JOURNAL OF THEORETICAL PHYSICS & CRYPTOGRAPHY
www.IJTPC.org/Publications

Quasi-Linear Time Fully Homomorphic Public Key Encryption Algorithm (ZK111)*

Zaryab Khan
Rochester Institute of Technology, Rochester, NY, USA
Accepted: 25th October, 2012

ABSTRACT
ZK111 is a quasi-linear O(n · log n) fully Homomorphic encryption (FHE) algorithm. It works on a novel idea of "perfectly colorblind function" which is nothing but a very unique way of creating p-adic ring homomorphism from p-adic ring X to p-adic ring Y. Unlike its other FHE counterparts, there is NO "noise" in the cipher text. ZK111 preserves full algebraic homomorphism without compromising its quasi-linear efficiency. It is not only FHE, but also the fastest public-key encryption (PKE) algorithm known as of today.. Both encryption and decryption are O(n · log n) and algorithm still retains it FHE property. Hence it solves the most intriguing and outstanding problem of cryptography. ZK111 is a probabilistic encryption algorithm by nature, and despite being FHE; it is absolutely immune to chosen and adoptive plaintext attack by design.

Keywords
Fully Homomorphic Encryption (FHE), Public Key Encryption (PKE), Unconditionally Secure, Transition Function, p-adic numbers, Ring Homomorphism, Algebraic Homomorphism.

1. INTRODUCTION
The fundamental idea of the algorithm is "we are all perfectly colorblind relative to each other". This idea was taken from a centuries old philosophical puzzle in which there is a man who has a very weird problem. Every time he looks at a color, it get swapped in his head with some other color. For example, if he looks at red, then as soon as red enters his retina, it becomes blue. when he looks at blue, it becomes green and green becomes red. The catch is that man "always" stays perfectly consistent with the labeling of the color. Labeling refers to the "name" of the color. When he is looking at "our red", he would call it red but "actually" he is looking at blue, because our red becomes his blue and so on. Question is; is there a way we can detect that this person is "color-blind" as long as he stays perfectly consistent with the labeling? Answer is; NO, we can't detect his color-blindness. Secret Key would be that morphisim which swaps colors and public key would be those labels "red, blue, green etc". We mathematically achieve this using p-adic ring homomorphism ψ from one p-adic ring extension X to another p-adic ring extension Y. These extensions are unramified. We call ψ "color-blinder". ψ is that secret relation (morphism) between X and Y .

*Patents Pending: PCT/US12/54333, PCT/US12/55998

CAUTION: We NEVER substitute any numerical value for either x or y.
This encryption algorithm is all about the coefficients of indeterminate x and y.
There are two publically known pre-decided parameters namely.
1) p which is a prime number.

2) Message space *MS*={(0, p^{ms})} where ms is an integer.

As usual, we have two keys. Secret Key which we denote by
(a) SK = { ker (X), $\varphi^{-1}(y)$, a_0^{-1} }

$\phi(x)$ and $\phi^{-1}(y)$ are linear relations such that $\phi^{-1}(\phi(x)) = x$. $\phi(x)$ is called "transition function" as it helps us transit from the p-adic ring X to p-adic ring Y. a_0 is a random integer which lies in the interval (0, p^{ms}) and gcd (a_0, p^{ms})=1.

$X=Z_p/(p(x)/p^{\alpha})$. Z_p means ring of p-adic integers. ker(*X*) = $(p(x)/p^{\alpha})$. Although p is a prime and public but α is kept a secret and no-one knows the value of α.We call α "adic-number". p(x) is an irreducible polynomial. The only known relation between α and *ms* is $\alpha > ms$.

Public Key is denoted by

(b) $PK = \{ker(Y), e_{k1}, e_{k2}\}$

e_{k1} and e_{k2} are called encryption key 1 and encryption key 2 respectively. They are polynomials in the indeterminate y. $Y=Zp/(q(y)/p^{\beta})$. $ker(Y)=(q(y)/p^{\beta})$.
NOTE: There is only one known relation between α and β which is $\beta > \alpha+2$. So we never precisely know how small α is.

1. ALGORITHM

The Algorithm consists of five crucial sub-routines namely

i) Color-blinder ψ and its inverse ψ^{-1}
ii) Ring Construction X and Y.
iii) Method to generate e_{k1} and e_{k2}.
iv) Encryption Function E(m).
v) Decryption Function D(c).

1.1 Color-blinder ψ

Color-Blinder is technically nothing but a homomorphism which relates the ring X to ring Y. Color-Blinder is technically nothing but a homomorphism which relates the ring X to ring Y. In order to define ψ we first have to define the "transition function" ϕ(x) . ϕ(x) is constructed as follows.
0) pick secret integer α and public integer β such that $\alpha > ms$ and $\beta > \alpha+2$.
1) pick a secret linear polynomial ax+b such that $gcd(a, p^{\alpha}) = gcd(b, p^{\beta})=1$.
2) find additive inverses of a and b mod p^{α} and denote them a' and b'.
3) Check if $-a'x-b' = ax+b \pmod{p^{\alpha}}$.
If True, we approve and store −a' and−b'.
4) We define $\phi(x) = -a'y-b' \bmod p^{\beta}$. We actually "lifted" the coefficients from mod p^{α} to mod p^{β} and homomorphically transited x from ring X to p-adic ring Y according to the relation $\phi(x) = -a'y-b' \bmod p^{\beta}$
5) Given $\phi(x) = -a'y-b' \bmod p^{\beta}$, We define $\phi^{-1}(y)$ as follows
$\phi^{-1}(y) = (-a')^{-1} x + ((-a')^{-1}) b' \bmod p^{\alpha}$

NOTE: The coefficients –a' and –b' (mod p^{β}) becomes a and b when reduced (mod p^{α}).

Color-Blinder ψ takes a polynomial with indeterminate "x" as its parameter and "transform and lift" it's coefficients homomorphically to another polynomial in "y" according to the given recipe.
Take any polynomial $u(x)=c_n x^n + c_{n-1} x^n \ldots + c_0$ such that its coefficients c_i lie within the interval $[0, p^{\alpha})$.
ψ transform u(x) it to another polynomial u'(y) according to the following method.
$\psi(u(x)) = u(\phi(x)) = u'(y) \bmod ((p^{\beta}/q(x))$

$\psi^{-1}(u'(y) = u'(\phi^{-1}(y)) = u(x)$.

This completes the description of ϕ^{-1}, ϕ , ψ, ψ^{-1}

1.2 Ring construction X and Y

We already know the secret integer α and publically known integer β and secret morphism ϕ(x). We have to keep the cofficients of –a'x – b' and ax + b secrets otherwise ϕ won't be secret. Let's start the construction of p-adic extension rings X and Y . We start from p-adic extension X which is a secret and then construct Y, which will be public. Prime p and integer ms are public by default.

1) pick an irreducible polynomial $p^{-}(x)$ of degree n as you would normally do to create a finite field F_{pn} . $p^{-}(x)$ does NOT have to necessarily monic, but must be irreducible.
2) for each coefficient c_i of $p^{-}(x)$, we add some random gibberish such that reduction (mod p) would give $p^{-}(x)$. We call this "random lift of $p^{-}(x)$." Recipe is as follows

$$c_i + \Sigma^{\alpha-1}_{j=1} k_j \times p^j .$$

k_j lies within the interval $[0,p^{\alpha}]$ and chosen randomly.
The resulting polynomial is p(x). check $p(x) = p^{-}(x) \pmod p$. If true, we store p(x) and approve it to be used as a part of our Secret Key.
NOTE: Just like $p^{-}(x)$; *p(x) is also irreducible over the p-adic ring with fix mod* $Zp/(p^{\alpha})$.
Now we are ready to construct the unramified p-adic extension X according to the recepie
given below.

1) R is a p-adic ring with a fix (mod p^{α}).
2) $R = Zp/(p^{\alpha})$.
3) We create unramified p-adic ring extension $X=R[x]/(p(x)/p^{\alpha})$. As we can see the kernel of X is $(p(x)/p^{\alpha})$ and it is kept secret.

Now we will create unramified p-adic ring extension Y from X according to the following recipe.

1) $R'=Zp/(p^{\alpha})$. α and β were already chosen at the beginning of the algorithm and we also worked out p(x). α and β obey relation $\beta > \alpha+2$.
2) We will create q(y) by plugging in p(x) in ψ as explained below. ϕ(x) was chosen and defined beforehand.
3) $\psi(p(x)) = p(\phi(x))) = q(y)$.
(CAUTION: We should not try to make q(y) a monic.)

1.3 Key Generation e_{k1} , e_{k2}

1) Randomly select a polynomial d_1 belongs to p-adic ring X and an integer a_0 such that $gcd(a_0,p^{ms})= 1$. a_0 will be the constant term of the polynomial d_1 . store the inverse of a_0 which will also be the part of our secret key.
2) Compute the p-adic exponential
3) $k1 = \exp(p \times d_1) \bmod (p(x)/p^{\alpha})$ Remember in most of the p-adic cases the $\log(k1) \neq p \times d_1$ because of non-canonical nature of p-adic exponential. It also serves as a strength of this cryptosystem.
4) $d_1' = \log(k1)$
5) $k1' = \exp(d_1')$.
6) Select another polynomial d_2 just like we selected d_1 but for d2 we do NOT care about the constant term a_0. It will not matter in case of d_2.
7) Pick an integer n such that $ms < n < \alpha$. We will compute the p-adic exponential
8) $k_2 = \exp(p^n \times d_1) \bmod (p(x)/p^{\alpha})$
9) Compute d'_2 and k'_2 analogous to step 4 and 5.

10) randomly pick two polynomials t1 and t2 just like we picked d_1 and d_2.
11) Compute their teichmuller lifts tch1 and tch2 respectively.
12) Compute $k_1'' = tch1 \times k'_1 \bmod (p(x)/p^{\alpha})$
13) Pick two other random polynomials r1 and r2 just like we picked t1 and t2.
14) Compute $k_1''' = \psi(k_1'')$.
15) Compute the $k_2''' = \psi(k_2'')$ Here we have homomorphically transformed and lifted k_1'' and k_2''.
16) Compute r1' = ψ (r1) and compute r2' analogous to r1'.

REMEMBER: We already computed the kernel of $Y = (p^{\beta}/q(y))$ beforehand. k_1''' , k_2''' , r1' and r2' all belong to Y after being transformed and lifted by function ψ.

17) Pick a random integer i such that $\alpha < i < \beta$.
18) Compute $e_{k1} = k_1''' \times (r1')^{p\wedge I} \bmod (q(x)/p^{\beta})$
19) Compute $e_{k2} = k_2''' \times (r2)^{p\wedge I} \bmod (q(x)/p^{\beta})$
20) Pick a random numerical teichmuller character and multiply it to both e_{k1} and e_{k2}

NOTE: In each of the step 19 and 20, we pick a different *i.*

We are ready to publish our public key

PK = {e_{ke} , e_{k2}, ker(Y) = $(q(y)/p^{\beta})$}

and the corresponding secret key will be

SK = {ker(X) = $(p(x)/p^{\alpha})$, $\phi^{-1}(y)$, a_0^{-1}}

1.4 Encryption Function

1) Pick a random number *I* which belongs to the interval $(0,p^{ms})$
2) Pick a random integer i such that $\beta - 3 < i < \beta$
3) Flip a binary coin (T or F)

if True, pick a NUMERICAL teichmuller character N belongs to ring Y.

$$E(m) = (e_{k1})^{m} \times (e_{k2})^{rnd} \times (I)^{p\wedge i} \times N \bmod (q(y)/p^{\beta})$$

if False

$$E(m) = (e_{k1})^{m} \times (e_{k2})^{rnd} \times (I)^{p\wedge i} \bmod (q(y)/p^{\beta})$$

where rnd is random integer.

1.5 Decryption Function

set E(m) = *c* Decryption function takes *c* as an input parameter and process to "precisely" recover plaintext without adding anything like "noise" in the decrypted cipher text. Decryption is a three step process.

1) $c' = \psi^{-1}(c)$.
2) $c'' = (\log(c') \bmod (p(x)/p^{\alpha}))$

take the constant term of c'' and call it *m'* and discard rest of the polynomial.

3) $m = (a_0^{-1}) \times (m') \bmod p^{ms})/p$

Now the p is prime we are performing simple division by p. It is NOT p inverse.

2. FULLY HOMOMORPHIC FEATURE

This algorithm preserves full algebraic homomorphism with in the range $(0,p^{ms})$

2.1 Addition

$E(m1) \cdot E(m2) = E(m1 + m2)$.

2.2 Multiplication

$(E(m1))^{m2} = E(m1 \cdot m2)$

2.3 How and why decryption works?

Now first of all we need to recall that logrithm of teichmullar character is 0. In a fixed-mod ring, teichmuller character form a cyclic group. Most important of all, we do the reduction of teichmuller character mod p^{v} would still be a teichmuller character w.r.t mod p^{v} .
If you look at $(r1')^{p\wedge i}$, here $i>\alpha$ therefore $\psi^{-1}((r1')^{p\wedge i})$ will do the reduction and homomorphic inverse transformation back to p-adic ring X and $\psi^{-1}((r1')^{p\wedge i})$ is nothing but teichmuller character once again!
Another important fact; recall that p-adic log is multiplicative. $\log(u \cdot v) = \log(u) + \log(v)$.
Now if we look at $c = (e_{k1})^{m} \times (e_{k2})^{rnd} \times (I)^{p\wedge i} \bmod (q(y)/p^{\beta})$, you will observe that $\psi^{-1}((I)^{p\wedge i})$ is also a teichmuller character. In the step 2 of the decryption sub-routine, when we take the log of $\psi^{-1}(c)$, we actually get
$c'' = (m \cdot \log(tch1) + m \cdot p \cdot d1 + m \cdot \log(tch2) + m \cdot \log((r1)^{p\wedge i}) + rnd \cdot \log((r2)^{p\wedge i}) + rnd \cdot p^{n} \cdot d_2 \bmod (p(x)/p^{\alpha}))$.
All of the terms of the above expressions are going to be zero but "$rnd \cdot p^{n} \cdot d_2 + m \cdot p \cdot d1$". Once we take reduction mod p^{ms}. $rnd \cdot p^{n} \cdot d_2$ also vanishes because $p^{n} > p^{ms}$.
Then Pick the constant term of the of the remaining polynomial and it will be "$a_0 \cdot m \cdot p$". Then we simply take inverse of a_0 and divide by p to recover our message m. Now if you can observe, its actually nothing but p-adic exponential in disguise.

3. SECURITY AND EFFICIENCY ANALYSIS

3.1 Security

The security of this algorithm chiefly relies upon finding the transition function $\phi(x) = -a'y - b' \bmod p^{\beta}$. The claim is "it NOT even logically possible to find out φ(x) via brute force which implies that the system is Shanon Secure or simply "Information Theoretic Secure". This claim is based upon two assertions.
1) Given a ciphertext c, its mathematically impossible to recover the plaintext m.

2) Given public key PK its mathematically impossible to recover secret key SK.
We go about the proof using random oracle model.

Given a ciphertext *c*, it is not possible to recover the plaintext *m* because there are multiple values of *m* which would map to c depending upon the value of "rnd" and $(I)^{p\wedge i}$ and of course the outcome of the coin flip which would multiply a teichmuller integer N if true else it will not get multiplied. This further complicates the situation for the attacker using the brute force because once again there is an equal probability of the any two plain texts being mapped to the same cipher text depending upon the values of coin flip and then $(I)^{p\wedge i}$. Feed a cipher text c to a random oracle and say the output was m'. You could feed the same cipher text again and you simply change the value of our random gibberish rnd and $(I)^{p\wedge i}$ and you will get a different plain text. Now for the person having private key will be able to recover the actual exact plain text but a brute forcer will not be able to do so. There is equal probability that the given cipher-text "c " corresponds to any one of the plaintexts m with in the interval $(0,p^{ms})$. Therefore assertion (1) corresponds to the criteria of being Shanon Secure.
Now we look at the assertion 2. Given public key PK, It is not possible to find ϕ^{-1}, ϕ alike. The reason is quite simple. In order to recover secret key, we need to get ker (X). In order to get to the ker(X) we need two necessary components.

1) ϕ^{-1} or ϕ
2) adic number α

The p-adic exponential is non-canonical hence we must have to find p(x) and p^{α} in order to know anything about k_1 and a_0. But once again In order to get to the p(x) we need ϕ^{-1} and ϕ^{-1} is a secret by default. Hence it is mathematically impossible to find the ker(X).

Even if I give you all the possible combinations of (-a',-b') and different adic numbers α', it will still be impossible to get ϕ^{-1} because you can't verify which combination is "cryptographically" right one. They will all make mathematical sense but only one is "actually" right. Hence the problem of finding ϕ^{-1} does NOT even fall into the category of NP-Hard problems. It's beyond that because there is mathematically no way to verify which combination is the right one because none of them has an answer NO. They all are mathematically correct.

3.2 Efficiency

Using pre-computed the Secret Key and Public Key, the cost of encryption is nothing but O(n · log n) because encryption only involves exponentiation and multiplication and using fast exponentiation by squaring, we can cut down the number of multiplications to "log n" and faster multiplication algorithm cost only O(n). Hence we never exceed O(n · log n)
Decryption involves Multiplication and taking the p-adic logarithm. The time complexity logarithm can be expedited to O (n · log n) using algorithm described by Dan Bernstein. As you can observe the most expensive operation we have to perform is logarithm and it is quasi-linear. This ends the description of the algorithm.

4. CONCLUSION

ZK111 is perhaps the only known public-key encryption algorithm which is also Information Theoretic Secure. On top of being so secure, it is extremely easy to implement because we
only have to create two rings which are homoamorphically related to each other by the .The most surprising element of this algorithm is its fully-homomorphic property which also preserves full algebriac homomorphism. Unlike its other so-called fully homomophic counterpart Gentry's Scheme there is absolutely no concept of noise in the decrypted plaintext.
The main reason is that the algorithm never uses anything like *ideal lattices* which are computationally very expensive. Therefore the time complexity of ZK111 is literally quasi-linear w.r.t the length of the message m. This is perhaps the biggest security breakthrough the cloud computing has been waiting for. This algorithm can be implemented as a block and stream cipher. Cutting the long story short, ZK111 is information theoretic secure, quasi-linear O(n· logn), fully-homomorphic, public key encryption algorithm. It is one of its kind because all of the above mentioned features were never present in one single algorithm.

5. REFERENCES

[1] Shannon C. E., A Mathematical Theory of Communication, The Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, July, October, 1948.
[2] http://cr.yp.to/lineartime/multapps-20041007.pdf.
[3] Gentry Craig., Fully homomorphic encryption using ideal lattices STOC '09 Proceedings of the 41st annual ACM symposium on Theory of computing Pages 169-178.
[4] Kazuy Kato., Iwasawa theory and generalizations, in Sanz-Solé, Marta; Soria, Javier; Varona, Juan Luis et al., International Congress of Mathematicians. Vol. I, Eur. Math. Soc., Zürich, pp. 335-357, 2007. (doi:10.4171/022-1/14, ISBN 978-3-03719-022-7, MR 2334196 12)