Fully Homomorphic Encryption from LWE

Zvika Brakerski

(Stanford)

Based on joint works with:

Vinod Vaikuntanathan

(University of Toronto)

Craig Gentry

(IBM)

Post-Quantum Webinar,
November 2011

Outsourcing Computation

Outsourcing Computation

x

$f(x)$

Function

f

Outsourcing Computation

It's everywhere!

x

$f(x)$

Function

f

Outsourcing Computation

It's everywhere!

x

search query

$f(x)$

Search results

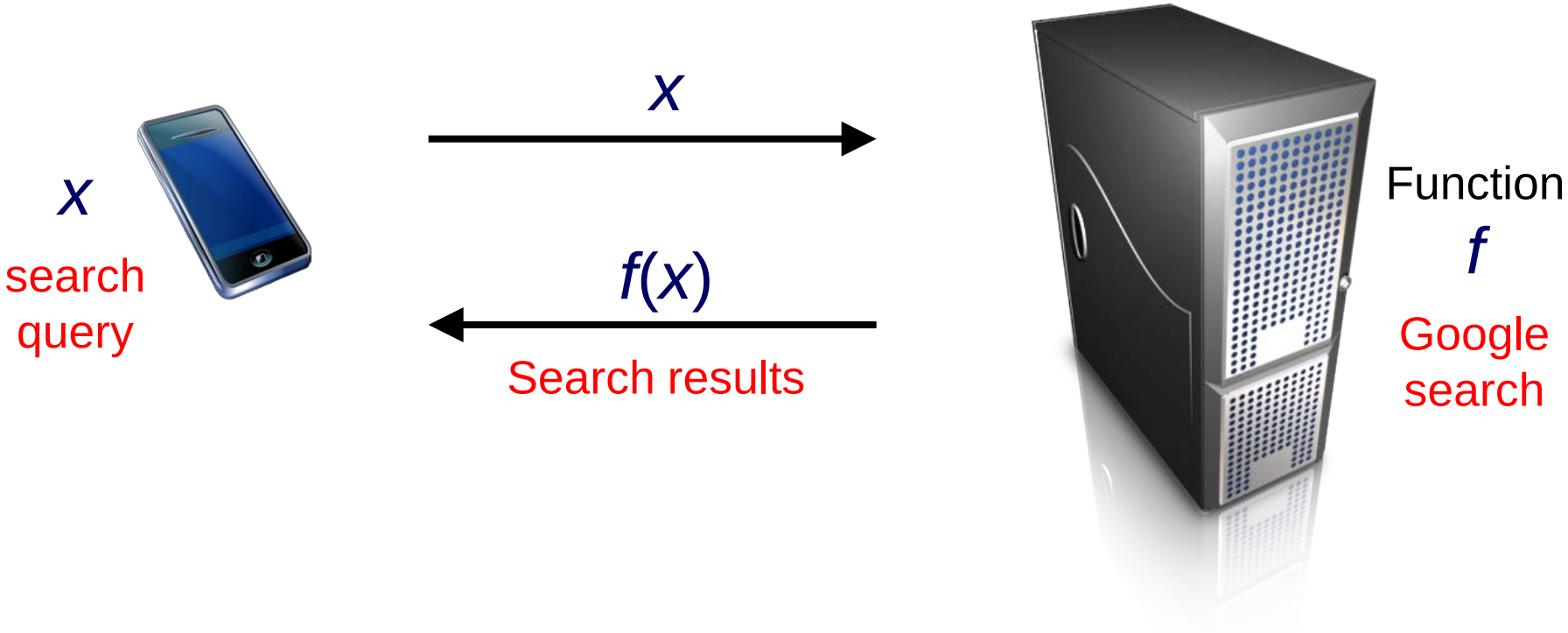
Function

f

Google search

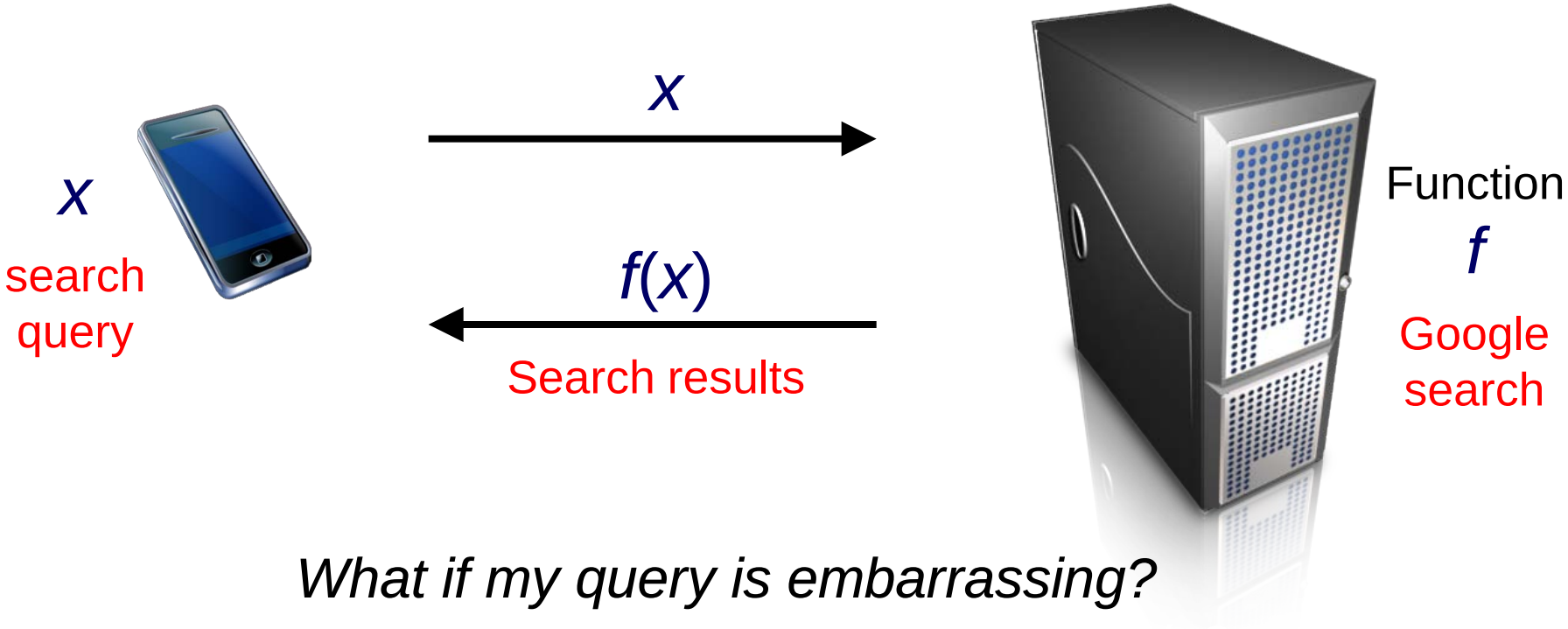
Outsourcing Computation

It's everywhere!



Outsourcing Computation

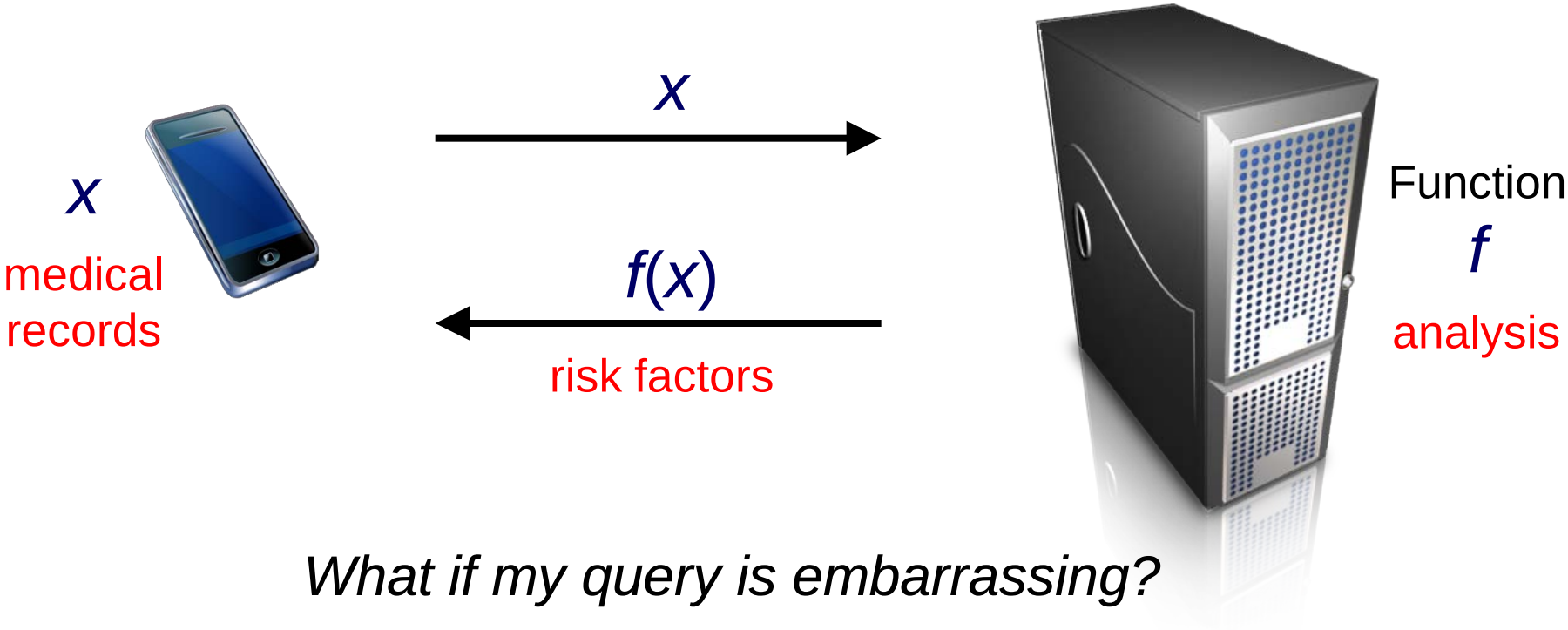
It's everywhere!



What if my query is embarrassing?

Outsourcing Computation

It's everywhere!

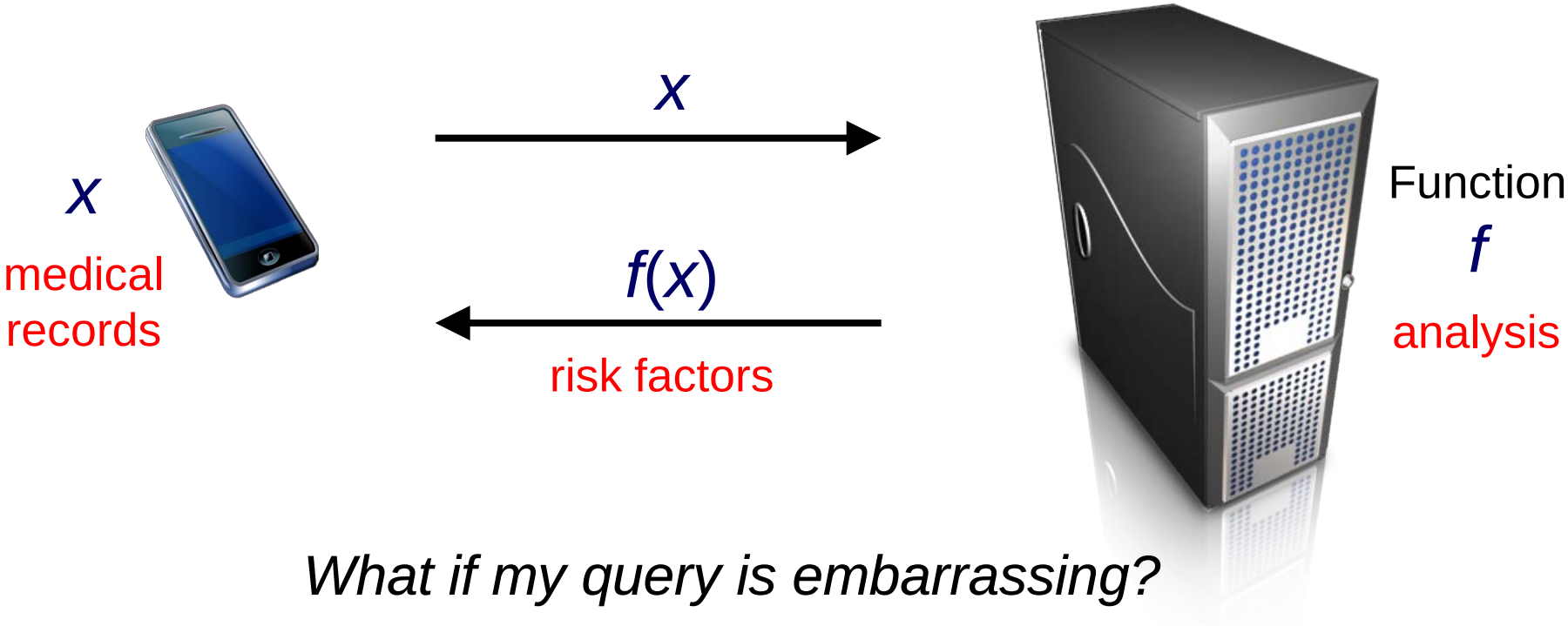


What if my query is embarrassing?

What if I'm sending my medical records?

Outsourcing Computation

It's everywhere!



What if my query is embarrassing?

What if I'm sending my medical records?

Want Privacy!

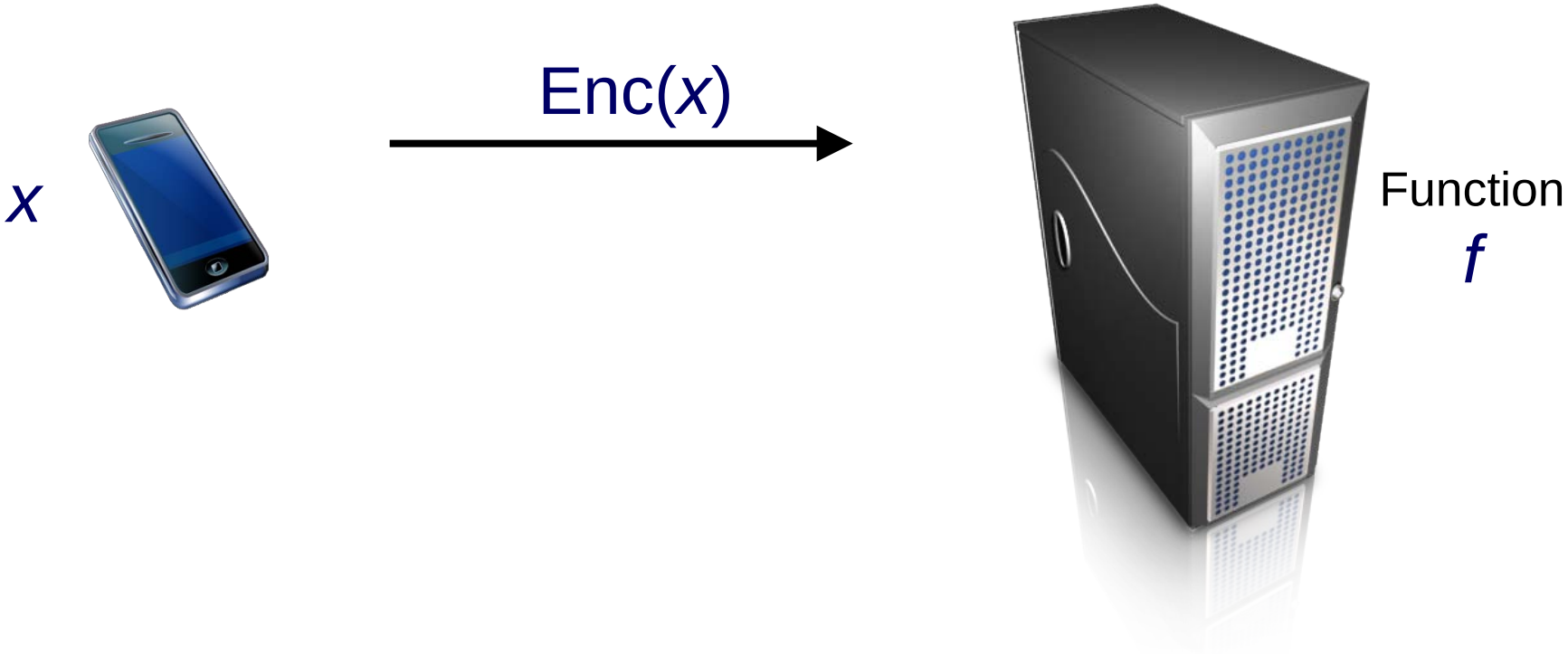
Outsourcing Computation – Privately

x

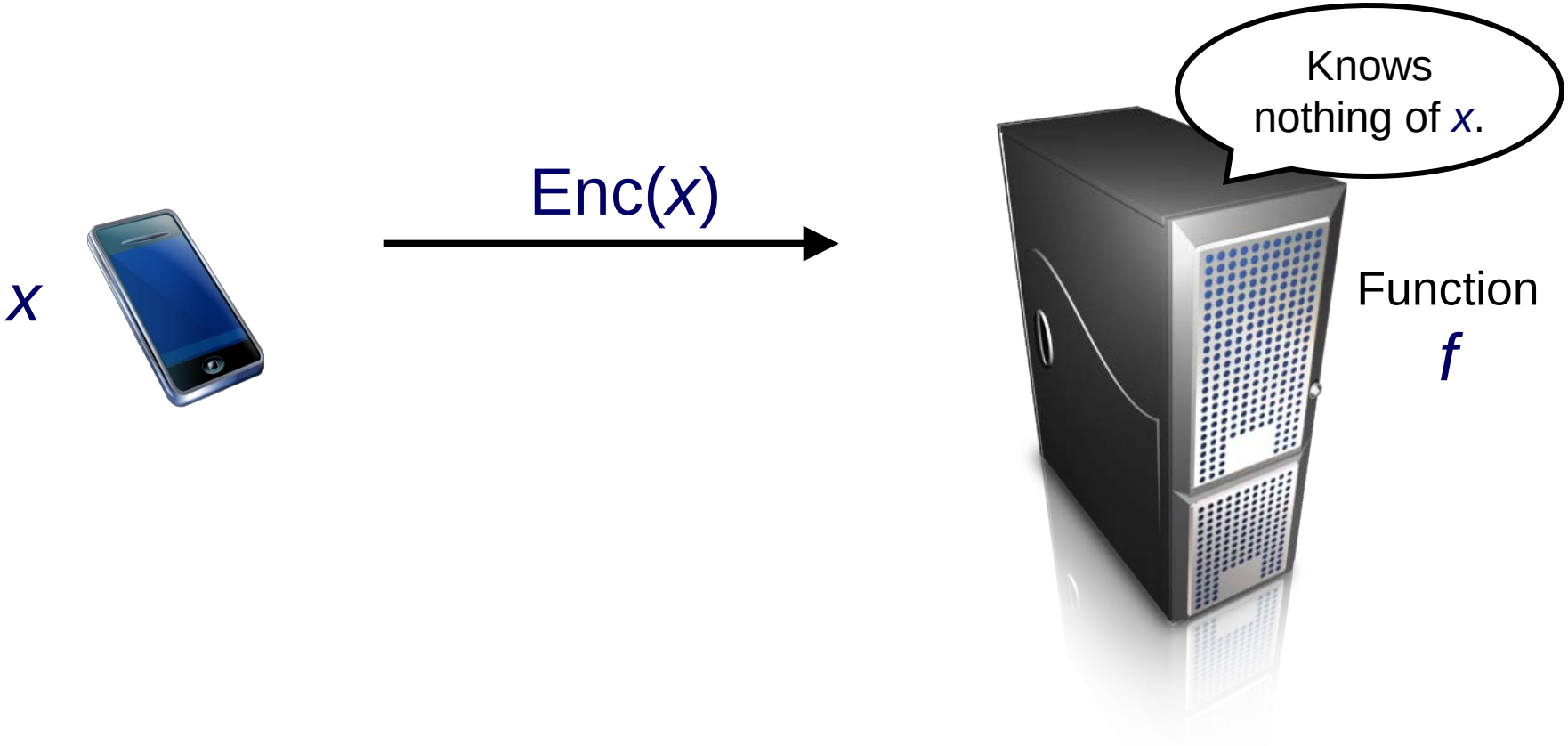
Function

f

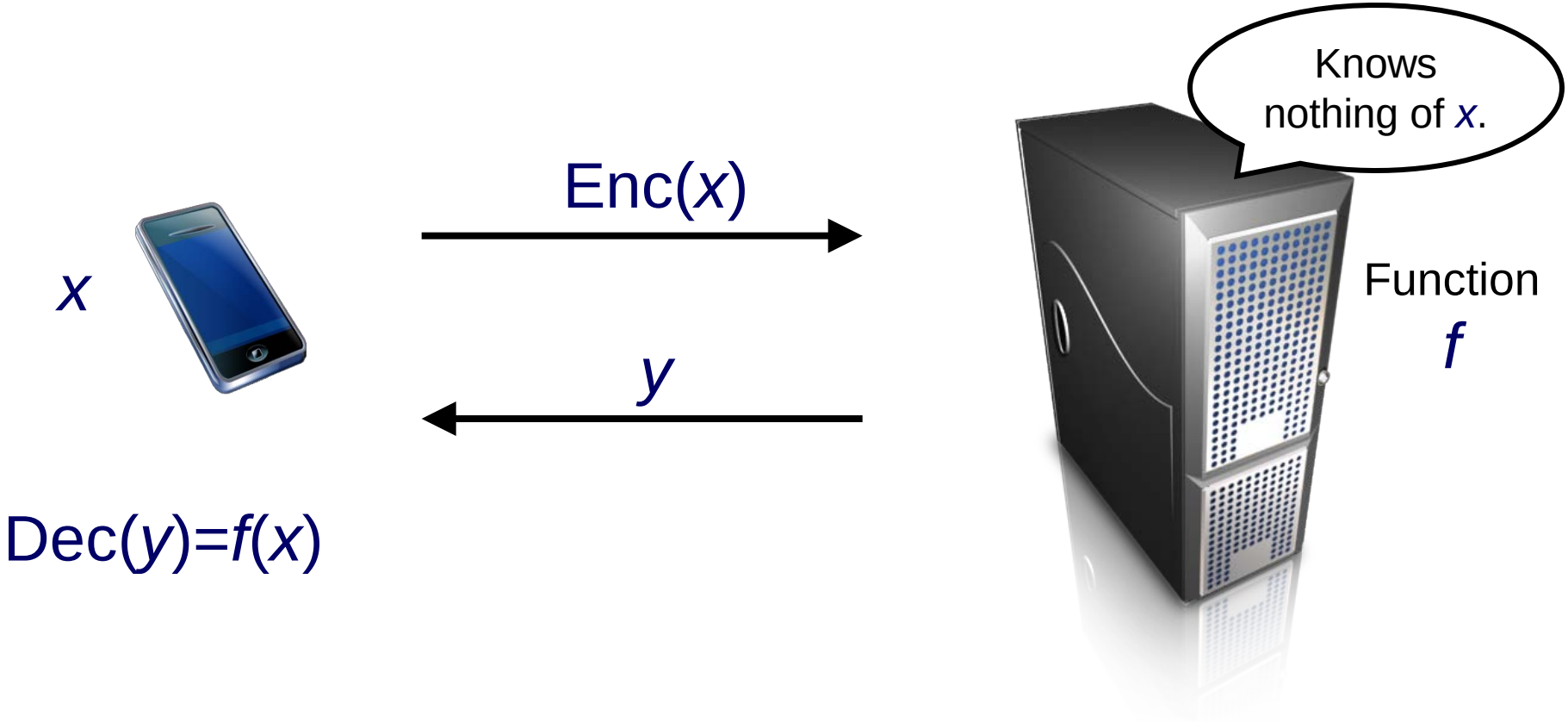
Outsourcing Computation – Privately



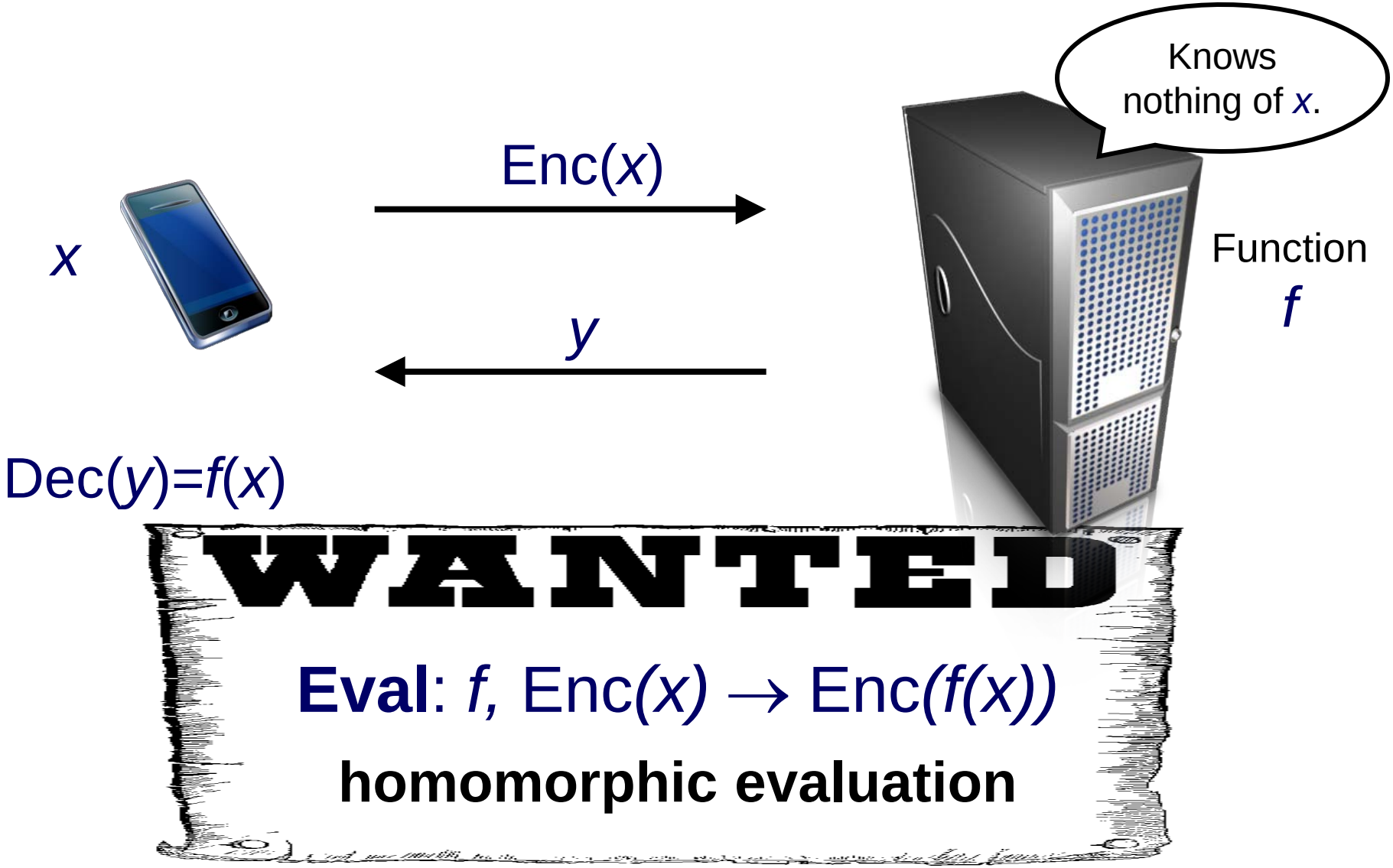
Outsourcing Computation – Privately



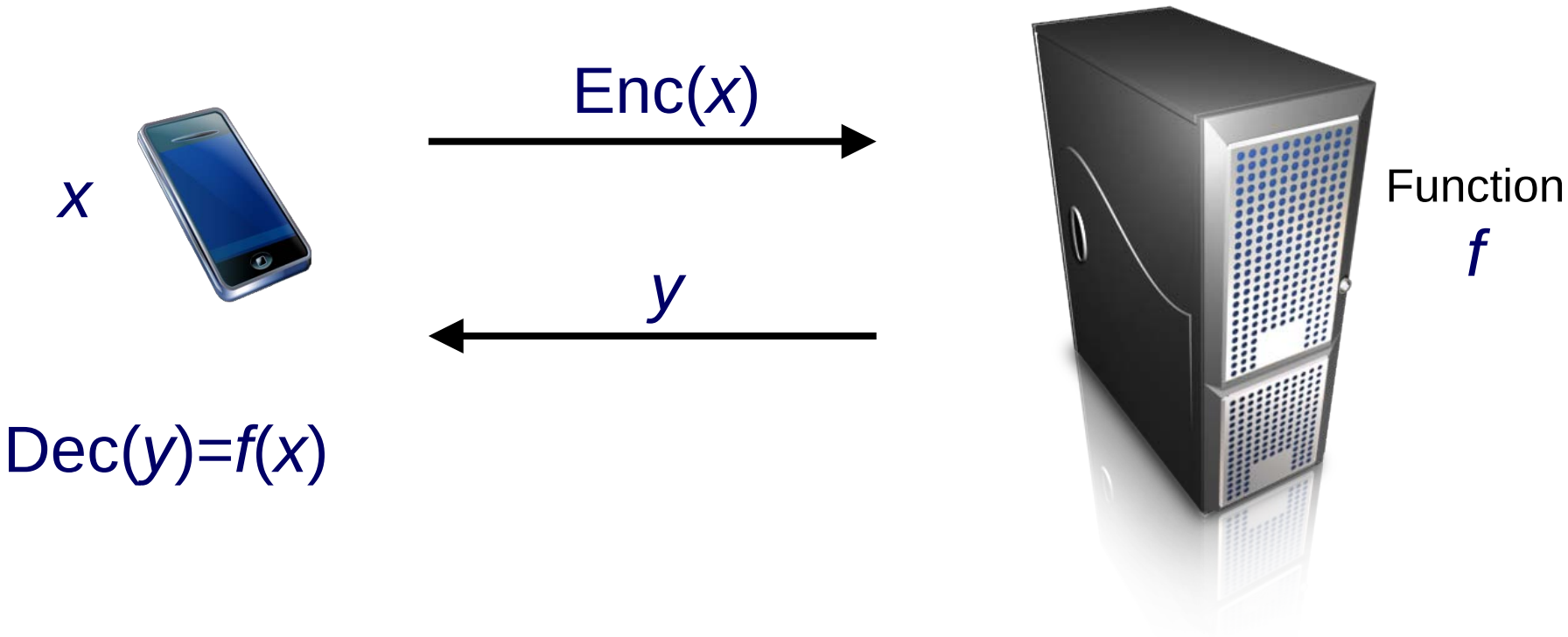
Outsourcing Computation – Privately



Outsourcing Computation – Privately



Fully Homomorphic Encryption (FHE) [RAD78]



Fully Homomorphic Encryption (FHE) [RAD78]

sk , *pk*

x

Enc(*x*)

y

Dec(*y*)=*f*(*x*)

evk

evaluation key

Function

f

Fully Homomorphic Encryption (FHE) [RAD78]

sk

x

Enc(*x*)

y

Dec(*y*)=*f*(*x*)

evk

evaluation key

Function

f

Fully Homomorphic Encryption (FHE) [RAD78]

sk

x

$\mathrm{Enc}(x)$ →

← $y = \mathrm{Eval}_{evk}(f, \mathrm{Enc}(x))$

evk

evaluation key

Function

f

Correctness guarantee:

$\mathrm{Dec}_{sk}(y)=f(x)$

Fully Homomorphic Encryption (FHE) [RAD78]

sk

x

$\mathrm{Enc}(x)$ →

← $y = \mathrm{Eval}_{evk}(f, \mathrm{Enc}(x))$

evk

evaluation key

Function f

Correctness guarantee:

$\mathrm{Dec}_{sk}(y)=f(x)$

Privacy guarantee (semantic security [GM82]):

$\mathrm{Enc}(x) \cong \mathrm{Enc}(0)$

Fully Homomorphic Encryption (FHE) [RAD78]

sk

x

$\mathrm{Enc}(x)$ →

← $y = \mathrm{Eval}_{evk}(f, \mathrm{Enc}(x))$

evk

evaluation key

Function f

Correctness guarantee:

$\mathrm{Dec}_{sk}(y)=f(x)$

Privacy guarantee (semantic security [GM82]):

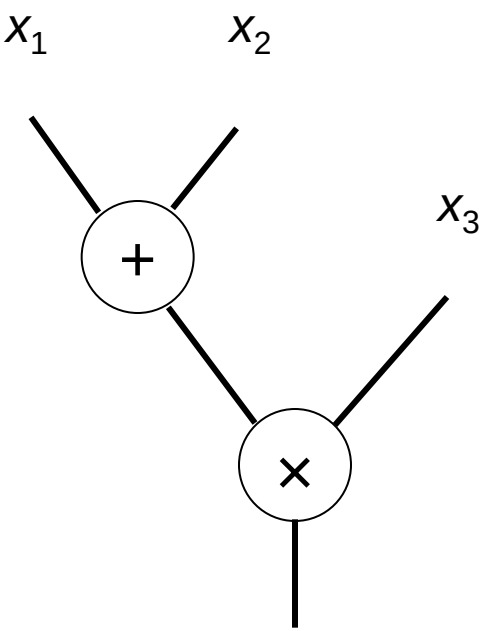
$\mathrm{Enc}(x) \cong \mathrm{Enc}(0)$

"Fully" = Evaluate all (efficient) f

Add & Mult Are Universal

(a la [BGW88])

Arith. Circuit (+, ×) over GF(2).

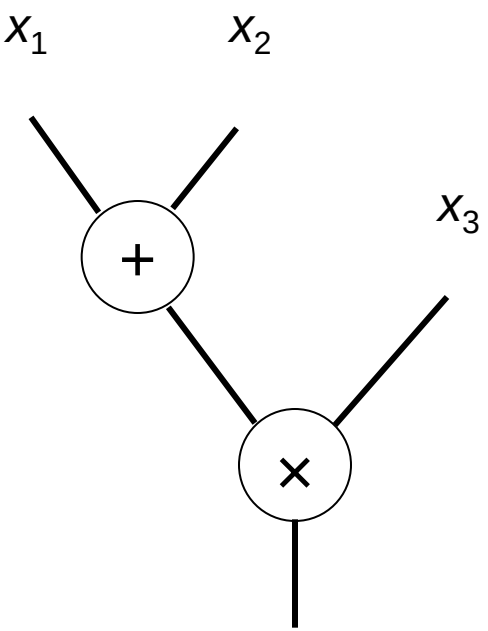


Add & Mult Are Universal

(a la [BGW88])

Arith. Circuit (+, ×) over GF(2).

$f(x_1,x_2,x_3)=(x_1+x_2)\cdot x_3$



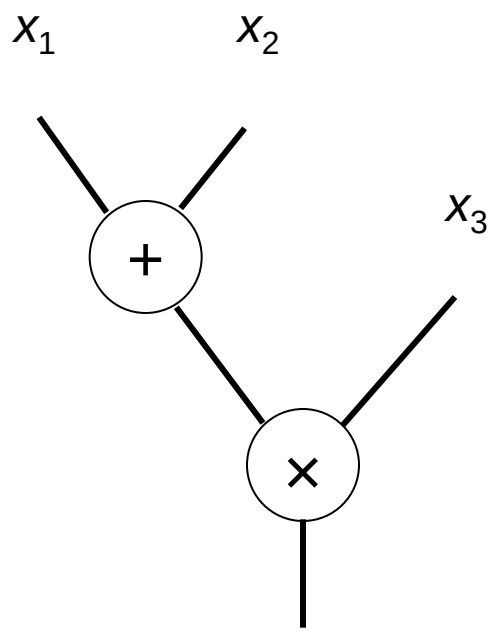
Add & Mult Are Universal

(a la [BGW88])

(+,×) over GF(2) ≡ Boolean (XOR,AND)
= Universal set

Arith. Circuit (+, ×) over GF(2).

$f(x_1,x_2,x_3)=(x_1+x_2)\cdot x_3$



Add & Mult Are Universal

(a la [BGW88])

$(+,\times)$ over GF(2) $\equiv$ Boolean (XOR,AND)
= Universal set

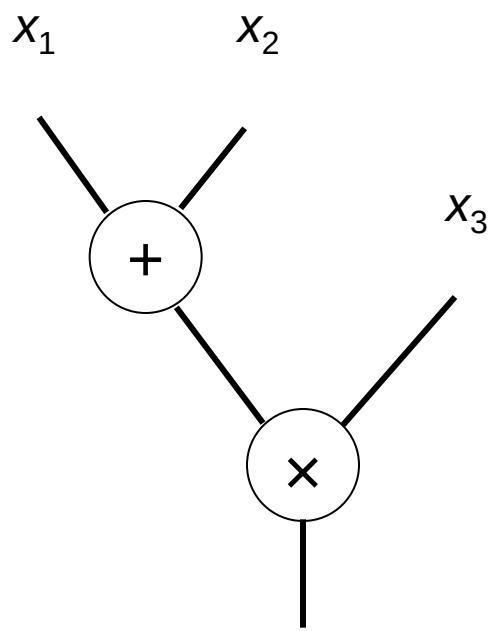
If we had:

- Eval(+, Enc(x_1), Enc(x_2)) $\Rightarrow$ Enc(x_1+x_2)
- Eval($\times$, Enc(x_1), Enc(x_2)) $\Rightarrow$ Enc($x_1 \cdot x_2$)

then we are done.

Arith. Circuit $(+, \times)$ over GF(2).

$f(x_1,x_2,x_3)=(x_1+x_2)\cdot x_3$



Add & Mult Are Universal

(a la [BGW88])

$(+,\times)$ over GF(2) $\equiv$ Boolean (XOR,AND)
= Universal set

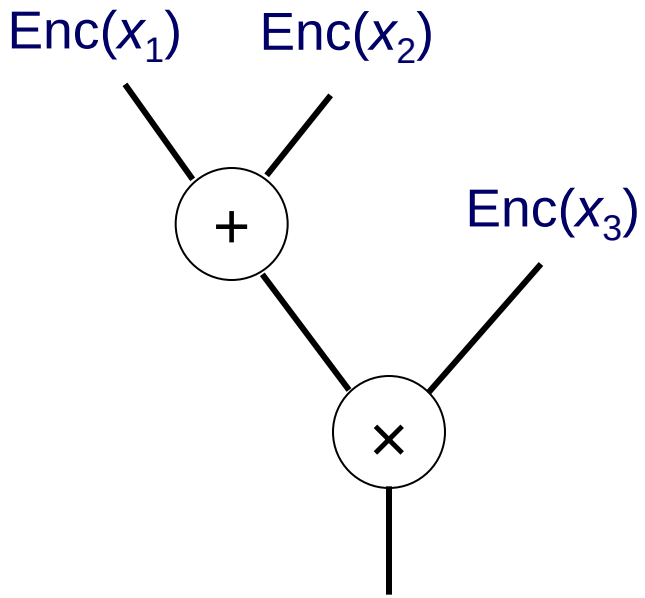
If we had:

- Eval(+, Enc(x_1), Enc(x_2)) $\Rightarrow$ Enc(x_1+x_2)
- Eval($\times$, Enc(x_1), Enc(x_2)) $\Rightarrow$ Enc($x_1 \cdot x_2$)

then we are done.

Arith. Circuit $(+, \times)$ over GF(2).

$f(x_1,x_2,x_3)=(x_1+x_2)\cdot x_3$



Add & Mult Are Universal

(a la [BGW88])

$(+,\times)$ over GF(2) $\equiv$ Boolean (XOR,AND)
= Universal set

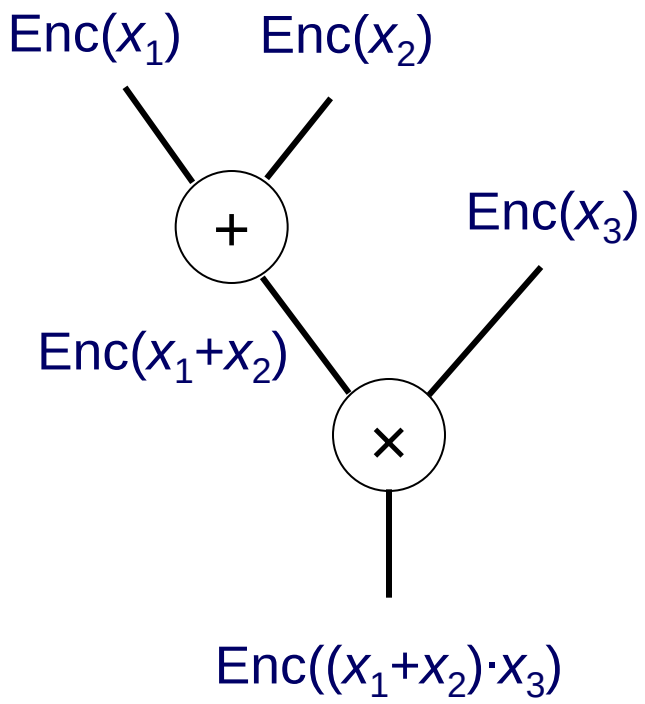
If we had:

- Eval(+, Enc(x_1), Enc(x_2)) $\Rightarrow$ Enc(x_1+x_2)
- Eval($\times$, Enc(x_1), Enc(x_2)) $\Rightarrow$ Enc($x_1 \cdot x_2$)

then we are done.

Arith. Circuit $(+, \times)$ over GF(2).

$f(x_1,x_2,x_3)=(x_1+x_2)\cdot x_3$



Add & Mult Are Universal

(a la [BGW88])

$(+,\times)$ over GF(2) $\equiv$ Boolean (XOR,AND)
= Universal set

If we had:

- $\mathrm{Eval}(+, \mathrm{Enc}(x_1), \mathrm{Enc}(x_2)) \Rightarrow \mathrm{Enc}(x_1+x_2)$
- $\mathrm{Eval}(\times, \mathrm{Enc}(x_1), \mathrm{Enc}(x_2)) \Rightarrow \mathrm{Enc}(x_1 \cdot x_2)$

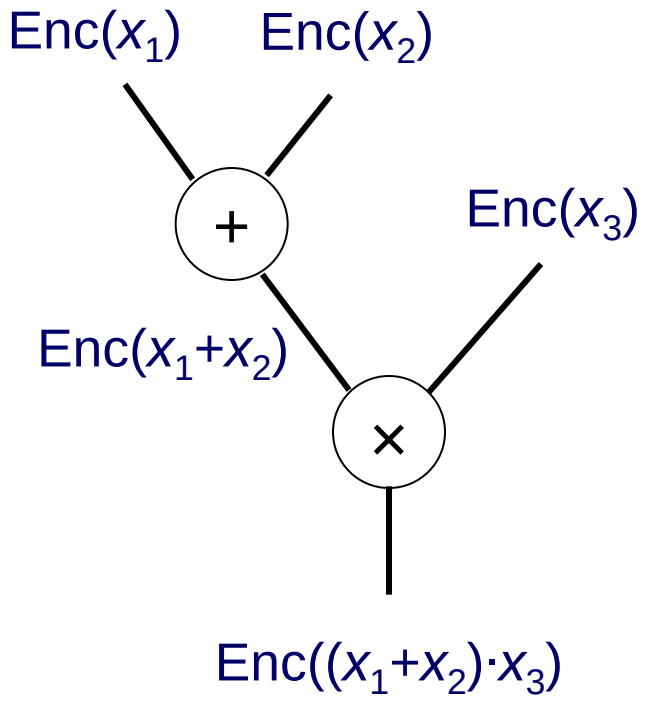
then we are done.

Partial solutions:

- Only add [GM82,P99,R05,…].
- Only mult [G84,…].
- Add + single mult [BGN05,GHV10].
- Add + mult (w/ ciphertext blowup) [SYY99,GHV10,MGH10].

Arith. Circuit $(+, \times)$ over GF(2).

$f(x_1,x_2,x_3)=(x_1+x_2)\cdot x_3$



Gentry's Breakthrough [G09,G10]
First Candidate FHE

Gentry's Breakthrough [G09,G10]
First Candidate FHE

Bootstrapping Theorem [G09]**:** (Qualitative)

"Deep enough" HE $\Rightarrow$ **FHE**

Gentry's Breakthrough [G09,G10]
First Candidate FHE

Bootstrapping Theorem [G09]**:** (Qualitative)

"Deep enough" HE $\Rightarrow$ **FHE**

Deep enough = Deeper than decryption circuit

Gentry's Breakthrough [G09,G10]
First Candidate FHE

Gentry's Breakthrough [G09,G10]
First Candidate FHE

Bootstrapping Theorem [G09]**:**

d-HE + dec. depth $< d$ $\Rightarrow$ “leveled” **FHE**

Gentry's Breakthrough [G09,G10]
First Candidate FHE

evk grows with eval. depth

Bootstrapping Theorem [G09]**:**

d-HE + dec. depth $< d$ $\Rightarrow$ "leveled" **FHE**

Gentry's Breakthrough [G09,G10]
First Candidate FHE

Bootstrapping Theorem [G09]**:**

d-HE + dec. depth < *d* $\Rightarrow$ "leveled" **FHE**

evk grows with eval. depth

Eval for any depth *d* circuit (aka "somewhat" HE)

Gentry's Breakthrough [G09,G10]
First Candidate FHE

evk grows with eval. depth

Bootstrapping Theorem [G09]**:**

d-HE + dec. depth $< d$ $\Rightarrow$ "leveled" **FHE**

+ circular security $\Rightarrow$ **FHE**

Gentry's Breakthrough [G09,G10]
First Candidate FHE

evk grows with eval. depth

Bootstrapping Theorem [G09]**:**

d-HE + dec. depth < *d* $\Rightarrow$ "leveled" **FHE**

+ circular security $\Rightarrow$ **FHE**

Gentry's construction:

Gentry's Breakthrough [G09,G10]
First Candidate FHE

evk grows with eval. depth

Bootstrapping Theorem [G09]**:**

d-HE + dec. depth < *d* $\Rightarrow$ "leveled" **FHE**

+ circular security $\Rightarrow$ **FHE**

Gentry's construction:

d-HE with dec. depth **>** *d*

Ideal lattice assumption.

Gentry's Breakthrough [G09,G10]
First Candidate FHE

evk grows with eval. depth

Bootstrapping Theorem [G09]**:**

d-HE + dec. depth < *d* $\Rightarrow$ "leveled" **FHE**

+ circular security $\Rightarrow$ **FHE**

Gentry's construction:

d-HE with dec. depth > *d*

Ideal lattice assumption.

+

"Squash" to dec. depth < *d*

Sparse Subset-Sum assumption.

Gentry's Breakthrough [G09,G10]
First Candidate FHE

evk grows with eval. depth

Bootstrapping Theorem [G09]**:**

d-HE + dec. depth < *d* $\Rightarrow$ "leveled" **FHE**

+ circular security $\Rightarrow$ **FHE**

Gentry's construction:

d-HE with dec. depth > *d*

Ideal lattice assumption.

+

"Squash" to dec. depth < *d*

Sparse Subset-Sum assumption.

+

Circular security assumption

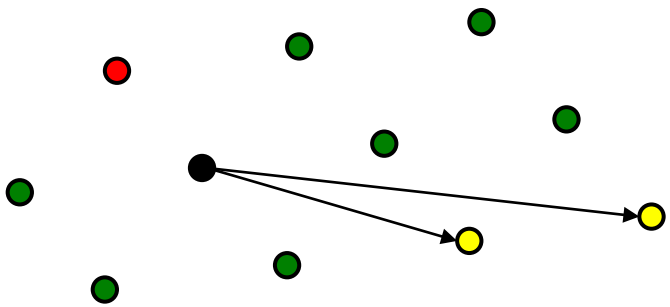
Since Gentry

- Additional candidate FHE schemes:
  - [vDGHV10]: Approx. GCD + sparse subset sum (via squashing).
  - [BV11a]:
    - Ring-LWE + sparse subset sum (via squashing) / "sparse-ring-LWE".
    - Circular secure *d*-HE (not bootstrappable).
- Efficiency improvements of Gentry's scheme [SV10, SS10, GH11].

Q: Arbitrary Lattice?

Q: Arbitrary Lattice?

Short-vectors in lattice:

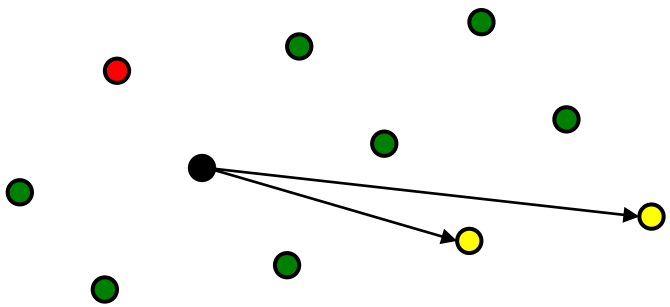


Fundamental algorithmic problem – extensively studied.

[LLL82,K86,A97,M98,AKS03,MR04,MV10]

Q: Arbitrary Lattice?

Short-vectors in lattice:



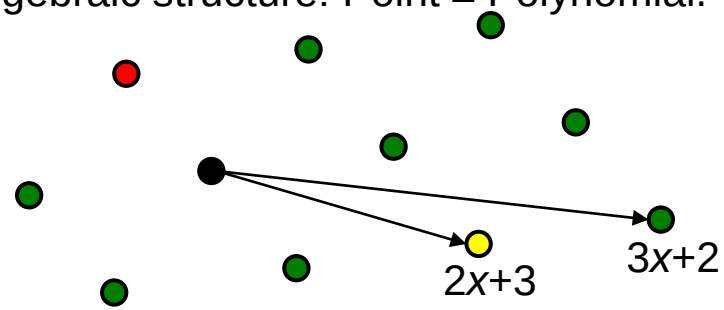
Fundamental algorithmic problem – extensively studied.

[LLL82,K86,A97,M98,AKS03,MR04,MV10]

Vs.

Short-vectors in ideal lattice:

Algebraic structure: Point ≡ Polynomial.

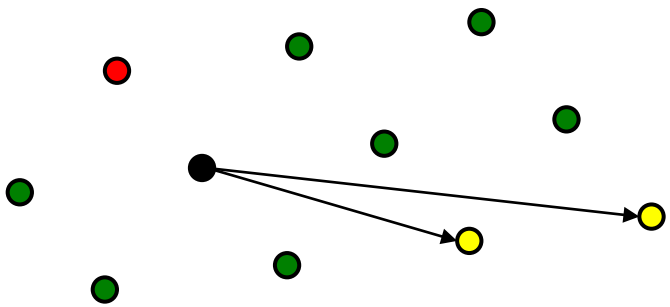


Mostly studied in cryptographic context.

[M04,PR07,LM07,LPR10]

Q: Arbitrary Lattice?

Short-vectors in lattice:



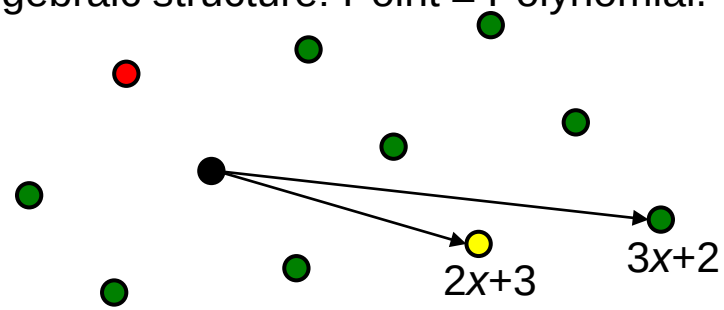
Fundamental algorithmic problem – extensively studied.

[LLL82,K86,A97,M98,AKS03,MR04,MV10]

Vs.

Short-vectors in ideal lattice:

Algebraic structure: Point ≡ Polynomial.



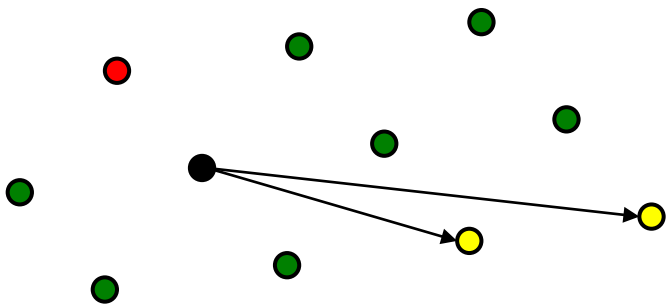
Mostly studied in cryptographic context.

[M04,PR07,LM07,LPR10]

Useful but risky assumption…

Q: Arbitrary Lattice?

Short-vectors in lattice:



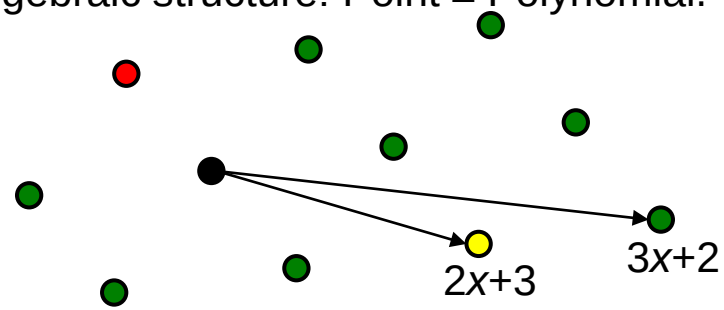
Fundamental algorithmic problem – extensively studied.

[LLL82,K86,A97,M98,AKS03,MR04,MV10]

Vs.

Short-vectors in ideal lattice:

Algebraic structure: Point ≡ Polynomial.



Mostly studied in cryptographic context.

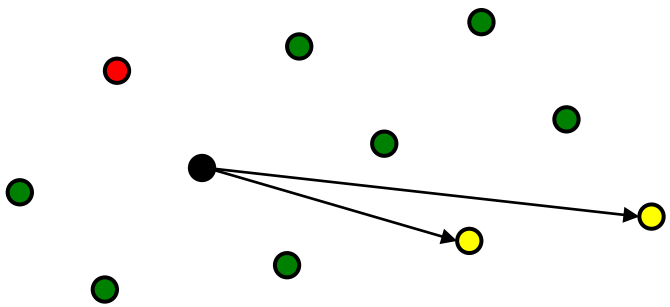
[M04,PR07,LM07,LPR10]

Useful but risky assumption…

Base FHE on standard lattice assumptions?

Q: Arbitrary Lattice?

Short-vectors in lattice:



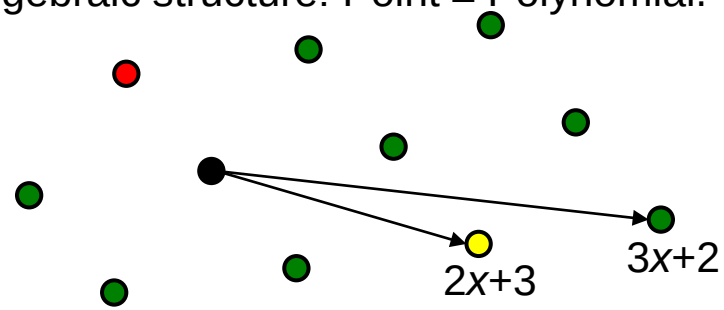
Fundamental algorithmic problem – extensively studied.

[LLL82,K86,A97,M98,AKS03,MR04,MV10]

Vs.

Short-vectors in ideal lattice:

Algebraic structure: Point ≡ Polynomial.



Mostly studied in cryptographic context.

[M04,PR07,LM07,LPR10]

Useful but risky assumption…

Base FHE on standard lattice assumptions?

(Are ideal assumptions inherent to FHE?)

Q: No Squashing?

Q: No Squashing?

Sparse subset-sum assumption and the squashing method:

- Average case assumption, fairly untested.
- Forcing solution (that works!) to “short blanket”.

Q: No Squashing?

Sparse subset-sum assumption and the squashing method:

- Average case assumption, fairly untested.
- Forcing solution (that works!) to “short blanket”.

Do without squashing and additional assumption?

Q: No Squashing?

Sparse subset-sum assumption and the squashing method:

- Average case assumption, fairly untested.
- Forcing solution (that works!) to “short blanket”.

Do without squashing and additional assumption?

Concurrently [GH11]: Remove squashing under ideal lattice assumption (using arith. representation of dec. circuit).

The Real World

The Real World

People actually want to *use* these schemes…

The Real World

People actually want to *use* these schemes…

In known schemes: Key generation and Eval are exhausting.

The Real World

People actually want to *use* these schemes…

In known schemes: Key generation and Eval are exhausting.

Q: Implement FHE efficiently?

Q: Circular Security?

Q: Circular Security?

Two roads to bootstrapping:

Q: Circular Security?

Two roads to bootstrapping:

I. No circular assumption

evk grows with depth.

$\Rightarrow$ “leveled” FHE

Q: Circular Security?

Two roads to bootstrapping:

I. No circular assumption

evk grows with depth.

$\Rightarrow$ "leveled" FHE

II. Assume "circular security"

Encryption of *sk* itself is secure.

$\Rightarrow$ *evk* remains short.

Q: Circular Security?

Two roads to bootstrapping:

I. No circular assumption

evk grows with depth.

$\Rightarrow$ "leveled" FHE

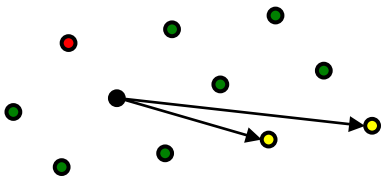
II. Assume "circular security"

Encryption of *sk* itself is secure.

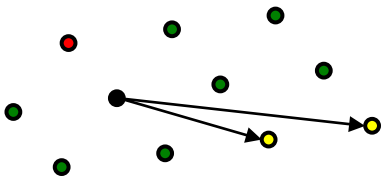
$\Rightarrow$ *evk* remains short.

Short eval. key without additional assumption?

Our Results:
Fully Homomorphic Encryption



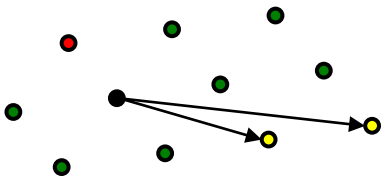
Our Results:
Fully Homomorphic Encryption



Short-vector is hard to approx in worst-case arbitrary lattice.

LWE ("learning with errors") assumption.

Our Results:
Fully Homomorphic Encryption



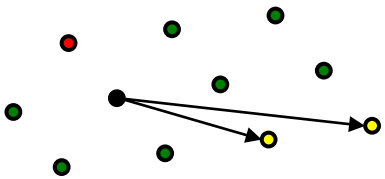
Short-vector is hard to approx in worst-case arbitrary lattice.

LWE ("learning with errors") assumption.

No squashing.

Direct *d*-HE with decryption depth << *d*.

Our Results:
Fully Homomorphic Encryption



Short-vector is hard to approx in worst-case arbitrary lattice.

LWE ("learning with errors") assumption.

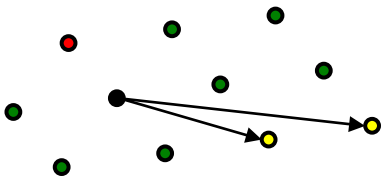
No squashing.

Direct *d*-HE with decryption depth << *d*.

Efficiency improvement.

- Short ciphertext $\Rightarrow$ efficient decryption (as efficient as non-hom. schemes).
- Trivial key generation: no structure required.

Our Results:
Fully Homomorphic Encryption



Short-vector is hard to approx in worst-case arbitrary lattice.

LWE ("learning with errors") assumption.

No squashing.

Direct d-HE with decryption depth $\ll d$.

Efficiency improvement.

- Short ciphertext $\Rightarrow$ efficient decryption (as efficient as non-hom. schemes).
- Trivial key generation: no structure required.

Leveled FHE without bootstrapping.

- Conceptual contribution, although less efficient and worse parameters.

Our Results: Private Information Retrieval (PIR)

- **PIR** = Oblivious retrieval from "huge" DB of size N. [CGKS95,KO97,CMS99]
  - Quality measure: Communication complexity.

- **Our protocol:** $\tilde{O}(\log(N))$-communication.
  - Trivial LB = $\log(N)$. We are nearly optimal!
  - Previous best schemes [CMS99,L05,GR05,G09] $\sim\log^3(N)$.

New Ideas

New Ideas

- **Re-linearization.**

  $\Rightarrow$ “Shallow” multiplicative homomorphism.

- **(Dimension-)Modulus reduction.**

  $\Rightarrow$ Noise management for “deep” circuits.

  $\Rightarrow$ Short ciphertexts as by-product.

Talk Outline

Talk Outline

- The LWE assumption.
- Re-linearization and modulus reduction.
- FHE using bootstrapping.
- (Leveled) FHE without bootstrapping.
- PIR protocol.
- Conclusion and open problems.

Learning With Errors (LWE) [R05]

Learning With Errors (LWE) [R05]

LWE$_{n,q}$ **:** For random secret $\boldsymbol{s} \in \boldsymbol{Z}_q^n$, for any m=poly(n):
sample random $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_m \in \boldsymbol{Z}_q^n$;
"small" noise $e_1,\ldots,e_m \in \mathcal{E}$ ($|e_i| \le T << q$).

$$\left\{\begin{array}{c} (\boldsymbol{a}_1, b_1 = \langle \boldsymbol{a}_1, \boldsymbol{s}\rangle + e_1) \\ (\boldsymbol{a}_2, b_2 = \langle \boldsymbol{a}_2, \boldsymbol{s}\rangle + e_2) \\ \ldots \\ (\boldsymbol{a}_m, b_m = \langle \boldsymbol{a}_m, \boldsymbol{s}\rangle + e_m) \end{array}\right\} \cong \left\{\begin{array}{c} (\boldsymbol{a}_1, u_1) \\ (\boldsymbol{a}_2, u_2) \\ \ldots \\ (\boldsymbol{a}_m, u_m) \end{array}\right\}$$

Learning With Errors (LWE) [R05]

LWE$_{n,q}$: For random secret $\boldsymbol{s} \in \boldsymbol{Z}_q^n$, for any m=poly(n):
sample random $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_m \in \boldsymbol{Z}_q^n$;
"small" noise $e_1,\ldots,e_m \in \mathcal{E}$ ($|e_i| \le T << q$).

$$\left\{ \begin{array}{c} (\boldsymbol{a}_1, b_1 = \langle \boldsymbol{a}_1, \boldsymbol{s} \rangle + e_1) \\ (\boldsymbol{a}_2, b_2 = \langle \boldsymbol{a}_2, \boldsymbol{s} \rangle + e_2) \\ \ldots \\ (\boldsymbol{a}_m, b_m = \underbrace{\langle \boldsymbol{a}_m, \boldsymbol{s} \rangle + e_m}_{\text{"noisy" random linear equation}}) \end{array} \right\} \cong \left\{ \begin{array}{c} (\boldsymbol{a}_1, u_1) \\ (\boldsymbol{a}_2, u_2) \\ \ldots \\ (\boldsymbol{a}_m, \underbrace{u_m}_{\text{uniform in } \boldsymbol{Z}_q}) \end{array} \right\}$$

Learning With Errors (LWE) [R05]

LWE$_{n,q}$ **:** For random secret $\boldsymbol{s} \in \boldsymbol{Z}_q^n$, for any m=poly(n):
sample random $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_m \in \boldsymbol{Z}_q^n$;
"small" noise $e_1,\ldots,e_m \in \mathcal{E}$ ($|e_i| \le T << q$).

$T \approx \sqrt{n}$

$$\left\{ \begin{array}{c} (\boldsymbol{a}_1, b_1 = \langle \boldsymbol{a}_1, \boldsymbol{s} \rangle + e_1) \\ (\boldsymbol{a}_2, b_2 = \langle \boldsymbol{a}_2, \boldsymbol{s} \rangle + e_2) \\ \ldots \\ (\boldsymbol{a}_m, b_m = \underbrace{\langle \boldsymbol{a}_m, \boldsymbol{s} \rangle + e_m}_{\text{"noisy" random linear equation}}) \end{array} \right\} \cong \left\{ \begin{array}{c} (\boldsymbol{a}_1, u_1) \\ (\boldsymbol{a}_2, u_2) \\ \ldots \\ (\boldsymbol{a}_m, \underbrace{u_m}_{\text{uniform in } \boldsymbol{Z}_q}) \end{array} \right\}$$

Learning With Errors (LWE) [R05]

"Hermite normal form"

LWE$_{n,q}$ **:** For random secret $\boldsymbol{s} \in \cancel{\boldsymbol{Z}_q^n}\ \mathcal{E}^n$, for any m=poly(n):
sample random $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_m \in \boldsymbol{Z}_q^n$;
"small" noise $e_1,\ldots,e_m \in \mathcal{E}$ ($|e_i| \le T << q$).

$T \approx \sqrt{n}$

$$\left\{\begin{array}{c} (\boldsymbol{a}_1, b_1 = \langle \boldsymbol{a}_1, \boldsymbol{s}\rangle + e_1) \\ (\boldsymbol{a}_2, b_2 = \langle \boldsymbol{a}_2, \boldsymbol{s}\rangle + e_2) \\ \ldots \\ (\boldsymbol{a}_m, b_m = \underbrace{\langle \boldsymbol{a}_m, \boldsymbol{s}\rangle + e_m}_{\text{"noisy" random linear equation}}) \end{array}\right\} \cong \left\{\begin{array}{c} (\boldsymbol{a}_1, u_1) \\ (\boldsymbol{a}_2, u_2) \\ \ldots \\ (\boldsymbol{a}_m, \underbrace{u_m}_{\text{uniform in } \boldsymbol{Z}_q}) \end{array}\right\}$$

Learning With Errors (LWE) [R05]

"Hermite normal form"

LWE$_{n,q}$: For random secret $\boldsymbol{s} \in \cancel{\boldsymbol{Z}_q^n}\ \mathcal{E}^n$, for any m=poly(n):
sample random $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_m \in \boldsymbol{Z}_q^n$;
"small" noise $e_1,\ldots,e_m \in \mathcal{E}$ ($|e_i| \le T << q$).

$T \approx \sqrt{n}$

$$\left\{\begin{array}{c} (\boldsymbol{a}_1, b_1 = \langle \boldsymbol{a}_1, \boldsymbol{s}\rangle + e_1) \\ (\boldsymbol{a}_2, b_2 = \langle \boldsymbol{a}_2, \boldsymbol{s}\rangle + e_2) \\ \ldots \\ (\boldsymbol{a}_m, b_m = \underbrace{\langle \boldsymbol{a}_m, \boldsymbol{s}\rangle + e_m}_{\text{"noisy" random linear equation}}) \end{array}\right\} \cong \left\{\begin{array}{c} (\boldsymbol{a}_1, u_1) \\ (\boldsymbol{a}_2, u_2) \\ \ldots \\ (\boldsymbol{a}_m, \underbrace{u_m}_{\text{uniform in } \boldsymbol{Z}_q}) \end{array}\right\}$$

Worst-Case Hardness (quantum [R05] **/ classical** [P09]**):**

Solve LWE $\Rightarrow$ (q/T)-apx. for short-vector problems in *worst-case* lattices.

(Best known apx. factor: 2^n in poly time; or $2^{\sqrt{n}}$ in $2^{\sqrt{n}}$ time.)

Symmetric Encryption with LWE

Symmetric Encryption with LWE

(omitting public-key part)

Symmetric Encryption with LWE

(omitting public-key part)

- **KeyGen:**
  - Sample random $\boldsymbol{s} \in \mathcal{E}^n$ and set $sk = \boldsymbol{s}$.

- **Bit encryption** $\mathrm{Enc}_{\boldsymbol{s}}(m)$ **:**
  - Sample random $\boldsymbol{a} \in \boldsymbol{Z}_q^n$ and noise e, output $(\boldsymbol{a}, b) \in \boldsymbol{Z}_q^n \times \boldsymbol{Z}_q$,

    where $b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + 2e + m \in \boldsymbol{Z}_q$.

  - **Semantic security by LWE:** $(\boldsymbol{a}, b) \cong (\boldsymbol{a}, u)$

    (odd modulus $q \Rightarrow 2$ is invertible in $\boldsymbol{Z}_q$)

- **Decryption:** $\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{a}, b) = (\, b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle \,) \pmod 2$.

  - **Correctness:** $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle = b - \sum a[\,i\,] \cdot s[\,i\,] = m + 2e$ (over $\boldsymbol{Z}_q$).

    $\Rightarrow$ decryption succeeds if $e < q/4$.

Symmetric Encryption with LWE

(omitting public-key part)

- **KeyGen:**
  - Sample random $\boldsymbol{s} \in \mathcal{E}^n$ and set $sk = \boldsymbol{s}$.

- **Bit encryption** $\mathrm{Enc}_{\boldsymbol{s}}(m)$ **:**
  - Sample random $\boldsymbol{a} \in \boldsymbol{Z}_q^n$ and noise e, output $(\boldsymbol{a}, b) \in \boldsymbol{Z}_q^n \times \boldsymbol{Z}_q$,

    where $b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + 2e + m \in \boldsymbol{Z}_q$.

  - **Semantic security by LWE:** $(\boldsymbol{a}, b) \cong (\boldsymbol{a}, u)$

    (odd modulus $q \Rightarrow 2$ is invertible in $\boldsymbol{Z}_q$)

- **Decryption:** $\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{a}, b) = (\, b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle \,) \pmod 2$.

  - **Correctness:** $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle = b - \sum a[\,i\,] \cdot s[\,i\,] = m + 2e$ (over $\boldsymbol{Z}_q$).

    $\Rightarrow$ decryption succeeds if $e < q/4$.

Symmetric Encryption with LWE

(omitting public-key part)

- **KeyGen:**
  - Sample random $\boldsymbol{s} \in \mathcal{E}^n$ and set $sk = \boldsymbol{s}$.

- **Bit encryption** $\mathrm{Enc}_{\boldsymbol{s}}(m)$ **:**
  - Sample random $\boldsymbol{a} \in \boldsymbol{Z}_q^n$ and noise e, output $(\boldsymbol{a}, b) \in \boldsymbol{Z}_q^n \times \boldsymbol{Z}_q$,

    where $b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + 2e + m \in \boldsymbol{Z}_q$.

  - **Semantic security by LWE:** $(\boldsymbol{a}, b) \cong (\boldsymbol{a}, u)$

    (odd modulus $q \Rightarrow 2$ is invertible in $\boldsymbol{Z}_q$)

- **Decryption:** $\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{a}, b) = (\, b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle \,) \pmod 2$.

  - **Correctness:** $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle = b - \sum a[\,i\,] \cdot s[\,i\,] = m + 2e$ (over $\boldsymbol{Z}_q$).

    $\Rightarrow$ decryption succeeds if $e < q/4$.

Additive Homomorphism

Additive Homomorphism

$(\boldsymbol{a}, b)$

$b - \sum a[\,i\,]\cdot s[\,i\,] = m + 2e$

$(\boldsymbol{a'}, b')$

$b' - \sum a'[\,i\,]\cdot s[\,i\,] = m' + 2e'$

Additive Homomorphism

$(\boldsymbol{a}, b)$ $\qquad$ $(\boldsymbol{a'}, b')$

$b - \sum a[i] \cdot s[i] = m + 2e$ $\qquad$ $b' - \sum a'[i] \cdot s[i] = m' + 2e'$

Add the ciphertexts: $(\boldsymbol{a}_{add}, b_{add}) = (\boldsymbol{a} + \boldsymbol{a'}, b+b')$

$$b - \sum a[i] \cdot s[i] = m + 2e$$

$$b' - \sum a'[i] \cdot s[i] = m' + 2e'$$

$$(b+b') - \sum \left(a[i] + a'[i]\right) \cdot s[i] = (m+m') + 2(e+e')$$

$$\Rightarrow \mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{a}_{add}, b_{add}) = (m+m')+2e'' \pmod 2 = (m+m') \pmod 2$$

Additive Homomorphism

$(\boldsymbol{a}, b)$ $\qquad$ $(\boldsymbol{a'}, b')$

$b - \sum a[i] \cdot s[i] = m + 2e$ $\qquad$ $b' - \sum a'[i] \cdot s[i] = m' + 2e'$

Add the ciphertexts: $(\boldsymbol{a}_{\text{add}}, b_{\text{add}}) = (\boldsymbol{a} + \boldsymbol{a'}, b+b')$

$$
\begin{array}{rl}
 & b - \sum a[i] \cdot s[i] = m + 2e \\
+ & b' - \sum a'[i] \cdot s[i] = m' + 2e' \\
\hline
 & (b+b') - \sum \left(a[i] + a'[i]\right) \cdot s[i] = (m+m') + 2(e+e')
\end{array}
$$

$\Rightarrow \text{Dec}_{\boldsymbol{s}}(\boldsymbol{a}_{\text{add}}, b_{\text{add}}) = (m+m')+2e'' \pmod 2 = (m+m') \pmod 2$

Additive Homomorphism

$(\boldsymbol{a}, b)$ $\qquad\qquad$ $(\boldsymbol{a}', b')$

$b - \sum a[i] \cdot s[i] = m + 2e$ $\qquad\qquad$ $b' - \sum a'[i] \cdot s[i] = m' + 2e'$

Add the ciphertexts: $(\boldsymbol{a}_{\text{add}}, b_{\text{add}}) = (\boldsymbol{a} + \boldsymbol{a}', b+b')$

$$
\begin{array}{rl}
 & b - \sum a[i] \cdot s[i] = m + 2e \\
+ & b' - \sum a'[i] \cdot s[i] = m' + 2e' \\
\hline
 & (b+b') - \sum \left(a[i] + a'[i]\right) \cdot s[i] = (m+m') + 2\underbrace{(e+e')}_{e''}
\end{array}
$$

$\Rightarrow \text{Dec}_{\boldsymbol{s}}(\boldsymbol{a}_{\text{add}}, b_{\text{add}}) = (m+m') + 2e'' \pmod 2 = (m+m') \pmod 2$

Multiplicative Homomorphism

$(\boldsymbol{a}, b)$

$b - \sum a[\,i\,]\cdot s[\,i\,] = m + 2e$

$(\boldsymbol{a'}, b')$

$b' - \sum a'[\,i\,]\cdot s[\,i\,] = m' + 2e'$

Multiplicative Homomorphism

$(\boldsymbol{a}, b)$ $\qquad$ $(\boldsymbol{a'}, b')$

$b - \sum a[i] \cdot s[i] = m + 2e$ $\qquad$ $b' - \sum a'[i] \cdot s[i] = m' + 2e'$

Multiply ciphertexts?

$$b - \sum a[i] \cdot s[i] = m + 2e$$

$$b' - \sum a'[i] \cdot s[i] = m' + 2e'$$

$$\left(b - \sum a[i] \cdot s[i]\right) \cdot \left(b' - \sum a'[i] \cdot s[i]\right) = (m+2e) \cdot (m'+2e')$$

$$h_0 + \sum h_i \, s[i] + \sum h_{i,j} \, s[i] s[j] = mm' + 2(2ee' + me' + m'e)$$

Multiplicative Homomorphism

$(\boldsymbol{a}, b)$ $\qquad$ $(\boldsymbol{a'}, b')$

$b - \sum a[i] \cdot s[i] = m + 2e$ $\qquad$ $b' - \sum a'[i] \cdot s[i] = m' + 2e'$

Multiply ciphertexts?

$$\begin{array}{r} b - \sum a[i] \cdot s[i] = m + 2e \\ \times \quad b' - \sum a'[i] \cdot s[i] = m' + 2e' \\ \hline \end{array}$$

$$\left(b - \sum a[i] \cdot s[i]\right) \cdot \left(b' - \sum a'[i] \cdot s[i]\right) = (m+2e) \cdot (m'+2e')$$

$$h_0 + \sum h_i \, s[i] + \sum h_{i,j} \, s[i] s[j] = mm' + 2(2ee' + me' + m'e)$$

Multiplicative Homomorphism

$(\boldsymbol{a}, b)$ $\qquad$ $(\boldsymbol{a'}, b')$

$b - \sum a[i] \cdot s[i] = m + 2e$ $\qquad$ $b' - \sum a'[i] \cdot s[i] = m' + 2e'$

Multiply ciphertexts?

$$\begin{array}{rl} & b - \sum a[i] \cdot s[i] = m + 2e \\ \times & b' - \sum a'[i] \cdot s[i] = m' + 2e' \\ \hline \end{array}$$

$$\left(b - \sum a[i] \cdot s[i]\right) \cdot \left(b' - \sum a'[i] \cdot s[i]\right) = (m+2e) \cdot (m'+2e')$$

$$h_0 + \sum h_i \, s[i] + \sum h_{i,j} \, s[i] s[j] = mm' + 2(\underbrace{2ee' + me' + m'e}_{e'' \,:\, |e''| \approx |e|^2 \leq T^2})$$

Multiplicative Homomorphism

$(\boldsymbol{a}, b)$ $\qquad$ $(\boldsymbol{a'}, b')$

$b - \sum a[i] \cdot s[i] = m + 2e$ $\qquad$ $b' - \sum a'[i] \cdot s[i] = m' + 2e'$

Multiply ciphertexts?

$$\begin{array}{rl} & b - \sum a[i] \cdot s[i] = m + 2e \\ \times & b' - \sum a'[i] \cdot s[i] = m' + 2e' \\ \hline \end{array}$$

$$\left(b - \sum a[i] \cdot s[i]\right) \cdot \left(b' - \sum a'[i] \cdot s[i]\right) = (m+2e) \cdot (m'+2e')$$

$$\underbrace{h_0 + \sum h_i\, s[i] + \sum h_{i,j}\, s[i] s[j]} = mm' + 2(\underbrace{2ee' + me' + m'e}_{e'' \,:\, |e''| \approx |e|^2 \le T^2})$$

The coefficients h_i, $h_{i,j}$ are computable from input ciphertexts $(\boldsymbol{a}, b)$, $(\boldsymbol{a'}, b')$.

What's the output ciphertext?

Multiplicative Homomorphism

What's the Ciphertext?

$$h_0 + \sum h_i\, s[i] + \sum h_{i,j}\, s[i]s[j] = mm' + 2e''$$

- **Ciphertext = Coefficients $\{h_i\}$, $\{h_{i,j}\}$?**
  - Decrypt with secret key $\boldsymbol{s}$:

$$h_0 + \sum h_i\, s[i] + \sum h_{i,j}\, s[i]s[j] \pmod 2$$

$$= mm' + 2e'' \pmod 2$$

$$= mm' \pmod 2.$$

- **Problem:** Ciphertext contains $\sim n^2$ elements.
  - Size blows up with number of multiplications.

Find a more compact representation?

Re-Linearization

$$h_0 + \sum h_i \, s[i] + \sum h_{i,j} \, s[i] s[j] = mm' + 2e''$$

Re-Linearization

$$h_0 + \sum h_i \, s[i] + \sum h_{i,j} \, s[i] s[j] = mm' + 2e''$$

Find linear function of ***s*** that represents this quadratic func.

Re-Linearization

$$h_0 + \sum h_i \, s[i] + \sum h_{i,j} \, s[i] s[j] = mm' + 2e''$$

Find linear function ~~of **s**~~ of new secret ***s'*** that represents this quadratic func.

Re-Linearization

$$h_0 + \sum h_i\, s[i] + \sum h_{i,j}\, s[i]s[j] = mm' + 2e''$$

Find linear function ~~of $\boldsymbol{s}$~~ of new secret $\boldsymbol{s'}$ that represents this quadratic func.

New **KeyGen**:

- Sample $\boldsymbol{s}, \boldsymbol{s'} \in \boldsymbol{Z}_q^n$ and set $sk = (\boldsymbol{s}, \boldsymbol{s'})$.
- Evaluation key evk : sample $\boldsymbol{A}_{i,j}$, $E_{i,j}$

$$\forall i,j. \quad \left(\boldsymbol{A}_{i,j}\,,\, B_{i,j} = \langle \boldsymbol{A}_{i,j}\,,\, \boldsymbol{s'} \rangle + 2E_{i,j} + s[i]s[j]\right)$$

$$\forall i. \quad \left(\boldsymbol{A}_{i}\,,\, B_{i} = \langle \boldsymbol{A}_{i}\,,\, \boldsymbol{s'} \rangle + 2E_{i} + s[i]\right)$$

- **We get:** $s[i]s[j] \approx B_{i,j} - \langle \boldsymbol{A}_{i,j}\,,\, \boldsymbol{s'} \rangle$

Re-Linearization

$$h_0 + \sum h_i\, s[i] + \sum h_{i,j}\, s[i]s[j] = mm' + 2e''$$

Find linear function ~~of $\boldsymbol{s}$~~ of new secret $\boldsymbol{s'}$ that represents this quadratic func.

New **KeyGen**:

- Sample $\boldsymbol{s}, \boldsymbol{s'} \in \boldsymbol{Z}_q^n$ and set $sk = (\boldsymbol{s}, \boldsymbol{s'})$.
- Evaluation key evk : sample $\boldsymbol{A}_{i,j}$, $E_{i,j}$

$$\left.\begin{array}{ll} \forall i,j. & \left(\boldsymbol{A}_{i,j}\,,\, B_{i,j} = \langle \boldsymbol{A}_{i,j}\,,\, \boldsymbol{s'} \rangle + 2E_{i,j} + s[i]s[j]\right) \\ \forall i. & \left(\boldsymbol{A}_{i}\,,\, B_{i} = \langle \boldsymbol{A}_{i}\,,\, \boldsymbol{s'} \rangle + 2E_{i} + s[i]\right) \end{array}\right\} \text{LWE} \Rightarrow \text{Security still holds.}$$

- **We get:** $s[i]s[j] \approx B_{i,j} - \langle \boldsymbol{A}_{i,j}\,,\, \boldsymbol{s'} \rangle$

Re-Linearization

$$h_0 + \sum h_i\, s[i] + \sum h_{i,j}\, s[i]s[j] = mm' + 2e''$$

Find linear function ~~of $\boldsymbol{s}$~~ $\underbrace{}_{\text{of new secret } \boldsymbol{s'}}$ that represents this quadratic func.

New **KeyGen**:

- Sample $\boldsymbol{s}, \boldsymbol{s'} \in \boldsymbol{Z}_q^n$ and set $sk = (\boldsymbol{s}, \boldsymbol{s'})$.
- Evaluation key evk : sample $\boldsymbol{A}_{i,j}$, $E_{i,j}$

$$\left.\begin{array}{ll} \forall i,j. & \left(\boldsymbol{A}_{i,j}\,,\ B_{i,j} = \langle \boldsymbol{A}_{i,j}\,,\ \boldsymbol{s'}\rangle + 2E_{i,j} + s[i]s[j]\right) \\ \forall i. & \left(\boldsymbol{A}_{i}\,,\ B_{i} = \langle \boldsymbol{A}_{i}\,,\ \boldsymbol{s'}\rangle + 2E_{i} + s[i]\right) \end{array}\right\} \text{LWE} \Rightarrow \text{Security still holds.}$$

- **We get:** $\underbrace{s[i]s[j]}_{\substack{\textit{Quadratic}\text{ function} \\ (\text{in } \boldsymbol{s})}} \approx \underbrace{B_{i,j} - \langle \boldsymbol{A}_{i,j}\,,\ \boldsymbol{s'}\rangle}_{\substack{\textit{Linear}\text{ function} \\ (\text{in } \boldsymbol{s'})}}$

Re-Linearization

$$h_0 + \sum h_i\, s[i] + \sum h_{i,j}\, s[i]s[j] = mm' + 2e''$$

Find linear function ~~of $\boldsymbol{s}$~~ $\overbrace{}^{\text{of new secret } \boldsymbol{s'}}$ that represents this quadratic func.

New **KeyGen**:

- Sample $\boldsymbol{s}, \boldsymbol{s'} \in \boldsymbol{Z}_q^n$ and set $sk = (\boldsymbol{s}, \boldsymbol{s'})$.
- Evaluation key evk : sample $\boldsymbol{A}_{i,j}$, $E_{i,j}$

$$\left.\begin{array}{ll} \forall i,j. & \left(\boldsymbol{A}_{i,j}\,,\ B_{i,j} = \langle \boldsymbol{A}_{i,j}\,,\ \boldsymbol{s'}\rangle + 2E_{i,j} + s[i]s[j]\right) \\ \forall i. & \left(\boldsymbol{A}_{i}\,,\ B_{i} = \langle \boldsymbol{A}_{i}\,,\ \boldsymbol{s'}\rangle + 2E_{i} + s[i]\right) \end{array}\right\} \text{LWE} \Rightarrow \text{Security still holds.}$$

- **We get:** $\boxed{\underbrace{s[i]s[j]}_{\substack{\textit{Quadratic}\text{ function} \\ (\text{in } \boldsymbol{s})}} \approx \underbrace{B_{i,j} - \langle \boldsymbol{A}_{i,j}\,,\ \boldsymbol{s'}\rangle}_{\substack{\textit{Linear}\text{ function} \\ (\text{in } \boldsymbol{s'})}}}$

Re-Linearization

$$h_0+\sum h_i\, s[i] + \sum h_{i,j}\, s[i]s[j] = mm'+2e''$$

Plug back into quadratic equation:

$$\underbrace{h_0+\sum h_i\,(B_i - \langle \boldsymbol{A}_i, \boldsymbol{s}'\rangle) + \sum h_{i,j}\,(B_{i,j} - \langle \boldsymbol{A}_{i,j}, \boldsymbol{s}'\rangle)}_{\text{Linear in } \boldsymbol{s}'} = mm'+2e'''$$

c func.

Nev

- Sample $\boldsymbol{s},\boldsymbol{s}' \in \boldsymbol{Z}_q^n$ and set $sk = (\boldsymbol{s},\boldsymbol{s}')$.
- Evaluation key evk : sample $\boldsymbol{A}_{i,j}$, $E_{i,j}$

$$\forall i,j. \quad (\boldsymbol{A}_{i,j}, B_{i,j} = \langle \boldsymbol{A}_{i,j}, \boldsymbol{s}'\rangle + 2E_{i,j} + s[i]s[j])$$

$$\forall i. \quad (\boldsymbol{A}_i, B_i = \langle \boldsymbol{A}_i, \boldsymbol{s}'\rangle + 2E_i + s[i])$$

LWE $\Rightarrow$ Security still holds.

- t:

$$\underbrace{s[i]s[j]}_{\substack{\textit{Quadratic}\text{ function}\\ (\text{in } \boldsymbol{s})}} \approx \underbrace{B_{i,j} - \langle \boldsymbol{A}_{i,j}, \boldsymbol{s}'\rangle}_{\substack{\textit{Linear}\text{ function}\\ (\text{in } \boldsymbol{s}')}}$$

Re-Linearization

$$h_0 + \sum h_i \left(B_i - \langle \boldsymbol{A}_i , \boldsymbol{s}' \rangle\right) + \sum h_{i,j} \left(B_{i,j} - \langle \boldsymbol{A}_{i,j} , \boldsymbol{s}' \rangle\right) = mm' + 2e'''$$

Re-Linearization

$$h_0+\sum h_i \left(B_i - \langle \boldsymbol{A}_i , \boldsymbol{s'}\rangle\right) + \sum h_{i,j} \left(B_{i,j} - \langle \boldsymbol{A}_{i,j} , \boldsymbol{s'}\rangle\right) = mm'+2e'''$$

Regrouping, we can now define $(\boldsymbol{a}_{\text{mult}}, b_{\text{mult}})$:

$$\boldsymbol{a}_{\text{mult}} = \sum h_i \boldsymbol{A}_i + \sum h_{i,j} \boldsymbol{A}_{i,j} \in \boldsymbol{Z}_q^n$$

$$b_{\text{mult}} = h_0+\sum h_i B_i + \sum h_{i,j} B_{i,j} \in \boldsymbol{Z}_q$$

Correctness:

$$\begin{aligned}
\text{Dec}_{\boldsymbol{s'}}(\boldsymbol{a}_{\text{mult}}, b_{\text{mult}}) &= \left(b_{\text{mult}} - \langle \boldsymbol{a}_{\text{mult}}, \boldsymbol{s'}\rangle\right) \pmod 2 \\
&= \left(h_0+\sum h_i B_i + \sum h_{i,j} B_{i,j}\right) - \left\langle \left(\sum h_i \boldsymbol{A}_i + \sum h_{i,j} \boldsymbol{A}_{i,j}\right), \boldsymbol{s'}\right\rangle \pmod 2 \\
&= h_0+\sum h_i (B_i - \langle \boldsymbol{A}_i , \boldsymbol{s'}\rangle) + \sum h_{i,j} (B_{i,j} - \langle \boldsymbol{A}_{i,j} , \boldsymbol{s'}\rangle) \pmod 2 \\
&= m \cdot m' \pmod 2
\end{aligned}$$

Re-Linearization

$$h_0 + \sum h_i \left(B_i - \langle \boldsymbol{A}_i , \boldsymbol{s}' \rangle\right) + \sum h_{i,j} \left(B_{i,j} - \langle \boldsymbol{A}_{i,j} , \boldsymbol{s}' \rangle\right) = mm' + 2e'''$$

Single multiplication accomplished!

For greater depth, repeat using new secrets ***s''***, ***s'''***…

Re-Linearization

$$h_0+\sum h_i\,(B_i - \langle \boldsymbol{A}_i\,,\,\boldsymbol{s'}\rangle) + \sum h_{i,j}\,(B_{i,j} - \langle \boldsymbol{A}_{i,j}\,,\,\boldsymbol{s'}\rangle) = mm'+2e'''$$

Single multiplication accomplished!

For greater depth, repeat using new secrets ***s'', s'''***…

Problem: Noise amplitude grows from T to $\approx T^2$.

(Actually, from nT to $(nT)^2$.)

Modulus Reduction: Reducing Noise Amplitude

Modulus Reduction: Reducing Noise Amplitude

Naïve approach: Divide everything by nT.

Modulus Reduction: Reducing Noise Amplitude

Naïve approach: Divide everything by nT.

Can you believe that this actually works?

Modulus Reduction: Reducing Noise Amplitude

Naïve approach: Divide everything by nT.

Can you believe that this actually works?

$$(\mathbf{a}, b) \in \mathbf{Z}_q^{\,n} \times \mathbf{Z}_q \longrightarrow (\mathbf{a}^*, b^*) \in \mathbf{Z}_{(q/nT)}^{\,n} \times \mathbf{Z}_{(q/nT)}$$

$$a^*_{[i]} = a_{[i]}/(nT) \; ; \; b^* = b/(nT)$$

Modulus Reduction: Reducing Noise Amplitude

Naïve approach: Divide everything by nT.

Can you believe that this actually works?

$$(\mathbf{a}, b) \in \mathbf{Z}_q^{\,n} \times \mathbf{Z}_q \longrightarrow (\mathbf{a}^*, b^*) \in \mathbf{Z}_{(q/nT)}^{\,n} \times \mathbf{Z}_{(q/nT)}$$

$$a^*_{[i]} = a_{[i]}/(nT) \; ; \; b^* = b/(nT)$$

Special rounding: Round so that **LSB** doesn't change.

Modulus Reduction: Reducing Noise Amplitude

Naïve approach: Divide everything by nT.

Can you believe that this actually works?

$$(\mathbf{a}, b) \in \mathbf{Z}_q^{\,n} \times \mathbf{Z}_q \;\Longrightarrow\; (\mathbf{a}^*, b^*) \in \mathbf{Z}_{(q/nT)}^{\,n} \times \mathbf{Z}_{(q/nT)}$$

$$a^*_{[i]} = a_{[i]}/(nT)\ ;\ b^* = b/(nT)$$

Special rounding: Round so that **LSB** doesn't change. Why?

Modulus Reduction: Reducing Noise Amplitude

Naïve approach: Divide everything by nT.

Can you believe that this actually works?

$$(\mathbf{a}, b) \in \mathbf{Z}_q^n \times \mathbf{Z}_q \longrightarrow (\mathbf{a}^*, b^*) \in \mathbf{Z}_{(q/nT)}^n \times \mathbf{Z}_{(q/nT)}$$

$$a^*_{[i]} = a_{[i]}/(nT) \; ; \; b^* = b/(nT)$$

Special rounding: Round so that **LSB** doesn't change. Why?

Rounding guarantees correct decryption:

$$b^* - \sum a^*[i] \cdot s[i] \pmod 2 \;=\; b - \sum a[i] \cdot s[i] \pmod 2 \;=\; m$$

Modulus Reduction: Reducing Noise Amplitude

Naïve approach: Divide everything by nT.

Can you believe that this actually works?

$$(\mathbf{a}, b) \in \mathbf{Z}_q^n \times \mathbf{Z}_q \longrightarrow (\mathbf{a}^*, b^*) \in \mathbf{Z}_{(q/nT)}^n \times \mathbf{Z}_{(q/nT)}$$

$$a^*[i] = a[i]/(nT) \; ; \; b^* = b/(nT)$$

Special rounding: Round so that **LSB** doesn't change. Why?

Rounding guarantees correct decryption:

$$b^* - \sum a^*[i] \cdot s[i] \pmod 2 \;=\; b - \sum a[i] \cdot s[i] \pmod 2 \;=\; m$$

Noise amplitude after modulus reduction:

$$\underbrace{(nT)^2/(nT)}_{\text{scaled down "original" noise}} + \underbrace{nT}_{\text{rounding error}} = O(nT)$$

Re-linearization and Modulus Reduction – Recap

Re-linearization and Modulus Reduction – Recap

- Eval for mult. depth = 1.
- Modulus reduced: $q \Rightarrow q/(nT)$.
- Noise amplitude at most nT.

Deeper Circuit? Just repeat!

- Mult. depth = d.
- Modulus reduced: $q \Rightarrow q/(nT)^d$.
- Noise amplitude at most nT.

Re-linearization and Modulus Reduction – Recap

- Eval for mult. depth = 1.
- Modulus reduced: $q \Rightarrow q/(nT)$.
- Noise amplitude at most nT.

Deeper Circuit? Just repeat!

- Mult. depth = d.
- Modulus reduced: $q \Rightarrow q/(nT)^d$.
- Noise amplitude at most nT.

Ciphertext decryptable if:

$$nT < (q/(nT)^d)/4$$

$$\Rightarrow q > 4(nT)^{d+1} = n^{O(d)}$$

Can We Bootstrap?

Can We Bootstrap?

Need d-HE scheme with decryption depth < d.

Can We Bootstrap?

Need d-HE scheme with decryption depth < d.

Our decryption depth:

$$d = \log(n) + \log\log(nT) = O(\log(n))$$

Independent of q !

Can We Bootstrap?

Need d-HE scheme with decryption depth < d.

Our decryption depth:

$$d = \log(n) + \log\log(nT) = O(\log(n))$$

Independent of q !

We decrypt **after** the hom. eval., so q is already reduced all the way down to $O(nT)$.

Can We Bootstrap?

Need d-HE scheme with decryption depth < d.

Our decryption depth:

$$d = \log(n) + \log\log(nT) = O(\log(n))$$

Independent of q !

We decrypt **after** the hom. eval., so q is already reduced all the way down to $O(nT)$.

To bootstrap:

$$q = n^{O(d)} = n^{O(\log n)}$$

Can We Bootstrap?

Need d-HE scheme with decryption depth < d.

Our decryption depth:

$$d = \log(n) + \log\log(nT) = O(\log(n))$$

Independent of q !

We decrypt **after** the hom. eval., so q is already reduced all the way down to $O(nT)$.

To bootstrap:

$$q = n^{O(d)} = n^{O(\log n)}$$

Is this secure?

Can We Bootstrap?

Need d-HE scheme with decryption depth < d.

Our decryption depth:

$$d = \log(n) + \log\log(nT) = O(\log(n))$$

Independent of q !

We decrypt **after** the hom. eval., so q is already reduced all the way down to $O(nT)$.

To bootstrap:

$$q = n^{O(d)} = n^{O(\log n)}$$

Is this secure?

Very much so! As hard as quasi-poly apx. to short vectors in arbitrary lattices! (Not known even in time $2^{n^{1-\epsilon}}$.)

FHE with Bootstrapping

- No “squashing” whatsoever.
- Ciphertext size: $n \log (nT)$.
  - Post-bootstrapping size “survives”.
- Decryption depth: $O(\log(n))$.
- Security: $\mathbf{LWE}_{n,q=n^{O(\log n)}}$
  $\Rightarrow$ quasy-poly apx. to short-vectors.

FHE with Bootstrapping

- No “squashing” whatsoever.
- Ciphertext size: $n \log (nT)$.
  - Post-bootstrapping size “survives”.
- Decryption depth: $O(\log(n))$.
- Security: $\mathbf{LWE}_{n,q=n^{O(\log n)}}$
  $\Rightarrow$ quasy-poly apx. to short-vectors.

Assumption is “too good”…

Can we get more features from a (somewhat) stronger assumption?

FHE with Bootstrapping

- No “squashing” whatsoever.
- Ciphertext size: $n \log (nT)$.
  - Post-bootstrapping size “survives”.
- Decryption depth: $O(\log(n))$.
- Security: $\mathbf{LWE}_{n,q=n^{O(\log n)}}$
  $\Rightarrow$ quasy-poly apx. to short-vectors.

Assumption is “too good”…

Can we get more features from a (somewhat) stronger assumption?

Yes…

(Leveled) FHE *without* Bootstrapping

(Leveled) FHE *without* Bootstrapping

Recall:

$$q = n^{O(d)}$$

(Leveled) FHE *without* Bootstrapping

Recall:

$$q = n^{O(d)}$$

Bigger $q \Rightarrow$ deeper circuits.

So how big can q get (securely)?

(Leveled) FHE *without* Bootstrapping

Recall:

$$q = n^{O(d)}$$

Bigger $q \Rightarrow$ deeper circuits.

So how big can q get (securely)?

Best known attacks run in time $\sim 2^{\Omega(n/\log q)}$

$$\Rightarrow q = 2^{\sqrt{n}} \text{ is fairly safe}$$

$$\Rightarrow d = \widetilde{\Omega}(\sqrt{n})$$

(Leveled) FHE *without* Bootstrapping

Recall:

$$q = n^{O(d)}$$

Bigger $q \Rightarrow$ deeper circuits.

So how big can q get (securely)?

Best known attacks run in time $\sim 2^{\Omega(n/\log q)}$

$$\Rightarrow q = 2^{\sqrt{n}} \text{ is fairly safe}$$

$$\Rightarrow d = \widetilde{\Omega}(\sqrt{n})$$

Let's look at it backwards:

(Leveled) FHE *without* Bootstrapping

Recall:

$$q = n^{O(d)}$$

Bigger $q \Rightarrow$ deeper circuits.

So how big can q get (securely)?

Best known attacks run in time $\sim 2^{\Omega(n/\log q)}$

$$\Rightarrow q = 2^{\sqrt{n}} \text{ is fairly safe}$$

$$\Rightarrow d = \widetilde{\Omega}(\sqrt{n})$$

Let's look at it backwards:

$\forall d$, set $n \approx d^2$ and obtain d-HE scheme with ciphertext length $n \log(nT) \approx d^2$.

(Leveled) FHE *without* Bootstrapping

Recall:

$$q = n^{O(d)}$$

Bigger $q \Rightarrow$ deeper circuits.

So how big can q get (securely)?

Best known attacks run in time $\sim 2^{\Omega(n/\log q)}$

$$\Rightarrow q = 2^{\sqrt{n}} \text{ is fairly safe}$$

$$\Rightarrow d = \widetilde{\Omega}(\sqrt{n})$$

Let's look at it backwards:

Post-processing can reduce ciphertext length.

$\forall d$, set $n \approx d^2$ and obtain d-HE scheme with ciphertext length $n \log(nT) \approx d^2$.

(Leveled) FHE *without* Bootstrapping

Recall:

$$q = n^{O(d)}$$

Bigger $q \Rightarrow$ deeper circuits.

So how big can q get (securely)?

Best known attacks run in time $\sim 2^{\Omega(n/\log q)}$

$$\Rightarrow q = 2^{\sqrt{n}} \text{ is fairly safe}$$

$$\Rightarrow d = \widetilde{\Omega}(\sqrt{n})$$

Let's look at it backwards:

Post-processing can reduce ciphertext length.

$\forall d$, set $n \approx d^2$ and obtain d-HE scheme with ciphertext length $n \log(nT) \approx d^2$.

No bootstrapping whatsoever!

Talk Outline

- The LWE assumption.
- Re-linearization and modulus reduction.
- FHE using bootstrapping.
- (Leveled) FHE without bootstrapping.
- PIR protocol.
- Conclusion and open problems.

Single-Server PIR [CGKS95,KO97,CMS99]

Index

$x \in [N]$

Database

DB

$|DB| = N \approx 2^n$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

evk

eval. key

Index

$x \in [N]$

Database

DB

$|DB| = N \approx 2^n$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

Index

$x \in [N]$

"Enc"(x)

evk

eval. key

Database

DB

$|DB|=N \approx 2^n$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

evk

eval. key

Index

$x \in [N]$

$\xrightarrow{\text{“Enc”}(x)}$

$\xleftarrow{y = \text{“Eval”}(\boldsymbol{DB}, \text{“Enc”}(x))}$

Database

DB

$|\boldsymbol{DB}| = N \approx 2^n$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

evk

eval. key

Index

$x \in [N]$

$\xrightarrow{\text{“Enc”}(x)}$

$\xleftarrow{y = \text{“Eval”}(\boldsymbol{DB}, \text{“Enc”}(x))}$

Database

DB

$|\boldsymbol{DB}| = N \approx 2^n$

$\mathrm{Dec}(y) = \boldsymbol{DB}[x]$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

evk

eval. key

Index

$x \in [N]$

$\xrightarrow{\text{“Enc”}(x)}$

$\xleftarrow{y = \text{“Eval”}(\boldsymbol{DB}, \text{“Enc”}(x))}$

Database

DB

$|\boldsymbol{DB}| = N \approx 2^n$

Correctness guarantee:

$\mathrm{Dec}(y) = \boldsymbol{DB}[x]$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

evk

eval. key

Index

$x \in [N]$

$\xrightarrow{\text{“Enc”}(x)}$

$\xleftarrow{y = \text{“Eval”}(\boldsymbol{DB}, \text{“Enc”}(x))}$

Database

DB

$|\boldsymbol{DB}| = N \approx 2^n$

Correctness guarantee:

$\mathrm{Dec}(y) = \boldsymbol{DB}[x]$

Privacy guarantee:

$\text{”Enc”}(x) \cong \text{“Enc”}(0)$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

evk

eval. key

Index

$x \in [N]$

$\xrightarrow{\text{“Enc”}(x)}$

$\xleftarrow{y = \text{“Eval”}(\boldsymbol{DB}, \text{“Enc”}(x))}$

Database

$\boldsymbol{DB}$

$|\boldsymbol{DB}|=N \approx 2^n$

Correctness guarantee:

$\mathrm{Dec}(y)=\boldsymbol{DB}[x]$

Privacy guarantee:

$\text{”Enc”}(x) \cong \text{“Enc”}(0)$

Communication complexity:

$cc=|\text{”Enc”}(x)|+|y|$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

evk
eval. key

Index
$x \in [N]$

$\xrightarrow{\text{“Enc”}(x)}$

$\xleftarrow{y = \text{“Eval”}(\boldsymbol{DB}, \text{“Enc”}(x))}$

Database
DB

$|\boldsymbol{DB}| = N \approx 2^n$

Correctness guarantee:
$\mathrm{Dec}(y) = \boldsymbol{DB}[x]$

Privacy guarantee:
$\text{”Enc”}(x) \cong \text{“Enc”}(0)$

Communication complexity:
$cc = |\text{”Enc”}(x)| + |y|$

FHE ⇒ PIR

Use our FHE naïvely:

$$cc = n \cdot \log(nT) \cdot \log(N) \approx \tilde{O}(\log^2 N)$$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

evk

Index

$x \in [N]$

$\xrightarrow{\text{“Enc”}(x)}$

$y = \text{“Eval”}(\boldsymbol{DB}, \text{“Enc”}(x))$

$\xleftarrow{}$

Database

DB

$|\boldsymbol{DB}| = N \approx 2^n$

Single-Server PIR [CGKS95,KO97,CMS99]

sk

evk

Index

$x \in [N]$

$\xrightarrow{\text{“Enc”}(x)}$

$\xleftarrow{y = \text{“Eval”}(\boldsymbol{DB}, \text{“Enc”}(x))}$

Database

DB

$|\boldsymbol{DB}| = N \approx 2^n$

Reducing comm. complexity:

- Enc(x) using different, more efficient, scheme.
- Hom. decrypt efficient ciphertext and use as before.
- Using known efficient schemes: $cc = \tilde{O}(\log N)$.

Single-Server PIR [CGKS95,KO97,CMS99]

sk, *sym*

Enc(*sym*), *evk*

Index

$x \in [N]$

$\mathrm{Enc}_{sym}(x)$ →

← $y = \text{“Eval”}(\boldsymbol{DB}, \text{“Enc”}(x))$

Database

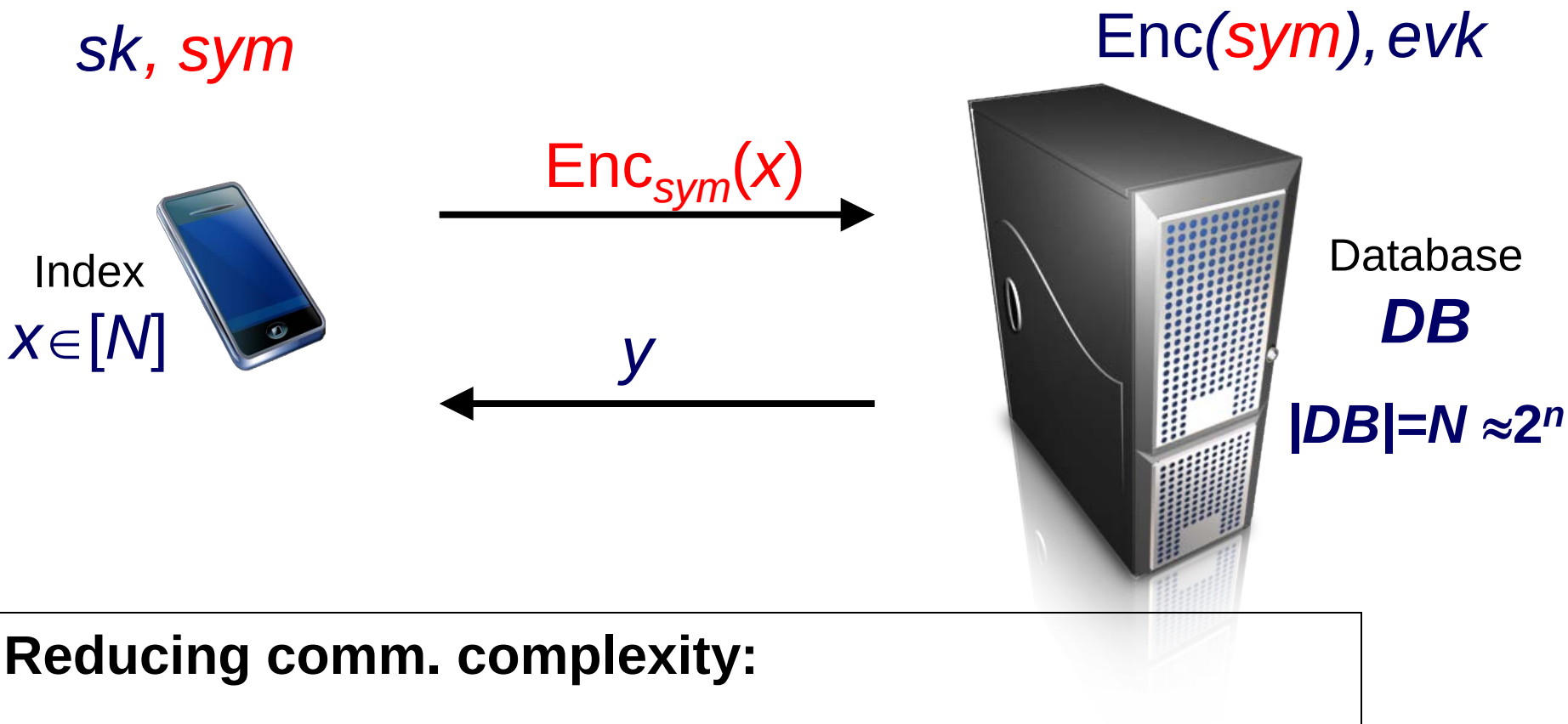
DB

$|DB| = N \approx 2^n$

Reducing comm. complexity:

- Enc(x) using different, more efficient, scheme.
- Hom. decrypt efficient ciphertext and use as before.
- Using known efficient schemes: $cc = \tilde{O}(\log N)$.

Single-Server PIR [CGKS95,KO97,CMS99]



Reducing comm. complexity:

- Enc(x) using different, more efficient, scheme.
- Hom. decrypt efficient ciphertext and use as before.
- Using known efficient schemes: $cc = \tilde{O}(\log N)$.

Single-Server PIR [CGKS95,KO97,CMS99]

sk, sym

Enc(sym), evk

Index

$x \in [N]$

$\mathrm{Enc}_{sym}(x)$ →

← y

Database

DB

$|DB|=N \approx 2^n$

$\mathrm{Enc}_{sym}(x)+\mathrm{Enc}(sym) \Rightarrow \mathrm{Enc}(x)$

$y = \mathrm{Eval}(DB, \mathrm{Enc}(x))$

Reducing comm. complexity:

- Enc(x) using different, more efficient, scheme.
- Hom. decrypt efficient ciphertext and use as before.
- Using known efficient schemes: $cc = \tilde{O}(\log N)$.

Conclusion

- **FHE is not so complicated anymore…**
  - No ideals.
  - No squashing or sparse subset sum assumption.
  - **Efficiency!** Short ciphertexts, keys are easy to generate and are fairly unstructured.
  - **Can't lose:** Don't care about homomorpism? Use our scheme anyway – same efficiency and security.

- Private Information Retrieval.
  - Don't pay more than $\tilde{O}(\log(N))$.

Open Problems

- Remove circular security assumption.
- Go from quasi-polynomial to strictly polynomial LWE modulus.
- Improve efficiency, possibly using ideals [BGV11, LNV11, GHS11].
- New notions of security [BSW11].

Questions?