# Alice and Bob in Cipherspace - Brian Hayes

# Computing arbitrary functions of encrypted data - Craig Gentry

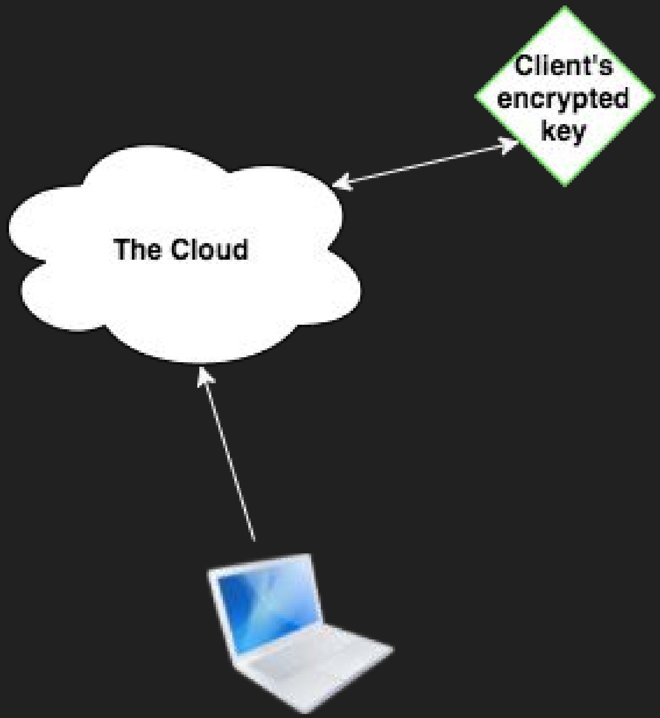
# Secure Information Aggregation for Smart Grids Using Homomorphic Encryption - Fengjun Li, Bo Luo, Peng Liu

Michael Dubell & Emelie Widegren

# Motivation/Problem

# Live demo

# Current state of **Privacy**

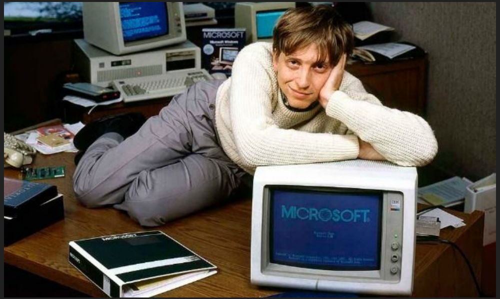



# Can *you* trust the cloud?

What if the cloud changes its code?

# How does homomorphic encryption solve privacy?

- Only user can decrypt

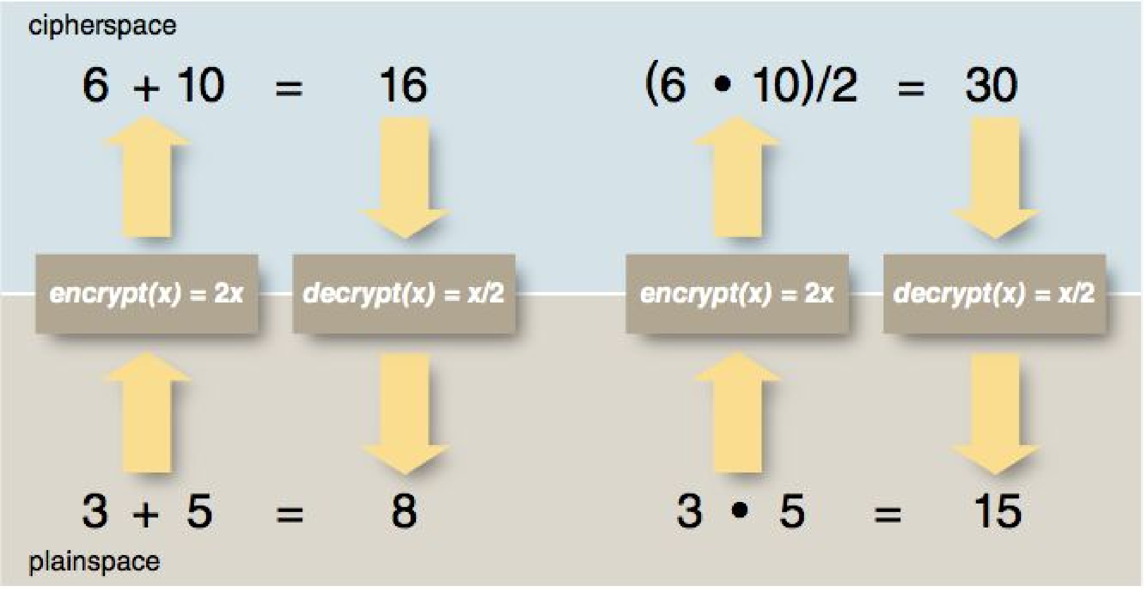



- Cloud can only do mathematical operations on the ciphertext
- Can not decrypt ciphertext


Brent

# Tell us more about this crypto magic

# Partial homomorphic encryption



El Gamal

Benaloh

RSA

Naccache-Stern

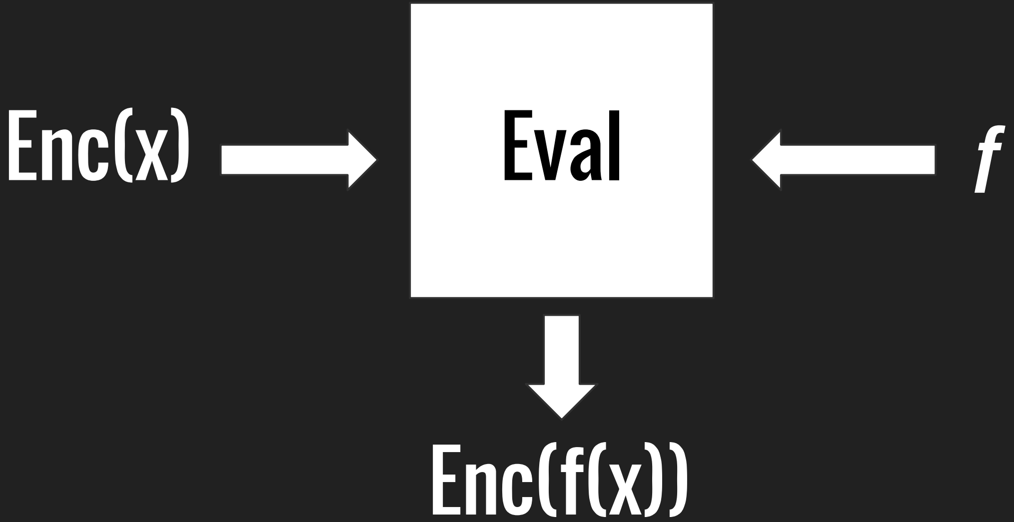
Boneh-Goh-Nissim

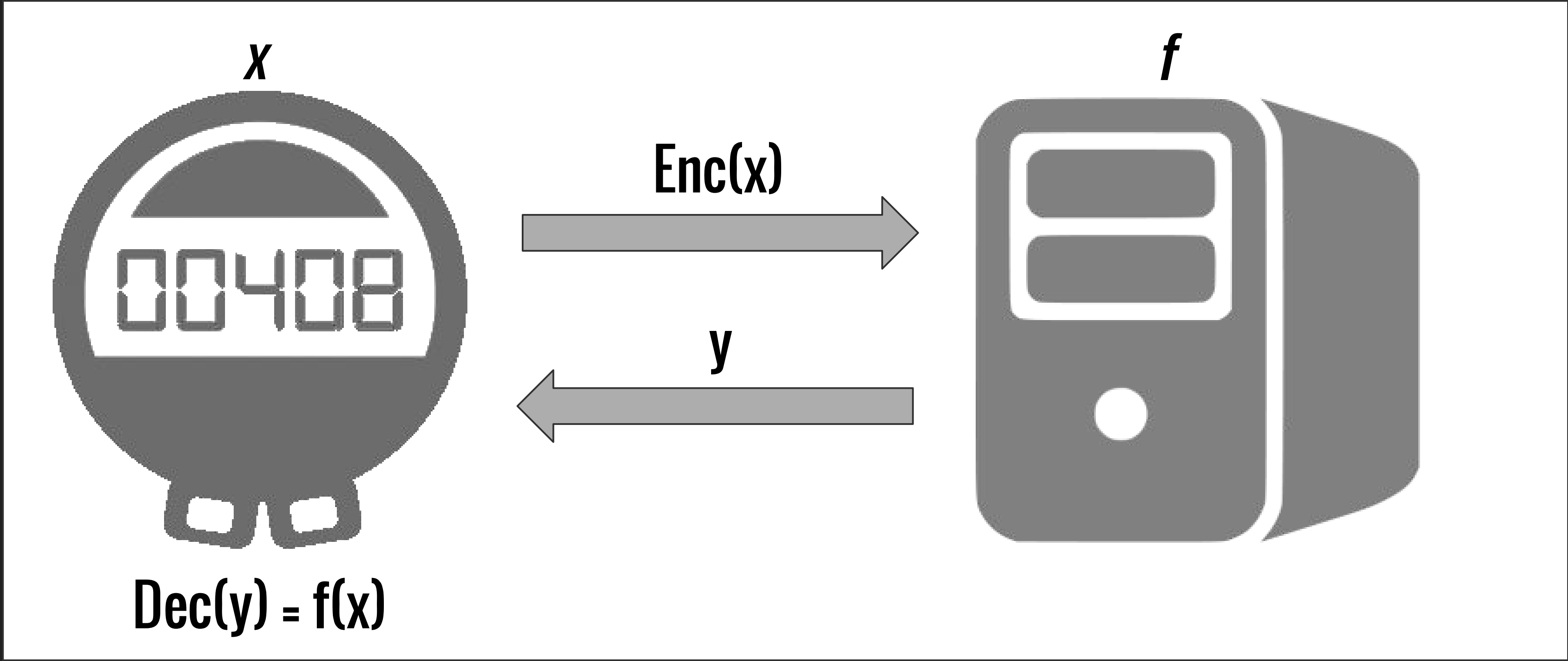
Paillier

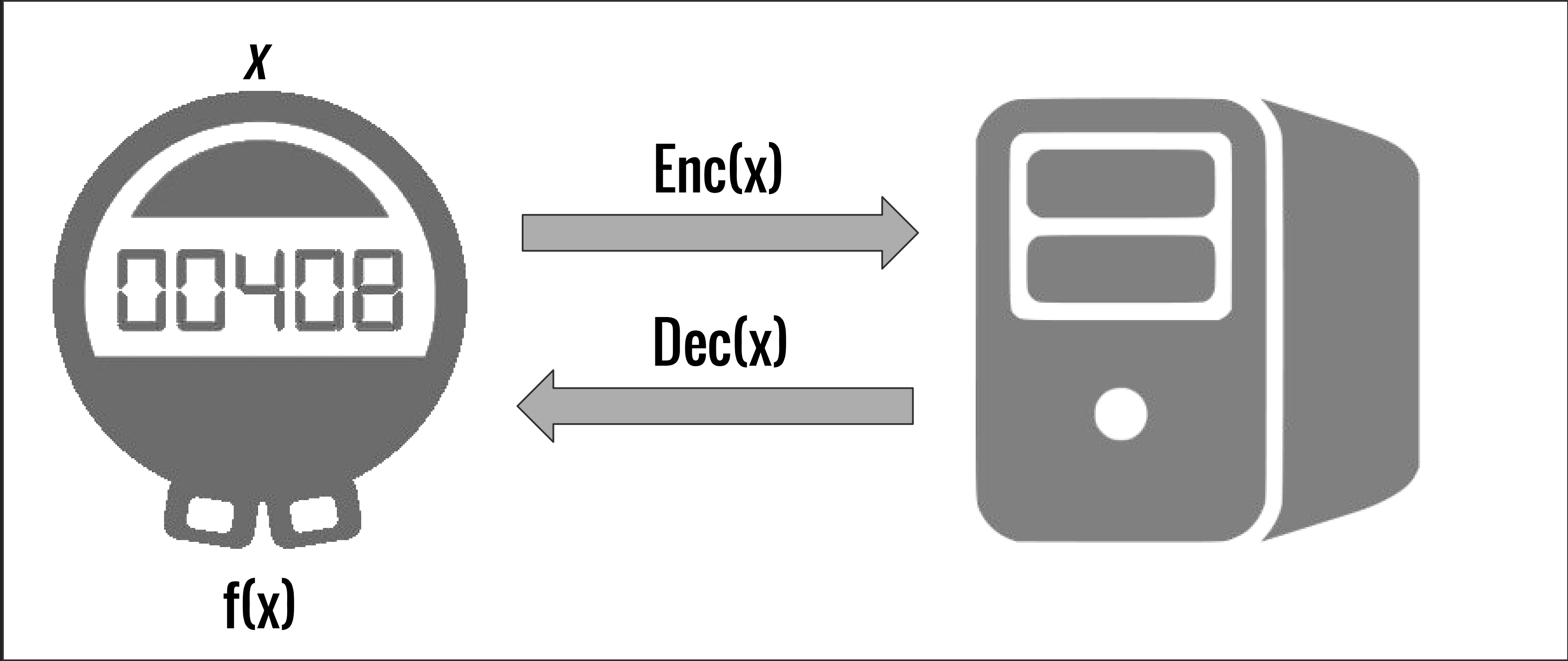
Goldwasser-Micali

# Fully homomorphic encryption

- Supports <u>arbitrary</u> computation on ciphertexts




x
00408
Dec(y) = f(x)
Enc(x)
y
f


x
00408
Enc(x)
Dec(x)
f(x)

# Fully homomorphic encryption

From a  Somewhat Homomorphic Encryption scheme
to a Fully Homomorphic Encryption scheme

Craig Gentry

# An Encryption Scheme:

$\text{KeyGen}_{\varepsilon}(\lambda)$ -> (sk, pk)

$\text{Decrypt}_{\varepsilon}(sk,c)$ -> m'

$\text{Encrypt}_{\varepsilon}(pk,m)$ -> c

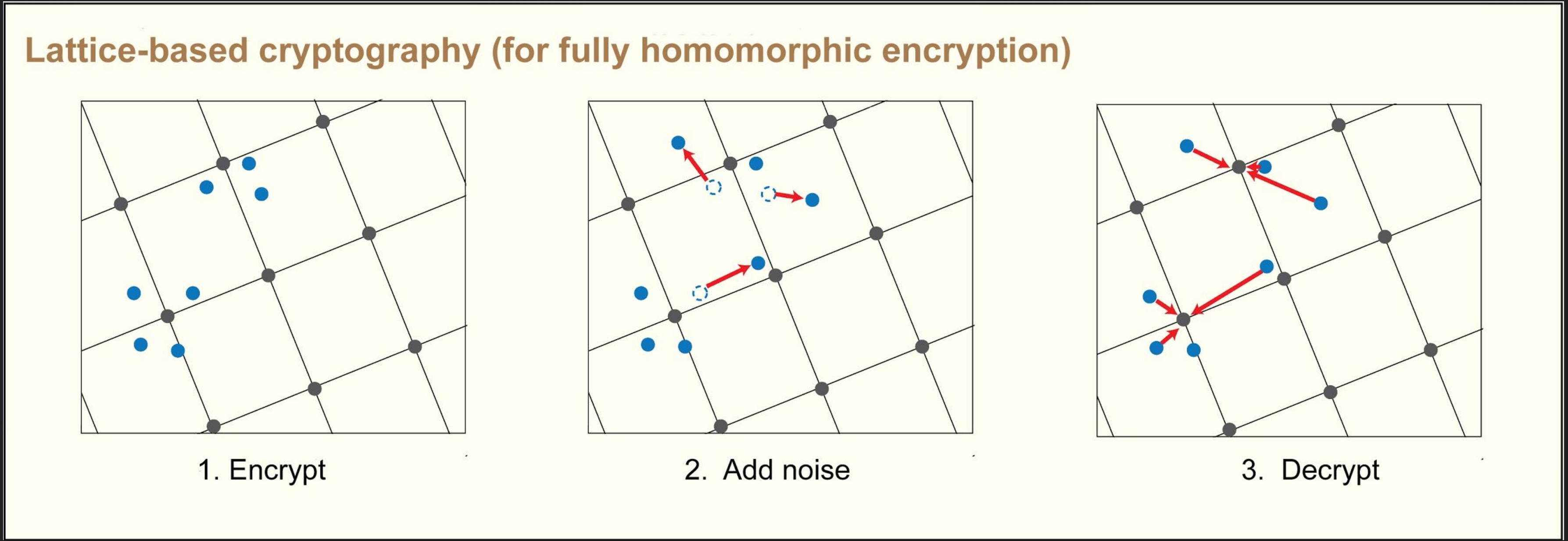
$\text{Evaluate}_{\varepsilon}(f,c_1,\ldots,c_i)$ -> c

The secret key, sk, is a random P-bit odd integer p.

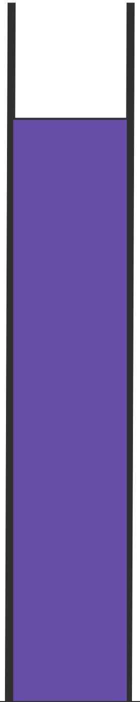
# A somewhat homomorphic encryption scheme

- A limited number of functions supported.

# Lattice-based cryptography



# Bootstrapping

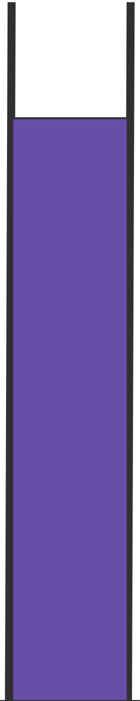


noise = p/2

Addition doubles noise
Multiplication squares

noise = 0

# Bootstrapping



noise = p/2

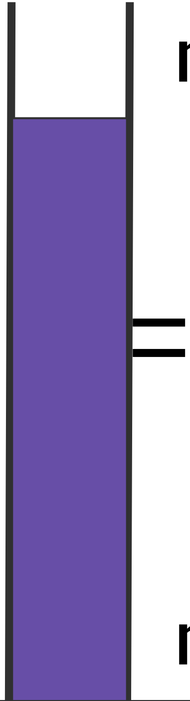
We want noise-reduction

DECRYPTION

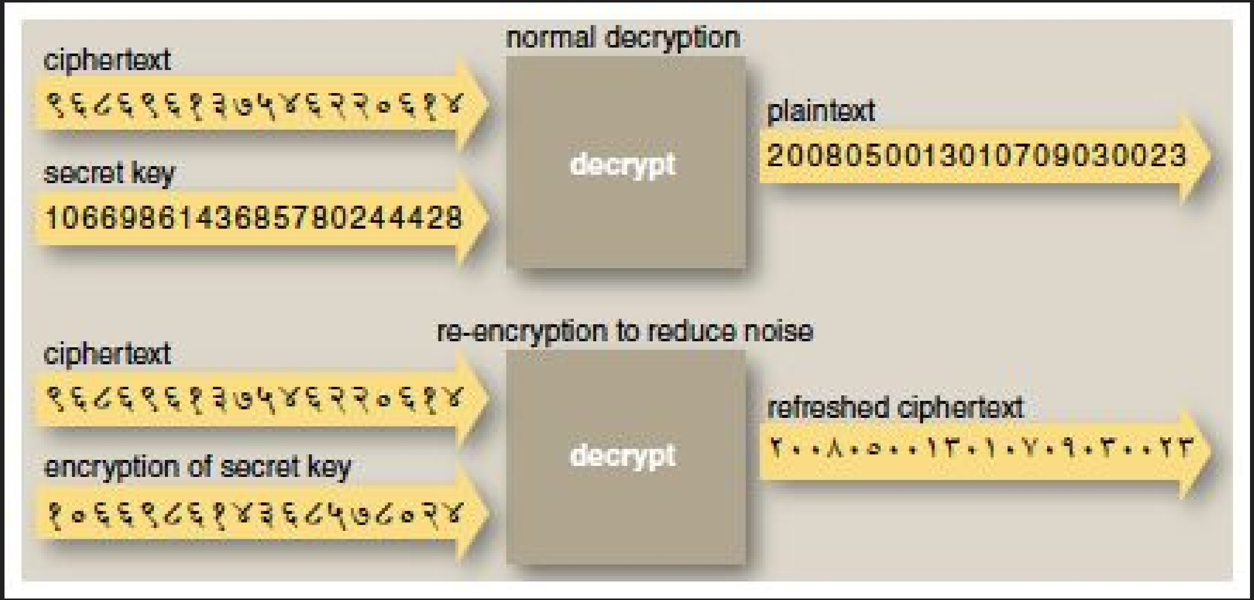
- But we can’t give out the secret key

noise = 0

# Bootstrapping



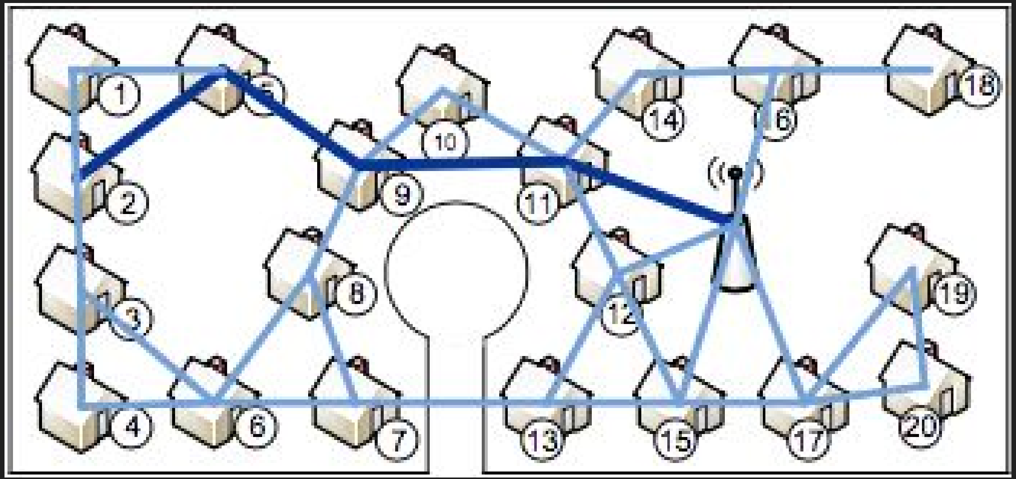

We can't release the secret key but we can release Enc(secret key)


normal decryption
ciphertext
secret key
1066986143685780244428
decrypt
plaintext
2008050013010709030023
re-encryption to reduce noise
ciphertext
encryption of secret key
decrypt
refreshed ciphertext
٢٠٠٨٠٥٠٠١٣٠١٠٧٠٩٠٣٠٠٢٣

# Is Gentry's scheme practical?

# Implementing homomorphic encryption in the smart grid

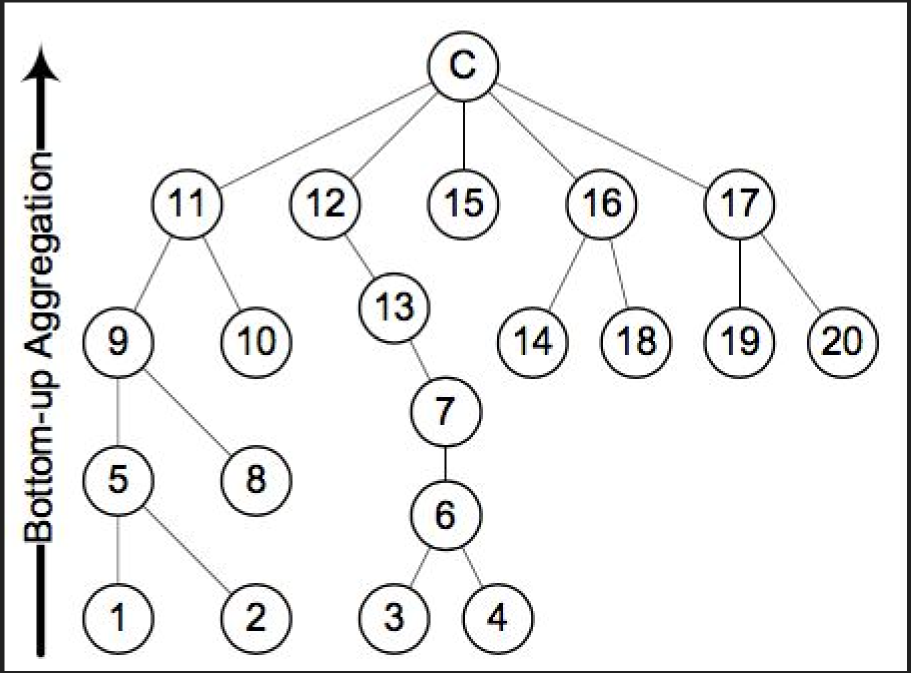
# An example of smart grid communication in a neighborhood.




[-] Excessive network traffic

[-] High overhead for collector

# Improved communication graph




Collector calculates a BFS Tree

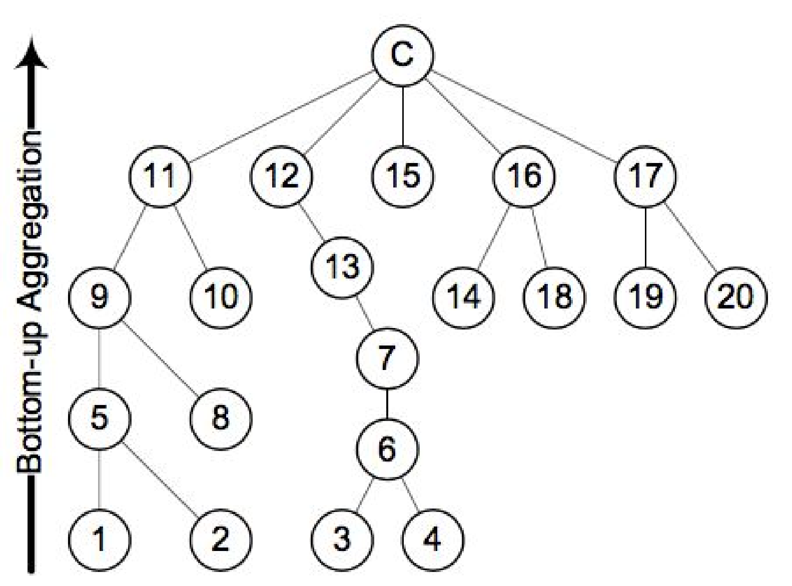
[+] Collector offloads computation

[+] Computation is done in a distributed manner

[+] Tree can easily be rebalanced

# How is homomorphic encryption used in this scheme?

# System design




**7-tuple message**
{TID, Trigger, Data, Collect, Operation, Destination, Key}

**Child node**
Generate its readings, encrypts data and sends to parent

**Parent node**
Collects data from all children, appends its own reading
Sends data to its parent or collector

**Collector node**

- Collects data from children
- Performs some operation
- Decrypts data

# Paillier Cryptosystem - Key Generation

**Key Generation**

1. Pick two large prime numbers p and q;
2. $N = p \cdot q$ and $\lambda = \mathrm{lcm}(p-1, q-1)$, where lcm represents least common multiple.
3. Select a random number g where $g \in Z^*_{n^2}$
4. Set function $L(u)$ as: $L(u) = (u - 1)/N$
5. Ensure that *N* divides the order of g: check if $L(g^\lambda \bmod N^2)$ and *n* are co-prime, i.e. $\gcd(L(g^\lambda \bmod N^2), N) = 1$
6. (N, g) is the public key pair.
7. (p, q) is the private key pair.

# Paillier Cryptosystem - Encryption

1. We want to encrypt the message: $m \in Z^*_N$
2. Select a random number: $r \in Z^*_N$
3. Encrypt $m$ using: $c = E(m) = g^m \cdot r^N \bmod N^2$

# Paillier Cryptosystem - Decryption

1. We want to decrypt ciphertext: $c \in Z^*_{N^2}$

2) Decrypt with: $m = \mathsf{D}(c) = \left(\frac{L(c^\lambda \mod N^2)}{L(g^\lambda \mod N^2)}\right) \mod N$

Given $c_1 = E(m_1)$ and $c_2 = E(m_2)$, $\forall m_1, m_2 \in Z_N$, we have $D(c_1 \cdot c_2 \mod N^2) = m_1 + m_2 \mod N$

Medical Applications

Financial Applications

Advertising and Pricing

Voting

# Summary

- Properties of homomorphic encryption
- Partial homomorphism
- Fully homomorphism
- An example how it can be integrated into the smart grid