International Journal of Research
e-ISSN: 2348-6848 & p-ISSN 2348-795X Vol-5, Special Issue-11
International Conference on Multi-Disciplinary Research - 2017 held in February, 2018 in Hyderabad, Telangana State, India organised by GLOBAL RESEARCH ACADEMY - Scientific & Industrial Research Organisation (Autonomous), Hyderabad.

A Research Analysis of Leveled Fully Homomorphic Encryption from Bootstrappable Encryption Generically

Prof.Dr.G.Manoj Someswar [1], D.Narasimha Raju[2]

1.Research Supervisor, Dr.APJ Abdul Kalam Technical University, Lucknow,U.P., India
2.Research scholar, Dr.APJ Abdul Kalam Technical University, Lucknow,U.P., India

Abstract

We propose the principal completely homomorphic encryption plot, tackling a focal open issue in cryptography. Such a plan enables one to figure subjective capacities over encoded information without the decoding key { i.e., given encryptions E(m1); : ; E(mt) of m1; : ; mt, one can proficiently process a smaller cipher text that scrambles f(m1; : ; mt) for any effectively calculable capacity f. This issue was postured by Rivest et al. in 1978.

Completely homomorphic encryption has various applications. For instance, it empowers private inquiries to an internet searcher { the client presents a scrambled inquiry and the web index figures a concise encoded reply while never taking a gander at the question free. It additionally empowers looking on encoded information { a client stores scrambled files on a remote file server and can later have the server recover just files that (when decoded) fulfil some boolean requirement, despite the fact that the server can't unscramble the files all alone. All the more comprehensively, completely homomorphic encryption enhances the efficiency of secure multiparty calculation.

Our development starts with a to some degree homomorphic boost rappable" encryption plot that works when the capacity f is the plan's own unscrambling capacity. We at that point demonstrate how, through recursive self-implanting, boots trappable encryption gives completely homomorphic encryption. The development makes utilization of difficult issues on perfect cross sections.

Keywords: ***Security of the Abstract Scheme, Bootstrappable Encryption, Computational Complexity and Security, KDM-Secure Boots trappable Encryption, The Ideal Coset***

INTRODUCTION

Bootstrappable Encryption

Assume we have an encryption scheme E that compactly evaluates some set of circuits C_E. We want to use E to construct a homomorphic encryption scheme that can handle arbitrary circuits. In this Chapter we prove a fundamental result: that if C_E contains (slight augmentations of) E's *own decryption circuit* D_E { i.e., if E compactly evaluates" its (augmented) decryption circuit { then we can use E to construct an efficient scheme that handles circuits of arbitrary depth.

A bit more specifically, for any integer d, we use E to construct a scheme $E^{(d)}$ that can compactly evaluate circuits of depth up to d. The decryption circuit for $E^{(d)}$ is still D_E; the secret key and ciphertexts are the same size as in E. The public key in $E^{(d)}$ consists of d + 1 public keys from E, together with a chain of encrypted E secret keys { the first E secret key encrypted under the second E public key, and so on. In short, the family of schemes $fE^{(d)}\ g$ is leveled fully homomorphic. We base the semantic security of $E^{(d)}$ on that of E using a hybrid argument; as usual with hybrid arguments, the reduction loses a factor linear in d. In Chapter 4.3, we describe how one can obtain a fully homomorphic encryption scheme (where the public key size does not depend on the maximum number of levels we want to evaluate) by assuming key-dependent-message (KDM) security, specifically *circular-security* { i.e., that one can safely encrypt a E secret key under its associated public key.

Since this critical property of E { that it can compactly evaluate (slight augmentations of) its own decryption circuit { is self-referential and universal, we give it the obvious name: bootstrappability. Why should bootstrappability be such a powerful feature? At a high level, the reason is that bootstrappability allows us periodically to refresh" ciphertexts associated to interior nodes in a circuit; we can refresh for an arbitrary number of levels in the circuit, and thus can evaluate circuits of arbitrary depth. To refresh" a ciphertext that encrypts a plaintext ¼ under E public key pk_i, we *re-encrypt* it under pk_{i+1} and then *homomorphically apply the decryption circuit* to the result, using the secret key sk_i that is encrypted under pk_{i+1}, thereby obtaining an encryption of ¼ under pk_{i+1}. Homomorphically evaluating the decryption circuit decrypts the *inner* ciphertext under pk_i, but within homomorphic encryption under pk_{i+1}. The implicit decryption refreshes" the ciphertext, but the plaintext is never revealed; the plaintext is always covered by at least one layer of encryption. Now that the ciphertext is refreshed, we can continue" correctly evaluating the circuit.[1]

To see how this works mathematically, begin by considering the following algorithm, called Recrypt. For simplicity, suppose the plaintext space P is $f0;\ 1g$ and D_E is a boolean circuit in C_E. Let $(sk_1;\ pk_1)$ and $(sk_2;\ pk_2)$ be two E key-pairs. Let $\tilde{A}_1$ be an encryption of 1/4 *2 P* under pk_1. Let

sk_{1j} be an encryption of the *j*-th bit of the *first secret key* sk_1 under the *second public key* pk_2. Recrypt takes as these things as input, and outputs an encryption of ¼ under pk_2.

Recrypt(pk_2; D_E ; $h\overline{sk_{1j}i}$; $\tilde{A}_1$).

Set $\overline{\tilde{A}_{1j}}$ $\tilde{A}$ $\overset{R}{\text{Encrypt}}_E$ (pk_2; $\tilde{A}_{1j}$) where ~~$\tilde{A}_{1j}$~~ is ~~the~~ *j*-th bit of $\tilde{A}_1$ Set $\tilde{A}_2$ $\tilde{A}$ $\text{Evaluate}_E(pk_2; D_E; hhsk_{1j}i; h\tilde{A}_{1j}ii)$ Output $\tilde{A}_2$

Above, the Evaluate algorithm takes in all of the bits of sk_1 and all of the bits of $\tilde{A}_1$, each encrypted under pk_2. Then, *E* is used to evaluate the decryption circuit homomorphically. The output $\tilde{A}_2$ is thus an encryption under pk_2 of $\text{Decrypt}_E(sk_1; \tilde{A}_1)$ *!* ¼. The Recrypt algorithm implies a proxy one-way re-encryption scheme [19]. Roughly speaking, a one-way proxy re-encryption scheme allows the owner of sk_1 to generate a tag that enables an untrusted proxy to convert an encryption of ¼ under pk_1 to an encryption of ¼ under pk_2, but not the reverse. In our case, the tag is $hsk_{1j}i$, the encrypted secret key.[2] Strictly speaking, the security model for proxy re-encryption typically requires the security of the delegator's secret key, even against a collusion of delegatee's who also get to see the delegating tags. However, this requirement seems unnecessary, since a delegatee will be able to decrypt ciphertexts directed to the delegator anyway. In the Recrypt algorithm above, the plaintext ¼ is doubly encrypted at one point { under both pk_1 and pk_2. Depending on the encryption scheme *E*, however, this double encryption might be overkill. Suppose WeakEncrypt_E is an algorithm such that the image of WeakEncrypt_E (pk; ¼) is always a subset of the image of Encrypt_E (pk; ¼). Then we can replace the first step of Recrypt_E with:

Set $\overline{\tilde{A}_{1j}}$ $\tilde{A}$ $\overset{R}{\text{WeakEncrypt}}_E$ (pk_2; $\tilde{A}_{1j}$) where $\tilde{A}_{1j}$ is the *j*-th bit of $\tilde{A}_1$

Let us call this relaxation $\text{Recrypt}'_E$. The main point of this relaxation is that WeakEncrypt does not need to be semantically secure for $\text{Recrypt}'_E$ to be a secure one-way proxy re-encryption scheme, or for $\text{Recrypt}'_E$ to be useful toward bootstrapping (as we will see below). Thus, depending on *E*, WeakEncrypt_E can be very simple { e.g., for some schemes, and in particular for the ideal-lattice-based scheme that we describe later, WeakEncrypt_E might leave the input bits" entirely unmodified. This will unfortunately not help us much in terms of making the encryption scheme boots trappable; essentially, it will add one circuit level to what *E* can evaluate. However, it will affect the eventual *computational complexity* of our scheme, since it will require less computation to apply the decryption circuit homo-morphically to cipher texts in which the outer encryption is

weak.[3] Another way of viewing this relaxation is that we only need to be able to evaluate non-uniform decryption circuits, where the ciphertext is hard-wired" into the circuit (making this circuit simpler than the normal" decryption circuit that takes the ciphertext (and secret key) as input.

To be bootstrappable, *E* needs to be able to compactly evaluate not only its decryption circuit, which merely allows recryptions of the same plaintext, but also slightly augmented versions of it, so that we can perform binary operations on plaintexts and make actual progress through a circuit.

Let D_E be *E*'s decryption circuit, which takes a secret key and ciphertext as input, each formatted as an element of $P^{`(,)}$, where *P* is the plaintext space. Let ¡ be a set of gates with inputs and output in *P*, which includes the trivial gate (input and output are the same). We call a circuit composed of multiple copiesof D_E connected by a single *g* gate (the number of copies equals the number of inputs to *g*) a *g*-augmented decryption circuit." We denote the set of *g*-augmented decryption circuits, *g 2* ¡, by D_E(¡).

As before, let C_E denote the set of circuits that *E* can compactly evaluate. We say that *E* is *bootstrappable* with respect to ¡ if D_E (¡) *µ* C_E :For example, if ¡ includes the trivial gate and NAND, *E* is bootstrappable with respect to ¡ if C_E contains D_E and the circuit formed by joining two copies of D_E with a NAND gate. Remarkably, as we will show, if there is a scheme *E* that can compactly evaluate only these two circuits, then there is a scheme that is leveled fully homomorphic.[4]

We could relax the bootstrappability definition slightly to say that *E* only needs to be able to homomorphically evaluate its (augmented) decryption circuit when the input ciphertext is weakly encrypted, similar to the relaxation Recrypt^0_E above. But, this makes the definition of bootstrappable more cumbersome; we will continue with the definition above, and remind the reader occasionally that the relaxation can be used.

From the informal description above, it should already be somewhat clear how to use a bootstrappable encryption scheme to construct a leveled fully homomorphic one; below, we give a more formal description. Let *E* be bootstrappable with respect to a set of gates For any integer *d* ¸ 1, we use *E* to construct a scheme $E^{(d)}$ = ($\text{KeyGen}_E(d)$ *;* $\text{Encrypt}_E(d)$ *;* $\text{Evaluate}_E(d)$ *;* $\text{Decrypt}_E(d)$) that can handle all circuits of depth *d* with gates in ¡. Note that in the description below we encrypt the secret keys in reverse order; the only reason is that this ordering simplifies our description of the recursion in Evaluate. When we refer to the level of a wire in *C*, we mean the level of the gate for which the wire is an input. We use the notation D_E(¡; ±) to refer to the set of circuits that equal a ±-depth circuit with gates in ¡ augmented by D_E (copies of D_E become inputs to the ±-depth circuit).

$\text{KeyGen}_E(d)$ (¸*; d*). Takes as input a security parameter ¸ and a positive integer *d*. For '='(¸) as it sets*:*

R
$(sk_i; pk_i) \tilde{A} KeyGen_E(.)$ for $i\ 2\ [0; d]$

R
$sk_{ij} \tilde{A} Encrypt_E(pk_{ij1}\ ; sk_{ij})$ for $i\ 2\ [1; d]; j\ 2\ [1; `]$

where $sk_{i1}; : : : ; sk_{i`}$ is the representation of sk_i as elements of P. It outputs the secret key

$sk^{(d)} \tilde{A} sk_0$ and the public key $pk^{(d)} \tilde{A} (hpk_i; hsk_{ij}i)$. Let $E^{(\pm)}$ refer to the sub-system that uses $sk^{(\pm)} \tilde{A} sk_0$ and $pk^{(\pm)} \tilde{A} (hpk_i i_{i2[0;\pm]}; hsk_{ij} i_{i2[1;\pm]})$ for $\pm \cdot d$.

$Encrypt_E(d)$ $(pk^{(d)}; ¼)$. Takes as input a public key $pk^{(d)}$ and a plaintext $¼\ 2\ P$. It outputs a

R
ciphertext $\tilde{A} \tilde{A} Encrypt_E$ $(pk_d; ¼)$.

$Decrypt_E(d)$ $(sk^{(d)}; \tilde{A})$. Takes as input a secret key $sk^{(d)}$ and a ciphertext $\tilde{A}$ (which should be an encryption under pk_0). It outputs $Decrypt_E(sk_0; \tilde{A})$.

$Evaluate_E(\pm)$ $(pk^{(\pm)}; C_\pm; a_\pm)$. Takes as input a public key $pk^{(\pm)}$, a circuit $C_\pm$ of depth at most with gates in ¡, and a tuple of input ciphertexts $a_\pm$ (where each input ciphertext should be under $pk_\pm$). We assume that each wire in $C_\pm$ connects gates at consecutive levels; if not, add trivial gates to make it so. If $\pm = 0$, it outputs a_0 and terminates. Otherwise, it does the following:

- Sets $(C^y_{\pm i1}; a^y_{\pm i1}) \tilde{A} Augment_E(\pm)$ $(pk^{(\pm)}; C_\pm; a_\pm)$.
- Sets $(C_{\pm i1}; a_{\pm i1}) \tilde{A} Reduce_E(\pm i1)$ $(pk^{(\pm i1)}; C^y_{\pm i1}; a^y_{\pm i1})$.
- Runs $Evaluate_E(\pm i1)$ $(pk^{(\pm i1)}; C_{\pm i1}; a_{\pm i1})$.

$Augment_E(\pm)$ $(pk^{(\pm)}; C_\pm; a_\pm)$. Takes as input a public key $pk^{(\pm)}$, a circuit $C_\pm$ of depth at most $_\pm$ with gates in ¡, and a tuple of input ciphertexts $a_\pm$ (where each input ciphertext should be under $pk_\pm$). It augments $C_\pm$ with D_E ; call the resulting circuit $C^y_{\pm i1}$. Let $a^y_{\pm i1}$ be the tuple of ciphertexts formed by replacing each input ciphertext $\tilde{A}\ 2\ a_\pm$ by the tuple $hhsk_{\pm j}i; h\tilde{A}_j ii$, where $\tilde{A}_j$ $\tilde{A}$ $WeakEncrypt_E(\pm i1)$ $(pk^{(\pm i1)}; \tilde{A}_j)$ and the $\tilde{A}_j$'s form the properly-formatted representation of $\tilde{A}$ as elements of P. It outputs $(C^y_{\pm i1}; a^y_{\pm i1})$.

$Reduce_E(\pm)$ $(pk^{(\pm)}; C^y_\pm; a^y_\pm)$. Takes as input a public key $pk^{(\pm)}$, a tuple of input ciphertexts $a^y_\pm$ (where each input ciphertext should be in the image of $Encrypt_E(\pm)$), and a circuit $C^y_\pm\ 2\ D_E(¡; \pm + 1)$. It sets $C_\pm$ to be the sub-circuit of $C^y_\pm$ consisting of the first $\pm$ levels. It sets $a_\pm$ to be the induced input ciphertexts of $C_\pm$, where the ciphertext $\tilde{A}^{(w)}_\pm$ associated to wire w at level $\pm$ is set to $Evaluate_E$ $(pk_\pm; C^{(w)}_\pm; a^{(w)}_\pm)$, where $C^{(w)}_\pm$ is the sub-circuit of $C^y_\pm$ with output wire w, and $a^{(w)}_\pm$ are the input ciphertexts for $C^{(w)}_\pm$. It outputs $(C_\pm; a_\pm)$.

High-level review of the Evaluate algorithm. We are given a circuit C_d of, say, d levels with gates in ¡. For each input wire w of C_d, there is an associated input ciphertext $\tilde{A}_w$encrypted under pk_d. We are also given an encryption scheme E that compactly evaluates circuits in D_E (¡).

Note that we have not assumed that E can evaluate gates in ¡; we have only assumed it can evaluate gates in ¡ that are augmented by the decryption circuit. So, our first step is to augment C_d by placing copies of D_E at the leaves of C_d (as in Augment), thereby constructing $C_d{}^y{}_{i1}$. Now, what are the input ciphertexts for our new circuit $C_d{}^y{}_{i1}$? Reconsider the algorithm $Recrypt^0{}_E$. In $Recrypt^0{}_E$, we begin with a ciphertext $\tilde{A}_1$ encrypt-ing ¼ under pk_1 for the single wire w, and an empty" circuit C_1 (since $Recrypt^0{}_E$ doesn't actually evaluate any gates, it just generates a new encryption of the same plaintext). Our next step was to augment C_1 with the decryption circuit D_E to obtain C_2 $\tilde{A}$ D_E. The input ciphertexts $^a{}_2$ to C_2 included the encrypted secret key bits, and the weakly encrypted bits of $\tilde{A}_1$. We then showed that the ciphertext generated by $\tilde{A}_2$ $\tilde{A}$ $Evaluate_E(pk_2; C_2; {}^a{}_2)$, which is also associated to wire w, also encrypts ¼, but now under pk_2.

In the full scheme above, the ciphertexts that we associate to the decryption circuit that was attached to wire w are analogous to the ones we used in $Recrypt^0{}_E$: the encrypted secret key (sk_d under pk_{di1}), and the re-encryption ciphertexts of $\tilde{A}_w$ under pk_{di1}. By the correctness of Recrypt, the ciphertext *now* associated to w (after performing $Evaluate_E$) should encrypt the same plaintext as $\tilde{A}_w$, but now under pk_{di1}.

The Reduce step simply performs this Evaluate up to the wire w, and one level beyond. We know that Evaluate can correctly continue one level beyond the wire w, because (by assumption) E can evaluate not just the decryption circuit attached to w, but can evaluate a circuit containing one ¡-gate above w. Reduce outputs C_{di1} and ciphertexts associated to C_{di1}'s input wires. We have made progress, since C_{di1} is one level shallower than C_d. We perform this entire process d ¡ 1 more times to obtain the final output ciphertexts.

we said that Evaluate takes as input ciphertexts that are \fresh" outputs of Encrypt. However, we note $Evaluate_E(\pm)$ also operates correctly on ciphertexts output by Evaluate. (For $\pm < d$ above, this is precisely what $Evaluate_E(\pm)$ does.)

Correctness, Computational Complexity and Security of the Generic Construction

Here we state and prove some theorems regarding the generic construction. Regarding correctness, we have the following theorem.[5]

Let E be bootstrappable with respect to a set of gates ¡. Then $E^{(d)}$ com-pactly evaluates all circuits of depth d with gates in ¡ { i.e., if ¡ is a universal set of gates, the family $E^{(d)}$ is leveled fully homomorphic. Proof. (Theorem 4.2.1) First, we define a convenient notation: let $D(\pm; w; C; {}^a)$ denote the plaintext value for wire w in circuit C induced by the decryptions (under $sk_\pm$) of the ciphertexts ^{a} associated to C's input wires. If C is empty (has no gates), then the input wires are the

same as the output wires, and $D(\pm; w; C; {}^{a})$ just denotes the decryption of the single ciphertext $\tilde{A}$ 2 ^{a} associated to w. To prove correctness, it succes to show that

$$D(d; w_{out}; C_d; {}^{a}{}_d) = D(0; w_{out}; C_0; {}^{a}{}_0) \quad (1)$$

for every output wire w_{out} of C_0 (at level 0). First, when $(C_{\pm}{}^{y}{}_{i1}; {}^{ay}{}_{\pm i1})$ $\tilde{A}$ Augment$_E(\pm)$ $(pk^{(\pm)}; C_{\pm}; {}^{a}{}_{\pm})$, we claim that $D(\pm; w; C_{\pm}; {}^{a}{}_{\pm}) = D(\pm_i 1; w; C_{\pm}{}^{y}{}_{i1}; {}^{ay}{}_{\pm i1})$ for any wire w at level at most $\pm_i 1$. This follows from the correctness of Recrypt (generalized beyond a boolean plaintext space and boolean circuits), and the fact that circuits $C_{\pm}$ and $C_{\pm}{}^{y}{}_{i1}$ are identical up to level $\pm$ ¡ 1.

Next, when $(C_{\pm}; {}^{a}{}_{\pm})$ $\tilde{A}$ Reduce$_E(\pm)$ $(pk^{(\pm)}; C_{\pm}{}^{y}; {}^{ay}{}_{\pm})$, we have $D(\pm; w; C_{\pm}{}^{y}; {}^{ay}{}_{\pm}) = D(\pm; w; C_{\pm}; {}^{a}{}_{\pm})$ for any wire at level at most $\pm$. This follows from the correctness of Evaluate$_E$ over circuits in D (¡), and the fact that circuits C^y and C are identical up to level $\pm$.

E $\pm$ $\pm$

Note that ¡ is arbitrary. For example, each gate in ¡ could be a circuit of (ANDs, ORs, NOTs) of depth m and fan-in 2; for this value of ¡, It implies the scheme correctly evaluates boolean circuits up to depth d ¢ m.

We need to check that the computational complexity of Evaluate$_E(d)$ is reasonable { e.g., that recursive applications of Augment do not increase the effective circuit size exponentially.

For a circuit C of depth at most d and size s (defined here as the number of wires), the computational complexity of applying Evaluate$_E$(d) to C is dominated by at most s ¢ ` ¢ d applications of WeakEncrypt$_E$ and at most s ¢ d applications of Evaluate$_E$ to circuits in $D_E$ (¡), where ` is as in this research paper. Proof: There is a pre-processing step to ensure that all wires in the circuit connect gates at consecutive levels; clearly, this step increases the number of wires in the circuit by at most a multiplicative factor of d. It remains to prove that, for the pre-processed circuit, the computational complexity is dominated by at most s^0 ¢ ` applications of Encrypt and at most s^0 applications of Evaluate$_E$ to circuits in D_E (¡), where s^0 is the size of the pre-processed circuit. The complexity of Augment$_E(\pm)$ $(pk^{(\pm)}; C_{\pm}; {}^{a}{}_{\pm})$ is dominated by replacing each ciphertext A2 ${}^{a}{}_{\pm}$ by the ciphertexts hhsk$_{\pm j}$i; hÃ$_j$ii; generating the hÃ$_j$i's involves jW$_{\pm}$j ¢ ` applications of WeakEncrypt$_E$ , where W$_{\pm}$ is the set of wires at level $\pm$. Summing over all $\pm$, there are at most $s^0$ ¢ ` applications of WeakEncrypt$_E$.

□

The complexity of Reduce$_E(\pm)$ $(pk^{(\pm)}; C_{\pm}{}^{y}; {}^{ay}{}_{\pm})$ is dominated by the evaluation of $C_{\pm}{}^{(w)}$ for each w 2 $W_{\pm}$, which involves $jW_{\pm}j$ applications of Evaluate$_E$ to circuits in D_E (¡). Summing over all $\pm$, there are at most s^0 such applications. The theorem follows.

Finally, assuming the semantic security of E, we prove the semantic security of $E^{(d)}$. Theorem 4.2.3. *Let A be an algorithm that* $(t; {}^{2})$*-breaks the semantic security of* $E^{(d)}$*. Then, there is an algorithm B that* $(t^0; {}^{20})$*-breaks the*

semantic security of E for t^0 ¼ *t* ¢ ` *and* $^{20}$ ¸ $^2$=`(*d* + 1), *for* ` *as in. Proof.* (Theorem 4.2.3) Let (*E*)` be equivalent to *E*, but with plaintext space *P* `, where Encrypt$_{(E)}$` involves up to ` invocations of *E* and a concatenation of the results. We use a hybrid argument to show that *B* ($t^{00}$; $^{200}$)-breaks the semantic security of (*E*)` for t^{00} ¼ *t* and ^{200} ¸ ^{2}=(*d* + 1), from which the result follows.For *k* 2 [0*; d*], let Game *k* denote a game against $E^{(d)}$ in which everything is exactly as in the real-world game, except that for all *i* 2 [1*; k*] the challenger sets

$$(sk_i^0 ; pk_i^0) \tilde{A} KeyGen_E(¸) \quad \text{and} \quad sk_{ij} \tilde{A} Encrypt_E(pk_{i1} ; sk_{ij}^0)$$

In other words, for *i* 2 [1*; k*], sk_{ij} is the encryption (under pk_{i1}) of the *j*-th bit of a *random* secret key sk^0_i unrelated to sk_i. Game *d* + 1 is identical to Game *d*, except that the challenger ignores *b* and (¼$_0$; ¼$_1$), generates a random plaintext ¼ of the appropriate length, and encrypts ¼ to construct the challenge ciphertext. Let ^2_k denote the adversary's advantage in Game *k*.

Since Game 0 is identical to the real world attack, the adversary's advantage is ^{2} by assumption. Also, $^2_{d+1}$ = 0, since the challenge is independent of *b*. Consequently, for some *k* 2 [0*; d*], it must hold that *j*^2_k *¡* $^2_{k+1}$*j* ¸ ^{2}=(*d* + 1); ¯x this value of *k*. *B* uses *A* to break (*E*)` as follows. *B* receives from the challenger a public key pk. *B* generates the secret and public values exactly as in Game *k*, except that it replaces its *0* *0* R original value of pk_k with pk. Also, if *k* < *d*, it generates a dummy key pair (sk_{k+1}; pk_{k+1}) *Ã* KeyGen$_E$(¸), sets ¼$_0$ *Ã* sk_{k+1} and ¼$_1$ *Ã* sk^0_{k+1}, and requests a challenge ciphertext (under ` R pk) encrypting either ¼$_0$; ¼$_1$ 2 *P* . The challenger generates ¯ *Ã* *f*0; 1*g* and sends a tuple of ciphertexts *hÃ*$_j$*i* encrypting the bits *h*¼$_{¯j}$*i*. *B* replaces its original tuple *h*$sk_{(k+1)j}$*i* with the tuple *hÃ*$_j$*i*. One can verify that the public values are generated exactly as in Game *k* + ¯. *B* sends the public values to *A*. R Eventually, *A* requests a challenge ciphertext on ¼$_0$ or ¼$_1$. *B* sets *b Ã* *f*0; 1*g*. If *k* < *d*, R R *B* sends the values *Ã*$_j$ *Ã* Encrypt$_E$ (pk_d; ¼$_{bj}$). If *k* = *d*, *B* generates random ¼ *Ã* *P* and asks R the challenger for a challenge ciphertext on ¼$_b$ or ¼. The challenger generates ¯ *Ã* *f*0; 1*g* and encrypts ¼$_b$ or ¼ accordingly, and *B* forwards the challenge to *A*. *A* sends a bit b^0. *B* sends bit ¯^{0} *Ã* *b* © b^0 to the challenger. One can verify that the challenge is generated as in Game *k* + ¯. Since *B*'s simulation has the same distribution as Game *k* + ¯, and the probability that outputs 0 is $^2_{k+¯}$. The result follows.

Fully Homomorphic Encryption from KDM-Secure Boots trappable Encryption

The length of the public key in $E^{(d)}$ is proportional to *d* (the depth of the circuits that can be evaluated). It would be preferable to have a construction *E* ¤ where the public key size is completely independent of the circuit depth, a construction that is fully homomorphic rather than merely leveled fully homomorphic. Here is the obvious way to make the public key pk¤ of *E*¤ short: for *E* key pair (sk*;* pk), pk¤ includes only pk and (the bits" of) sk encrypted under pk. In other

words, we have a *cycle* (in fact, a *self-loop* in this example) of encrypted secret keys rather than an acyclic chain. It is clear that $E^{¤}$ is correct: the recursive algorithm Evaluate$_{E}$¤ works as before, except that the implicit recryptions generate \refreshed" ciphertexts under the same public key.

Why didn't we present this construction in the first place? Using an *acyclic* chain of encrypted secret keys allowed us to base the security of $E^{(d)}$ on E using a hybrid argument; this hybrid argument breaks down when there is a cycle. In general, a semantically secure encryption scheme is not guaranteed to be *KDM-secure* { i.e., secure when the adversary can request the encryptions of *key-dependent messages*, such as the secret key itself. Typ-ically, when we prove an encryption scheme semantically secure, there is not an obvious *attack* when the adversary is given the encryption of a key-dependent message.[7] However, KDM-security is highly nontrivial to prove. The problem is precisely that the usual hybrid argument breaks down.[6]

It proposed the acyclic, leveled approach as a way to remove the need for KDM-security. Our initial approach had actually been to use $E^{¤}$ (with the self-loop), and assume, or try to prove, KDM-security.Let us review (a restriction of) the definition of KDM-security. We will say a scheme E is KDM-secure if all polynomial-time adversaries A have negligible advantage in the following KDM-security game.

KDM-Security Game.

Setup(¸; n). The challenger sets (sk$_i$; pk$_i$) Ã KeyGen(¸) for $i \overset{R}{2}$ [0; n ¡ 1] for integer n = poly(¸). It chooses a random bit $b \overset{R}{Ã}$ f0; 1g. If b = 0, then for i 2 [0; n ¡ 1] and j 2 [1; `], it sets sk$_{ij}$ $\overset{R}{Ã}$ Encrypt$_E$ (pk$_{(i_j 1) \bmod n}$; sk$_{ij}$), where sk$_{ij}$ is the jth \bit" of sk$_i$. If b = 1, it generates the sk$_{ij}$ values as encryptions of random secret keys, unrelated to pk$_0$; : : : ; pk$_{n_j 1}$. It sends the public keys and encrypted secret keys to A.

Challenge and Guess. Basically as in the semantic security game.

This definition of KDM-security is a restriction of the general setting [18, 68, 22], where A can select multiple functions f, and request the encryption of f(sk$_0$; : : : ; sk$_{n_j 1}$). However, when E is a bootstrappable encryption scheme, A can use the cycle of encrypted secret keys in our game to generate the encryption of f(sk$_0$; : : : ; sk$_{n_j 1}$) under any pk$_i$, as long as f can be computed in polynomial time. Hence, we only need to consider our restricted setting [65]. We have the following theorem.

Suppose E is KDM-secure and also bootstrappable with respect to a uni-versal set of gates ¡. Then, E $^{¤}$, obtained from E as described above (with the self-loop), is semantically secure (and fully homomorphic).

The theorem is a straightforward consequence of the fact that, from *any* loop of public keys and

encrypted secret keys that includes (pk_0; sk_0), one can compute an encryption of sk_0 under pk_0. There does not seem to be any advantage in having pk¤ contain any cycle of encrypted secret keys other than a self-loop.

Absent proof of KDM-security in the plain model, one way to obtain fully homomorphic encryption from bootstrappable encryption is simply to *assume* that the underlying boot-strappable encryption scheme is also KDM-secure. This assumption, though unsatisfying, does not seem completely outlandish. While an encrypted secret key is very useful in a bootstrappable encryption scheme { indeed, one may view this as the essence of bootstrap-pability { we do not see any actual attack on a bootstrappable encryption scheme that provides a self-encrypted key.

Fully Homomorphic Encryption from Bootstrappable En-cryption in the Random Oracle Model

Above, we constructed a fully homomorphic encryption $E^{¤}$ from a bootstrappable encryp-tion scheme E basically by adding a self-loop" { a E secret key sk encrypted under its corresponding public key pk { to the $E^{¤}$ public key pk¤. We showed that $E^{¤}$ should inherit the semantic security of E, *under the assumption* that E is KDM-secure { in particular, under the assumption that it is safe" to reveal a direct encryption of a secret key un-der its own public key (as opposed to some possibly-less-revealing non-identity function of the secret key). Can we provide any evidence that $E^{¤}$ is semantically secure without this assumption?[9] Here we provide some evidence in the random oracle model. First, given a *leveled* fully homomorphic scheme $E^{(d)}$ and a hash function, we define an intermediate scheme $E^{(d)y}$. $E^{(d)y}$ is the same as $E^{(d)}$, except for the following. The public key includes a hash function

(*d*

0 ` R *0* R)

$H : P \; ! \; P$. Also, in KeyGen, one generates $r \tilde{A} P$, sets $r_j \tilde{A} \text{Encrypt}_E(d)(pk ; r_j)$ for j [1; *0*], sets ¾ $H(r)$ *?* sk_0 , and includes (r_j ; ¾) in the public key. (Assume *?* is

2 $\tilde{A}$ $h—i$

some invertible operation such that $a ? b$ would completely hide $b \; 2 \; P$ if $a \; 2 \; P$ were a one-time pad.) In other words, the $E^{(d)y}$ public key includes some additional information: an encryption of the the secret key sk_0, where the encryption uses a hash function that will be treated as a random oracle in the security analysis.

Next, we prove the following theorems.

If $E^{(d)}$ is semantically secure, then $E^{(d)y}$ is semantically secure in the random oracle model.

Theorem 4.4.2. Suppose E is leveled circuit private (in addition to being bootstrappable) and let $E^{(d)y}$ and E¤ be constructed from E as described above.

International Journal of Research
e-ISSN: 2348-6848 & p-ISSN 2348-795X Vol-5, Special Issue-11
International Conference on Multi-Disciplinary Research - 2017 held in February, 2018 in Hyderabad, Telangana State, India organised by GLOBAL RESEARCH ACADEMY - Scientific & Industrial Research Organisation (Autonomous), Hyderabad.

Then, if $E^{(d)\ y}$ is semantically secure (in the plain model), and the circuit required to compute the hash function H and invert the ? operation is at most d levels, then E¤ is semantically secure.

The result here should be quite surprising. The scheme E¤ does not even contain a hash function, and yet we are basically claiming that it is secure in the random oracle model! This is the first instance that we are aware of where one scheme is proven secure in the random oracle model, and then a second scheme's security is based on the first scheme, even though the second scheme does not use a hash function. How is this possible? First, let us consider in this research paper. This

theorem basically just states the previously known result that it is easy to construct a KDM-secure encryption scheme in the random oracle model. This is because the random oracle allows the reduction to construct a fake" ciphertext purportedly encrypting the secret key, such that the adversary finds out that it was fake only after it has queried the random oracle; this query gives the reduction all of the information that it needs to solve the underlying problem. In our particular case, $E^{(d)y}$ has a loop among (sk_0; pk_0)*; : : : ;* (sk_d; pk_d), because $E^{(d)}$ reveals direct encryptions of sk_i under $pk_{i¡1}$ for *i 2* [1*; d*], and $E^{(d)y}$ also reveals an *indirect* encryption (*hrji; ¾*) of sk_0 under pk_d (\indirect," because encryption in *E* does not normally use a hash function). However, the reduction algorithm in the proof of Theorem 4.4.1 will construct ¾ simply as a random string { i.e., it does not actually need to know anything about sk_0.

It perhaps the more surprising result. But the result is actually a simple consequence of the fact that: given a correctly constructed $E^{(d)\ y}$ public key, the reduction algorithm can generate an *E*-encryption of sk_0 under pk_0, as needed for the *E*¤ public key. How do we generate the latter ciphertext? The reduction algorithm is given *hrji*, an encryption of *r* under pk_d. It simply uses the leveled homomorphism and the circuit corresponding to the hash function *H* to compute a ciphertext that encrypts *H*(*r*) from the ciphertext that encrypts *r*. Then, given that ciphertext and the value of ¾ = *H*(*r*) *?* sk_0, it computes a ciphertext that encrypts sk_0 *in the natural way* { i.e., directly, rather than with the hash function. We assumed that the hash function *H* and the *?* operation can be computed with a circuit of depth at most *d*; therefore, our leveled homomorphic scheme $E^{(d)}$ has enough levels to evaluate this circuit. Consequently, we obtain a \natural" encryption of sk_0 (i.e., under *E*) under some public key pk_i for *i* ¸ 0, and we can use Recrypt operations to obtain a natural encryption of sk_0 under pk_0. This ciphertext is an output of $Evaluate_E$, but circuit privacy guarantees that the ciphertext is distributed as if it were output directly by $Encrypt_E$.

Although one can view (*hrji; ¾*) as an encryption" of sk_0, this encryption" function is not the usual encryption function and it might have a very complex decryption circuit, much more complex than D_E . In particular, we cannot assume that its decryption circuit is in C_E. This why we needed many (*d*) levels in the leveled scheme to recover sk_0, and could not immediately use a self-loop from the outset.[10]

So, if E^{α} is secure in the random oracle model despite not using a hash function, does that imply that it is secure in the plain model? Certainly not. The obstacle to this conclusion is obviously that random oracles cannot be instantiated in general. A bit more specifically, however, the obstacle is that the proof of Theorem 4.4.2 depends crucially on the correctness of the ciphertext ($h\#_j i$; ¾) in $E^{(d)}_{y}$ to construct (homomorphically) an encryption of sk_0 under pk_0 as needed for the E^{α} public key; however, in the proof of the ciphertext is not correct (except with negligible probability): the adversary funds out that it was fake only after it has queried r to the random oracle, giving the reduction all the information it needs.

Proof : Let A be an algorithm that attacks the semantic security of $E^{(d)y}$; from A, we construct an algorithm B that attacks the semantic security of $E^{(d)}$. B will actually request $`^0$ + 1 challenge ciphertexts; thus, the reduction loses a factor of $`^0$ + 1 under the usual hybrid argument.

The challenger gives B a $_E(d)$ public key. It also sets a bit b Ã $f0; 1g$. B selects two messages $r^{(0)}; r^{(1)}$ 2 $P^{`^0}$ and sends them to the challenger. The challenger sets $^{a}$ Ã Encrypt(pk ; $r^{(b)}$) : $j$ [1; $`^0$] and sends back ^{a}. The following is included in the public key that B sends to A: the public key for $E^{(d)}$ sent by the challenger, the set of ciphertexts ^{a}, and ¾ Ã P .

A requests a challenge ciphertext on one $¼_0; ¼_1$ 2 P. B forwards the query to the challenger, who responds with a ciphertext encrypting $¼_b$, which B forwards to A. Eventually, either A queries some r^0 2 $fr^{(0)}; r^{(1)}g$ to the random oracle, or A finishes with a guess b^0. In the former case, B sets b^0 so that $r^0 = r^{(b0)}$. In either case, B sends b^0 as its guess to the challenger.

Let p be the probability that A queries some r^0 2 $fr^{(0)}; r^{(1)}g$ to the random oracle. B's simulation appears perfect to A if it does not query some r^0 2 $fr^{(0)}; r^{(1)}$ g; in this case, which occurs with probability 1 ¡ p, A's advantage is at least ². Since A's view is independent of $r^{(1¡b)}$, the probability that it queries $r^{(b)}$ to the random oracle is at least p ¡ q_H =$jPj^{`^0}$, where $q_H$ is the number of random oracle queries make by $A$. Overall $B$'s advantage in guessing $b^0$ is at least $(1 ¡ p)^2$ + $p$ ¡ $q_H$ =$jPj^{`^0}$ ¸ ² ¡ q_H =$jPj^{`^0}$.

Proof: The proof is essentially a simple consequence of the fact that, given a public key for $E^{(d)y}$, it is easy to generate the public key for E^{α} homomorphically.

Let A be an algorithm that breaks the semantic security of E^{α}. We use A to construct an algorithm B that breaks the semantic security of $E^{(d)y}$.

B receives a $E^{(d)y}$ public key from the challenger. This public key consists of $hpk_i i_{i2[0;±]}$, $hsk_{ij} i_{i2[1;±]}$, $h\#_j i_{j2[1;`^0]}$, and ¾ = $H(r)$? sk_0. From $h\#_j i$, B uses the homomorphism of $E^{(d)}$ to compute ciphertexts ^{a} that encrypt $H(r)$. It encrypts ¾,

and then uses the homomorphism to recover to obtain an encryption of sk_0 from the encryptions of $H(r)$ and ¾ (inverting the *?* operation). By assumption, these homomorphic operations take at most d levels. If it takes only $\pm < d$ levels, and we obtain an encryption of sk_0 under $pk_{d_i \pm}$, then we can perform Recrypt operations until we have the desired encryption of sk_0 under pk_0. By circuit privacy, this ciphertext is distributed properly. *B* includes the encryption of sk_0 under pk_0 as the encrypted secret key contained in the public key for E^{α} that it provides to *A*.

A requests a challenge ciphertext on one $¼_0; ¼_1 \; 2 \; P$. *B* forwards the query to the challenger, who responds with a ciphertext encrypting $¼_b$. *B* uses Recrypt operations to obtain an encryption of $¼_b$ under pk_0 and forwards the result to *A*. *A* sends a guess b^0, which *B* forwards to the challenger.

Clearly, *B*'s advantage is the same as *A*'s.

□

An Abstract Scheme Based on the Ideal Coset Problem

Our goal now is to construct a bootstrappable encryption scheme, a scheme that can ho-momorphically evaluate a rich set of circuits that includes its own decryption circuit, plus some." In the past, attempts to construct fully homomorphic encryption have focused solely on *maximizing* the complexity of the circuits that the scheme can evaluate. Our notion of bootstrapability gives us a different way of attacking the problem { by *minimizing* the complexity of the scheme's decryption circuit.

Our strategy for minimizing the circuit complexity of decryption is to construct our scheme using *ideal lattices*, since decryption in lattice-based cryptosystems is typically dom-inated by a simple operation, such as an easily parallelizable matrix-vector multiplication (in contrast to, say, RSA, where decryption involves exponentiation, an operation not even known to be in NC). We begin describing the ideal-lattice-based scheme in Chapter 7, after providing some basic background on ideal lattices in Chapter 6.

In this Chapter, we describe our strategy for maximizing the evaluative capacity"[11] of the scheme abstractly, without reference to lattices. Generally speaking, our exposition strategy throughout the paper is to defer technical lattice details for as long as possible. One reason is to make the presentation more modular, and therefore easier to understand. Another reason is that some of our techniques { e.g., bootstrapping, and using techniques from server-aided cryptography to squash the decryption circuit" { maybe applicable to schemes that use different underlying mathematics { e.g., linear codes, or something less similar to lattices.

The Ideal Coset Problem

We saw in this research paper that many previous homomorphic encryption schemes base security on some *ideal membership problem (IMP)*. For example, in the Polly Cracker" scheme by Fellows and Koblitz, the public key consists of some multivariate polynomials that generate the ideal *I* of polynomials having a common root x, and ¼ is encrypted by outputting a sample *Ã Ã* ¼ + *I*. One can easily see that this is semantically secure if it is hard to distinguish membership in *I* { in particular, deciding whether *Ã* ¡¼ *2 I*. Unfortunately, one can also see that homomorphic operations, especially multiplication, expand the ciphertext size potentially exponentially in the depth.

Since we will ultimately use lattices, we apparently need a different abstract approach, since it is easy to distinguish membership in a lattice *L*: given a basis B of *L* and t *2* R^n, one simply determines whether t mod B = 0 mod *B*. Instead, we base security on an *ideal coset problem (ICP)*, which we will state abstractly in terms of rings and ideals. Recall that a *ring R* is an algebraic object that is closed under addition `+' and multiplication `£' and additive inverse, with an additive identity `0' and multiplicative identity `1'. An *ideal I* of a ring *R* is a subset satisfying *a + b 2 I* and *r £ a 2 I* for all *a; b 2 I* and *r 2 R*. The sum and product of two ideals *I* and *J* are, respectively, *fi + j : i 2 I; j 2 Jg* and the additive closure of *fi £ j : i 2 I; j 2 Jg*. Two ideals *I* and *J* are *relatively prime* if *I + J = R*. For

example, if R = Z, the ideals (2) (the even integers) and (5) (the integers divisible by 5) are relatively prime: (2) + (5) = (1).

Now, the ideal coset problem (ICP) is as follows.

Definition 5.1.1 (Ideal Coset Problem (ICP)). Fix R, B_I , algorithm IdealGen, and an algorithm $Samp_1$ that e±ciently samples R. The challenger sets b Ã $f0; 1g$ and $(B_J^{sk}; B_J^{pk}) Ã$ IdealGen($R; B_I$). If b = 0, it sets r Ã $Samp_1^R$ (R) and t Ã r mod B_J . If b = 1, it samples t uniformly from R mod B_J^{pk}. The problem: guess b given (t; B_J^{pk}).

Basically the ICP asks one to decide whether t is uniform modulo J, or whether it was chosen according to a known clumpier" distribution induced by $Samp_1$. Of course, the ICP will be impossible if $Samp_1$ also samples uniformly modulo J, but the security of our encryption scheme will rely on the ICP being hard for a clumpier" instantiation of $Samp_1$; the hardness of the problem depends on the particular instantiation of $Samp_1$. Note that it is possible for the ICP to be hard even when the IMP is easy.

An Abstract Scheme

We start by describing our initial attempt simply in terms of rings and ideals; we bring in ideal lattices later. In our initial scheme E, we use a fixed ring R that is set appropriately according to a security parameter ¸. We also use a fixed basis B_I of a ideal I ½ R, and an algorithm IdealGen($R; B_I$) that outputs public and secret bases B^{pk}_J and B^{sk}_J of some (variable) ideal J, such that $I + J = R$ { i.e., I and J are relatively prime. We assume that if t 2 R and B_M is a basis for ideal M ½ R, then the value t mod B_M is unique and can be computed efficiently { i.e., the coset t + M has a unique, efficiently-computable \distinguished representative" with respect to the basis B_M . We use the notation R mod B_M to denote the set of distinguished representatives of $r + M$ over r 2 R, with respect to the particular basis B_M of M. We also use an algorithm Samp(B_I ; x) that samples from the coset x + I.

In the scheme, Evaluate takes as input a circuit C whose gates perform operations modulo B_I . For example, an Add_{BI} gate in C takes two terms in R mod B_I , and outputs a third term in R mod B_I , which equals the sum of the first two terms modulo I.

KeyGen($R; B_I$). Takes as input a ring R and basis B_I of I. It sets (B^{sk}_J ; B^{pk}_J) Ã IdealGen($R; B_I$). The plaintext space P is (a subset of) R mod B_I . The public key pk includes R, B_I , B^{pk}_J, and Samp. The secret key sk also includes B^{sk}_J.

Encrypt(pk; ¼). Takes as input the public key pk and plaintext ¼ 2 P. It sets $Ã^0$ Ã Samp(B_I ; ¼) and outputs $Ã$ Ã $Ã^0$ mod B^{pk}_J. Decrypt(sk; $Ã$). Takes as input the secret key sk and a ciphertext $Ã$. It outputs

$$¼ Ã (Ã \bmod B^{sk}_J) \bmod B_I$$

Evaluate(pk; C; ª). Takes as input the public key pk, a circuit C in some permitted set C_E of circuits composed of Add_{BI} and Mult_{BI} gates and a set of input ciphertexts ª. It invokes Add and Mult, given below, in the proper sequence to compute the output ciphertext $\tilde{A}$. (We will describe C_E when we consider correctness below. If desired, one could use di®erent arithmetic gates.)

Add(pk; $\tilde{A}_1$; $\tilde{A}_2$). Outputs $\tilde{A}_1 + \tilde{A}_2 \bmod B^{pk}_J$.

Mult(pk; $\tilde{A}_1$; $\tilde{A}_2$). Outputs $\tilde{A}_1$ £ $\tilde{A}_2 \bmod B^{pk}_J$.

Concerning IdealGen, it is define the secret basis B^{sk}_J de¯nes a lattice $L(B^{sk}_J)$ for a (possibly fractional) ideal that *contains* J, rather than being exactly J.

Now, let us consider correctness, which is a highly nontrivial issue in this paper. The following definitions provide structure for our analysis.

To begin, we observe that the scheme is actually using two di®erent circuits. First, Evaluate takes a mod-B_I circuit C as input. This circuit is implicitly applied to plaintexts. Second, Evaluate applies a circuit related to C, which we call the *generalized circuit*, to the ciphertexts; this circuit uses the ring operations (not modulo I).[12]

Let C be a mod-B_I circuit. We say generalized circuit $g(C)$ of C is the circuit formed by replacing C's Add_{BI} and Mult_{BI} operations with addition `+' and multiplication `£' in the ring R. Here are a few more definitions relevant to below, which concerns correctness. (X_{Enc} and X_{Dec}). Let X_{Enc} be the image of Samp. Notice that all ciphertexts output by Encrypt are in X_{Enc} +J. Let X_{Dec} equal $R \bmod B^{sk}_J$, the set of distinguished representatives of cosets of J wrt the secret basis B^{sk}_J.

Definition 5.2.4 (Permitted Circuits). Let

$$C_E^{\,0} = fC : 8(x_1; : : : ; x_t)\ 2\ X_{\text{Enc}}^{\,t};\ g(C)(x_1; : : : ; x_t)\ 2\ X_{\text{Dec}}g$$

In other words, $C_E^{\,0}$ is the set of mod-B_I circuits that, when generalized, the output is always in X_{Dec} if the inputs are in X_{Enc}. (The value t will of course depend on C.) If C_E μ $C_E^{\,0}$, we say that C_E is a set of permitted circuits.

$\tilde{A}$ is a *valid ciphertext* wrt E public key pk and permitted circuits C_E if it equals Evaluate(pk; C; ª) for some C 2 C_E, where each $\tilde{A}$ 2 ª is in the image of Encrypt. The circuit C may be the identity circuit, in which case the output of Evaluate is simply an output of Encrypt.

Finally, we prove correctness with respect to C_E . Theorem 5.2.6. Assume C_E is a set of permitted circuits containing the identity circuit. E is correct for C_E { i.e., Decrypt correctly decrypts valid ciphertexts. CHAPTER 5. AN ABSTRACT SCHEME BASED ON THE IDEAL COSET PROBLEM61

Proof. For ciphertexts $^{a} = f\tilde{A}_1; : : : ; \tilde{A}_t g$, $\tilde{A}_k = ¼_k + i_k + j_k$, where $¼_k$ *2* P, i_k *2* I, j_k *2* J, and $¼_k + i_k$ *2* X_{Enc}, we have

$$\text{Evaluate}(\text{pk}; C; {}^{a}) = g(C)({}^{a}) \bmod \text{B}^{\text{pk}}{}_J \; 2 \; g(C)(¼_1 + i_1; : : : ; ¼_t + i_t) + J$$

If C *2* C_E, we have $g(C)(X_{\text{Enc}}; : : : ; X_{\text{Enc}})$ *2* X_{Dec} and therefore

$$\begin{aligned} \text{Decrypt}(\text{sk}; \text{Evaluate}(\text{pk}; C; {}^{a})) &= g(C)(¼_1 + i_1; : : : ; ¼_t + i_t) \bmod \text{B}_I \\ &= g(C)(¼_1; : : : ; ¼_t) \bmod \text{B}_I \\ &= C(¼_1; : : : ; ¼_t) \end{aligned}$$

as required.

□

The bottom line is that we have proven that E is correct for permitted circuits, and our goal now is to maximize this set. The permitted circuits are defined somewhat indirectly; they are the circuits for which the error" $g(C)(x_1; : : : ; x_t)$ of the output ciphertext is small (i.e., lies inside X_{Dec}) when the input ciphertexts are in the image of Encrypt_E. When we begin to instantiate the abstract scheme with lattices and give geometric interpretations of X_{Enc} and X_{Dec}, the problem of maximizing C_E will have a geometric °avor.

Again, we note the rather confusing fact that C automatically" reduces the result modulo B_I , since it uses mod-B_I gates. It does not particularly matter how these mod-B_I gates are implemented; in particular, it is more confusing than helpful to imagine a boolean implementation of these gates. Instead, one should just observe that the generalized circuit manages to lazily emulate these gates, reducing its output modulo B_I at the end of the computation. C's mod-B_I operations are never actually implemented;" they only occur implicitly. Later, when we consider whether our scheme is bootstrappable, and analyze the depth of the decryption circuit in terms of mod-B_I gates, it will again be tempting to consider how these gates are \implemented." But in fact these gates are given" in the sense that they are emulated (without any intermediate reduction steps) by the usual ring operations.

Security of the Abstract Scheme

For the following abstract instantiation" of Samp, and where I is a principle ideal generated by some s *2* R (and s is encoded in B_I), we provide a simple proof of semantic security based on the ICP. $\text{Samp}(\text{B}_I ;$ x). Run r $\tilde{A}$ $\text{Samp}_1(R)$. Output x + r £ s. Obviously, the output is in x + I since s *2* I.

Suppose that there is an algorithm A that breaks the semantic security of E with advantage ^{2} *when it uses* Samp. *Then, there is an algorithm B, running in about the same time as A, that solves the ICP with advantage* ^{2}=2.

Proof. The challenger sends B a ICP instance (t*;* $\text{B}^{\text{pk}}{}_J$). B sets s, and sets the other compo-nents of pk in the obvious way using the ICP instance. When A requests a challenge cipher- text on one of

¼0; ¼1 2 P, *B* sets a bit ¯ Ã f0; 1*g* and sends back Ã Ã ¼– + t £ s mod B*J* . A sends back a guess ¯0, and *B* guesses *b*0 Ã ¯ © ¯0.

If *b* = 0, we claim that *B*'s simulation is perfect; in particular, the challenge ciphertext has the correct distribution. When *b* = 0, we have that t = r + j, where r was chosen according to Samp1 and j 2 *J*. So, Ã Ã ¼– + t £ s = ¼– + r £ s mod B$^{pk}_{J}$; the ciphertext is thus well-formed. In this case *A* should have advantage ², which translates into an advantage of ² for *B*. If *b* = 1, then t is uniformly random modulo *J*. Since the ideal *I* = (s) is relatively prime to *J*, t£s is uniformly random modulo *J*, and consequently Ã is a uniformly random element of *R* mod B$^{pk}_{J}$ that is independent of ¯. In this case *A*'s advantage is 0. Overall, *B*'s advantage is ²=2.

References:

1. M. Bellare, A. Boldyreva, and S. Micali. Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In *Proc. of Eurocrypt '00*, pages 259{274. Springer, 2000.

2. J. Benaloh. Veri¯able secret-ballot elections. Ph.D. thesis, Yale Univ., Dept. of Comp. Sci., 1988.

3. J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *Proc. of SAC '02*, LNCS 2595, pages 62{75. Springer, 2002.

4. M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. *Eurocrypt '98*, LNCS 1403, pp. 127{144.

5. D. Boneh and M. Franklin. E±cient Generation of Shared RSA Keys. J. ACM, vol. 48, no. 4. Pages, 702-722. ACM, 2001. Preliminary version in Crypto 1997.

6. D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. *TCC '05*, LNCS 3378, pp. 325{341.

7. D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-Secure Encryption from Decision Di±e-Hellman. In *Proc. of Crypto '08*, LNCS 5157, pages 108{125.

8. D. Boneh and R. Lipton. Searching for Elements in Black-Box Fields and Applications. In *Proc of Crypto '96*, LNCS 1109, pages 283{297. Springer, 1996.

9. J. Boyar, R. Peralta, and D. Pochuev. On the Multiplicative Complexity of Boolean Functions over the Basis (^; ©; 1). Theor. Comput. Sci. 235(1), pp. 43{57, 2000.

10. E. Brickell and Y. Yacobi. On Privacy Homomorphisms. In *Proc. of Eurocrypt '87*, LNCS 304, pages 117{125. Springer, 1988.

11. J.Y. Cai and A. Nerurkar. An improved worst-case to average-case connection for lattice problems. In *Proc. of FOCS '97*, pages 468{477.

12. R. Canetti. Personal communication, 2008.

13. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proc. of STOC '98*, pages 209{218. ACM, 1998.

14. R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proc. of ACM CCS '07*.

15. R. Canetti, H. Krawczyk, and J.B. Nielsen. Relaxing chosen-ciphertext security. In *Proc. of Crypto '03*, pages 565{582. Springer, 2003.