**//Partially Linearly Homomorphic Encryption (CGS),**

**//Key Generation function**

**// Input of this function is a security parameter n**

**//Output of this function is (p, q, g, a, G)**

// a is the secret key

//g, p and q are Domain(system) parameters, common.

// G is the public key.

, //q is a randomly picked n-bit prime;

p=2q+1, // p is a prime

; // g is the generator of the subgroup ; //if

**;**

***// Encryption Function for the CGS***

**// Input to this function is a pair (Pk**

**//**  M is the message space (in our case B=10000)

//Pk is the public key, Pk=G

//g and q are Domain(system) parameters, common.

**// Output of this function is a cipher text C**

// ; is a cipher text space (i.e. CGS cipher text space)

// C is a sequence of two group elements; C=[x,y]

**:=function()**

;

**Return** [x,y];

***// Decryption Function for the CGS***

**// Input to this function is a pair (sk**

// ; is a cipher text space (i.e. CGS cipher text space).

// C is a sequence of two group elements; C= [x, y];

//sk is the secret key, sk=a.

//p is Domain(system) parameter, common.

//G is public key.

**// Output of this function is the message m**

**: = function ()**

,

**Return** i;

***// Evaluation Functions for the CGS***

**// Addition function in CGS**

**// Input to this function is a pair**

// ; is a cipher text space (i.e. CGS cipher text space),

//g, p, G are Domain(system) parameters, common.

**// Output of this function is a cipher text**

// ; is a cipher text space (i.e. CGS cipher text space)

// is a sequence of two group elements; C=[x,y]

**EncryptedAdd:=function**

**Return** [x,y];

*//Correctness:*

//=([]+ [])

//

**//**

**//**

**//**

**//**

//

**// Scalar multiplication function in CGS**

**// Input to this function is a pair**

//s is a scalar, .

// ; is a cipher text space (i.e. CGS cipher text space),

//g, p, G are Domain(system) parameters, common.

**// Output of this function is a cipher text**

// ; is a cipher text space (i.e. CGS cipher text space)

// is a sequence of two group elements; C=[x,y]

**EncryptedSmult:=function**

**Return** [x,y];

*//Correctness:*

*//*

**//**

**//**

**//**

//

**// Linear combination function (Addition and Scalar multiplication) in CGS**

**// Input to this function is a pair of sets**

//**;**  are a scalar, .

// are cipher texts, ; is a cipher text space (i.e. CGS cipher text space),

//g, p, G are Domain(system) parameters, common.

**// Output of this function is a cipher text**

// ; is a cipher text space (i.e. CGS cipher text space)

// is a sequence of two group elements; C= [x, y]

**EncryptedAddSmultGeneral:=function**

;

for i=1 to k do

End for;

**Return** [x,y];

*//Correctness:*

//

//

//

**//**

**//**

**//**

**//**

//

//Boosting Linearly Homomorphic Encryption (LHE) for multiplication, HE=(KeyGen, Enc, Dec, Eval)

**//Key Generation *Function for the BOOSTED-CGS***

// Same as CGS key generation

**// Input to this function is a**

// is the security parameter.

**// Output of this function is a pair (Pk, sk)**

**Pk**=public key G; **sk**=secret key a; g, p and q are Domain(system) parameters, common;

//)

***// Encryption Function for the BOOSTED-CGS***

// Input to this function is a pair (Pk

//Pk is the public key (g, q, G)

// Output of this function is a cipher text C

// where is a message space and is a cipher text space (i.e. CGS cipher text space)

//=

**Enc :=function()**

// b is randomly chosen by randomized encryption algorithm from message space M

//

;

**Return**

***// Decryption Function for the BOOSTED-CGS***

***//Level-1 decryption function***

**// Input to this function is a pair (sk**

// sk is the secret key

// where is a message space and is a cipher text space (i.e. CGS cipher text space)

//=

**// Output of this function is the message m**

**Dec\_1:=function(sk,C)**

**Return**

***//Level-2 decryption function***

**// Input to this function is a pair (sk**

// sk is the secret key

// where is a cipher text space (i.e. CGS cipher text space)

// =

**// Output of this function is the message m**

**Dec\_2:=function(sk,C)**

//

**Return**

**//Evaluation functions (EVAL)**

// is the addition is CGS cipher text space, and is the multiplication is CGS cipher text space,

***//Addition in the BOOSTED-CGS level 1***

// **Input to this function is a pair**

// is a level-1 ciphertext.

// where is a message space and is a cipher text space (i.e. CGS cipher text space)

// ==

**// Output of this function is a level-1 cipher text**

**Add\_1 :=function()**

;

**Return**

*//Correctness:*

//

//; ;

//;

//;

***//Multiplication in the BOOSTED-CGS level 1***

// **Input to this function is a pair**

// is a level-1 ciphertext.

// where is a message space and is a cipher text space (i.e. CGS cipher text space)

// ==

**// Output of this function is a level-2 cipher text**

**Mult\_1 :=function()**

**Return**

*//Correctness:*

//

//; ;

**//**

//

//

//

//;

***//Addition in the BOOSTED-CGS level 2***

// **Input to this function is a pair**

// are a level-2 ciphertexts.

// , where is a cipher text space (i.e. CGS cipher text space)

// ==

**// Output of this function is a level-2 cipher text**

**Add\_2 :=function()**

**Return**

*//Correctness:*

//

//

// i.e. .

//

// i.e. .

//

//

//

//.

***//Scalar multiplication in the BOOSTED-CGS level 1***

// **Input to this function is a pair (c,C)**

// c ϵ M is a scalar, where M is a message space

//C is a level-1 ciphertext

//C where is a cipher text space (i.e. CGS cipher text space)

// C=

**// Output of this function is a level-1 cipher text**

**CMult\_1 :=function(c,)**

**Return**

*//Correctness:*

//

//

//;

//

***//Scalar multiplication in the BOOSTED-CGS level 2***

// **Input to this function is a pair (c,C)**

// c ϵ M is a scalar, where M is a message space

//C is a level-2 ciphertext

// , where is a cipher text space (i.e. CGS cipher text space)

// =where is a cipher text space (i.e. CGS cipher text space)

**// Output of this function is a level-1 cipher text**

**CMult\_2 :=function(c,)**

**Return**

*//Correctness:*

//

//

//

//**.**