Identity-Based Encryption with e^{th} Residuosity and its Incompressibility

Dan Boneh
Stanford University
dabo@cs.stanford.edu

Rio LaVigne
Stanford University
riol@stanford.edu

Manuel Sabin†
California State University Sacramento
msabin27@gmail.com

Abstract

Identity Based Encryption schemes were first constructed with, and often have been since, bilinear mappings. There were, however, some successful attempts to construct IBE schemes based on more traditional number theoretic problems. The first of which was the Cocks scheme, which based its security on the quadratic residuosity problem. In this paper, the Cocks scheme is generalized to e^{th} residuosity so that more than one bit may be encrypted in a message. However, like the Cocks scheme, it suffers from massive ciphertext expansion. We consequently propose a space efficient variation, although it is found to not be secure and we detail an attack for any attempt of compression that utilizes this variation or any generalization of it.

1 Introduction

Since Shamir first introduced the concept of an Identity Based Encryption (IBE) scheme in 1984 [7], the intuitive flexibility and convenience of using an arbitrary string as a public key made many practical applications of IBE quickly apparent. However, it was not until 2001 that the Boneh-Franklin scheme [2] introduced the first working IBE system using bilinear pairings. While bilinear pairings provide efficiency in time and space, they rely on a relatively new computational hardness assumption.

Soon after the Boneh-Franklin scheme, Cocks introduced an elegant IBE scheme based on quadratic residuosity (QR) [3]. While the scheme is not very space efficient and appears to lack structure for extensions such as Hierarchical IBE [5], it succeeds in basing IBE on the very traditional QR assumption. Similarly, Boneh, Gentry, and Hamburg proposed a scheme [1] without pairings that is space efficient and based on ternary quadratic forms. Both use a RSA composite N and while the Cocks scheme encrypts a single bit at a time, thus expanding an ℓ-bit plaintext to a ciphertext of size $2\ell \cdot \log_2 N$, the latter scheme would only expand it to about a size of $\ell + \log_2 N$.

When considering the Cocks scheme, it is not only natural to entertain the possibility of generalizing it to e^{th} residuosity but also practical as it might allow more than one bit to be encrypted at a time. We propose and prove semantically secure a scheme that uses e^{th} residuosity for a prime e. As expected, more than one bit may now be encrypted, but not efficiently. About e elements of $\mathbb{Z}_N$ must be sent to encrypt a $\log_2 e$ message. That is, for an ℓ-bit message, the ciphertext is about the size $2^{\ell \cdot \log_2 N}$. In this way generalizing past $e = 2$, the Cocks scheme, quickly become far less efficient unless a our scheme is modified.

Because of its inefficiency, we propose a variation of the scheme that has a constant length of ciphertext which, moreover, only comprises 3 elements of $\mathbb{Z}_N$. If it were secure, this would mean that an ℓ-bit message would always have a ciphertext of size $3 \cdot \log_2 N$ regardless of ℓ. However, we show that not only is it insecure, but that any compression of the ciphertext using the same technique is insecure, even if it were variable length.

†Supported by TRUST and NSF

2 Definitions

2.1 Trivial and Degenerate Roots of Unity

Although we may often consider e^{th} roots of unity in their usual complex form, in $\mathbb{Z}_N$ they may actually all exist within the ring itself. In fact, if we require that e divides $p-1$ and $q-1$, then it can be shown that all e roots of unity will be in respectively both $\mathbb{Z}_p$ and $\mathbb{Z}_q$. CRTing these yields all e^2 roots of unity in $\mathbb{Z}_N$. Note that a primitive root of unity in this case is still one that simply generates e roots of unity rather than, impossibly, all e^2.

Our scheme requires that one of these roots of unity is published, but it should be forewarned that some roots unity would allow an adversary to factor N. These are called degenerate roots of unity. First, the trivial root of unity is simply the unity, 1, itself. However, any other root of unity that is congruent to 1 modulo either p or q is degenerate. Say, for example, that a root of unity, ζ is congruent to 1 modulo p, then $\gcd(N, \zeta - 1) = p$ and the factorization is known.

This distinction is crucial to set up the scheme properly.

2.2 Power Residue Symbols and the ER Assumption

We denote the eth power residue symbol of some integer x modulo a positive integer N as $\left(\frac{x}{N}\right)_e$. For the rest of the paper we will refer to the eth power residue symbol when we say symbol; the modulus being used should be apparent within the context. We will not go into detail on the symbol's properties except that it will always result in an eth root of unity, for further details see its treatment in [4] and its derivation in [6, Chapter 14]. Also, we assume that taking symbols is efficient and can be done without knowing the factorization of the modulus used for the symbol; a polynomial algorithm for computing the symbol is given by Squirrel [8].

We denote $\mathrm{ER}(N)$ as the set of eth residues in $\mathbb{Z}_N$, and $\mathrm{PR}(N)$ as the set of elements in $\mathbb{Z}_N$ that have a symbol of 1. Note that $\mathrm{ER}(N)$ is a subset of $\mathrm{PR}(N)$.

Intuitively the eth residuosity assumption is that, given a number that has a symbol of 1, it is hard to tell if that number is an eth residue or not. We formally state this below.

e^{th} Residuosity (ER) Assumption: For a PPT algorithm $\mathrm{RSAgen}(\lambda)$ that generates two equal size primes p, q

- Let $\mathcal{P}_{\mathrm{ER}}(\lambda)$ be the distribution:

$$(p, q) \xleftarrow{\mathrm{R}} \mathrm{RSAgen}(\lambda),\ N \leftarrow pq,\ v \xleftarrow{\mathrm{R}} \mathrm{ER}(N),\ \mathrm{output}(N, v)$$

- Let $\mathcal{P}_{\mathrm{ENR}}(\lambda)$ be the distribution:

$$(p, q) \xleftarrow{\mathrm{R}} \mathrm{RSAgen}(\lambda),\ N \leftarrow pq,\ v \xleftarrow{\mathrm{R}} \mathrm{PR}(N) \backslash \mathrm{ER}(N),\ \mathrm{output}(N, v)$$

Define the ER advantage of adversary $\mathcal{A}$ against RSAgen as

$$\mathrm{ERAdv}_{\mathcal{A},\mathrm{RSAgen}}(\lambda) = \left| \mathrm{P}[\mathcal{A}(N, v) = 1 : (N, v) \xleftarrow{\mathrm{R}} \mathcal{P}_{\mathrm{ER}}(\lambda)] - \mathrm{P}[\mathcal{A}(N, v) = 1 : (N, v) \xleftarrow{\mathrm{R}} \mathcal{P}_{\mathrm{ENR}}(\lambda)] \right|$$

If $\mathrm{ERAdv}_{\mathcal{A},\mathrm{RSAgen}}(\lambda) \leq \epsilon$ where ϵ is negligible for all PPT algorithms $\mathcal{A}$, then we say that the ER assumption holds for RSAgen.

In general the ER assumption is believed to hold. However, our scheme cannot use it exactly in this form.

2.3 Modified ER Assumption

For the setup of our scheme it is necessary to publish a non-degenerate root of unity, ζ, since finding a non-trivial root of unity is hard without the factorization of N. This, however, is new information for the adversary and we must slightly modify the ER assumption to accommodate the new information while still believe it to be secure.

If we let $\mathcal{Z}$ be the set of non-degenerate, nontrivial roots of unity in $\mathbb{Z}_N$, the definitions follow similarly.

Modified e^{th} Residuosity (MER) Assumption: For a PPT algorithm RSAgen(λ) that generates two equal size primes p, q

- Let $\mathcal{P}'_{\text{ER}}(\lambda)$ be the distribution:

$$(p, q) \xleftarrow{\text{R}} \text{RSAgen}(\lambda),\ N \leftarrow pq,\ v \xleftarrow{\text{R}} \text{ER}(N),\ \zeta \xleftarrow{\text{R}} \mathcal{Z},\ \text{output}(N, v, \zeta)$$

- Let $\mathcal{P}'_{\text{ENR}}(\lambda)$ be the distribution:

$$(p, q) \xleftarrow{\text{R}} \text{RSAgen}(\lambda),\ N \leftarrow pq,\ v \xleftarrow{\text{R}} \text{PR}(N)\backslash\text{ER}(N),\ \zeta \xleftarrow{\text{R}} \mathcal{Z},\ \text{output}(N, v, \zeta)$$

Define the MER advantage of adversary $\mathcal{A}$ against RSAgen as

$$\text{MERAdv}_{\mathcal{A},\text{RSAgen}}(\lambda) = \left|\text{P}[\mathcal{A}(N, v, \zeta) = 1 : (N, v, \zeta) \xleftarrow{\text{R}} \mathcal{P}'_{\text{ER}}(\lambda)] - \text{P}[\mathcal{A}(N, v, \zeta) = 1 : (N, v, \zeta) \xleftarrow{\text{R}} \mathcal{P}'_{\text{ENR}}(\lambda)]\right|$$

Again, if $\text{MERAdv}_{\mathcal{A},\text{RSAgen}}(\lambda)$ is negligible for all PPT algorithms $\mathcal{A}$, then we say that the MER assumption holds for RSAgen. While this is not exactly the traditional ER assumption, we believe it to hold. Note that the MER assumption is exactly the ER assumption except that the adversary is given a random non-degenerate e^{th} root of unity along with the usual N and v.

3 The Scheme

3.1 Setup (λ)

First decide upon a prime e for our e^{th} residuosity scheme. Note that any e^{th} root of unity will be primitive since e is prime; this will be found as useful in proving security. Using the security parameter λ, generate two RSA primes, p and q, such that $e|p-1$ and $e|q-1$. Let the public parameters, PP, be $N = pq$, e, a non-degenerate primitive e^{th} root of unity in $\mathbb{Z}_N$, call it ζ, and a publicly available hash, $\mathcal{H}$, that maps an arbitrary binary string to an eth residue in $\mathbb{Z}_N^*$. And let the master secret, MSK, be p and q.

$$PP \leftarrow (N, e, \zeta, \mathcal{H})$$
$$MSK \leftarrow (p, q)$$

3.2 KeyGen ($MSK, \mathcal{H}, ID$)

Only a trusted Private Key Generator (PKG) can perform this section of the scheme as only they have MSK. Upon a user's request for their secret key, sk, the PKG will verify the users ID and then set sk to the eth root of $v = \mathcal{H}$. We assume that knowing the factorization of N allows for this to be done by the PKG.

$$sk \leftarrow r = v^{\frac{1}{e}}$$

3.3 Encrypt (PP, ID, m')

When sending an encrypted message to a user with an identity, ID, there public key must first be derived. This can be done as $pk \leftarrow v = \mathcal{H}(ID)$.

A message, m', may be chosen as any integer from 0 to $e-1$. This, however, must be encoded as $m = \zeta^{m'}$ to be compatible with the scheme. Encryption proceeds as follows.

$$f \xleftarrow{\text{R}} \mathbb{Z}_N[x] \text{ of degree } e-1$$
$$g \leftarrow f^e (\text{mod } x^e - v)$$

g is now a $e-1$ degree polynomial which can be expressed as

$$g(x) = \sum_{i=0}^{e-1} a_i x^i$$

Note that $f^e(x) = g(x) + h(x)(x^e - v)$ for some $h(x) \in \mathbb{Z}_N[x]$ by definition of a polynomial modulus. This means that $f^e(r) = g(r) + h(r)(r^e - v) = g(r) + h(r)(v - v) = g(r)$. And so knowing the secret key, r, allows us to reveal a point of $f^e(x)$ which is otherwise not possible by just knowing $g(x)$ and not an eth root of v. This will be useful during decryption.

Now, choose a transport key $t \xleftarrow{R} \mathbb{Z}_N^*$. Since t is a random unit, then $\left(\frac{t}{N}\right)_e$ will be an entirely random e^{th} root of unity. Thus, multiply our message, $m = \zeta^{m'}$, by $\left(\frac{t}{N}\right)_e$ also yields an entirely random e^{th} root of unity. This is how we encrypt m but we must also give enough information for the secret key holder to decrypt. To do this we also send each coefficient of $g(x)$ divided by t and send the full ciphertext

$$CT = \left[\frac{a_o}{t}, \frac{a_1}{t}, \frac{a_2}{t}, \ldots, \frac{a_{e-1}}{t}, m\left(\frac{t}{N}\right)_e\right]$$

Intuitively, sending $\frac{g(x)}{t}$ embeds a "hint" about what t is so that m can be recovered, but we will later argue that without knowing an eth root of v this hint cannot be unlocked.

3.4 Decrypt $(sk, CT = [c_o, c_1, c_2, \ldots, c_{e-1}, d])$

When a user receives a message that was encrypted using their public key $pk \leftarrow v$, they may decrypt it using their corresponding secret key $sk \leftarrow r = v^{\frac{1}{e}}$. In general, they would like to find a z such that $\left(\frac{z}{N}\right)_e = \left(\frac{t}{N}\right)_e^{-1}$ as this would allow them to easily recover the message as $d\left(\frac{z}{N}\right)_e = d\left(\frac{t}{N}\right)_e^{-1} = \left(m\left(\frac{t}{N}\right)_e\right)\left(\frac{t}{N}\right)_e^{-1} = m\left(\left(\frac{t}{N}\right)_e\left(\frac{t}{N}\right)_e^{-1}\right) = m$.

We find such a z by first reconstructing the polynomial $\frac{g(x)}{t}$ from the coefficients given in the ciphertext.

$$\frac{g(x)}{t} \leftarrow \sum_{i=0}^{e-1} c_i x^i$$

We may now set z equal to $\frac{g(r)}{t}$. Remember that r, being the eth root v, makes it so that $g(r) = f^e(r)$ as alluded to when discussing encryption. We now see that $z = t^{-1}g(r) = t^{-1}f^e(r)$ and that z has our desired property:

$$\left(\frac{z}{N}\right)_e = \left(\frac{t^{-1}f^e(r)}{N}\right)_e = \left(\frac{t^{-1}}{N}\right)_e \left(\frac{f^e(r)}{N}\right)_e = \left(\frac{t}{N}\right)_e^{-1} \left(\frac{f(r)}{N}\right)_e^e = \left(\frac{t}{N}\right)_e^{-1}$$

So, this z can be easily used to recover m as was described above, and thus recover m'. In this way, the coefficients of $\frac{g(x)}{t}$ contain enough of a "hint" about t to calculate $\left(\frac{t}{N}\right)_e^{-1}$ if the secret key is known.

4 Security

In the encryption step, there are two random selections: one of a polynomial in $\mathbb{Z}_N[x]$, and the other a transport key in $\mathbb{Z}_N$. Besides the public parameters and the message, these entirely decide CT. However, there are other transport key/function pairs that yield the same exact ciphertext, where ciphertext from now on will be meant as the coefficients of $\frac{g}{t}$ - i.e. all of CT except the last piece, d, that contains the encrypted message.

Of course, for a given ciphertext, every transport key that could possibly yield it must have the same symbol, otherwise decryption is not reliable. That is, the ciphertext would not be enough of a hint to find the transport key's symbol if it is not uniquely determined by the ciphertext. In fact, this idea will motivate our proof of security as we consider these transport key/function pairs with v as an eth residue and as an eth nonresidue.

First, we consider what the transport key/function pairs actually look like. When encrypting a message, a function f is randomly chosen as an $e-1$ degree polynomial from $\mathbb{Z}_N[x]$ as well as transport key t from $\mathbb{Z}_N^*$. We now consider alternate transport keys, t', and functions, f', as follows: $t'_{ij} \equiv tv^i \pmod{p} \equiv tv^j \pmod{q}$ and, correspondingly, $f'_{ij} \equiv f * x^i \pmod{p} \equiv f * x^j \pmod{q}$, for $i, j = 0, 1, ..., e-1$. It should be noted that f'_{ij} can be thought of as an $e-1$ degree polynomial from $\mathbb{Z}_N[x]$ even though it's degree is often greater than $e-1$; we may instead think of f'_{ij} as f'_{ij} (mod$x^e - v$) which is of degree $e-1$ and is one that could have been selected when originally selecting f. Thinking of f'_{ij} as degree $e-1$ allows us to acknowledge it as a possible selection during encryption yet doesn't affect the analysis below.

We now consider what ciphertext the pair f'_{ij}, t'_{ij} and consider the ciphertext, just modulo p for now, it would yield according to our encryption scheme.

$$\frac{(f'_{ij})^e}{t'_{ij}} \equiv \frac{(f * x^i)^e}{tv^i} \pmod{p} = \frac{f^e * (x^e)^i}{tv^i} \equiv \frac{g * v^i}{tv^i} \pmod{x^e - v} = \frac{g}{t} = CT$$

This shows that, modulo p, the pair f'_{ij}, t'_{ij} yields the exact same ciphertext as the pair f, t. A similar analysis can be done modulo q to see that f'_{ij}, t'_{ij} produce the same ciphertext both modulo p and q, and so they must also produce the same ciphertext modulo N. Thus f'_{ij}, t'_{ij} pairs must yield the same ciphertext as f, t do modulo N, and so a given ciphertext has e^2 different possible transport key/function pairs that could have been chosen during encryption to produce it. But, again, since the symbol of the transport key is used to encrypt the message, each of these possible transport keys must have the same symbol so that the message decrypts reliably.

$$\left(\frac{t'_{ij}}{N}\right)_e = \left(\frac{t'_{ij}}{p}\right)_e \left(\frac{t'_{ij}}{q}\right)_e = \left(\frac{tv^i}{p}\right)_e \left(\frac{tv^j}{q}\right)_e = \left(\frac{t}{p}\right)_e \left(\frac{t}{q}\right)_e \left(\frac{v^i}{p}\right)_e \left(\frac{v^j}{q}\right)_e = \left(\frac{t}{N}\right)_e \left(\frac{v}{p}\right)_e^i \left(\frac{v}{q}\right)_e^j$$

And since v is an eth residue by construction, $\left(\frac{v}{p}\right)_e = \left(\frac{v}{q}\right)_e = 1$. So, $\left(\frac{t'_{ij}}{N}\right)_e = \left(\frac{t}{N}\right)_e$ as was needed. And so, even though we cannot recover the exact transport key used as there is no unique one for a ciphertext, we can uniquely recover the symbol of the transport key which is all that is needed for decryption. However we did use the fact that v is an eth residue to determine the symbol. Considering v as an eth nonresidue will give us a way to security.

4.1 v as an eth Nonresidue

While our decryption process requires that an eth root of v is used, the encryption process does not even require that an eth root of v exists. Because of this, ciphertext may be created with any v, whether it is an eth residue or nonresidue.

Now considering v as an eth nonresidue, we will also stipulate that $\left(\frac{v}{N}\right)_e = 1$ which is also necessary when discussing the eth residuosity problem.

Now that it is established that ciphertexts can be created with such a v, we again consider the transport key/function pairs that all yield a given ciphertext. They may be constructed in the same way as before and it can be shown in an identical fashion that, for an f, t pair, the corresponding f'_{ij}, t'_{ij} pairs yield the same

ciphertext, as none of the analysis required v to be an eth residue. However, our analysis of the symbol of the transport keys did use the fact that v was an eth residue to show that $\left(\frac{t'_{ij}}{N}\right)_e = \left(\frac{t}{N}\right)_e$. However, without this fact the most that can be said is that $\left(\frac{t'_{ij}}{N}\right)_e = \left(\frac{t}{N}\right)_e \left(\frac{v}{p}\right)_e^i \left(\frac{v}{q}\right)_e^j$.

Since $\left(\frac{v}{N}\right)_e = 1$, it can be said that $\left(\frac{v}{p}\right)_e = \left(\frac{v}{q}\right)_e^{-1}$. This means that both $\left(\frac{v}{p}\right)_e$ and $\left(\frac{v}{q}\right)_e$ are e^{th} roots of unity not equal to 1. Moreover, they are primitive roots of unity since e is prime. Since they are both primitive, then $\left(\frac{v}{p}\right)_e^i \left(\frac{v}{q}\right)_e^j$ will result in all possible e^{th} roots of unity exactly e times each as i and j independently vary. This not only means that sometimes $\left(\frac{t'_{ij}}{N}\right)_e \neq \left(\frac{t}{N}\right)_e$, but that $\left(\frac{t'_{ij}}{N}\right)_e$ will uniformly be a random e^{th} root of unity if i and j are random.

We already knew that the transport key that was used for a ciphertext could not be recovered as it was not unique, but this tells us that we cannot even recover the symbol of the transport key. In fact, the symbol could have uniformly been anything, and so m also looks like a uniformly chosen e^{th} root of unity. The ciphertext simply does not contain enough information to recover the correct symbol of the transport key more than one-e^{th} of the time.

4.2 Proving Semantic Security

We are now ready to prove semantic security by reducing breaking the MER problem to breaking semantic security. That is, we will assume that there is a semantic security adversary, $\mathcal{B}$, and leverage them to create a MER adversary, $\mathcal{A}$.

Assume that when the semantic security adversary, $\mathcal{B}$, produces to equal length messages, m_0 and m_1, he can distinguish which of the randomly selected messages was returned encrypted to him with probability $\frac{1}{2} + \epsilon$ where ϵ is non-negligible. That is, $\mathcal{B}$ breaks semantic security by definition.

For an MER adversary, $\mathcal{A}$, now, when they are given an RSA composite N, a non-degenerate root of unity ζ in $\mathbb{Z}_N$, and a v in $\mathbb{Z}_N$ with symbol 1, they can give all of these parameters to $\mathcal{B}$. Then, $\mathcal{A}$ may act as a semantic security challenger against $\mathcal{B}$. As $\mathcal{B}$ attempts to break semantic security they will output m_0 and m_1 to which $\mathcal{A}$ will choose $b \xleftarrow{\mathrm{R}} \{0, 1\}$ and encrypt m_b to return to $\mathcal{B}$. $\mathcal{A}$ may then output 1 whenever $\mathcal{B}$ wins the semantic security game.

If v is in fact an eth residue, then our scheme is set up properly and $\mathcal{B}$, having the parameters, is simply playing a properly setup semantic security game, in which it can win with non-negligible probability. This is to say that $(N, v, \zeta) \xleftarrow{\mathrm{R}} \mathcal{P}'_{ER}(\lambda)$ and that $\mathcal{A}$ will output 1 with the same probability that $\mathcal{B}$ is correct in its guess: $\mathrm{P}[\mathcal{A}(N, v, \zeta) = 1 : (N, v, \zeta) \xleftarrow{\mathrm{R}} \mathcal{P}'_{\mathrm{ER}}(\lambda)] = \frac{1}{2} + \epsilon$.

However, if v is not an eth residue, then the scheme is not properly set up. We have already seen above that in this scenario ciphertexts do not contain enough information and that the encrypted message then looks entirely like a uniformly chosen e^{th} root of unity. $\mathcal{B}$ can no longer break semantic security when the scheme is set up incorrectly like this as the ciphertext does not hold the information to do so. And since it is entirely uniform, $\mathcal{B}$ has their best chances of winning by guessing randomly at m_0 or m_1. This is to say that $(N, v, \zeta) \xleftarrow{\mathrm{R}} \mathcal{P}'_{ENR}(\lambda)$ and that $\mathcal{A}$ will output 1 with the same probability that $\mathcal{B}$ is correct in its guess: $\mathrm{P}[\mathcal{A}(N, v, \zeta) = 1 : (N, v, \zeta) \xleftarrow{\mathrm{R}} \mathcal{P}'_{\mathrm{ENR}}(\lambda)] = \frac{1}{2}$. And so by definition

$$\mathrm{MERAdv}_{\mathcal{A},\mathrm{RSAgen}}(\lambda) = \left|\mathrm{P}[\mathcal{A}(N, v, \zeta) = 1 : (N, v, \zeta) \xleftarrow{\mathrm{R}} \mathcal{P}'_{\mathrm{ER}}(\lambda)] - \mathrm{P}[\mathcal{A}(N, v, \zeta) = 1 : (N, v, \zeta) \xleftarrow{\mathrm{R}} \mathcal{P}'_{\mathrm{ENR}}(\lambda)]\right|$$

$$= \left|\left(\frac{1}{2} + \epsilon\right) - \frac{1}{2}\right| = \epsilon$$

As we've established ϵ is non-negligible, so the MER problem is broken as we are able to distinguish v a residue or nonresidue. Thus, if we believe the MER problem to be hard, then breaking semantic security of this scheme must also be hard.

5 Comparison to Cocks IBE

While most of the Cocks IBE can be rephrased from the original paper [3] to match our generalization, there are some aspects that differ.

First, the Cocks scheme does does not necessitate that a user's public key is a quadratic residue but instead puts restrictions on p and q so that either the public key or the negative of it is a quadratic residue. This creates ambiguity for the encryptor and, as such, two ciphertexts must be sent using both possible keys. Our scheme does not use this strategy. Instead our public key must be an eth residue for our scheme to work and, as such, proves security in a similar but slightly different manner. This has the advantage of eliminating ambiguity so only one ciphertext needs to be sent.

Also, the Cocks scheme uses the transport key's symbol itself as the message, whereas we use it to encrypt the message. It would not be difficult to change our method to use this technique as well and would allow us to shorten our ciphertext by one element of $\mathbb{Z}_N$.

Finally, the Cocks scheme does not choose an entirely random degree 1 polynomial but instead indexes into a family of polynomials of the form $f(x) = t + x$ by choosing a random transport key t. With this known family, the Cocks scheme is allowed to not send all coefficients of $f^2 (\text{mod } x^2 - v)$ as one of them will always be 2 and so not necessary to send. We have not yet found this to generalize nicely and have not found a family that will give known coefficients without forfeiting security.

6 Space-Efficient Variation

While our scheme is now shown to be a semantically secure generalization of the Cocks scheme which can now encrypt multiple bits at a time, it is very inefficient; the number of elements of $\mathbb{Z}_N$ that need to be sent grows exponentially with the number of bits we wish to encrypt.

We propose a space-efficient variation of our scheme that only needs to send 3 elements of $\mathbb{Z}_N$ for any number of bits we wish to be able to encrypt. Unfortunately, this is too ambitious and we prove it insecure. Moreover, we show even less ambitious attempts of the same variation fails and that no compression at all is possible in this manner.

6.1 Setup (λ)

This scheme is very similar to the original. We choose e, p, q, and ζ in an identical way as before. The only thing created differently is the publicly available hash. $\mathcal{H}$ is now chosen to map an arbitrary binary string to a *quadratic residue*, rather than an eth residue, in $\mathbb{Z}_N^*$.

$$PP \leftarrow (N, e, \zeta, \mathcal{H})$$
$$MSK \leftarrow (p, q)$$

6.2 KeyGen $(MSK, \mathcal{H}, ID)$

Again, only a trusted Private Key Generator (PKG) can perform this algorithm. Upon a user's request for their secret key, sk, the PKG will verify the users ID and then set sk to the *square* root of $v = \mathcal{H}$.

$$sk \leftarrow r = v^{\frac{1}{2}}$$

6.3 Encrypt (PP, ID, m')

Similarly, when sending an encrypted message to a user with an identity, ID, there public key must first be derived. This can as $pk \leftarrow v = \mathcal{H}(ID)$.

A message, m', may be chosen as any integer from 0 to $e-1$. This, however, must be encoded as $m = \zeta^{m'}$ to be compatible with the scheme. Encryption is identical except that we instead use $x^2 - v$ in our modulus.

$$f \xleftarrow{\text{R}} \mathbb{Z}_N[x] \text{ of degree } e-1$$
$$g \leftarrow f^e (\text{mod } x^2 - v)$$

g is now a degree 1 polynomial which can be expressed as

$$g(x) = a_0 + a_1 x$$

This is our compression: now only two coefficients need to be sent. Since we are now modding by $x^2 - v$ it can be seen why we needed the secret key to be the square root of v to unlock the message. That is, $f^e(x) = g(x) + h(x)(x^2 - v)$ for some $h(x) \in \mathbb{Z}_N[x]$. This means that $f^e(r) = g(r) + h(r)(r^2 - v) = g(r) + h(r)(v - v) = g(r)$. Again, this method of unlocking a point of $f^e(x)$ is useful in decryption.

We again choose a random transport key, t, from $\mathbb{Z}_N^*$ and send

$$CT = \left[\frac{a_o}{t}, \frac{a_1}{t}, m\left(\frac{t}{N}\right)_e\right]$$

As can be seen, our ciphertext is now only 3 elements of $\mathbb{Z}_N$ regardless of e.

6.4 Decrypt $(sk, CT = [c_o, c_1, d])$

Similar to before, for a user with the secret key r, they may recover the message by first reconstructing $\frac{g(x)}{t} \leftarrow c_0 + c_1 x$. As mentioned in encryption, r being the square root of v implies that $g(r) = f^e(r)$. And so $t^{-1}g(r) = t^{-1}f^e(r)$, and the user may calculate

$$\left(\frac{t^{-1}g(r)}{N}\right)_e = \left(\frac{t^{-1}f^e(r)}{N}\right)_e = \left(\frac{t^{-1}}{N}\right)_e \left(\frac{f^e(r)}{N}\right)_e = \left(\frac{t}{N}\right)_e^{-1} \left(\frac{f(r)}{N}\right)_e^e = \left(\frac{t}{N}\right)_e^{-1}$$

With this, decryption can easily be performed.

7 Incompressibility

We will show that our general scheme cannot be compressed in any way similar to our proposed space-efficient variation while remaining secure. We will start by attacking our space-efficient scheme and then show a general attack for any attempt to compress the ciphertext with a similar method to ours.

7.1 An Attack mod $x^2 - v$

7.1.1 Failure of Proof of Security

In our space-efficient scheme we mod our f^e by $x^2 - v$ to obtain a degree 1 polynomial with only two coefficients. Doing so, however, creates problems with security that are quickly alluded to: if we attempt to recreate our proof of security for our original scheme it fails.

Let's first consider why the original scheme worked. In general, we used the fact that, for a given ciphertext, there are f'_{ij}, t'_{ij} pairs such that $\frac{(f'_{ij})^e}{t'_{ij}} \equiv \frac{f^e}{t} (\text{mod } x^e - v)$. This was helpful since the symbol of t' could entirely "scramble" the message depending on whether v was an eth residue or not.

Consider being able to find very general f'_{ij}, and t'_{ij} so that $f'_{ij} = fa$ and $t'_{ij} = tb$ for $a, b \in \mathbb{Z}_N$. Then, if $\frac{(f'_{ij})^e}{t'_{ij}} = \frac{f^e}{t}$, it would follow that

$$\frac{(f')^e}{t'} = \frac{(fa)^e}{tb} = \frac{f^e a^e}{tb} = \frac{f^e}{t}$$

This implies that $a^e \equiv b \ (mod\ N)$, and the problem with this equality is seen when we consider the symbol of t'_{ij}:

$$\left(\frac{t'_{ij}}{N}\right)_e = \left(\frac{tb}{N}\right)_e = \left(\frac{t}{N}\right)_e \left(\frac{b}{N}\right)_e = \left(\frac{t'}{N}\right)_e \left(\frac{a^e}{N}\right)_e = \left(\frac{t}{N}\right)_e \left(\frac{a}{N}\right)_e^e = \left(\frac{t}{N}\right)_e$$

And so, regardless of whether v is an eth residue, the symbol of all t'_{ij} is constant and does not "scramble" the message uniformly when v is a nonresidue.

We were able to get around this general problem by choosing a so that it is not in $\mathbb{Z}_N$ but is instead in $\mathbb{Z}_N[x]/\langle x^e - v\rangle$. This is what we implicitly did in the general scheme: a was chosen to be x. When a is chosen from $\mathbb{Z}_N^*[x]/\langle x^e - v\rangle$ then it is no longer necessitated that $a^e \equiv b$ $(mod\ N)$ but instead that $a^e \equiv b$ $(mod\ x^e - v)$.

This allowed that $\frac{(f'_{ij})^e}{t'_{ij}} \equiv \frac{f^e}{t} (\text{mod } x^e - v)$, while still allowing b to be an eth nonresidue mod N so that the symbols of t'_{ij} could vary.

This is exactly why the original scheme worked. But now raising a to the eth power is no longer meaningful since we are no longer modding out by $x^e - v$. We were able to use $a = x$ and since x *acts* as the eth root of v in $\mathbb{Z}_N[x]/\langle x^e - v\rangle$, then raising it to the eth power allowed us to replace it with v when modded by $x^e - v$ so that everything canceled properly. We would like, for security, for a to be something meaningful when raised to the eth power and modded by $x^2 - v$, but this cannot happen in the same way as before.

While this shows that our previous proof of security is no longer applicable, it does not necessarily exclude the scheme from being secure. We, however, propose an attack that completely breaks the scheme.

7.1.2 The Attack

An adversary who intercepts a ciphertext may use it recreate the polynomial $\frac{g}{t} = c_0 + c_1 x$. Note that $t^{-1}g \in \mathbb{Z}_N/\langle x^2 - v\rangle$. Moreover, note that, by construction, g is an eth residue in $\mathbb{Z}_N/\langle x^2 - v\rangle$; f is the eth root of g mod $x^2 - v$. And so

$$\left(\frac{t^{-1}g}{\langle x^2 - v\rangle}\right)_e = \left(\frac{t^{-1}}{\langle x^2 - v\rangle}\right)_e \left(\frac{g}{\langle x^2 - v\rangle}\right)_e = \left(\frac{t^{-1}}{\langle x^2 - v\rangle}\right)_e \left(\frac{f}{\langle x^2 - v\rangle}\right)_e^e = \left(\frac{t^{-1}}{\langle x^2 - v\rangle}\right)_e$$

The question now is if $\left(\frac{t}{\langle x^2 - v\rangle}\right)_e^{-1}$ has anything to do with $\left(\frac{t}{N}\right)_e^{-1}$, which could be used for decryption.

To see what we can learn about $\left(\frac{t}{N}\right)_e^{-1}$, let's first consider the fact that $\mathbb{Z}_N \cong \mathbb{Z}_p \times \mathbb{Z}_q$ and, moreover, that $\mathbb{Z}_N[x] \cong \mathbb{Z}_p[x] \times \mathbb{Z}_q[x]$. With this, it is easy to see that $\mathbb{Z}_N[x]/\langle m(x)\rangle \cong (\mathbb{Z}_p[x] \times \mathbb{Z}_q[x])/\langle (m(x)_p, m(x)_q)\rangle$ for $m(x) \in \mathbb{Z}_N[x]$ and where $m(x)_p$ and $m(x)_q$ respectively denote $m(x)$ mod p and mod q.

We can now consider $(x^2 - v, x^2 - v) \in \mathbb{Z}_p[x] \times \mathbb{Z}_q[x]$ and see that $(x^2 - v, x^2 - v) = (x^2 - v, 1)(1, x^2 - v)$. This implies that

$$\left(\frac{t^{-1}}{\langle x^2 - v\rangle}\right)_e = \left(\frac{(t_p^{-1}, t_q^{-1})}{\langle (x^2 - v, x^2 - v)\rangle}\right)_e = \left(\frac{(t_p^{-1}, t_q^{-1})}{\langle (x^2 - v, 1)\rangle}\right)_e \left(\frac{(t_p^{-1}, t_q^{-1})}{\langle (1, x^2 - v)\rangle}\right)_e$$

From here we should note that it can be shown that $(\mathbb{Z}_p[x] \times \mathbb{Z}_q[x])/\langle (m(x), 1)\rangle \cong \mathbb{Z}_p[x]/\langle m(x)\rangle$ since everything in $\mathbb{Z}_q[x]$ must be 0 when modding by 1. This implies that, for example, $\left(\frac{(t_p^{-1}, t_q^{-1})}{\langle (x^2 - v, 1)\rangle}\right)_e = \left(\frac{t_p^{-1}}{\langle x^2 - v\rangle}\right)_e$ where the latter is calculated in $\mathbb{Z}_p[x]$. And so we find that

$$\left(\frac{t^{-1}}{\langle x^2 - v\rangle}\right)_e = \left(\frac{t_p^{-1}}{\langle x^2 - v\rangle}\right)_e \left(\frac{t_q^{-1}}{\langle x^2 - v\rangle}\right)_e = \left(\frac{t_p^{-1}}{\langle x - r\rangle}\right)_e \left(\frac{t_p^{-1}}{\langle x + r\rangle}\right)_e \left(\frac{t_q^{-1}}{\langle x - r\rangle}\right)_e \left(\frac{t_q^{-1}}{\langle x + r\rangle}\right)_e$$

The last equality here comes for the power residue symbol being a multiplicative function. Now, considering $\left(\frac{t_p^{-1}}{\langle x - r\rangle}\right)_e$, we see that the norm of $x - r$ in $\mathbb{Z}_p[x]$ is $\|x - r\| = |\mathbb{Z}_p[x]/\langle x - r\rangle| = p$. And since $\mathbb{Z}_p[x]/\langle x - r\rangle$ is a field as $\mathbb{Z}_p[x]$ is a PID and $x - r$ is irreducible, then the eth residue symbol of $h(x) \in \mathbb{Z}_p[x]$ can be defined as

$$\left(\frac{h(x)}{\langle x - r\rangle}\right)_e = h(x)^{\frac{\|x - r\| - 1}{e}} = h(x)^{\frac{p-1}{e}} (\text{mod } x - r)$$

Moreover, if $h(x) = a \in \mathbb{Z}_p$, then modding it, or powers of it, by $x - r$ does not affect it, although it does remain in $\mathbb{Z}_p$. And so

$$\left(\frac{a}{\langle x - r\rangle}\right)_e = a^{\frac{p-1}{e}} (\text{mod } x - r) = a^{\frac{p-1}{e}} (\text{mod } p)$$

But this last part is simply the definition of $\left(\frac{a}{p}\right)_e$ and so we get that $\left(\frac{a}{\langle x-r\rangle}\right)_e = \left(\frac{a}{p}\right)_e$. Similar analyses yield the following.

$$\left(\frac{t^{-1}}{\langle x^2-v\rangle}\right)_e = \left(\frac{t_p^{-1}}{\langle x-r\rangle}\right)_e \left(\frac{t_p^{-1}}{\langle x+r\rangle}\right)_e \left(\frac{t_q^{-1}}{\langle x-r\rangle}\right)_e \left(\frac{t_q^{-1}}{\langle x+r\rangle}\right)_e = \left(\frac{t^{-1}}{p}\right)_e \left(\frac{t^{-1}}{p}\right)_e \left(\frac{t^{-1}}{q}\right)_e \left(\frac{t^{-1}}{q}\right)_e$$
$$= \left(\left(\frac{t^{-1}}{N}\right)_e\right)^2$$

If we can take the square root of this then we completely recover $\left(\frac{t^{-1}}{N}\right)_e$ and can then decrypt. And it is easy to take the square root of an eth root of unity where e is prime. Therefore an adversary can completely decrypt any ciphertext he receives by simply using reciprocity to compute the eth residue symbol of the polynomial given in the ciphertext mod $x^2 - v$.

7.2 An Attack mod $x^\delta - v$

After seeing that the space-efficient scheme failed, a natural question may be to ask if using $x^2 - v$ may have been too ambitious a compression and if some $x^\delta - v$ may instead be used so that there are only $\delta < e$ elements of $\mathbb{Z}_N$ that need to be sent. We may even settle for a δ that is a function of e. Another natural question is why the attack against $x^2 - v$ does not work on our original scheme. We answer both of these questions here.

If our space-efficient scheme is modified so that we instead use δ^{th} residues as public keys and their roots as secret keys, as well as mod by $x^\delta - v$ to compress the polynomial to δ coefficients, we show that it is insecure. In general, this modified scheme can be seen to work similarly as the space-efficient scheme for encryption and decryption, but the same problem arises when attempting to prove security. Moreover, our attack that allows full decryption of a ciphertext in the space-efficient scheme generalizes to break this modified scheme as well.

Similar to before, an adversary who intercepts a ciphertext may use it to recreate the polynomial $t^{-1}g$ with its δ coefficients. It can then similarly be argued that

$$\left(\frac{t^{-1}g}{\langle x^\delta - v\rangle}\right)_e = \left(\frac{t^{-1}}{\langle x^\delta - v\rangle}\right)_e = \left(\frac{(t_p^{-1}, t_q^{-1})}{\langle (x^\delta - v, x^\delta - v)\rangle}\right)_e = \left(\frac{(t_p^{-1}, t_q^{-1})}{\langle (x^\delta - v, 1)\rangle}\right)_e \left(\frac{(t_p^{-1}, t_q^{-1})}{\langle (1, x^\delta - v)\rangle}\right)_e$$
$$= \left(\frac{t_p^{-1}}{\langle x^\delta - v\rangle}\right)_e \left(\frac{t_q^{-1}}{\langle x^\delta - v\rangle}\right)_e$$

While we don't know exactly how $x^\delta - v$ reduces as we did with $x^2 - v$, we do know it factors uniquely in both $\mathbb{Z}_p[x]$ and $\mathbb{Z}_q[x]$ as they are UFDs. That is, $x^\delta - v = \prod_{i=0}^{n} \alpha_i(x)$ for unique monic irreducibles $\alpha_i(x)$ in $\mathbb{Z}_p[x]$, and $x^\delta - v = \prod_{i=0}^{m} \gamma_i(x)$ for unique monic irreducibles $\gamma_i(x)$ in $\mathbb{Z}_q[x]$.

For a give $\alpha_i(x)$ we know that $\mathbb{Z}_p[x]/\langle \alpha_i(x)\rangle$ is a field since $\alpha_i(x)$ is irreducible. Moreover, $\|\alpha_i(x)\| = |\mathbb{Z}_p[x]/\langle \alpha_i(x)\rangle| = p^{deg(\alpha_i(x))}$. Then, similar to before, for some $h(x) \in \mathbb{Z}_p[x]$

$$\left(\frac{h(x)}{\langle \alpha_i(x)\rangle}\right)_e = h(x)^{\frac{\|\alpha_i(x)\|-1}{e}} = h(x)^{\frac{p^{deg(\alpha_i(x))}-1}{e}} (\text{mod } \alpha_i(x))$$

And, again, if $h(x) = a \in \mathbb{Z}_p$ then

$$\left(\frac{a}{\langle \alpha_i(x)\rangle}\right)_e = a^{\frac{p^{deg(\alpha_i(x))}-1}{e}} (\text{mod } p) = \left(a^{\frac{p-1}{e}}\right)^{\frac{1-p^{deg(\alpha_i(x))}}{1-p}} = \left(\frac{a}{p}\right)_e^{\frac{1-p^{deg(\alpha_i(x))}}{1-p}}$$

The last equalities are gotten simply from some exponent manipulation and definition of the eth residue symbol. Now note that $\frac{1-p^{deg(\alpha_i(x))}}{1-p}$ is the geometric series $\sum_{i=0}^{deg(\alpha_i(x))-1} p^i$. Since exponentiation is being

done here amongst the eth roots of unities, the exponent may be reduced modulo e. That is, we can reduce $\sum_{i=0}^{deg(\alpha_i(x))-1} p^i$ to $\sum_{i=0}^{deg(\alpha_i(x))-1} 1^i = deg(\alpha_i(x))$. And so finally we can say that

$$\left(\frac{a}{\langle \alpha_i(x)\rangle}\right)_e = \left(\frac{a}{p}\right)_e^{deg(\alpha_i(x))}$$

We can similarly say that for $a \in \mathbb{Z}_q$

$$\left(\frac{a}{\langle \gamma_i(x)\rangle}\right)_e = \left(\frac{a}{q}\right)_e^{deg(\gamma_i(x))}$$

Thus,

$$\left(\frac{t^{-1}}{\langle x^\delta - v\rangle}\right)_e = \left(\frac{t_p^{-1}}{\langle x^\delta - v\rangle}\right)_e \left(\frac{t_q^{-1}}{\langle x^\delta - v\rangle}\right)_e = \prod_{i=0}^{n}\left(\frac{t_p^{-1}}{\langle \alpha_i(x)\rangle}\right)_e \prod_{i=0}^{m}\left(\frac{t_q^{-1}}{\langle \gamma_i(x)\rangle}\right)_e$$

$$= \prod_{i=0}^{n}\left(\frac{t^{-1}}{p}\right)_e^{deg(\alpha_i(x))} \prod_{i=0}^{m}\left(\frac{t^{-1}}{q}\right)_e^{deg(\gamma_i(x))} = \left(\frac{t^{-1}}{p}\right)_e^{\sum_{i=0}^{n} deg(\alpha_i(x))} \left(\frac{t^{-1}}{q}\right)_e^{\sum_{i=0}^{m} deg(\gamma_i(x))}$$

$$= \left(\frac{t^{-1}}{p}\right)_e^{\delta} \left(\frac{t^{-1}}{q}\right)_e^{\delta} = \left(\left(\frac{t^{-1}}{N}\right)_e\right)^{\delta}$$

And finally, since we are in the eth roots of unity where e is prime, taking the δth root and recovering $\left(\frac{t^{-1}}{N}\right)_e$ should be easy. This completely breaks the scheme and shows that no δ will be secure, and thus no compression can be done in this manner. We also can see why this attack does not work on our original scheme.

If $\delta = e$, a similar analysis would show that

$$\left(\frac{t^{-1}}{\langle x^e - v\rangle}\right)_e = \left(\left(\frac{t^{-1}}{N}\right)_e\right)^{e} = 1$$

And so no information is gained with this attack.

8 Conclusion

We have a secure and working scheme that relies on a, modified, e^{th} residuosity problem. While it is currently inefficient, it shows that a generalization is possible, which may be able to be made efficient in the future. We also show a somewhat natural idea of compression and show the less natural way in which it is insecure and how it can be broken. Moreover, we show that it is completely incompressible for any generalization of the failed compression method.

It is still currently an open problem as to whether an *efficient* IBE scheme can be made from general e^{th} residuosity. Future work in attempting to compress the ciphertexts in our scheme may also be an interesting problem. It is also unsolved as to whether Hierarchical IBE can be obtained from quadratic or e^{th} residuosity.

One idea of compression is to, like the Cocks scheme, choose f from a family of functions rather than randomly so that there will be known coefficients or ratios of coefficients that do not need to be sent. Such a family has not yet been discovered and it tends to be the case that the properties of the families begin to break down after modding by $x^e - v$.

References

[1] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In proceedings of *FOCS 2007*, pages 647-57, 2007.

[2] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586-615, 2003. extended abstract in Crypto '01.

[3] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 26-8, 2001.

[4] David Madell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In *Journal of Cryptology*, 26:39-74, 20013.

[5] Jeremy Horwitz and Ben Lynn. Towards hierarchical identity-based encryption. In Lars Knudsen, editor, *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*, pages 466-81. Springer, 2002.

[6] Kenneth Ireland and Michael Rosen. *A classical introduction to modern number theory*, volume 84 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1990.

[7] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology—CRYPTO 1984*, volume 196 of *LNCS*, pages 47-53. Springer-Verlag, 1984.

[8] Douglas Squirrel. Computing reciprocity symbols in number fields, 1997. Undergraduates thesis, Reed College.