Fully Homomorphic Encryption from the ground up

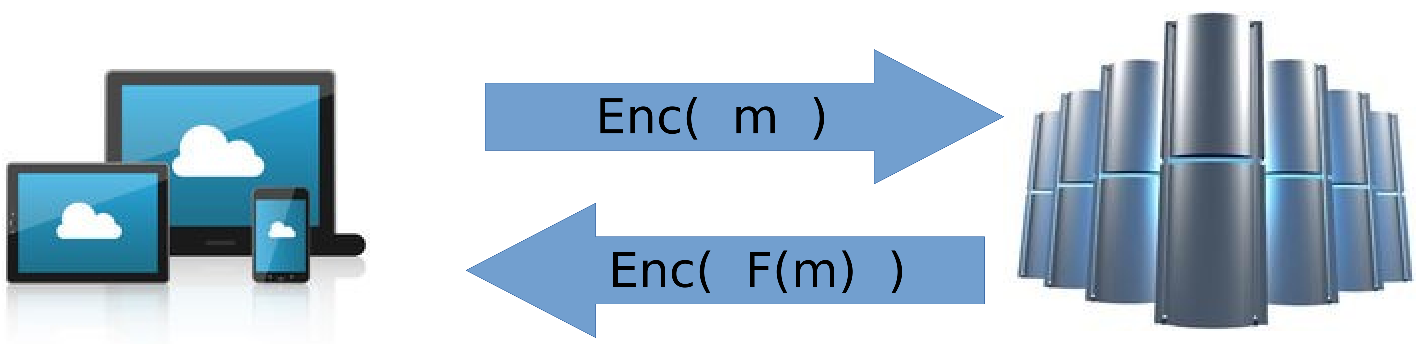
Daniele Micciancio
(UC San Diego)

Eurocrypt 2019

(Fully Homomorphic) Encryption

- Encryption: used to protect data at rest or in transit

- Fully Homomorphic Encryption: supports arbitrary computations on encrypted data



FHE Timeline

- Concept originally proposed by
  Rivest, Adleman, Dertouzos (1978)
- Gentry's breakthrough (2009)
  - First candidate solution
  - Bootstrapping technique
- Much subsequent work (2010-2019 …)
  - Basing security on standard (lattice) assumptions
    [BV11,B12,AP13,GSW13,BV14,…]
  - Efficiency improvements
    [GHS12,BGH13,AP13/14,DM15,CP16,CGGI16/17,CKKS17,MS18,…]
  - Implementations:
    HElib, SEAL, PALISADE, FHEW, TFHE, HeaAn, Λολ, NFLlib, …

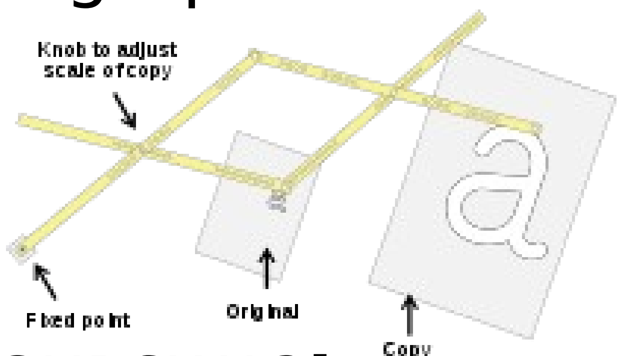
Outline

- FHE: background and sample applications
- Lattice Cryptography
  - Key properties of lattice cryptography that make it so useful to build FHE and other applications
- Generic FHE construction
  - Symmetric Encryption
  - Public Key Encryption
  - Linearly Homomorphic Encryption
  - Fully Homomorphic Encryption

FHE applications

- Direct applications:
  - Secure outsourcing of computation
- Powerful tool: “Cryptographic Pantograph”

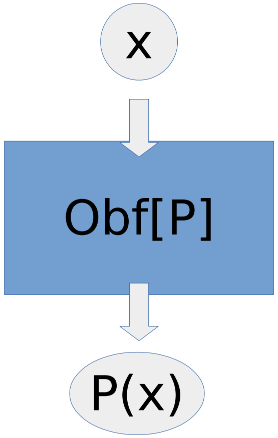
FHE for NC1 → FHE for PTIME

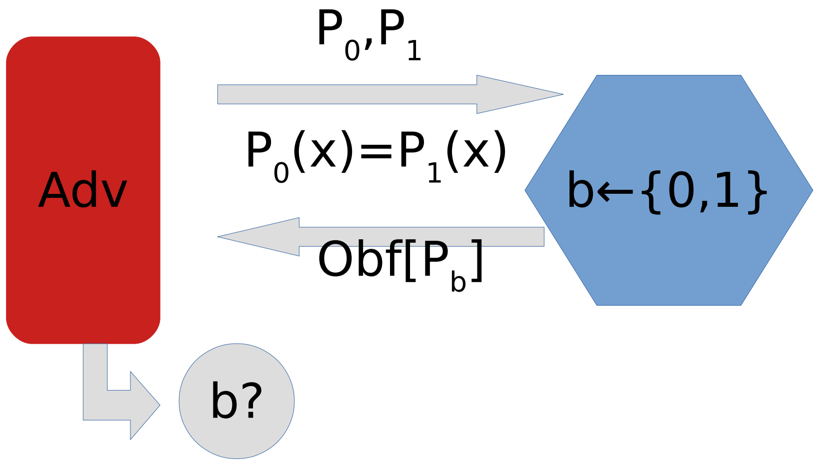


  - FHE [Gentry09]
  - (Indistinguishability) Obfuscation [GGHRSW13]
  - Functional Encryption [GKPVZ13]
  - Correlation Intractable Hash Functions [PS19], [CCHLRRW19]
  - ….

Sample Application 1

- ***(Indistinguishability) Obfuscation***
  - Obf: Program → Program
  - Correctness: Obf[P] (x) = P(x)
  - Security: $P_0(x)=P_1(x) \rightarrow Obf[P_0] \sim Obf[P_1]$

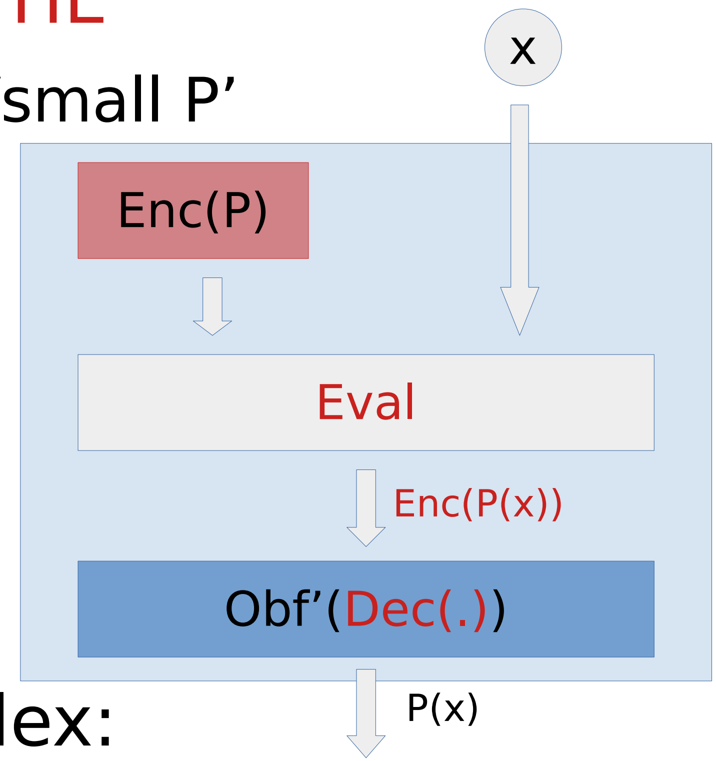




Bootstrap Obfuscation

- Bootstrapping Obfuscation using FHE
  - Obf': obfuscation scheme for simple/small P'
- Obf[P] = (Enc(P),Obf'[Dec(.)])
  - (Enc,Dec,Eval)←FHE.KeyGen
- Obf[P](x) = Dec(e)
  - Obf'[Dec(.)] (Eval(Enc(P),x))

    = Dec(Enc(P(x))) = P(x)
- Actual scheme is a bit more complex:
  - encrypt/evaluate P twice, under two different FHE keys
  - check consistency before decryption



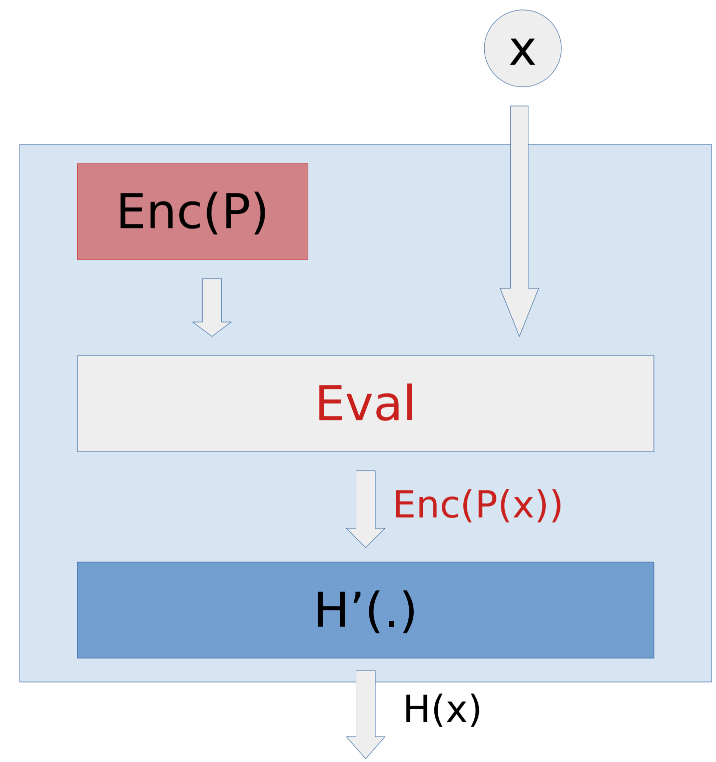
Sample Application 2

- ***Correlation Intractable Hash Functions***
  - Hash function $H(x)$, Relation $R=\{(x,f(x)) : x\}$
  - Security: Hard to find x such that $R(x,H(x))$
- H="Random oracle" is "trivially" secure
- Applications:
  - Fiat-Shamir Signatures in the Standard Model
  - Remove interaction in public coin protocols
  - Non-Interactive Zero-Knowledge

Bootstrapping Correlation Intractability

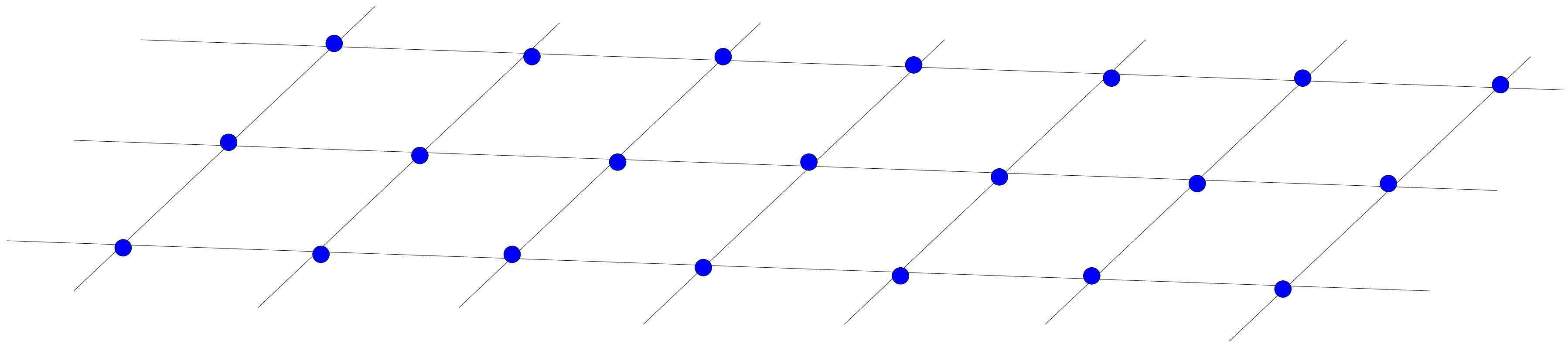
- H’: CI Hash function for simple relation

  R(x,y) = “y=Dec(x)”, for some Dec←FHE.KeyGen
- H: CI Hash function for arbitrary P
  - (Enc,Dec,Eval)←FHE.KeyGen
  - C=Enc(P)
  - H(x)=H’(Eval(C,x))
- Security:
  - Assume H(x)=P(x)
  - Let c=Eval(C,x)=Enc(P(x))
  - Then H’(c)=H(x)=P(x)=Dec(c)



Lattice cryptography

- Lattices: regular sets of vectors in n-dim space



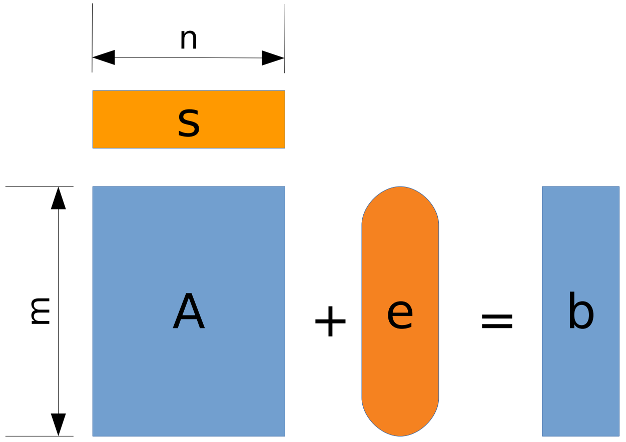
- Many attractive features:
  - Post-Quantum secure candidate
  - Simple, fast and easy to parallelize
  - Versatile (FHE and much more)

$$\begin{bmatrix}4\\1\\6\\2\\3\end{bmatrix} + \begin{bmatrix}8\\1\\7\\3\\3\end{bmatrix} = \begin{bmatrix}12\\2\\13\\5\\6\end{bmatrix}$$

Why Lattice Cryptography?

- Lattices → Encryption
  - weak linear homomorphic properties
  - simple (linear) decryption algorithm
  - circular secure: $Enc_s(s)$ does not leak s
- This is enough to obtain
  - multiplication by arbitrary constants
  - multiplications between ciphertexts
  - fully homomorphic encryption

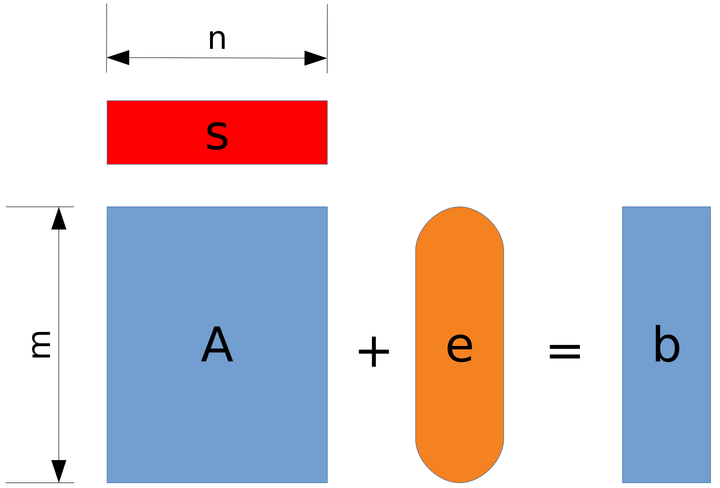
Learning With Errors (LWE)



- LWE function family:
  - Key: $A \in Z_q[n\mathrm{x}m]$
  - $\mathrm{LWE}_A(s,e) = As + e \pmod q$
  - Small $|e|_{max} < \beta = O(\sqrt{n})$
  - q,m=poly(n)
  - Injective version of Ajtai's SIS function
- Regev (2005): assuming quantum hard lattice problems
  - LWE_A is one-way: Hard to recover (s,e) from [A,b]
  - $b=\mathrm{LWE}_A(s,e)$ is indistinguishable from uniform over $Z_q[m]$
  - [BLPRS13] hard under classical reductions

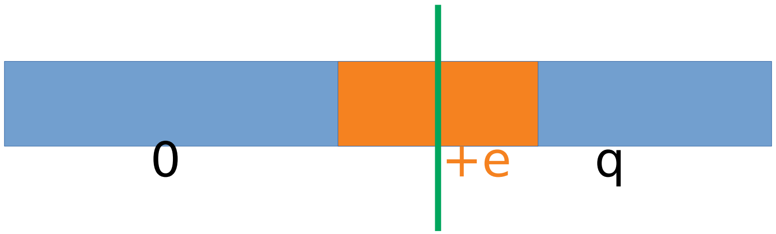
Encrypting with LWE

- Idea: Use $b = \mathrm{LWE}_A(s, e)$ as a one-time pad
- Private key encryption scheme:
  - secret key: $s \in Z_q^n$,
  - message: $m \in Z^m$
  - encryption randomness: $[A, e]$
  - $E_s(m; [A, e]) = [A, b+m]$
- [BFKL93],[GRS08]
  - Learning Parity with Noise (LPN): q=2
  - If LWE_A is one-way, then $b = As + e$ is pseudo-random
- Regev LWE: $q \to \mathrm{poly}(n)$

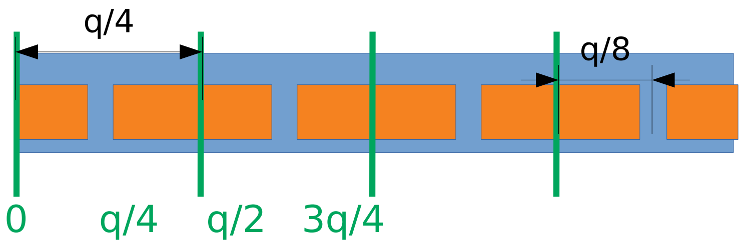


Noisy Decryption

- $E_s(m;[A,e]) = [A,b+m]$ where $b = As+e$
- Decryption:
  - $D_s([A,b+m]) = (b+m) - As = m+e \mod q$



  - Low order bits of m are corrupted by e
- Fix: scale m, and round:



Weak Linear Homomorphism

- $[A_1, A_1 s + e_1 + m_1] + [A_2, A_2 s + e_2 + m_2]$

  $= [(A_1 + A_2), (A_1 + A_2)s + (e_1 + e_2) + (m_1 + m_2)]$

$E_s(m;\beta)$: encryption of m with error $|e| < \beta$

- $E_s(m_1;\beta_1) + E_s(m_2;\beta_2) \subset E_s(m_1 + m_2;\beta_1 + \beta_2)$

Circular Security

- $E_s(m; [A,e]) = [A,b+m]$, where $b=As+e$
- $D_s([A,b+m]) = (b+m) - As = m+e$
- $D_s([-A,0]) = 0+As = As$
- Easy to compute encryptions of (linear functions of) the secret key s!
- Random encryptions:

$$[-A,0]+E_s(0;\beta)=E_s(As;\beta)$$

Decryption is also linear

- $D_s(A,b) = b - As = m+e$
- Linear in the ciphertext (A,b)
- Linear *in the secret key* $s'=(-s,1)$
  - $D_{s'}(A,b) = [A,b]s'=m+e$
  - $D_{cs'}(A,b) = [A,b](cs')=cm+ce$
- Remark:
  - Only approx. decryption is linear
  - Exact decryption involves non-linear rounding

Operations on Ciphertexts

- Add: $E(m_1;\beta_1)+E(m_2;\beta_2) \subset E(m_1+m_2;\beta_1+\beta_2)$
- Neg: $-E(m;\beta) = E(-m;\beta)$
- Mul: $c*E(m;\beta) = E(c*m;\ c*\beta)$
- Const: $[O,m] \in E(m;0)$
- Key: $[-A,0] \in E(As;0)$

Weak linear homomorphic properties:

- can perform a limited number of additions and multiplications by small constants
- decryption is linear in the secret key $s'=(-s,1)$
- circular security: $E(As)$ does not leak s

Public Key Encryption

- Public Key:

  $[a_1,b_1] = E_s(0), \ldots, [a_n,b_n]=E_s(0)$

- Encrypt(m): $(\Sigma_i\ r_i * [a_i,b_i]) + (0,m)$
  - $E_s(0)+\ldots+E_s(0)+E_s(m;0) = E_s(m)$
- Decrypt normally using secret key
- [Regev05] LWE Public Key Encryption
- [Rothblum11]: any weakly linear homomorphic encryption implies public key encryption

Multiplication by any constant

- $E'[m] = (E[m], E[2m], E[4m], \ldots, E[2^{\log(q)} m])$
- Multiplication by $c \in Z_q$:
  - Write $c = \Sigma_i\, c_i\, 2^i$, where $c_i \in \{0,1\}$
  - Compute $\Sigma_i\, c_i\, E[2^i\, m] = E[\Sigma_i\, c_i\, 2^i\, m] = E[cm]$
- $cE'[m] = E[cm]$
- We can also compute $E'[cm]$:

$$c * E'[m] = (cE'[m], (2c)E'[m], \ldots, (2^{\log q} c)E'[m])$$

$$= (E[cm], E[(2c)m], \ldots, E[(2^{\log q} c)m]) = E'[cm]$$

Multiplication via Homomorphic Decryption

- Idea:
  - Encryption E(m) = (a,as+e+m) is linearly homomorphic
  - Decryption D(a,b) = b – as = m+e is linear in s'=(-s,1)
  - We can decrypt homomorphically using an encryption of s'
- Details
  - Given: E(m)=(a,b) and E'(s')=(E'(-s),E'(1))
  - Compute E(m)*E'(s') = a*E'(-s)+b*E'(1)=E(m)
- More interesting:
  - Given E(m) and E'(cs')
  - Compute E(m)*E'(cs') = E(cm)

Homomorphic "decrypt and multiply"

- $E''(c) = E'(cs') = E'(\text{"}E(m) \rightarrow c*m\text{"})$
- $E''(c) = \{E(\alpha_i c)\}_i$ for some $\alpha_i(s)$
- Homomorphic Properties:
  - $E''(m_1) + E''(m_2) = E''(m_1+m_2)$
  - $E''(m_1)*E''(m_2)$

    $=\{E(\alpha_i m_1)*E''(m_2)\}_i$

    $=\{E(\alpha_i m_1*m_2)\}$

    $= E''(m_1*m_2)$

FHE

- E'' encryption scheme supporting
  - $E''(m_0)+E''(m_1) = E''(m_0+m_1)$
  - $E''(m_0)*E''(m_1) = E''(m_0*m_1+e)$
- Not quite a FHE yet:
  - E'' can evaluate any arithmetic circuit
  - But noise grows with computation
- Effectively:
  - can only evaluate small circuits / branching programs
- Bootstrapping: FHE(NC1) → FHE(PTIME)

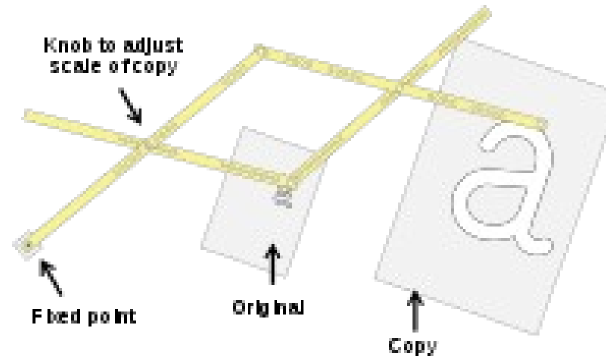
most significant bit (msb)
$x = (q/2)m + e \pmod q$
$|e|<q/4,\ m\in\{0,1\}$
$\mathrm{msb}(x+q/4) = b$

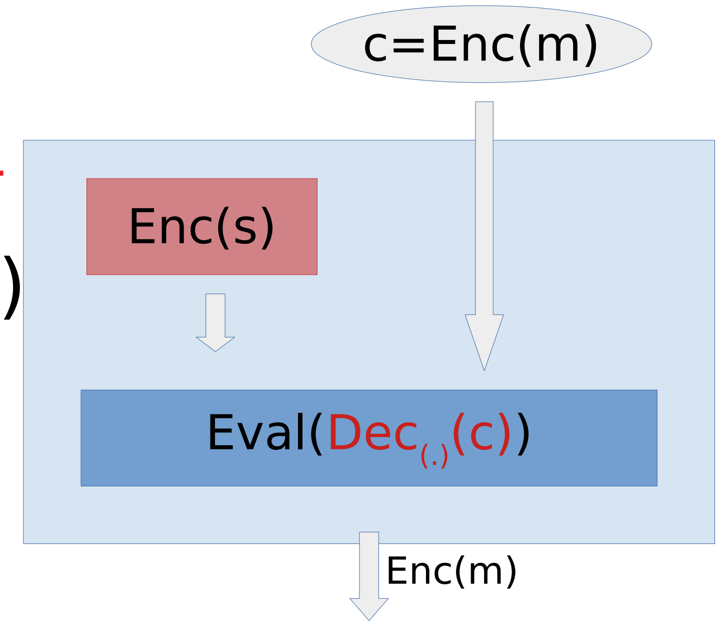
Bootstrapping FHE

- Let $c = \mathrm{Enc}_s(m*(q/2)+e)$
- $f_c(s)=\mathrm{msb}(\mathrm{Dec}_s(c))*(q/2)=m*(q/2)$
- Eval f_c homomorphically on

  $\{s\}=\mathrm{Enc}_s(s)$
- $f_c(\{s\})= \{f_c(s)\} = \{\mathrm{msb}(\mathrm{Dec}_s(c))\}$

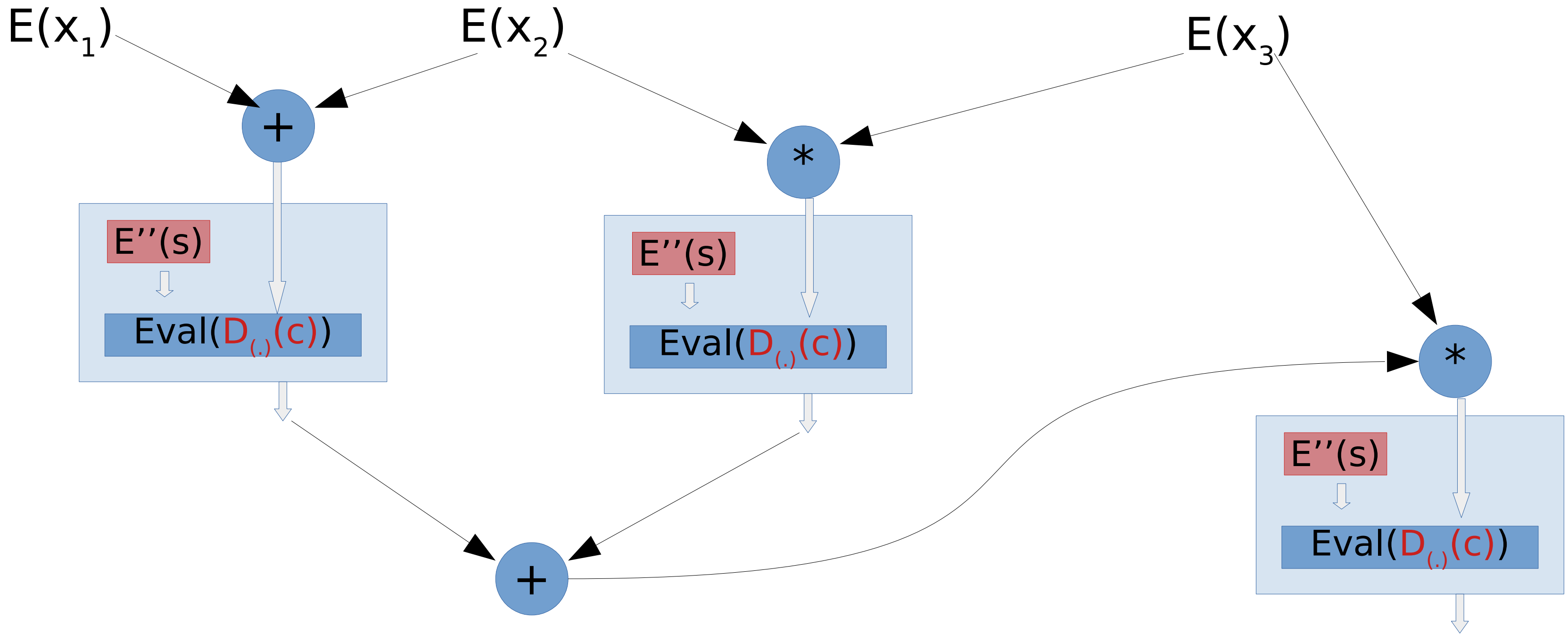
  $= \{m*(q/2)\} = \mathrm{Enc}_s(m*(q/2))$
- Output noise depends on $\mathrm{msb}\circ\mathrm{Dec}_{\{s\}}$, but not on e





Composing FHE computations

- Output noise depends on $\mathrm{Dec}_{[s]}$, but not c.
- $\mathrm{Enc}(m*(q/2); q/4) \rightarrow \mathrm{Enc}(m*(q/2); \beta \ll q/4)$
- Can compose arbitrarily many gates, while keeping noise small

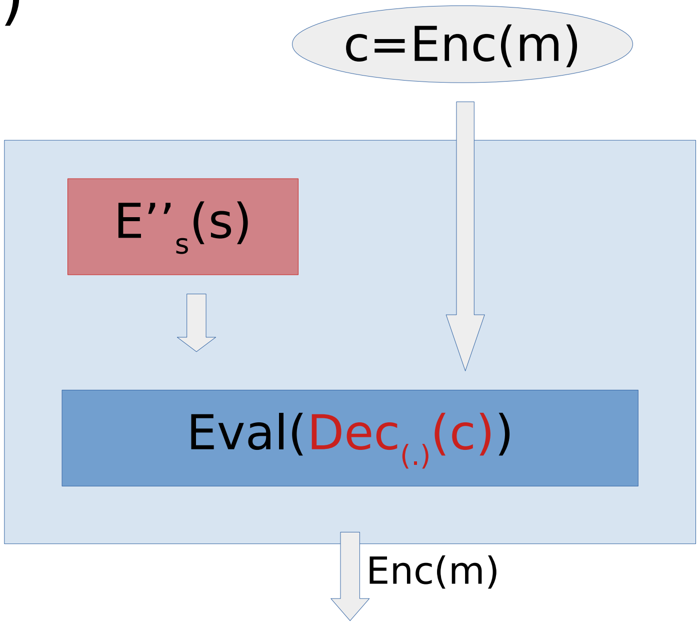


Requirements

- Correctness:
  - Need “exact” decryption Dec(Enc(m))=m
  - Achieved by scaling and rounding

    round((q/2)m+e) = msb((q/2)m + e)
- Circular security:
  - Need to encrypt s under E''_s
  - Circular security of $E''_s(s)$

    still an open problem
  - Not needed for Leveled FHE



Summary

- Lattice (LWE) encryption E
  - Circular secure: $E_s(s)$
  - Linear approx. decryption D(s)
  - Transform E → E’’ (provably secure encryption)

    E’’ can evaluate arbitrary (low depth) function
- Bootstrapping
  - Nonlinear (but still low depth) rounding function
  - Can be computed by E’’
  - Open problem: circular security of $E''_s(s)$

Homomorphic Decrypt&Round

- Dec(s[i],[a[i],b]) = round(b-$\sum_i$a[i]s[i])

  = msb((q/4)+b-$\sum_i$a[i]s[i])
- Assume for simplicity s[i] ∈ {0,1}
- Write all numbers in binary:
  - b+(q/8)=$\sum_j 2^j b_j$, -a[i]=$\sum_j 2^j a_j$[i], where b_j,a_j[i]∈{0,1}
- Want to compute and round
  - R = $\sum_j 2^j(b_j+\sum_i a_j[i]s[i])$
  - Output most significant bit msb(R) = (2R/q) mod 2

Homomorphic Decryption

- Dec(s[i],[a[i],b]) = msb(∑...s[i])

	b_k	...	...	b_3	b_2	b_1	b_0	
+	$a_k[1]$	...	...	$a_3[1]$	$a_2[1]$	$a_1[1]$	$a_0[1]$	* s[1]
+	$a_k[2]$	...	...	$a_3[2]$	$a_2[2]$	$a_1[2]$	$a_0[2]$	* s[2]
...	...	...	...	...	...	...	...	...
+	$a_k[n]$	...	...	$a_3[n]$	$a_2[n]$	$a_1[n]$	$a_0[n]$	* s[n]

  - Homomorphic in s:
  - Enc(s[1]),….,Enc(s[n]) → msb(∑)

Homomorphic Decryption

- Dec(s[i],[a[i],b]) = msb($\sum$...s[i])

	1	...	...	1	0	1	0	
+	0	...	...	1	1	0	1	* s[1]
+	1	...	...	1	0	1	0	* s[2]
...	...	...	...	...	...	...	...	...
+	1	...	...	0	0	1	1	* s[n]

  - Homomorphic in s:
  - Enc(s[1]),....,Enc(s[n]) → msb($\sum$)

Homomorphic Decryption

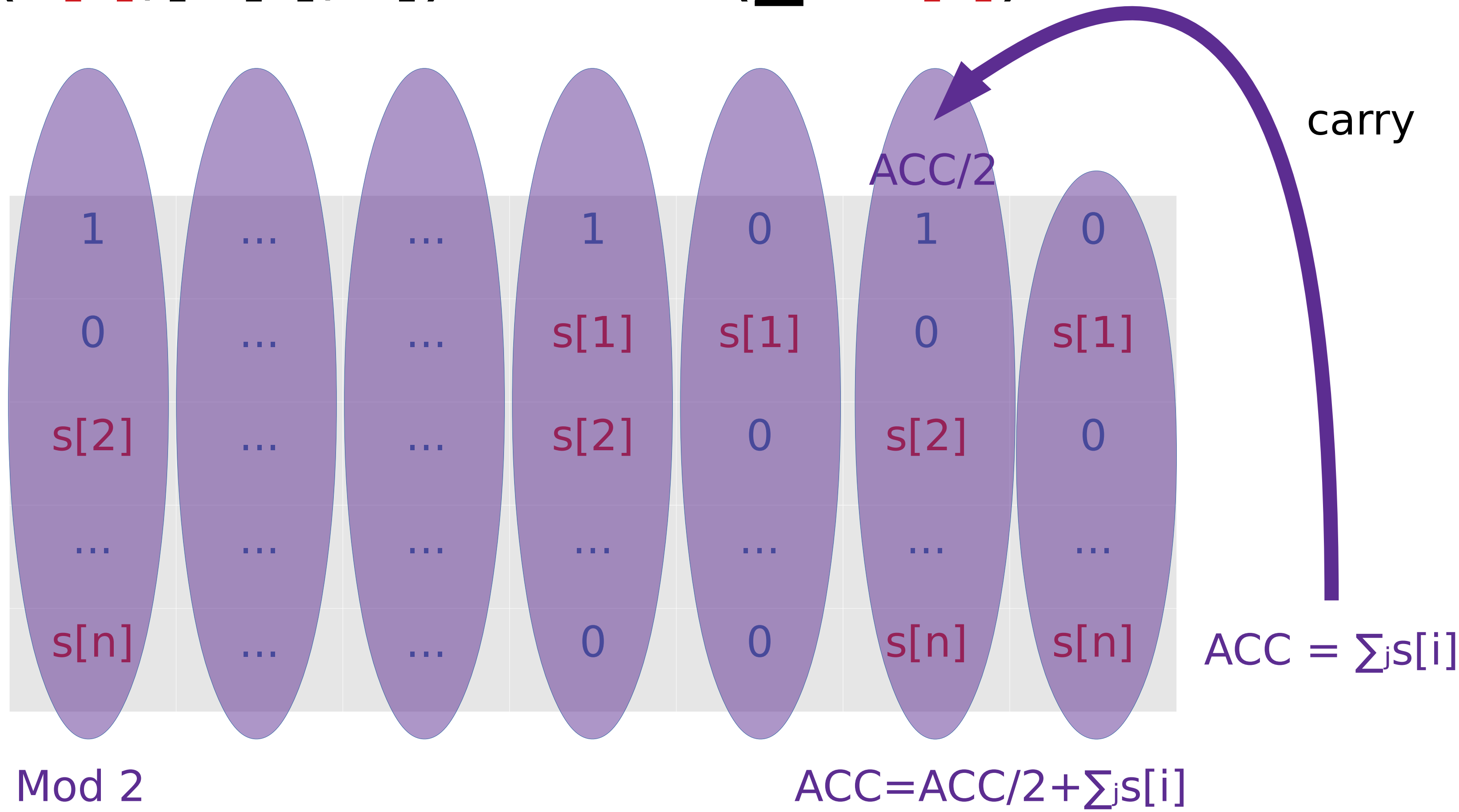
- Dec(s[i],[a[i],b]) = msb(∑...s[i])

	1	...	...	1	0	1	0
+	0	...	...	s[1]	s[1]	0	s[1]
+	s[2]	...	...	s[2]	0	s[2]	0
...	...	...	...	...	...	...	...
+	s[n]	...	...	0	0	s[n]	s[n]

  - Homomorphic in s:
  - Enc(s[1]),....,Enc(s[n]) → msb(∑)

Homomorphic Decryption

- Dec(s[i],[a[i],b]) = msb(∑...s[i])

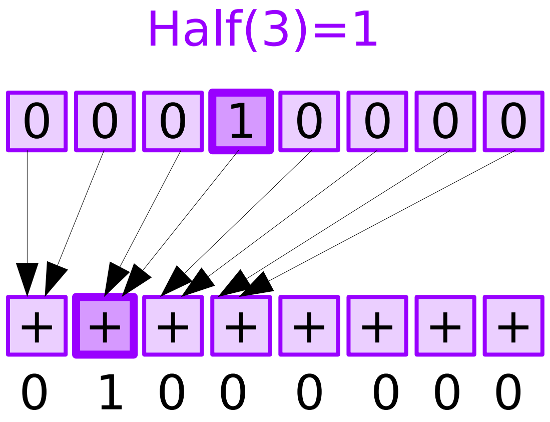


Cryptographic Accumulator

- ACC[v] holds values $v \in \{0,\ldots,N=2(n+1)\}$
- Local Operations:
  - Increment: ACC[v] → ACC[v+1]
  - Half: ACC[v] → ACC[v/2]
  - Mod2: ACC[v] → ACC[v mod 2]
- Accum: ACC[v] , E’’[$s \in \{0,1\}$] → ACC[v+s]
- Extract: ACC[v] → Enc(v=1)

ACC local operations

- [AP14] ACC[v;β]=(c[0],…,c[N]) where
  - $c[v]=\mathrm{Enc}(1;\beta_v)$, $c[u]=\mathrm{Enc}(0;\beta_u)$
  - $\beta = \sum_i \beta_i$
- f(ACC[v;β]) = ACC[f(v);β]
  - $c'[v] = \sum \{ c[u] \mid f(u)=v \}$
  - $\sum \{\} = \mathrm{Enc}(0;0) = [0,0]$
- Increment, Half, Mod2: choose appropriate f
- Extract(ACC) = c[1]



Accumulate

- Accumulate: $ACC[v] + E''[s] \rightarrow ACC[v+s]$
- Compute $A_0 = ACC[v]$, $A_1 = ACC[v+1]$
- Select:
  - $ACC[v+s] = A_s = A_0 * (1-E''[s]) + A_1 * E''[s]$
- All operations supported by our E’’

Credits

- Most techniques used in this construction proposed independently in other works
  - Linearity of lattice cryptography [BM97],[LMPR08]
  - Multiplication gadget matrix (1,2,4,…) [Ajtai99],[BV]++
  - Approximate decryption [CKKS17] HEAAN
  - E’’ : essentially equivalent to [GSW13]
  - Accumulators [AP14]. See also [DM15],[CGGI16].
- Only new technique: bootstrapping via schoolbook addition algorithm

Concluding remarks

- Simple HE from basic building blocks
  - Regev LWE: mod-q variant of [BFKL93],[GRS08]
  - "CryptoComputing for NC1" [SYY99]
- FHE = Simple HE + Bootstrapping [G09]
  - Main efficiency bottleneck in practice
  - Main theoretical open problem: circular security
- Other applications? Yes!
  - Translate between FHE and MPC [CLOPS13],[BGG18],
  - Homomorphic Commitments [GSW13],[PS19]
  - Homomorphic Secret Sharing [BKS19]
  - Symmetric Crypto with algebraic structure [AMPR19]

References

- [BFKL93] Blum,Furst,Kearns,Lipton
- [BM97] Bellare, Micciancio
- [SYY99] Sander,Young,Yung
- [LMPR08] Lyubashevsky,Micciancio,Peikert,Rosen
- [GRS08] Gilbert,Robshaw,Seurin
- [G09] Gentry
- [BV11,14] Brakerski, Vaikuntanathan
- [CLOPS13] Choudhury,Loftus,Orsini,Patra,Smart
- [GGHRSW13] Garg,Gentry,Halevi,Raykova,Sahai,Waters
- [GKPVZ13] Goldwasser,Kalai,Popa,Vaikuntanathan,Zeldovich
- [GSW13] Gentry,Sahai,Waters
- [BLPRS13] Brakerski,Langlois,Peikert,Regev,Stehle
- [AP14] Alperin-Sherif, Peikert
- [CGGI16/17] Chilotti,Gama, Georgieva, Izabachene
- [CKKS17] Cheon,Kim,Kim,Song
- [BGG18] Boura,Gama,Georgieva
- [CCHLRRW19] Canetti, Chen, Holmgren,Lombardi,Rothblum,Rothblum,Wichs
- [PS19] Peikert, Shiehian
- [BKS19] Boyle, Kohl, Scholl
- [AMPR19] Alamati,Montgomery,Patranabis,Roy

Thank You!

Questions?

Thank You!

Questions?