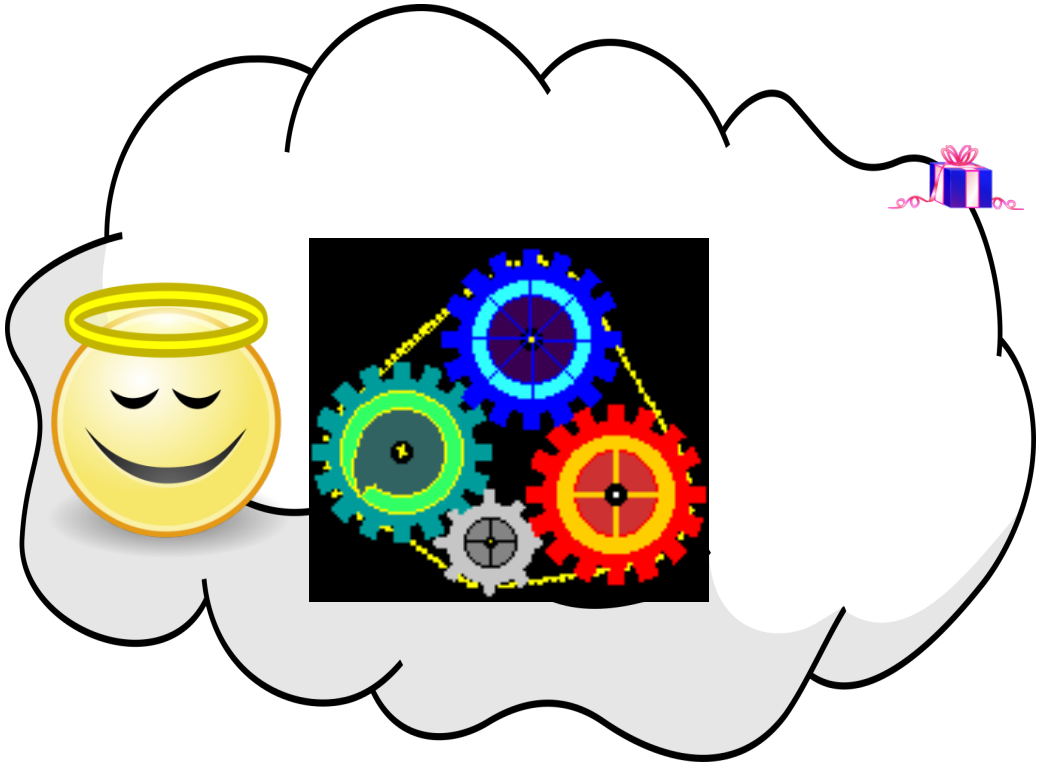
Computing with Encrypted Data and Programs

Shai Halevi (IBM Research)

CCC Symposium --- May 10, 2016

The Wonderful Cloud



The Wonderful Cloud

not so

Cryptography to the Rescue?

- Wouldn't it be nice to be able to…
  - Encrypt my data before sending to cloud
  - While still allowing the cloud to search/ sort/edit/… this data on my behalf
  - Keeping the data in encrypted form
    - Without shipping it back and forth to be decrypted

Cryptography to the Rescue?

- Wouldn't it be nice to be able to…
  - Encrypt my queries to the cloud
    - While still letting the cloud process them
  - Cloud returns encrypted answers
    - that I can decrypt

Homomorphic Encryption

Delegation: Encrypting x and decrypting f(x) is cheaper than computing f(x) myself.

The special sauce!
Running Eval should be efficient

"I want 1) the cloud to process my data 2) even though it is encrypted.

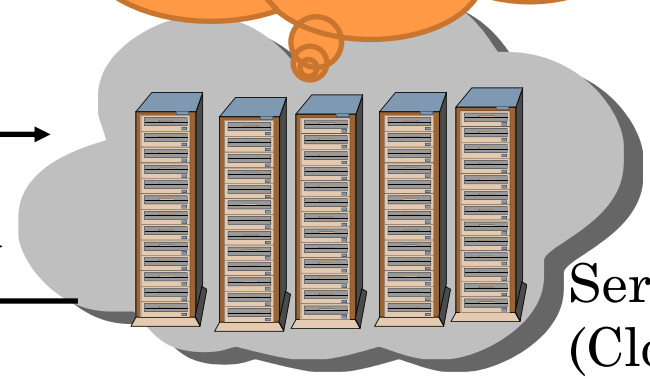
Run
Eval[f, $Enc_k(x)$]
= $Enc_k[f(x)]$

Alice
(Input: data x, key k)

$Enc_k(x)$

function f

This could be encrypted too.



Server
(Cloud)

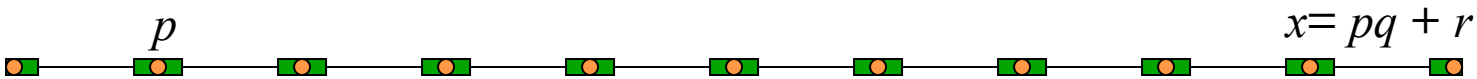
$Enc_k[f(x)]$

f(x)

Brief History

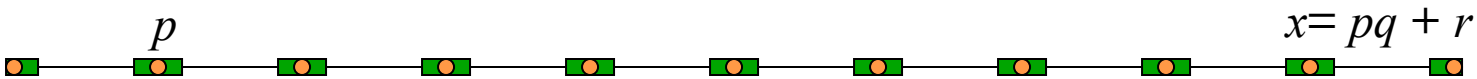
- Possibility noted in the early days of public-key encryption [RAD'78]
- Many "somewhat homomorphic" schemes over the years
  - Can only compute (very) limited functions
  - E.g., only linear functions
- First "fully homomorphic" PKE in [Gen'09]
  - FHE can compute any function (in principle)
- Rapid advances since then
  - Better security, much better efficiency

How Can This Be?



- A simple (symmetric) example [vDGHV'10]:
  - Bit-by-bit encryption (plaintext space is {0,1})
  - Secret key is an odd integer p
  - Ciphertexts are integers close to multiples of p
  - $\boldsymbol{ct \leftarrow p \cdot q + r}$ (with $|\boldsymbol{r}| \ll \boldsymbol{p}$)
    - The encrypted bit is the LSB of the "noise" r (zero when r is even, one when it is odd)
  - Add/mult the integer ciphertexts correspond to add/mult of the plaintext bits (mod 2)
    - As long as the noise remains $\ll p$

How Can This Be?



- A simple (symmetric) example [vDGHV'10]:
  - Bit-by-bit encryption (plaintext space is {0,1})
  - Secret key is a prime
  - Ciphertext size is polylog of n
  - Add/mult
    - Th
      (zero whe
  - Add/mult the ciphertexts correspond to add/mult of the plaintext bits (mod 2)
    - As long as the noise remains $\ll p$

Any function can be implemented from addition & multiplication operations

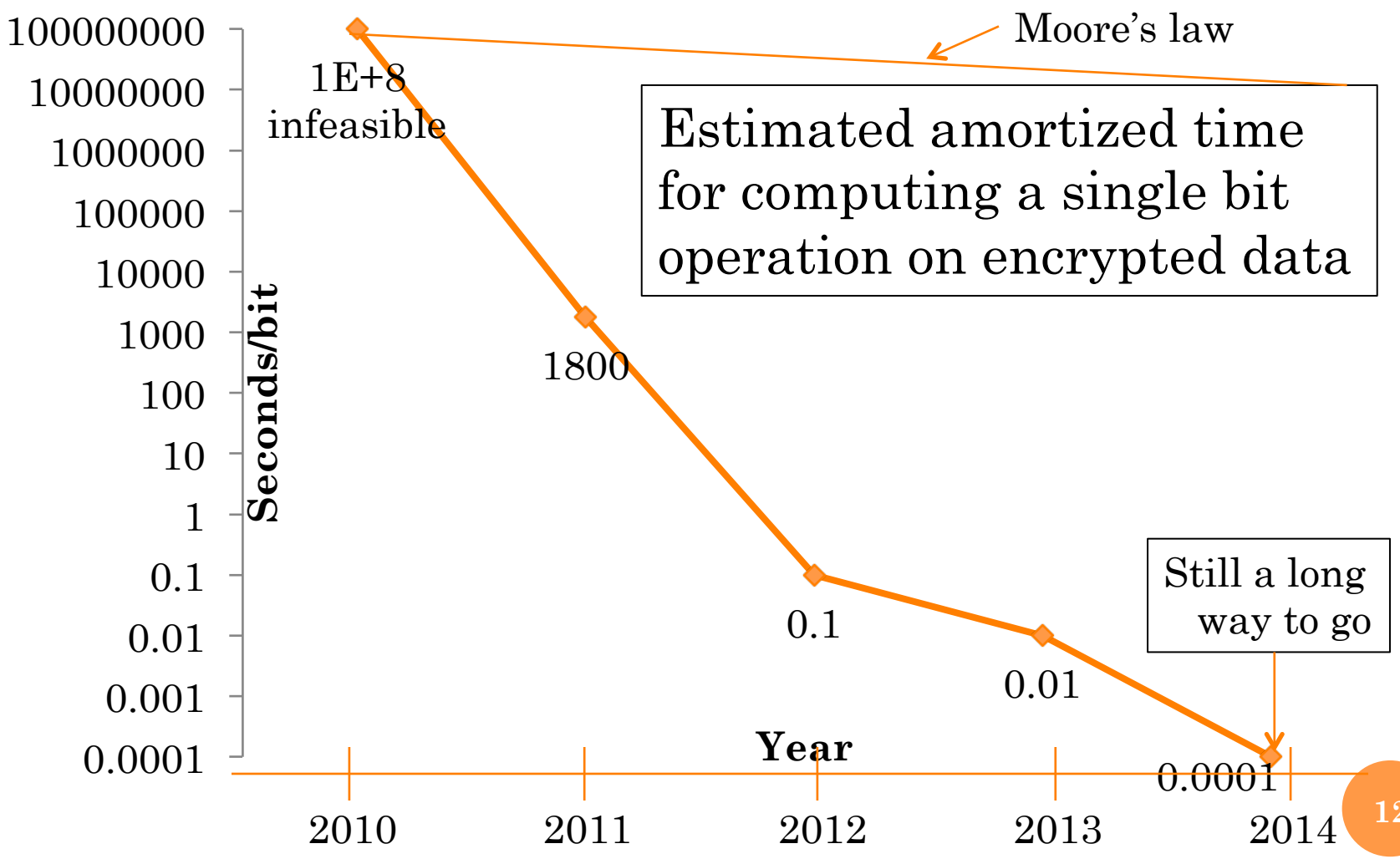
Three Generations of FHE

- 1G. First plausible candidate in [Gen'09]
  - Ciphertext is "noisy"
  - Noise grows with computation
    - Once too noisy, the "signal" is lost
  - Noise exponential in the degree of the function
  - ➔Parameters must be huge, to allow large noise
- 2G. [BV'11, BGV'12,…]: Better noise control
  - Noise grows linearly with degree
  - "Ciphertext packing": many plaintext elements packed in a single ciphertext

Three Generations of FHE

- 1G. Fast accumulation of noise
- 2G. Better noise management + packing
- 3G. [GSW13,…]: "Asymmetric" noise growth
  - Very slow noise growth for some circuits
- But slow noise growth in 3G is incompatible with ciphertext-packing (as far as we know)
- For efficiency, we have a choice:
  - 2G+packing (faster asymptotically)
  - or 3G+small-noise (sometimes faster in practice)

Speed of FHE


Moore's law
Estimated amortized time for computing a single bit operation on encrypted data
Still a long way to go
1E+8
infeasible
1800
0.1
0.01
0.0001
Seconds/bit
100000000
10000000
1000000
100000
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
Year
2010
2011
2012
2013
2014

Beyond Homomorphic Encryption

- Attribute-based Encryption (ABE)
- Functional Encryption (FE)
- Code Obfuscation

Limitations of FHE

- FHE is very powerful
- But access to data is all-or-nothing
  - Without the secret key, all you see is a "meaningless ciphertext"
  - If you have the secret key, you can read the result but also intermediate values
- Computation is unrestricted
  - Can't limit the functions that can be computed on encrypted data

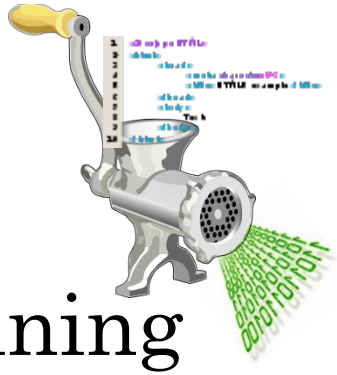
Attribute-Based Encryption (ABE) [S84, SW05…]

- One PK, many "partial" secret keys
  - Each key associated with some attributes
  - Encrypt m under PK and policy P
  - Only key with attributes satisfying P can decrypt m
- Useful for controlling access to m
  - Access-control "baked" into ciphertext
- But no computation on encrypted data
  - Decryption recovers m unmodified

What We Want…

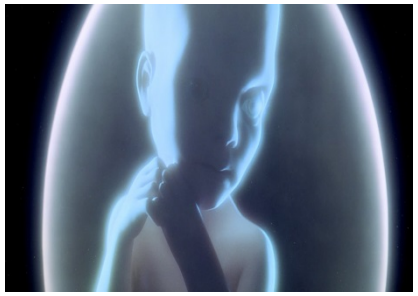
- FHE and ABE's Love Child
- Functional Encryption (FE): Controlled encrypted computation
  - Each key is restricted to one specific f
  - Can compute $f(m)$ from $\mathrm{ENC}(m)$ using $SK{\downarrow}f$
- Unlike FHE: gets $f(m)$ in the clear
  - But only for this one function f, on this m
- Another "similar" construct: code obfuscation, secrets in software

Code Obfuscation



- "Encrypting" programs, maintaining their functionality
  - Program P → "Encrypted program" P'
  - Given P' and any x, compute $P'(x)=P(x)$
  - But otherwise P' hides whatever secrets that P depends on
- Example: patching software
  - Patch includes description of vulnerability
  - "Encrypted patch" works the same, but hides the vulnerability

What We Think We Have…



- FHE and ABE's Love Child, but not fully developed
- "Proof of concept" obfuscation, FE
  - Using "multilinear maps"
  - Security is unclear
  - Performance even worse than FHE in 2010
- Blooming theory on use of FE, obfuscation
  - Marvelous constructions, links to other concepts in crypto, computer-science

The Road Ahead

- FHE, ABE, FE, Obfuscation
  - Very powerful tools
- Open the door to new application
  - Used to be "science fiction"
  - E.g., software agents that can hide secrets even from the hosts that run them
- FHE, ABE on the road to usability
  - Can already be used in niche application
- FE, obfuscation still in their infancy

The Road Ahead

- A related topic: verifiable computation
  - Integrity for cloud computing
  - Alice delegate work to the cloud, want a proof that the results are correct
- Great progress here too
  - Also on the road to usability

Questions?