A DATA PSEUDONYMIZATION PROTOCOL FOR SMART GRIDS

Giulia Mauri
22-05-2013
**SMART GRID**
A vision for the future — a network of integrated microgrids that can monitor and heal itself.

- **Smart appliances**
  - Can shut off in response to frequency fluctuations.

- **Demand management**
  - Use can be shifted to off-peak times to save money.

- **Solar panels**
  - Offices

- **Processors**
  - Execute special protection schemes in microseconds.

- **Sensors**
  - Detect fluctuations and disturbances, and can signal for areas to be isolated.

- **Storage**
  - Energy generated at off-peak times could be stored in batteries for later use.

- **Generators**
  - Energy from small generators and solar panels can reduce overall demand on the grid.

- **Industrial plant**

- **Wind farm**

- **Central power plant**

- **Isolated microgrid**

- **Disturbance in the grid**

Source: Nature
Introduction (II)

- Detailed energy consumption measurements allow:
  - ✔ Timely management of energy distribution,
  - ✔ Efficient grid monitoring,
  - ✔ Energy forecasting and provisioning.
  - ✖ Inference of customers’ personal habits,
  - ✖ Identifying and tracking customers,
  - ✖ Exposing customer behaviour for commercial benefits.
Why anonymizing metering data?

According to NIST, “Smart Grid data should be anonymized wherever possible to limit the potential for computer matching of records.”

Different approaches have been proposed for data anonymization, including:

- Generalization
- Perturbation
- Pseudonimization
- Aggregation
Some open problems need to be solved:

- Multiple data Consumers,
- Low computational load on Meters,
- Frequent re-pseudonymization,
- Identity recovery, if necessary.
Three different sets of nodes are comprised in proposed architecture:

1. Producers, $p$
2. PPNs, $n$
3. Consumers, $c$
The pseudonymization protocol consists of a tuple of algorithms:

- Setup$(1^l) \rightarrow (k_d, \text{params})$
- pSend$(\text{param, } i, p, x_i^p) \rightarrow (e_i^p(1), \ldots, e_i^p(n), \ldots, e_i^p(N), ID_p, r_i^p)$
- PPNSend$(\text{param, } i, n, ID_p, r_i^p, e_i^p(n)) \rightarrow (PD_c^p, e_i^p(n))$
- cReceive$(\text{param, } i, c, PD_c^p, e_i^p(1), \ldots, e_i^p(N)) \rightarrow (PD_c^p, x_i^p)$

- The encryption algorithm used in pSend is the Shamir Secret Sharing Scheme, that we assume to be unconditionally secure.
✓ Full Pseudonymization:

\[ \Pr(full - p = 1) \leq \frac{1}{2} + \text{negl}(l) \]

✓ Full Pseudonymization with Perfect Forward Anonymity:

\[ \Pr(full - p - pfa = 1) \leq \frac{1}{2} + \text{negl}(l) \]

✓ Unconditionally Indistinguishable Encryption:

\[ \Pr(blind = 1) = \frac{1}{2} \]
The experiment for an algorithm $A$ and a parameter $l$, assumes an adversary Consumer $c^*$, and focuses on two Producers $ID1,ID2$.

- The Setup outputs the system parameters.

- The first and second Producers execute pSend and output the messages: $msg^1_1 ... msg^1_N, msg^2_1 ... msg^2_N$.

- Each PPN receives the two messages, and calls the PPNSend. Then each PPN sends two messages $pmsg^p_n$ with $p \in \{1,2\}$ to the Consumers.

- Finally each Consumer runs cReceive and obtains the measurement with the pseudonym.

- The malicious Consumer $c^*$ executes $A$ and outputs $p' \in \{1,2\}$.

- The output of the experiment is 1 if $p' = p$, and 0 otherwise.
We use a keyed one-way function with trapdoor $E_{K_e}(m,r) = y$ with the following properties:

- The Configurator generates public/private key pair and keeps the private key and distributes the public key.

- Our implementation of $E_{K_e}$ builds upon RSA with OAEP.

- Each PPN calculates the pseudonym as:

$$PD_c^p = E_{K_e}[ID_p || c || \frac{i}{\alpha}, \alpha, w_p^c]$$

- $ID_p$: Producer’s Identity
- $c$: Consumer’s Identity
- $i$: round identifier
- $\alpha$: length of the validity time span of pseudonym
- $w_p^c$: security nonce
Other security properties

- There exists a p.t. algorithm that, given the private key, can **recover the identity** of Producer from its pseudonym.
  - ✓ This property is consequence of Configurator having the private key.
- Before sending its data, the **Producer is aware** of the set of Consumer monitoring its data.
  - ✓ This happens thanks to the message *SpecifyMonitoringSet*.
- Given a pair of distinct Producers’ identities \((p, p')\) and the same Consumer \(c\), or a pair of distinct Consumers \((c, c')\) and the same Producer \(p\), the **output** of the function \(E_{k_e}\) is **always different**.
  - ✓ This property is consequence of using the ciphering function that relies on RSA with OAEP.
At every round, the following procedure is repeated:

1. Producer $p$ divides its measurements in $t$ shares and gives one share to each of the $t$ PPNs.
Communication Protocol (II)
Shamir Secret Sharing Scheme

At every round, the following procedure is repeated:

1. Producer $p$ divides its measurements in $t$ shares and gives one share to each of the $t$ PPNs.

2. PPN $n$ receives a share from Producer $p$ destined to Consumer $c$, computes the pseudonym basing on $p$ and $c$ and sends it to $c$ together with the share.
At every round, the following procedure is repeated:

1. Producer $p$ divides its measurements in $t$ shares and gives one share to each of the $t$ PPNs.

2. PPN $n$ receives a share from Producer $p$ destined to Consumer $c$, computes the pseudonym basing on $p$ and $c$ and sends it to $c$ together with the share.

3. Consumer $c$ combines the shares associated to the same pseudonym and recovers the measurements.
Comparison and Results

- We compare the performance of our proposed protocol with:

  ✓ **Mixing Scheme**: it moves the computational load on the Producer that computes its pseudonym and creates the mixing packet. The Producers encrypt with RSA and the PPNs only forward messages.

  ✓ **Proxy Re-Encryption Scheme**: it guarantees that a collusion of all PPN can’t obtain the relation between Producer’s measurement and identity. The Producers and PPNs encrypt with Paring based algorithm.

- We evaluate the number of sent messages and the computational cost in the three scheme.
## Results (I)

### Number of Sent Messages

<table>
<thead>
<tr>
<th>Asymptotic Values</th>
<th>Mixing Scheme</th>
<th>Shamir Secret Sharing</th>
<th>Proxy Re-Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messages sent by each Producer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>C</td>
<td>)</td>
<td>(</td>
</tr>
<tr>
<td>Messages sent by each PPN</td>
<td>(</td>
<td>P*</td>
<td>C</td>
</tr>
</tbody>
</table>
Results (I)
Number of Sent Messages

- **Sharing**
- **Mixing**
- **Re-encryption**

Number of messages sent by each PPN

Number of messages sent by each Producer

- $|P|=200$, $|N|=5$
## Results (II)
### Computational Costs

<table>
<thead>
<tr>
<th>Asymptotic Values</th>
<th>Mixing Scheme</th>
<th>Shamir Secret Sharing</th>
<th>Proxy Re-Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Producer</td>
<td>$</td>
<td>C</td>
<td>* \text{Cost}(\text{RSA}_{\text{enc}})$</td>
</tr>
<tr>
<td>PPN</td>
<td>$</td>
<td>P</td>
<td>*</td>
</tr>
<tr>
<td>Consumer</td>
<td>$</td>
<td>P</td>
<td>*\text{Cost(\text{RSA}_{\text{dec}})}$</td>
</tr>
</tbody>
</table>
## Results (II) Computational Costs

<table>
<thead>
<tr>
<th>C=10</th>
<th>Mixing</th>
<th>Sharing</th>
<th>Re-Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Producer</td>
<td>5,12 ms</td>
<td>0,10 ms</td>
<td>21,43 ms</td>
</tr>
<tr>
<td>PPN</td>
<td>-</td>
<td>-</td>
<td>&gt; 5 min</td>
</tr>
<tr>
<td>Consumer</td>
<td>4,86 s</td>
<td>2,11 s</td>
<td>&gt; 5 min</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C=50</th>
<th>Mixing</th>
<th>Sharing</th>
<th>Re-Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Producer</td>
<td>25,60 ms</td>
<td>0,10 ms</td>
<td>21,43 ms</td>
</tr>
<tr>
<td>PPN</td>
<td>-</td>
<td>-</td>
<td>&gt;&gt; 5 min</td>
</tr>
<tr>
<td>Consumer</td>
<td>4,86 s</td>
<td>2,11 s</td>
<td>&gt; 5 min</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C=100</th>
<th>Mixing</th>
<th>Sharing</th>
<th>Re-Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Producer</td>
<td>51,20 ms</td>
<td>0,10 ms</td>
<td>21,43 ms</td>
</tr>
<tr>
<td>PPN</td>
<td>-</td>
<td>-</td>
<td>&gt;&gt; 5 min</td>
</tr>
<tr>
<td>Consumer</td>
<td>4,86 s</td>
<td>2,11 s</td>
<td>&gt; 5 min</td>
</tr>
</tbody>
</table>

\[ |P| = 1000, |N| = 5 \]

Processor: 2.7 GHz Intel Core i7, Memory: 4GB 1333 MHz DDR3
We propose a pseudonymization protocol for smart metering measurements,

The protocol allows collecting metering data without revealing the association between users’ identities and their pseudonyms,

We described a possible implementation of the proposed algorithm,

This work evaluates the security guarantees and the performance the algorithm achieves.

We compare three different solutions in terms of number of sent messages and computational costs.

Results show that the most suitable protocol is the one based on pseudonymization with the Shamir secret sharing scheme.
THANK YOU
Appendix: Related Work (I)

- Solutions proposed for protecting user data in AMI (Advanced Metering Infrastructure):
  - Zero-knowledge cryptographic protocols [1],
  - Data aggregation [2],
  - Escrow services [3].

- Our solution has been proposed in:


At every round, the following procedure is repeated:

1. Producer $p$ generates the measurements and computes its pseudonym. It creates the mixing packet, composed by measurement and pseudonym, that is sent to a randomly chosen PPN $n$.

2. PPN $n$ forwards the packet to the Consumer $c$, to whom the message is destined.

3. Consumer $c$ recovers the individual data by decrypting the packets.
Proxy Re-Encryption Scheme

At every round, the following procedure is repeated:

1. Producer $p$ encrypts its measurements based on the identity of PPN, to whom the message is destined.

2. PPN $n$ computes the pseudonym basing on $p$ and $c$, re-encrypts the packet and sends it to $c$.

3. Consumer $c$ recovers the individual data by double decrypting the packets.