INTERNATIONAL JOURNAL OF CLOUD COMPUTING TECHNOLOGICAL IDEAS (IJCCTI)
ISSN:0974-8656 / http://satyanspublications.weebly.com/ijccti.html /
VOL 2-NO1 PP 100-105 (2012)

Sreenivas Velagapudi, BN Goswami
Research Scholar,Professor
Department of Computer Science
Jiwaji University
Gwalior,M.P, India.

Cloud Bootstraple Encryption Schemes

Abstract

Assume we have an encryption scheme ε that compactly evaluates some set of cycles *C*ε . We want to use ε to construct a automorphic encryption scheme that can handle arbitrary cycles. In this Chapter we prove a fundamental result: that if Cε contains (slight augmentations of) ε's own decryption cycle Dε – i.e., if ε "compactly evaluates" its (augmented) decryption cycle – then we can use ε to construct an efficient scheme that handles cycles of arbitrary depth.

A bit more specifically, for any integer *d*, we use ε to construct a scheme $\varepsilon^{(d)}$ that can compactly evaluate cycles of depth up to *d*. The decryption cycle for $\varepsilon^{(d)}$ is still Dε ; the secret key and ciphertexts are the same size as in ε. The public key in $\varepsilon^{(d)}$ consists of $d + 1$ public keys from ε, together with a chain of encrypted ε secret keys – the first ε secret key encrypted under the second ε public key, and so on. In short, the family of schemes $\{\varepsilon^{(d)}\}$ is leveled automorphic. We base the semantic security of $\varepsilon^{(d)}$ on that of ε using a hybrid argument; as usual with hybrid arguments, the reduction loses a factor linear in *d*. We describe how one can obtain a automorphic 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 ε secret key under its associated public key.

Keywords : Cloud Bootstrapple, Ideal Lattices, Random Scheme, Encryption

Introduction

Automorphically evaluating the decryption cycle 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 cycle.

To see how this works mathematically, begin by considering the following algorithm, called Recrypt. For simplicity, suppose the plaintext space P is {0, 1} and $D\varepsilon$ is a boolean cycle in $C\varepsilon$. Let (sk_1, pk_1) and (sk_2, pk_2) be two ε key-pairs. Let ψ_1 be an encryption of $\phi \in P$ under pk_1 . Let $\overline{sk_{1j}}$ 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 ϕ under pk_2.

Recrypt(pk_2, Dε, $\left\langle \overline{sk_{1j}} \right\rangle$, ψ_1).

Set $\overline{\psi_{1j}} \xleftarrow{R}$ *Encrypt*ε (pk_2, ψ_{1j}) where ψ_{1j} is the j-th bit of ψ_1

Set $\psi_2 \xleftarrow{R}$ *Evaluate*ε $\left(pk_2, D\varepsilon, \left\langle \left\langle \overline{sk_{1j}} \right\rangle \left\langle \overline{\psi_{1j}} \right\rangle \right\rangle \right)$

Output ψ_2

Above, the *Evaluate* algorithm takes in all of the bits of sk1 and all of the bits of ψ_1, each encrypted under pk_2. Then, ε is used to evaluate the decryption cycle automorphically. The output ψ_2 is thus an encryption under pk_2 of *Decrypt*ε $(sk_1, \psi_1) \rightarrow \phi$.

Methodology

Here we state and prove some theorems regarding the generic erection. Regarding correctness, we have the following theorem.

Theorem Let ε be bootstrappable with respect to a set of gates Γ. Then $\varepsilon^{(d)}$ compactly evaluates all cycles of depth d with gates in Γ– i.e., if $\Gamma$is a universal set of gates, the family $\varepsilon^{(d)}$ is leveled automorphic.

Proof. First, we define a convenient notation: let $D(\delta, w, C, \psi)$ denote the plaintext value for wire w in cycle C induced by the decryptions (under sk_δ) of the ciphertexts ψ 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 $D(\delta, w, C, \psi)$ just denotes the decryption of the single ciphertext $\psi \in \Psi$ associated to w. To prove correctness, it suffices to show that $D(d, w_{out}, C_d, \psi_d) = D(0, w_{out}, C_0, \psi_0)$ for every output wire out of C_0 (at level 0).

First, when $C^{\dagger}_{\delta-1}, \psi^{\dagger}_{\delta-1} \rightarrow$ *Augment*$\varepsilon^{(\delta)}$ $(pk^{(\delta)}, C_\delta, \psi_\delta)$, we claim that
$D(\delta, w, C_\delta, \psi) = D(\delta-1, w,)$ for any wire w at level at most $\delta-1$. This follows from the correctness of *Recrypt* (generalized beyond a boolean plaintext space and boolean cycles), and the fact that cycles C_δ and are identical up to level $\delta-1$.

Next, when $(C_\delta, \psi_\delta) \rightarrow$ *Reduce*$\varepsilon^{(\delta)}$$(pk^{(\delta)},)$, we have $D(\delta, w,) = D(\delta, w, C_\delta, \psi)$ for any wire at level at most δ. This follows from the correctness of *Evaluate*ε over cycles in $D\varepsilon(\Gamma)$, and the fact that cycles and C_δ are identical up to level δ. From these two claims, Equation follows.

Note that Γ is arbitrary. For example, each gate in Γ could be a cycle of (ANDs, ORs, NOTs) of depth m and fan-in 2; for this value of Γ, Theorem 4.2.1 implies the scheme correctly evaluates boolean cycles up to depth $d{\cdot}m$.

We need to check that the computational complexity of *Evaluate*$\varepsilon^{(d)}$ is reasonable – e.g., that recursive applications of Augment do not increase the effective cycle size exponentially.

Cloud Automorpic Encryption (CAE) from KDM-Cloud Secure Bootstrappable Encryption

The length of the public key in $\varepsilon^{(d)}$ is proportional to d (the depth of the cycles that can be evaluated). It would be preferable to have a erection ε^* where the public key size is completely independent of the cycle depth, a erection that is automorphic rather than merely leveled automorphic. Here is the obvious way to make the public key pk* of ε^* short: for ε key pair (s_k, p_k), pk* includes only p_k and (the "bits" of) s_k encrypted under p_k. 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 ε^* is correct: the recursive algorithm *Evaluate*ε^* works as before, except that the implicit recryptions generate "refreshed" ciphertexts under the same public key.

Let us review (a restriction of) the definition of KDM-security. We will say a scheme ε is KDM-secure if all polynomial-time adversaries A have negligible advantage in the following KDM-security game.

KDM-Security Game.

Setup(λ, n). The challenger sets (sk_i, pk_i) $\xrightarrow{R}$ *KeyGen*(λ) for i $\in$ [0, n – 1] for integer n = poly(λ). It chooses a random bit b {0, 1}. If b = 0, then for i $\in$ [0, n – 1] and $j \in$ [1,], it sets $\overline{sk_{ij}}$ $\xrightarrow{R}$ *Encrypt*ε ($pk_{(i-1) \bmod n}$, sk_{ij}), where sk_{ij} is the jth "bit" of sk_i. If b = 1, it generates the $\overline{sk_{ij}}$ values as encryptions of random secret keys, unrelated to pk_0 ,..., pk_{n-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-1}). However, when ε 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-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.

Theorem Suppose ε is KDM-secure and also bootstrappable with respect to a universal set of gates Γ. Then, ε^*, obtained from ε as described above (with the self-loop), is semantically secure (and automorphic).

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 automorphic encryption from bootstrappable encryption is simply to assume that the underlying bootstrappable 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 bootstrappability – we do not see any actual attack on a bootstrappable encryption scheme that provides a self-encrypted key.

Cloud Ideal Coset Problem on Random Scheme

Our goal now is to construct a bootstrappable encryption scheme, a scheme that can automorphically evaluate a rich set of cycles that includes its own decryption cycle, "plus some." In the past, attempts to construct cloud automorphic encryption scheme have focused solely on *maximizing* the complexity of the cycles 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 cycle.

Our strategy for minimizing the cycle complexity of decryption is to construct our scheme using ideal lattices, since decryption in lattice-based cryptosystems is typically dominated 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, after providing some basic background on ideal lattices.

In this Chapter, we describe our strategy for maximizing the "evaluative capacity" 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 cycle" – maybe applicable to schemes that use different underlying mathematics – e.g., linear codes, or something less similar to lattices.

Cloud Ideal Coset Problem

We saw that many previous automorphic 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 ϕ is encrypted by outputting a sample $\psi \xrightarrow{R} \phi + I$. One can easily see that this is semantically secure if it is hard to distinguish membership in I – in particular, deciding whether $\psi - \phi \in I$. Unfortunately, one can also see that automorphic 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 $t \in □^n$, 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 '$\times$' and additive inverse, with an additive identity '0' and multiplicative identity '1'. An *ideal I* of a ring R is a subset satisfying $a + b \in I$ and $r \times a \in I$ for all $a, b \in I$ and $r \in R$. The sum and product of two ideals I and J are, respectively, $\{i + j : i \in I, j \in J\}$ and the additive closure of $\{i \times j : i \in I, j \in J\}$. Two ideals I and J are relatively prime if $I + J = R$. For example, if $R = □$, 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 (Ideal Coset Problem (ICP)). Fix R, B_I, algorithm *IdealGen*, and an algorithm $Samp_1$ that efficiently samples R. The challenger sets b $\xrightarrow{R}$ {0, 1} and $\left(B_J^{sk}, B_J^{pk}\right)$ *IdealGen*(R, B_I). If b = 0, it sets $r \xrightarrow{R} Samp_1(R)$ and $t \rightarrow r \bmod B_J^{pk}$. 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.

Random 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 ε, we use a fixed ring R that is set appropriately according to a security parameter λ. We also use a fixed basis B_I of a ideal $I \subset R$, and an algorithm *IdealGen*(R, B_I) that outputs public and secret bases B_J^{pk} and B_J^{sk} of some (variable) ideal J, such that $I + J = R$ – i.e., I and J are relatively prime. We assume that if $t \in R$ and B_M is a basis for ideal $M \subset R$, 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 $r \in R$, 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 cycle 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_J^{sk}, B_J^{pk}) $\underset{\to}{R}$ *IdealGen*(R, B_I). The plaintext space P is (a subset of) R mod B_I. The public key p_k includes R, B_I, B_J^{pk} and *Samp*. The secret key sk also includes B_J^{pk}.

Encrypt(p_k, π). Takes as input the public key pk and plaintext $\phi \in P$. It sets

$\psi' \to$ *Samp*(B_I, π) and outputs $\psi' \to \psi$ mod .

Decrypt(s_k, ψ). Takes as input the secret key s_k and a ciphertext ψ. It outputs

$\phi \to (\psi \bmod B_J^{sk}) \bmod B_I$

Evaluate(p_k, C, Ψ). Takes as input the public key p_k, a cycle C in some permitted set $C\varepsilon$ of cycles composed of Add_{BI} and $Mult_{BI}$ gates and a set of input ciphertexts Ψ. It invokes *Add* and *Mult*, given below, in the proper sequence to compute the output ciphertext ψ. (We will describe $C\varepsilon$ when we consider correctness below. If desired, one could use diûerent arithmetic gates.)

Add(p_k, ψ_1, ψ_2). Outputs $\psi_1 + \psi_2$ mod B_J^{pk}

Mult(p_k, ψ_1, ψ_2). Outputs $\psi_1 \times \psi_2$ mod B_J^{pk}

Remark Concerning *IdealGen*, it is fine if the secret basis B_J^{sk} defines a lattice $L(B_J^{sk})$ 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 different cycles. First, Evaluate takes a mod-B_I cycle C as input. This cycle is implicitly applied to plaintexts. Second, Evaluate applies a cycle related to C, which we call the generalized cycle, to the ciphertexts; this cycle uses the ring operations (not modulo I).

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 different cycles. First, Evaluate takes a mod-B_I cycle C as input. This cycle is implicitly applied to plaintexts. Second, Evaluate applies a cycle related to C, which we call the

generalized cycle, to the ciphertexts; this cycle uses the ring operations (not modulo *I*).

Definition (Generalized Cycle). Let *C* be a mod-B_I cycle. We say generalized cycle $g(C)$ of *C* is the cycle 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 which concerns correctness.

Definition (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* mod B_J^{sk}, the set of distinguished representatives of cosets of *J* wrt the secret bass B_J^{sk}.

Conclusion

Finally we conclude that by proving above thermos we can achieve the a Cloud Bootstraple Encryption Schemes and we have solved the cloud coset problems by using random functions.

References

[1] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. Mathematische Annalen 296(4) (1993) 625–635.

[2] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, K. Yang. On the (Im)possibility of Obfuscating Programs. In Proc. of Crypto '01, LNCS 2139, pages 1–18.

[3] D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. In Proc. of STOC '86, pages 1–5.

[4] D. Beaver. Minimal-latency secure function evaluation. In Proc. of Eurocrypt '00, pages 335350. Springer, 2000.

[5] M. Bellare and A. Sahai. Non-malleable encryption. Equivalence between two notions, and an indistinguishability-based characterization. In Proc. of Crypto '99, LNCS 1666, pages 519–536. Springer, 1999.

[6] 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.

[7] J. Benaloh. Verifiable secret-ballot elections. Ph.D. thesis, Yale Univ., Dept. of Comp. Sci., 1988.

[8] 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.

[9] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. Eurocrypt '98, LNCS 1403, pp. 127–144.

[10] D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. J. ACM, vol. 48, no. 4. Pages, 702-722. ACM, 2001. Preliminary version in Crypto 1997.