Simple Homomorphisms of Cocks IBE and Applications

Rio LaVigne*

November 29, 2016

Abstract

The Cocks Identity Based Encryption (IBE) scheme, proposed in 2001 by Clifford Cocks, has been the standard for Quadratic Residue-based IBE. It had been long believed that this IBE did not have enough structure to have homomorphic properties. In 2013, Clear, Hughes, and Tewari (Africacrypt 2013) created a Cocks scheme derivative where they viewed ciphertexts as polynomials modulo a quadratic. While the scheme was homomorphic, it required sending twice as much information per ciphertext as the original Cocks scheme. A recent result by Joye (PKC 2016) used complex algebraic structures to demonstrate the fact that Cocks IBE, on its own, is additively homomorphic.

In this work, we build upon the results from CHT and Joye. We take the simple intuition from CHT, that ciphertexts can be seen as polynomials, but also demonstrate that we only need to send as much data as in the original Cocks scheme. This perspective leads to better intuition as to why these ciphertexts are homomorphic and to explicit efficient algorithms for computing this homomorphic addition.

We believe that our approach will facilitate other extensions of Cocks IBE. As an example, we exhibit a two-way proxy re-encryption algorithm, which arises as a simple consequence of the structure we propose. That is, given a re-encryption key, we can securely convert a ciphertext under one key to a ciphertext under the other key and vice-versa (hence two-way).

1 Introduction

Identity-Based Encryption (IBE) is a special kind of public key encryption that lets any string be a public key. The first two IBE schemes were the elliptic-curve pairing based scheme by Boneh and Franklin, and the Quadratic Residue (QR) based scheme by Cocks, both in 2001 [BF01, Coc01]. In 2008, Gentry,

*MIT. Email: rio@mit.edu This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1122374. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

Peikert, and Vaikuntanathan got an IBE from lattices [GPV08]. Unlike pairings and lattices, QR-based schemes rely on the same simple algebra as RSA, and the security assumption is the first assumption used in probabilistic encryption (Goldwasser-Micali [GM84]). However, while pairing-based and lattice-based IBE constructions have been extended to Hierarchical IBE, Functional Encryption, Predicate Encryption, etc., progress wtih QR-based IBEs has been comparatively slow [BBG05, ABB10, BSW11, SW05, Boy13, AFV11, GVW15, GPSW06]. Our goal in this paper is to uncover the surprising simple structure in Cocks IBE, extending the results of Clear, Hughes, and Tewari [CHT13], and Joye [Joy16].

In 2013, Clear, Hughes, and Tewari [CHT13] made the major observation that a Cocks ciphertext c can be viewed as a polynomial modulo a quadratic, i.e. a linear function $2x + c$. Decryption is evaluating the polynomial at the secret key and then taking the Jacobi symbol: $\left(\frac{2r+c}{N}\right) = m$. They created a derivative IBE scheme (CHT) based on these linear functions. The advantage of their framework was that the homomorphic properties were straightforward: homomorphic addition was as simple as multiplying the two polynomials together and then taking them modulo the quadratic, resulting in another linear ciphertext of the form $ax + b$. The downside of this variant is that it requires twice as much bandwidth as the original Cocks scheme; users need to send 2 elements in $\mathbb{Z}_N$ to describe a linear function, instead of just one.

This past year, Joye in [Joy16] demonstrated that the Cocks scheme *already* has homomorphic properties without the CHT modifications. Here is a quick look at how he accomplishes this. First, he shows that Cocks ciphertexts are part of a multiplicative structure based on the algebraic torus [RS03]. This torus is a group defined from a quadratic field extension. He is able to define a similar multiplicative structure (although no longer a group) by "removing" the square roots from the ring and then adding them back as an extension. Intuitively, this scheme works because the multiplicative structure only breaks if we are able to find a square root, and by the Quadratic Residuosity Assumption (QRA), this happens only with negligible probability. By treating Cocks ciphertexts as a multiplicative group (multiplying ciphertexts multiplies their Jacobi symbols in decryption), he is able to homomorphically add ciphertexts.

Joye's proof, while striking, is unfortunately complicated, which we believe obscures some of its potential applications. In this paper, we will combine the best of CHT and Joye's work [CHT13, Joy16] by showing we can achieve Joye's homomorphic guarantees with a simpler analysis of Cocks' scheme. And indeed, this leads us to our new application: identity-based proxy re-encryption of Cocks ciphertexts.

1.1 Our Results

Our contribution is two-fold. First, we demonstrate simple, length-preserving homomorphic addition of Cocks ciphertexts. Our generalization goes through the beautiful linear function lens proposed by CHT, and we show how to acheive length-preservation with little effort. Our main technical lemma gives a way to

re-randomize these linear function ciphertexts. We note that Joye goes through a similar conceptual re-randomization step, but without proving convergence (though intuitively it appears to work). We show that our rerandomization step, with overwhelming probability, converges in an expected constant number of iterations—in fact 2.

Our second contribution is an application: two-way proxy re-encryption of Cocks ciphertexts. Two-way proxy re-encryption allows for a third party with a re-encryption key for two public keys, R and R', to convert a ciphertext encrypted with R to a ciphertext encrypted with R' and vice-versa, hence two-way. This application relies on our ability to re-randomize a ciphertext and convert from CHT ciphertexts to Cocks ciphertexts.

1.2 A Technical Overview

Cocks IBE and Linear Functions. We will start with describing the Cocks IBE scheme. From there, we can show how a Cocks ciphertext can be seen as a linear function as in CHT [CHT13], and then why this perspective is useful.

A Cocks IBE ciphertext consists of two values $(c, \hat{c})$ in $\mathbb{Z}_N$ where $N = pq$ for primes $p \equiv q \equiv 3 \mod 4$. c was encrypted under public key R and $\hat{c}$ was encrypted under $-R$. Since R is an element with Jacobi symbol 1, either it is a square mod p and mod q, or a square in neither. Because $p \equiv q \equiv 3 \mod 4$, -1 is a non-square both mod p and mod q, meaning exactly one of R and $-R$ has a square root r. Thus exactly one of c and $\hat{c}$ contains information about the message. So, without loss of generality we will assume $R = r^2$ and work only with the c component of the ciphertext.

To encrypt a message $m \in \{\pm 1\}$ under a public key $R = r^2$, we choose a $t \xleftarrow{\$} \mathbb{Z}_N$ so that $\left(\frac{t}{N}\right) = m$, and let $c = t + \frac{R}{t}$. We decrypt with $\left(\frac{2r+c}{N}\right)$. The Cocks ciphertext, when viewed as a linear function as in CHT, is $2x + c$. Similarly, if we have a linear function ciphertext $ax + b$ and $a = 2$, then we could treat it as a Cocks ciphertext and only need to send the constant term in a ciphertext – the linear term is assumed to be 2.

Consider two Cocks ciphertexts c and c' encryptions of m and m' respectively. If we multiply their linear function forms together, we get a polynomial $g(x) = (2x + c) \cdot (2x + c') = 4x^2 + 2(c + c')x + cc'$. Notice that if we then mod $g(x)$ out by $x^2 - R$ to get $\bar{g}(x)$, $g(r) = \bar{g}(r)$. So, for linear function $\bar{g}(x) = 2(c + c')x + cc' + 4R$, when we decrypt

$$\left(\frac{\bar{g}(r)}{N}\right) = \left(\frac{g(r)}{N}\right) = \left(\frac{2r + c}{N}\right) \cdot \left(\frac{2r + c'}{N}\right) = m \cdot m'.$$

However, $\bar{g}(x)$ is no longer a Cocks ciphertext. It is a linear function of the form $ax + b$. We need some way to normalize it so that we only need to send the constant term in our ciphertext. If $\left(\frac{a/2}{N}\right) = 1$, then we can divide through by $a/2$ to get $2x + 2b/a$ as a ciphertext (corresponding to the Cocks ciphertext

$2b/a$), and when we decrypt, we get

$$\left(\frac{2r + 2b/a}{N}\right) = \left(\frac{2/a}{N}\right) \cdot \left(\frac{ar + b}{N}\right) = 1 \cdot m.$$

If $a/2$ does not have Jacobi symbol 1, we will need some way to re-randomize the ciphertext so that, with constant probability, we change the linear term. Our re-randomization step is that for input $ax + b$, we can multiply by a random square $(cx + d)^2$ and $1/t$ where t has Jacobi symbol 1 to get $ex + f = (ax + b)(cx+d)^2/t \pmod{x^2 - R}$. This leads us to our main technical lemma. Lemma 1, states that for almost all elements $ax + b$ in $\mathbb{Z}_n[x]/(x^2 - R)$, then the linear term after this re-randomization will have Jacobi symbol 1 with probability almost exactly $1/2$.

With this lemma, we can re-randomize almost any ciphertext in $\mathbb{Z}_N[x]/(x^2 - R)$ and are guaranteed with probability half that the symbol of the linear term will change. Note that a negligible fraction of possible ciphertexts do not have this property, which we show in remark 4.2, and so we need to prove this lemma. Our other main lemma, lemma 2, shows this re-randomization step truly re-randomizes a ciphertext, which is necessary for our application, proxy re-encryption.

Overview of Proxy Re-encryption. Proxy re-encryption was introduced in 1998 by Blaze, Bleumer, and Strauss (BBS), where they provided some informal definitions and a technique for bidirectional re-encryption of ElGamal ciphertexts [BBS98]. Proxy re-encryption allows a proxy P to re-encrypt messages given a re-encryption key. If P is given the re-encryption key from Alice to Bob, then P can take a message encrypted under Alices public key and output the same message encrypted under Bob's public key without learning anything about their secret keys or the message. If P can also re-encrypt messages from Bob to be messages from Alice, this scheme is called *bidirectional*. If P can only re-encrypt one way, the scheme is called *unidirectional*.

Proxy re-encryption is quite useful. Take, for example, key escrow. A key escrow service might contain re-encryption keys for user keys to the FBI's key. The service could then mediate which user messages the FBI could decrypt without learning anything about the user messages or user secret keys. Without re-encryption, users would need to trust the escrow service with their secret keys. Another simple application is just having a server that can distribute encrypted files to many users under many different keys. The load of re-encrypting and distributing these files falls on the server, which will only have the encrypted files and re-encryption keys – no secret keys. So, even if the server were compromised, no important information is leaked.

So far, we have seen that bidirectional and unidirectional proxy re-encryption exist for some regular PKE and IBE schemes [BBS98, SC09]. For regular PKE, we have bi- and unidirectional proxy re-encryption for both pairing-based and Diffie-Hellman-based schemes [AFGH06, CH07, SC09]. In the IBE case, we are able to get both bi- and unidirectional proxy re-encryption under the DBDH

assumption [AFGH06, CH07]. In 2006, Ateniese et al. showed their schemes were CPA secure [AFGH06, GA07], and provided new definitions for what it meant to be CPA secure. A year later, Canetti and Hohenberger formalized CCA security for proxy re-encryption [CH07]. So far, however, these results are only for IBE schemes based on DBDH. In this work, we provide bidirectional proxy re-encryption keys for an IBE based on QRA, Cocks.

Our re-encryption scheme is surprisingly simple. Ciphertexts from CHT are of the form $ax + b$ where decryption is taking the symbol using a secret key r: $\left(\frac{ar+b}{N}\right)$. To generate a re-encryption key from $R = r^2$ to $R' = r'^2$, we output $T = r/r'$, just the ratio between the secret keys. Now, we take our ciphertext $ax + b$ and can just transform it into $aTx + b$. Then, when we decrypt, we have that

$$\left(\frac{aTr' + b}{N}\right) = \left(\frac{a(r/r')r' + b}{N}\right) = \left(\frac{ar + b}{N}\right).$$

Since we have the ability to take a linear function ciphertext from CHT and convert it into a Cocks ciphertext, we can just take $aTx+b$ and convert that back into a standard Cocks ciphertext under the appropriate public key. In practice, we will want to hide the re-encryption key, which will involve re-randomizing the CHT ciphertext, invoking lemma 2, but this is the main idea.

1.3 Other Related Work

One derivative of the Cocks scheme, generalized to higher power residue symbols, was created by Boneh, LaVigne, and Sabin [BLS13]. In this scheme, several bits can be sent at once, but at the cost of greatly increasing the ciphertext size. This work used a representation of ciphertexts similar to that in CHT. It was discovered that the ciphertext size was, unfortunately, much less efficient than the original Cocks scheme. The representation of ciphertexts as polynomials is key in this work, and although the homomorphic properties of this derivation were not explored, much like in CHT, they are there.

Another quadratic residue IBE is the one developed by Boneh, Gentry and Hamburg (BGH) in 2007 [BGH07], which was able to increase the bandwidth. Cocks scheme sends one bit at a time, while BGH can send polynomially many with the ciphertext size increasing by one bit for each bit sent. The price is that the secret and public keys increase by one element in $\mathbb{Z}_N$ for each bit, and that encryption time is relatively expensive (quartic). And while this scheme is able to significantly decrease the size of ciphertexts, it does not seem to have homomorphic properties.

Proxy re-encryption was introduced in 1998 by Blaze, Bleumer, and Strauss for ElGamal ciphertexts [BBS98]. Their construction was for two-way (bidirectional) proxy re-encryption. Later, proxy re-encryption was defined in the one-way (unidirectional) sense for some pairings-based cryptosystems [AFGH06, CH07], and in 2009, Shao and Cao got unidirectional proxy-reencryption under the DDH assumption [SC09]. Our security game will be based on the one developed by Green et al. in 2006 for Chosen Plaintext Attack (CPA) security [GA07, AFGH06]. We will have to modify the game slightly since our proxy

re-encryption is not adaptively, only selectively, secure. We will also use the definitions introduced by Canetti and Hohenberger in 2007, where they give an explicit construction for a bidirectional, Chosen-Ciphertext-Attack secure public key encryption scheme based on Decisional Bilinear Diffie Hellman [CH07].

2 Review of Cocks IBE

2.1 Definition of IBE

First, we need to define what an IBE scheme is. Let $\mathcal{M}$ be the message space, and $\mathcal{ID}$ be the space of identities. An IBE scheme consists of a tuple of four algorithms:

- $\mathsf{Setup}(1^k)$. Outputs the master secret MSK and public parameters PP. MSK is kept secret inside of a trusted third party.
- $\mathsf{Keygen}(MSK, PP, id)$. The trusted third party outputs a secret key sk_{id} for an identity id.
- $\mathsf{Enc}(PP, id, m \in \mathcal{M})$. Outputs c, an encryption of a message under the identity id.
- $\mathsf{Dec}(PP, id, c, sk_{id})$. Outputs m, a decryption of c.

The correction constraint is that $\mathsf{Dec}(PP, id, \mathsf{Enc}(PP, id, m), sk_{id}) = m$.

2.2 Definition of Cocks IBE

This is a quick review of the Cocks IBE scheme, detailed in [Coc01]. Look at the original paper for proofs of correctness and security. Assume that we have access to a global secure hash function $H : \mathcal{ID} \to \mathbb{Z}_N^\times$ (where $\mathcal{ID}$ is the space of identities for the Cocks scheme).

The IBE consists of a tuple of algorithms:

- $\mathsf{Setup}(1^k)$. Outputs a modulus $N = pq$ and a non-square u with Jacobi symbol 1 as the master public key and p, q as the master secret key where $|N| = k$.
- $\mathsf{Keygen}(N, p, q, id)$. Takes $H(id) = R$. If R is a square, we compute a square root r so that $r^2 = R \mod N$ (possible given the factorization of N). If R is a non-square, we compute the square root of uR, finding r so that $r^2 = uR$. We output r as the secret key.
- $\mathsf{Enc}(N, id, m \in \{\pm 1\})$. Find two random $t_1, t_2 \xleftarrow{\$} \mathbb{Z}_N^\times$ so that $\left(\frac{t_1}{N}\right) = \left(\frac{t_2}{N}\right) = m$.

  Let $c_1 = t_1 + \frac{R}{t_1}$ and $c_2 = t_2 + \frac{uR}{t_2}$. Output the ciphertext $\mathbf{c} = (c_1, c_2)$.
- $\mathsf{Dec}(N, r, \mathbf{c} = (c_1, c_2))$. If $R = H(id)$ is a square, compute $m = \left(\frac{c_1 + 2r}{N}\right)$. If R is not a square, compute $m = \left(\frac{c_2 + 2r}{N}\right)$.

In Cocks original description of the scheme, he uses $p = q = 3 \mod 4$, and $u = -1$. This is a generalization to any odd p and q that Joye shows works just as well [Joy16].

2.3 Simplified Cocks Ciphertexts

The way that we are able to add two ciphertexts (in Joye's paper, CHT, and this work [Joy16, CHT13]) is to consider each ciphertext $\mathbf{c} = (c_1, c_2)$ and $\mathbf{c}' = (c_1', c_2')$, and then perform some function Hom on the components to get

$$\mathbf{c}'' = (\mathsf{Hom}(c_1, c_1'), \mathsf{Hom}(c_2, c_2')).$$

However, it is cumbersome to always consider a Cocks ciphertext as two separate pieces, one corresponding to the public key R ($c_1 = t_1 + \frac{R}{t_1}$) and the other corresponding to uR ($c_2 = t_2 + \frac{uR}{t_2}$), where u is a non-square. Assume without loss of generality that R is a square. Note that for c_2, all information about the Jacobi symbol t_2 is lost information theoretically, and therefore, no amount of adding components of other ciphertexts together will ever recover the symbol of t_2. So, while discussing the mathematical structure behind Cocks, we assume that R is a square, and that $t + \frac{R}{t}$ is a ciphertext for public key R.

3 The ciphertext as a linear function

Here we define how to see a Cocks ciphertext as a linear function. This perspective was pointed out in CHT [CHT13].

Consider a simplified Cocks ciphertext (defined in 2.3) $c = t + \frac{R}{t}$. We define a linear function $f(x) = 2x + c$. To decrypt, we simply take $\left(\frac{f(r)}{N}\right)$. The following is a descryption for why this works; this is a warmup for how homomorphically adding ciphertexts will work.

To generate f, we take $t \xleftarrow{\$} \mathbb{Z}_N^*$ so that $\left(\frac{t}{N}\right) = m$. Then we let $g(x) = (t - x)^2 = t^2 + 2tx + x^2$ and define

$$f(x) = \frac{g(x) \pmod{x^2 - R}}{t} = \frac{t^2 + R + 2tx}{t} = t + \frac{R}{t} + 2x.$$

This perspective on the Cocks ciphertext makes it clear why decryption is correct. We see that $g(x)$ is always a square regardless of the value of x, but when we take $\bar{g}(x) = g(x) \pmod{x^2 - R}$, $\bar{g}(x)$ is not necessarily equal to $g(x)$ except when x satisfies $x^2 - R$. In particular $\bar{g}(r) = g(r) + (r^2 - R)h(r) = g(r)$. So, when we decrypt,

$$f(r) = \frac{\bar{g}(r)}{t} = \frac{g(r)}{t} \implies \left(\frac{f(r)}{N}\right) = \left(\frac{t}{N}\right) \cdot \left(\frac{g(r)}{N}\right) = \left(\frac{t}{N}\right) = m.$$

3.1 Additive homomorphisms among linear function ciphertexts

If we view Cocks ciphertexts as linear functions, $f_1(x) = 2x + c_1$ and $f_2(x) = 2x + c_2$, and let $f_3'(x) = f_1(x) \cdot f_2(x) \pmod{x^2 - R}$, then $f_3'(x)$ "decrypts" into the product of the first and second messages:

$$\left(\frac{f_3'(r)}{N}\right) = \left(\frac{f_1(r) \cdot f_2(r)}{N}\right) = \left(\frac{f_1(r)}{N}\right) \cdot \left(\frac{f_2(r)}{N}\right).$$

Again, we are using the fact that for any polynomial $g(x)$, $g(r) = g(r) \pmod{x^2 - R}$.

Notice, though, that $f_3'(x) = c_1c_2 + 4R + 2(c_1 + c_2)x$ is in general not a proper Cocks ciphertext. In the next section, section 4, we will show that we can re-randomize a ciphertext so that $\left(\frac{c_1+c_2}{N}\right) = 1$. Once we have that, we can divide f_3' by $c_1 + c_2$ to get a Cocks ciphertext with a linear function of the form $f_3(x) = c_3 + 2x$, where

$$c_3 = \frac{c_1c_2 + 4R}{c_1 + c_2}.$$

Notice that c_3 is exactly the expression for homomorphically adding ciphertexts as described by Joye [Joy16]. Joye has one way of "re-randomizing" the ciphertext c_1. We will take a slightly different approach using the linear function representation of ciphertexts, a la CHT [CHT13].

4 Adding Cocks Ciphertexts

The basic algorithm for adding ciphertexts c_1 and c_2, simplified ciphertexts as defined in section 2.3, will be to multiply their linear function forms together, $ax+b = (2x+c_1)\cdot(2x+c_2)$ and then to rerandomize the resulting linear function $ax + b$ until we can divide by $a/2$, e.g. until $\left(\frac{a/2}{N}\right) = 1$.

4.1 Converting ciphertexts: from a linear function to Cocks ciphertext

Recall a linear ciphertext is of the form $ax + b$ and is decrypted by taking the Jacobi symbol after plugging r into x: $\left(\frac{ar+b}{N}\right)$. We want to convert this into a Cocks ciphertext of the linear form $2x + c$. If $\left(\frac{2/a'}{N}\right) = 1$, then we can simply multiply $ax + b$ through by $2/a$ which does not change the symbol. If not, then we need some way to re-randomize this linear ciphertext.

This re-randomizing step will be to homomorphically add an encryption of 1 as linear functions (as in CHT [CHT13]). We will prove that this changes the linear coefficient and with probability very close to $1/2$, the new linear coefficient will have the correct Jacobi symbol.

Algorithm 1: $\mathsf{Convert}(N, R, ax+b)$

Input: Public parameter N, public key R, and linear function ciphertext $ax+b$
Output: Cocks Ciphertext c

while $\left(\frac{2/a}{N}\right) \neq 1$ **do**
  $ax+b \leftarrow \mathsf{Rerand}(ax+b)$;
end
return $c = 2b/a$

Algorithm 2: $\mathsf{Rerand}(N, R, ax+b)$

Input: Public parameter N, public key R, and linear ciphertext $a'x+b'$
Output: Linear ciphertext $a'x+b'$

Let $c, d \xleftarrow{\$} \mathbb{Z}_N$;
Let $t \xleftarrow{\$} \mathbb{Z}_N$ so that $\left(\frac{t}{N}\right) = 1$;
Compute $a'x+b' = (ax+b)\frac{(cx+d)^2}{t} \pmod{x^2-R}$;
return $a'x+b'$

4.1.1 Rerand is correct

Quickly we will show that this algorithm is correct. Recall that for any polynomial $g(x)$, $g(r) = g(r) \pmod{x^2-R}$. So,

$$\left(\frac{(a'r+b')}{N}\right) = \left(\frac{(ar+b)(cr+d)^2/t}{N}\right) = \left(\frac{ar+b}{N}\right)\cdot\left(\frac{cr+d}{N}\right)^2\cdot\left(\frac{t}{N}\right) = \left(\frac{a'r+b'}{N}\right).$$

4.1.2 Convert is correct

This is also straightforward. Once $\left(\frac{2/a}{N}\right) = 1$, then when we return $c = 2a/b$, we are really computing $(2/a)(ax+b) = 2x + 2b/a$ and returning the constant term. So, when we decrypt c like a Cocks ciphertext, we get

$$\left(\frac{2r+c}{N}\right) = \left(\frac{2r+2a/b}{N}\right) = \left(\frac{2/a}{N}\right)\cdot\left(\frac{ar+b}{N}\right) = \left(\frac{ar+b}{N}\right),$$

which is exactly the decryption of the linear function ciphertext.

4.2 Why Convert runs quickly

First, we will remark on why it is not obvious that re-randomization just works. For an element $ax+b \in \mathbb{Z}_N[x]/(x^2-R)$ where $b/a = r$, then for any $cx+d \in \mathcal{R}_N$,

$$ex+f = (ax+b)(cx+d)^2 \pmod{x^2-R} \text{ has } \left(\frac{e}{N}\right) = \left(\frac{a}{N}\right).$$

Proof. Let $ax + b \in \mathcal{R}$ have $b = ar$. We have that $ax + b = a(x + r)$. Now, for any $cx + d \in \mathcal{R}$,

$$\begin{aligned}(ax + b)(cx + d)^2 \pmod{x^2 - R} &= a(x + r)(2cdx + Rc^2 + d^2) \\ &= a\left((Rc^2 + d^2 + 2crd)x + r(Rc^2 + d^2) + 2Rcd\right).\end{aligned}$$

We can factor the linear term $a(Rc^2+d^2+2crd) = a(rc+d)^2$, and so the Jacobi symbol of this term is always $\left(\frac{a(rc+d)^2}{N}\right) = \left(\frac{a}{N}\right)$. □

There are some elements $ax + b$ that will never terminate in Convert. The elements described in remark 4.2 are elements in $\mathbb{Z}_N[x]/(x^2 - R)$ that do not have inverses, which is a small set within the ring. We will show that there very few elements that could cause Convert to loop forever; in fact, it is only the elements $ax + b$ that do not have multiplicative inverses in $\mathbb{Z}_N[x]/(x^2 - R)$ that cause problems in Convert. This is proved by lemma 1.

Lemma 1. *If $ax + b \in \mathbb{Z}_N[x]/(x^2 - R)$ has a multiplicative inverse,*

$$\Pr_{c,d \xleftarrow{\$} \mathbb{Z}_N^*, t \xleftarrow{\$} \mathbb{J}_1}\left[\left(\frac{e}{N}\right) = 1 \text{ where } ex + f = (ax + b)\frac{(cx + d)^2}{t}\right] = \frac{1}{2}.$$

Sketch of Proof Consider the case where $ax + b$ is invertibale and a square in $\mathbb{Z}_N[x]/(x^2 - R)$. So, we can write $ax + b = (a'x + b')^2 \pmod{x^2 - R}$. $ax + b$ has a multiplicative inverse, and so $a'x + b'$ also has a multiplicative inverse. Since $cx + d$ is a uniformly random element in $\mathbb{Z}_N[x]/(x^2 - R)$, $(cx + d)^2$ is just as likely to be produced as $((a'x + b')(cx + d))^2 = (ax + b)(cx + d)^2$. So, we just need to show that $(cx + d)^2/t$ has the correct distribution of linear terms.

When we expand $(cx + d)^2/t$ we get the linear term is $2cd/t$. Since c, d, and t are all chosen independently, we get that $2cd/t$ is uniformly distributed over $\mathbb{Z}_N^*$, and so the Jacobi symbol of $2cd/t$ is also randomly distributed over ± 1.

The other cases, where $ax+b$ is not a square, behave similarly, but have some subtlety. We need to work mod p, mod q, expressing $ax+b$ as a square or a square times a non-square, and then invoke the Chinese Remainder Theorem. □

The full proof of this lemma is involved and requires a lot of extra notation, and so is formally proved in appendix A.1.

Given this lemma, Convert is expected to loop only a constant number of times. Explicitely, since with overwhelming probability the input $ax + b$ will have an inverse mod $x^2 - R$, and both c and d from line 1 of Rerand will be in $\mathbb{Z}_N^*$, theorem 1 states that with probability $1/2$, Rerand will re-randomize the linear term so that it passes the while loop. Thus, Convert is expected to loop twice.

4.3 The homomorphic algorithm

Given two Cocks ciphertexts c_1 and c_2, here is how we can add them:

Algorithm 3: $\mathsf{Hom}(N, R, c_1, c_2)$

Input: Public parameter N, public key R, and Cocks ciphertexts c_1 and c_2

Output: Cocks ciphertext c_3

Compute $a'x + b' = (2x + c_1)(2x + c_2) \pmod{x^2 - R}$;

$c_3 \leftarrow \mathsf{Convert}(N, R, a'x + b')$;

return c_3

We know that our converting algorithm Convert 1 runs quickly and correctly, and so proving correctness is simply another application of $g(r) = g(r) \pmod{x^2 - R}$ for all polynomials g, explained in section 3.

4.4 Full Cocks Ciphertext Homomorphism

So far we have been working with simplified Cocks ciphertexts 2.3. Consider full Cocks ciphertexts of the form $\mathbf{c}_1 = (c_1, c_1')$ and $\mathbf{c}_2 = (c_2, c_2')$, where c_1 and c_2 are encrypted under public key R and c_1 and c_1' are encrypted under public key uR – where u is a known non-square of Jacobi symbol 1. The homomorphic addition is simply $\mathsf{Hom}(N, R, uR, \mathbf{c}_1, \mathbf{c}_2) = (\mathsf{Hom}(N, R, c_1, c_2), \mathsf{Hom}(N, uR, c_1', c_2')) = \mathbf{c}_3$. We just consider each component of the Cocks ciphertext separately.

5 Application: Bidirectional Proxy Re-encryption

Because we now can see these ciphertexts as linear functions, we get the following method for two-way proxy re-encryption for simplified Cocks ciphertexts, as in section 2.3.

For two secret keys r and r' with public keys $R = r^2$ and $R' = r'^2$ respectively, we let our re-encryption key be $T = r/r'$.

Algorithm 4: $\mathsf{ReEncrypt}(N, R, R', c, T)$

Input: Public parameter N, public keys R and R' to re-encrypt between, Cocks ciphertext c encrypted under R, and proxy re-encryption key T

Output: Cocks ciphertext c' encrypted under R'

$a'x + b' \leftarrow \mathsf{Rerand}(N, R', 2Tx + c)$;

$c' \leftarrow \mathsf{Convert}(N, R', a'x + b')$;

return c'

Correctness of ReEncrypt

Correctness of this algorithm is simple. When we attempt to decrypt c', we get $\left(\frac{2r'+c'}{N}\right) = \left(\frac{a'r'+b'}{N}\right)$ from the correctness of Convert under public key R'.

Then, by correctness of Rerand, $\left(\frac{a'r'+b'}{N}\right) = \left(\frac{2Tr'+c}{N}\right)$. Now, if $T = r/r'$, as it should, $\left(\frac{2(r/r')r'+c}{N}\right) = \left(\frac{2r+c}{N}\right)$. Thus, $\left(\frac{2r'+c'}{N}\right) = \left(\frac{2r+c}{N}\right)$.

Soundness of ReEncrypt

The soundness property is that T is not leaked by the new ciphertext. We need lemma 2, which tells us we sufficiently re-randomize a ciphertext after applying the re-encryption key. Let $\left(\mathbb{Z}_N[x]/(x^2-R)\right)^*$ denote the multiplicative group within the ring $\mathbb{Z}_N[x]/(x^2-R)$, and "decrypting" be evaluating at r and taking the Jacobi symbol. Let $\mathbb{J}_1 \subset \mathbb{Z}_N^*$ be the elements of $\mathbb{Z}_N$ with Jacobi symbol 1. We define the distribution $\mathcal{D}$ to be statistically close to the output of Rerand given input $ax+b$:

$$\mathcal{D}_{ax+b} := \left\{ (ax+b)\frac{(cx+d)^2}{t} : cx+d \xleftarrow{\$} \left(\mathbb{Z}_N[x]/(x^2-R)\right)^*, t \xleftarrow{\$} \mathbb{J}_1 \right\}.$$

The subscript will be omitted when it is clear.

Lemma 2. *If* $(ax+b) \in \left(\mathbb{Z}_N[x]/(x^2-R)\right)^*$ *decrypts to* 1*, then*

$$\mathcal{D}_{ax+b} \equiv \left\{ \textit{uniform } ex+f \textit{ where } \left(\frac{er+f}{N}\right) = 1 \right\}.$$

If $(ax+b) \in \left(\mathbb{Z}_N[x]/(x^2-R)\right)^*$ *decrypts to* -1*, then*

$$\mathcal{D}_{ax+b} \equiv \left\{ \textit{uniform } ex+f \textit{ where } \left(\frac{er+f}{N}\right) = -1 \right\}.$$

Sketch of Proof We will only go through the proof of one specific case in this sketch. The other cases are proved in a similar manner in appendix A.2.

Consider just the case where $ax+b$ decrypts to 1 and is a square in $\mathbb{Z}_N[x]/(x^2-R)$: we can write $ax+b = (a'x+b')^2 \pmod{x^2-R}$. Now, we get that the distribution produced by $\mathcal{D}$ is equally likely to output $(cx+d)^2/t$ as it is to output $((a'x+b')(cx+d))^2/t$. So, we just consider $\mathcal{D}$ choosing $cx+d$ uniformly at random from $\left(\mathbb{Z}_N[x]/(x^2-R)\right)^*$.

For this proof sketch, assume that all $ex+f$ that decrypt to 1 are always of the form $(e'x+f')^2/t'$ for $\left(\frac{t'}{N}\right) = 1$. This is proved in lemma 5 in appendix A. Consider the case where both t and t' are squares. This means, we are choosing $(cx+d)^2$ so that it is equal to $\frac{t}{t'}(e'x+f')^2$. And since $\frac{t}{t'} = k^2$ is a square we can re-write $\frac{t}{t'}(e'x+f')^2 = (ke'x+kf')^2$. So, over all possible choices of $cx+d$ in the multiplicative group, there are 4 possible solutions $(cx+d)^2 = (ke'x+kf')^2$ by the Chinese Remainder Theorem. Now, when considering this probability,

we need to ensure that t is a square, otherwise there does not exist any solution:

$$\begin{aligned}
&\Pr_{cx+d,t\xleftarrow{\$}\mathbb{J}_1}\left[\frac{(cx+d)^2}{t}=\frac{(e'x+f')^2}{t'}\right]\\
&=\sum_{T \text{ is a square}}\Pr_{t\xleftarrow{\$}\mathbb{J}_1}[t=T]\cdot\Pr_{cx+d}\left[(cx+d)^2=\frac{T}{t'}(e'x+f')^2\right]\\
&=\sum_{T \text{ is a square}}\frac{2}{\phi(N)}\cdot\frac{4}{|\left(\mathbb{Z}_N[x]/(x^2-R)\right)^*|}\\
&=\left(\frac{\phi(N)}{4}\cdot\frac{2}{\phi(N)}\right)\cdot\frac{4}{|\left(\mathbb{Z}_N[x]/(x^2-R)\right)^*|}\\
&=\frac{2}{|\left(\mathbb{Z}_N[x]/(x^2-R)\right)^*|}.
\end{aligned}$$

In this case, we get the probability is exactly uniform, since half of the elements in $\left(\mathbb{Z}_N[x]/(x^2-R)\right)^*$ decrypt to 1. In the case where t' in $(e'x+f')^2/t'$ is a non-square, t must also be a non-square in order for there to exist a solution $cx+d$ (so $(cx+d)^2=t/t'(e'x+f')^2$). So, the only thing that changes in this part of the proof from the last is that we sum over non-squares T, and so we get the same probability, $\frac{2}{|\left(\mathbb{Z}_N[x]/(x^2-R)\right)^*|}$, that $\mathcal{D}$ produces $ex+f$.

The other cases, where $ax+b$ decrypts to -1 or is not a square, are handled by looking at $ax+b$ mod p and mod q separately. □

The full proof is detailed in the appendix A.2.

This lemma means that the output of Rerand $a'x+b'$ looks like a fresh ciphertext encrypted under R' that decrypts to the same thing as $2x+c$ with all but negligible probability, and hence the new ciphertext (even after $\mathsf{Convert}$) contains no information about T.

5.1 Proving Selective Security

Security is subtle in this case. With two-way re-encryption keys, we can view each of these keys as an edge in a graph between public keys/identities. If a user has a secret key to any identity and a set of re-encryption keys, that user can decrypt any message encrypted by a public key in the connected component containing that user's key.

5.1.1 Definition of Security

We will adapt the security game from Green and Ateniese [GA07] (section 3), which gives us selective security against a chosen plaintext attack. Let $\mathcal{A}$ be an adversary. This will be under the random oracle model: we answer queries for $H(id)$ with a random oracle.

1. *Setup.* Run $\mathsf{Setup}(1^k)$ to get the msk and mpk. Give mpk to $\mathcal{A}$.

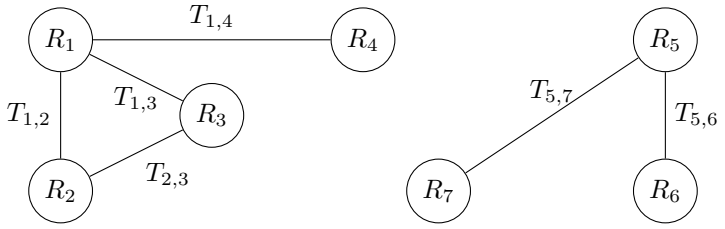

Figure 1: How we should visualize the re-encryption keys $T_{i,j}$. Given a secret key for any node and a set of reencryption keys, we get all of the secret keys in that connected component, but we should get no more information about the other keys. For example, if an adversary has the secret key to R_2 and all of the shown $T_{i,j}$, the adversary can also get secret keys to R_1, R_3, and R_4, but not the secret keys for R_5, R_6, or R_7.

2. *Commit.* $\mathcal{A}$ commits to an identity id^* that it will attack.
3. *Query.* Allow $\mathcal{A}$ to make polynomially many secret-key, public key, and re-encryption key queries. All of these queries must be made at once.[1]
   (a) For a re-encryption key query from $\mathcal{A}$ of the form id_1, id_2, we return $rk_{1,2} = \mathsf{RKGen}(sk_1, sk_2)$ with the exception that $\mathcal{A}$ is not allowed to query for a re-encryption key that allows $\mathcal{A}$ to trivially decrypt messages under id^* using re-encryption keys.
   (b) For id query from $\mathcal{A}$, we return $sk_{id} = \mathsf{Keygen}(mpk, msk, id)$ with the exception that $\mathcal{A}$ is not allowed to query for secret keys in a connected component with id^*.
4. *Challenge.* $\mathcal{A}$ outputs (m_0, m_1). We choose a random $b \xleftarrow{\$} \{0,1\}$ and give $\mathcal{A}$ the ciphertext $c = \mathsf{Enc}(mpk, id, m_b)$.
5. *Guess.* $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. If $b = b'$, then $\mathcal{A}$ wins.

The scheme is secure under this model if the advantage the adversary $\mathcal{A}$ has is negligible. That is,

$$\left|\Pr\left[\mathcal{A}(1^k) = 1 : b = 1\right] - \Pr\left[\mathcal{A}(1^k) = 1 : b = 0\right]\right| < \mathrm{negl}(k).$$

5.1.2 Playing the Security Game

Here we will show that if there is an adversary who can break our security game, then this adversary can break the Quadratic Residuosity assumption 3.

[1] The Cocks' IBE can get away with having some of these queries be adaptive. As long as we know if an id is in a connected component with the challenge id^*.

Definition 3. *The **Quadratic Residuosity Assumption** states that for all PPT adversaries $\mathcal{A}$, if $\mathcal{A}$ is given a composite RSA modulus $N = pq$, and an element $a \in \mathbb{Z}_N^\times$ such that $\left(\frac{a}{N}\right) = 1$, then*

$$|\Pr[\mathcal{A}(N, a) = 1 : a \in \mathbf{QR}_N] - \Pr[\mathcal{A}(N, a) = 1 : a \notin \mathbf{QR}_N]| < \text{negl}(|N|).$$

So, assume $\mathcal{A}$ can break the security game. We will construct an adversary $\mathcal{B}$ to complete the reduction.

1. $\mathcal{B}$ gets N and an element R^* with Jacobi symbol 1.
2. *Setup* for $\mathcal{A}$. $\mathcal{B}$ inputs public parameter N to $\mathcal{A}$.
3. *Commit* for $\mathcal{A}$. $\mathcal{A}$ commits to a public identity id^* that it will attack.
4. *Query* for $\mathcal{A}$. $\mathcal{A}$ outputs a polynomially-sized set of re-encryption key, public key, and secret key queries.
   - $\mathcal{B}$ separates re-encryption key queries into those in a connected component with id^* and those in other components.
   - For each edge in re-encryption key queries in the connected component with id^*, $\mathcal{B}$ simulates the hashes of the public keys in the following manner. $\mathcal{B}$ builds each re-encryption key as an edge in the component and defines the public keys based off of them.
     - Let $H(id^*) = R^*$
     - If the edge is not completing a cycle, then we are connecting a node id to a new node id'. Denote $H(id) = R$, we choose $T \xleftarrow{\$} \mathbb{Z}_N^\times$ and let $H(id') = \frac{R}{T^2}$.
     - If the edge is completing a cycle, then we consider the nodes, in the cycle labeled in order $1, 2, \cdots, k$ with public keys $R_1, \cdots, R_k$. We have T_i translates between R_i and R_{i+1}, and given the previous construction, $R_{i+1} = \frac{R_i}{T_i^2}$.
       We define $T_k = \frac{1}{T_1 T_2 \cdots T_{k-1}}$.
   - For each id not in a connected component, we just generate secret key-public key pairs, $r \xleftarrow{\$} \mathbb{Z}_N^\times$ and $R = r^2$ and we let $H(id) = R$. We answer secret key queries on id with r, public key query with R, and for a re-encryption key query between id and id', we return $\frac{r}{r'}$, a correct translate key.
5. *Challenge* for $\mathcal{A}$. Since we can only encrypt a single bit, $\mathcal{A}$ outputs $(0, 1)$. $\mathcal{B}$ chooses a random $b \xleftarrow{\$} \{0, 1\}$. $\mathcal{B}$ then chooses a random $t \xleftarrow{\$} \mathbb{Z}_N^\times$ so that $\left(\frac{t}{N}\right) = (-1)^b$ and produces the ciphertext $c = t + \frac{R^*}{t}$. $\mathcal{B}$ gives c to $\mathcal{A}$.
6. *Guess* for $\mathcal{A}$. $\mathcal{A}$ outputs a bit b'.
7. *Guess* for $\mathcal{B}$. If $b' = b$, then $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0.

In order for this to work, we need to make the case that we are correctly simulating responses to the public key, private key, and re-encryption key queries. For each re-encryption key query, T between public keys R and R', T must be the ratio of the (arbitrary) square roots of R and R'. That is, $T^2 = \frac{R}{R'}$.

- If R and R' are not part of the connected component with T, then we have direct access to the roots, r and r', and correctly output $T = \frac{r}{r'}$.
- If R and R' are part of the connected component with T, then we have two cases. If T was adding a new node, then R' was chosen so that $R' = \frac{R}{T^2}$, and hence $T^2 = \frac{R}{R'}$.

  If T was connecting a cycle, then we label T as T_k, the translate key between R_1 and R_k, and look at the entire cycle used to define T_k , $R_1, \cdots, R_k$. Expanding T_k^2, we get, as desired,

$$T_k^2 = \frac{1}{T_1^2 T_2^2 \cdots T_{k-1}^2} = \frac{1}{T_1^2} \cdots \frac{1}{T_{k-1}^2} = \frac{R_2}{R_1} \cdot \frac{R_3}{R_2} \cdots \frac{R_k}{R_{k-1}} = \frac{R_k}{R_1}.$$

Since we will never answer secret key queries about the connected component with id^*, we can simulate answers to all of $\mathcal{A}$'s queries.

5.2 Re-encryption of Full Cocks Ciphertexts

Throughout this section, we have shown how to re-encrypt a simplified Cocks ciphertext 2.3. Re-encrypting a full Cocks ciphertext, $\mathbf{c} = (c_1, c_2)$, requires more subtlety than just applying ReEncrypt to each component. In particular, given two public keys $R = h(id)$ and $R' = h(id')$ and the re-encryption key between them $T = r/r'$, we do not know if R or uR is a square or if R' or uR' is a square. So, we must include an extra bit in T: if R and R' are both squares or non-squares, the bit is 0, and the bit is 1 otherwise. The algorithm is detailed in RKGen, below.

Algorithm 5: RKGen(N, u, R, r, R', r')

Input: Public parameters N and u, public key R and corresponding secret key r, public key R' and corresponding secret key r'

Output: Re-encryption key from public key R to R', (T, b)

```
Let b = 0;
if r^2 r'^2 = uRR' then
    Let b = 1;
end
Let T ← r/r';
return (T, b)
```

Now, the re-encryption algorithm for a Cocks ciphertext needs to re-encrypt the component corresponding to the square to the component corresponding to the other square. So, the full proxy re-encryption algorithm is in ReEncrypt_Full.

Algorithm 6: ReEncrypt_Full$(N, u, R, R', (T, b), \mathbf{c} = (c_1, c_2))$

Input: Public parameters N and u, public keys R and R', the re-encryption key from R to R' and bit (T, b), and the full Cocks ciphertext $\mathbf{c}$ encrypted under R.

Output: Full Cocks ciphertext $\mathbf{c}'$ encrypted under R'.

if $b = 0$ **then**
  Let $\mathbf{c}' = (\mathsf{ReEncrypt}(N, R, R', c_1, T), \mathsf{ReEncrypt}(N, uR, uR', c_2, T))$;
end
if $b = 1$ **then**
  Let $\mathbf{c}' = (\mathsf{ReEncrypt}(N, uR, R', c_2, T), \mathsf{ReEncrypt}(N, R, uR', c_1, T))$;
end
return $\mathbf{c}'$

ReEncrypt_Full is correct.

There are four cases to consider depending which of R, uR, R', and uR' are squares.

- R and R' are both squares. RKGen will output $(T = r/r', 0)$. ReEncrypt_Full assigns $\mathbf{c}' = (c_1', c_2')$. We will decrypt with r' using c_1'. Since R and R' are both squares and ReEncrypt is correct, we correctly decrypt c_1' as we decrypt c_1.
- R and R' are both non-squares. RKGen will output $(T = r/r', 0)$, as in the first case. However, when we decrypt with r', we will decrypt the second component, c_2'. Since uR and uR' are both squares in this case and ReEncrypt is correct, we correctly decrypt c_2' with r' as we decrypt c_2 with r.
- R is a non-square and R' is a square. This means uR and R' are both squares and we want to decrypt using the first component, c_1'. RKGen outputs $(T = r/r', 1)$. Because ReEncrypt is correct, when we decrypt c_1' with r' as though we were decrypting c_2 with r.
- R is a square and R' is a non-square. This means R and uR' are both squares and we will want to decrypt using the second component of $\mathbf{c}'$, c_2'. RKGen outputs $(T, 1)$. Because ReEncrypt is correct, when we try to decrypt the output of ReEncrypt_Full, we correctly decrypt c_2' with r' as though we were decrypting c_1 with r.

6 Conclusion

Although it was already known that Cocks ciphertexts had homomorphic properties, we identified the simple structure that makes homomorphic addition possible. We also demonstrated how to use this structure to get bidirectional

proxy re-encryption. We showed how to statistically re-randomize a ciphertext. The structure revealed in the proof of re-randomizing, we believe, is powerful. It shows that the multiplicative group of ciphertexts behaves well with respect to Jacobi symbols and squares. Perhaps these concepts can lead to other applications, like functional encryption, or be applied to other QR schemes, like BGH [BGH07]. In particular, maybe BGH has homomorphic properties as well.

Acknowledgments

I would like to thank Vinod Vaikuntanathan and Shafi Goldwasser for bringing this quesiton to my attention, and Vinod for working with me and for his advice on putting this project together. I would also like to thank Aloni Cohen for his invaluable assistance on the last details of these proofs.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. *Efficient Lattice (H)IBE in the Standard Model*, pages 553–572. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, February 2006.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 21–40. Springer, 2011.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. *Hierarchical Identity Based Encryption with Constant Size Ciphertext*, pages 440–456. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. *Divertible protocols and atomic proxy cryptography*, pages 127–144. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pages 213–229, London, UK, UK, 2001. Springer-Verlag.

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *48th Annual IEEE*

Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings, pages 647–657, 2007.

[BLS13] Dan Boneh, Rio LaVigne, and Manuel Sabin. Identity-based encryption with e^{th} residuosity and its incompressibility. TRUST Conference, poster presentation, 2013.

[Boy13] Xavier Boyen. Attribute-based functional encryption on lattices. In *Theory of Cryptography*, pages 122–142. Springer, 2013.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. *Functional Encryption: Definitions and Challenges*, pages 253–273. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 185–194, New York, NY, USA, 2007. ACM.

[CHT13] Michael Clear, Arthur Hughes, and Hitesh Tewari. *Homomorphic Encryption with Access Policies: Characterization and New Constructions*, pages 61–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, London, UK, UK, 2001. Springer-Verlag.

[GA07] Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, pages 288–306, 2007.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Annual Cryptology Conference*, pages 503–523. Springer, 2015.

[Joy16] Marc Joye. *Identity-Based Cryptosystems and Quadratic Residuosity*, volume 9614, pages 225–254. Springer, 2016.

[RS03] Karl Rubin and Alice Silverberg. *Advances in Cryptology - CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings*, chapter Torus-Based Cryptography, pages 349–365. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[SC09] Jun Shao and Zhenfu Cao. *CCA-Secure Proxy Re-encryption without Pairings*, pages 357–376. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 457–473, 2005.

A Technical details of Convert and Rerand.

Here we go through some technical lemmas for why algorithms Convert and Rerand work. In particular, our goal is to show first that Rerand randomizes the Jacobi symbol of the linear term of a CHT ciphertext, and then that Rerand actually statistically re-randomizes a linear ciphertext. We will need a few lemmas along the way.

Notation and the ring of ciphertexts.

Within $\mathbb{Z}_N$, we let $\mathbb{J}_1$ denote elements with Jacobi symbol 1 and $\mathbb{J}_{-1}$ denote the elements with Jacobi symbol -1. Squares mod N are $\mathbf{QR}_N$ and squares mod p are $\mathbf{QR}_p$.

All of our operations done in this linear-function model will be in the ring $\mathbb{Z}_N[x]/(x^2 - R)$. Let $\mathcal{R}_N = \mathbb{Z}_N[x]/(x^2 - R)$, and $\mathcal{R}_p = \mathbb{Z}_p[x]/(x^2 - R)$. To denote the multiplicative groups of these rings, we write $\mathcal{R}_N^*$ and $\mathcal{R}_p^*$. Note that $|\mathcal{R}_N^*| = |\mathcal{R}_p^*| \cdot |\mathcal{R}_q^*| = (p^2 - 2p + 1)(q^2 - 2q + 1)$. So $\mathcal{R}_N^*$ is the overwhelmingly large fraction or $\mathcal{R}_N$, which has order p^2q^2; almost all elements in $\mathcal{R}_N$ have multiplicative inverses.

A ciphertext $ax + b$ decrypts to 1 if $\left(\frac{ar+b}{N}\right) = 1$ and to -1 if $\left(\frac{ar+b}{N}\right) = -1$. We denote the set of ciphertexts in $\mathcal{R}_N^*$ that decrypt to 1 as $\mathcal{C}_1$ and those that decrypt to -1 to be $\mathcal{C}_{-1}$. Note that when using $\mathcal{C}_1$ and $\mathcal{C}_{-1}$, we are only referring to ciphertexts in the multiplicative group $\mathcal{R}_N^*$.

Working Modulo a Prime p

Lemma 4. *If $ar + b$ is a square mod a prime p, then we can write $ax + b$ as $(cx + d)^2$ in $\mathcal{R}_p^*$. If $ar + b$ is a non-square mod a prime p, then we can take any $u \notin \mathbf{QR}_p$ and write $ax + b$ as $u(cx + d)^2 \pmod{x^2 - R}$.*

Proof. Let $u \in \mathbb{Z}_p^*$, but $u \notin \mathbf{QR}_p$.

All squares in $\mathcal{R}_p^*$ can be written as $(cx+d)^2$. Notice that $u(cx+d)^2$ cannot be a square in $\mathcal{R}_p^*$. Since squares account for exactly half of $\mathcal{R}_p^*$, we can write all non-squares as $u(cx + d)^2$ – u as a member of the group is a bijection as a multiplicative map from $\mathcal{R}_p$ to itself, and thus $u(cx + d)^2$ maps to a different non-square for each square $(cx + d)^2$.

For a contradiction, assume $ar + b \in \mathbf{QR}_p$, but $ax + b$ is not a square. This means $ax+b = u(cx+d)^2 \pmod{x^2-R}$ for some $cx+d$. But, when we evaluate at r, $(ar + b) = u(cr + b)^2$,

$$\left(\frac{u(cr+b)^2}{p}\right) = \left(\frac{u}{p}\right) \cdot \left(\frac{cr+b}{p}\right)^2 = -1 \neq \left(\frac{ar+b}{p}\right).$$

This is a contradiction, and so $ar + b$ being a quadratic residue mod p implies $ax + b$ is a sqaure in $\mathcal{R}_p^*$.

If we let $ar+b \notin \mathbf{QR}_p$. We get that $ax+b = u(cx+d)^2$ for the same reason: if $ax + b = (cx + d)^2$, then evaluation at r results in a contradiction. □

Working In $\mathcal{R}_N$

Lemma 5. *All $ax + b \in \mathcal{C}_1$ are of the form $\frac{(a'x+b')^2}{t}$ where $\left(\frac{t}{N}\right) = 1$. Similarly, all linear function encryptions $ax + b \in \mathcal{C}_{-1}$ are of the form $\frac{(a'x+b')^2}{t}$ where $\left(\frac{t}{N}\right) = -1$.*

Proof. First, assume $ax+b \in \mathcal{C}_1$. We have two cases of decryption to deal with: one where $ar + b$ is a square in both $\mathbb{Z}_p$ and $\mathbb{Z}_q$ and the other where it is a square in neither.

- $ar+b \in \mathbf{QR}_p$ and $\mathbf{QR}_q$. From lemma 4, we know that $ax+b$ is a square in both $\mathcal{R}_p^*$ and $\mathcal{R}_q^*$. This means that mod p, we can write $ax+b = (c_p x+d_p)^2$ and mod q, $ax + b = (c_q x + d_q)^2$. By the Chinese Remainder Theorem (CRT) , we can thus write $ax + b = (cx + d)^2$. Now let $\gamma \in \mathbb{Z}_N^*$,

  $$ax + b = \frac{(\gamma cx + \gamma d)^2}{\gamma^2} = \frac{(a'x + b')^2}{\gamma^2},$$

  and γ^2 is guaranteed to have Jacobi symbol 1 because it is a square.

- $ar + b \notin \mathbf{QR}_p$ or $\mathbf{QR}_q$. Again from lemma 4, $ax + b$ is neither a square in $\mathcal{R}_p^*$ or $\mathcal{R}_q^*$. Again by CRT and lemma 4, $ax + b = u(cx + d)^2$. Now, we can do the same trick as before, to get

  $$ax + b = \frac{(\gamma cx + \gamma d)^2}{\gamma^2/u} = \frac{(a'x + b')^2}{\gamma^2/u}.$$

Notice that γ^2/u is neither a square mod p or mod q and thus has Jacobi symbol 1 mod N.

The case where $ax + b \in \mathcal{C}_{-1}$ is similar. Now, without loss of generality, we can assume $ar + b \in \mathbf{QR}_p$ and $ar + b \notin \mathbf{QR}_q$. Lemma 3 tells us that mod p, $ax + b = 1 \cdot (c_p x + d_p)^2 \mod p$ and $ax + b = u_q(c_q x + d_q)^2 \mod q$. By the Chinese Remainder Theorem, we can find a $t \in \mathbb{Z}_N$ so that $t \equiv 1 \mod p$ and $t \equiv 1/u_q \mod q$, as well as $a' \equiv c_p \mod p$ and $c_q \mod q$ and $b' \equiv d_p \mod p$ and $d_q \mod q$. We can rewrite

$$ax + b \equiv_N \frac{(a'x + b')^2}{t} \pmod{x^2 - R}.$$

Now we have $\left(\frac{t}{N}\right) = \left(\frac{1}{p}\right) \cdot \left(\frac{u_q}{q}\right) = -1$ as desired. □

A.1 Proving lemma 1

Now we can prove lemma 1, which shows that algorithm Convert terminates in polynomial time.

Proof of Lemma 1

This proof will be re-writing $ax + b$ in terms of being a square in $\mathcal{R}_p$ and $\mathcal{R}_q$ or not.

First, we know that we can rewrite $ax + b = \frac{(a'x+b')^2}{t'}$ where $\left(\frac{t'}{N}\right) = \left(\frac{ar+b}{N}\right)$ by lemma 5. Now, the term in our probability $(ax + b)\left(\frac{(cx+d)^2}{t}\right)$ becomes $\frac{((a'x+b')(cx+d))^2}{tt'}$. Since $a'x + b'$ will also have an inverse in $\mathcal{R}_N$, and we are choosing $cx + d$ at random, we are just as likely to choose c, d as we are to choose c', d' where $(c'x + d') = (a'x + b')(cx + d) \pmod{x^2 - R}$. The term we are trying to bound the probability on is $\frac{(cx+d)}{t}$ where $\left(\frac{t'}{N}\right) = \left(\frac{ar+b}{N}\right)$.

Let $m = \left(\frac{ar+b}{N}\right)$, the decryption of our ciphertext. We now have that the probability we are looking at is

$$\Pr_{c,d \xleftarrow{\$} \mathbb{Z}_N^*, t \xleftarrow{\$} \mathbb{J}_1}\left[\left(\frac{e}{N}\right) = 1 \text{ where } ex + f = (ax + b)\frac{(cx + d)^2}{t}\right]$$
$$= \Pr_{c,d \xleftarrow{\$} \mathbb{Z}_N^*, t \xleftarrow{\$} \mathbb{J}_m}\left[\left(\frac{e}{N}\right) = 1 \text{ where } ex + f = \frac{(cx + d)^2}{t}\right].$$

Now, we expand $\frac{1}{t} \cdot (cx + d)^2 \pmod{x^2 - R}$, our linear term is just $\frac{2}{t}cd$. We

can analyze

$$\Pr_{c,d\xleftarrow{\$}\mathbb{Z}_N^*,t\xleftarrow{\$}\mathbb{J}_m}\left[\left(\frac{2cd/t}{N}\right)=1\right]=\sum_{\gamma\in\mathbb{J}_1}\Pr_{c,d\xleftarrow{\$}\mathbb{Z}_N^*,t\xleftarrow{\$}\mathbb{J}_m}\left[\frac{2cd}{t}=\gamma\right]$$
$$=\sum_{\gamma\in\mathbb{J}_1}\sum_{T\in\mathbb{J}_m}\Pr_{t\in\mathbb{J}_m}[t=T]\sum_{C\in\mathbb{Z}_N^*}\Pr_{c\in\mathbb{Z}_N^*}[c=C]\cdot\Pr_{d\in\mathbb{Z}_N^*}\left[d=\frac{T\gamma}{2C}\right]$$
$$=\sum_{\gamma\in\mathbb{J}_1}\sum_{T\in\mathbb{J}_m}\frac{2}{\phi(N)}\sum_{A\in\mathbb{Z}_N^*}\frac{1}{\phi(N)}\cdot\frac{1}{\phi(N)}$$
$$=\frac{\phi(N)}{2}\cdot\left(\frac{\phi(N)}{2}\frac{2}{\phi(N)}\right)\left(\phi(N)\cdot\frac{1}{\phi(N)}\right)\cdot\frac{1}{\phi(N)}=\frac{1}{2}.$$

□

Lemma 1 means that any linear ciphertext $ax+b$ decrypted by $\left(\frac{ar+b}{N}\right)$ can be converted into a Cocks ciphertext by algorithm 1 Convert , which employs a re-randomizing algorithm 2 Rerand.

A.2 Proof of lemma 2

We now have enough machinery to prove lemma 2: Rerand statistically re-randomizes ciphertexts.

First, recall the distribution $\mathcal{D}_{ax+b}$, defined on an element in $\mathcal{R}_N$.

$$\mathcal{D}_{ax+b}:=\left\{(ax+b)\frac{(cx+d)^2}{t}:cx+d\xleftarrow{\$}\mathcal{R}_N^*,t\xleftarrow{\$}\mathbb{J}_1\right\},$$

which is statistically close to the output of Rerand. We say statistically close becauase c and d are chosen randomly from $\mathbb{Z}_N$. With overwhelming probability, $cx+d$ will have a multiplicative inverse in $\mathcal{R}_N^*$. So, we will prove that $\mathcal{D}$ randomizes ciphertexts that have inverses, and because the distribution from $\mathcal{D}$ is statistically close to the actual output of Rerand and the set of ciphertexts that have inverses is statistically close to the set of all ciphertexts, Rerand statistically re-randomizes a ciphertext.

Proof of Lemma 2 Recall that our goal is to show that if $ax+b\in\mathcal{C}_1$, then

$$\mathcal{D}\equiv\left\{ex+f\text{ where }ex+f\xleftarrow{\$}\mathcal{C}_1\right\},$$

and if $ax+b\in\mathcal{C}_{-1}$, then

$$\mathcal{D}\equiv\left\{ex+f\text{ where }ex+f\xleftarrow{\$}\mathcal{C}_{-1}\right\}.$$

We will define an alternative distribution. Let $m=\left(\frac{ar+b}{N}\right)$, and let $\mathcal{D}'$ be

$$\mathcal{D}':=\left\{\frac{(cx+d)^2}{t}:cx+d\xleftarrow{\$}\mathcal{C}_1,t\xleftarrow{\$}\mathbb{J}_m\right\}.$$

We will first show that $\mathcal{D} \equiv \mathcal{D}'$. As in the proof of theorem 1, we can rewrite $ax+b = \frac{(a'x+b')^2}{t'}$ for some t with Jacobi symbol 1 using lemma 5. So, expanding our output from $\mathcal{D}$, $(ax+b) = \frac{((a'x+b')(cx+d))^2}{tt'}$. Since we are choosing $(cx+d)$ from the multiplicative group $\mathcal{R}_N^*$ at random and $a'x+b'$ is also in $\mathcal{R}_N^*$, the probability of $\mathcal{D}$ chooses $(cx+d)$ for its output is the same as the probability $\mathcal{D}$ chooses $(a'x+b')(cx+d) \pmod{x^2-R}$. Since $\left(\frac{tt'}{N}\right) = \left(\frac{t}{N}\right)$, the probability $\mathcal{D}$ outputs a specific $\frac{(cx+d)^2}{t}$ with $\left(\frac{t}{N}\right) = m$ is equivalent to the probability $\mathcal{D}'$ outputs that element in $\mathcal{R}_N^*$.

Our goal now is to show that $\mathcal{D}'$ outputs, uniformly, a ciphertext that decrypts to $\left(\frac{ar+b}{N}\right)$. Note that, by counting, $|\mathcal{C}_1| = |\mathcal{R}_N^*|/2$, since exactly half of the elements in $\mathcal{R}_N^*$ are squares divided by elements of Jacobi symbol 1 and the other half are squares divided by elements of Jacobi symbol -1. This means the probability that a uniform distribution on $\mathcal{C}_1$ outputs $ex+f$ is $\frac{2}{|\mathcal{R}_N^*|}$.

Let $ex+f \in \mathcal{C}_1$. We will analyze the probability $\mathcal{D}'$ outputs $ex+f$. By lemma 5, $ex+f = \frac{(e'x+f')^2}{\gamma}$ where $\gamma \in \mathbb{J}_1$. We have two cases, $\gamma \in \mathbf{QR}_N$ and $\gamma \notin \mathbf{QR}_N$:

- $\gamma \in \mathbf{QR}_N$. The probability that $\mathcal{D}'$ outputs $ex+f$ is the probability that $\frac{(cx+d)^2}{t} = \frac{(e'x+f')^2}{\gamma}$ when we randomly choose $cx+d \in \mathcal{R}_N^*$ and $t \in \mathbb{J}_1$:

$$\begin{aligned}
&\Pr_{cx+d \xleftarrow{\$} \mathcal{R}_N^*, t \xleftarrow{\$} \mathbb{J}_1}\left[\frac{(cx+d)^2}{t} = \frac{(e'x+f')^2}{\gamma}\right] \\
&= \sum_{T \in \mathbb{J}_1} \Pr_{t \xleftarrow{\$} \mathbb{J}_1}[t = T] \cdot \Pr_{cx+d \xleftarrow{\$} \mathcal{R}_N^*}\left[(cx+d)^2 = \frac{T}{\gamma}(e'x+f')^2\right].
\end{aligned}$$

  Notice that this equation only has a solution in $cx+d$ if $t \in \mathbf{QR}_N$. Otherwise, we are trying to solve $(cx+d)^2 = \frac{t}{\gamma}(e'x+f')^2 \pmod{x^2-R}$ when $(cx+d)^2$ is a square, but $\frac{t}{\gamma}(e'x+f')^2$ is not.

  Now, assuming that $\frac{T}{\gamma} \in \mathbf{QR}_N$, we can let $k^2 = \frac{T}{\gamma}$ and rewrite $k^2(e'x+f'^2) = (ke'x+kf')^2 = (\hat{e}x+\hat{f})^2$. We are looking for the probability that a random $cx+d$ is a solution to $(cx+d)^2 = (\hat{e}x+\hat{f})^2 \pmod{x^2-R}$. We need to know how many solutions there are to this. We will use CRT.

  Mod p, there are exactly two solutions $cx+d \in \mathcal{R}_p^*$ to $(\hat{e}x+\hat{f})^2$: at least two because $\pm(\hat{e}x+\hat{f})$ are both solutions, and no more than two because the size of squares is exactly half of $\mathcal{R}_p^*$. Mod q there are also exactly two solutions. This means, mod N there are 4 total solutions. Now, when we

continue to analyze this probability, we have

$$\begin{aligned}
&\sum_{T \in \mathbb{J}_1} \Pr_{t \xleftarrow{\$} \mathbb{J}_1}[t = T] \cdot \Pr_{cx+d \xleftarrow{\$} \mathcal{R}_N^*}\left[(cx+d)^2 = \frac{T}{\gamma}(e'x+f')^2\right] \\
&= \sum_{T \in \mathbf{QR}_N} \frac{2}{\phi(N)} \cdot \frac{4}{|\mathcal{R}_N^*|} \\
&= \left(\frac{\phi(N)}{4} \cdot \frac{2}{\phi(N)}\right) \cdot \frac{4}{|\mathcal{R}_N^*|} \\
&= \frac{2}{|\mathcal{R}_N^*|}.
\end{aligned}$$

So, in this case, the distribution $\mathcal{D}'$ is the same as uniform.

- $\gamma \notin \mathbf{QR}_N$. We can use the same analysis tricks, except T must also not be in $\mathbf{QR}_N$, but still must have Jacobi symbol 1. So,

$$\begin{aligned}
&\Pr_{cx+d \xleftarrow{\$} \mathcal{R}_N^*, t \xleftarrow{\$} \mathbb{J}_{-1}}\left[\frac{(cx+d)^2}{t} = \frac{(e'x+f')^2}{\gamma}\right] \\
&= \sum_{T \notin \mathbf{QR}_N} \frac{2}{\phi(N)} \cdot \frac{4}{|\mathcal{R}_N^*|} \\
&= \left(\frac{\phi(N)}{4} \cdot \frac{2}{\phi(N)}\right) \cdot \frac{4}{|\mathcal{R}_N^*|} \\
&= \frac{2}{|\mathcal{R}_N^*|}.
\end{aligned}$$

The case where $ax+b, ex+f \in \mathcal{C}_{-1}$ is proved in exactly the same manner. □