Private and Secure Smart Meter Data Analytics

Rune Hylsberg Jacobsen and Ali Marandi
Aarhus University, Electrical and Computer Engineering

The 9th DACH+ Conference on Energy Informatics,
Sierre, Switzerland, October 29-30, 2020
Outline

Introduction
- Smart meter data analytics
- System concept and threat model
- Need for privacy-preservation

Homomorphic encryption for smart energy
- Homomorphic cryptosystems
- Application to data analytics

Decentralizing trust in smart energy
- Blockchain for smart energy services
Smart meter deployment in Europe

EU Third Energy Package

- Roll-out target of 80% market penetration for electricity by 2020 – subjected to a long-term cost-benefit analysis (CBA).
- Installation of metering points (#):
  - 183.394.298 (up to 2020)
  - 192.469.258 (expected by end of 2020)

Danish DataHub

Purpose and operation

• Ensure uniform communication and standardized processes for energy actors → competition → customer benefits.
• Gathering all information about the Danes' electricity.
• Operated by the Danish TSO EnergiNet.
• Smart electricity meter data with 15 min. time resolution.
• Privacy restrictions (administratively).
Smart energy grid

Smart meter as an enabler

- Bi-directional communications
- Near real-time monitoring
- Demand forecast and planning
- Fraud detection

Figure source [1]
Home area network (HAN)

Internet of Things devices – sensors, actuators, gateways

- Constrained devices
  - Limited computation power, memory, energy supply
  - Lossy, wireless communication environments

- Trade-offs
  - Distributions of data processing: Local/Edge/Cloud
  - Level of security & privacy enforcements

Source: Ref. [19]
Smart meter data aggregation

Temporal and spatial and aggregation

- Gateways (GWs) and centralized servers (e.g. control center) are responsible for SM data aggregations.
- Temporal aggregation of individual SMs.
- Spatial aggregation in a Neighborhood Area Network (NAN).
Simple data analytics

Temporal and spatial aggregation

\( N \) number of SMs in a NAN

\( T \) temporal aggregation period (billing period)

\( d \) fine timestamp, \( x^i_d \): SM’s (plaintext) data at timestamp \( d \)

\( x_{TA}(i) \) total temporal aggregation for SM \( i \) during period \( T \)

\( x_{SA}(d) \) total spatial aggregation of a BAN in \( d \) timestamp

\[
\begin{align*}
    x_{TA}(i) &= \sum_{d=1}^{T} x^i_d \quad i = 1, \ldots, N \\
    x_{SA}(d) &= \sum_{i=1}^{N} x^i_d \quad d = 1, \ldots, T
\end{align*}
\]
Smart energy services

Computational requirements

- Billing might use coarse-grained temporally aggregated data (e.g., monthly consumption). Requires sum and average.
- Demand/consumption forecasting uses fine-grained spatial aggregation. Might require regression analysis, period matching or machine learning.
- Uniformity of consumption uses variance analysis.
- Observing the impact of price strategies on consumption uses one-way ANalysis Of VAriance (ANOVA).
- Anomaly detection might use >2 degree operations, e.g., statistical comparisons, machine learning.
Need for security and privacy protection

Threat model

Data plane

- Eavesdropping
- Intrusion (SMs, gateways, control center)
- Information secrecy (Gateways should not know the values of spatially-or-temporally aggregated SM data)

Control plane

- (Access) control over personal data
- Security key disclosure risk
- Non-repudiation (all the interactions between users, gateways, control center, TSO, DSO)
- Availability – centralized servers
Need for security and privacy protection

Trust model

Data plane
- Data authenticity and integrity
- Tamper-evidence

Control plane
- Entity authentication
- Trust relationships
- Access rights
Detecting user behavior from meter data

Application of load disaggregation

Source: Refs. [10, 13]
Detecting user behavior from meter data

Natural language report generation synthesized from model

"Peter (User ID: 123) living in XYZ (Home ID: 2) was watching TV from 18:00 to 20:45 on 01 May 2015."
### Societal view on privacy

**European directives relevant to data processing in smart meters**

- **European Data Protection Directive** which governs the processing of personal data by data controllers and grants rights to individuals.

- **European Privacy and Electronic Communications Directive** which aims to make it technology neutral.

  - Personal data processing is allowed only if specific legal purposes apply.
  - Personal data gathered for one purpose cannot be used for another purpose without permission.
  - There are limitations on the personal data transfer to other countries.
  - There is a strict obligation to ensure adequate security.

Source: Ref. [13]
Privacy preservation

Challenges with traditional security schemes

End-to-end encryption **lacks scalability** to support the large number of smart meters.

- Need to encrypt smart meter data instead of securing communication paths.

Privacy-preservation techniques such as down-sampling, aggregation, anonymization of data lead to **reduced amount of information**.

- Vitally important to keep high-resolution smart meter data private to avoid the disclosure of users' behaviors.
Homomorphic encryption
Homomorphic encryption (HE) schemes

Overview and expected usability

• **Paillier cryptosystem** is additively homomorphic (~ simple summation and only one multiplication).

• **Boneh-Goh-Nissim (BGN) cryptosystem** could support one multiplication by leveraging bi-linear mapping (~ statistical operations with degree \( \leq 2 \)).

• **Somewhat HE (SHE) cryptosystem** can support limited number of multiplications (~ statistical operations even with degree \( \geq 2 \)).

• **Gentry cryptosystem** can support arbitrary multiplication depth (e.g., neural networks).
Cryptosystems

Core algorithms

- Key generation
- Encryption
- Decryption
Paillier cryptosystem

Key generation

- Large prime numbers $p$ and $q$ so that $\gcd(pq, (p - 1)(q - 1)) = 1$
- Compute RSA modulus $n = pq$, $\lambda = lcm(p - 1, q - 1)$
- Select a random integer generator $g \in \mathbb{Z}_{n^2}^*$ and set

$$\alpha = (L(g^\lambda \mod n^2)^{-1} \mod n$$

- $L(u)$ is the quotient of $\frac{u - 1}{n}$
- Public key will be $PK = (n, g)$; Secret key will be $SK = (\lambda, \alpha)$. 
Paillier cryptosystem

Encryption and Decryption

Encryption

• For a message $m \in \mathbb{Z}_n$ and a selected random number $r \in \mathbb{Z}^*_n$, the cipher text is
  
  $$ c = E(m) = g^m r^n \mod n^2 $$

Decryption

• For a cipher text $c \in \mathbb{Z}^*_n$, the below equation gives the plain text
  
  $$ m = D(c) = L(c^\lambda \mod n^2)(\alpha \mod n) $$
Paillier cryptosystem

Homomorphic Properties

- If $m_1, m_2, r_1$ and $r_2$ are random numbers, the following homomorphic properties hold using Paillier crypto system:

$$E(m_1, r_1)E(m_2, r_2) = E(m_1 + m_2, r_1 r_2) \mod n^2; \text{ and}$$

$$(E(m_1, r_1))^{m_2} = E(m_1 m_2, r_1^{m_2}) \mod n^2$$
Data partitioning

Data aggregation with multiple sources

Pairwise nodes

<table>
<thead>
<tr>
<th>S (1)</th>
<th>S (2)</th>
<th>S (3)</th>
<th>S (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,3</td>
<td>1,4</td>
<td>1,4</td>
<td>2,3</td>
</tr>
</tbody>
</table>

Plain text data

<table>
<thead>
<tr>
<th>U1</th>
<th>U2</th>
<th>U3</th>
<th>U4</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>6</td>
<td>10</td>
<td>9</td>
</tr>
</tbody>
</table>

Partitioned data

<table>
<thead>
<tr>
<th>U1</th>
<th>U2</th>
<th>U3</th>
<th>U4</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>
Secure linear regression

Linear regression with homomorphic encryption

Linear regression on large amount of SM data requires cloud computing

\[ y = X\beta + \epsilon \]

Computationally feasible linear regression without disclosing individual SM data (Ref. [5])

- Smart meter readings are encrypted using Paillier HE. Linear regression requires one multiplication, thus Paillier crypto system is applicable.
## Design choices

### Comparison of homomorphic encryption schemes

<table>
<thead>
<tr>
<th>Crypto system</th>
<th>Permitted operations</th>
<th>Strong point</th>
<th>Limitations</th>
<th>Supported services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paillier</td>
<td>Adding and multiplying ciphertext</td>
<td>Simplicity</td>
<td>Only one multiplication. Cannot do addition and multiplication at the same time</td>
<td>Billing and distribution grid load estimations</td>
</tr>
<tr>
<td>BGN</td>
<td>Adding and multiplying ciphertext</td>
<td>Can do any number of additions and just one multiplication at the same time</td>
<td>Complex for constrained devices</td>
<td>Variance, one-way ANOVA, any statistics with degree $\leq 2$</td>
</tr>
<tr>
<td>FHE (Gentry)</td>
<td>Any arbitrary statistics</td>
<td>Can support any number of multiplications and additions</td>
<td>Yet, too complex in practice</td>
<td>Any service</td>
</tr>
<tr>
<td>Somewhat HE</td>
<td>Any number of additions but limited number of multiplications</td>
<td>Less complex than FHE, i.e., more practical</td>
<td>Limited multiplication depth</td>
<td>Can support statistics with degree $\geq 2$</td>
</tr>
</tbody>
</table>
Homomorphic encryption with arbitrary computations
Scenario

How can we have privacy preservation for arbitrary data analytics?

Alice (consumer) → Bob (cloud provider) → Result
Case Study - Convolutional Neural Network

- ReLUs are the activation functions.
- We define and compute the number of operations for the layers.
Computing on encrypted data

Logic circuit representations

<table>
<thead>
<tr>
<th>SUM</th>
<th>PRODUCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>AND</td>
</tr>
</tbody>
</table>

If you can compute sums and multiplications on encrypted bits you can compute ANY function on encrypted inputs.
Fully Homomorphic Encryption (FHE)

Bootstrapped version for FHE

- Key Generation: \( pk^*, sk^* = \text{KeyGen}(\lambda, \alpha, \beta) \)
- Encryption: \( c^* = (c, z) = \text{Enc}(pk^*, m) \)
  - Introduces noise to \( m \)
- Decryption: \( m = \text{Dec}(sk^*, c^*) \)
- Recryption - homomorphic itself
  - Helps to remove the noise

Homomorphic properties

\[
D_{sk}(c_1 + c_2) = D_{sk}(c_1) + D_{sk}(c_2) \quad \text{(addition)} \\
= m_1 + m_2
\]

\[
D_{sk}(c_1 c_2) = D_{sk}(c_1) D_{sk}(c_2) \quad \text{(multiplication)} \\
= m_1 m_2
\]

Source: Refs. [9] and [14]
Noise accumulation

Somewhat homomorphic encryption example - symmetric

<table>
<thead>
<tr>
<th>Private key</th>
<th>Encryption</th>
<th>Decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p \in R {0, 1}^{\lambda^2}$</td>
<td>$E_p(m) = m' + pq = c$</td>
<td>$D_p(c) = ((m' + pq \mod p) \mod 2 = m$</td>
</tr>
<tr>
<td>Select random element in set</td>
<td>Nonce $q \in R {0, 1}^{\lambda^5}$</td>
<td></td>
</tr>
<tr>
<td>Security parameter</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- For two messages $m_1$ and $m_2$ the homomorphic properties only holds when $m'_1 + m'_2 < p$ and $m'_1 \cdot m'_2 < p$
- If $m'_3 = m'_1 \cdot m'_2 > p$ we may write $m'_3 = p + r$ for $r > 0$
- When reducing modulo $p$ during decryption, the parity of $r$ is not guaranteed to be the same as the parity of $m'_3$ hence we cannot guarantee the result after reducing with modulo 2
Bootstrapping

Fixes the issue of noise accumulation

• The process of bootstrapping is to perform a **recryption** of a ciphertext.

• By doing so the accumulated **noise is reset** to the initial noise of a "fresh" ciphertext plus the noise added by boot-strapping.
Bootstrapping

Fixes the issue of noise accumulation

• The process of bootstrapping is to perform a *recryption* of a ciphertext.

• By doing so the accumulated noise is reset to the initial noise of a "fresh" ciphertext plus the noise added by boot-strapping.
Implementation

- CPU: Intel i5-3570k @ 3.4 GHz, 8 GB RAM, Linux OS
- Single convolutional layer with 10 channels.
- 9x9 Kernels, 2x2 Mean pooling Softmax with 10 classes
- 10810 training weights estimated through backpropagation.
- Training based on plaintext.
- Plaintext reached classification accuracy of 91.2%

MNIST dataset

Classification of the MNIST dataset in implementation in Python 3.6

- 28x28 pixel images of handwritten digits
- 60,000 for training, 10,000 for testing
Results

- Measured up to 20 bits of security – not enough for \textit{e.g.} birthday attacks.
- Ideal parameters: $\lambda = 80$, $\alpha = 15$, $\beta = 10449$
- Bootstrap time on average dominates other costs by 10X.
- Total time does not scale well.
- Nevertheless, optimizations can be made.
Results

- Bootstrap time on average dominates other costs by various orders of magnitude

<table>
<thead>
<tr>
<th>Time Cost</th>
<th>$t_{\text{add}}$</th>
<th>$t_{\text{mul}}$</th>
<th>$t_{\text{boot}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaintext</td>
<td>$3.65 \times 10^{-7}$ s</td>
<td>$3.40 \times 10^{-7}$ s</td>
<td>-</td>
</tr>
<tr>
<td>Ciphertext</td>
<td>$1.26 \times 10^{-3}$ s</td>
<td>$9.54 \times 10^{-3}$ s</td>
<td>$20.66$ s</td>
</tr>
</tbody>
</table>

$\lambda = 9, \alpha = 2, \beta = 1 \rightarrow$ For fast initial computations

A classification using an encrypted MNIST image and 18 bits fixed point numbers is estimated to take $\sim 10^{12}$ s (about 50,000 years!!)
Decentralizing trust with blockchain
Blockchain

Motivation
- Contracts, transactions, and the records of them are among the defining structures in our economic, legal, and political systems.
  - Protecting assets
  - Defines organizational boundaries
  - Establish and verify identities

Key elements
- Transactions (tx)
- Consensus protocols
- Blocks
- Smart contracts

About
- Blockchain is an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way.
- Blockchain has the potential to create new foundations for our economic and social systems (not disruption).
Blockchain

• Hash-pointers
  — Pointers to where information is stored and a cryptographic hash of the data.
• Data may consist of transactions
• Application: tamper-evident log
Smart Contracts

Digitalized version of contract
Fully or partially self-enforcing / executing
Stored and replicated on a blockchain

Enforces rules for blockchain members to follow to maintain the ledger.
Software code (chain code) that executes on the blockchain.
Provide a set of functions allowing transaction operations to be made against the blockchain.

```javascript
async queryPermission(stub, args) {
  if (args.length !== 1) {
    throw new Error('Incorrect number of arguments. Expecting permissionID');
  }
  let permissionId = args[0];
  let permissionInBytes = await stub.getState(permissionId);
  if (!permissionInBytes || permissionInBytes.toString().length <= 0) {
    throw new Error('permissionId does not exist!');
  }
  return permissionInBytes;
}
```
Public Keys as identities

- Public key cryptography
  - Key generator function: \((sk, pk) \leftarrow \text{gen()}\)
  - Common algorithms: RSA, ECC
- If you see a signature \((\text{sig})\) on a message \((\text{msg})\) such that \(\text{verify}(pk, msg, \text{sig}) == \text{true}\) you may think of \(pk\) saying “[msg]”.
  - To “speak for” \(pk\), you must know the matching secret \(sk\)
- You can make a new identity by creating new random key-pairs \((sk, pk)\) → may give anonymity
- The identity \(pk\) is controlled by the entity that knows the corresponding \(sk\)

Enabling decentralized identity management. No central part of control.
Peer to peer (P2P) networks

- Network architecture style - structured/unstructured
- Decentralized network of peers
- Dynamically formed
- Enabled by TCP/IP

A blockchain is typically managed by a P2P network, where all nodes (peers) adhere to a protocol for inter-node communication and new block validation.

A peer is an entity in the network which maintains a ledger. Peers execute chain code and perform transaction operations on the ledger.

Chord structures P2P network
Types of blockchain frameworks

**Public blockchain** (e.g. Ethereum)
- Open for participation, open consensus process
- Transparent, transactions are visible
- Secured by a combination of incentives and cryptographic validation, i.e., proof-of-work

**Private blockchain**
- Transactions are controlled by a central authority
- Some transactions may be public
- Higher efficiency and speed of processing transactions

**Consortium blockchain** (e.g. Hyperledger Fabric)
- Partly private and transactions can optionally be public
- Controlled by pre-selected members
- Partially decentralized blockchain

### Comparison of different types of blockchain frameworks

<table>
<thead>
<tr>
<th></th>
<th>Public</th>
<th>Consortium</th>
<th>Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customizable blockchain</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Low-cost transaction</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Level of privacy</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Level of trust</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Known identities</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Incentive</td>
<td>Informative or material</td>
<td>Group goal</td>
<td>Individual goal</td>
</tr>
<tr>
<td>Private communication</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

C – Committer
E – Endorser
Cryptographic Hash Functions

**Attributes**
- Takes any string as input
- Fixed-sized output
- Efficiently computable

**Security properties**
- Collision-free
- Hiding
- Puzzle-friendly

**Collision-free:** Nobody can find \( x \) and \( y \) such that \( x \neq y \) and \( H(x) = H(y) \)

Application: Message digest
If we know \( H(x) = H(y) \) it is safe to assume that \( x = y \)
Cryptographic Hash Functions

Attributes
- takes any string as input
- Fixed-sized output
- Efficiently computable

Security properties
- Collision-free
- Hiding
- Puzzle-friendly

Hiding: Given $H(x)$, it is infeasible to find $x$

$x$ needs to be taken from a set that is very spread out, e.g., $H(r|x)$ where $r$ is a secret value chosen from a highly spread-out probability distribution.

Application: Commitment
Commit to a value and reveal it later i.e., we want to “seal a value in an envelope” and “open the envelope” later.
Cryptographic Hash Functions

Attributes
- takes any string as input
- Fixed-sized output
- Efficiently computable

Security properties
- Collision-free
- Hiding
- Puzzle-friendly

**Puzzle-friendly**: For every possible output value \( y \) if \( r \) is chosen from a distribution with high min-entropy, then it is infeasible to find \( x \) such that \( H(r|x) = y \)

No solving strategy better than trying every possible value of \( x \) for a given \( r \).

**Application: Search puzzle**
Given a puzzle id \( (r=id) \) try to find the “solution” \( x \) such that \( H(id|x) \in Y \) where \( Y \) is the target set of possible \( y \)’s given all possible \( x \)’s
Consensus algorithms

Consensus idea: how do nodes agree on the maintained ledger.

Proof-based consensus algorithms

• A node must perform sufficient computation to prove itself and achieve the right to append a new block to the chain and receive the reward.
  • Proof-of-work (PoW); proof-of-stake (PoS)

Vote-based consensus algorithms

• Nodes are free to join or withdraw from the blockchain network.
  • All the nodes communicate to verify a block.
  • Most nodes must vote for a block to be appended.
# Comparison

## Vote-based vs proof-based consensus

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Vote-based consensus algorithm</th>
<th>Proof-based consensus algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement making basement</td>
<td>From majority of the node decisions</td>
<td>Following nodes performing enough proof (PoW, PoS, etc.)</td>
</tr>
<tr>
<td>Nodes can join freely</td>
<td>No</td>
<td>Mostly</td>
</tr>
<tr>
<td>Number of nodes executing</td>
<td>Limited</td>
<td>Mostly unlimited</td>
</tr>
<tr>
<td>Decentralization</td>
<td>Low</td>
<td>Mostly high</td>
</tr>
<tr>
<td>Trust</td>
<td>Less trustful</td>
<td>Mostly trustful</td>
</tr>
<tr>
<td>Nodes identities are managed</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Security threat</td>
<td>Less serious</td>
<td>More serious</td>
</tr>
<tr>
<td>Award</td>
<td>Mostly no</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table source: Ref. [6]
Customers rules and how to meet them

Customers rules

- **Data capture** including the clear statement of what data are allowed to be captured, stored in databases and for what purposes.
- **Data use** – how the customer data are used.
- **Data storage** – how the data are stored and if they are secured.
- **Data sharing** – what data and how can data be shared with other parties.

Solution

- Use of blockchain to manage storage, access, and sharing rights of users' data.
Permissioned blockchain

Access control to private data

- Data, files and computer programs are assets just much like digital currencies (coins).

- The ownership and access rights can be treated much like coins in a crypto currency.

Source: Ref. [15]
Compound identities

Used for policy check function

- An ID owned by a party (user), which a guest (service) has limited access to it.
- Compound ID consists of **signing key pairs** for the user and the service and a **symmetric key** to encrypt and decrypt the data

- External view of a compound ID: 
  \[ \text{Compound}^{(\text{pub})}_{u,s} = (p_{s}^{u{s}}_{\text{sig}}, p_{s}^{u{s}}_{\text{sig}}) \]

- Complete compound ID: 
  \[ \text{Compound}_{u,s} = (p_{s}^{u{s}}_{\text{sig}}, s_{s}^{u{s}}_{\text{sig}}, p_{s}^{s_{u}}_{\text{sig}}, s_{s}^{s_{u}}_{\text{sig}}, s_{\text{enc}}^{u{s}}) \]

Source: Ref. [15]
Blockchain for Smart Energy Systems
Smart energy service concept

Smart energy cloud services

Service cloud

Local energy market

Energy market

Supply of electricity

Smart home

PV

Sensors & actuators

Fedders

400 V

10 kV

GW

TS

BCN

SM

Smart energy cloud services

Distribution grid

Electricity supplier

PV

Wind turbines

AARHUS UNIVERSITET DEPARTMENT OF ENGINEERING
Automated energy billing

With smart contracts

- Energy consumption will be communicated to blockchain creating a tamper-free log of meter readings.
- Secure Element (SE) provides secure storage/execution.
- Energy prices (+models) are defined into Smart Contract.
- The consumer bill is created according to the consumption and rules enforced by the Smart Contract.
- The output will be written into the blockchain
  - A transparent and tamper-proof record
  - Non-repudiation
Demand Response (DR)

Efficient DR allotment mechanism

- Aggregators submit bids to Tier 1.
- DR bids are signed using secure elements.
- Certifying Authorities (CAs) monitor DR marketplace.
- DR bids are collected at Tier 1.
- An optimization problem is solved to select several bidders (aggregators).
- Selected aggregators and set of accepted/rejected bids are written into the blockchain.

Source: Ref. [17]
Multi-signature wallets

Decentralizing escrow account setting

- Each aggregator has a 3-signature wallet.
- Money outflow requires signatures of the majority (Aggr.-TSO, TSO-CA, CA-Aggr.)
- Payment decisions are taken off-blockchain.
- Payment executions are done on-blockchain.

- CA is trusted here and can arbitrate in case of discrepancies.
- All of this happens after DR bid collection, and DR allotment (Optimization), i.e., selecting only some of the aggregators.

Source: Ref. [17]
Future privacy-preserving smart meter data analytics
Key takeaways

Technologies are available for smart meter data analytics
- Arbitrary computation with encrypted data → too computational heavy
- Lighter homomorphic cryptosystems can be applied with simple data analytics (statistics with low degree)
- Blockchain can decentralize trust in the smart energy system

Smart energy services
- Privacy-preserving smart meter data analytics: descriptive statistics, regression, anomaly detection etc.
- Automatic Billing based on encrypted data using smart contracts
- Demand response settlement with blockchain
Research needs

Key research challenges

• Energy cost (homomorphic encryption, BC consensus)
• Storage cost of homomorphic encryption
• Scalable key management with distributed trust
• Creating value from privacy-preservation
References I


References II


