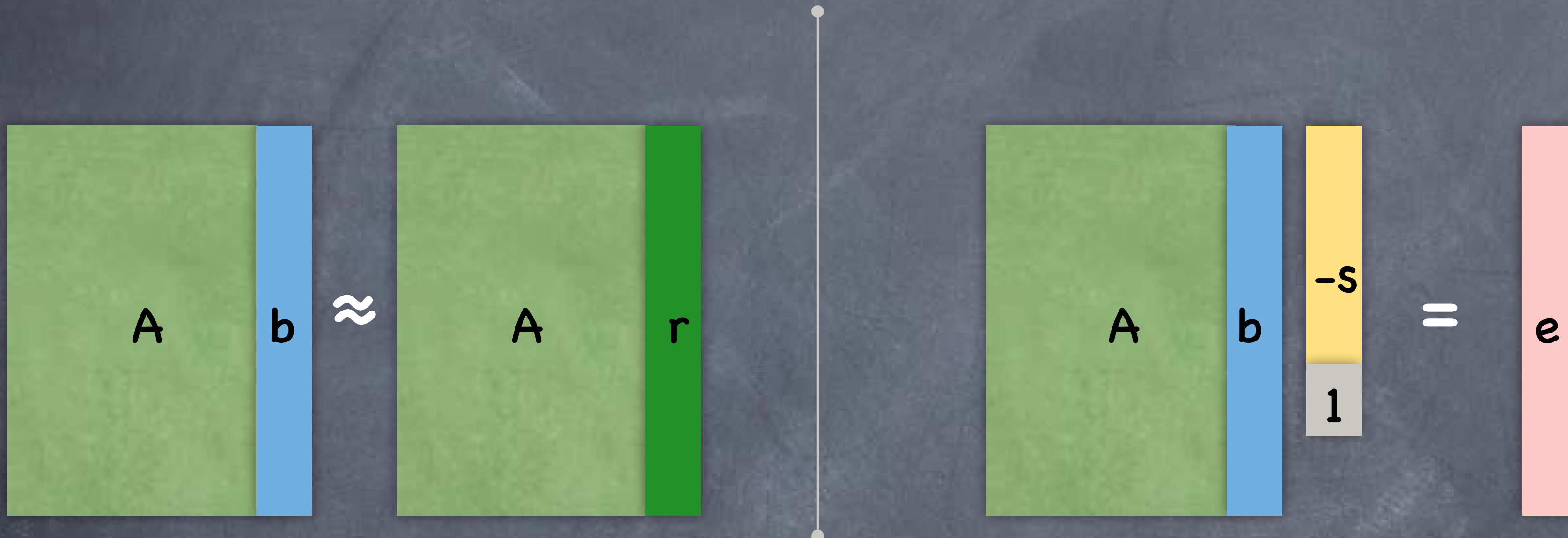
Fully Homomorphic Encryption

Lecture 21

Recall

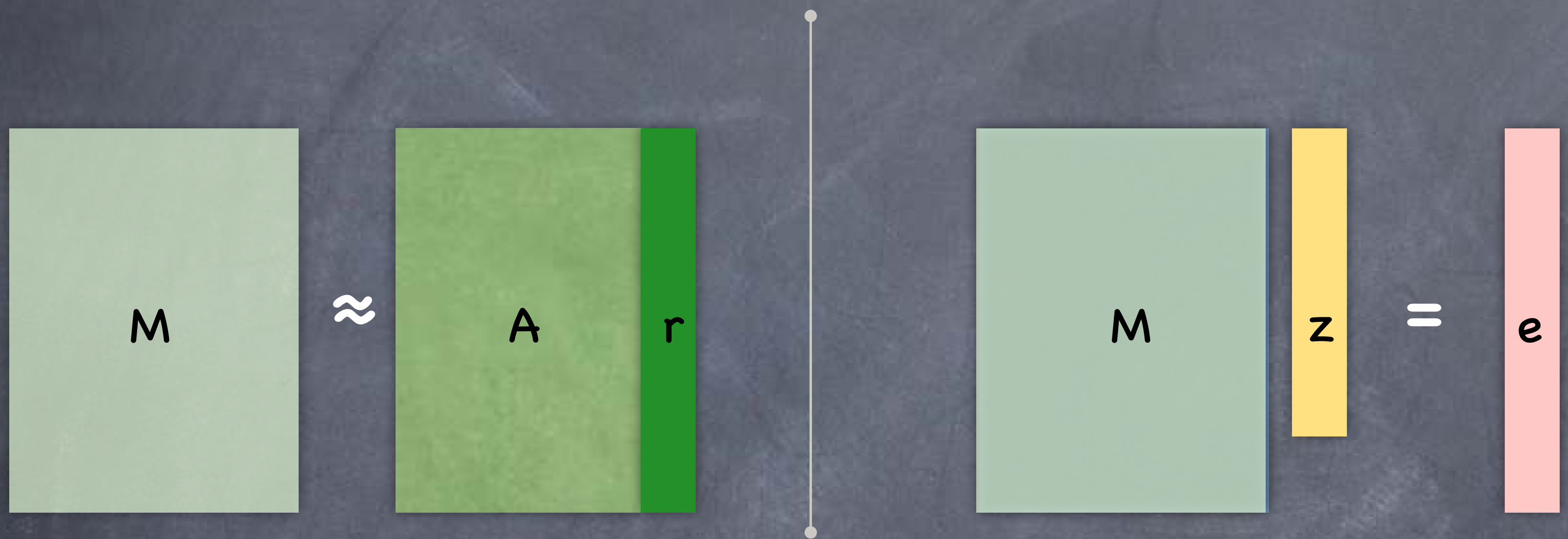
Learning With Errors



- LWE (decision version): $(A, A\underline{s}+\underline{e}) \approx (A, \underline{r})$, where A random matrix in $A \in \mathbb{Z}_q^{m\times n}$, $\underline{s}$ uniform, $\underline{e}$ has "small" entries from a Gaussian distribution, and $\underline{r}$ uniform.
- Average-case solution for LWE $\Rightarrow$ Worst-case solution for GapSVP (for appropriate choice of parameters)

Recall

Learning With Errors



- A pseudorandom matrix $M \in \mathbb{Z}_q^{m\times n'}$ and $\underline{\mathbf{z}} \in \mathbb{Z}_q^{n'}$ s.t. entries of $M\underline{z}$ are all small

Recall

Gentry-Sahai-Waters

The Actual Scheme

- Supports messages $\mu \in \{0,1\}$ and NAND operations up to an a priori bounded depth of NANDs
- Public key: Pseudorandom $M \in \mathbb{Z}_q^{m \times n}$ s.t. m >> n log q
  Private key: non-zero $\underline{z}$ s.t. $M\underline{z}$ has small entries
- Enc(μ) = $M^T R + \mu G$ where $R \leftarrow \{0,1\}^{m \times kn}$ and $G \in \mathbb{Z}_q^{n \times kn}$

  (G is the matrix to reverse bit-decomposition)
- $Dec_z(C)$: $\underline{z}^T C = \underline{\delta}^T + \mu \underline{z}^T G$ where $\underline{\delta}^T = e^T R$

Decrypting G yields 1

- $NAND(C_1,C_2)$: $G - C_1 \cdot B(C_2)$
  - $\underline{z}^T C_1 \cdot B(C_2) = \underline{z}^T C_1 \cdot B(C_2) = (\underline{\delta}_1^T + \mu_1 \underline{z}^T G)\, B(C_2)$
    $= \underline{\delta}_1^T B(C_2) + \mu_1 \underline{z}^T C_2 = \underline{\delta}^T + \mu_1 \mu_2 \underline{z}^T G$
    where $\underline{\delta}^T = \underline{\delta}_1^T B(C_2) + \mu_1 \underline{\delta}_2^T$ has small entries
  - In general, error gets multiplied by kn. Allows depth $\approx \log_{kn} q$

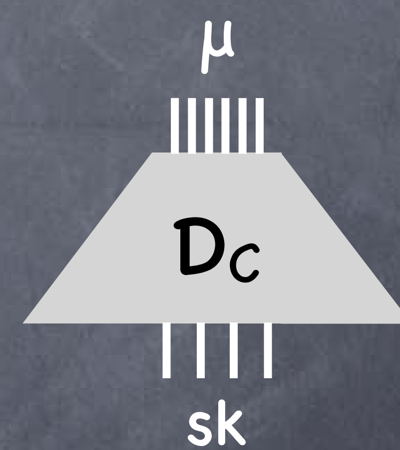
Only "left depth" counts, since $\underline{\delta} \leq k \cdot n \cdot \underline{\delta}_1 + \underline{\delta}_2$

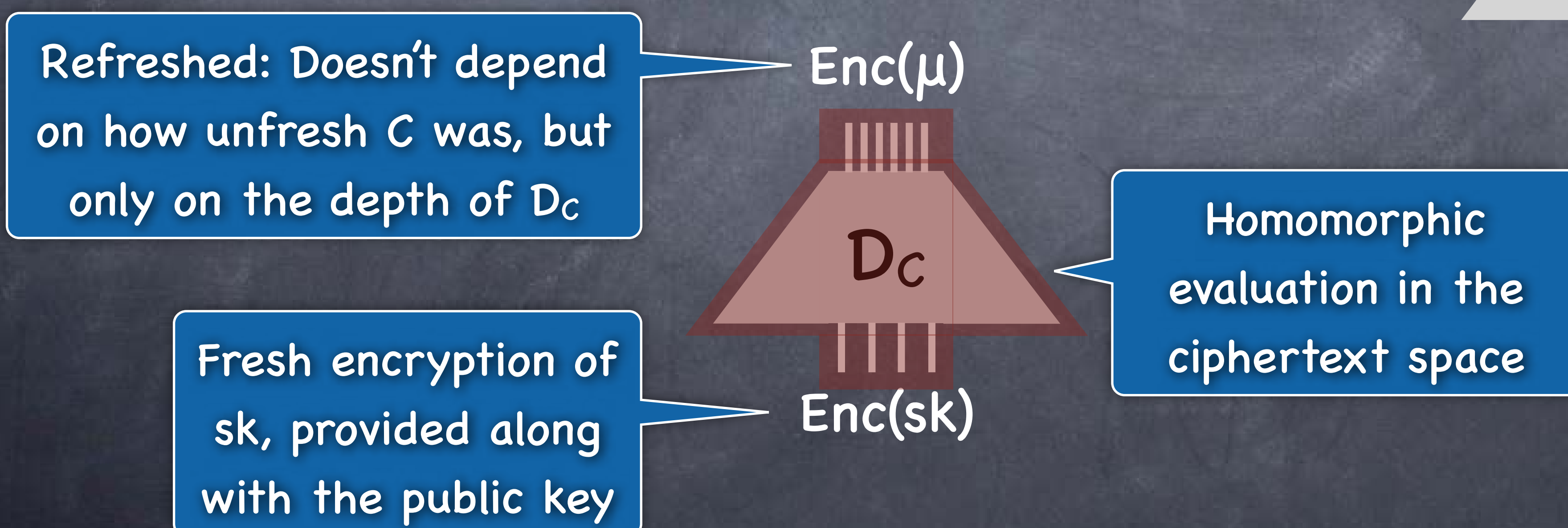
Bootstrapping

- Removing the need for an a priori bound
- Main idea: Can "refresh" the ciphertext to reduce noise
  - Refresh: homomorphically decrypt the given ciphertext under a fresh layer of encryption
    - cf. Degree reduction via share-switching: Homomorphically reconstruct under a fresh layer of sharing
    - But here, the reconstruction operation (i.e., decryption) is not known to the party doing the refresh, because the secret-key is not known
    - Idea: Give an encryption of the secret-key and use homomorphism!
  - Will consider decryption of a given ciphertext as a function applied to the secret-key: $D_C(sk) := Dec(C,sk)$

Bootstrapping

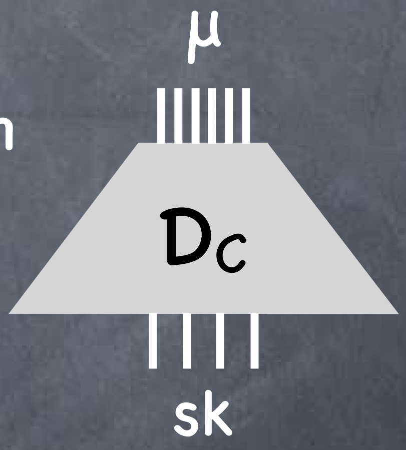
- Given a ciphertext C and hence the decryption function D_C s.t. $D_C(sk) := Dec(C,sk)$
- Also given: an encryption of sk (beware: circularity!)
- Goal: a fresh ciphertext with message $D_C(sk)$

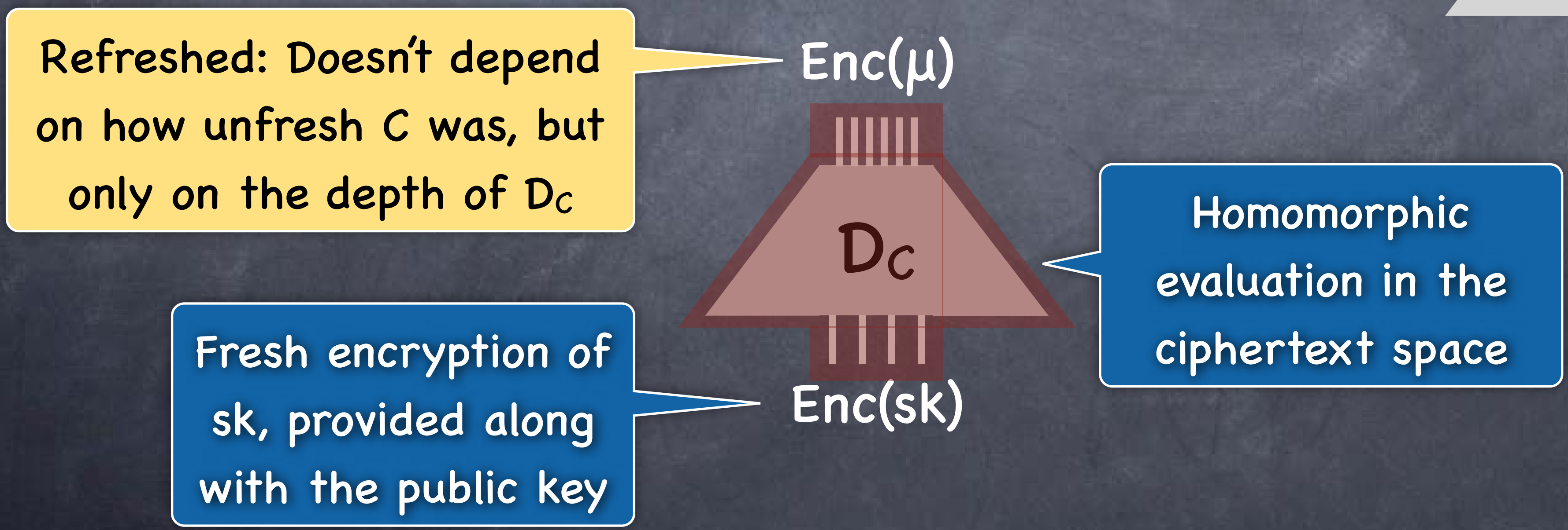




Bootstrapping

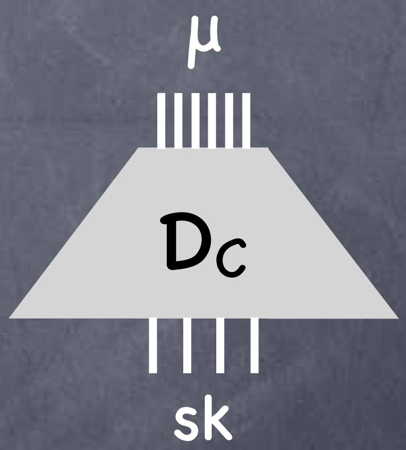
- If depth of D_C s.t. $D_C(sk) := Dec(C,sk)$ is strictly less than the depth allowed by the homomorphic encryption scheme, a ciphertext C can be strictly refreshed
  - Then can carry out at least one more operation on such ciphertexts (before refreshing again)



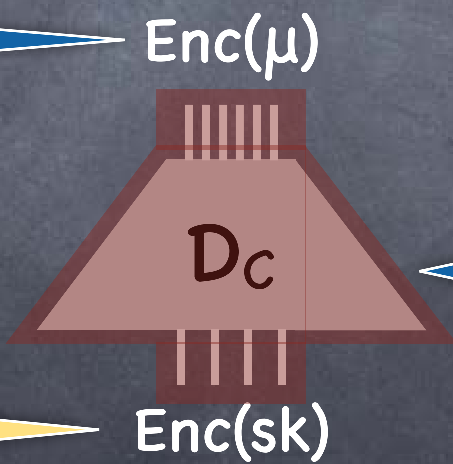


Bootstrapping

- Circularity: Encrypting the secret-key of a scheme under the scheme itself
    - Can break security in general!
- LWE does not by itself imply security
- Stronger assumption: "Circular Security of LWE"



Refreshed: Doesn't depend on how unfresh C was, but only on the depth of D_C



Homomorphic evaluation in the ciphertext space

Fresh encryption of sk, provided along with the public key

Bootstrapping GSW

- Supports log(k) depth computation with poly(k) complexity
- Need low depth decryption (as a function of secret-key)
- $Dec_z(C)$: $\underline{z}^T C = \underline{\delta}^T + \mu \underline{z}^T G$ where $\underline{\delta}^T = e^T R$
  - And then check if the result is close to $\underline{0}^T$ or $\underline{z}^T G$
  - How?
  - Multiply by $B(\underline{w})$ where last coordinate of $\underline{w}$ is $\lfloor q/2 \rfloor$ and other coordinates 0
  - $\underline{z}^T C\, B(\underline{w}) = \underline{\delta}^T B(\underline{w}) + \mu \underline{z}^T \underline{w} = \varepsilon + \mu \lfloor q/2 \rfloor$
    - Has most significant bit = μ (since error $|\varepsilon| \ll q/4$)
- $Dec_z(C)$: MSB($\underline{z}^T C\, B(\underline{w})$). All operations mod q.
  - If q were small (poly(k)) this would be small depth (log(k))
  - Problem: q is super-polynomial in security parameter k
  - Idea: Can change modulus for decryption!

Modulus Switching for GSW

- $\mathrm{Dec}_z(C)$: MSB($\underline{z}^T Y$ % q), where Y = C B($\underline{w}$)
  - $\underline{z}^T Y = \varepsilon_0 + \mu\,(q/2) + aq$ (for some $a \in \mathbb{Z}$)
- To switch to a smaller modulus $p < q$:
  - Consider $Y' = \lceil (p/q)\,Y \rfloor$. Let $\Delta = Y' - (p/q)Y$.
  - $\underline{z}^T Y' = (p/q)\,\underline{z}^T Y + \underline{z}^T \Delta$

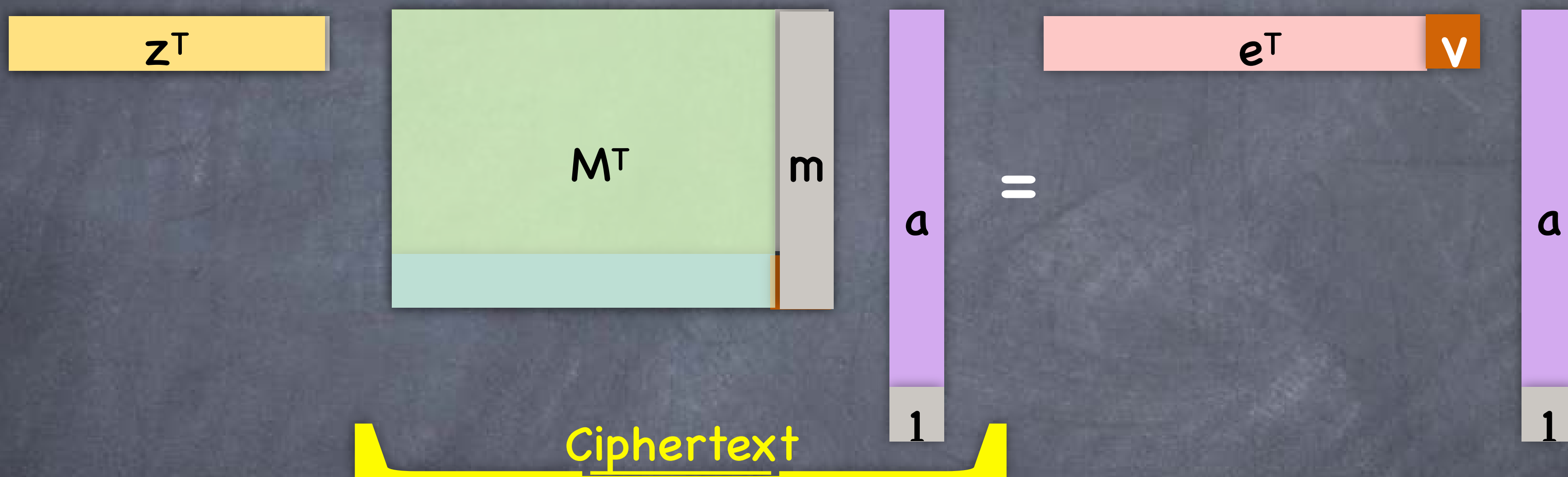
    $= \varepsilon_1 + \mu\,(p/2) + ap$ where $\varepsilon_1 = (p/q)\varepsilon_0 + \underline{z}^T \Delta$
  - Need $\underline{z}^T \Delta$ to be small. But $\underline{z}^T = [\ -\underline{s}^T\ 1\ ]$ for $\underline{s}$ uniform in $\mathbb{Z}_q^n$.
    - Fix: LWE with small $\underline{s}$ is as good as with uniform $\underline{s}$ [Exercise]
- Final bootstrapping:
  - Given C, let $Y' = \lceil (p/q)\ C\ B(\underline{w}) \rfloor$ where p small (poly(k)). Define function $D_{Y'}$ which does decryption mod p. Homomorphically evaluate $D_{Y'}$ on encryption of $\underline{z}$ mod p (encryption is mod q).

Other FHE Schemes

- Gentry (2009)
- Brakerski-Vaikuntanathan, Brakerski-Gentry-Vaikuntanathan (2011-12)
- Brakerski and Fan-Vercauteren (2012)
- Gentry-Sahai-Waters (2013)
- ...
- Schemes based on Ring LWE allow <u>batching</u>: encoding multiple messages into a single message, using Chinese Remainder Theorem
- Many of these schemes obtain Levelled FHE without bootstrapping

Recall

PKE from LWE



- Ciphertext $\underline{C} = M^T\underline{\mathbf{a}} + \underline{\mathbf{m}}$; $\underline{\mathbf{m}}$ encodes the message and $\mathbf{a} \in \{0,1\}^m$
- Decryptng: From $\underline{\mathbf{z}}^T\underline{C} = \underline{\mathbf{e}}^T\underline{\mathbf{a}} + \underline{\mathbf{z}}^T\underline{\mathbf{m}}$ where $\underline{\mathbf{e}}^T\underline{\mathbf{a}}$ is small. To allow decoding from this for, say $\mu \in \{0,1\}$, let $\underline{\mathbf{z}}^T\underline{\mathbf{m}} = v \approx \mu(q/2)$.
- Variant: $\underline{\mathbf{e}}$ has (small) even entries and $\underline{\mathbf{m}}^T = (0 \ldots 0\ \mu)$. Then $(\underline{\mathbf{z}}^TC)\ \%\ q = \mu \pmod 2$.

BGV Scheme: Overview

$m^T = (0 \dots 0\ \mu)$
and $\underline{e}$ has even entries

- Ciphertext $C = M^T\mathbf{\underline{a}} + \mathbf{\underline{m}}$; $\mathbf{\underline{m}}$ encodes $\mu \in \{0,1\}$ and $\mathbf{a} \in \{0,1\}^m$
- Decrypting: $(\underline{\mathbf{z}}^T\underline{C}\ \%\ q)\ \%\ 2$.
- Already supports homomorphic addition (upto a certain number of levels, determined by q, size of noise and dimension m)
- To support a single homomorphic multiplication, consider moving to a different key (and dimensions) after one multiplication, so that $\underline{\mathbf{z}_{new}}^T\underline{C}\ \%\ q = (\underline{\mathbf{z}}^T\underline{C_1}\ \%\ q)\ (\underline{\mathbf{z}}^T\underline{C_2}\ \%\ q)\ \ (\text{mod } 2)$
  - Want $\underline{\mathbf{z}_{new}}^T\underline{C}\ \%\ q\ \%\ 2 = (\underline{\mathbf{z}}^T\underline{C_1}\ \%\ q\ \%\ 2)\ (\underline{\mathbf{z}}^T\underline{C_2}\ \%\ q\ \%\ 2)$
    $= (\underline{\mathbf{z}}^T\underline{C_1})\ (\underline{\mathbf{z}}^T\underline{C_2})\ \%\ q\ \%\ 2$ (when each $\underline{\mathbf{z}}^T\underline{C_b}\ \%\ q < \sqrt{q}$)
    - $(\underline{\mathbf{z}}^T\underline{C_1})\ (\underline{\mathbf{z}}^T\underline{C_2}) = \Sigma_{ij}\ z_iC_{1,i}\ z_jC_{2,j} = \Sigma_{ij}\ (z_i \cdot z_j)(C_{1,i} \cdot C_{2,j})$.
      So can take $\underline{\mathbf{z}_{new}} = \underline{\mathbf{z}} \otimes \underline{\mathbf{z}}$ and $\underline{C} = \underline{C_1} \otimes \underline{C_2}$.

BGV Scheme: Overview

- To support a single homomorphic multiplication, let $\underline{C} = \underline{C_1} \otimes \underline{C_2}$ and move to key $\underline{\mathbf{z}_{big}} = \underline{\mathbf{z}} \otimes \underline{\mathbf{z}}$
- To allow repeated multiplications, need to do dimension reduction (cf. degree reduction in BGW)
  - Will use bit-decomposition operation $B(\cdot)$ and its inverse G
  - To switch from $\underline{C}$ w.r.t $\underline{\mathbf{z}_{big}}$ to $\underline{C'}$ w.r.t keys $(M', \underline{\mathbf{z}'})$ (where $\underline{\mathbf{z}'}^T M' = \underline{\mathbf{e}'}^T$ has small even entries), preserving message:
    - Include $D = (M' + Z_{big}\, G)$ in the public-key, where $Z_{big} = [0|\ \underline{\mathbf{z}_{big}}]^T$ (so that $\underline{\mathbf{z}'}^T Z_{big} = \underline{\mathbf{z}_{big}}^T$).
  - Switching: let $C' = D \cdot B(\underline{C})$. Then $\underline{\mathbf{z}'}^T C' = \underline{\mathbf{e}'}^T B(\underline{C}) + \underline{\mathbf{z}_{big}}^T \underline{C}$.
- Noise kept under control by repeated modulus switching
  - Levelled FHE, with lowest level using the highest modulus

FHE in Practice

- Several implementations in recent years
  - Prominent ones based on schemes of Fan-Vercauteren (FV) and Brakerski-Gentry-Vaikuntanathan (BGV) with various subsequent optimisations
    - BGV implementations: HELib (IBM), Λ ο λ
    - FV implementations: SEAL (Microsoft), FV-NFLlib (CryptoExperts), HomomorphicEncryption R Package …
  - Both based on "Ring LWE"
- Moderately fast
  - E.g., HELib can apply AES (encipher/decipher) to about 200 plaintext blocks using an encrypted key in about 20 minutes (a bit faster without bootstrapping, if no need to further compute on the ciphertext)