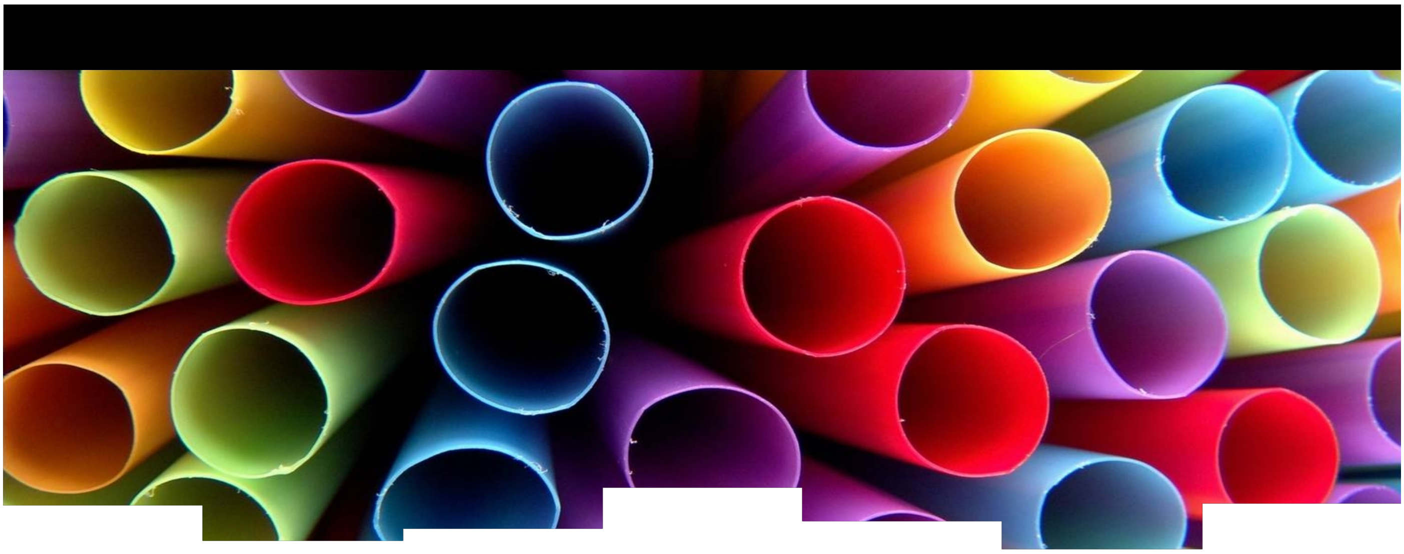
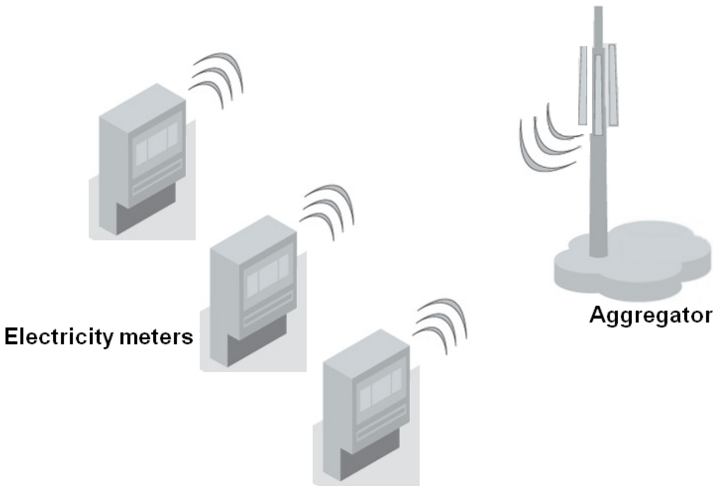
A Scalable Scheme for Privacy-Preserving Aggregation of Time-Series Data



Marc Joye · Benoît Libert

technicolor

Smart Metering



- Frequent aggregates of consumption over a population of users is useful to finely tune the service
  - better prices by adapting the load or forecasting the supply
  - useful to rapidly detect anomalies on the grid
- BUT privacy issues. . .

Trusted Aggregator

Easy case

- User i encrypts her input x_i under the aggregator's public key pk and sends $c_i = \text{Encrypt}_{pk}(x_i)$
- Aggregator decrypts each c_i, obtains $\{x_i\}_{i=1}^{n}$ and reveals the aggregate $\sum_{i=1}^{n} x_i$

Additively Homomorphic Encryption

$$\text{Encrypt}_{pk}(x_i) \star \text{Encrypt}_{pk}(x_j) = \text{Encrypt}_{pk}(x_i + x_j)$$

- ElGamal's cryptosystem

Sender		**Receiver**
	$\xleftarrow{pk}$	$pk = (\langle g \rangle, q, h, y = g^s)$
random $r \in_R \mathbb{Z}_q$		
$c_{i,1} = g^r$ and $c_{i,2} = h^{x_i} y^r$	$\xrightarrow{(c_{i,1}, c_{i,2})}$	$h^{x_i} = c_{i,2} {c_{i,1}}^{-s}$
		$\Longrightarrow x_i = \ldots$
		(requires x_i to be "small")

- Remark: $(c_{i,1}, c_{i,2}) \star (c_{j,1}, c_{j,2}) = (c_{i,1} \cdot c_{j,1}, c_{i,2} \cdot c_{j,2})$
  $= (g^{r+r'}, h^{x_i+x_j} y^{r+r'})$

Additively Homomorphic Encryption

$$\text{Encrypt}_{pk}(x_i) \star \text{Encrypt}_{pk}(x_j) = \text{Encrypt}_{pk}(x_i + x_j)$$

- Paillier's cryptosystem

Sender | **Receiver**

$$pk = N,\ sk = \lambda$$

$$\xleftarrow{pk}$$

random $r \in_R \mathbb{Z}_N^*$

$$c_i = (1+N)^{x_i} r^N \bmod N^2 \quad \xrightarrow{c_i} \quad {c_i}^{\lambda} \equiv (1 + x_i \lambda N) \pmod{N^2}$$

$$\implies x_i = \ldots$$

- Remarks: (1) $c_i \star c_j = c_i \cdot c_j$
  $= (1+N)^{x_i+x_j}(rr')^N \pmod{N^2}$
  (2) $(1+N)^x \equiv 1 + Nx \pmod{N^2}$

Trusted Aggregator (Optimized Version)

Easy case

- User i encrypts her input x_i under the aggregator's public key pk and sends $c_i = \text{Encrypt}_{pk}(x_i)$
- Aggregator computes $C = \prod_{i=1}^{n} c_i$, decrypts C and reveals the aggregate $\sum_{i=1}^{n} x_i$

Limitation

Aggregator must be trusted not to reveal the x_i's

Untrusted Aggregator

More difficult case

Can we do the same without trusting the aggregator?

- The problem: make sure that aggregator only obtains $\sum_{i=1}^{n} x_i$

technicolor

Solution #1: Splitting Roles

Add an extra party: the Crypto Service Provider (CSP)

1. User i encrypts her input x_i under the CSP's public key pk and sends $c_i = \mathrm{Encrypt}_{pk}(x_i)$ to the aggregator
2. Aggregator aggregates the c_i's as
$$C = \prod_{i=1}^{n} c_i = \mathrm{Encrypt}_{pk}\left(\sum_{i=1}^{n} x_i\right)$$
and sends C to the CSP
3. CSP decrypts C and returns the aggregate $\sum_{i=1}^{n} x_i$

technicolor

Solution #2: Secret Sharing

Distribute the decryption capability among two (or more) parties

Solution #3: Interaction

Users sequentially aggregate their inputs

1. Centralized aggregator holds a public key *pk*
2. User *i* receives an encrypted "aggregate-so-far"

$$C_{i-1} = \text{Encrypt}_{pk}(x_1 + x_2 + \ldots + x_{i-1})$$

   and appends its input, $C_i = C_{i-1} \star \text{Encrypt}_{pk}(x_i)$, and send C_i to user $i+1$
3. Aggregator receives $C_n = \text{Encrypt}_{pk}(x_1 + x_2 + \ldots + x_n)$, decrypts it, and obtains $\sum_{i=1}^{n} x_i$

Solution #4: Aggregator-Oblivious Evaluation

Shi-Hubert-Chan-Rieffel-Chow-Song (NDSS 2011)

- Aggregator obtains the sum $\sum_{i=1}^{n} x_i$ and nothing else, without interaction
- Involves a setup phase where an off-line TTP provides secret keys to users and aggregator
- AO security: corrupting some users and/or the aggregator only reveals partial sums $\sum_{i \in \mathcal{C}} x_i$ for some $\mathcal{C} \subset \{1, \ldots, n\}$

Shi *et al.* Scheme

Setup phase

- Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q and $H : \{0,1\}^* \to \mathbb{G}$ be a hash function
- Aggregator has $s_0 \in \mathbb{Z}_q$ and user i gets $s_i \in \mathbb{Z}_q$ s.t. $\sum_{i=0}^{n} s_i = 0$

Aggregation

- At period t, user i encrypts x_i as

$$c_i = g^{x_i} \cdot H(t)^{s_i}$$

- Aggregator computes $C = H(t)^{s_0} \cdot \prod_{i=1}^{n} c_i$ and obtains

$$C = g^{\sum_{i=1}^{n} x_i} \cdot H(t)^{\sum_{i=0}^{n} s_i} = g^{\sum_{i=1}^{n} x_i}$$

which yields $\sum_{i=1}^{n} x_i$

Analysis

Theorem (Shi *et al.*, NDSS 2011)

The scheme provides aggregator-oblivious security under the ***Decision Diffie-Hellman*** *(DDH) assumption (in the random oracle model)*

DDH: *Given* $(g, g^a, g^b, T) \in \mathbb{G}^3$*, no PPT algorithm can decide if* $T = g^{ab}$ *or* $T \in_R \mathbb{G}$

Drawbacks

- Decryption is somewhat expensive
  - requires the computation of a discrete log in DDH group $\mathbb{G}$
- Security reduction is pretty loose
  - includes a degradation factor of $O(n^3)$

$\Longrightarrow$ the scheme is not scalable

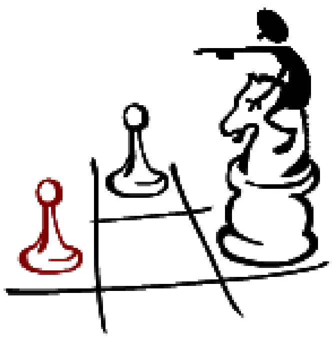
Our Solution: Scalable AO Scheme

- Challenge left open by Shi *et al.*(NDSS 2011):
  - How to effciently compute sums over large plaintext spaces?
  - Investigate other algebraic settings
- Natural candidates: Paillier-like systems

Problem

The proof offered by Shi *et al.* does not extend to this setting

- requires DDH groups of known (prime) orders



Our Solution: Scalable AO Scheme (2)

Setup phase

- Let $N = pq$ and $H : \{0,1\}^* \rightarrow \mathbb{Z}^*_{N^2}$ be a hash function
- Aggregator has $s_0 \in \pm\{0,1\}^{2\ell}$ and user i gets $s_i \in \pm\{0,1\}^{2\ell}$ s.t. $\sum_{i=0}^{n} s_i = 0$ (over $\mathbb{Z}$)

Aggregation

- At period t, user i encrypts x_i as

$$c_i = (1+N)^{x_i} \cdot H(t)^{s_i} \bmod N^2$$

- Aggregator computes $C = H(t)^{s_0} \cdot \prod_{i=1}^{n} c_i \bmod N^2$ and obtains

$$C \equiv (1+N)^{\sum_{i=1}^{n} x_i} \cdot H(t)^{\sum_{i=0}^{n} s_i} \equiv 1 + N \cdot (\textstyle\sum_{i=1}^{n} x_i) \pmod{N^2}$$

which yields $\sum_{i=1}^{n} x_i$ as $(C-1)/N$

Security Analysis

Theorem

The scheme provides aggregator-oblivious security under the ***Decision Composite Residuosity*** *(DCR) assumption (in the ROM)*

DCR: *Given $N = pq$, no PPT algorithm can distinguish the distibutions*

$$\mathcal{D}_0 = \{x^N \bmod N^2 \mid x \in_R \mathbb{Z}^*_{N^2}\} \quad \textit{and} \quad \mathcal{D}_1 = \{x \in_R \mathbb{Z}^*_{N^2}\}$$

- Tighter security reduction (degradation of $O(q_{enc})$)
  - i.e., indep. of the number of users
- Reminder: Shi *et al.* scheme has a degradation of $O(n^3 \cdot q_{hash})$

Security Proof

- Sketch: use a sequence of games
  - Game 0: real game
  - Game 1: change the output of random oracle queries
    - If $t = t^\star$, set $H(t^\star) \in_R \mathbb{Z}_{N^2}^*$
    - If $t \neq t^\star$, set $H(t)$ as a N-th residue
- Under DCR assumption, Game 0 and Game 1 are indistinguishable
- Key observation: in Game 1, no information about $\{s_i \bmod N\}_{i \in \mathcal{S}^\star}$ is leaked to the adversary

  $\Rightarrow$ $c_{i,t^\star} = (1+N)^{x_i} \cdot H(t^\star)^{s_i}$ perfectly hides x_i

Summary

Aggregator-oblivious aggregation is now scalable and practical

- Fast decryption (aggregation)
- On-line/off-line efficiency: only one on-line multiplication
- Tighter security (independent of the number of users)

Some research problems

- Eliminate the random oracle model
- Compute other statistics (weighted sum, standard deviation, etc.) without interaction