Fully Homomorphic Encryption

A Simple Construction & Open Problems

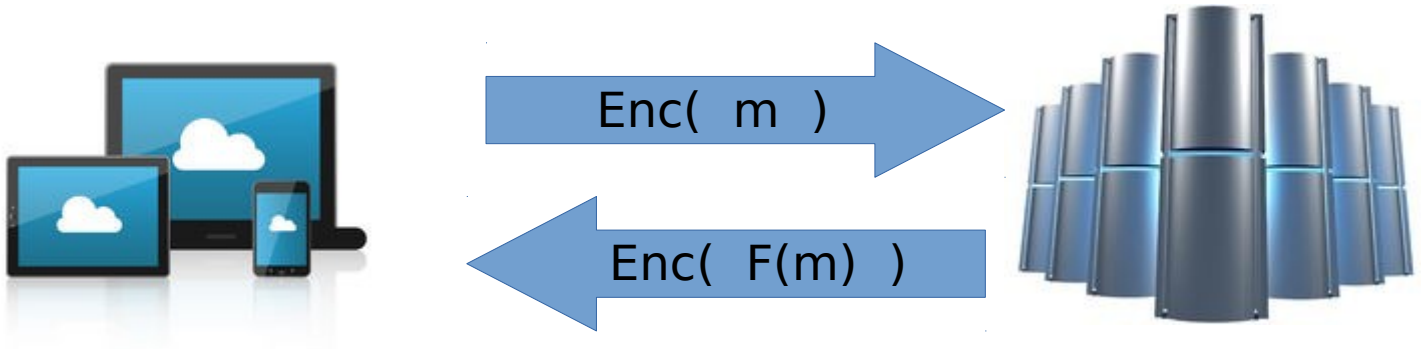
Daniele Micciancio
(UC San Diego)

IMACC 2017

Fully Homomorphic Encryption (FHE)

- 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 (2010-2017 ...)
  - Basing security on standard (lattice) assumptions

    [BV'11,B'12,AP13,GSW'13,BV'14,...]
  - Efficiency improvements and Implementations

    [GHS'12,BGH'13,AP'13,AP'14,DM'15,CP'16,CGGI'16,...]

This Talk

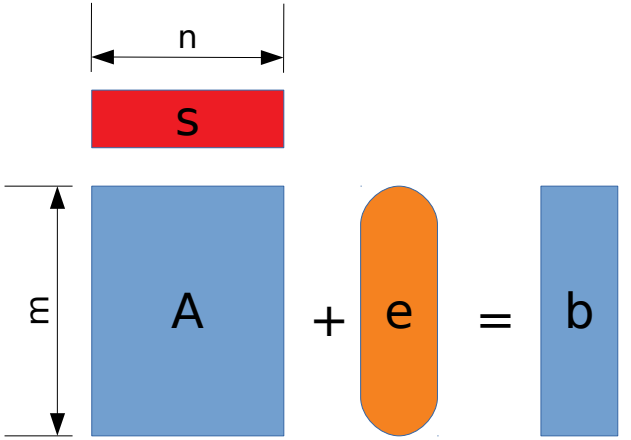
- Simple, self contained description of FHE
  - Focus: simplicity, elementary construction
  - Goal: theoretical understanding of FHE
  - Aside: efficiency, implementation
- Security of Encryption cycles
  - Main theoretical problem about FHE still open
  - Relates to many other problems in cryptography

Learning With Errors (LWE)

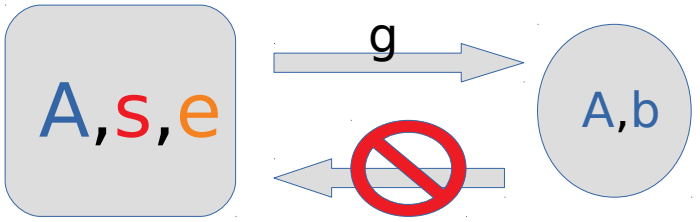
- LWE Problem (Regev, 2005)
  - Given: A and b = As+e

    where $(A,s,e) \in (Z_q^{m \times n}, Z_q^{n}, [\pm\beta]^m)$

    $n=2^k$, $\beta \approx \sqrt{n} \ll q = n^4$, $m=\text{poly}(n)$
  - Goal: find s



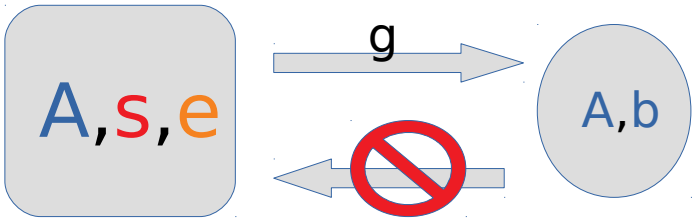
- Regev One-Way Function:
  - g(A,s,e) = [A,b=As+e]
  - Invert g: find s, A, and e = b-As

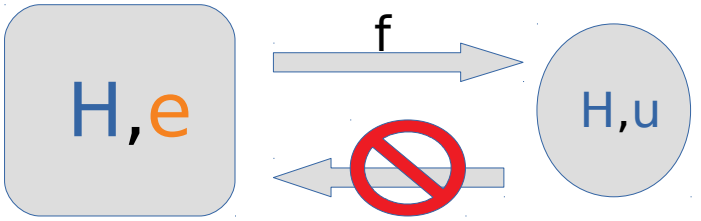


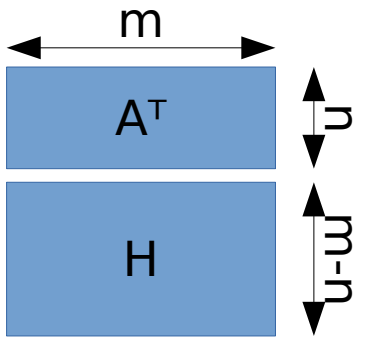
- Equivalent: Decoding random linear code A from error e

Duality and Ajtai's function

- [Regev'05] g(A,s,e) = [A,b=As+e]
- [Ajtai'96] f(H,e)=(H,He)
- Inverting f:
  - Given (H,u), find small e such that He=u
  - Short Integer Solution (SIS) Problem
  - Syndrome decoding problem
- Problems are equivalent when
  - $H \in Z_q^{(m-n)xm}$, HA=0 mod q
  - Input e follows the same distribution







LWE vs SIS

- LWE: g(A,s,e)=[A,As+e], SIS: f(H,e)=[H,He], but
  - LWE: very small e → g(A,s,e) injective
  - SIS: larger e, f(H,e) ≈ Uniform($Z_q^{(m-n)x(m+1)}$)
- Theorem: [Regev'05,...]

  If g(A,s,e)=(A,As+e) is a one-way function

  then [A,As+e] ≈$_c$ Uniform($Z_q^{mx(n+1)}$)

Encrypting with LWE

- Idea: Use [A,b=As+e] as a one-time pad
- Private key encryption scheme:
  - secret key: $s \in Z_q^n$, "message" $m \in Z$
  - encryption randomness: [A,e]
  - Enc(s, m; [A,e]) = [A,As+e] + Gm
- Notes:
  - "message" $m \in Z_q$
  - G: (public) encoding matrix, to be chosen
  - Gm = Enc(s,m;[O,0]) is an encryption of m
- Blum, Furst, Kearns & Lipton [Crypto 1993] (for q=2)

Linear Homomophism

- $\mathrm{Enc}(s, m_1; A_1, e_1) + \mathrm{Enc}(s, m_2; A_2, e_2)$

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

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

  $= \mathrm{Enc}(s, (m_1 + m_2); (A_1 + A_2), (e_1 + e_2))$

- $\mathrm{Enc}(m; \beta)$:
  encryption of m with error $||e||_1 = \sum_i |e_i| < \beta$

- $\mathrm{Enc}(m_1; \beta_1) + \mathrm{Enc}(m_2; \beta_2) = \mathrm{Enc}(m_1 + m_2; \beta_1 + \beta_2)$

Operations on Ciphertexts

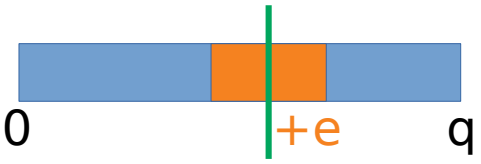
- All operations are publicly computable
- Add: $\mathrm{Enc}(m_1;\beta_1)+\mathrm{Enc}(m_2;\beta_2)$
  $= \mathrm{Enc}(m_1+m_2;\beta_1+\beta_2)$
- Mul: $c*\mathrm{Enc}(m;\beta) = \mathrm{Enc}(c*m;\ c*\beta)$
- One: $G \in \mathrm{Enc}(1;0)$
- Zero: $O \in \mathrm{Enc}(0;0)$
- Neg: $-\mathrm{Enc}(m;\beta) = \mathrm{Enc}(-m;\beta)$
- Not: $G - \mathrm{Enc}(m;\beta) = \mathrm{Enc}(1-m;\beta)$

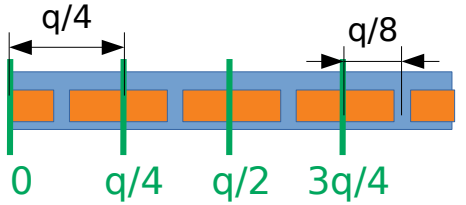
Noisy Decryption: Regev LWE

- G=[0,…,0,1]
- Enc(s,m;[A,e])=[A,As+e]+Gm = [A,b]

  where b = As+e+m
- Decryption:
  - Dec'(s,[A,b]) = b-As = m+e mod q
  - Low order bits of m are corrupted by e
  - Encode msg ∈ Z_4 in ***high order bits***: m = (q/4)msg
  - Round Dec'(s,[A,b])=(q/4)msg+e to closest multiple of (q/4)
- Correctness: if |e|<β<q/8:

  Dec(s,[A,b]) = round(Dec'(s,[A,b])) = msg





Error Growth

- If $q>8\beta$, we can decrypt $\mathrm{Enc}((q/4)msg;\beta)$
- If $q>16\beta$, we can decrypt

  $\mathrm{Enc}((q/4)msg_1,q/16) + \mathrm{Enc}((q/4)msg_2,q/16)$

  $= \mathrm{Enc}((q/4)(msg_1+msg_2),q/8)$
- If $q>8k\beta$, we can decrypt

  $\mathrm{Enc}((q/4)msg_1,\beta)+\ldots+\mathrm{Enc}((q/4)msg_k,\beta)$

  $= \mathrm{Enc}((q/4)(msg_1+\ldots+msg_k),q/8)$

HE is not an FHE!

- LWE allows to compute linear functions on ciphertexts
  - We want to compute non linear functions too!
- Even within the limited setting of linear functions:
  - we can only perform a bounded number of additions, before the errors get too large

Gentry Bootstrapping

- [Gentry, FOCS 2009]
  - Refresh: Enc(s,m;q/8) → Enc(z,m;q/16)
- Consider the function
  - $f_c(s)=Dec(s,c)$
- Compute f_c homomorphically on [s]=Enc(z,s)
  - c = Enc(s,m;β<q/8), [s] = Enc(z,s)
  - $f_c([s])= [f_c(s)] = [Dec(s,c)] = [m] =Enc(z,m)$
- [m]=Enc(z,m;β') where β' depends only on f_c.
- Set z=s and β'=q/16:

  Enc(m_1;q/16)+Enc(m_2;q/16) = Enc(m_1+m_2;q/8) → Enc(m_1+m_2;q/16)

FHEW

m_1	m_2	m_1+m_2	sum/2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

- Idea: [Ducas, Micciancio, 2015]
  - Use arithmetics modulo 4
  - Bootstrapping + Compute:

    Enc(s,m;q/8) → Enc(s,floor(m/2);q/16)
- Enough to compute arbitrary circuits:
  - $m_1, m_2 \in \{0,1\} \subset Z_4 = \{0,1,2,3\}$
  - AND(m_1,m_2) = floor((m_1+m_2)/2)
  - NOT(m) = 1-m
- Cannot do this working directly mod 2
  - All unary gates mod 2 (0,1,id,not) are linear!

Homomorphic Decryption

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

  $= \mathrm{msb}((q/8)+b-\sum_i a[i]s[i])$

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

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(∑...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(∑)

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])

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 Decryption

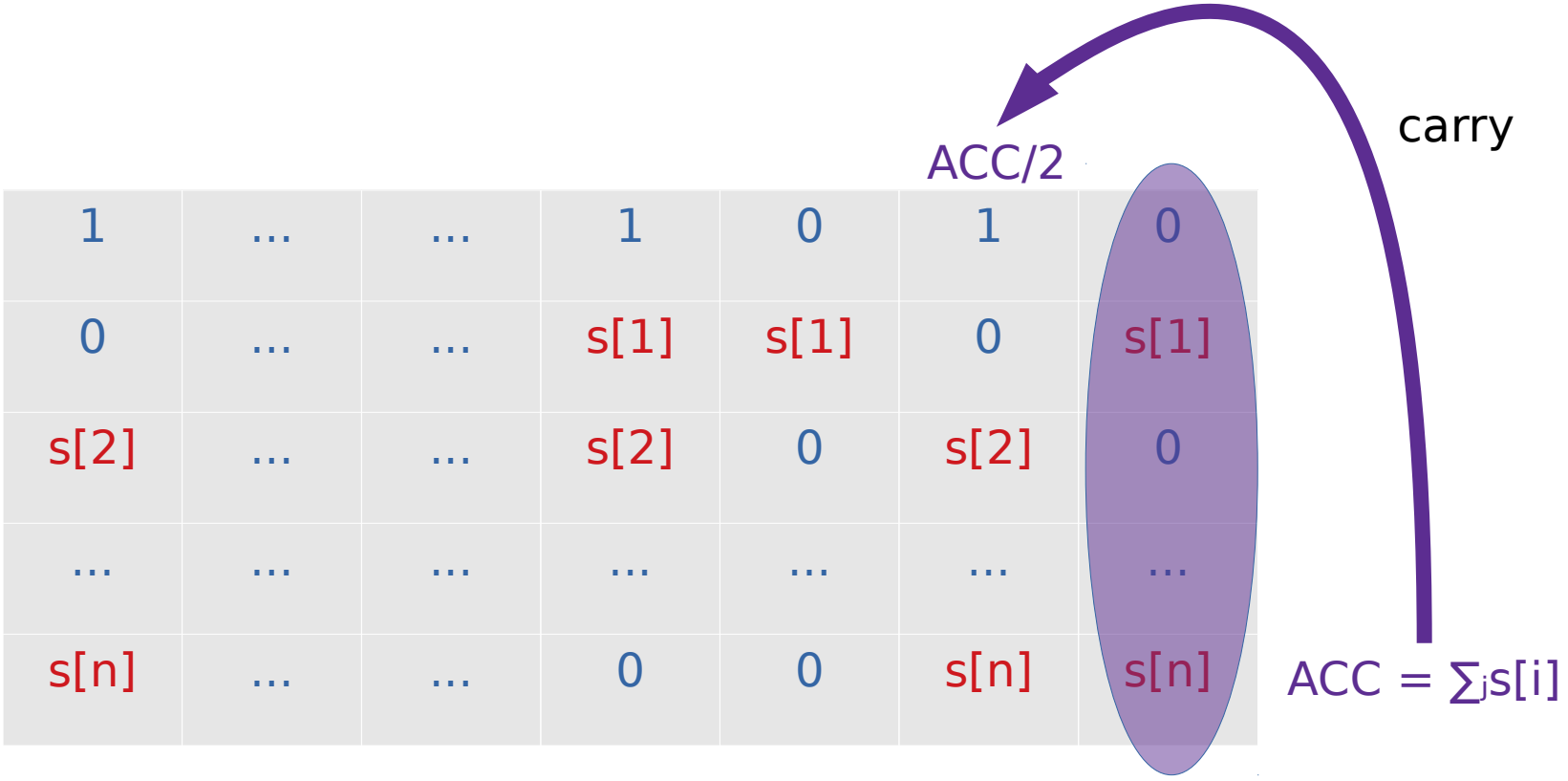
- Dec(s[i],[a[i],b]) = msb(∑...s[i])

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]

$ACC = \sum_j s[i]$

Homomorphic Decryption

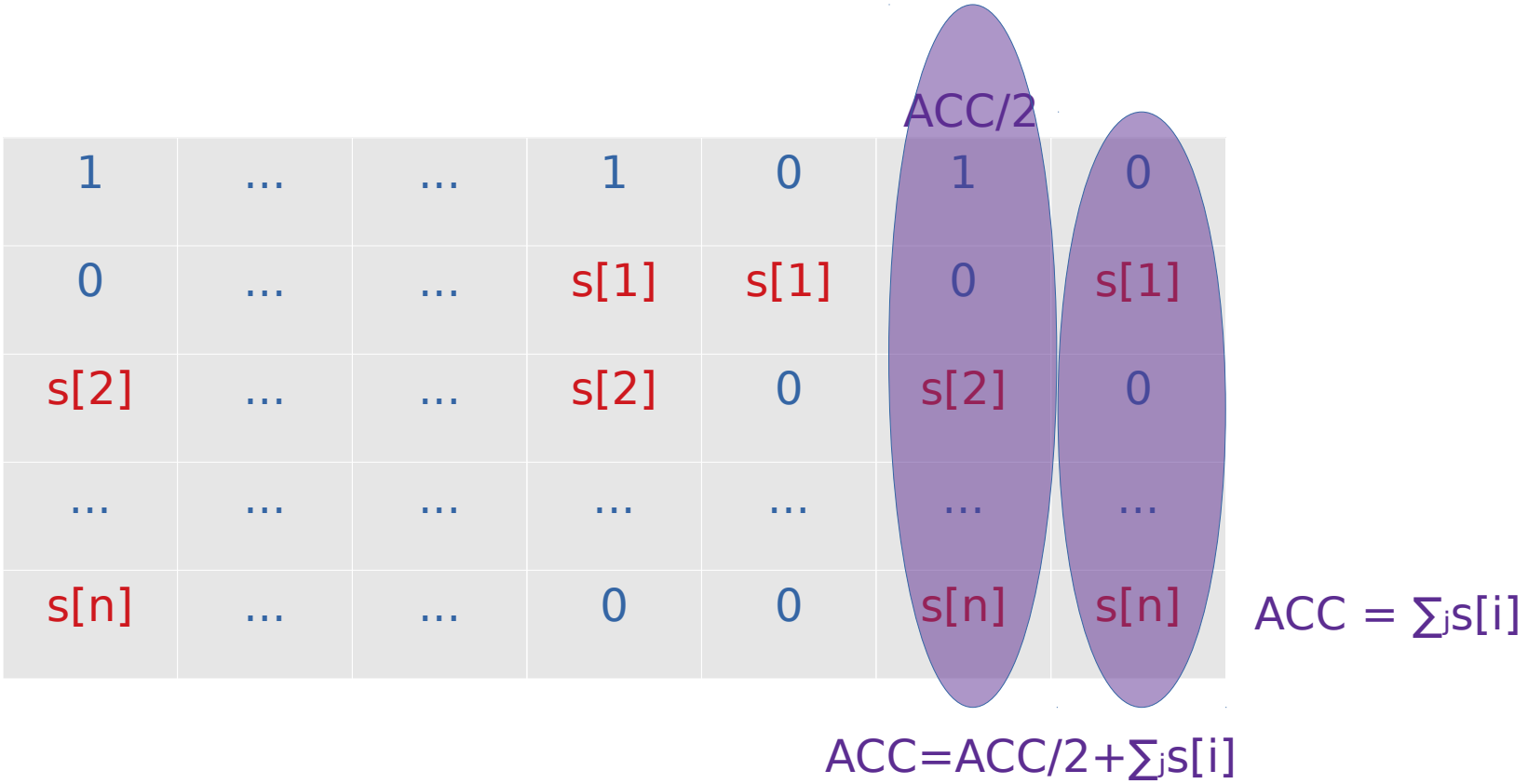
- Dec(s[i],[a[i],b]) = msb(∑...s[i])



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]

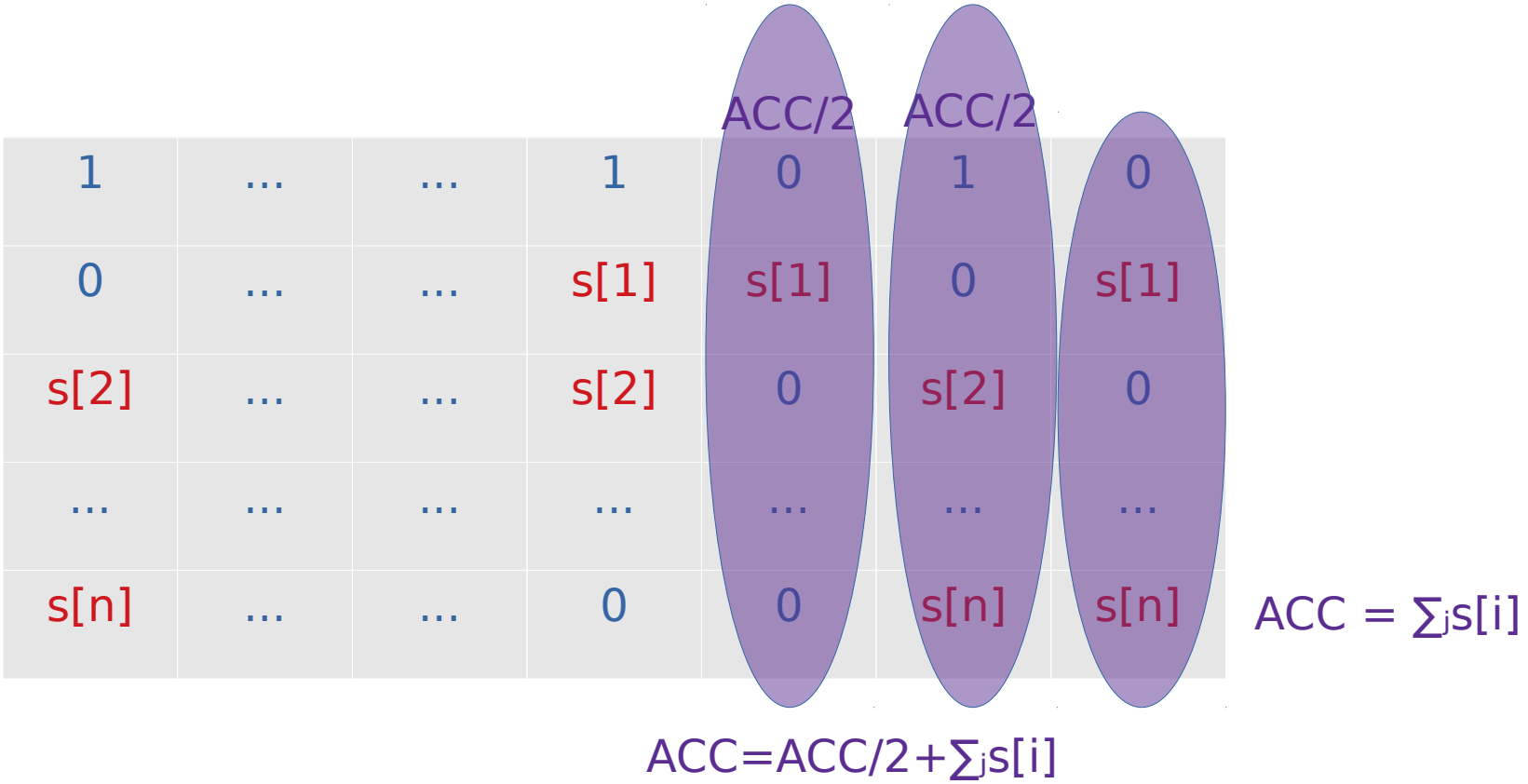
Homomorphic Decryption

- Dec(s[i],[a[i],b]) = msb(∑...s[i])



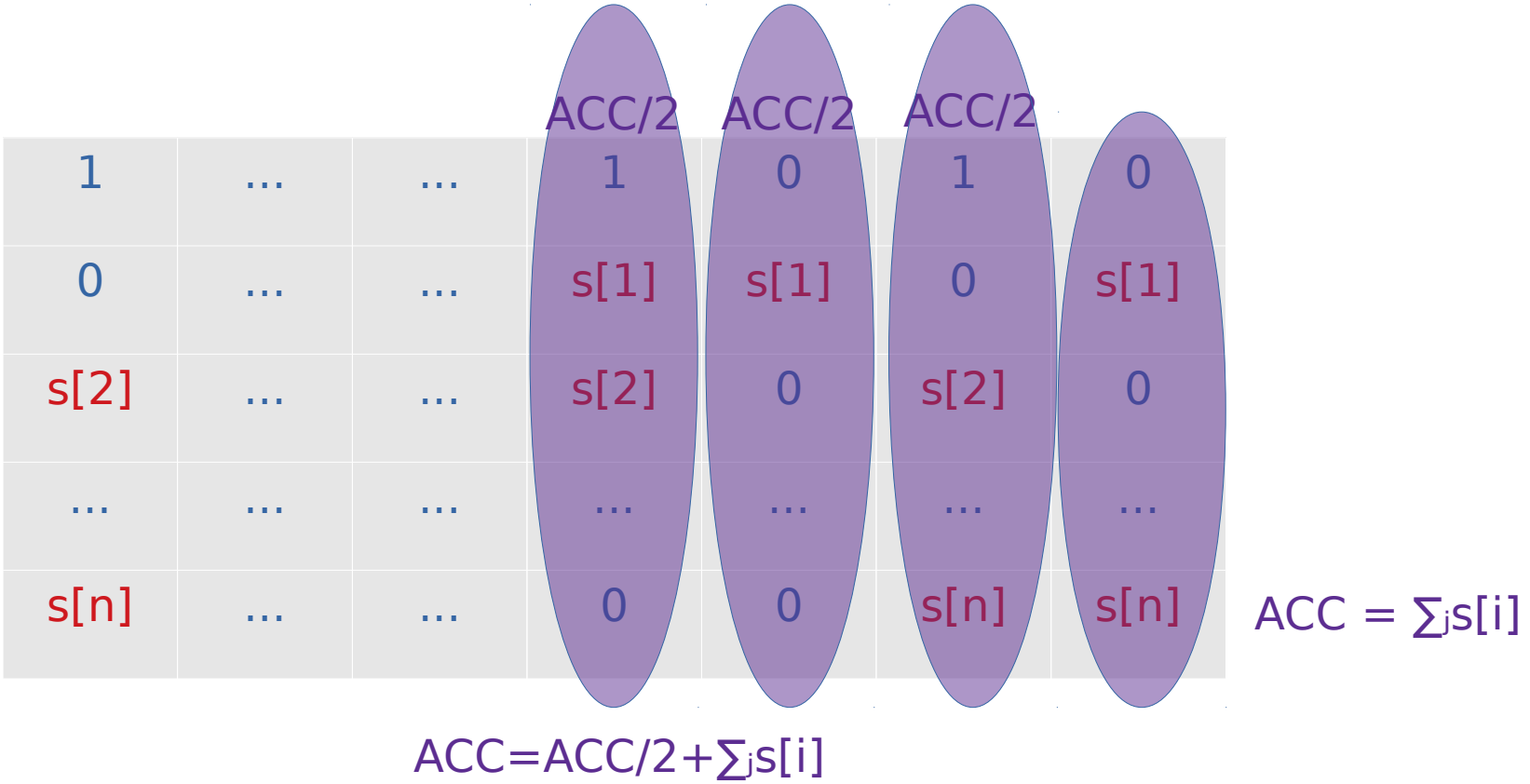
Homomorphic Decryption

- Dec(s[i],[a[i],b]) = msb(∑...s[i])



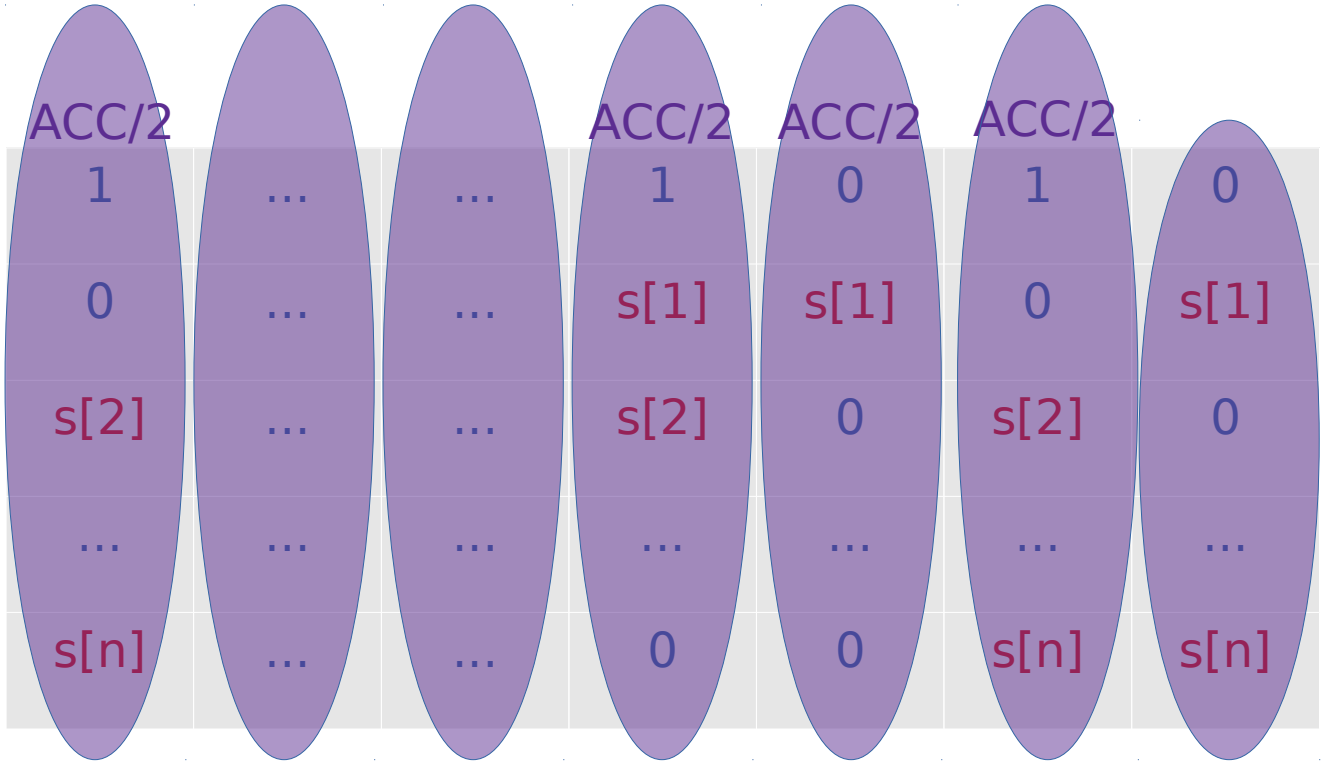
Homomorphic Decryption

- Dec(s[i],[a[i],b]) = msb(∑...s[i])



Homomorphic Decryption

- Dec(s[i],[a[i],b]) = msb(∑...s[i])



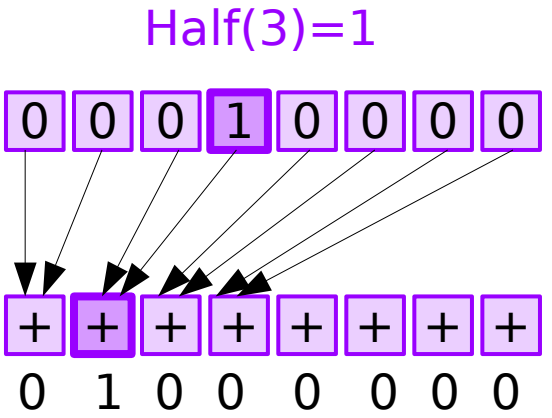
msb(∑) = ACC mod 2

Cryptographic Accumulator

- ACC[v] holds values $v \in \{0,\ldots,N=2(n+1)\}$
- 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)

Alperin-Sheriff & Peikert ACC

- ACC[v;β]=(c[0],…,c[N]) where
  - c[v]=Enc(1;β_v), c[u]=Enc(0;β_u)
  - $\beta = \sum_i \beta_i$
- f(ACC[v;β]) = ACC[f(v);β]
  - c'[v] = ∑ { c[u] | f(u)=v }
  - ∑ {} = Enc(0;0)
- Increment, Half, Mod2: choose your f!
- Extract(ACC) = c[1]



Accumulate

- Accumulate: ACC[v] + E[s] → 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]
- Need an encryption scheme E supporting
  - E[s] → (1 - E[s]) = E[1-s]
  - Enc(m) * E[s] → Enc(m*s)

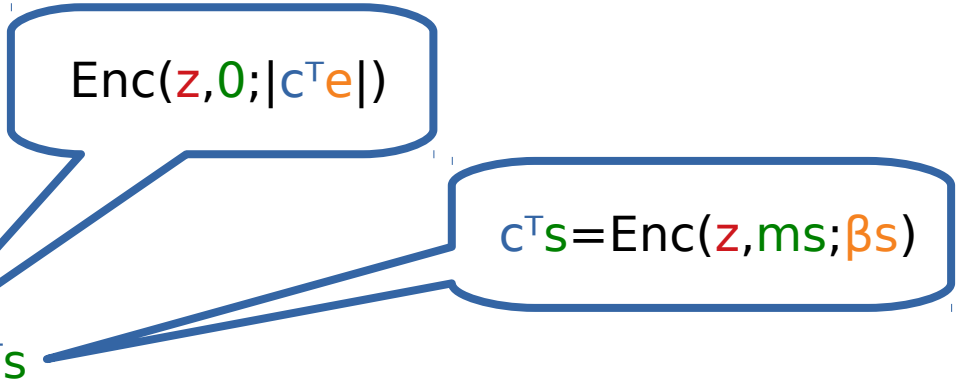
GSW Encryption

- Gentry, Sahai, Waters (Crypto 2013)
  - LWE variant with G=$\mathbf{I}$
  - Enc'[s, m; [A,e]] = [A,As+e] + m$\mathbf{I}$
- Last row is a Regev LWE ciphertext
  - LastRow([A,As+e] + m$\mathbf{I}$) = [A,As+e+m]
  - Can still decrypt using s
  - … but we don't care ….
- We can multiply ciphertexts!

GSW Multiplication

- $Enc'(z,s; A,e) = [A, Az+e] + s\mathbf{I}$
- For any $c^T = Enc(z,m;\beta)$

$$c^T * Enc'(z,s; A,e) = c^T([A, Az+e] + s\mathbf{I})$$
$$= [c^T A, c^T Az + c^T e] + c^T s$$
$$= Enc(z, ms; |c^T e| + \beta s)$$



- Problem: $|c^T e|$ is too big!
- Solution: $E[s] = (Enc'(s), Enc'(2s), Enc'(4s), \ldots, Enc'((q/2)s))$
- $c = \sum_j 2^j c_j$ where $c_j \in \{0,1\}^{n+1}$
- $c^T * E[s] = \sum_j c_j^T * Enc'(2^j s) = \sum_j Enc(0; |c_j^T e_j|) + c_j^T 2^j s$

$$= Enc(0; \textstyle\sum_j |c_j^T e_j|) + \sum_j c_j^T 2^j s$$

where $\sum_j c_j^T 2^j s = c^T s = Enc(ms; \beta s)$

Summary

- LWE encryption (Regev 2005 / BFKL 1993)
  - Linearly homomorphic: supports only addition
  - Noisy: bounded number of additions
- Bootstrapping (Gentry 2009)
  - Reduce noise by homomorphic decryption
- FHEW mod4 trick (Ducas, Micciancio 2015)
  - Arbitrary computations: bootstrap(x/2)
- Implement homomoprhic decryption
  - Accumulators (AlperinSheriff, Peikert 2014)
  - New decryption algorithm (schoolbook addition)
- Easy to implement, but not very efficient

Efficiency

- Circuit C: |C| bootstrapping
  - 1 Bootstrapping: (n*log q) ACC operations
  - 1 ACC operation: |ACC|=O(n) GSW products
  - 1 GSW product: log q matrix-vector mult. = $O(n^2 \log q)$
- Total: $O(n^4 \log^2 q)$ arithmetic ops per gate
  - Say, $n=256=2^8$, $q=n^4$: Total=2^{42} ops per gate
- If s is not binary:
  - multiply by another log q = 4 log n = 32 = 2^5
- Perhaps even larger for high security

Security

- Given E[z,s[i]]
  - Enc(s,m_1;q/16)+Enc(s,m_2;q/16) = Enc(s,m_1+m_2;q/8) → Enc(z,m_1+m_2;q/16)
  - Nees one key (and one E(z,s)) for every layer of the circuit
- Given E[s,s[i]]
  - Enc(s,m_1;q/16)+Enc(s,m_2;q/16) = Enc(s,m_1+m_2;q/8) → Enc(s,m_1+m_2;q/16)
  - But, does E[s,s[i]] leak information about s?

Circular Insecurity

- Can you recover s from Enc(s,s)?
- Can you distinguish Enc(s,s) from Enc(s,0)?
- Goldwasser, Micali (1982): Yes!
  - Enc(s,s)=(0,s), Enc(s,m)=(1,Enc'(s,m))
- What about bit encryption Enc(s,s[i])
  - Rothblum 2013: you can build insecure schemes using multilinear maps
  - Goyal, Koppula, Waters 2017: you can build them even from LWE

Applications of Circular Security

- Special case of Key-Dependent-Message KDM security
  - Black, Rogaway, Shrimpton 2002: defined and achieved in the Random Oracle model
- Many applications beyond FHE:
  - Security of Formal encryption (Abadi,Rogaway 2002, Micciancio 2009, …)
  - Anonymous credentials (Camenish, Lysyanskaya 2001)
  - Trapdoor functions, CCA security, (Hajiabadi, Kapron, 2015)
  - …

Circular Secure Encryption

- You can build circular secure encryption from:
  - DDH:Boneh,Halevi,Hamburg,Ostrovsky(2008)
  - LWE: Applebaum, Cash, Peikert, Sahai (2009)
  - QR: Brakerski, Goldwasser (2010)
- Extensions to KDM security:
  - Brakerski, Goldwasser, Kalai (2011)
  - Applebaum (2011)
  - Malkin, Teranishi, Yung (2011)

Circular Security for FHE

- We built FHE from
  - LWE Enc()=[a,a^Ts+e+s[i]] : used for linear gates
  - GSW E[]=[A,As+e]+$\mathbf{I}$s[i] : used for boostrapping
- Need circular bit-security of GSW scheme E[]
- LWE scheme Enc is circular secure! [ACPS'09]

  (-1,0,…,0,0) = Enc(s,s[1];(-1,0,…,0,0))

- Open Problem: What about the GSW variant?

  E[s,s[i];A,e] = [A,As+e]+$\mathbf{I}$s[i]
  - Break it, or
  - Prove it is as hard to break as solving LWE.

Thank You!

Questions?