What Could Deskside Supercomputers Do For The Power Grid?

Franz Franchetti

ECE, Carnegie Mellon University
`www.spiral.net`

Co-Founder, SpiralGen
`www.spiralgen.com`

**Joint work with
Tao Cui and Cory Thoma**

This work was supported by NSF, SRC (SGRC), and Intel

Computing Resources In Smart Grids

1 flop/s = one floating-point operation (addition or multiplication) per second

mega (M) = 10^6, giga (G) = 10^9, tera (T) = 10^{12}, peta (P) = 10^{15}, exa E) = 10^{18}

Computing systems in 2010

Cell phone
1 CPUs
1 Gflop/s
$300
1 W power

Laptop
2 CPUs
20 Gflop/s
$1,200
30 W power

Server
12 CPUs, 2 GPUs
2 Tflop/s
$10,000
1 kW power

HPC
200 CPUs
20 Tflop/s
$700,000
8 kW power

#1 supercomputer
224,162 CPUs
2.3 Pflop/s
$100,000,000
7 MW power

Power grid computing resources

Power grid scenario

- Central servers (planning, contingency analysis)
- Autonomous controllers (smart grids)
- Operator workstations (decision support)

Outline

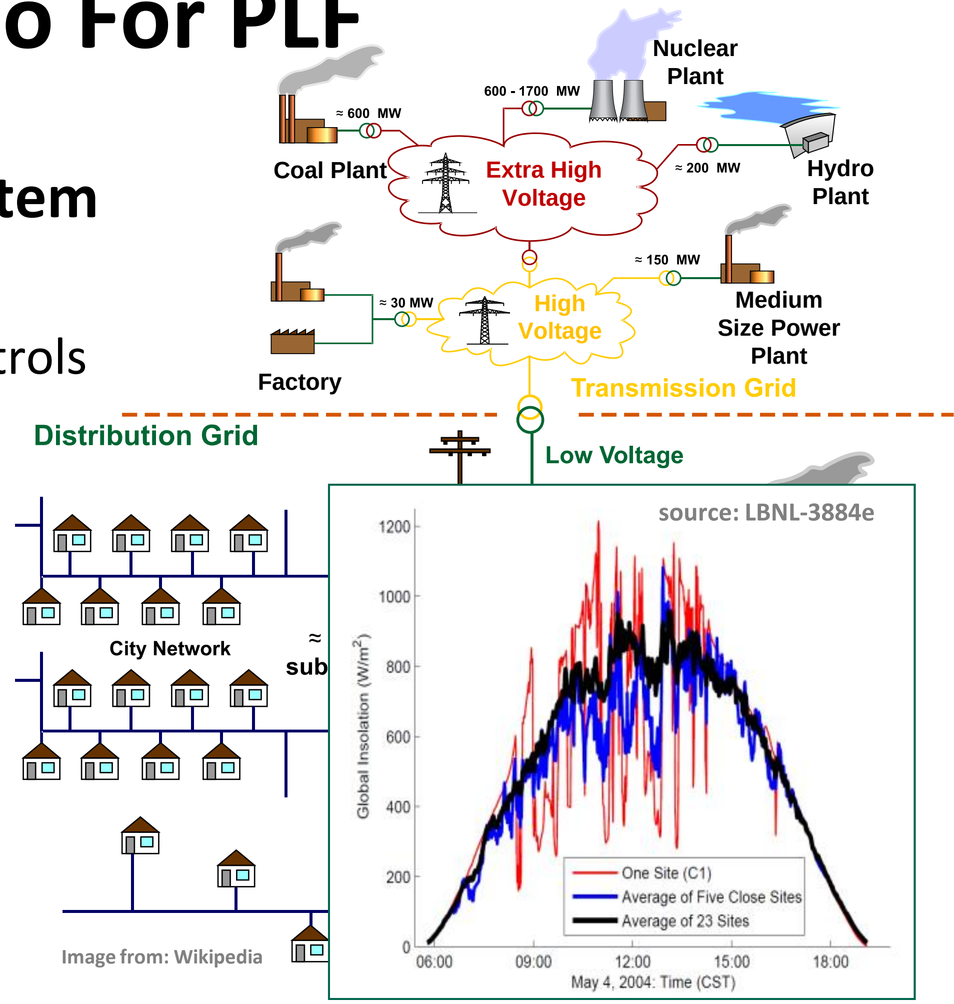
- **Example 1: Probabilistic Power Flow**
- Example 2: Privacy-Preserving Smart Meter
- Summary

T. Cui and F. Franchetti, **A Multi-Core High Performance Computing Framework for Distribution Power Flow.** The 43rd North American Power Symposium (NAPS), Boston, USA, Aug 2011.

T. Cui and F. Franchetti, **A Multi-Core High Performance Computing Framework for Probabilistic Solutions of Distribution Systems.** IEEE PES General Meeting 2012, San Diego, CA, USA.

Real-Time Monte Carlo For PLF

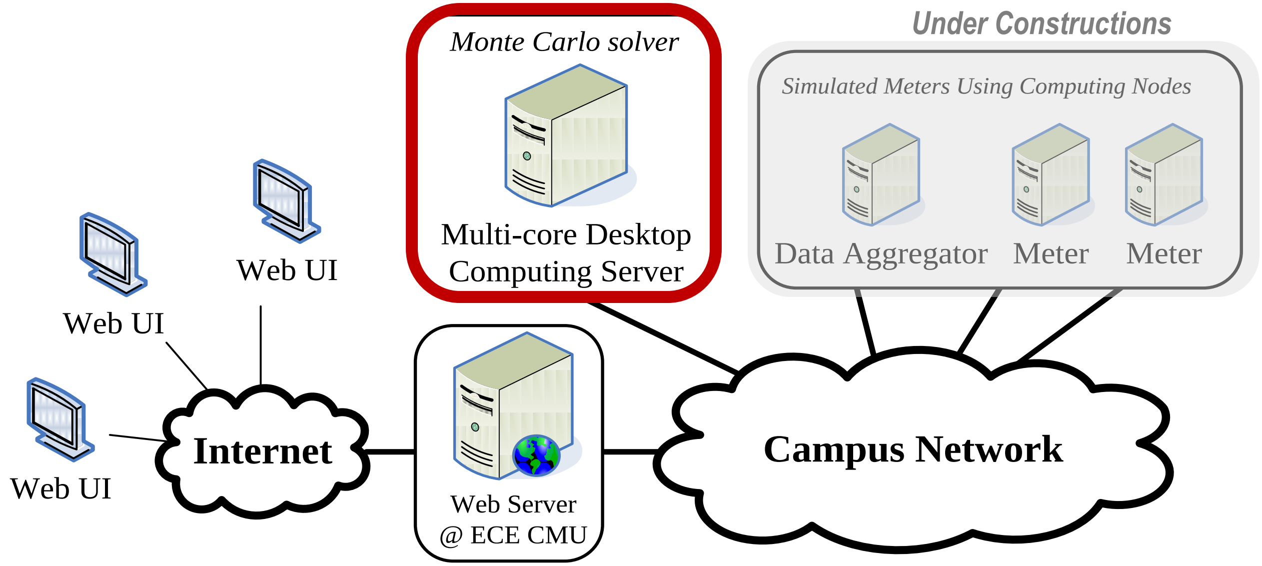
- **Conventional Distribution System**
  - Passively receiving power
  - Few real time monitoring or controls
- **Smart Distribution System**
  - AMI, smart meters, intelligence
  - Renewable energy resources
  - Large load (e.g. PHEVs)
- **Uncertainties**
  - Solar, wind, stochastic in nature
  - Load with large variance, etc.



- **An Efficient Computational Tool for Dealing the Uncertainties**

Design And Implementation

- **Distribution System Probabilistic Monitoring System (DSPMS)**
  - System Structure:



- MCS solver running on Multi-core Desktop Server (Code optimization)
- Results published via ECE Web Server (TCP/IP socket, SSH)
- Web based dynamic User Interface by client side scripts (JavaScript)
- Simulate meters using computing nodes (Python + Twisted)

Parallelization And Optimization

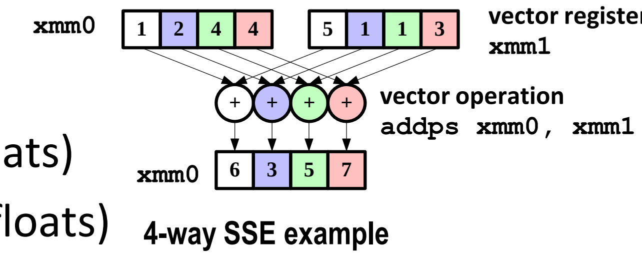
- **SIMD parallelization**

  SIMD: Single Instruction Multiple Data

  SSE: Streaming SIMD Extensions (128 bit, 4 floats)

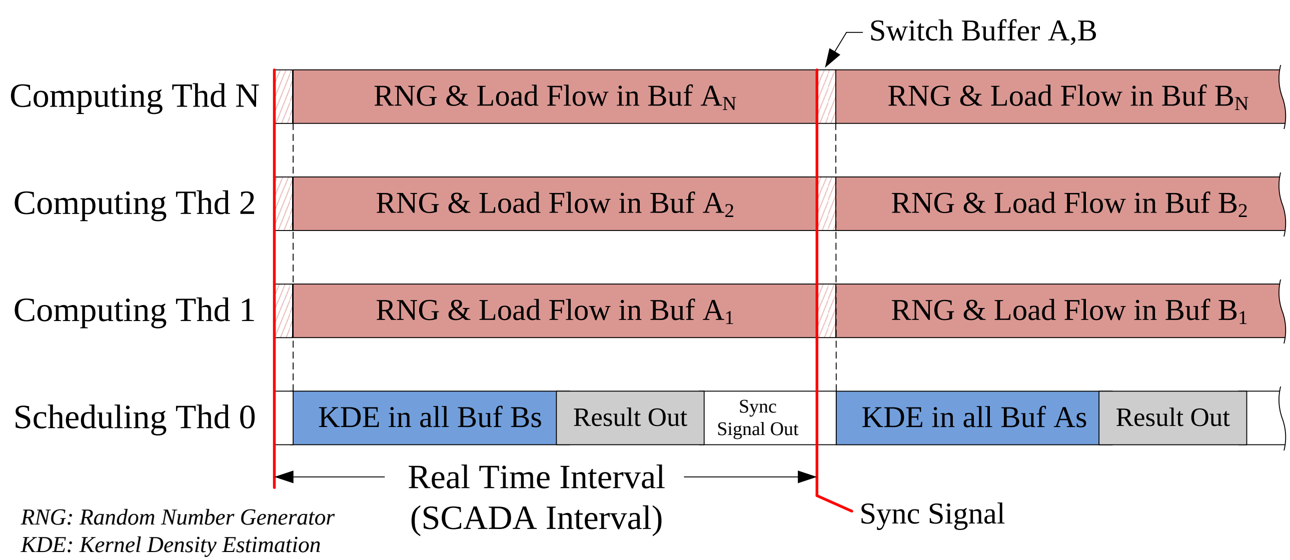
  AVX: Advanced Vector eXtensions (256 bit, 8 floats)

  MIC (512 bit, 16 floats)



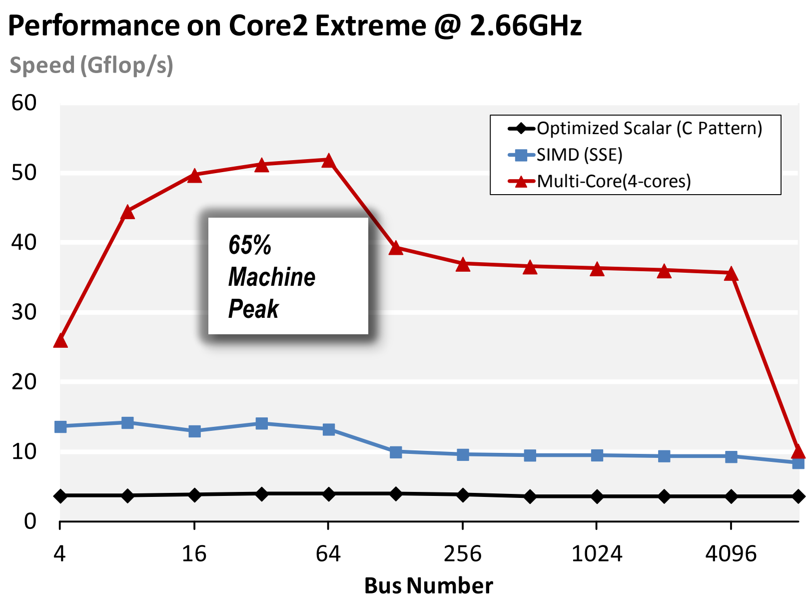
- **Multicore parallelization**

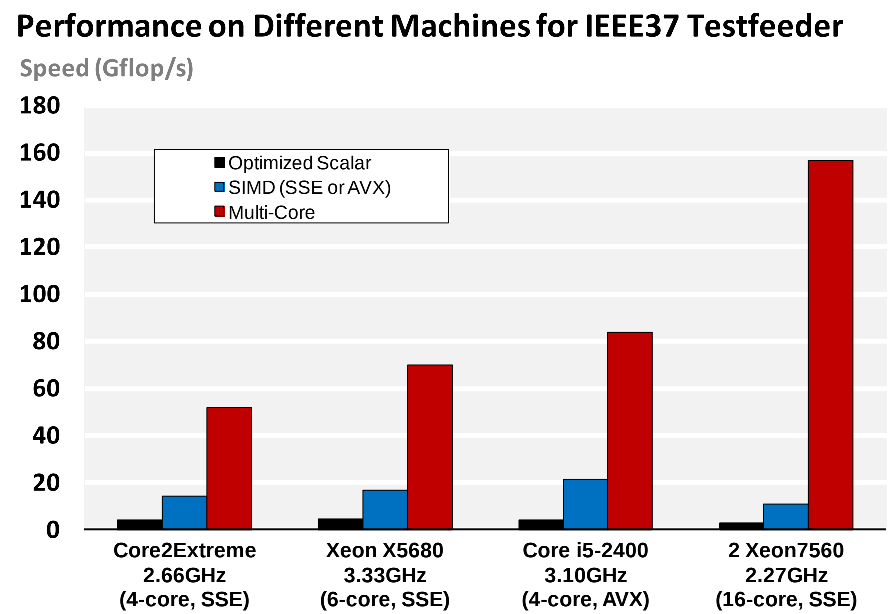
  SMP and SMT parallelism, thread pools



Performance Results: 1M PLF/s per Second

- Performance of Optimized Code, Mass Amount Load Flow



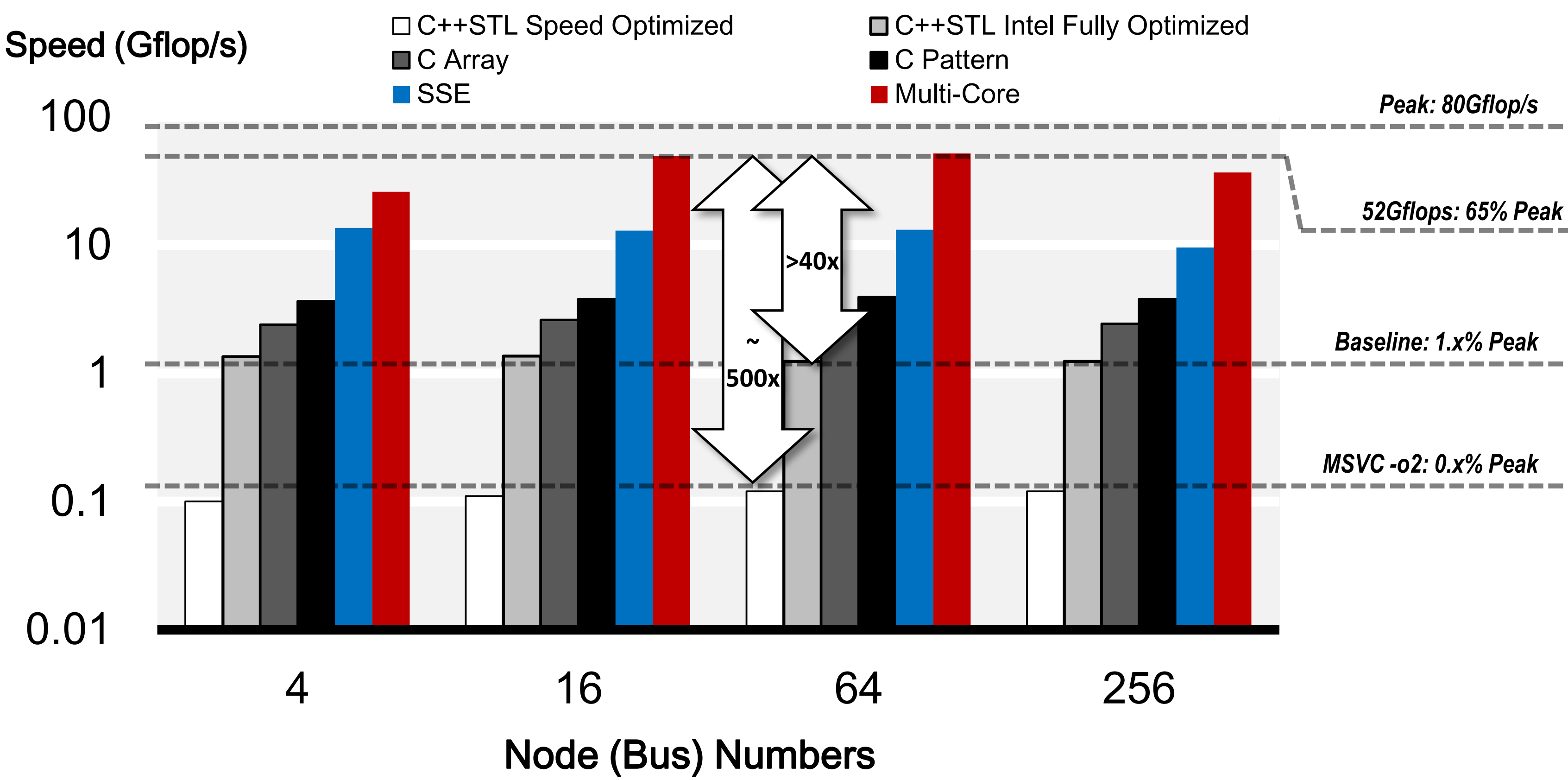


Translate speed (Gflop/s) into run time:

Problem Size (IEEE Test Feeders)	Approx. flops	Approx. Time / Core2 Extreme	Approx. Time / Core i5	Baseline. C++ ICC –o3 (~300x faster then Matlab)	Comments
IEEE37: one iteration	12 K	~ 0.3 us	~ 0.3 us		
IEEE37: one load flow (5 Iter)	60 K	~ 1.5 us	~ 1.5 us		0.01 kVA error
IEEE37: 1 million load flow	60 G	~ < 2 s	~ < 1 s	~ 60 s (>5 hrs Matlab)	SCADA Interval: 4 seconds
IEEE123: 1 million load flow	200 G	~ < 10 s	~ < 3.5 s	~ 200 s (>15 hrs Matlab)	

Optimization Impact

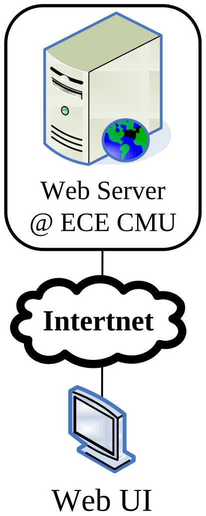
Optimization Gain on Core 2 Extreme (4-core SSE)

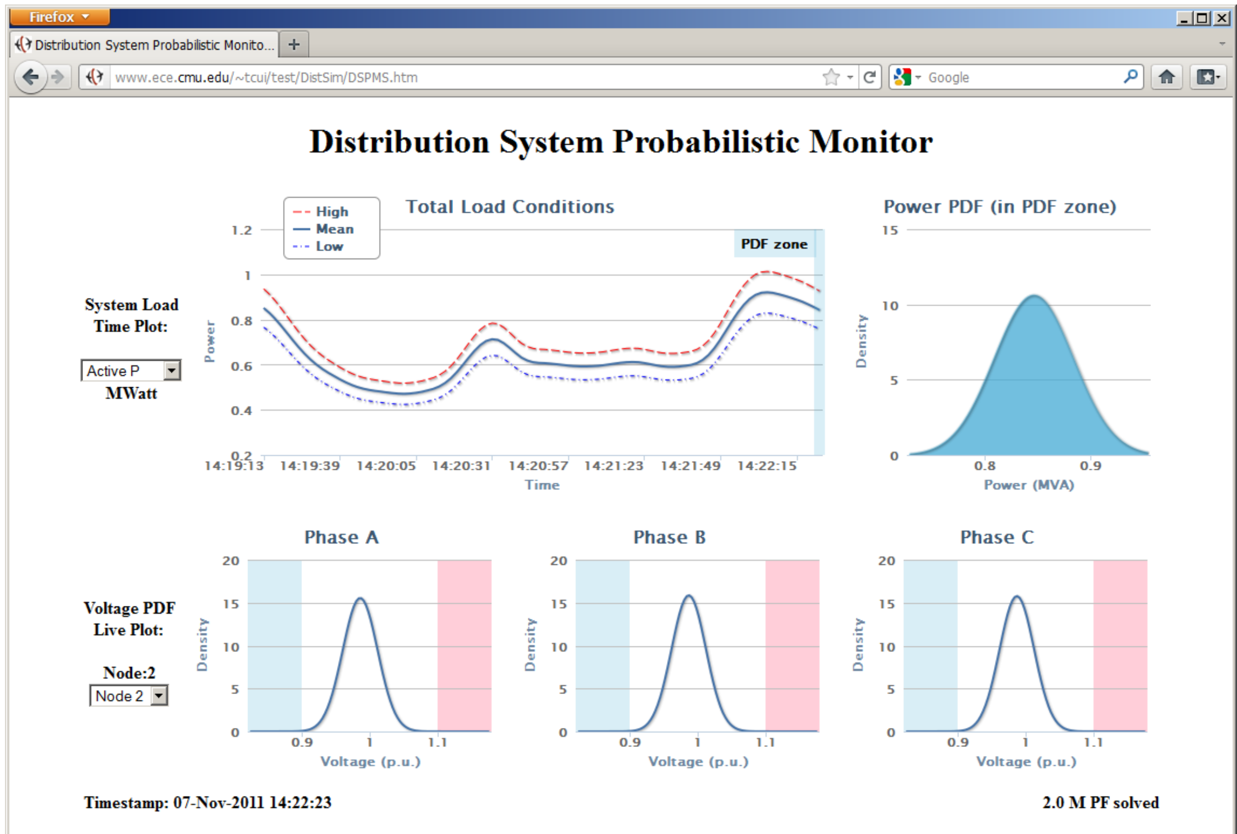


Duplicating & connecting multiple IEEE 4 Node Testfeeder

System Prototype

- **Distribution System Probabilistic Monitoring System (DSPMS)**
  - Web Server and User Interface





- Demo Link: www.ece.cmu.edu/~tcui/test/DistSim/DSPMS.htm

Outline

- Example 1: Probabilistic Power Flow
- **Example 2: Privacy-Preserving Smart Meter**
- Summary

A "Cell Phone" Plan for Electricity

"Cell phone"-like plan

- **Off-peak $/kWh:** Cost per kWh in "off-peak" condition
- **Peak kWh:** pre-bought allowance of peak kWh
- **Excess peak $/kWh:** Cost per peak kWh once the allowance is exhausted

Billing Algorithm

- **Off-peak mode:** Total power consumed in the subnet is under the critical threshold
- **Peak mode:** Total power consumed in the subnet is exceeding the critical threshold
  Above-average customers enter peak mode
  Below-average customers stay in off-peak mode
- **Self-regulating:** Customers either pay for expensive peak power or drop below average
- **Privacy Issue: Requires real-time power measurement from customers**

Solution: Secure Multiparty Computation

- Arithmetic using real-time power without leaking customer's consumption
- Customers can check that utility does not cheat
- Utility can check that customer did not cheat

Technology: Public Key Cryptosystems

Asymmetric encryption

- **Publicly key:** Send to anybody in directory by encrypting with user's public key
- **Private key:** only the recipient can decrypt the messages with private key
- **Digital signature:** When signed with private key, everybody can verify authenticity

Example: ElGamal Encryption (based on discrete logarithm)

Key generation [edit]

The key generator works as follows:

- Alice generates an efficient description of a multiplicative cyclic group G of order q with generator g. See below for a discussion on the required properties of this group.
- Alice chooses a random x from $\{0, \ldots, q-1\}$.
- Alice computes $h = g^x$.
- Alice publishes h, along with the description of G, q, g, as her **public key**. Alice retains x as her **private key** which must be kept secret.

Encryption [edit]

The encryption algorithm works as follows: to encrypt a message m to Alice under her public key (G, q, g, h),

- Bob chooses a random y from $\{0, \ldots, q-1\}$, then calculates $c_1 = g^y$.
- Bob calculates the shared secret $s = h^y$. Since a new s is computed for every message s is also called an ephemeral key.

The steps above can be computed ahead of time.

- Bob converts his secret message m into an element m' of G.
- Bob calculates $c_2 = m' \cdot s$.
- Bob sends the ciphertext $(c_1, c_2) = (g^y, m' \cdot h^y) = (g^y, m' \cdot (g^x)^y)$ to Alice.

Decryption [edit]

The decryption algorithm works as follows: to decrypt a ciphertext (c_1, c_2) with her private key x,

- Alice calculates the shared secret $s = c_1^x$
- and then computes $m' = c_2 \cdot s^{-1}$ which she then converts back into the plaintext message m.

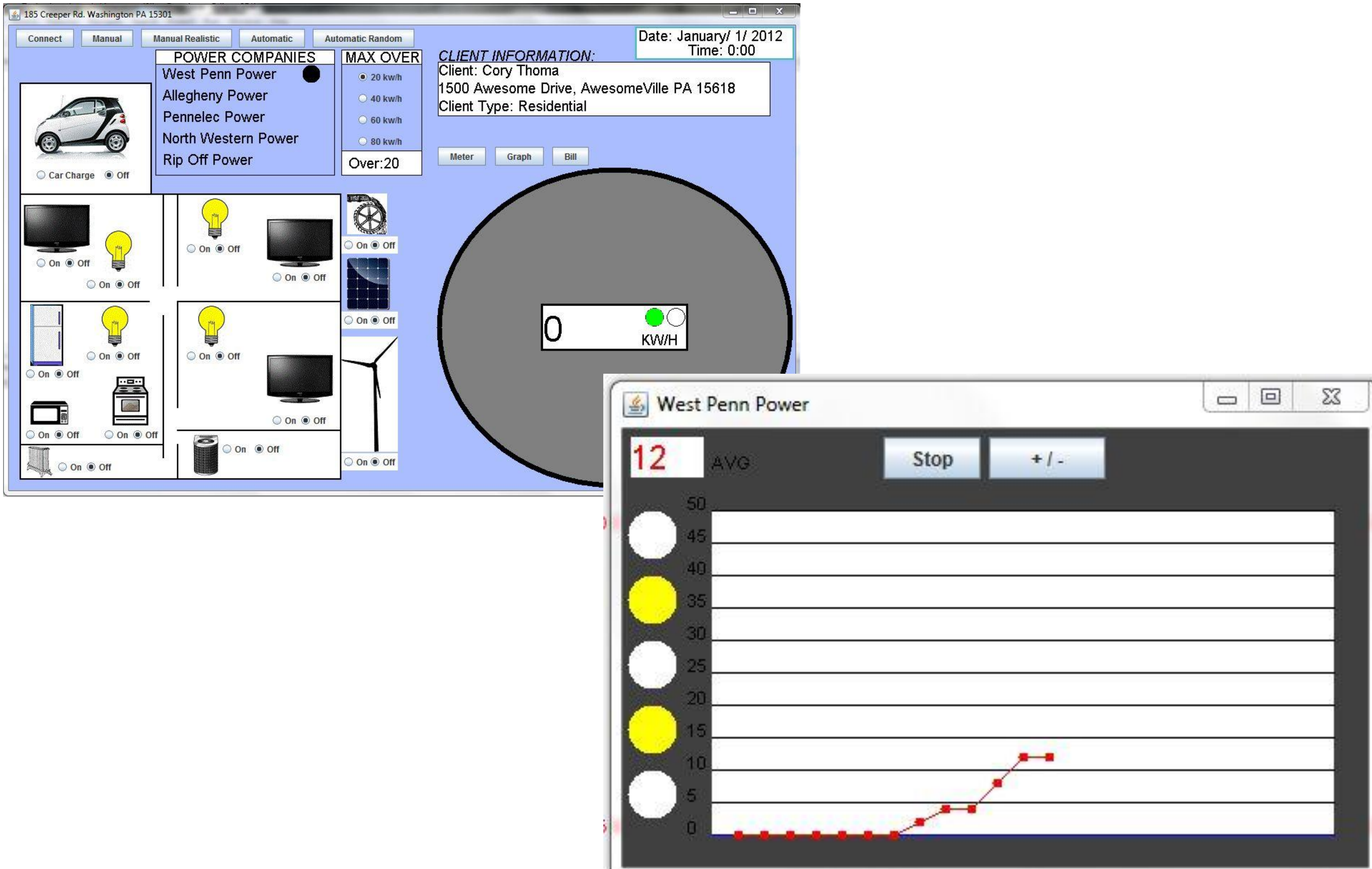
The decryption algorithm produces the intended message, since

$$c_2 \cdot s^{-1} = m' \cdot h^y \cdot (g^{xy})^{-1} = m' \cdot g^{xy} \cdot g^{-xy} = m'.$$

The ElGamal cryptosystem is usually used in a hybrid cryptosystem. I.e., the message itself is encrypted using a symmetric cryptosystem and ElGamal is then used to encrypt the key used for the symmetric cryptosystem. This allows encryption of messages that are longer than the size of the group G.

http://www.wikipedia.org

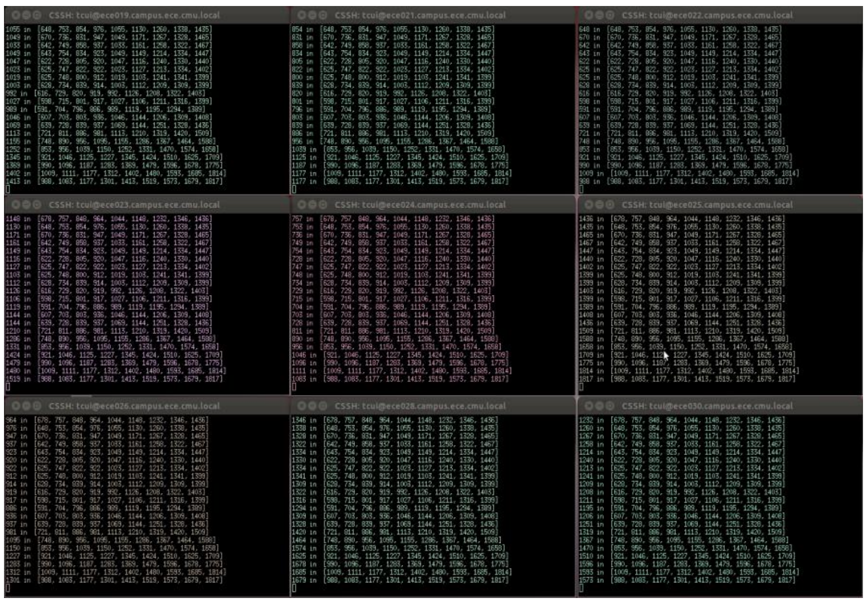
Example of Privacy-Preserving Smart Meters



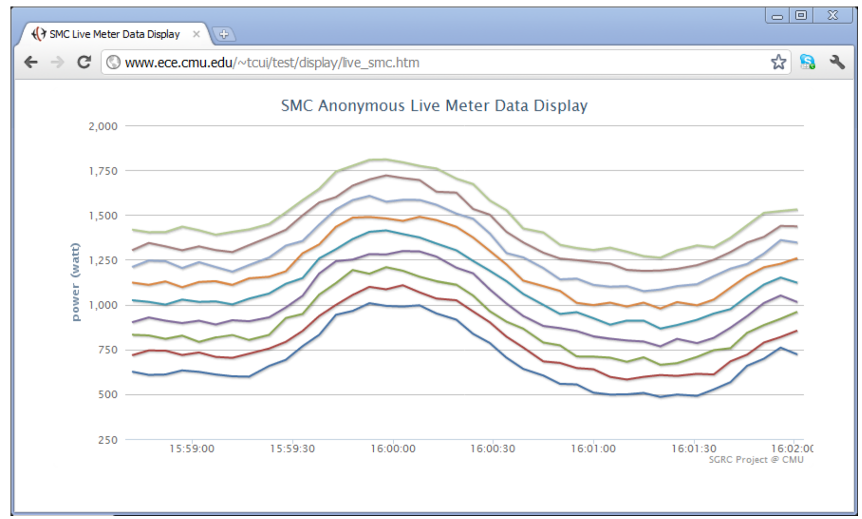
SMC Based Anonymous Smart Meters

- **User keeps privacy, but everybody knows everything**

User's private view (each terminal)



Public view



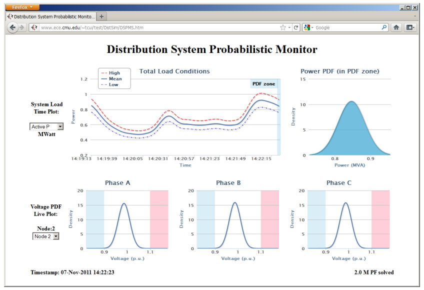
Every user knows their own ID and data and everybody else's anonymized data

- **Demo link: www.ece.cmu.edu/~tcui/test/display/live_smc.htm**

Summary

- **Deskside supercomputers**
  economically pack unprecedented power

- **Example 1: Real-time Monte Carlo simulation**
  for probabilistic load flow computation



- **Example 2: Secure Multiparty Computation**
  enables privacy-enhanced smart meters