Notes on GGH15

Travis Scholl

February 1, 2019

Abstract

In this talk, we will summarize a candidate multilinear map construction due to Gentry-Gorbunov-Halevi (GGH15). While the original key exchange protocol using GGH15 is broken, new safeguards have recently been proposed which aim to prevent this and other "zeroizing" attacks. Our goal will be to understand the construction, attack, and proposed safeguards.

1 Construction

Recall how in GGH13, we had different "levels" of encodings of plaintext. In GGH15, our encodings will be relative to paths in a graph. The plaintext space will be a ring.

1.1 Framework

Let G be a connected directed acyclic graph with d vertices. We denote an encoding of a plaintext S relative to a path $u \rightsquigarrow v$ in G by $[S]_{u \rightsquigarrow v}$. Our system needs to support the following operations:

Encode: Given a plaintext S and a path $u \rightsquigarrow v$, output an encoding $[S]_{u \rightsquigarrow v}$.

Addition: Given encodings $[S_1]_{u \rightsquigarrow v}$ and $[S_2]_{u \rightsquigarrow v}$, output $[S_1 + S_2]_{u \rightsquigarrow v}$.

Multiplication: Given encodings $[S_1]_{u \rightsquigarrow v}$ and $[S_2]_{v \rightsquigarrow w}$, output $[S_1 S_2]_{u \rightsquigarrow w}$. Here $u \rightsquigarrow w$ is the concatenation of the paths $u \rightsquigarrow v$ and $v \rightsquigarrow w$.

Zero Testing: Given $[S]_{u \rightsquigarrow v}$, determine whether $S = 0$.

Extract: Given $[S]_{u \rightsquigarrow v}$, extract some bits that depend only on S, u, and v.

1.2 Implementation

$$\begin{aligned} q &= \text{ a large}^{1} \text{ prime, e.g. } q \approx 2^{200}, \\ n, m &= \text{ integers with } m \approx n \log q, \text{ e.g. } n \approx 4000,\ m \approx 800000 \\ G &= \text{ DAG, e.g. a straight line } u_1 \to \cdots \to u_d, \\ A_u &= \text{ random matrix in } \mathbb{Z}^{n \times m} \text{ associated to vertex } u, \\ \tau_u &= \text{ trapdoor information for } A_u, \text{ see below,} \\ \text{Plaintext} &= S \in \mathbb{Z}^{n \times n} \text{ with short (e.g. } \ll q^{1/d}) \text{ entries} \end{aligned}$$

The operations above are implemented as follows:

Encode: Given a plaintext S and a path $u \rightsquigarrow v$, output a short matrix $D \in \mathbb{Z}^{m \times m}$ such that

$$A_u D \equiv S A_v + E \mod q,$$

where E is an error matrix in $\mathbb{Z}^{n \times m}$ with small ($\ll q^{1/d}$) norm. Note D can be computed efficiently using a private precomputed trapdoor τ_u.[2]

Addition: Given $D_1 = [S_1]_{u \rightsquigarrow v}$ and $D_2 = [S_2]_{u \rightsquigarrow v}$, output $D_1 + D_2 = [S_1 + S_2]_{u \rightsquigarrow v}$.

Multiplication: Given $D_1 = [S_1]_{u \rightsquigarrow v}$ and $D_2 = [S_2]_{v \rightsquigarrow w}$, output $D_1 D_2 = [S_1 S_2]_{u \rightsquigarrow w}$.

Zero Testing: Given $D = [S]_{u \rightsquigarrow v}$, output whether $A_u D \mod q$ is small.

Extract: Given $D = [S]_{u \rightsquigarrow v}$, output the top bits of $A_u D \mod q$.

Remark 1. Notice that encodings really don't have to do anything with the graph. Instead, you can think of encodings as relative to pairs (u, v) coming from an arbitrary set (i.e. the vertex set). Because encoding is not a public function, the graph visualization is useful to visualize legal multiplications.

1.3 Correctness

The non-obvious parts of the implementation are multiplication, zero testing, and extraction.

For zero testing, recall that $A_u D \equiv S A_v + E \mod q$. If $S = 0$, then $A_u D \equiv E$, so $A_u D$ will be small. Otherwise, because A_v should look random, $S A_v$ should be large. Hence $A_u D$ will be large.

For multiplication, note that

$$A_u (D_1 D_2) \equiv (S_1 S_2) A_w + (S_1 E_2 + E_1 D_2) \mod q.$$

Since S_i, E_i, and D_i are all small, it follows that $D_1 D_2$ is small and $S_1 E_2 + E_1 D_2$ is also small.

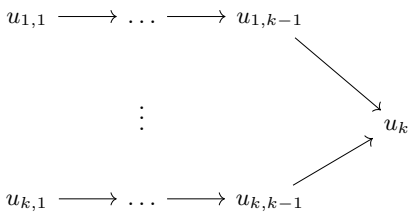
[2] In the original paper [GGH15], the matrices are transposed. We are following the syntax given in [BGMZ18] because it means a trapdoor is a basis for the right kernel instead of the left.

For extraction, note that $A_u D \equiv SA_v + E \mod q$. So we can compute $A_u D \mod q$. Since E is small, the top bits of $A_u D \mod q$ match those of $SA_v \mod q$, which is independent of the error matrix E. Note that here we are assuming that E has small positive entries. If we allow E to have small negative entries, we could apply a "shift". Let Δ be a matrix where every entry is a constant greater than the maximum absolute value in any error matrix E. Then we can extract the top bits of $A_u D + \Delta$ which will match the top bits of $SA_v + \Delta$. Again, this main point is that the resulting bits are independent of the error E.

2 Example Protocol: Key Exchange

Suppose parties $1, \ldots, k$ want to perform agree on a shared key. Also for this protocol, we restrict the plaintext space to commuting small matrices.

The protocol will run on the following graph. It is a collection of k chains of length $k+1$, such that every chain ends on the same vertex.



As part of the setup, we will publish lots of encodings of plaintext elements $T_1, \ldots, T_N$ with respect to every edge.

1. Party i chooses a secret plaintext S_i. This is done by choosing an linear combination of the $\{T_1, \ldots, T_N\}$ with small coefficients. Note that party i does not need to know the T_i, only their encodings on each edge. This because if $S_i = \sum a_j T_j$ then $[S_i]_{u \rightsquigarrow v} = \sum a_j [T_j]_{u \rightsquigarrow v}$.
2. Party i publishes an encoding of S_i with respect to the edges $u_{j,j+i-1} \to u_{j,j+i}$ for $j = 1, \ldots, k$ except for the edge $u_{i,0} \to u_{i,1}$.
3. The shared secret can be computed as follows. Party i computes an encoding of $D = [\prod S_i]_{u_{i,1} \rightsquigarrow u_k}$ and applies the extraction function.

2.1 A Concrete Walkthrough

Figure 1 shows the published/hidden information used in the key exchange. Each party can use the public information to compute an encoding of the product of the S_i using their "chain". For example, party $i = 2$ takes the product of $[S_3]_{u_{2,0} \to u_{2,1}}$, $[S_1]_{u_{2,1} \to u_{2,2}}$, and $[S_2]_{u_{2,2} \to u_3}$. This gives an encoding $D = [S_3 S_1 S_2]_{u_{2,0} \to u_3}$. Note that $A_{u_{2,0}} D \equiv S_3 S_1 S_2 A_{u_3} + E \mod q$ for some error E. Then the top bits of $A_{u_{2,0}} D$ does not depend on the choice $i = 2$.

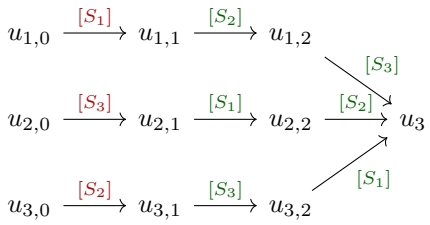


Figure 1: Demonstration of 3-way key exchange. Green represents published and red represents hidden. Each plaintext is encoded with respect to the edge it lies over.

3 Security

3.1 Key Exchange

The key exchange given was broken in [CLLT16]. Their attack generates an encoding of a users private key, i.e. $[S_i]_{u_{i,0} \to u_{i,1}}$. A variant of GGH15 given in [BGMZ18] aims to fix this with added safeguards. This attack and the proposed safeguards will be discussed in another talk.

3.2 Lattice Trapdoors

The encoding scheme we used relies on being able to generate a random matrix A along with some "trapdoor" information τ such that it is easy to generate small solutions to $Ax = b \mod q$. The following lemma shows why we should expect short solutions to exist. This lemma comes from Lemma 5.1 of https://eprint.iacr.org/2007/432.pdf which cites [Ajt96a].

Lemma 2. *Let q be a prime and $m \geq 2n \log q$. Then for all but at most a q^{-n} fraction of matrices $A \in (\mathbb{Z}/q\mathbb{Z})^{n \times m}$, every vector in $b \in (\mathbb{Z}/q\mathbb{Z})^n$ admits a preimage $x \in \{0,1\}^m$ such that $Ax = b$. That is, every vector can be written as a subset sum of the columns of A.*

Remark 3. One way to interpret this result is that it says that $T(H) = \mathbb{R}^n$ where H is the hypercube in $\mathbb{R}^m$ and T is the linear transformation determined by A.

One of the main results in [Ajt96b] is that we can efficiently generate a "random" matrix $A \in \mathbb{Z}^{n \times m}$ along with a short basis T for the lattice $\Lambda^\perp = \{v \in \mathbb{Z}^m : Av \equiv 0 \mod q\}$. Notice that $\Lambda^\perp$ has full rank (it has rank m as a $\mathbb{Z}$-module) because it includes the columns of qI_m where I_m is the $m \times m$ identity matrix. The quotient of $\Lambda^\perp$ by qI_m is the null space of $A \mod q$ as a matrix in $(\mathbb{Z}/q\mathbb{Z})^{n \times m}$. With high probability, $A \mod q$ has rank n and nullity $m - n$.

Given A and T as above, we can now solve the problem of finding short solutions to $Ax = b \mod q$.

To find a short solution to the equation $Ax \equiv b \mod q$, we find any solution and subtract off the projection onto the vectors in T.[3]

3.3 LLL

The Lenstra-Lenstra-Lovász (LLL) algorithm takes a basis for a lattice and returns a short basis. The algorithm runs in polynomial time. Given a full rank lattice in $\mathbb{R}^d$, the running time of LLL is roughly $O(d^6 \log^3 B)$ where B is the largest component of any of the given basis vectors.

Recall that $\Lambda^\perp(A_u)$ is a full rank lattice in $\mathbb{R}^m$ (it is full rank because it contains the columns of qI_m). The best public basis available will include vectors of norm $\approx q$. To run LLL would require roughly $O(m^6 \log^3 q) = O(n^6 \log^9 q)$. For the choices listed above, this is $(4000)^6 \log^9(2^{200}) \approx 2^{135}$. Therefore running LLL is likely to be computationally infeasible.

3.4 Zeroizing Attacks

In this section, we will describe how encodings of 0 can lead to trapdoors. Suppose we had a non-trivial[4] encoding C of 0, i.e. $C = [0]_{u \rightsquigarrow v}$ for some path $u \rightsquigarrow v$. So $A_u C \equiv E \mod q$ for some small error matrix E.

Set

$$A'_u = \begin{bmatrix} A_u & I_n \end{bmatrix} \text{ and } C' = \begin{bmatrix} C \\ -E \end{bmatrix}.$$

Then we have $A'_u C' \equiv 0 \mod q$. Moreover, C' is small. Recall that a trapdoor T for A_u is a short basis for $\Lambda^\perp(A)$. Here C' is almost a trapdoor for A'_u except that $\Lambda^\perp(A'_u)$ has rank $m+n$ but C' has rank m.

If we have another encoding of 0, we can repeat the same process to get another C'. Together, these have $2m$ vectors. Since $m \gg n$, we should be able to find $m+n$ linearly independent vectors between the columns of the two C'. This gives us a basis T' for $\Lambda^\perp(A'_u)$. In particular, we have constructed an $(m+n) \times (m+n)$ small matrix T' with $A'_u T' \equiv 0 \mod q$.

Now that we have a trapdoor for A'_u, we can use it to decode elements as follows. Suppose D is an encoding of some plaintext S relative to a path ending at u. That is,

$$A_w D \equiv S A_v + E \mod q.$$

We can extend this to a relation over the larger matrices

$$\begin{bmatrix} A_w D & 0 \end{bmatrix} \equiv S \begin{bmatrix} A_u & I_n \end{bmatrix} + \begin{bmatrix} E & -S \end{bmatrix}.$$

Using

$$B' = \begin{bmatrix} A_w D & 0 \end{bmatrix} \text{ and } E' = \begin{bmatrix} E & -S \end{bmatrix},$$

we can write this as

$$B' \equiv S A'_u + E' \mod q.$$

[3] The straightforward projection may yield information about the basis. So instead we sample a nearby vector according to some pre-determined probability distribution, see [GPV07, Sec. 4.2] for more details.

[4] A trivial encoding of 0 is just the 0 matrix, which corresponds to an error matrix $E = 0$.

This is an instance of the learning-with-errors problem (LWE).

We can solve for S using our trapdoor. Notice that by construction,

$$B'T' \equiv E'T' \mod q.$$

Recall that E' and T' are both small, so the usual lift of $E'T' \mod q$ must equal $E'T'$ as a matrix over $\mathbb{Z}$. That is, $B'T' = E'T'$ over $\mathbb{Z}$, not just $\mathbb{Z}/q\mathbb{Z}$. Now working over $\mathbb{Q}$, we can solve for $E' = (E'T')(T')^{-1}$. Once we have E' we can recover $S \mod q$, which is the same as S because S is small.

Remark 4. Note that this scheme attempts to limit the possible operations on encodings: only multiplications that agree with the graph structure are legal. This is supposed to make it more difficult to create valid encodings of 0.

Remark 5. The matrix T' is not a trapdoor for A_u, only for A'_u. It is unclear if you can recover a trapdoor for A_u given T'.

3.5 Recovering A_u

Suppose we know a bunch of encodings $D_1, \ldots, D_k$ and their corresponding plaintexts $S_1, \ldots, S_k$ along a path $u \rightsquigarrow v$. So

$$A_u D_j \equiv S_j A_v + E_j \mod q.$$

We will show that if we know A_u and a trapdoor T_v for A_v, then can recover A_v.

Recall that $A_u D_j T_v \equiv E_j T_v \mod q$. Since E_j and T_v are both small, we have that $A_u D_j T_v = E_j T_v$ over $\mathbb{Z}$. Hence we compute $A_u D_j T_v$, reduce modulo q, lift back to $\mathbb{Z}$, and then multiply by T_v^{-1} and we get E_j.

Now we can compute $A_u D_j - E_j \equiv S_j A_v \mod q$. Given enough of these equations, we can recover A_v.

An open question is whether this attack can be extended to use only an encoding of 0 on $u \rightsquigarrow v$ instead of a full trapdoor.

References

[Ajt96a] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 99–108. ACM, New York, 1996.

[Ajt96b] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 99–108. ACM, New York, 1996.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In *Advances in cryptology—EUROCRYPT 2017. Part II*, volume 10211 of *Lecture Notes in Comput. Sci.*, pages 103–129. Springer, Cham, 2017.

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of ggh15: Provable security against zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *Theory of Cryptography*, pages 544–574, Cham, 2018. Springer International Publishing.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In *Advances in cryptology—CRYPTO 2016. Part II*, volume 9815 of *Lecture Notes in Comput. Sci.*, pages 607–628. Springer, Berlin, 2016.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of cryptography. Part II*, volume 9015 of *Lecture Notes in Comput. Sci.*, pages 498–527. Springer, Heidelberg, 2015.

[GPV07] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. Cryptology ePrint Archive, Report 2007/432, 2007. https://eprint.iacr.org/2007/432.