Privacy and Security in Aggregation Protocols

Iraklis Leontiadis
joint work with Kaoutar Elkhiyaoui
and Refik Molva
What this talk is not about
Outline

- Privacy in Aggregation Protocols with dynamicity and stronger privacy model
- Verifiability in Aggregation
Problem

Energy forecasting

Medical data analysis

User Profiling

Overall Transactions

\[ \sum_{i=0}^{n} x_i = ? \]
Existing solutions

- Trusted Aggregator
  - Unrealistic

- Secret sharing [ET2012]
  - Increased communication costs

  - Orthogonal to our goal

- Trusted Dealer [SCRCS2011, JL2013, GMP2014]
  - Strong trust assumption
State of the Art [SCRCS2011]

- **Setup(k):**
  - $\mathbb{G}$ a cyclic group with a generator $g$ and prime order $p$
  - Trusted Dealer distributes:
    - secret keys $sk_i \in \mathbb{Z}_p$.
    - $sk_0 = - \sum_{i=1}^{n} sk_i$ to the Aggregator.
    - $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{G}$

- **Encrypt($x_i, t$):**
  - $c_{i,t} = g^{x_i H(t)^{sk_i}} \in \mathbb{Z}_p$

- **Aggregate:**
  - $V = H(t)^{sk_0} \prod_{i=1}^{n} c_{i,t} = g^{\sum_{i=1}^{n} c_{i,t}} g \in \mathbb{Z}_p$
  - $\sum_{i=1}^{n} c_i = \log_g(V)$

- Vulnerable to user failures
- No dynamicity
- Expensive decryption
- Fully trusted dealer
State of the Art contd. [JL2013]

- Setup(k):
  - $N = pq$ for primes $p, q$ ($l$ the size of $N$)
  - Trusted Dealer distributes:
    - secret keys $sk_i \in \{0,1\}^{2l}$ to the users.
    - $sk_0 = -\sum_{i=1}^{n} sk_i$ to the Aggregator.
    - $H(\cdot): \mathbb{Z}_N \to (\mathbb{Z}_N^2)^*$

- Encrypt($x_i, t$):
  - $c_{i,t} = (1 + x_{i,t}N)H(t)^{sk_i} \mod N^2$

- Aggregate:
  - $V_t = H(t)^{sk_0} \prod_{i=1}^{n} c_{i,t} = (1 + \sum_{i=1}^{n} x_{i,t} N) \mod N^2$
  - $\sum_{i=1}^{n} x_{i,t} = \frac{V_t - 1}{N} \in \mathbb{Z}$

- Vulnerable to user failures
- No dynamicity
- Fully trusted dealer
Drawbacks of previous solutions

- **Functionality**
  - No Dynamic group management
  - Not resilient to user failures

- **Privacy**
  - Fully trusted key dealer
Idea of Solution

- Users generate their secret keys.
- Semi trusted Collector.
- Blinded secret keys.
Privacy Requirements

- **Aggregator obliviousness:**
  - *Aggregator* learns nothing but the aggregate.

- **Collector obliviousness:**
  - *Collector* learns nothing.
Functionality Enhancements

- Dynamicity
Functionality Enhancements

- Dynamicity
- Fault-Tolerance
Tools

1. JL encryption.
2. Self-generated keys (by users).
3. Responsibility splitting mechanism.
Our scheme

\[ aux_{i,t} = H(t)^{sk_A sk_i} \]
\[ c_{i,t} = (1 + x_{i,t}N)H(t)^{sk_i} \mod N^2 \]

1. \( P_t = \prod_{i=1}^{n} (c_{i,t})^{sk_A} = (1 + N \sum_{i=1}^{n} x_{i,t})^{sk_A} H(t) \sum_{i=1}^{n} (x_{i,t} sk_A) \mod N^2 \)

2. \( I_t = \frac{aux_{1,t}^{-1}}{N} \)

3. \( \sum_{i=1}^{n} x_{i,t} = I_t sk_A^{-1} \mod \mathbb{Z}_N \)

\[ sk_A \in \{0,1\}^{2l} \]

\[ pk_A = H(t)^{sk_A} \]
Dynamicity and Resiliency to failures

\[ c_{i,t} = (1 + x_{i,t}N)H(t)^{sk_I} \mod N^2 \]

\[ aux_{i,t} = H(t)^{sk_A^{sk_I}} \]

\[ aux_t = \prod_{i=1}^{n+1} H'(t)^{sk_A^{sk_I}} \]

\[ pk_A = H(t')^{sk_A} \]

\[ sk_A \in \{0,1\}^{2l} \]

\[ sk_1 \]

\[ sk_2 \]

\[ sk_3 \]

\[ sk_4 \]

\[ sk_n \]

\[ sk_{n+1} \]

\[ sk_{n+2} \]

\[ sk_{n+3} \]

\[ sk_{n+4} \]
Privacy analysis

- **Aggregator Obliviousness** based on:
  - DCR in \((\mathbb{Z}/\mathbb{Z}_N^2)^*\),

- **Collector Obliviousness** based on:
  - DCR in \((\mathbb{Z}/\mathbb{Z}_N^2)^*\),
  - QR in \(\mathbb{Z}_N^*\)
  - DDH in the subgroup of QR in \((\mathbb{Z}_N)^*\)
Evaluation

- Theoretical

<table>
<thead>
<tr>
<th>Entity</th>
<th>Computation</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>$2 \text{ EXP} + 1 \text{ MULT} + 1 \text{ ADD} + 1 \text{ HASH}$</td>
<td>$4 \cdot 1$</td>
</tr>
<tr>
<td>Aggregator</td>
<td>$2 \text{ EXP} + 2 \text{ DIV} + (n - 1) \text{ MULT} + 1 \text{ HASH}$</td>
<td>$2 \cdot 1$</td>
</tr>
<tr>
<td>Collector</td>
<td>$(n - 1) \text{ MULT}$</td>
<td>$2 \cdot 1$</td>
</tr>
</tbody>
</table>

- Experimental (Charm framework on Python 3.2.3, Intel Core i5 CPU M 560 @ 2.67GHz 4Cores with 8GB of memory running Ubuntu 12.04 32bit)

<table>
<thead>
<tr>
<th>Values</th>
<th>[1-10]</th>
<th>[1-100]</th>
<th>[1-1000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>$110.13 \mu s$</td>
<td>$112.23 \mu s$</td>
<td>$114.57 \mu s$</td>
</tr>
<tr>
<td>1024</td>
<td>$116.50 \mu s$</td>
<td>$117.15 \mu s$</td>
<td>$118.34 \mu s$</td>
</tr>
<tr>
<td>2048</td>
<td>$116.99 \mu s$</td>
<td>$118.23 \mu s$</td>
<td>$120.83 \mu s$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Users</th>
<th>350</th>
<th>700</th>
<th>1000</th>
<th>2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>0.26 s</td>
<td>2.40 s</td>
<td>9.65 s</td>
<td>49.92 s</td>
</tr>
<tr>
<td>1024</td>
<td>0.65 s</td>
<td>5.82 s</td>
<td>24.16 s</td>
<td>123.19 s</td>
</tr>
<tr>
<td>2048</td>
<td>1.01 s</td>
<td>9.37 s</td>
<td>39.34 s</td>
<td>198.12 s</td>
</tr>
</tbody>
</table>

Encryption time per user | Aggregation time
Recap

- Aggregation of time series data
  - Fast
  - Dynamic
  - Resilient to user failures
  - Relaxed trust assumption

- How?
  - $(1 + N)^x = 1 + Nx \mod N^2$ [JL2013].
  - Users generate keys independently.
  - Responsibility splitting mechanism.
  - Untrusted collector.

Easily Extended for: AVG, VAR, STD, RMS
Verifiability in Aggregation

- Energy forecasting
- Medical data analysis
- User Profiling
- Overall Transactions

\[ \sum_{k=0}^{n} x_i \]
Existing approaches

- Homomorphic MACS [AB2009, CF2013, GW2013]
- Verifiable Outsourced Computations
- Stateless: Disconnect data owner from verification
- Oblivious: Verifiability information preserves obliviousness
- Aggregation: Multiple users contribute to computation.
Idea

- Homomorphic tags
- Rerandomize encoding
- Trusted key dealer
SOVAP Description

- **Setup(1^k):**
  - Trusted Dealer distributes:
    - secret keys $sk_i, a \in \mathbb{Z}_p$ to users.
    - $(p, g_2, g, G, G_2, G_T, g_2 \sum_{i=1}^{n} sk_i, g_2^a)$

- **Tag($x_i, t, sk_i, a$):**
  - $tag = g^a H(t)^{sk_i} \in \mathbb{Z}_p$

- **Aggregate($\{tag\}$):**
  - proof$= \prod_{i=1}^{n} tag_{i,t} = H(t)^{\sum_{i=1}^{n} sk_i} g^asum$

- **Verify(sum, proof)**
  
  $e(proof, g_2) \overset{?}{=} e(H(t), g_2^{\sum_{i=1}^{n} sk_i}) e(g^{sum}, g_2^a)$
Security

- Untrusted Aggregator, honest verifier

- An untrusted Aggregator cannot forge a valid proof:
  - Proof’ s.t Verify(proof’, sum’) = ✓

- Proof based on DL in $G_2$
Privacy

- Honest but Curious Verifier, Aggregator

- Tags achieve Obliviousness for the sum.

- Proof based on computational hiding property of Pedersen Commitments.
Recap

- Privacy with Dynamicity and relaxed trust assumptions
  - Semi trusted collector
  - Secret keys independently chosen
  - Responsibility splitting mechanism

- Verifiability in Aggregate in protocols
  - Homomorphic tags.
  - Rerandomization of value encoding.
Looking Ahead

- Advanced aggregate functionalities
- Collusion resistant (?)
- Verifiability with dynamicity
- Extra security properties: Anonymity for users.
Questions?

Thank you!!!
Iraklis Leontiadis
leontiad@eurecom.fr
References

- [LEM2014]: Private and Dynamic Time-Series Data Aggregation with Trust Relaxation, CANS2014
- [GMP2014]: Privacy-Enhanced Participatory Sensing with Collusion Resistance and Data Aggregation, CANS2014
- [ET2012]: Private Computation of Spatial and Temporal Power Consumption with Smart Meters, ACNS 2012
- [CSS2012]: Privacy-Preserving Stream Aggregation with Fault Tolerance, Financial Cryptography 2012
- [RN2010]: Differentially private aggregation of distributed time-series with transformation and encryption, SIGMOD Conference 2010
- [SCRCS2011]: Privacy-Preserving Aggregation of Time-Series Data, NDSS 2011
- [JL2013]: A Scalable Scheme for Privacy-Preserving Aggregation of Time-Series Data, Financial Cryptography 2013
References cont’d

- [AB2009]: Homomorphic macs: Mac-based integrity for network coding, *ACNS 2009*
- [CF2013]: Practical homomorphic macs for arithmetic circuits, *EUROCRYPT 2009.*
- [GW2013]: Fully homomorphic message authenticators, *ASIACRYPT 2013.*
- [JMSW2002]: Homomorphic signature schemes, *CT-RSA 2002*