GGH13

Shahed Sharif

January 26, 2019

1 Outline of the construction

The ideas of the construction are a bit weird, so I will first give a very sketchy outline. The goal is to construct a k-way cryptographic multilinear map; that is, we want groups G_1, G_k and a cryptographic multilinear map

$$m : (G_1)^k \to G_k.$$

Let $R = \mathbb{Z}[x]/(x^n + 1)$, where n is a security parameter. Garg-Gentry-Halevi's construction first secretly chooses a prime ideal $I \subset R$, then uses the ring structure of R/I to obtain a multilinear map. To obtain a cryptographically secure construction, users do not work directly with R/I, but rather with *encodings* of R/I. Encodings will be elements of R/q, where q is a large rational prime. The encodings taken together give a sort of graded ring homomorphism

$$(R/I)^k \to (R/q)^k.$$

Of course, since we do not assume that $q \mid \mathrm{Nm}(I)$, this is nonsense from a strictly mathematical viewpoint. However, from an algorithmic viewpoint, everything works out ... with high probability.

2 Framework: graded encodings

The goal is to come up with a cryptographic k-graded encoding system, which consists of a ring A and sets $S_0, S_1, \ldots, S_k$ along with the following:

1. for each i, an efficient probabilistic encoding algorithm $E_i : A \rightsquigarrow S_i$;
2. if $a, b \in A$ with $a \neq b$, then $E_i(a) \neq E_i(b)$ for any instance of E_i;
3. an associative binary operation $+$ on S_i such that if $a, b \in A$, then $E_i(a) + E_i(b)$ is a potential output of $E_i(a + b)$;
4. a similar property for additive inversion; and
5. an efficiently computable graded multiplication operation $S_i \times S_j \to S_{i+j}$ for $i + j \leq k$, where $E_i(a) \cdot E_j(b)$ is a potential output of $E_{i+j}(ab)$.

Furthermore, we want the following publicly available efficient algorithms:

Sampling Output $E_0(a)$ uniformly randomly in $a \in A$.

Encoding Given $E_0(a)$ and i, output $E_i(a)$.

Ring operations The $+, -, \cdot$ operations above should be efficiently computable.

Zero test Given $E_k(a)$, determine if $a = 0$.

Extraction Given $E_k(a)$, compute a hash value that depends only on a. Furthermore, the hash outputs should be uniformly distributed with respect to S_k. That is, assuming extraction outputs an N-bit hash, then as $s \in S_k$ varies uniformly, the outputs should vary nearly uniformly over the space of all N-bit strings.

A word on these. For encoding, we really only care about the $i = 1$ case. The zero test procedure together with the ring operations allow us to compute equality at the kth level; that is, given $s, s' \in S_k$, we can determine whether they are kth level encodings of the same ring element. The extraction procedure gives us a cryptographically useful output—a string to be used for symmetric key, for example.

It turns out that neither the zero test nor the extraction methods will work perfectly. However, they will succeed with high probability.

The multilinear map is just the multiplication map

$$(S_1)^k \to S_k.$$

To get $k + 1$ party Diffie-Hellman, proceed as follows. Each of $k + 1$ parties samples an element from S_0; say the ith party chooses p_i ($i = 0, 1, \ldots, k$). Each party publishes $E_1(p_i)$. The shared key is then the extraction of $E_k(\prod p_i)$. Party j can compute the shared key as the extraction of

$$E_0(p_j) \prod_{i \neq j} E_1(p_i).$$

We want the corresponding Diffie-Hellman problem to be difficult.

3 Specifics of GGH construction

Let $R = \mathbb{Z}[x]/(x^n + 1)$, and write ζ for the class of x. Let K be the fraction field of R. Our construction will depend on the following choices, which are public unless otherwise specified:

- q, a large rational prime.
- A private parameter $g \in R$. Our choice of ring A depends on g.
- A private parameter $z \in R/q$. The definition of our encoding maps will depend on z.

- $y \in R/q$. Users will be able to compute the encoding maps using y.
- $x_1, \ldots, x_m \in R/q$. These are noise terms that will randomize the encoding maps. We will essentially have $x_i = b_i/z$ for some short $b_i \in (g)$.
- $p_{zt} \in R/q$. This parameter will enable both zero testing and hashing.

Throughout I will be using the words "short", "long", "small", and "big". Short and long mean that with respect to a specific length function, the length of an element is small, respectively big. Since I won't be going through technical details, I won't make the notions of small and big too precise; in most of the below, small just means $< q^{1/N}$ for an appropriate choice of N. (For $i \geq 1$, $N = 8i/k$.)

Given $\alpha \in K$, if we write α as a polynomial in ζ, we say α is *short* if the Euclidean length of its coefficient vector is small. GGH show that with high probability, α short implies that α^{-1} is short. Then let $g \in R$ be a secretly generated short element whose inverse is also short. Set $I = (g)$. We additionally choose g so that I is prime, although it is unclear if this is really necessary.

Choose q to be a big rational prime. For $\alpha \in R$, write α as an integral polynomial in ζ. Let $[\alpha]_q$ be the image of α in R/q. We represent $[\alpha]_q$ as the polynomial obtained by replacing the coefficients with integers which lie in the interval $(-q/2, q/2)$. We will frequently conflate an expression $[\alpha]_q$ with its natural lift to R.

We define

$$E_0 : R/I \rightsquigarrow R/q$$

as follows. For $e + I \in R/I$, choose a short representative c; the shortness of g implies that such a c exists. Define

$$E_0(e + I) = [c]_q.$$

Note that since c is not uniquely determined, E_0 is a probabilistic algorithm. Also, as c is a short representative and q is large, E_0 is injective, in the sense that if $e + I \neq e' + I$, then $E_0(e + I)$ can never equal $E_0(e' + I)$.

Let $z \in R/q$ be a randomly determined secret element. We will write $1/z$ for the inverse of z mod q. Further define

$$E_i : R/I \rightsquigarrow R/q$$
$$E_i(e + I) = [c/z^i]_q.$$

The E_i enjoy the same properties as E_0. The factor of $1/z^i$ serves to "randomize" the image inside of R/q.

Users can use E_0 as a *sampling* algorithm; that is, sampling $E_0(e)$ is essentially equivalent to choosing a short element c of R at random and outputting $[c]_q$.

Note that since z is secret, users cannot use the definition above to compute E_i for $i \geq 1$. The encoding step, however, only requires that given $E_0(e + I)$, a user should be able to compute $E_i(e + I)$. To do this, we compute some

$y = E_1(1+I)$, say $y = [a/z]_q$ with $a \in 1+I$, and publish y. Observe that $yE_{i-1}(e+I)$ is a level i encoding of e: let c be a short element of $e+I$. Then since a is short, ca is another short element of $e+I$, and furthermore $E_i(e+I)$ could be given by $[ca/z]_q$. On the other hand,

$$\begin{aligned} yE_{i-1}(e+I) &= [a/z]_q \cdot [c/z^{i-1}]_q \\ &= [ca/z^i]_q. \end{aligned}$$

However, this method of encoding at level i is insecure because an attacker can solve the associated Diffie-Hellman problem! Specifically, an attacker can *divide* by y in R/q to obtain $E_0(e+I)$ from $E_1(e+I)$, which breaks Diffie-Hellman. Thus we also add some noise; that is, our algorithm to compute a level i encoding is to compute $yE_{i-1}(e+I)$, as above, but then to add an element $[\varepsilon/z^{i-1}]_q$, where $\varepsilon \in I$ is a small noise term. Thus, our level i encoding algorithm is to compute

$$\left[\frac{ca+\varepsilon}{z^i}\right]_q$$

The noise term is obtained as follows. Choose short $b_1, \ldots, b_m \in I$. Set $x_i = [b_i/z]_q$; these are level 1 encodings of 0, and we publish them as part of the specification. Then the error term ε/z is obtained as a random integer linear combination of the x_i. Thus for example

$$E_1(e+I) = \left[cy + \sum \alpha_i x_i\right]_q.$$

The ring operations are the obvious ones. Note that we need the c's to be short relative to q in order for these operations to be well-defined! That is, if c, c' are short elements of two cosets and $i + j \le k$, then we should have that the natural lift of $[cc']_q$ to R is cc'; that is, the coefficients of cc' should all lie in the interval $(-q/2, q/2)$. For similar reasons, addition breaks down if there are too many summands.

For zero-testing, we choose ephemeral $h \in R$, $h \notin I$, which is not too short—specifically, its length should be about $\sqrt{q}$. Then publish $p_{zt} = [hz^k/g]_q$. Given a level k encoding $u = [c/z^k]_q$, we check if u is an encoding of 0 (equivalently, that $c \in I$) by computing the Euclidean length of $[up_{zt}]_q$. (To compute the length, we lift to $R \cong \mathbb{Z}^n$ in the natural way.) We now show that with high probability, the product is relatively short (say, length $< q^{3/4}$) if and only if $c \in I$.

Lemma 3.1. *If $c \in I$, then $[up_{zt}]_q$ is relatively short.*

Proof. Suppose $c \in I$. Then

$$\begin{aligned} [up_{zt}]_q &= \left[\frac{c}{z^k}\frac{hz^k}{g}\right]_q \\ &= [ch/g]_q \\ &= [c'h]_q \end{aligned}$$

where $c' = c/g$. But $c \in I$, so $c' \in R$. Furthermore since c, g^{-1} are short, so is c'. Therefore $c'h$ will not be much longer than h. □

Lemma 3.2. *If $[up_{zt}]_q$ is relatively short, then $c \in I$.*

Proof. We have $[up_{zt}]_q = [ch/g]_q$. Let w be the natural lift of $[up_{zt}]_q$ to R. Then $gw \equiv ch \pmod{q}$. By hypothesis, g, w, c are short relative to q. Furthermore, h has size around $\sqrt{q}$. In particular, none of the coefficients of the products gw, ch will have size bigger than q. Therefore $gw = ch$. Since $I = (g)$ is prime, we have $c \in I$. □

Lastly, we must define extraction (the hash function). To do this for a level k encoding u, we just take the most significant r bits of the coefficients $[up_{zt}]_q$ for appropriate r, concatenate, and apply a standard hash function. The reason this works is that if u, u' are two encodings of the same coset, then $u - u'$ is a level k encoding of 0, and so $[(u - u')p_{zt}]_q$ will be small. Equivalently, the first few significant bits of each coefficient will be zero. Thus, the first few significant bits of coefficients of $[up_{zt}]_q$ and of $[u'p_{zt}]_q$ will coincide. A similar argument shows that if u, u' are encodings of different cosets, then the extraction function returns different results.

4 Security

GGH make an argument for security, but it turns out that their scheme is *not* secure. The scheme was broken in Hu-Jia 2015 by a *zeroizing attack*, so-called because it makes use of the zero-testing parameter p_{zt} to attack the system. Specifically, Hu-Jia are able to solve the Diffie-Hellman problem. I will not cover that attack at this time.

GGH do make mention of another attack. First of all, GGH argue that from the publicly available data, an attacker can recover a $\mathbb{Z}$-basis of I. Presumably, it is not difficult to then obtain a generator. But if the attacker can recover a *short* generator of I, then they can solve the Decisional Diffie-Hellman problem. Finding a short generator of a principal ideal is a well-known problem, called the *short principal ideal problem*, or SPIP. If we know the unit group of R, then it is not difficult to obtain a short generator from an arbitrary one.

We give a brief outline of how this works.

Computing a basis for I. We first show how to compute a basis for $I = (g)$. The idea is that we compute $\mathbb{Z}$-bases for the fractional ideals (h) and (hg), which allows us to compute a basis for (g).

An adversary generates products of the form

$$t = [(x_{i_1} \cdots x_{i_r}) \cdot (u_1 \cdots u_s) \cdot y^{k-r-s}]_q$$

where $r \geq 1$, and the u_i are randomly chosen level 1 encodings. We have that for some short elements $b_{i_j}, e_i \in R$,

$$x_{i_j} = [b_{i_j} g/z]_q \text{ and } u_i = [e_i/z]_q.$$

Therefore

$$t = [(\prod b_{i_j} \prod e_i) a^{k-r-s} \cdot g^r / z^k]_q.$$

Write α for the parenthetical expression. Let $v = [t \cdot p_{zt}]_q$. Then

$$v = [\alpha a^{k-r-s} g^{r-1} h]_q$$

But since all of the factors are sufficiently short with respect to q, by taking the natural lift to R, the adversary has in fact computed the product

$$\alpha a^{k-r-s} g^{r-1} h.$$

By varying the choices of the x_{i_j} and the u_i, the adversary can obtain many elements of the ideal (h), and eventually a basis for (h). Using the same procedure but with $r \geq 2$, the adversary obtains a basis for (gh). They then compute the fractional ideal $(1/h)$, and from there, a basis for $(1/h) \cdot (gh) = (g)$.

From a short generator to DDH. Now suppose our adversary can not only compute a basis for (g), but can in fact compute a generator $g' = dg$ which is also short. We will also assume that $R/I \cong \mathbb{Z}/N\mathbb{Z}$ for some integer N; this occurs with "good" probability. Earlier we assumed that I is prime, and under this assumption, R/I is cyclic with certainty.

We now show how to solve the Decisional Diffie-Hellman problem. The algorithm to do so is as follows:

Input: $u_0, \cdots, u_k$, where $u_i = E_1(p_i)$ are level 1 encodings; and a level k encoding w.

Output: If w is the shared Diffie-Hellman key associated to the p_i, return YES. Otherwise, return NO.

1. Set $p'_{zt} = [g' p_{zt}]_q = [dhz^k]_q$.
2. Set
$$v_1 = [p'_{zt} w]_q \text{ and } v_2 = [p'_{zt} \prod_{i=1} u_i]_q.$$
3. Compute a homomorphism $\varphi : R \to \mathbb{Z}/N\mathbb{Z}$ with kernel I. (Details later.)
4. Set $\eta = \varphi(v_1)\varphi(v_2)^{-1}$.
5. Lift η mod I to a short element $p \in R$. That is, we should have $e \equiv \eta \pmod{I}$.
6. Let $u = E_1(p)$.
7. If $[uy^{k-1}]_q = [u_0 y^{k-1}]_q$, return YES. Otherwise, return NO.

We now explain the algorithm.

Recall that w is the shared Diffie-Hellman key if

$$w = [p_0 \prod_{i=1}^{k} u_i]_q.$$

Let us assume that this is the case. We know that $u_i = (p_i + \varepsilon_i g)/z$ for small ε_i (coming from the noise term in the level 1 encoding). Thus

$$w = \left[\frac{cg + \prod_{i=0} p_i}{z^k} \right]_q$$

for some short c. This implies that

$$v_1 = \left[dh(cg + \prod_{i=0} p_i) \right]$$

and

$$v_2 = [dh(c'g + \prod_{i=1} p_i)]_q.$$

Since all elements in these expressions are short relative to q, we can remove the "mod q" brackets.

To compute φ, we find a $\mathbb{Z}$-basis for I, compute a Hermite normal form, and reduce with respect to the associated basis. The Hermite normal form should have diagonal terms $N, 1, 1, \ldots, 1$, so this reduction should be easy.

Now $v_1 \equiv dh \prod_{i=0} p_i \pmod{I}$ and $v_2 \equiv dh \prod_{i=1} p_i \pmod{I}$, so the mod I quotient v_1/v_2 is congruent to p_0. In particular, $p \equiv p_0 \pmod{I}$. Since g' is short, our lift p can also be chosen to be short. In particular, $E_k(p)$ should give the same result as $E_k(p_0) = [y^{k-1} u_0]_q$.

If w is *not* the shared key, then with high probability $p \not\equiv p_0 \pmod{I}$, and the algorithm will output NO.

A List of symbols

This is a summary of the symbols used in GGH, and the entities they represent.

R	$\mathbb{Z}[x]/x^n + 1$, $n = 2^m$
q	$q \in \mathbb{Z}$ large prime
g	$g \in R$ short, secret
I	(g)
z	$z \in R/q$ secret
y	$[a/z]_q$, $a \in 1 + I$ short
b_i	$b_i \in I$ short
x_i	$x_i = [b_i/z]_q$
E_0	$E_0(e + I) = [c]_q$, $c \in e + I$ short
E_1	$E_1(e + I) = [cy + \sum \alpha_i x_i]_q$, α_i random small integers
h	$h \in R, h \notin I$ medium length
p_{zt}	$p_{zt} = [hz^k/g]_q$