: Note $n^{base}$ such that $n^{base}.value() = \Gamma^{in}$. Bid $b$.
Output: Cipher $\mathfrak{C}^{Aux}$. Transaction $tx^{in}$. Public
        information about provided funds and bid
        $info_{bid}$. Proof $\pi_{reg}$.
Generate $sk^{in}$ Compute $pk^{in} = KeyGen(sk^{in})$
Generate $sk^{out}$ Compute $pk^{out} = KeyGen(sk^{out})$
Compute $tx^{in} = Transaction(\mathbf{n}^{base}, sk^{base}, \Gamma^{in}, pk^{in})$
Parse $tx^{in}$ as $(sn^{base}, (cm^{in}, \mathfrak{C}^{in}), \pi)$
Sample $r_{Enc}$ Store $r_{Enc}$
Compute $info_{bid} = Leak(\Gamma^{in}, b)$
Compute $\mathfrak{C}^{Aux} = Enc_{r_{Enc}}(pk_{\mathcal{T}}, (\Gamma^{in}, b, sk^{in}, pk^{out}))$
Compute $Prove(\mathbb{x}, \mathbb{w}) \to \pi_{reg}$ with
    Instance: $\mathbb{x} = (pk_{\mathcal{T}}, \mathfrak{C}^{Aux}, cm^{in}, info_{bid})$
    Witness: $\mathbb{w} = (\mathbf{n}^{in}, b, sk^{in}, pk^{out}, r_{Enc})$
    Statement:
    $\mathfrak{C}^{Aux} = Enc_{r_{Enc}}(pk_{\mathcal{T}}, (\Gamma^{in}, b, sk^{in}, pk^{out}))$
    $info_{bid} = Leak(\Gamma^{in}, b)$
    $\mathbf{n}^{in} = (\Gamma^{in}, pk^{in}, \rho^{in}, r^{in}, cm^{in})$
    $pk^{in} = KeyGen(sk^{in})$
    $cm^{in} = Com(\Gamma^{in} || pk^{in} || \rho^{in}, r^{in})$
return $(\mathfrak{C}^{Aux}, tx^{in}, info_{bid}, \pi_{reg})$
```

The ledger checks that proofs $\pi_{\mathsf{Enc,i}}$ and transactions $\mathsf{tx}_{\mathsf{i}}^{\mathsf{in}}$ are valid. If these conditions are satisfied, then the ledger realizes "in" transactions appending $\mathsf{tx}_{\mathsf{i}}^{\mathsf{in}}$ to $\mathsf{TxList}_{\mathsf{temp}}$, $\mathsf{sn}_{\mathsf{i}}^{\mathsf{base}}$ to the permanent SnList and $\mathsf{cm}_{\mathsf{i}}^{\mathsf{in}}$ to the temporary $\mathsf{CmList}_{\mathsf{temp}}$. It also adds $\mathsf{info}_{\mathsf{bid,i}}$ to $\mathsf{InfList}$ and $(\mathfrak{C}_{\mathsf{i}}^{\mathsf{Aux}}, \pi_{\mathsf{reg,i}})$ to $\mathsf{AuxList}$.

The ledger announces the closing of the registration phase at time $\mathsf{t}_2 = \mathsf{t}_1 + \Delta_1$, preventing any further alteration to $\mathsf{TxList}_{\mathsf{temp}}$, $\mathsf{CmList}_{\mathsf{temp}}$ or $\mathsf{AuxList}$.

Auction Phase:

The ledger opens the auction phase at time t_2 and lets the trusted third party $\mathcal{T}$ compute the auction for a delay Δ_2 specified by the protocol during which no "out" transaction can be appended to the ledger excepts if it comes from the trusted third party $\mathcal{T}$ and has the form we will soon specify.

$\mathcal{T}$ extracts the list of ciphered notes $[\mathfrak{C}_{\mathsf{i}}^{\mathsf{in}}]_{\mathsf{i}=1}^{\mathsf{n}}$ from $\mathsf{TxList}_{\mathsf{temp}}$, and the corresponding list of ciphered bids, input secret key and output public key $[\mathfrak{C}_{\mathsf{i}}^{\mathsf{Aux}}]_{\mathsf{i}=1}^{\mathsf{n}}$ from $\mathsf{AuxList}$.

$\mathcal{T}$ runs the algorithm 3 $\mathsf{Exchange}_{\mathcal{F}}$ obtaining $(\mathsf{tx}^{\mathsf{out}}, \mathsf{info}, \pi_{\mathcal{F}})$ and sends it to the ledger.

Algorithm 3: $\mathsf{Exchange}_{\mathcal{F}}([\mathfrak{C}_{\mathsf{i}}^{\mathsf{in}}]_{\mathsf{i}=1}^{\mathsf{n}}, [\mathfrak{C}_{\mathsf{i}}^{\mathsf{Aux}}]_{\mathsf{i}=1}^{\mathsf{n}}, \mathsf{sk}_{\mathcal{T}}) \to (\mathsf{tx}^{\mathsf{out}}, \mathsf{info}, \pi_{\mathcal{F}})$

```
Input: Ciphered notes $[\mathfrak{C}_i^{in}]_{i=1}^n$. Ciphered bids
       $[\mathfrak{C}_i^{Aux}]_{i=1}^n$. Secret key $sk_{\mathcal{T}}$.
Output: Transaction $tx^{out}$. Public info info. Proof $\pi_{\mathcal{F}}$.
Compute $[Dec(sk_{\mathcal{T}}, \mathfrak{C}_i^{Aux})]_{i=1}^n = [(\Gamma_i^{in}, b_i, sk_i^{in}, pk_i^{out})]_{i=1}^n$
Compute $[Dec(sk_i^{in}, \mathfrak{C}_i^{in})]_{i=1}^n = [\mathbf{n}_i^{in}]_{i=1}^n$
Compute $(\Gamma_{\mathcal{T}}^{out}, [\Gamma_i^{out}]_{i=1}^n, info) = \mathcal{F}(\Gamma_{\mathcal{T}}^{in}, [\Gamma_i^{in}, b_i]_{i=1}^n)$.
Compute
  $tx^{out} = Transaction([\mathbf{n}_i^{in}]_{i=1}^n, [sk_i^{in}]_{i=1}^n, [\Gamma_i^{out}]_{i=1}^n, [pk_i^{out}]_{i=1}^n)$
Parse $tx^{out}$ as $([sn_i^{in}]_{i=1}^n, [cm_i^{out}, \mathfrak{C}_i^{out}]_{i=1}^n, \pi)$
Compute $Prove(\mathbb{x}, \mathbb{w}) \to \pi_{\mathcal{F}}$ with
    Instance: $\mathbb{x} = ([sn_i^{in}]_{i=1}^n, [cm_i^{out}]_{i=1}^n, info, AuxList = [\mathfrak{C}_i]_{i=1}^n)$
    Witness: $\mathbb{w} = ([\mathbf{n}_i^{in}]_{i=1}^n, [\mathbf{n}_i^{out}]_{i=1}^n, sk_{\mathcal{T}})$
    Statement:
    $[(\Gamma^{in}, b_i, sk_i^{in}, pk_i^{out})]_{i=1}^n = [Dec(sk_{\mathcal{T}}, \mathfrak{C}_i)]_{i=1}^n$
    $[\mathbf{n}_i^{in}]_{i=1}^n = [(\Gamma_i^{in}, pk_i^{in}, \rho_i^{in}, r_i^{in}, cm_i^{in})]_{i=1}^n$
    $[sn_i^{in}]_{i=1}^n = [PRF_{sk_i^{in}}(\rho_i^{in})]_{i=1}^n$
    $(\Gamma_{\mathcal{T}}^{out}, [\Gamma_i^{out}]_{i=1}^n, info) = \mathcal{F}(\Gamma_{\mathcal{T}}^{in}, [\Gamma_i^{in}, b_i]_{i=1}^n)$
    $[\mathbf{n}_i^{out}]_{i=1}^n = [(\Gamma_i^{out}, pk_i^{out}, \rho_i^{out}, r_i^{out}, cm_i^{out})]_{i=1}^n$
    $[cm_i^{out}]_{i=1}^n = [Com(\Gamma_i^{out} || pk_i^{out} || \rho_i^{out}, r_i^{out})]_{i=1}^n$
return $(tx^{out}, info, \pi_{\mathcal{F}})$
```

The ledger checks proof $\pi_{\mathcal{F}}$ and transaction $\mathsf{tx}^{\mathsf{out}}$. If these conditions are satisfied, then the ledger realizes an "out" transaction appending the serial numbers $[\mathsf{sn}_{\mathsf{i}}^{\mathsf{in}}]_{\mathsf{i}=1}^{\mathsf{n}}$ to $\mathsf{SnList}_{\mathsf{temp}}$ and the commitments $[\mathsf{cm}_{\mathsf{i}}^{\mathsf{out}}]_{\mathsf{i}=1}^{\mathsf{n}}$ to CmList. It also adds info to $\mathsf{InfList}$ and $\pi_{\mathcal{F}}$ to $\mathsf{AuxList}$.

The ledger announces the closing of the auction phase at time $\mathsf{t}_3 = \mathsf{t}_2 + \Delta_2$

Receiving Phase:

At t_3 there are two cases: either the auctioneer performed the exchange during the auction phase or they failed to do it sending no message or incorrect message to the ledger.

- *Case 1: $\mathcal{T}$ correctly performed the exchange.* Each participant reads $\mathsf{TxList}_{\mathsf{temp}}$ and adds their received funds for $\mathsf{pk}^{\mathsf{out}}$ to their permanent wallet using $\mathsf{sk}^{\mathsf{out}}$.
- *Case 2: $\mathcal{T}$ failed to perform the exchange.* Each participant i withdraws their engaged funds by running $\mathsf{Withdraw}$, obtaining $(\mathsf{tx}_{\mathsf{draw,i}}^{\mathsf{out}}, \pi_{\mathsf{draw,i}})$ and sends it to the ledger.

Algorithm 4: $\mathsf{Withdraw}(\mathsf{CmList}_{\mathsf{temp}}, \mathsf{r}_{\mathsf{enc}}, \mathsf{n}^{\mathsf{in}}, \mathsf{pk}_{\mathcal{T}}, \mathsf{sk}^{\mathsf{in}}, \mathfrak{C}_{\mathsf{i}}) \to (\mathsf{tx}^{\mathsf{out}}_{\mathsf{draw}}, \pi_{\mathsf{draw}})$

```
Input: $\mathsf{CmList}_{\mathsf{temp}}$. Randomness $\mathsf{r}_{\mathsf{enc}}$. Notes $\mathsf{n}^{\mathsf{in}}$.
       Auctioneer's public key $\mathsf{pk}_{\mathcal{T}}$. Secret key $\mathsf{sk}^{\mathsf{in}}$.
Output: Transaction $\mathsf{tx}^{\mathsf{out}}_{\mathsf{draw}}$. Proof $\pi_{\mathsf{draw}}$.
Compute $\mathsf{tx}^{\mathsf{out}}_{\mathsf{draw}} = \mathsf{Transaction}(\mathsf{n}^{\mathsf{in}}, \mathsf{sk}^{\mathsf{in}}, \Gamma^{\mathsf{in}}, \mathsf{pk}^{\mathsf{out}})$
Parse $\mathsf{tx}^{\mathsf{out}}_{\mathsf{draw}}$ as $(\mathsf{sn}^{\mathsf{in}}, (\mathsf{cm}^{\mathsf{out}}, \mathfrak{C}^{\mathsf{out}}), \pi)$
Compute $\mathsf{Prove}(\mathbb{x}, \mathbb{w}) \to \pi_{\mathsf{draw}}$ with
    Instance: $\mathbb{x} = (\mathsf{sn}^{\mathsf{in}}_{\mathsf{i}}, \mathsf{cm}^{\mathsf{out}}_{\mathsf{i}}, \mathsf{pk}_{\mathcal{T}}, \mathfrak{C}_{\mathsf{i}})$
    Witness: $\mathbb{w} = (\mathbf{n}^{\mathsf{in}}_{\mathsf{i}}, \mathbf{n}^{\mathsf{out}}_{\mathsf{i}}, \mathsf{sk}^{\mathsf{in}}_{\mathsf{i}}, \mathsf{b}_{\mathsf{i}}, \mathsf{r}_{\mathsf{Enc}})$
    Statement: $\mathfrak{C}_{\mathsf{i}} = \mathsf{Enc}_{\mathsf{r}_{\mathsf{Enc}}}(\mathsf{pk}_{\mathcal{T}}, (\mathsf{b}_{\mathsf{i}}, \mathsf{sk}^{\mathsf{in}}_{\mathsf{i}}, \mathsf{pk}^{\mathsf{out}}_{\mathsf{i}}))$
    $\mathbf{n}^{\mathsf{out}}_{\mathsf{i}} = (\Gamma^{\mathsf{out}}, \mathsf{pk}^{\mathsf{out}}, \rho^{\mathsf{out}}, \mathsf{r}^{\mathsf{out}}, \mathsf{cm}^{\mathsf{out}})$
    $\mathsf{cm}^{\mathsf{out}} = \mathsf{Com}(\Gamma^{\mathsf{out}} || \mathsf{pk}^{\mathsf{out}} || \rho^{\mathsf{out}}, \mathsf{r}^{\mathsf{out}})$
    $\mathsf{sn}^{\mathsf{in}} = \mathsf{PRF}_{\mathsf{sk}^{\mathsf{in}}}(\mathbf{n}^{\mathsf{in}}.\mathsf{seed}())$
return $(\mathsf{tx}^{\mathsf{out}}_{\mathsf{draw}}, \pi_{\mathsf{draw}})$
```

Fig. 1: Caption

The ledger checks the validity of $\mathsf{tx}^{\mathsf{out}}_{\mathsf{draw}}$ and π_{draw}. If both are valid, it appends $\mathsf{tx}^{\mathsf{out}}_{\mathsf{draw}}$ to $\mathsf{TxList}_{\mathsf{temp}}$, $\mathsf{sn}^{\mathsf{in}}$ to $\mathsf{SnList}_{\mathsf{temp}}$ and $\mathsf{cm}^{\mathsf{out}}$ to CmList.

Discussion:

The proof π_{reg} in registration phase guarantees the participant sent usable funds and bid to the auctioneer. The proof $\pi_{\mathcal{F}}$ guarantees the auctioneer computed well the exchange result and use the requested $\mathsf{pk}^{\mathsf{out}}_{\mathsf{i}}$ to send founds. π_{draw} guarantees that the fund are withdrawn by the correct participant on their requested $\mathsf{pk}^{\mathsf{out}}_{\mathsf{i}}$ using $\mathsf{sk}^{\mathsf{in}}$. Participants need to store $\mathsf{r}_{\mathsf{enc}}$ generated during the register phase to be able compute this latter proof in case the auctioneer misbehaved[7].

The only known privacy weakness our protocol concedes is that the auctioneer, while not learning directly the identity of participants due to the use of one time temporary addresses, can learn about bids and final outcomes of participants. One way to overcome this problem is for participants to split their bids and funds in several ones running Register several times, at least if the exchange function $\mathcal{F}$ allows it (e.g. is linear).

V. Application to Energy Market with Double Auction

In this section, we briefly explain how our protocol can be integrated in the context of microgrids and local energy market with P2P exchanges between prosumers.

A. Description

We consider a microgrid with distinct prosumers sharing a distributed public ledger upon which is implemented a Zerocash-like infrastructure. A Distribution System Operator (DSO) manages the microgrid via smart meters having access to energy consumption and production of each prosumer in real time. We assume that each prosumer smart meter can communicate both with its owner and with the ledger, but our protocol can be easily adapted to communication with the ledger only, at the price of higher computational requirement from the smart meter [8].

Energy units are represented on-chain by fungible energy tokens. To a token is associated a certain amount of Wh, which can vary with time, location, energy origin and so on.

Each prosumer i has a pair $(\mathsf{pk}_{\mathsf{i}}, \mathsf{sk}_{\mathsf{i}})$ of public-private keys. Each of their respective smart meters has a pair $(\mathsf{pk}_{\mathsf{DSO,i}}, \mathsf{sk}_{\mathsf{DSO,i}})$ of public-private keys.

A delay δ is fixed such that at every time step, the smart meter of i reports anonymously its energy production to the ledger. To do so, it creates a note $\mathbf{n}^{\mathsf{pro}} = (\Gamma^{\mathsf{pro}}, \mathsf{pk}_{\mathsf{i}}, \mathsf{r}^{\mathsf{pro}}, \rho^{\mathsf{pro}}, \mathsf{cm}^{\mathsf{pro}})$ with corresponding amount of energy token Γ^{pro} (in particular, if i has not produced energy at all, Γ^{pro} has 0 energy token), produces a signature σ of $\mathsf{cm}^{\mathsf{pro}}$ with its private key $\mathsf{sk}_{\mathsf{DSO,i}}$ and sends $(\mathsf{cm}^{\mathsf{pro}}, \sigma)$ to the ledger. The smart meter also sends $\mathsf{n}^{\mathsf{pro}}$ to i.

The ledger checks whether σ is valid before appending $\mathsf{cm}^{\mathsf{pro}}$ to CmList.

When a prosumer consumes energy, they redeems the corresponding amount of energy tokens contained in some of their notes $[\mathbf{n}^{\mathsf{con}}_{\mathsf{j}}]^{\mathsf{m}}_{\mathsf{j}=1}$ to the DSO's smart meter via a transaction $\mathsf{tx}^{\mathsf{con}} = \mathsf{Transaction}([\mathbf{n}^{\mathsf{con}}_{\mathsf{j}}]^{\mathsf{m}}_{\mathsf{j}=1}, \mathsf{sk}_{\mathsf{i}}, \Gamma^{\mathsf{con}}, \mathsf{pk}_{\mathsf{DSO,i}})$ appended to the ledger.

The SBExM mechanism corresponding to double auction classically used in energy market is specified as follows:

Participants or prosumers are either energy sellers or buyers. The set of good's type T has two elements $\{\$, \odot\}$ for coins and energy tokens. Provided funds and bids $(\Gamma^{\mathsf{in}}, \mathsf{b})$ correspond respectively to engaged energy tokens to be sold and minimum asked price per unit, or to engaged coins and maximum proposed price per unit. The Leak function is given by $\mathsf{Leak}(\Gamma, \mathsf{b}) \to \mathsf{b}$. The exchange function $\mathcal{F}$ determines the price p where offer and demand meet and exchanges the goods provided by the sellers asking price $\mathsf{b} < \mathsf{p}$ and buyers proposing price $\mathsf{b} > \mathsf{p}$ (see [1] for details). We also impose that $\mathcal{F}$ outputs the selling price p as info.

Load profiles measured by smart meters are never centralized by the DSO. The only way for an attacker to access consumption (or production) of i is to know the secret key $\mathsf{sk}_{\mathsf{DSO,i}}$ stored in their smart meter (or their secret key sk_{i}).

However, our architecture still allows the prosumer and/or their smart meter to selectively transmit any required information (eventually adding some noise in data) about their load profile to any third party (such as DSO) having access to ledger, convincing this third party that the transmitted data are correct.

To do so, we need that the commitments appended to the ledger be timestamped. If the ledger is maintained as a

[7]It is necessary that the participant i proves they are using $\mathbf{n}^{\mathsf{in}}_{\mathsf{i}}$ to create $\mathbf{n}^{\mathsf{out}}_{\mathsf{i}}$, otherwise a malicious auctioneer $\mathcal{T}$ could use the funds of one participant in the withdrawal of another one, preventing the first to retrieve their funds.

[8]On the other hand, supposing smart meter has only one communication (i.e with the ledger) reduces the attack surface.

blockchain, we can use the timestamp of the block where they first appear. Additionally, we assume that the prosumer (or their smart meter) stores all the received/emitted notes $\mathbf{n}^{\mathsf{pro}}$ and $\mathbf{n}^{\mathsf{con}}$. Load profile Load_i can be recovered from the tokens $\Gamma^{\mathsf{pro/con}}$ stored in these notes and the timestamp associated to corresponding commitments $\mathsf{cm}^{\mathsf{pro/con}}$. From this, the function $\mathcal{G} : \mathsf{Load} \rightarrow \mathsf{data}$ whose input is the load profile and outputs are the information data to be revealed is computed. The prosumer (or their smart meter) also computes a zero-knowledge proof $\pi_{\mathcal{G}}$, having for instance the timestamped CmList and the output data_i, for witness the notes $\mathbf{n}^{\mathsf{pro/con}}$ and for statement $\mathsf{data}_\mathsf{i} = \mathcal{G}(\mathsf{Load}_\mathsf{i})$. data_i and $\pi_{\mathcal{G}}$ are transmitted to the third party, which simply verifies the proof.

B. Implementation

1) Methodology: The performance of the system is predominantly influenced by the proofs, which means their optimization directly impacts the overall efficiency and effectiveness of the protocol. Consequently, the implementation [9] focuses on these core proofs ($\pi_{\mathsf{draw}}, \pi_{\mathcal{F}}, \pi_{\mathsf{reg}}$ and π_{Tx}). To achieve this, we used Gnark [18], an open source library written in Go.

The implementation was executed on an Ubuntu virtual machine equipped with 128GB of RAM and powered by an AMD EPYC Milan processor, clocked at 2.45GHz.

We instantiated both $\mathsf{PRF}_\mathsf{r}(\mathsf{x})$ and $\mathsf{Com}(\mathsf{x}, \mathsf{r})$ as $\mathsf{H}(\mathsf{x}||\mathsf{r})$ with the hash function MiMC, an hash function specifically designed to be efficient in ZKP applications. The encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ employed is based on the Diffie-Hellman key exchange protocol, followed by an arithmetic one-time pad for ensuring secure message transmission. This combination provides strong security guarantees while maintaining computational efficiency.

The zero-knowledge proof system used in our implementation is Groth16, a widely-adopted ZKP construction due to its concise proof sizes and efficient verification. In our case, it has been instantiated with BW6_761, a pairing-friendly elliptic curve specifically designed to offer optimal performance. The counterpart of Groth16 is that a trusted third party has to generate a trusted setup at initialisation. In case of corruption an adversary could generate valid proofs for false statements, so this task cannot be handled by the auctioneer that we want to keep almost-not trusted.

2) Results: The figure 1 and 2 highlight respectively the time required for proof generation and proof verification for the four types of proofs used in our protocol. The costs of all the other computational steps of algorithm executed by participant and auctioneer, such that computation of hashes and encryptions, are negligible compared to proof generation. We also focused on proof verification because this operation is performed by all nodes maintaining the ledger, not only by participants of the exchange.

[9]https://github.com/HamzaZF/Privacy-Preserving-Exchange-Mechanism

Algorithm Register (resp Withdraw) run by the participants needs one execution of $\mathsf{Prove} \rightarrow \pi_{\mathsf{reg}}$ (resp $\mathsf{Prove} \rightarrow \pi_{\mathsf{draw}}$) and one execution of $\mathsf{Prove} \rightarrow \pi_{\mathsf{tx}}$, which can be done in less than 2 seconds. Moreover, it only takes a few milliseconds to verify it. This is due to the fact that Groth16 verification is independent of the circuit size.

The proofs consists in three elliptic curve points, each one of size 768 bits in compressed form. The cipher $\mathfrak{C}^{\mathsf{Aux}}$ produced in encryption phase has a size of four field elements of size 377 bits plus one elliptic curve point of size 384[10], commitments and spending numbers both have a size of 377 bits, and cipher produced by transaction algorithms has also a size of 4*377+384 bits. So in total, not taking into account the potential information intentionally made public by the chosen exchange mechanism, running Register requires storing 8762bits on the public ledger, $\mathsf{Exchange}_{\mathcal{F}}$ $(n * 2262 + 4608)$bits, and Whitdraw 6870bits.

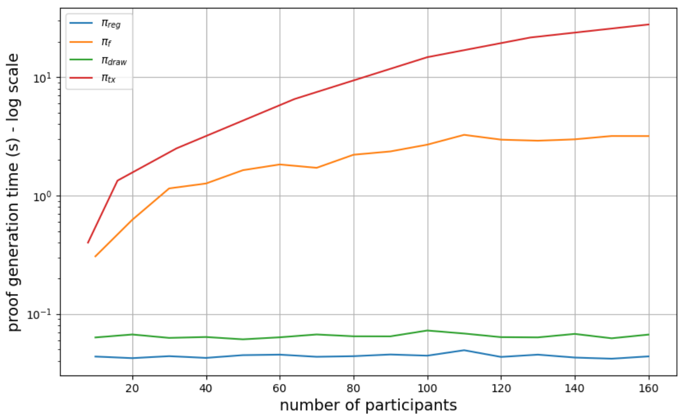

Fig. 2: Execution time of $\mathsf{Prove}(\mathbb{x}, \mathbb{w}) \rightarrow \pi$ for $\pi_{\mathsf{draw}}, \pi_{\mathcal{F}}, \pi_{\mathsf{reg}}$ and π_{Tx} as the number of participant grows.

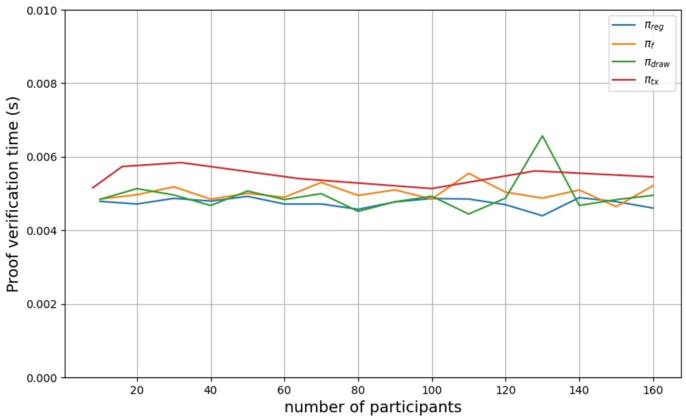

Fig. 3: Verification time of $\mathsf{Verify}(\pi, \mathbb{x}) \rightarrow \mathsf{b} \in \{0, 1\}$ for $\pi_{\mathsf{draw}}, \pi_{\mathcal{F}}, \pi_{\mathsf{reg}}$ and π_{Tx} as the number of participant grows.

3) Discussion: Consistent proof verification times and the efficiency of the proof generation process show that our solu-

[10]Elliptic curve points used by Groth16 are not the same size as those used by the encryption scheme.

tion is well suited to large-scale environments, and can therefore be applied in a practical context. Indeed, for smart meters, the proof generation time is independent of the number of participants (i.e. n=1), which means that large-scale adoption can be envisaged without compromising system performance.

On the other hand, the auctioneer must generate proofs that are dependent on the number of participants, and must therefore have sufficient computing power to fulfill its role as this number grows.

An auction between ten participants without auctioneer failure would require little less than 4.5kB of storage on public ledger. While being costfull for a distributed ledger and orders of magnitude higher than a naive solution without privacy, this is still reasonable in practice.

VI. Conclusion

We proposed a general privacy-preserving exchange mechanism for the energy market in the context of microgrid deployment.

The flexibility and high level of privacy of our application is due to the development of a general anonymous sealed-bid exchange mechanism protocol based on our Zerocash-like infrastructure introducing a notion of "almost-not" trusted auctioneer.

Looking ahead, optimizing the computational efficiency of our protocol will be crucial in scaling its application to more complex and larger smart grid architectures. Such improvements will enable us to support larger networks with minimal impact on performance, ensuring that privacy is maintained without compromising scalability. This dual focus on privacy and efficiency positions our mechanism as a viable solution for the future of decentralized energy markets, fostering both trust and innovation.

References

[1] J. Horta, D. Kofman, D. Menga, and A. Silva, "Novel market approach for locally balancing renewable energy production and flexible demand," in *2017 IEEE International Conference on Smart Grid Communications, SmartGridComm*. IEEE, 2017, pp. 533–539.

[2] J. Horta, E. Altman, M. Caujolle, D. Kofman, and D. Menga, "Real-time enforcement of local energy market transactions respecting distribution grid constraints," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2018, pp. 1–7.

[3] M. Foti and M. Vavalis, "Blockchain based uniform price double auctions for energy markets," *Applied Energy*, vol. 254, p. 113604, 2019.

[4] C. Zhang, T. Yang, and Y. Wang, "Peer-to-peer energy trading in a microgrid based on iterative double auction and blockchain," *Sustainable Energy, Grids and Networks*, vol. 27, p. 100524, 2021.

[5] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "*Zerocash: Decentralized Anonymous Payments from Bitcoin*," Cryptology ePrint Archive, Paper 2014/349, 2014.

[6] X. Jia, L. Wang, K. Cheng, P. Jing, and X. Song, "A blockchain-based privacy-preserving and collusion-resistant scheme (ppcr) for double auctions," *Digital Communications and Networks*, 2023.

[7] G. Sharma, D. Verstraeten, V. Saraswat, J.-M. Dricot, and O. Markowitch, "Anonymous fair auction on blockchain," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2021, pp. 1–5.

[8] L. Massoni Sguerra, P. Jouvelot, F. Coelho, E. j. Gallego Arias, and G. Memmi, "The price of smart contract privacy," *Submited to ACM DLT journal*, 2024.

[9] R. Sarenche, M. Salmasizadeh, M. H. Ameri, and M. R. Aref, "A secure and privacy-preserving protocol for holding double auctions in smart grid," *Information Sciences*, vol. 557, pp. 108–129, 2021.

[10] M. Wen, R. Lu, J. Lei, H. Li, X. Liang, and X. S. Shen, "Sesa: an efficient searchable encryption scheme for auction in emerging smart grid marketing," *Security and Communication Networks*, vol. 7, no. 1, pp. 234–244, 2014.

[11] K. Gai, Y. Wu, L. Zhu, M. Qiu, and M. Shen, "Privacy-preserving energy trading using consortium blockchain in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3548–3558, 2019.

[12] S. Zhang, M. Pu, B. Wang, and B. Dong, "A privacy protection scheme of microgrid direct electricity transaction based on consortium blockchain and continuous double auction," *IEEE Access*, vol. 7, pp. 151 746–151 753, 2019.

[13] D. Li, Q. Yang, W. Yu, D. An, X. Yang, and W. Zhao, "A strategy-proof privacy-preserving double auction mechanism for electrical vehicles demand response in microgrids," in *International Performance Computing and Communications Conference*, 2017, pp. 1–8.

[14] F. Reid and M. Harrigan, *An Analysis of Anonymity in the Bitcoin System*. New York, NY: Springer New York, 2013, pp. 197–223. [Online]. Available: https://doi.org/10.1007/978-1-4614-4139-7_10

[15] G. Oded, *Foundations of Cryptography: Volume 2, Basic Applications*, 1st ed. USA: Cambridge University Press, 2009.

[16] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, "Key-privacy in public-key encryption," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001, pp. 566–582.

[17] O. Goldreich, *Foundations of Cryptography*. Cambridge University Press, 2001, vol. 1.

[18] [Online]. Available: https://docs.gnark.consensys.io/