Privacy-Preserving Smart Grid Tariff Decisions with Blockchain-Based Smart Contracts

Fabian Knirsch, Andreas Unterweger, Günther Eibl and Dominik Engel

Abstract The smart grid changes the way how energy and information are exchanged and offers opportunities for incentive-based load balancing. For instance, customers may shift the time of energy consumption of household appliances in exchange for a cheaper energy tariff. This paves the path towards a full range of modular tariffs and dynamic pricing that incorporate the overall grid capacity as well as individual customer demands. This also allows customers to frequently switch within a variety of tariffs from different utility providers based on individual energy consumption and provision forecasts. For automated tariff decisions it is desirable to have a tool that assists in choosing the optimum tariff based on a prediction of individual energy need and production. However, the revelation of individual load patterns for smart grid applications poses severe privacy threats for customers as analyzed in depth in literature. Similarly, accurate and fine-grained regional load forecasts are sensitive business information of utility providers that are not supposed to be released publicly. This paper extends previous work in the domain of privacy-preserving load profile matching where load profiles from utility providers and load profile forecasts from customers are transformed in a distance-preserving embedding in order to find a matching tariff. The embeddings neither reveal individual contributions of customers, nor those of utility providers. Prior work requires a dedicated entity that needs to be trustworthy at least to some extent for determining the matches. In this paper we propose an adaption of this protocol, where we use blockchains and smart contracts for this matching process, instead. Blockchains are gaining widespread adaption in the smart grid domain as a powerful tool for public commitments and accountable calculations. While the use of a decentralized and trust-free blockchain for this protocol comes at the price of some privacy degradation (for which a mitigation is outlined), this drawback is outweighed for it enables verifiability, reliability and transparency.

Fabian Knirsch, Andreas Unterweger, Günther Eibl and Dominik Engel
Salzburg University of Applied Sciences, Josef Ressel Center for User-Centric Smart Grid Privacy, Security and Control, Urstein Süd 1, 5412 Puch bei Hallein, Austria. e-mail: fabian.knirsch@en-trust.at

1 Introduction

A global trend towards the modernization of energy grids is ongoing: Information and communication technologies are integrated into the energy grid infrastructures, creating so-called "smart grids". A multitude of new use cases become possible, including the balancing of power generation and consumption, electric mobility, renewable energy sources and real-time pricing. The modernization on the technical side is accompanied by a move towards deregulation on the regulatory side [1], thereby vastly increasing the number of choices on the side of the end consumer. In liberalized energy markets, the paradigm for the customers will be changed: They will be able to change tariff, provider or both frequently and to adapt dynamically. There are a number of pilot regions that have shown the benefit of this deregulation, e.g., the German *eEnergy* projects, most notably the *MeRegio* project [1].

Customers have to choose one tariff out of those offered from their local utilities. In order to select the optimal tariff, i.e., the one fitting their electricity consumption at a given time, customers need to match their current usage habits to the offered pricing schemes. As an example, consider an energy provider which is interested in moving energy consumption away from times with high demand because this decreases the amount of immediately available but expensive energy resources. In order to encourage customers to use energy at off-peak times, the provider could offer better tariffs for load profiles that are dissimilar to normal consumption profiles by, e.g., lowering a fixed-price tariff from 23 cents/kWh to 17 cents/kWh. As another example, in order to stimulate people charging their cars during the night instead of during the day, a load profile with high consumption during night might be associated with another cheap tariff. A convenient way to perform the matching of current and desired consumption profiles is for the customer to provide a current load profile, i.e., a recent energy usage record, to different energy providers, to a third party, or a blockchain-based smart contract providing a matchmaking service for tariffs offered by various energy providers.

However, while this is a convenient approach, there are privacy concerns: It has been shown that personal information, such as lifestyle, religion, habitual patterns, sleep-wake-cycles and activities can be extracted from load profiles (e.g., [2, 3]). On the side of the energy providers, there is also reluctance to disclose the load profiles that different tariffs are based on (e.g., typical load profiles of customer groups), as this is internal information that, in a liberated market, can make a difference in commercial success [1]. Therefore, while both, the consumer side and the provider side, are interested in facilitating optimal matching, the aforementioned are strong reasons against disclosing full information for the matchmaking process.

In [4], a method for tariff selection has been presented that preserves both, consumer privacy and the internal company data of energy providers. This is achieved by the use of embeddings where load profiles are transformed into a domain that does not allow to reconstruct the original load profiles, but preserves the distances. The method outperforms previously suggested methods. In particular, it limits data expansion, which is an issue when classical homomorphic encryption is used. The trade-off is matching accuracy, which in this method is not 100%. Depending on

choice of parameters, higher accuracy can be traded for increased data expansion. However, in real-word application scenarios, the accuracy of 93.5%, which can be realized with the proposed approach at negligible data expansion, is sufficient.

In this paper, we propose an adaption of the approach in [4]. Our proposed extension of this approach uses blockchains and smart contracts [5, 6]. While the original approach requires a third party for determining the minimum distance and the optimum match, respectively, we propose omitting the third party and replacing it with a fully decentralized and trust-less[1] network. Smart contracts allow every peer – customers and utility providers – to calculate and verify the results. This improves the existing solution in terms of verifiability, reliability and transparency.

The possible use cases of the proposed protocol are manifold. An exemplary use case would be for a number of utilities to (continually) analyze and cluster the energy usage data of their customers. Based on this analysis, different tariff groups can be created. Consumers who wish to select the tariff which is best suited to their current consumption habits can create a smart contract with their load profiles and commit it to a blockchain where both, the consumers and the utilities are peers. The utilities then commit a number of load profiles that represent tariff options and the smart contract evaluates the best-matching tariff. The result will be verifiable by all parties.

Note that during the whole process, no load information is disclosed by either the consumer or the utilities, i.e., the other peers cannot access the consumer data as it may reveal sensitive information about them [7, 8]. The tariff selection is then done decentralized and trust-less by executing the smart contract in the blockchain.

In summary, this paper extends the previously presented approach in [4] based on nearest neighbor embeddings [9] and oblivious transfer [10] for finding the best-matching tariff without directly revealing any load profiles to any involved party. This paper contributes (i) a fully decentralized and trust-less approach for tariff-matching; and (ii) an exemplary implementation as a smart contract. The proposed approach is designed for the preservation of privacy, i.e., as in [4], privacy relies on the security of embeddings [9]. While the privacy guarantees of the proposed approach are not as strong as the ones in [4], a way to mitigate this is outlined.

The rest of this paper is structured as follows: In Section 2, related work from the domains of secure distance computation, blockchains and smart contracts and other related approaches is discussed. Section 3 introduces the preliminaries such as design goals, the original profile matching protocol with a third party and blockchain-based smart contracts. In Section 4 the decentralized and trust-less protocol for tariff-matching based on a blockchain is presented in detail. Section 5 evaluates the proposed protocol, presents a conceptual implementation of the smart contract and compares the proposed protocol to related work. Section 6 summarizes this work and gives an outlook to future research.

[1] The trust-lessness of blockchains relies on the assumption that at least half of the computing power is spent by honest peers [5], as will be described in Section 3.3.1.

2 Related work

In this section, we provide an overview of related work. We distinguish between secure distance computation methods which are suitable to compare load profiles, existing work related to blockchains and smart contracts and other related work including oblivious transfer and smart-grid literature with tariff selection and profile matching.

2.1 Secure Distance Computation

The following related work describes secure distance computation algorithms. In our approach, we compare load profiles using Euclidean distance measures, which is a similar type of computation.

Mukherjee et al. [11] propose a method where a number of selected Fourier transform coefficients are permuted and communicated between the involved parties. The properties of the Fourier transform are used to provide limited guarantees on distance preservation based on the selection of transform coefficients. This makes their approach probabilistic with the number of coefficients retained providing a tradeoff between privacy and accuracy. Although our approach is probabilistic as well, the level of privacy is not influenced by the parameters which affect the accuracy of our approach.

Ravikumar et al. [12] describe an algorithm for secure Euclidean distance computation. Their approach is probabilistic as well, but requires a high number of vectors to be compared in order to come close to the actual distance values. In contrast, our approach finds the minimum distance in nearly all cases, allowing for near-perfect matching.

Wong et al. [13] introduce transformations of database points and query points that enable a ranking of the database points with respect to their nearness to the query points suitable for k-nearest-neighbor determination. However, their transformations are not distance-preserving which enhances security against attackers with known plaintexts, but limits possible applications.

Rane et al. [9] and Boufounos et al. [14] propose the use of distance-preserving embeddings for privacy-preserving matching and nearest-neighbor searches in the context of image retrieval. We apply these distance-preserving embeddings as one step in our proposed protocol and make use of their privacy-preserving properties.

Homomorphic cryptosystems can be used for secure distance computation [15, 16], e.g., in the context of image retrieval [17], fingerprint matching [18] and face recognition [19]. Kolesnikov et al. [20] combines homomorphic encryption for computing distances with Garbled circuits for choosing the point having the minimum distance to the query point. We provide a detailed comparison between our original approach, the approach based on blockchains and additive homomorphic cryptosystems in Section 5.3.

2.2 Blockchains and Smart Contracts

For the proposed protocol, blockchains are used as a distributed, public and permanent platform for deciding on an optimum tariff and storing this decision reproducibly. A blockchain as a public ledger for coin-based transactions is originally proposed in [5] for *Bitcoin*. Since then, a wide range of applications have been established that are based on that technology (e.g., [21, 22, 23, 6]). Some approaches improve the principles of Bitcoin in terms of privacy ([21]), while others propose a turing-complete blockchain for arbitrary calculations ([23, 6]).

In [22], the authors propose a protocol that uses a blockchain as an access-control manager that does not require a third party. This allows to reduce trust into a single entity, similarly to the approach proposed in this paper.

In [6], the author presents a turing-complete blockchain that can be used to build decentralized applications. All inputs and states are public in this approach, which limits privacy. In [23], the authors propose a similar concept that preserves user privacy by the use of zero-knowledge proofs.

2.3 Other Related Work

In our proposed approach, we make use of oblivious transfer [10, 24, 25]. It allows one of two parties to query one of an arbitrary number of items from the second party without revealing (i) which item has been requested; and (ii) any of the other items. A detailed description of the use within our protocol is provided in Section 3.2.

From a use case point of view, apart from oblivious transfer, related work includes literature on tariff decisions, load profile matching and forecasting as well as demand-response and demand-side-management. For the latter two, Palensky and Dietrich [26] provide an overview. In general, privacy concerns and communication overhead on the smart meter side are seldomly addressed in protocols for demand side management and tariff-matching. We focus on these privacy and communication overhead aspects by providing some examples below.

Caron and Kesidis [27] describe an approach where customers share load profile information so that the utility can achieve a smoother aggregate load profile. This is also possible with the approach proposed in our paper. However, the load profiles are not revealed to the utility, thus preserving the privacy of the customers.

Similarly, Shao et al. [28] attempt to reduce peak loads by incentivizing customers to shift time of use of electrical devices. This behavior can also be triggered when applying our approach to provide suitable template load profiles, albeit not in real-time. Again, the load profiles do not need to be shared with the utility as opposed to the approach by Shao et al.

Ramchurn et al. [29] follow a game-theoretic approach with customer incentives for reducing peak loads. They use a decentralized protocol, as opposed to our as well as to Caron's and Kesidis's [27] and Shao et al.'s [28] approach. The approach by

Ramchurn et al. defers certain loads with defined probability imposing constraints on the customer's choices. In contrast, our approach gives the customer full authority on tariff decisions.

Another game-theoretic approach for shifting energy consumption is proposed by Mohsenian-Rad et al. [30]. In this setting, the smart meters of the users interact in order to minimize overall energy consumption. Their approach requires peer-to-peer communication with transmission overhead depending on the number of smart meters involved. Conversely, our approach does not require any peer-to-peer communication whatsoever.

3 Preliminaries

This section introduces the preliminaries for the proposed protocol that uses blockchains and smart contracts for the matching process. First, the previously proposed protocol [4] that uses a dedicated third party for the matching process in presented. Second, the concept of blockchains and smart contracts are introduced as a decentralized and trust-less alternative for the original protocol.

The notation used throughout this paper is summarized in Table 1. $\mathbf{F}$ denotes a load profile forecast that is used by the consumer to represent the load pattern and $\mathbf{T}$ denotes a template load profile that is linked to a tariff from the utility providers. The embedding is explained in the next section and is denoted as $\tilde{\mathbf{F}}$ and $\tilde{\mathbf{T}}$, respectively.

U	utility provider
SM	smart meter
TP	third party
$\mathbf{F}$	load profile forecast
$\tilde{\mathbf{F}}$	embedded load profile forecast
$\mathbf{T}$	template load profile
$\tilde{\mathbf{T}}$	embedded template load profile
T	tariff
$\mathscr{U}$	set of utilities
$\mathscr{L}_u$	set of tariffs from utility u
(u^*, l^*)	index of best-matching utility/tariff
$E(\cdot), D(\cdot)$	encryption and decryption functions
$r \leftarrow_{\$} \mathbb{Z}$	sampling of a random number from $\mathbb{Z}$

Table 1 Overview of notation used in this paper.

3.1 Requirements

The objective of this paper is to rely on a fully decentralized and trust-less architecture for tariff-matching. For the matching process and the performed calculations, we define the following requirements:

1. **Transparency.** All calculations should be transparent to both, the utility providers and the customers. This prevents an adversary from forging matching results (i.e., a binding to committed values) and it also prevents the utility providers from later changing their template load profiles retroactively.
2. **Verifiability.** Calculations and the results, i.e., the optimum matching should be verifiable by all participants. This allows the customer and the utility providers to verify at any time that the found tariff has indeed a minimum distance with the submitted parameters.
3. **Reliability.** The matching should not rely on a single party. All participants should be able to perform the matching and therefore increase the fault-tolerance of this approach by decentralizing this crucial step in the protocol. In the original paper [4], the protocol requires a dedicated third party that acts as a broker for the matching.
4. **Privacy.** While fulfilling the first three requirements, the calculation should still be privacy preserving in accordance to the following privacy definition.

Privacy Definition: A protocol is privacy-preserving if none of the participants learns more than a particular, predefined function of the input data [31]. The proposed protocol is privacy preserving, if none of the participating entities has access to the original load profiles and only the minimum distance between the customers' load profile forecast and the template load profiles of the utility provider is revealed. At the end of the protocol, the smart meter only learns the tariff associated to a template load profile that has the minimum distance to the load profile forecast.

3.2 Profile Matching Protocol

This section is a summary of our previous protocol [4] which allows one smart meter to find an optimum tariff based on its load profile forecast. A number of template load profiles corresponding to tariffs from different utilities are used for this search. Throughout the process, none of the involved parties has access to the others' data. The process is therefore privacy-preserving. Note that the following protocol still makes use of the third party for finding the optimum match. It will later be shown how to replace this with a blockchain and smart contracts, respectively.

Figure 1 illustrates the different steps of our protocol which are described in detail below. The protocol is split into three phases, namely *initialization*, *matching* and *oblivious transfer*.

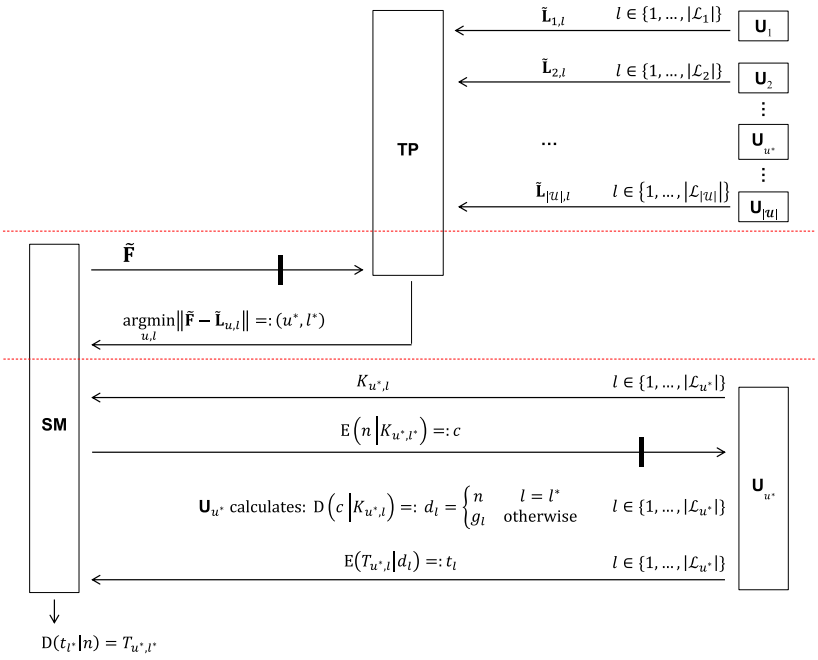

Fig. 1 Overview of the proposed protocol. All participating utilities $\mathbf{U}_u \in \mathscr{U}$ provide embedded template load profiles $\tilde{\mathbf{L}}_u$ corresponding to their tariffs $T_{u,l}$ to a third party **TP**. A smart meter **SM** sends its embedded load profile forecast $\tilde{\mathbf{F}}$ to this third party which returns the best-matching utility index u^* and its tariff index l^*. Subsequently, the smart meter can request information about the best-matching tariff T_{u^*,l^*} from the best-matching utility $\mathbf{U}_{u^*}$ through oblivious transfer. Black vertical bars indicate rate limiters.

3.2.1 Initialization

Let the set of participating utilities be denoted as $\mathscr{U}$. Each utility $\mathbf{U}_u$, $u \in \{1,\dots,|\mathscr{U}|\}$ (cf. Figure 1, right) has a list of tariffs $T_{u,l}$ with corresponding template load profiles $\mathbf{L}_{u,l}$ (denoted as set $\mathscr{L}_u$ in Figure 1 where utilities can have different numbers of load profiles $l \in \{1,\dots,|\mathscr{L}_u|\}$). For example, utility $\mathbf{U}_1$ has a standard tariff $T_{1,1}$ for day workers and a night-owl tariff $T_{1,2}$ for night workers. The corresponding template load profiles $L_{1,1}$ and $L_{1,2}$ show peak loads at different times of the day opposite to the respective working hours.

Template load profiles allow each utility to control demand and response in a fine-grained manner, e.g., by rewarding customers with atypical load profiles so that peak loads are avoided. Although the tariffs need to be known to the customers for billing and transparency, the template load profiles are considered to be private to the respective utility. Therefore, no details are shared with competing utilities. In practice, template load profiles and tariffs may be significantly more complex than the example described above, with privacy on the utility side being a much more pressing issue.

In order to keep the template load profiles private, each utility calculates an embedding $\tilde{\mathbf{L}}_{u,l} \in \{0,1\}^m$ for each of its original template load profiles $\mathbf{L}_{u,l} \in \mathbb{R}^k$ as the first step of the *initialization* phase:

$$\tilde{\mathbf{L}}_{u,l} = \left\lceil \frac{\mathbf{A} \cdot \mathbf{L}_{u,l} + \mathbf{W}}{\Delta} \right\rceil \mod 2 \tag{1}$$

$\mathbf{A}$ is a random $m \times k$ matrix with i.i.d. Gaussian elements with mean 0 and variance σ^2, and $\mathbf{W}$ is a random m-dimensional vector with i.i.d. uniform elements in the range $[0, \Delta]$. Δ is both, a quantization and a security parameter, and described in detail in [9, 14].

This embedding does not allow a potential attacker to reconstruct the original load profile, but preserves the distance between load profiles within a very small margin of error as described below. This enables comparisons of load profiles without the need to handle the respective original, private data.

The second step of the *initialization* phase of our protocol requires each utility to send all of its calculated embeddings $\tilde{\mathbf{L}}_{u,l}$ to a third party, denoted as **TP**. The need for this third party will become clear in the subsequent *matching* phase.

3.2.2 Matching

In this phase, a smart meter, denoted as **SM**, first creates a load profile forecast $\mathbf{F}$. It can either be based on past load profiles, e.g., of the current day or week, or on user input, e.g., prospective changes in work schedules. Similar to each utility in the preceding *initialization* phase, the smart meter first calculates an embedding of $\mathbf{F}$, denoted as $\tilde{\mathbf{F}}$. This way, the smart meter does not need to disclose its original load profile which may reveal sensitive information about the user.

As a second step, the embedding is sent to the third party, like the embeddings from the utilities in the previous phase. As a third step, the third party finds the best match for the load profile forecast out of the list of template load profiles from all utilities through the received embeddings. More precisely, it finds the template load profile with the smallest normalized Hamming distance to the forecast and outputs the template load profile index l^* as well as the corresponding utility index u^*, i.e.,

$$(u^*, l^*) = \underset{u,l}{\operatorname{argmin}} ||\tilde{\mathbf{F}} - \tilde{\mathbf{L}}_{u,l}||_1. \tag{2}$$

This is possible due to the distance-preserving property of the embeddings (as described in more detail in [9]), where the Euclidean distance of the original data vectors is proportional to the normalized Hamming distance of the embedded vectors with a configurable small error ε, i.e.,

$$||\tilde{\mathbf{F}} - \tilde{\mathbf{L}}_{u,l}||_1 \sim ||\mathbf{F} - \mathbf{L}_{u,l}||_2 + \varepsilon. \tag{3}$$

As a consequence, the probability that the best match in the original space and the best match in the embedded space coincide, is close to one:

$$\Pr\left[\underset{u,l}{\operatorname{argmin}}\,||\tilde{\mathbf{F}}-\tilde{\mathbf{L}}_{u,l}||_1 = \underset{u,l}{\operatorname{argmin}}\,||\mathbf{F}-\mathbf{L}_{u,l}||_2\right] = 1-\delta. \tag{4}$$

This probability is referred to as matching accuracy. The smaller ε is, the higher the accuracy is.

The result (u^*,l^*) of the matching operation is a pair of indices identifying the utility $\mathbf{U}_{u^*}$ of the best match and its tariff T_{l^*}. However, no information about any load profile is revealed. The tuple (u^*,l^*), is transmitted to the smart meter as a fourth and final step, allowing the smart meter to fetch the tariff information from the utility $\mathbf{U}_{u^*}$ directly in the next phase in order not to disclose the actual tariff T_{l^*} associated with the index l^*.

The third party needs to be involved in the calculation above since neither the smart meter nor any of the utilities can be completely trusted to correctly perform calculations on the data. In addition, malicious parties are considered, i.e., they could manipulate their own input to bias the result in their favor or they could use multiple different inputs to derive additional information about the other parties' data. This is avoided by the use of an independent third party with a rate limiter (vertical black bars in Figure 1) which prevents bulk-probing from the other parties.

The third party can be thought of as a neutral party which performs only computations, e.g., a proxy of the Council of European Energy Regulators (CEER) which strives for a fair tariff market and competition. However, since any party may be distrusted, including the third party, the latter is not allowed to perform calculations on the actual data, but on embeddings only. This is why the latter need to be calculated by the other parties. This way, the third party is not able to access the original load profiles of either the smart meter or the utilities.

Note that the third party may collect statistics on the matching results (i.e., the indices (u^*,l^*)) of all smart meters. However, this can be rendered futile if the utilities regularly shuffle their template load profiles' indices in the *initialization* phase e.g. each day. For example, $(u^*,l^*)=(1,1)$ means a standard tariff and $(u^*,l^*)=(1,2)$ a night-owl tariff, respectively, on one day, and the other way around, i.e., $(u^*,l^*)=(1,1)$ means a night-owl tariff and $(u^*,l^*)=(1,2)$ a standard tariff, respectively, on the next day.

Note that the third party could be omitted when using verifiable computing [32, 33]. However, this would induce substantial overhead which is critical on a device with limited capabilities, such as smart meters. In the subsequent *oblivious transfer* phase, the third party is not involved at all and hence never has access to the tariff T_{u^*,l^*} itself. Therefore, the third party only needs to be trusted to perform the correct calculation int the *matching* phase. This remaining level of trust will be replaced by a decentralized trust-less architecture for the blockchain-based protocol.

3.2.3 Oblivious Transfer

In the last phase of the embedding-based protocol, the smart meter sends a query to the utility $\mathbf{U}_{u^*}$ in order to obtain the best-matching tariff T_{l^*} based on its index l^*. The third party is not involved in this transaction and does therefore not obtain any information about the tariff itself apart from its index.

At this stage, the customer has not yet made the decision whether or not to switch to the best-matching tariff – they still need the tariff information for this. Thus, on the one hand, the smart meter must not disclose l^* to the utility since it would allow the utility to deduce information about the original load profile, even when the customer chooses not to switch to the matching tariff. On the other hand, the utility does not want to disclose all tariffs, some of which may exclusively be available to certain customers or groups. It only wants to disclose the tariff corresponding to the index l^*, but without being allowed to know this very index.

A solution for this is oblivious transfer [10, 24, 25]. It allows the smart meter to retrieve a tariff l^* from a vector of tariffs $T_{u^*,l}$, without the query (index) being known to the utility $\mathbf{U}_{u^*}$ and without any other tariffs being disclosed to the smart meter apart from T_{u,l^*}. In our use case, communication with $\mathbf{U}_{u^*}$ yields the tariff T_{u^*,l^*}.

The steps of the *oblivious transfer* phase can be summarized as follows: Initially, i.e., before any other communication, the utility $\mathbf{U}_{u^*}$ sends one key $K_{u^*,l}$ per template load profile $\mathbf{L}_{u^*,l}$ to the smart meter. The number of template load profiles is identical to the number of tariffs as described above. Thus, in total $|\mathscr{L}_{u^*}|$ keys are sent.

Secondly, the smart meter generates a nonce n which is encrypted with the $(l^*)^{\text{th}}$ key. The encrypted nonce, c, is then sent to the utility. This step requires a rate limitation (e.g., one query per day) on the utility's side since the smart meter could otherwise query all available tariffs by iterating through the available indices. The rate limit has to be chosen such that the maximum number of allowed queries is lower than or equal to the average frequency at which utilities update their tariffs. If, for instance, utilities update their template load profiles daily, a rate limit of one query per day is sufficient.

Thirdly, the utility decrypts c with all of its keys, yielding decryptions d_l. For the key with the index sent by the smart meter, the decryption yields the nonce n, while, for all other keys, the decryption result is a garbage value g_l. For the utility, however, these are indistinguishable from the nonce. Thus, the utility cannot find out the index l^*.

Fourthly, the utility encrypts all tariffs $T_{u^*,l}$ with the decryption results d_l as keys, i.e., the nonce known to the smart meter for the index l^* and garbage values g_l for the others. The encrypted tariffs, t_l, are then sent to the smart meter which decrypts the $(l^*)^{\text{th}}$ encrypted tariff with the previously generated nonce. This yields the desired tariff T_{u^*,l^*}. The other tariffs cannot be decrypted since the encryption and decryption keys do not match, thus do not leak any information to the smart meter.

3.3 Blockchains and Smart Contracts

This section introduces blockchains and the concept of smart contracts. Furthermore, an exemplary smart contract is presented, the application of a decentralized, trust-less architecture is motivated and privacy issues are discussed.

3.3.1 Overview

After having originally been proposed in [5], blockchains are gaining an increased adaption in many fields, e.g., [22, 23]. A blockchain is a trust-less and fully decentralized peer-to-peer system that is designed to hold immutable information once data is committed to the chain. Generally, a blockchain can therefore be described as a distributed, immutable database.

In the originally proposed Bitcoin protocol from [5] the blockchain is used to keep track of *coins*, i.e., a public list of how much coins are owned by each peer. Therefore, each block contains sender and receiver information, as well as the amount of coins to be transferred. This is called a *transaction* and – once confirmed by the peers – appends a new block to the chain that also includes the hash of the previous block and is therefore permanently linked to a series of previous transactions.

The public list of chained blocks can be verified by all peers in the network by checking the integrity of the new block and the correct calculation of the hash, respectively. While the generation of valid blocks consumes a considerable amount of computing power, this is also referred to as the *proof of work*. In order to prevent the tampering with already created blocks, all peers in the network agree on the longest valid chain. A chain is valid if it is verified by other peers. Accordingly, a block and transaction can be considered valid if it is followed by a sufficient amount of other valid blocks. Therefore, the blockchains is trust-less if at least half of the computing power for the proof of work is spent by honest peers [5].

In order to prevent the tampering with already created blocks, all peers in the network agree on the longest valid chain. A chain is valid if it is verified by other peers. Accordingly, a block and transaction can be considered valid if it is followed by a sufficient amount of other valid blocks. Therefore, the blockchains is trust-less if at least half of the computing power for the proof of work is spent by honest peers [5].

Note that at all times all states are publicly available and can be verified by all peers by simply checking the hashes from the very first block (also referred to as the genesis block) up to the last block. Peers in the network are identified by a private-public key pair (identifier or address).

3.3.2 Smart Contracts

The Bitcoin protocol is designed for a particular purpose. However, blockchains can be used to store arbitrary information. Recently, the application of blockchains for smart contracts has been proposed [23, 6, 34]. Ethereum[2], for instance, is a platform for executing smart contracts on a turing-complete virtual machine, the Ethereum Virtual Machine (EVM), where computing results are stored in a public blockchain. Similarly to the Bitcoin protocol, in Ethereum, a peer-to-peer network stores a decentralized ledger with state information. These states, however, include details about executed smart contracts. A smart contract is a program that can send and receive messages and execute some logic. For executing a contract, i.e., creating a new block, the miner is paid with *Ether*. There are a number of newly developed program languages that compile to EVM bytecode, such as Solidity[3] that resembles a JavaScript-like syntax.

Smart contracts can be deployed by every peer in the network. Once committed to the blockchain, a smart contract is identified by a unique address and can be called by other peers. The code of a smart contracts needs to be public as well, since all peers in the network must be capable of verifying the computation results. Smart contracts therefore expose one or more methods and hold state information in internal variables. As a simple example which is commonly used to demonstrate a smart contract (compare to, e.g., [23]) consider a rock-paper-scissors game as in Algorithm 1. Here, a smart contract is created that accepts an input from two players ("rock", "paper" or "scissors") and determines a winner. The notation for algorithms in this paper is summarized in Table 2. Assume this algorithm is deployed in the blockchain and is assigned a unique identifier or address. Players willing to bet can use this address and call the `commit` method in order to send their bet.

The first parameter of type ID is passed automatically as an argument to all methods and refers to the address of the caller. This is a concept that is also found in real-world implementations. For the algorithms in this paper this is made explicit for clarity. The address is an alpha-numeric value of approximately 40 characters, depending on the concrete protocol that is derived from the public key of a peer.

ID	data type holding an address in the blockchain
Hash	data type holding a hash (e.g., SHA-2)
HashMap<U,T>	data type mapping elements of type U to elements of type T
$\tilde{\mathbf{E}}$	data type holding an embedding
$\tilde{\mathbf{E}}[]$	data type holding an array of embeddings
$\oslash$	value representing null or undefined

Table 2 Overview of notation used in algorithms in this paper.

[2] https://www.ethereum.org/

[3] https://solidity.readthedocs.io/en/develop/

```
ID player1 = ⊘;
ID player2 = ⊘;
Hash hash1 = ⊘;
Hash hash2 = ⊘;
string bet1 = ⊘;
string bet2 = ⊘;

begin commit (ID sender, Hash hash)
    if player1 == ⊘ then
        player1 = sender;
        hash1 = hash;
    else if player2 == ⊘ then
        player2 = sender;
        hash2 = hash;
    else
        return "wait for next game";
end

begin open (ID sender, string bet, int r)
    if sender == player1 ∧ H(bet|r) == hash1 then
        bet1 = bet;
    else if sender == player2 ∧ H(bet|r) == hash2 then
        bet2 = bet;
    else
        return "invalid commitment";
end

begin evaluate (ID sender)
    if ¬ (bet1 == ⊘ ∨ bet2 == ⊘) then
        if bet1 == "rock" ∧ bet2 == "scissors" then
            return player1 + " wins";
        else if bet1 == "paper" ∧ bet2 == "scissors" then
            return player2 + " wins";
        else
            ... /* test for other cases */
    else
        return "commit bets and open commitment first";
    end
end
```

Algorithm 1: Smart contract for playing a rock-paper-scissors game. Both players send a commitment for their bet and once all bets are received, the players open their commitment and the winner can be determined by calling the `evaluate` method.

In order to prevent any player from learning the other players' value, a computationally hiding commitment scheme is used (see [35]). Instead of only sending a string *bet*, player i, $i \in \{1,2\}$, sends the hash of the string and a random number $r_i \leftarrow_{\$} \mathbb{Z}$, i.e., $h_i := H(\text{bet}_i|r_i)$ with $H(\cdot)$ being a collision-resistant universal hash function $H : \{0,1\}^x \to \{0,1\}^y$, $x, y \in \mathbb{Z}^+$, with, e.g., $y = 256$, and $\text{bet}_i|r_i$ denotes the concatenation of the bet and the random number. Due to the one-way property of the hash function and the random number which is only known to the player, the public

information in the blockchain does not reveal the bet, but only the hash. Once all players sent their commitments, the game can be evaluated. To do so, the players open their commitment by sending (bet_i, r_i) and the smart contract verifies whether $H(\text{bet}_i|r_i) = h_i$. If the hash of the commitment and the hash of the values bet_i and r_i are equal, the commitment is opened. Otherwise, the commitment is invalid and the winner cannot be determined until the commitment is opened correctly. The message flows for player i and the smart contract are shown in Figure 2. Note that there also exist information-theoretically secure binding schemes such as Pedersen commitments [36], which are out of scope for this work.

While such commitment schemes prevent other players from learning the bets during the game, they do not provide forward secrecy once the commitments are opened and the game is finished. This would, for instance, allow an adversary to learn the bets from a player over time[4]. A more advanced approach is to use zero-knowledge proofs, e.g., as presented in [23], where the authors present a blockchain model for smart contracts that allows for private parts that are never revealed publicly.

3.3.3 Summary

Generally, blockchains such as Bitcoin and blockchain-based smart contract implementations such as Ethereum are not privacy-preserving in the sense that the information remains undisclosed. In contrast, all the information committed to the blockchain is publicly available to all peers. Usually, the ownership of coins and other data is claimed by a private key that has been randomly generated once and does not allow to link to any individual. However, this kind of pseudonymity is not a strong privacy guarantee, as sender, receiver and the amount of data as well as state information are public. In order to privately process data in a blockchain, there are recent trends that strengthen the privacy, e.g., [21, 23].

In summary, blockchains offer the ability to provide a decentralized database that does not require trusting other peers with certain limits as discussed in Section 3.3.1. A consensus on the state of the database is found by a set of decentralized executed rules and proof-of-work mechanisms. Blockchains are therefore particularly suitable for minimizing or removing the role of a trusted or semi-trusted third party, while at the same time providing full transparency to all participants. However, in order to achieve privacy, blockchains need more sophisticated approaches, e.g., building on commitments or zero-knowledge proofs. It is later shown that, for preserving forward-secrecy in the proposed protocol, embeddings and commitments are not sufficient and stronger privacy guarantees as proposed in [23] are required.

[4] A player could change address for every game providing a means of pseudonymity for this use case. However, for the tariff matching, as shown later, we need to reveal the actual "players" at some point for the oblivious transfer which requires more sophisticated approches.

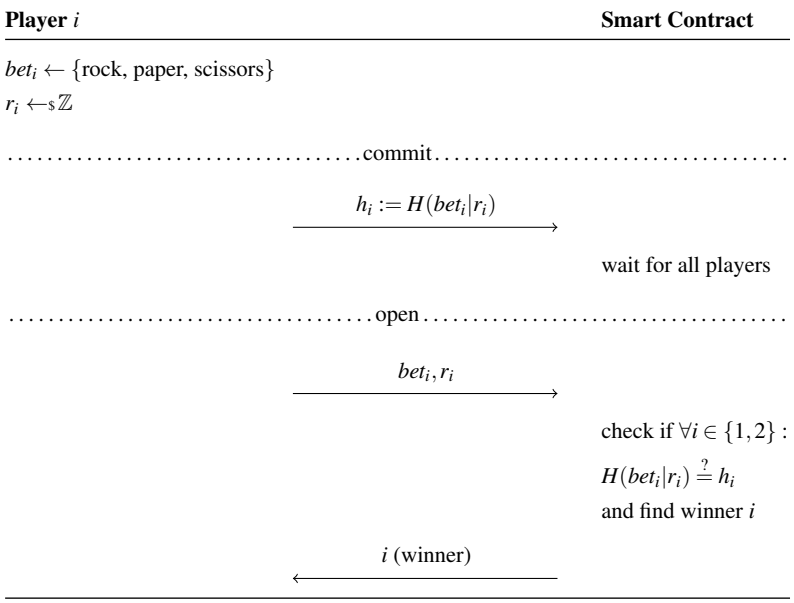

Fig. 2 Message flows for the rock-paper-scissors game for player i, $i \in \{1,2\}$, and the smart contract with a computationally hiding commitment scheme. First, each player chooses a bet and a random number, where $r \leftarrow_{\$} \mathbb{Z}$ denotes the drawing of the random number. Both values are hashed and committed. Second, once all players sent their values, the commitments are opened. If the values are correct, the winner is determined.

3.4 Assumptions

For the following protocol description, we define a number of properties for our blockchain-based profile matching protocol. First, a preliminary protocol is introduced that fulfills already three out of our four initial requirements: (i) Transparency; (iii) Verifiability; and (ii) Reliability. The fourth requirement, privacy, is only partly fulfilled. While the template load profiles and the load profile forecasts are embedded and do not allow for immediate recovery of the original data, all this information is publicly and permanently available and stored in the blockchain, i.e., forward secrecy is not fulfilled. An adversary could query the blockchain for previous requests and use this information to brute-force load profiles. The feasibility of brute-forcing embedded data is briefly discussed in [4].

In order to fulfill our privacy requirement regarding forward-secrecy, a way needs to be established that allows to calculate the matching in the blockchain while at the same time not revealing the inputs.

For the preliminary protocol, we assume a blockchain that allows to execute smart contracts (e.g., Ethereum). Therefore, all parties, i.e., $\mathbf{SM}$, $\mathbf{U}_u$ with $u \in \{1,\ldots,|\mathscr{U}|\}$ are peers in the blockchain and are thus able to execute smart contracts.

For the improved protocol we additionally assume the blockchain to be capable of privacy-preserving calculations (e.g., HAWK [23]).

4 Protocol Description

Figure 3 illustrates the different steps of our adaption of the original protocol for blockchains. The protocol can again be split into three phases, namely *initialization*, *matching* and *oblivious transfer*, with the *matching* phase being further split into a *commitment* and an *opening* phase (separated by red dotted lines). In the *initialization* phase, the same embedding approach as in the previous protocol is applied. In this phase, load curves are transformed into an embedding that does not allow to retrieve the original load profile, but preserves the ability to calculate distances. In contrast to the original protocol, the matching phase is handled within a public blockchain instead of a dedicated third party. The *oblivious transfer* phase remains the same as in the previous protocol and is therefore not discussed in detail in this section.

4.1 Initialization

Each utility $\mathbf{U}_u,\ u \in \{1,\ldots,|\mathscr{U}|\}$ (cf. Figure 3, right) has a list $\mathscr{L}_u$ of tariffs $T_{u,l}$ with corresponding template load profiles $\mathbf{L}_{u,l}$. As in [4], utilities can have different numbers of load profiles $l \in \{1,\ldots,|\mathscr{L}_u|\}$).

For each template load profile $\mathbf{L}_{u,l}$, its embedding $\tilde{\mathbf{L}}_{u,l} \in \{0,1\}^m$ is calculated by $\mathbf{U}_u$ as specified by Equation 1. This way, in the subsequent *matching* phase, utilities do not need to submit the plain template load profiles, but only an embedding for privacy reasons. Similarly, **SM** calculates the embedding $\tilde{\mathbf{F}}$ of its load profile forecast $\mathbf{F}$ and only sends $\tilde{\mathbf{F}}$ in the *matching* phase.

4.2 Matching

In contrast to the original paper, the *matching* phase is completely redesigned, fully decentralized and trust-less based on a blockchain. Using a blockchain and a smart contract for determining the distance poses some advantages over a dedicated entity **TP**, which corresponds to the initially stated requirements:

1. **Transparency.** Once a template load profile $\tilde{\mathbf{L}}_{u,l}$ or a load profile forecast $\tilde{\mathbf{F}}$ is written to the blockchain, this information is public. Every peer can view the data and the data cannot be altered by anyone after being encoded in the chain.
2. **Verifiability.** The calculation performed by the smart contract, i.e., the result of the calculation which in this case is the optimum match, is written immutably to

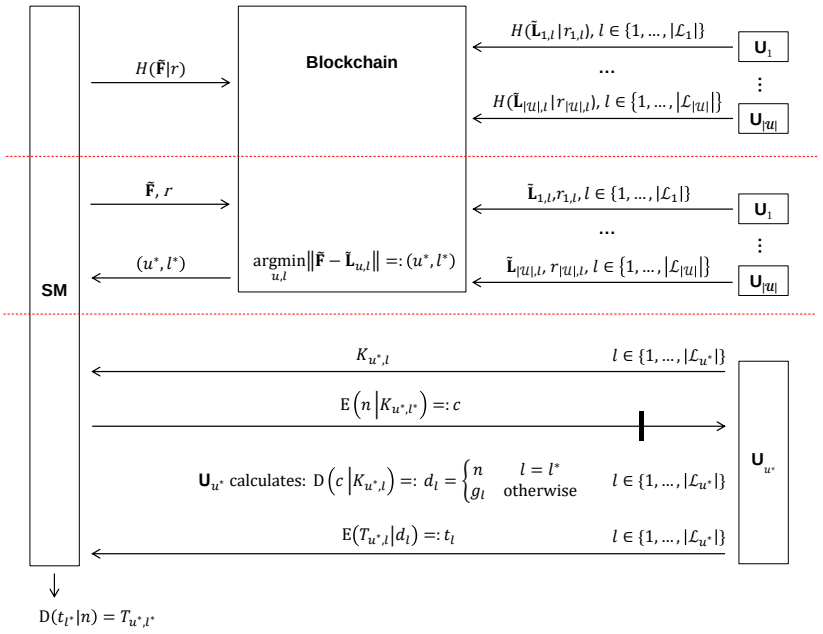

Fig. 3 Overview of the proposed protocol. After the *initialization* phase (not shown), all participating parties commit their load profiles to the block chain as the first part of the *matching* phase. In the second part, all parties open their commitments publicly in the blockchain. At the end of the *matching* phase, the established smart contract returns the best-matching utility index u^* and its tariff index l^*. The smart meter can then request information about the best-matching tariff through oblivious transfer as proposed in [4].

the blockchain as well. Therefore, every peer can later verify the chain of blocks and whether the calculated result is actually the best match. In particular, this also holds for the customer.

3. **Reliability.** Smart contracts that are based on blockchains are decentralized by design. There is no single entity that is responsible for performing the calculations, but every peer can calculate and verify results. This makes blockchains more reliable, compared to centralized architectures.

The *matching* phase that only needs to compute the hamming distance of the embedded load forecast and the embedded template load profile (as shown in Equation 2) is written as a smart contract. This smart contract is created by the smart meter **SM** which prepared an embedded load profile forecast $\tilde{\mathbf{F}}$ in the *initialization* phase and now wants to find an optimum tariff. The address in the blockchain of the smart meter and the load profile forecast are the initialization parameters for this smart contract. The contract is then bound to this particular smart meter. The binding assures that no other peer can trigger the evaluation method, i.e., the initial creator remains in control of the contract.

Once this smart contract is created, utility providers $\mathbf{U}_u$, $u \in \{1,\ldots,|\mathscr{U}|\}$ can commit their embedded template load profiles $\tilde{\mathbf{L}}_{u,l}$, with $l \in \{1,\ldots,|\mathscr{L}_u|\}$ by sending $H(\tilde{\mathbf{L}}_{u,l}|r_{u,l})$, where $r_{u,l}$ is the random number for the commitment as described in Section 3.3.2. The collection of load profiles can be realized by providing a public method that accepts an array of hashed template load profiles as a commitment from each utility in the *commitment* part (top part of Figure 3). As soon as a smart contract is established, the commitment of the load profile forecast from the customer as well as any committed template load profiles from the utility providers cannot be changed or overwritten in this particular instance. However, if a new smart contract is created, both, customer and utility providers can commit their load profiles again and the protocol starts over. The smart contract is open and accepts the commitments from the utility providers as long as the commitments are not opened.

If the smart meter received enough tariff options or after a certain amount of time has passed, **SM** and all $\mathbf{U}_u$ can trigger the opening method, thereby starting the *opening* part (middle part of Figure 3). If the commitments are opened correctly, the actual embedded load profiles are stored in the smart contract, **SM** triggers the evaluation method and the best-matching tariff (u^*,l^*) is returned. This also closes the current instance of the smart contract. Note that the index u^* of the utility corresponds to the **ID** or address of the utility in the blockchain.

4.3 Oblivious Transfer

The *oblivious transfer* phase does not deviate from the originally proposed protocol and is described in Section 3.2. The blockchain is only used for finding the minimum distance to the utility/tariff that matches best, i.e., (u^*,l^*). The smart meter then initiates the oblivious transfer in order to receive the desired tariff T_{u^*,l^*}.

5 Evaluation

In this section, we evaluate privacy-preserving load profile matching. For the evaluation we introduce an implementation for the smart contract that is independent of concrete programming languages or platforms. We then discuss the privacy and security properties of the blockchain-based approach and finally, the proposed approach is compared to related work.

5.1 Implementation

The following algorithms, Algorithm 2 and Algorithm 3, show how to implement a smart contract that performs tariff matching. Note that this is an implementation to

show the concept and that this code still has some practical limitations as discussed later.

While the following algorithm is independent of concrete smart contract programing languages (e.g., Solidity, Hawk), it follows common design principles [35] and can be easily turned into a program in a specific language. The smart contract consists of five global variables and five methods that implement the creation and evaluation of the smart contract as well as the ability to receive and verify commitments. The smart contract uses basic data types such as double precision floating-point types (*double*) and integer types (*int*), more complex data types such as *HashMap* and Arrays (denoted by square brackets), specific data types such as **ID** (storing an address of a node in the blockchain), $\tilde{\mathbf{E}}$ (storing an embedding, which is discussed in detail later) and **Hash** (storing a SHA-2 hash). The principle of this smart contract is similar to the implementation presented in Algorithm 1 for the rock-paper-scissors game. All parties first send hashed values as a commitment and then open their commitments, which is verified in the smart contract by comparing the hashes. Instead of determining a winner as in the game, the optimum tariff based on the minimum distance is found.

In the following, the methods for the smart contract are described in detail:

- `create`. This method is called upon the initial creation of the smart contract. The method expects two arguments, *sender* and a hashed value *loadforecast*. The first is the creator of the smart contract, i.e., the smart meter that wants to perform a tariff matching. The latter is a commitment for an embedding of the load profile forecast. Both values are used for initializing the smart contract. The commitment is calculated by the smart meter by hashing the embedded load profile forecast and a random number.
- `commit`. Once the smart contract is initialized, it can be found by utilities that want to offer a tariff. These utilities then call the *commit* method which expects two arguments, *sender*, which is the address of the utility, and *loadprofiles*, which is an array of hashed embedded template load profiles. Utilities can only once send an array of an arbitrary number of hashed template load profiles. This assures that utilities cannot change or revoke committed data for this instance of the smart contract. The commitment is calculated by the utilities by hashing the embedded template load profiles and a random number for each template load profile.
- `smopen`. The smart meter can open its commitment by sending the embedded load profile forecast and the random number that was used for calculating the hash. The smart contract verifies if the commitment is valid and stores the embedded load profile forecast for the evaluation.
- `uopen`. Similarly to the smart meter, utilities can open their commitments by sending the embedded template load profiles and the random numbers that were used for calculating the hashes. The smart contract verifies if the commitment is valid and stores the template load profiles for the evaluation.
- `evaluate`. After a certain amount of time or whenever the smart meter feels that enough utilities have committed and opened their template load profiles, the smart meter can call the *evaluate* method. This method can only be called if both,

the smart meter and the utilities have successfully opened their commitments and it can only be called by the owner of the smart contract. The method performs the actual matching (i.e., finding the minimum distance in the embedded dimension as shown in Equation 2) and returns both, the address of the utility with the best-matching tariff and the index of that tariff (u^*, l^*). This terminates the execution of the smart contract and the smart meter uses this information to run the *oblivious transfer* phase outside of the blockchain and the smart contract.

For an actual implementation of such a smart contract, there are a few aspects to be considered. First, the evaluation of the original protocol [4] has shown that an embedding dimension of $m = 8192$ is needed, which consequently leads to 8192 bit numbers that need to be processed for the matching. The EVM, for instance, has maximum word size of 256 bit which would require additional steps to process numbers of that size.

Second, the above smart contract fully relies on the privacy features of embeddings. After the smart meter opened its committed load profile forecast or a utility opened its committed template load profile, respectively, this information is publicly available and visible in the blockchain. For this purpose, however, privacy-preserving implementations (e.g., Hawk [23]) can be used, where zero-knowledge-proofs are employed in order to hide information in the blockchain.

5.2 Privacy and Security

The proposed protocol does not rely on a third party like [4], but uses a blockchain and smart contracts instead. Using these technologies, the four requirements introduced in Sections 3.1 and 3.4 are fulfilled, namely:

- **Transparency.** All calculations are transparent to the smart meter and the utility providers. This is guaranteed by the properties of the blockchain [5]. Once data is written to the blockchain, it cannot be changed and therefore prevents adversaries from forging matching results.
- **Verifiability.** Calculations and the results are verifiable by all participants. This is guaranteed, since the results of smart contracts are stored in the blockchain [23, 6] and can be verified through recomputing the blocks.
- **Reliability.** The matching does not rely on a single party, but the computation is decentralized and can be performed by any peer in the blockchain [6].
- **Privacy.** The privacy-preserving properties stem from [4], with one exception that will be discussed below. Since the blockchain stores data permanently and publicly available, additional features such as a commitment scheme and private blockchains are required [23].

In the following, the privacy properties of this protocol are discussed. While most of the privacy analysis is already conducted in [4], the replacement of the third party by a public blockchain requires a discussion of privacy of previously non-public data.

```
ID owner = ∅;
Hash loadforecasthash;
HashMap<ID, Hash[]>templatehashes = new HashMap<>();
Ẽ forecast = ∅;
HashMap<ID, Ẽ[]>templates = new HashMap<>();

begin create(ID sender, Hash loadforecast)
    owner = sender;
    loadforecasthash = loadforecast;
end

begin commit(ID sender, Hash[] loadprofiles)
    if ¬ templatehashes.contains(sender) then
        templatehashes.add(sender, loadprofiles);
    else
        return "load profile commitments already sent";
    end
end

begin smopen(ID sender, Ẽ loadforecast, int r)
    if sender == owner ∧ H(loadforecast|r) == loadforecasthash then
        forecast = loadforecast;
    else
        return "invalid commitment";
    end
end

begin uopen(ID sender, Ẽ[] loadprofiles, int[] r)
    if templatehashes.contains(sender) then
        int index = 0;
        foreach Hash embeddinghash in templatehashes.getValues(sender) do
            if ¬ (embeddinghash == H(loadprofiles[index]|r[index])) then
                return "invalid commitment";
            end
            index ++;
        end
        templates.add(sender, loadprofiles);
    end
end
```

Algorithm 2: This first portion of the smart contract for matching a load profile forecast to template load profiles shows the creation, commitment and open methods for both, the smart meter and the utilities. The parameters of type **ID** always refer to the caller. A smart meter initiates a matching by creating a smart contract with a commitment for its load profile forecast. Utilities then commit their template load profiles. The smart meter and the utilities can open the commitments before determining the best-matching tariff.

```
begin evaluate(ID sender)
    if sender == owner ∧ ¬ (forecast == ⊘) ∧ ¬ templates.empty() then
        double bestdistance = ∞;
        ID bestutility = ⊘;
        int besttariffindex = ⊘;
        foreach ID utility in templates do
            int index = 0;
            foreach Ẽ embedding in templates.getValues(utility) do
                double distance = ‖embedding − forecast‖;
                if distance < bestdistance then
                    bestdistance = distance;
                    bestutility = utility;
                    besttariffindex = index;
                end
                index ++;
            end
        end
        return (bestutility, besttariffindex);
    end
    return "not allowed";
end
```

Algorithm 3: This second portion of the smart contract for matching a load profile forecast to template load profiles shows the evaluation to find the minimum distance. If all commitments have been opened, the creator of the smart contract can evaluate the template load profiles and determine the best-matching tariff and the corresponding utility offering that tariff.

This protocol is privacy-preserving as none of the participating entities has access to the original load profiles of the others, but only to the embedded load profiles. In [4], the privacy features of the embedding approach are discussed in detail.

In [4], it is assumed that all parties involved in communication through our proposed protocol are authenticated, e.g., through X.509 certificates [37]. This is not necessary because, using a blockchain, all parties are already automatically authenticated using public key cryptography [5] where each peer in the blockchain has a unique key, also referred to as address.

Similarly, in [4], for the *initialization* and *matching* phases, it is assumed that all communication links are encrypted, e.g., by symmetric encryption such as AES [38]. In contrast, such an encryption of the communication links is not used in the proposed protocol. Instead, two techniques are applied that prevent others from breaking confidentiality. A commitment scheme is used to prevent other parties from learning embedded load profiles during the commitment phase of a smart contract. However, this does not prevent the embedded load profiles from being released once the smart contract has been evaluated, which requires to open the commitments.

In the proposed protocol as shown in Fig. 3, the parameters $\mathbf{A}$, $\mathbf{W}$ and Δ are only known to the smart meter and the utilities as they are in [4]. However, in the proposed protocol, the embedded load profile forecast $\tilde{\mathbf{F}}$ as well as the embedded

template load profiles $\tilde{\mathbf{L}}_{u,l}$ are publicly known since they are committed into the blockchain. As a consequence, the smart meter and all utilities know the embedding parameters and all embedded load profiles. Thus, privacy boils down to the question how much information can be inferred from both, the embedding parameters and the embedded load profiles.

Since the focus of this work is to show that blockchains can be used in principle for the protocol originally proposed in [4], the privacy analysis of this aspect is left as future work. However, it should be noted that privacy-preserving smart contracts as described in [23] can be used at the cost of reduced transparency. This way, embedded load profiles remain completely private even after the execution of a smart contract, thus solving this issue.

In Section 3.2.1 is is discussed that in [4] the third party may collect statistics on the matching results by collecting the indices (u^*, l^*). Principally, the same holds for the blockchain-based approach where these indices are the result of evaluating the smart contract. For the approach with a **TP**, utilities can regularly shuffle their template load profiles' indices. The same approach can be applied in the approach presented in this paper. Additionally, all parties are only identified by their **ID** or address in the blockchain. This pseudonymity in combination with the shuffling of the indices prevents other parties from learning such statistics. Furthermore, bulk probing of either the smart meter or the utility providers is a similar issue, which is discussed in detail in [4].

In the *oblivious transfer* phase, which is identical to [4], no additional encryption is necessary. The security properties of oblivious transfer are not discussed in this analysis, but can be found in literature, e.g., [10, 24, 25], since this analysis mainly focuses on the privacy impact of the information that is transferred.

5.3 Comparison to Related Work

In this section, the proposed blockchain-based approach is compared to existing approaches. Figure 4 shows a comparison of three approaches: the presented approach, the approach relying on a third party, which we extend in this paper, and an approach using homomorphic encryption. This section extends the discussion originally presented in [4].

The core purpose of the presented approach is to find similarities, i.e., nearest neighbors, of time series by Euclidean distance computation while preserving the privacy of all actors. In [9, 17, 39], approaches based on additively homomorphic encryption, e.g., the Paillier cryptosystem [40], are discussed which also allow calculating Euclidean distances. Thus, they can be used as an alternative to nearest-neighbor embeddings as proposed in this paper and in [4]. Therefore we compare our approach to the latter. Figure 4 (left) shows a simplified version of our proposed scheme in this paper, (center) of the scheme proposed in [4] and (right) approaches using homomorphic encryption.

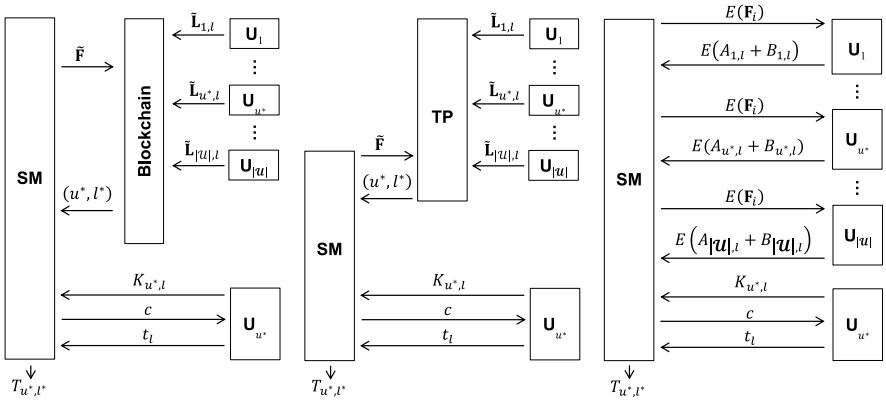

Fig. 4 Comparison of three distinct approaches for privacy-preserving tariff matching. From left to right: (i) blockchain-based approach as presented in this work; (ii) embedding-based approach that relies on a third party for matching as presented in [4]; and (iii) approach that uses homomorphic encryption proposed in [9, 17] and discussed in [4].

In the following, the approach using homomorphic encryption is described. Furthermore, the embedding-based approach from [4] is compared to the one using homomorphic encryption. Finally, our proposed approach, which is similar to the embedding-based approach is compared to the latter and the communication overhead is discussed.

In the homomorphic encryption-based approach the smart meter communicates with each utility separately sending a load profile forecast encrypted with an additively homomorphic scheme. Each utility responds with a partial result for each template load profile as shown below. From this, the smart meter can calculate the Euclidean distance in order to retrieve the index (u^*, l^*) of the best-matching template load profile. Finally, the smart meter and the utility u^* run the oblivious transfer protocol identically to our approach in order to retrieve the tariff T_{u^*,l^*}.

Like our proposed protocol, this homomorphic protocol comes without the need of a third party. By contrast, fetching the template load profiles as well as the distance computation itself is performed by the smart meter. All encryptions are performed with the public key, decryptions can only be performed by the smart meter, which is the only party owning the private key.

The Euclidean distance between the forecast load profile and each of the template load profiles of all utilities must be computed by using an additively homomorphic cryptosystem [9] in an identical manner, i.e., with the same modulus n of Paillier's cryptosystem. Therefore, for sake of readability, $\mathbf{L}_i$ is written instead of $\mathbf{L}_{u,l,i}$ and A, B are written instead of $A_{u,l}$ and $B_{u,l}$. The goal of the protocol is to privately compute

$$||\mathbf{F}-\mathbf{L}||_2 = \sum_i \mathbf{L}_i^2 - \sum_i 2\mathbf{F}_i\mathbf{L}_i + \sum_i \mathbf{F}_i^2 =: A+B+C. \tag{5}$$

First, the smart meter submits its encrypted load profile forecast values $E(\mathbf{F}_i)$ directly to each utility. As a single load profile forecast value is much smaller than the modulus, data packing is used to better exploit the input domain. Therefore, not a single value is encrypted, but all k values of the load profile forecast are packed, encrypted and sent as one message. Data packing is achieved by shifting values to a certain bit range, such that for all operations the value remains within that range [16], e.g., for k values and a range of b bits $v = v_1|v_2|\ldots|v_k = \sum_{i=1}^{k} v_i 2^{b(i-1)}$.

Using this encrypted load profile forecast and exploiting the homomorphic properties $E(x+y) = E(x)E(y)$ and $E(x)^c = E(cx)$, the utility can compute and send back the partial result

$$E(A+B) = E\left(\sum_i \mathbf{L}_i^2\right) \prod_i E(\mathbf{F}_i)^{-2\mathbf{L}_i} \tag{6}$$

to the smart meter. Exploiting the homomorphic property again, term C can be added and the smart meter gets the Euclidean distance by

$$||\mathbf{F}-\mathbf{L}||_2 = D\left(E\left(\sum_i \mathbf{F}_i^2\right) E(A+B)\right) \tag{7}$$

When following the above protocol, neither the utility knows the smart meter's load profile forecast, nor does the smart meter know the utility's template load profile. In addition, the inner product $\mathbf{F}\cdot\mathbf{L}$ is never revealed to any of them.

Now, the communication need, i.e., bandwidth requirement, is calculated and compared with this paper's solution. In the first step of the protocol, the smart meter needs to send k packed and homomorphically encrypted load values F_i to each of the $|\mathscr{U}|$ utilities. By encrypting a single value with the Paillier cryptosystem, a plaintext $p \in \mathbb{Z}_n^*$ results in a ciphertext $E(p) \in \mathbb{Z}_{n^2}^*$ leading to an expansion of the bit size by a factor of 2.

The second message is the encryption of

$$A+B = \sum_i \mathbf{L}_i^2 - \sum_i 2\mathbf{F}_i\mathbf{L}_i \in \left[-\sum_i F_i^2, \sum_i L_i^2\right] \tag{8}$$

where the lower limit follows from the fact that

$$||\mathbf{F}-\mathbf{L}||_2 = A+B+C \geq 0 \Rightarrow A+B \geq -C. \tag{9}$$

Since squaring of a number leads to an expansion of 2, both, the lower and upper limits need a maximum of $k\cdot 2s$ bits, where s is the bit size of a single load value F_i. Therefore, $A+B$ needs $2\cdot 2ks = 4ks$ bits. Because of subsequent data expansion by a factor of 2 due to encryption, finally, the ciphertext fits into $8ks$ bits, which would require a modulus of $8ks$ bits size for the Paillier cryptosystem with modulus n. If the modulus is smaller (see below for a practical example), the message can be split into multiple messages, the number of which is $\lceil \frac{8ks}{2n} \rceil = \lceil \frac{4ks}{n} \rceil$.

As a practical example consider $k = 96$ values of a day profile arising from a 15 minute measurement interval, a bit size of $s = 16$ and $|\mathscr{U}| = 5$ utilities, each having $|\mathscr{L}_u| = 20$ template load profiles. Therefore, a template load profile (as well as load profile forecast) is of size $4ks = 4 \cdot 96 \cdot 16 = 6144$ bits. According to latest NIST recommendations [41], a Paillier modulus of $n = 2048$ bits (expanding to 4096 bits) is chosen, which requires three messages of that size, since $\frac{8ks}{2n} = \frac{4ks}{n} = 3$.

For the homomorphic encryption approach, the smart meter sends its homomorphically encrypted load profile forecast to each of the $|\mathscr{U}|$ utilities. As described above, sending one load profile forecast requires three messages of 4096 bits size after encryption. Sending all load profile forecasts in this step therefore requires $3 \cdot 4096|\mathscr{U}|$ bits.

In addition, the *oblivious transfer* step at the end requires one message of 256 bits size (when using AES-256 [38]). Thus, the total bandwidth for sending needed by a single smart meter is 7.53 KiB (rounded to two decimal places).

Each utility responds to the requesting smart meter with a partial result (see Equation 6) for each of the $|\mathscr{L}_u|$ template load profiles, as described above. From this, the smart meter calculates the Euclidean distance. One template load profile requires $3 \cdot 4096$ bits, as described above. One utility therefore sends $3 \cdot 4096|\mathscr{L}_u|$ bits. Thus, $|\mathscr{U}|$ utilities send $3 \cdot 4096|\mathscr{L}_u||\mathscr{U}|$ bits, which are received by the smart meter.

In addition, the *oblivious transfer* step requires sending $|\mathscr{L}_u|$ messages of 256 bits size. The smart meter thus receives a total of $3 \cdot 4096|\mathscr{L}_u||\mathscr{U}| + 256|\mathscr{L}_u|$ bits $= 150.63$ KiB (rounded to two decimal places). The total required bandwidth is shown in Figure 5 (solid and dashed black lines).

With the embedding approach, the smart meter needs to send its load profile forecast consisting of m bits of data to the third party. All $|\mathscr{U}|$ utilities need to send $|\mathscr{L}_u|$ template load profiles of m bits each, totaling $|\mathscr{U}||\mathscr{L}_u|m$ bits. For the communication of the best-matching indices (u^*, l^*), $s = 16$ bits $= 2$ B should suffice.

Figure 5 shows the bandwidth requirement for the embedding approach presented in [4] (solid and dashed red lines, as well as dash-dotted blue line) for various embedding dimensions m and a third party. As discussed in [4], $m = 8192$ (dashed-dotted gray line) is a reasonable choice, resulting in an overhead of 100 KiB.

The bandwidth for sending and receiving required by smart meters is orders of magnitudes smaller for the embedding protocol than for the protocol with homomorphic encryption. This is important, since in practical scenarios the bandwidth connecting smart meters with other parties is likely to be low [42], especially when using power-line communication (PLC). While the total communication amount for the embedding method can even be higher than for the homomorphic method, most of it is needed for the communication from utilities to the third party where a much better connection is likely.

The approach based on homomorphic encryption does not require a third party to perform the distance calculation, since either of the participants is involved exactly once in exchanging messages and can limit the rate of requests in order to prevent chosen-plaintext attacks to learn about load profiles. However, the smart meter – which is usually a device with only limited computational capabilities – has to perform the distance computation for every template load profile from every

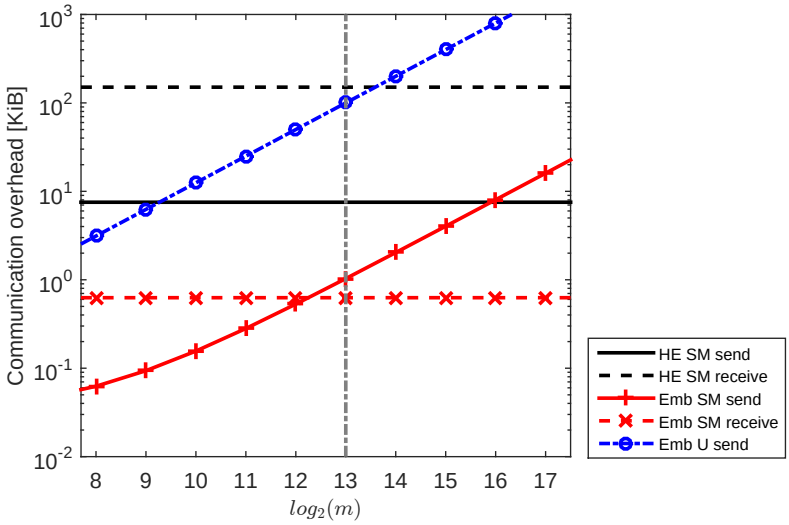

Fig. 5 Comparison of the required bandwidth for the protocol based on homomorphic encryption and the one from [4] with variable embedding dimension m. The bandwidth needed by the smart meter is considerably lower with the embedding protocol (denoted *Emb*) than with the protocol using homomorphic encryption (denoted *HE*). Utilities only need to send data to the **TP** using the embedding protocol (blue, dashed-dotted).

utility, which results in a total of $\sum_u |\mathscr{L}_u|$ distance computations. This is likely to be impractical for a device with low computational capabilities like a smart meter and thus another disadvantage compared to our approach.

However, the homomorphic approach calculates all distances exactly. This is not the case in the profile matching approaches presented in this paper and in [4]. In summary, the smaller overhead in data expansion comes at the cost of only near-perfect matching. However, the accuracy depends on m which can be chosen appropriately as shown in [4].

Finally, in terms of bandwidth the proposed protocol is nearly identical to the embedding-based protocol. As shown in Figure 4, the third party **TP** is replaced by a blockchain, but the same messages need to be sent.

However, there is an additional overhead due to the use of commitments – depending on the size of the hash y as discussed in Section 3.3. Assuming $y = 256$ for SHA-2, each commitment requires 512 bits in total – 256 bits for the hash as the commitment and 256 bits for the random number in the opening. The smart meter sends these 512 additional bits once for the load profile forecast, whereas each utility sends 512 additional bits per template load profile, resulting in a total required bandwidth of $512|\mathscr{L}_u|$. As shown in Figure 6, this additional overhead becomes negligible for large values of m. For $m = 8192$, as argued practically in [4], the sending overhead is approximately 6% for both, **SM** and **U**. The receiving overhead is identical to the embedding approach.

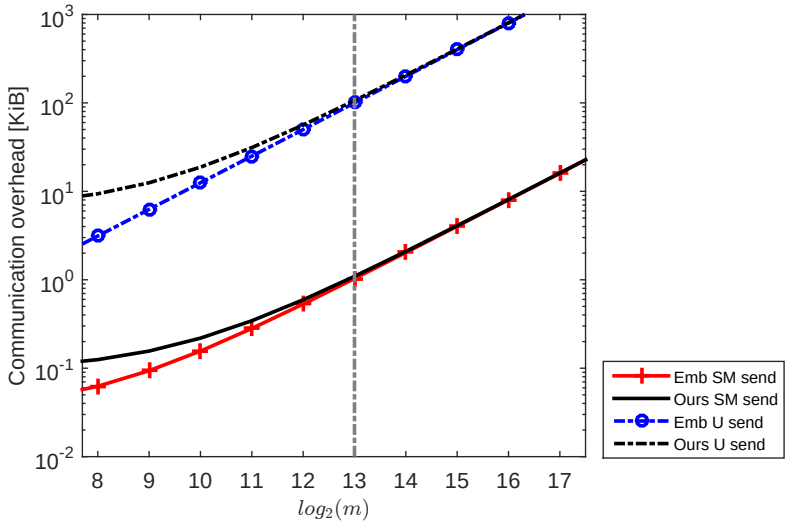

Fig. 6 Comparison of the required bandwidth for our proposed protocol and the one from [4], which both are based on embeddings. This figure uses the same scaling as Figure 5 with variable embedding dimension m. The bandwidth needed for our proposed protocol (denoted *ours*) is slightly higher for the smart meter and utilities (due to the commitments) than with the protocol using a **TP** (denoted *Emb*) that is not required in our protocol. While the required bandwidth for sending is slightly higher, both protocols perform equally well for receiving (not depicted).

6 Conclusion

We described a load profile matching protocol which enables tariff decisions in smart grids using blockchains and smart contracts. Our protocol finds the best-matching tariff for a customer with 93.5% accuracy, while ensuring transparency, verifiability and reliability. The proposed protocol outperforms homomorphic encryption and has comparable communication overhead to previous related work [4] without relying on a third party. Instead of a single third party, a blockchain and smart contracts are used which allow for decentralized, privacy-preserving tariff-decisions in the smart grid. As in [4], the privacy of all participants depends on the usage of embedding transformations with one exception that can be mitigated by the use of privacy-preserving smart contracts [23].

Future work will focus on an actual implementation in Solidity and an evaluation of the costs associated with the evaluation of the proposed smart contract, especially when using privacy-preserving smart contracts as proposed in [23].

Acknowledgments

The financial support by the Austrian Federal Ministry of Science, Research and Economy and the Austrian National Foundation for Research, Technology and Development is gratefully acknowledged. Funding by the Federal State of Salzburg is gratefully acknowledged. The authors would like to thank their partner Salzburg AG for providing real-world load data.

References

1. L. Karg, K. Kleine-Hegermann, M. Wedler, and C. Jahn, "E-Energy Abschlussbericht – Ergebnisse und Erkenntnisse aus der Evaluation der sechs Leuchtturmprojekte," Bundesministerium fr Wirtschaft und Technologie (German Federal Ministry for Economy and Technology), Tech. Rep., 2014, in German. [Online]. Available: http://www.digitale-technologien.de/DT/Redaktion/DE/Downloads/ab-gesamt-begleitforschung.pdf?__blob=publicationFile&v=4
2. M. Lisovich, D. Mulligan, and S. Wicker, "Inferring Personal Information from Demand-Response Systems," *IEEE Security & Privacy*, vol. 8, no. 1, pp. 11–20, 2010.
3. E. McKenna, I. Richardson, and M. Thomson, "Smart meter data: Balancing consumer privacy concerns with legitimate applications," *Energy Policy*, vol. 41, pp. 807–814, 2012.
4. A. Unterweger, F. Knirsch, G. Eibl, and D. Engel, "Privacy-preserving load profile matching for tariff decisions in smart grids," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 21, 2016. [Online]. Available: http://dx.doi.org/10.1186/s13635-016-0044-1
5. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Bitcoin.org*, p. 9, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
6. G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," Ethereum, Tech. Rep., 2017. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf
7. P. McDaniel and S. McLaughlin, "Security and Privacy Challenges in the Smart Grid," *IEEE Security Privacy Magazine*, vol. 7, no. 3, pp. 75–77, 2009.
8. G. Eibl and D. Engel, "Influence of Data Granularity on Smart Meter Privacy," *IEEE Transactions on Smart Grid*, vol. 6, no. 2, pp. 930–939, 2015.
9. S. D. Rane and P. Boufounos, "Privacy-Preserving Nearest Neighbor Methods: Comparing Signals Without Revealing Them," *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 18–28, 2013.
10. J. Kilian, "Founding Cryptography on Oblivious Transfer," in *ACM Symposium on Theory of Computing*. Chicago, IL, USA: ACM, 1988, pp. 20–31.
11. S. Mukherjee, Z. Chen, and A. Gangopadhyay, "A privacy-preserving technique for Euclidean distance-based mining algorithms using Fourier-related transforms," *VLDB Journal*, vol. 15, no. 4, pp. 293–315, 2006.
12. P. Ravikumar, W. W. Cohen, and S. E. Fienberg, "A Secure Protocol for Computing String Distance Metrics," *International Conference on Data Mining (ICDM)*, pp. 40–46, 2004.
13. W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN Computation on Encrypted Databases Categories and Subject Descriptors," *Proceddings of the 35th SIGMOD International Conference on Management of Data*, pp. 139–152, 2009. [Online]. Available: http://doi.acm.org/10.1145/1559845.1559862
14. P. T. Boufounos and S. Rane, "Efficient Coding of Signal Distances Using Universal Quantized Embeddings," *2013 Data Compression Conference (DCC)*, pp. 251–260, 2013.
15. J. H. Cheon, M. Kim, and K. Lauter, *Homomorphic Computation of Edit Distance*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, vol. 8976, pp. 194–212.
16. Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating Private Recommendations Efficiently Using Homomorphic Encryption and Data Packing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1053–1066, 2012.

17. S. D. Rane, W. Sun, and A. Vetro, "Secure distortion computation among untrusting parties using homomorphic encryption," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 1485–1488.
18. M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. D. Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, A. Piva, and F. Scotti, "A Privacy-compliant Fingerprint Recognition System Based on Homomorphic Encryption and Fingercode Templates," in *IEEE 4th International Conference on Biometrics: Theory, Applications and Systems, BTAS 2010*, 2010, pp. 1–7.
19. A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient Privacy-Preserving Face Recognition," in *Information, Security and Cryptology – ICISC 2009*, ser. Lecture Notes in Computer Science, D. Lee and S. Hong, Eds., vol. 5984. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 229–244.
20. V. Kolesnikov, A. R. Sadeghi, and T. Schneider, "Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima," *Lecture Notes in Computer Science*, vol. 5888, pp. 1–20, 2009.
21. E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Proceedings – IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 459–474.
22. G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proceedings – 2015 IEEE Security and Privacy Workshops, SPW 2015*, 2015, pp. 180–184.
23. A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 839–858.
24. A. C.-c. Yao, "How to Generate and Exchange Secrets," in *27th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1986, pp. 162–167.
25. D. Catalano, R. Cramer, G. DiCrescenzo, I. Darmgard, D. Pointcheval, and T. Takagi, *Provable Security for Public Key Schemes*. Basel: Birkhäuser Verlag, 2005.
26. P. Palensky and D. Dietrich, "Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 381–388, 2011.
27. S. Caron and G. Kesidis, "Incentive-Based Energy Consumption Scheduling Algorithms for the Smart Grid," in *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2010, pp. 391–396.
28. S. Shao, T. Zhang, M. Pipattanasomporn, and S. Rahman, "Impact of TOU Rates on Distribution Load Shapes in a Smart Grid With PHEV Penetration," in *2010 IEEE PES Transmission and Distribution Conference and Exposition: Smart Solutions for a Changing World*, 2010, pp. 1–6.
29. S. Ramchurn, P. Vytelingum, A. Rogers, and N. Jennings, "Agent-Based Control for Decentralised Demand Side Management in the Smart Grid," in *The 10th International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '11, vol. 1. Taipei, Taiwan: International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 5–12. [Online]. Available: http://eprints.soton.ac.uk/271985/
30. A.-H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous Demand-Side Management Based on Game-Theoretic Energy Consumption Scheduling for the Future Smart Grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 3, pp. 320–331, 2010.
31. F. Knirsch, "Privacy enhancing technologies in the smart grid user domain," *it - Information Technology, Thematic Issue: Recent Trends in Energy Informatics Research*, vol. 1, no. 59, pp. 13–22, 2017.
32. R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, "Secure Network Coding over the Integers," in *Public Key Cryptography – PKC 2010*, ser. Lecture Notes in Computer Science, D. Pointcheval and P. Q. Nguyen, Eds., vol. 6056. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 142–160.

33. D. Fiore, R. Gennaro, and V. Pastro, "Efficiently Verifiable Computation on Encrypted Data," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. Scottsdale, AZ, USA: ACM, 2014, pp. 844–855.
34. G. W. Peters and E. Panayi, "Understanding Modern Banking Ledgers through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money," in *Banking Beyond Banks and Money: A Guide to Banking Services in the Twenty-First Century*, T. Paolo, T. Aste, L. Pelizzon, and N. Perony, Eds. Cham: Springer International Publishing, 2016, pp. 239—-278.
35. K. Delmolino, M. Arnett, A. E. Kosba, A. Miller, and E. Shi, "Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab." in *Financial Cryptography and Data Security*. Barbados: International Financial Cryptography Association, 2016, pp. 79—-94.
36. T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology Crypto 91*, vol. 91, 1992, pp. 129–140.
37. ITU-T, "Recommendation ITU-T X.509 – Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks," 2012.
38. National Institute of Standards and Technology (NIST), "Specification for the Advanced Encryption Standard (AES)," 2001.
39. R. Lagendijk, Z. Erkin, and M. Barni, "Encrypted Signal Processing for Privacy Protection," *IEEE Signal Processing Magazine*, vol. 30, pp. 82–105, 2013.
40. P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology — EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, vol. 1592, pp. 223–238.
41. E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, and C. S. Division, "NIST 800-57: Computer Security," pp. 1–147, 2012.
42. A. Unterweger and D. Engel, "Resumable Load Data Compression in Smart Grids," *IEEE Transactions on Smart Grid*, vol. 6, no. 2, pp. 919–929, 2015. [Online]. Available: http://dx.doi.org/10.1109/TSG.2014.2364686