Cryptanalysis of GGH15 Multilinear Maps

Jean-Sébastien Coron[1], Moon Sung Lee[1],
Tancrède Lepoint[2], and Mehdi Tibouchi[3]

[1] University of Luxembourg
[2] CryptoExperts
[3] NTT Secure Platform Laboratories

Abstract. We describe a cryptanalysis of the GGH15 multilinear maps. Our attack breaks the multipartite key-agreement protocol in polynomial time by generating an equivalent user private key; it also applies to GGH15 with safeguards. We also describe attacks against variants of the GGH13 multilinear maps proposed by Halevi (ePrint 2015/866) aiming at supporting graph-induced constraints, as in GGH15.

1 Introduction

Multilinear maps. For the past couple of years, cryptographic multilinear maps have found numerous applications in the design of cryptographic protocols, the most salient example of which is probably the construction of indistinguishability obfuscation (iO) [GGH+13b]. The first multilinear maps candidate (GGH13) was described by Garg, Gentry and Halevi [GGH13a] from ideal lattices. It was then followed by another candidate (aka, CLT13) due to Coron, Lepoint and Tibouchi [CLT13] using the same techniques but over the integers, and later by a third candidate (GGH15) by Gentry, Gorbunov and Halevi [GGH15], related to the homomorphic encryption scheme from [GSW13].

Unfortunately, these candidates do not rely on well-established hardness assumptions, and recent months have witnessed a number of attacks (including [CHL+15], [CGH+15], [HJ16], [BGH+15], [PS15], [CFL+16]) showing that they fail to meet a number of desirable security requirements, and that they cannot be used to securely instantiate such and such protocols. Some attempts to protect against these attacks have also known a similar fate [CLT15, BGH+15]. The security of the constructions based on these multilinear maps is currently unclear to the community [Hal15]. While two recent works [CGH+15, MSZ16] have shown polynomial-time attacks against some obfuscation candidates, many iO candidates remain unaffected by the attacks proposed so far. The same cannot be said for the more immediate application of multilinear maps that is one-round multipartite key agreement.

One-round multipartite key-agreement protocol. Since its discovery in 1976, the Diffie–Hellman protocol [DH76] is one of the most widely used cryptographic protocol to create a common secret between two parties. A generalization

of this one-round protocol to three parties was proposed in 2000 by Joux [Jou00] using cryptographic *bilinear* maps; it was later extended to $k \geq 4$ parties assuming the existence of a cryptographic $(k-1)$-linear map by Boneh and Silverberg [BS02]. In a nutshell, the protocol works as follows: assuming some public parameters are shared by all the parties, each party broadcasts some data and keeps some data secret, and then by combining their secret data with the other parties' published values using the multilinear map, they can derive a shared common secret key.

The first candidates for a k-partite Diffie–Hellman key-agreement protocol for arbitrary k were described in [GGH13a, CLT13] using respectively the GGH13 and CLT13 multilinear maps candidates. Unfortunately, the protocols were later shown to be insecure in [HJ16, CHL+15]: using the public parameters and the broadcast data, an eavesdropper can recover the shared common secret key in polynomial time.

The GGH15 key-agreement protocol. Since the third proposed multilinear maps scheme, GGH15, does not fit the same graded encoding framework as the earlier candidates, one needs new constructions to use it to instantiate cryptographic protocols. And the first such application was again a Diffie–Hellman key-agreement protocol [GGH15, Section 5.1]. To avoid similar attacks as the one that targeted GGH13 and CLT13, based on encodings of zero, the protocol was designed in such a way that the adversary is never given encodings of the same element that could be subtracted without doing the full key-agreement computation. Namely, each party i has a directed path of matrices $\boldsymbol{A}_{i,1}, \ldots, \boldsymbol{A}_{i,k+1}$ all sharing the same end-point $\boldsymbol{A}_{i,k+1} = \boldsymbol{A}_0$, and has a secret value s_i. She can then publish encodings of s_i on the chains of the other parties in a "round robin" fashion, *i.e.* s_i is encoded on the j-th edge of the chain of the party $i' = j - i + 1$, with index arithmetic modulo k. The graph for 3 parties is illustrated in Figure 1.

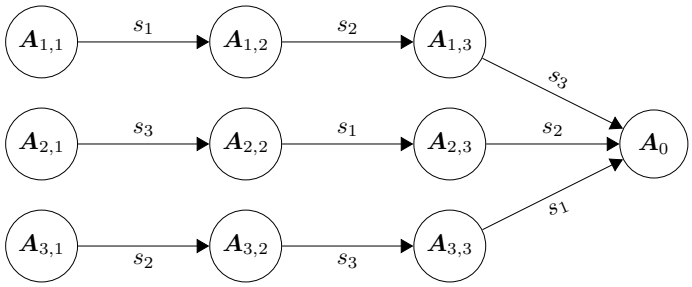

Fig. 1. Graph for a 3-partite key-agreement protocol with GGH15 multilinear maps.

On the i-th chain, Party i will then be able to multiply these encodings (the one he kept secret and the ones published by the other parties) to get an encoding of $\prod_j s_j$ relative to the path $\boldsymbol{A}_{i,1} \rightsquigarrow \boldsymbol{A}_0$. Now, since the encodings of s_i cannot

be mixed before the end-point $\boldsymbol{A}_0$, it seems difficult to obtain an encoding of 0 on an edge in the middle of the graph to mount "zeroizing attacks" [GGH15].

Halevi's candidate key-agreement protocols. As no attack was known on GGH15 multilinear maps and in an attempt to reinstate a key-agreement protocol for GGH13, Halevi recently proposed, on the Cryptology ePrint Archive, two variants of GGH13 supporting a similar key-agreement protocol [Hal15].[1] The first variant uses the "asymmetric" GGH13 scheme to handle the graph structure [Hal15, Section 7]. Namely, in basic GGH13 each encoding is multiplicatively masked by a power z^i of a secret mask z; in asymmetric GGH13, the encodings can be masked by powers of multiple z_j's. Therefore, in this new key-agreement protocol candidate, the public encodings are now associated with independent masks $z_{i,j}$'s such that their product yields the same value Z, *i.e.* $\prod_j z_{i,j} = Z$ for all i (so that the final encoding shall extract to the same shared key). The graph for 3 parties is illustrated in Figure 2.

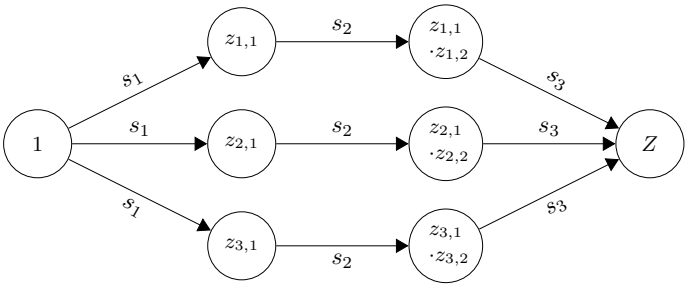

Fig. 2. Multipartite key agreement from asymmetric GGH13, with 3 parties, from [Hal15, Section 7].

Once again, the fact that the encodings of the same value s_i are multiplied with different masks gives hope that no encoding of 0 multiplied by a value other than Z can be obtained, and therefore that zeroizing attacks are impossible [GGH13a, CGH$^+$15].

A second variant of GGH13, which we refer to as Graph-GGH13, mimics the structure of GGH15 encodings more closely and is described in [Hal15, Section 6]. An encoding $c \in \alpha + gR$ relative to a path $u \rightsquigarrow v$ is now a matrix $\tilde{\boldsymbol{C}} = \boldsymbol{P}_u^{-1} \cdot \boldsymbol{C} \cdot \boldsymbol{P}_v$, where $\boldsymbol{C} \in \mathbb{Z}_q^{n \times n}$ is the multiply-by-c matrix, and the $\boldsymbol{P}_w$'s are secret random matrices. In the key-agreement protocol, each party i has a directed path of matrices $\boldsymbol{P}_{i,1}, \ldots, \boldsymbol{P}_{i,k+1}$ all sharing the same end-point $\boldsymbol{P}_{i,k+1} = \boldsymbol{P}_0$ and the same start-point $\boldsymbol{P}_{i,1} = \boldsymbol{P}_1$, and has a secret value s_i. She can then publish

[1] As mentioned in the last remark of the paper, although the key-agreement protocol can be described also based on CLT13, the attacks from [CGH$^+$15] can be used to break it.

encodings of s_i on the chains of the other parties in a "round robin" fashion. The graph for 3 parties is illustrated in Figure 3.

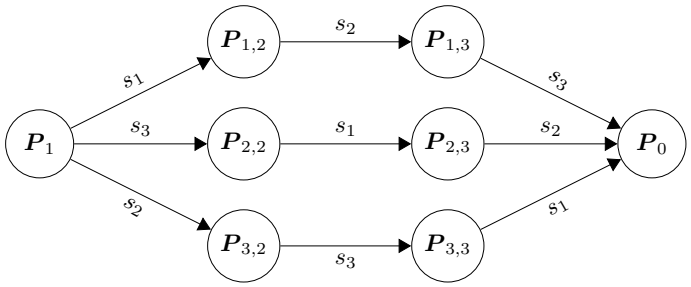

Fig. 3. Multipartite key agreement from GGH13 with graph constraints, with 3 parties, from [Hal15, Section 6].

And here again, the fact that the encodings corresponding to the same s_i are multiplied on the left and on the right by completely random matrices $\boldsymbol{P}_{i,j}$ makes it difficult to cancel them out and obtain an encoding of 0 without evaluating the full "chains" (that is, the operations of the key agreement itself).

Finally, in order to capture the intuition of what it means for an attacker to break the scheme, Halevi defined, for both schemes, the "core computational task" of an adversary as recovering any basis of the (hidden) plaintext space [Hal15, Section 2.2].

Our contributions. Our main contribution is to describe a cryptanalysis of the Diffie–Hellman key-agreement protocol when instantiated with GGH15 multilinear maps. Our attack makes it possible to generate an equivalent user private key in polynomial time, which in turn allows to recover the shared session key. Our attack proceeds in two steps: in the first step, we express the secret exponent of one user as a linear combination of some other secret exponents corresponding to public encodings, using a variant of the Cheon *et al.* attack [CHL$^+$15]. This does not immediately break the protocol because the coefficients of the linear combination can be large. In the second step, we use the previous linear combination to derive an encoding equivalent to the user private encoding, by correcting the error resulting from the large coefficients of the linear combination. Our attack also applies to GGH15 with safeguards; we extend the basic attack by using another linear relation to estimate the error incurred from the large coefficients, thus enabling to recover the shared session key.

In the full version of this paper [CLLT15], we also describe attacks that break both variants of GGH13 proposed by Halevi in [Hal15]. Our attacks apply some variant of the Cheon *et al.* attack [CHL$^+$15] to recover a basis of the secret

plaintext space R/gR in polynomial time. This was considered as the "core computational task of an attacker" in [Hal15].

Source code. A proof-of-concept implementation of our cryptanalysis of GGH15, using the Sage [Dev16] mathematics software system, is available at:

`http://pastebin.com/7kZHnTXY`

2 The GGH15 Multilinear Map Scheme

We briefly recall the GGH15 multilinear map scheme; we refer to [GGH15] for a full description. In the following we only consider the commutative variant from [GGH15, Section 3.2], as only that commutative variant can be used in the multipartite key-agreement protocol from [GGH15, Section 5.1].

2.1 GGH15 Multilinear Maps

The construction works over polynomial rings $R = \mathbb{Z}[x]/(f(x))$ and $R_q = R/qR$ for some degree n irreducible integer polynomial $f(x) \in \mathbb{Z}[x]$ and an integer q. The construction is parametrized by a directed acyclic graph $G = (V, E)$. To each node $u \in V$ a random row vector $\boldsymbol{A}_u \in R_q^m$ is assigned, where m is a parameter. An encoding of a small plaintext element $s \in R$ relative to path $u \rightsquigarrow v$ is a matrix with small coefficients $\boldsymbol{D} \in R^{m \times m}$ such that:

$$\boldsymbol{A}_u \cdot \boldsymbol{D} = s \cdot \boldsymbol{A}_v + \boldsymbol{E} \pmod q$$

where $\boldsymbol{E}$ is a small error vector of dimension m with components in R; we refer to [GGH15] for how such encoding $\boldsymbol{D}$ can be generated, based on a trapdoor sampling procedure from [MP12]. Only small plaintext elements $s \in R$ are encoded. As in [Hal15] we use the row vector notation for $\boldsymbol{A}_u$, rather than the column vector notation used in [GGH15].[2] It is easy to see that two encodings $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ relative to the same path $u \rightsquigarrow v$ can be added; namely from:

$$\begin{aligned} \boldsymbol{A}_u \cdot \boldsymbol{D}_1 &= s_1 \cdot \boldsymbol{A}_v + \boldsymbol{E}_1 \pmod q \\ \boldsymbol{A}_u \cdot \boldsymbol{D}_2 &= s_2 \cdot \boldsymbol{A}_v + \boldsymbol{E}_2 \pmod q \end{aligned}$$

we obtain:

$$\boldsymbol{A}_u \cdot (\boldsymbol{D}_1 + \boldsymbol{D}_2) = (s_1 + s_2) \cdot \boldsymbol{A}_v + \boldsymbol{E}_1 + \boldsymbol{E}_2 \pmod q .$$

Moreover two encodings $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ relative to path $u \rightsquigarrow v$ and $v \rightsquigarrow w$ can be multiplied to get an encoding relative to path $u \rightsquigarrow w$. Namely given:

$$\begin{aligned} \boldsymbol{A}_u \cdot \boldsymbol{D}_1 &= s_1 \cdot \boldsymbol{A}_v + \boldsymbol{E}_1 \pmod q \\ \boldsymbol{A}_v \cdot \boldsymbol{D}_2 &= s_2 \cdot \boldsymbol{A}_w + \boldsymbol{E}_2 \pmod q \end{aligned}$$

[2] With the column vector notation, the corresponding equation in [GGH15] is $\boldsymbol{D} \cdot \boldsymbol{A}_u = s \cdot \boldsymbol{A}_v + \boldsymbol{E} \pmod q$.

we obtain by multiplying the matrix encodings $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$:

$$\begin{aligned}\boldsymbol{A}_u \cdot \boldsymbol{D}_1 \cdot \boldsymbol{D}_2 &= (s_1 \cdot \boldsymbol{A}_v + \boldsymbol{E}_1) \cdot \boldsymbol{D}_2 \pmod q \\ &= s_1 \cdot s_2 \cdot \boldsymbol{A}_w + s_1 \cdot \boldsymbol{E}_2 + \boldsymbol{E}_1 \cdot \boldsymbol{D}_2 \pmod q \\ &= s_1 \cdot s_2 \cdot \boldsymbol{A}_w + \boldsymbol{E'} \pmod q\end{aligned}$$

for some new error vector $\boldsymbol{E'}$. Since s_1, $\boldsymbol{E}_1$, $\boldsymbol{E}_2$ and $\boldsymbol{D}_2$ have small coefficients, $\boldsymbol{E'}$ still has small coefficients (compared to q), and therefore the product $\boldsymbol{D}_1 \cdot \boldsymbol{D}_2$ is an encoding of $s_1 \cdot s_2$ for the path $u \rightsquigarrow w$.

Finally, given an encoding $\boldsymbol{D}$ relative to path $u \rightsquigarrow w$ and the vector $\boldsymbol{A}_u$, extraction works by computing the high-order bits of $\boldsymbol{A}_u \cdot \boldsymbol{D}$. Namely we have:

$$\boldsymbol{A}_u \cdot \boldsymbol{D} = s \cdot \boldsymbol{A}_w + \boldsymbol{E} \pmod q$$

for some small $\boldsymbol{E}$, and therefore the high-order bits of $\boldsymbol{A}_u \cdot \boldsymbol{D}$ only depend on the secret exponent s.

Remark 1. As emphasized in [GGH15], only the plaintext space of the s_i's is commutative, not the space of the encoding matrices $\boldsymbol{D}_i$. The ability to multiply the plaintext elements s_i in arbitrary order will be used in the multipartite key-agreement protocol below.

2.2 The GGH15 Multipartite Key-Agreement Protocol

We briefly recall the multipartite key-agreement protocol from [GGH15, Section 5.1]. We consider the protocol with k users. As illustrated in Figure 4 for $k = 3$ users, each user i for $1 \leq i \leq k$ has a directed path of vectors $\boldsymbol{A}_{i,1}, \ldots, \boldsymbol{A}_{i,k+1}$, all sharing the same end-point $\boldsymbol{A}_0 = \boldsymbol{A}_{i,k+1}$. The i-th user will use the resulting chain to extract the session key. Each user i has a secret exponent s_i. Each secret exponent s_i will be encoded in each of the k chains; the encoding of s_i on the j-th chain for $j \neq i$ will be published, while the encoding of s_i on the i-th chain will be kept private by user i. Therefore on the i-th chain only user i will be able to compute the session key. The exponents s_i are encoded in a "round robin" fashion; namely the i-th secret s_i is encoded on the chain of user j at edge $\ell = i + j - 1$, with index arithmetic modulo k. Only the vectors $\boldsymbol{A}_{i,1}$ for $1 \leq i \leq k$ are made public to enable extraction of the session-key; the others are kept private. We recall the formal description of the protocol in the full version of this paper [CLLT15].

We illustrate the protocol for $k = 3$ users. For the chain corresponding to User 1, we have the following encodings:

$$\begin{aligned}\boldsymbol{A}_{1,1} \cdot \boldsymbol{D}_{1,1} &= s_1 \cdot \boldsymbol{A}_{1,2} + \boldsymbol{F}_{1,1} \pmod q \\ \boldsymbol{A}_{1,2} \cdot \boldsymbol{D}_{1,2} &= s_2 \cdot \boldsymbol{A}_{1,3} + \boldsymbol{F}_{1,2} \pmod q \\ \boldsymbol{A}_{1,3} \cdot \boldsymbol{D}_{1,3} &= s_3 \cdot \boldsymbol{A}_0 + \boldsymbol{F}_{1,3} \pmod q\end{aligned}$$

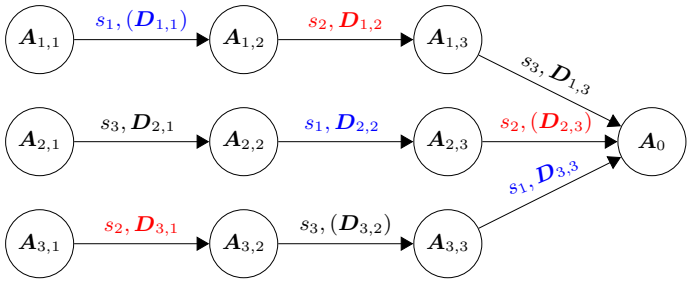

Fig. 4. Graph of a key agreement between 3 parties for GGH15. The vertices contain random vectors $\boldsymbol{A}_{ij}$, and encodings are represented on the edges. Each party is represented by a different color, keeps the encoding in parenthesis secret and publishes the two other encodings.

where $\boldsymbol{D}_{1,2}$ and $\boldsymbol{D}_{1,3}$ are public while $\boldsymbol{D}_{1,1}$ is kept private by User 1. Therefore User 1 can compute modulo q:

$$\begin{aligned}\boldsymbol{A}_{1,1} \cdot \boldsymbol{D}_{1,1} \cdot \boldsymbol{D}_{1,2} \cdot \boldsymbol{D}_{1,3} &= (s_1 \cdot \boldsymbol{A}_{1,2} + \boldsymbol{F}_{1,1}) \cdot \boldsymbol{D}_{1,2} \cdot \boldsymbol{D}_{1,3} \pmod q \\ &= (s_1 \cdot s_2 \cdot \boldsymbol{A}_{1,3} + s_1 \cdot \boldsymbol{F}_{1,2} + \\ &\quad \boldsymbol{F}_{1,1} \cdot \boldsymbol{D}_{1,2}) \cdot \boldsymbol{D}_{1,3} \pmod q .\end{aligned}$$

Letting $\hat{\boldsymbol{F}}_{1,2} := s_1 \cdot \boldsymbol{F}_{1,2} + \boldsymbol{F}_{1,1} \cdot \boldsymbol{D}_{1,2}$, we obtain:

$$\begin{aligned}\boldsymbol{A}_{1,1} \cdot \boldsymbol{D}_{1,1} \cdot \boldsymbol{D}_{1,2} \cdot \boldsymbol{D}_{1,3} &= \left(s_1 \cdot s_2 \cdot \boldsymbol{A}_{1,3} + \hat{\boldsymbol{F}}_{1,2}\right) \cdot \boldsymbol{D}_{1,3} \pmod q \\ &= s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0 + s_1 \cdot s_2 \cdot \boldsymbol{F}_{1,3} + \hat{\boldsymbol{F}}_{1,2} \cdot \boldsymbol{D}_{1,3} \pmod q .\end{aligned}$$

Since s_1, s_2 and s_3 are small and $\boldsymbol{F}_{1,3}$, $\hat{\boldsymbol{F}}_{1,2}$ and $\boldsymbol{D}_{1,3}$ have small components, User 1 can extract the most significant bits corresponding to $s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0$. Similarly User 2 will compute the session key using the following chain, where $\boldsymbol{D}_{2,1}$ and $\boldsymbol{D}_{2,2}$ are public while $\boldsymbol{D}_{2,3}$ is private to User 2:

$$\begin{aligned}\boldsymbol{A}_{2,1} \cdot \boldsymbol{D}_{2,1} &= s_3 \cdot \boldsymbol{A}_{2,2} + \boldsymbol{F}_{2,1} \pmod q \\ \boldsymbol{A}_{2,2} \cdot \boldsymbol{D}_{2,2} &= s_1 \cdot \boldsymbol{A}_{2,3} + \boldsymbol{F}_{2,2} \pmod q \\ \boldsymbol{A}_{2,3} \cdot \boldsymbol{D}_{2,3} &= s_2 \cdot \boldsymbol{A}_0 + \boldsymbol{F}_{2,3} \pmod q .\end{aligned}$$

Namely User 2 can compute:

$$\begin{aligned}\boldsymbol{A}_{2,1} \cdot \boldsymbol{D}_{2,1} \cdot \boldsymbol{D}_{2,2} \cdot \boldsymbol{D}_{2,3} &= (s_3 \cdot s_1 \cdot \boldsymbol{A}_{2,3} + s_3 \cdot \boldsymbol{F}_{2,2} + \\ &\quad \boldsymbol{F}_{2,1} \cdot \boldsymbol{D}_{2,2}) \cdot \boldsymbol{D}_{2,3} \pmod q \\ &= s_3 \cdot s_1 \cdot s_2 \cdot \boldsymbol{A}_0 + \boldsymbol{F} \pmod q\end{aligned}$$

for some small vector $\boldsymbol{F}$, and extract the same most significant bits corresponding to $s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0$; the same holds for User 3.

The previous encodings are generated by random linear combination of public encodings, corresponding to secret exponents $t_{i,\ell}$ for $1 \leq \ell \leq N$, for large enough N. More precisely, for each $1 \leq i \leq k$ one generates random small plaintext elements $t_{i,\ell}$ for $1 \leq \ell \leq N$, which are then encoded on all chains j at edge $i' = i + j - 1$ (with index modulo k), by $\boldsymbol{C}_{j,i',\ell}$. This means that for $k = 3$ users, we have the following encodings corresponding to User 1:

$$\begin{aligned}
\boldsymbol{A}_{1,1} \cdot \boldsymbol{C}_{1,1,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_{1,2} + \boldsymbol{E}_{1,1,\ell} \pmod q \\
\boldsymbol{A}_{2,2} \cdot \boldsymbol{C}_{2,2,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_{2,3} + \boldsymbol{E}_{2,2,\ell} \pmod q \\
\boldsymbol{A}_{3,3} \cdot \boldsymbol{C}_{3,3,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_{0} + \boldsymbol{E}_{3,3,\ell} \pmod q
\end{aligned}$$

and the tuple $(\boldsymbol{D}_{1,1}, \boldsymbol{D}_{2,2}, \boldsymbol{D}_{3,3})$ is generated by linear combination of the tuple $(\boldsymbol{C}_{1,1,\ell}, \boldsymbol{C}_{2,2,\ell}, \boldsymbol{C}_{3,3,\ell})$, so that the matrices $\boldsymbol{D}_{1,1}$, $\boldsymbol{D}_{2,2}$ and $\boldsymbol{D}_{3,3}$ encode the same secret exponent s_1; the same holds for users 2 and 3. We refer to the full version of this paper [CLLT15] for the formal description of the protocol.

3 Cryptanalysis of GGH15 Without Safeguards

In the following we describe a cryptanalysis of the multipartite key-agreement protocol based on GGH15 multilinear maps recalled in the previous section. Heuristically our attack recovers the session-key from public element in polynomial-time. Our attack proceeds in two steps.

1. In the first step, we are able to express one secret exponent s_1 as a linear combination of the other secret exponents $t_{1,\ell}$, using a variant of the Cheon *et al.* attack [CHL+15]. However this does not immediately break the protocol, because the coefficients are not small.
2. In the second step, we compute an equivalent of the private encoding of User 1 from the previous linear combination, by correcting the error due to the large coefficients. This breaks the key-exchange protocol.

3.1 Description With 3 Users

For simplicity we first consider the protocol with only 3 users; the extension to $k \geq 3$ users is relatively straightforward and described in the full version of this paper [CLLT15]. Therefore we consider the following 3 rows corresponding to the 3 users:

$$\begin{aligned}
\boldsymbol{A}_{1,1} \cdot \boldsymbol{D}_{1,1} &= s_1 \cdot \boldsymbol{A}_{1,2} + \boldsymbol{F}_{1,1} \pmod q \\
\boldsymbol{A}_{1,2} \cdot \boldsymbol{D}_{1,2} &= s_2 \cdot \boldsymbol{A}_{1,3} + \boldsymbol{F}_{1,2} \pmod q \\
\boldsymbol{A}_{1,3} \cdot \boldsymbol{D}_{1,3} &= s_3 \cdot \boldsymbol{A}_{0} + \boldsymbol{F}_{1,3} \pmod q
\end{aligned}
\qquad
\begin{aligned}
\boldsymbol{A}_{1,1} \cdot \boldsymbol{C}_{1,1,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_{1,2} + \boldsymbol{E}_{1,1,\ell} \pmod q \\
\boldsymbol{A}_{1,2} \cdot \boldsymbol{C}_{1,2,\ell} &= t_{2,\ell} \cdot \boldsymbol{A}_{1,3} + \boldsymbol{E}_{1,2,\ell} \pmod q \\
\boldsymbol{A}_{1,3} \cdot \boldsymbol{C}_{1,3,\ell} &= t_{3,\ell} \cdot \boldsymbol{A}_{0} + \boldsymbol{E}_{1,3,\ell} \pmod q
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{A}_{2,1} \cdot \boldsymbol{D}_{2,1} &= s_3 \cdot \boldsymbol{A}_{2,2} + \boldsymbol{F}_{2,1} \pmod q \\
\boldsymbol{A}_{2,2} \cdot \boldsymbol{D}_{2,2} &= s_1 \cdot \boldsymbol{A}_{2,3} + \boldsymbol{F}_{2,2} \pmod q \\
\boldsymbol{A}_{2,3} \cdot \boldsymbol{D}_{2,3} &= s_2 \cdot \boldsymbol{A}_{0} + \boldsymbol{F}_{2,3} \pmod q
\end{aligned}
\qquad
\begin{aligned}
\boldsymbol{A}_{2,1} \cdot \boldsymbol{C}_{2,1,\ell} &= t_{3,\ell} \cdot \boldsymbol{A}_{2,2} + \boldsymbol{E}_{2,1,\ell} \pmod q \\
\boldsymbol{A}_{2,2} \cdot \boldsymbol{C}_{2,2,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_{2,3} + \boldsymbol{E}_{2,2,\ell} \pmod q \\
\boldsymbol{A}_{2,3} \cdot \boldsymbol{C}_{2,3,\ell} &= t_{2,\ell} \cdot \boldsymbol{A}_{0} + \boldsymbol{E}_{2,3,\ell} \pmod q
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{A}_{3,1} \cdot \boldsymbol{D}_{3,1} &= s_2 \cdot \boldsymbol{A}_{3,2} + \boldsymbol{F}_{3,1} \pmod q \\
\boldsymbol{A}_{3,2} \cdot \boldsymbol{D}_{3,2} &= s_3 \cdot \boldsymbol{A}_{3,3} + \boldsymbol{F}_{3,2} \pmod q \\
\boldsymbol{A}_{3,3} \cdot \boldsymbol{D}_{3,3} &= s_1 \cdot \boldsymbol{A}_{0} + \boldsymbol{F}_{3,3} \pmod q
\end{aligned}
\qquad
\begin{aligned}
\boldsymbol{A}_{3,1} \cdot \boldsymbol{C}_{3,1,\ell} &= t_{2,\ell} \cdot \boldsymbol{A}_{3,2} + \boldsymbol{E}_{3,1,\ell} \pmod q \\
\boldsymbol{A}_{3,2} \cdot \boldsymbol{C}_{3,2,\ell} &= t_{3,\ell} \cdot \boldsymbol{A}_{3,3} + \boldsymbol{E}_{3,2,\ell} \pmod q \\
\boldsymbol{A}_{3,3} \cdot \boldsymbol{C}_{3,3,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_{0} + \boldsymbol{E}_{3,3,\ell} \pmod q
\end{aligned}$$

where all encodings $\boldsymbol{C}_{i,j,\ell}$ and $\boldsymbol{D}_{i,j}$ are public, except $\boldsymbol{D}_{1,1}$ which is private on Row 1, $\boldsymbol{D}_{2,3}$ is private on Row 2, and $\boldsymbol{D}_{3,2}$ is private on Row 3. The corresponding graph is illustrated in Figure 4. Note that on each row we have used the same index ℓ for $t_{1,\ell}$, $t_{2,\ell}$ and $t_{3,\ell}$, but on a given row one can obviously compute product of encodings for different indices.

First step: linear relations. In the first step of the attack, we show that we can express s_1 as a linear combinations of the $t_{1,\ell}$'s. For this we consider the rows 2 and 3, for which the encodings $\boldsymbol{D}_{2,2}$ and $\boldsymbol{D}_{3,3}$ corresponding to s_1 are public. In the remaining of the attack, we always consider a fixed index $\ell = 1$ for the encodings corresponding to $t_{3,\ell}$, and for simplicity we write $t_3 := t_{3,1}$, $\boldsymbol{C}_{1,3} := \boldsymbol{C}_{1,3,1}$, $\boldsymbol{C}_{2,1} := \boldsymbol{C}_{2,1,1}$ and $\boldsymbol{C}_{3,2} := \boldsymbol{C}_{3,2,1}$.

Since we always work with the same t_3, on Row 2 we define the product encodings $\hat{\boldsymbol{C}}_{2,2,\ell} := \boldsymbol{C}_{2,1} \cdot \boldsymbol{C}_{2,2,\ell}$, and on Row 3 we define the product encodings $\hat{\boldsymbol{C}}_{3,2,\ell} := \boldsymbol{C}_{3,1,\ell} \cdot \boldsymbol{C}_{3,2}$; recall that we use a fixed index for t_3. Therefore we can write:

$$\begin{aligned}
\boldsymbol{A}_{2,1} \cdot \hat{\boldsymbol{C}}_{2,2,\ell} &= t_{1,\ell} \cdot t_3 \cdot \boldsymbol{A}_{2,3} + \hat{\boldsymbol{E}}_{2,2,\ell} \pmod q \\
\boldsymbol{A}_{2,3} \cdot \boldsymbol{C}_{2,3,\ell} &= t_{2,\ell} \cdot \boldsymbol{A}_0 + \boldsymbol{E}_{2,3,\ell} \pmod q \\
\boldsymbol{A}_{3,1} \cdot \hat{\boldsymbol{C}}_{3,2,\ell} &= t_{2,\ell} \cdot t_3 \cdot \boldsymbol{A}_{3,3} + \hat{\boldsymbol{E}}_{3,2,\ell} \pmod q \\
\boldsymbol{A}_{3,3} \cdot \boldsymbol{C}_{3,3,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_0 + \boldsymbol{E}_{3,3,\ell} \pmod q
\end{aligned} \tag{1}$$

for some small error vectors $\hat{\boldsymbol{E}}_{2,2,\ell}$ and $\hat{\boldsymbol{E}}_{3,2,\ell}$.

For simplicity of notations, we first consider a fixed index i for the encodings corresponding to $t_{1,i}$, and we write $t_1 := t_{1,i}$, $\hat{\boldsymbol{C}}_{2,2} := \hat{\boldsymbol{C}}_{2,2,i}$ and $\boldsymbol{C}_{3,3} := \boldsymbol{C}_{3,3,i}$. Similarly we consider a fixed index j for the encodings corresponding to $t_{2,j}$ and we write $t_2 := t_{2,j}$, $\boldsymbol{C}_{2,3} := \boldsymbol{C}_{2,3,j}$ and $\hat{\boldsymbol{C}}_{3,2} := \hat{\boldsymbol{C}}_{3,2,j}$. We use similar notations for the corresponding error vectors.

All previous equations hold modulo q only. To get a result over R instead of only modulo q, we compute the difference between two rows, for the same product of secret exponents. More precisely, we compute:

$$\begin{aligned}
\boldsymbol{\omega} &= \boldsymbol{A}_{2,1} \cdot \hat{\boldsymbol{C}}_{2,2} \cdot \boldsymbol{C}_{2,3} - \boldsymbol{A}_{3,1} \cdot \hat{\boldsymbol{C}}_{3,2} \cdot \boldsymbol{C}_{3,3} \qquad (2) \\
&= t_1 \cdot t_3 \cdot t_2 \cdot \boldsymbol{A}_0 + t_1 \cdot t_3 \cdot \boldsymbol{E}_{2,3} + \hat{\boldsymbol{E}}_{2,2} \cdot \boldsymbol{C}_{2,3} \\
&\quad - t_2 \cdot t_3 \cdot t_1 \cdot \boldsymbol{A}_0 - t_2 \cdot t_3 \cdot \boldsymbol{E}_{3,3} - \hat{\boldsymbol{E}}_{3,2} \cdot \boldsymbol{C}_{3,3} \\
&= t_1 \cdot t_3 \cdot \boldsymbol{E}_{2,3} + \hat{\boldsymbol{E}}_{2,2} \cdot \boldsymbol{C}_{2,3} - t_2 \cdot t_3 \cdot \boldsymbol{E}_{3,3} - \hat{\boldsymbol{E}}_{3,2} \cdot \boldsymbol{C}_{3,3} \,. \qquad (3)
\end{aligned}$$

Namely the latter equation holds over R (and not only modulo q) because all the terms in (3) have small coefficients; namely the only term $t_1 \cdot t_2 \cdot t_3 \cdot \boldsymbol{A}_0$ with large coefficients modulo q is canceled when doing the subtraction.

We have that $\boldsymbol{\omega}$ is a vector of dimension m. Now an important step is to restrict ourselves to the first component of $\boldsymbol{\omega}$. Namely in order to apply the same technique as in the Cheon *et al.* attack, we would like to express $\boldsymbol{\omega}$ as the product of two vectors, where the left vector corresponds to User 1 and the right vector

corresponds to User 2. However due to the "round-robin" fashion of exponent encodings, for this we would need to swap the product $\hat{\boldsymbol{E}}_{3,2} \cdot \boldsymbol{C}_{3,3}$ appearing in (3), since $\hat{\boldsymbol{E}}_{3,2}$ corresponds to User 2 while $\boldsymbol{C}_{3,3}$ corresponds to User 1; this cannot be done if we consider the full vector $\boldsymbol{\omega}$. By restricting ourselves to the first component of $\boldsymbol{\omega}$, the product $\hat{\boldsymbol{E}}_{3,2} \cdot \boldsymbol{C}_{3,3}$ becomes a simple scalar product that can be swapped; namely the scalar product of $\hat{\boldsymbol{E}}_{3,2}$ by the first column vector $\boldsymbol{C}'_{3,3}$ of the matrix $\boldsymbol{C}_{3,3}$. We obtain the scalar:

$$\omega = t_1 \cdot t_3 \cdot E_{2,3} + \hat{\boldsymbol{E}}_{2,2} \cdot \boldsymbol{C}'_{2,3} - t_2 \cdot t_3 \cdot E_{3,3} - \boldsymbol{C}'_{3,3} \cdot \hat{\boldsymbol{E}}_{3,2}$$

where $\boldsymbol{C}'_{2,3}$ and $\boldsymbol{C}'_{3,3}$ are the first column vectors of $\boldsymbol{C}_{2,3}$ and $\boldsymbol{C}_{3,3}$ respectively, both of dimension m; similarly $E_{2,3}$ and $E_{3,3}$ are the first components of $\boldsymbol{E}_{2,3}$ and $\boldsymbol{E}_{3,3}$ respectively.

We can now write ω as the scalar product of 2 vectors, the left one corresponding only to User 1, and the right one corresponding only to User 2:

$$\omega = \begin{bmatrix} t_1 & \hat{\boldsymbol{E}}_{2,2} & E_{3,3} & \boldsymbol{C}'_{3,3} \end{bmatrix} \cdot \begin{bmatrix} t_3 \cdot E_{2,3} \\ \boldsymbol{C}'_{2,3} \\ -t_2 \cdot t_3 \\ -\hat{\boldsymbol{E}}_{3,2} \end{bmatrix}.$$

Note that the two vectors in the product have dimension $2m + 2$.

As in the Cheon *et al.* attack [CHL+15], we can now extend ω to a matrix by considering many left row vectors and many right column vectors. However instead of a square matrix as in the Cheon *et al.* attack, we consider a rectangular matrix with $2m + 3$ rows and $2m + 2$ columns. In Equation (2), this is done by considering $2m + 3$ public encodings $\hat{\boldsymbol{C}}_{2,2,i}$ and $\boldsymbol{C}_{3,3,i}$ corresponding to User 1, and similarly $2m + 2$ encodings $\boldsymbol{C}_{2,3,j}$ and $\hat{\boldsymbol{C}}_{3,2,j}$ corresponding to User 2, for $1 \leq i \leq 2m + 3$ and $1 \leq j \leq 2m + 2$. More precisely we compute as previously over R the following matrix elements, restricting ourselves to the first component:

$$(\boldsymbol{W})_{ij} = \boldsymbol{A}_{2,1} \cdot \hat{\boldsymbol{C}}_{2,2,i} \cdot \boldsymbol{C}'_{2,3,j} - \boldsymbol{A}_{3,1} \cdot \hat{\boldsymbol{C}}_{3,2,j} \cdot \boldsymbol{C}'_{3,3,i} \tag{4}$$

and as previously we can write:

$$(\boldsymbol{W})_{ij} = \begin{bmatrix} t_{1,i} & \hat{\boldsymbol{E}}_{2,2,i} & E_{3,3,i} & \boldsymbol{C}'_{3,3,i} \end{bmatrix} \cdot \begin{bmatrix} t_3 \cdot E_{2,3,j} \\ \boldsymbol{C}'_{2,3,j} \\ -t_{2,j} \cdot t_3 \\ -\hat{\boldsymbol{E}}_{3,2,j} \end{bmatrix}.$$

We obtain a $(2m + 3) \times (2m + 2)$ matrix $\boldsymbol{W}$ with:

$$\boldsymbol{W} = \underbrace{\begin{bmatrix} & \cdots & \\ t_{1,i} & \hat{\boldsymbol{E}}_{2,2,i} \quad E_{3,3,i} & \boldsymbol{C}'_{3,3,i} \\ & \cdots & \end{bmatrix}}_{\boldsymbol{A}} \cdot \underbrace{\begin{bmatrix} & t_3 \cdot E_{2,3,j} & \\ \vdots & \boldsymbol{C}'_{2,3,j} & \vdots \\ & -t_{2,j} \cdot t_3 & \\ & -\hat{\boldsymbol{E}}_{3,2,j} & \end{bmatrix}}_{\boldsymbol{B}}$$

where the matrix $\boldsymbol{A}$ has $2m+3$ rows vectors, each of dimension $2m+2$, and the matrix $\boldsymbol{B}$ has $2m+2$ column vectors, each of dimension $2m+2$; hence $\boldsymbol{B}$ is a square matrix.

By doing linear algebra, we can find a vector $\boldsymbol{u}$ over R of dimension $2m+3$ such that $\boldsymbol{u} \cdot \boldsymbol{W} = 0$, which gives:

$$(\boldsymbol{u} \cdot \boldsymbol{A}) \cdot \boldsymbol{B} = 0\,.$$

Heuristically with good probability the matrix $\boldsymbol{B}$ is invertible, which implies:

$$\boldsymbol{u} \cdot \boldsymbol{A} = 0\,.$$

Since the first column of the matrix $\boldsymbol{A}$ is the column vector given by the $t_{1,i}$'s, such vector $\boldsymbol{u}$ gives a linear relation among the secret exponents $t_{1,i}$.

Moreover, since the encodings $\boldsymbol{D}_{2,2}$ and $\boldsymbol{D}_{3,3}$ corresponding to s_1 are public, we can express s_1 as a linear combination of the $t_{1,i}$'s, over R. Namely we can define as previously the product encoding $\hat{\boldsymbol{D}}_{2,2} := \boldsymbol{C}_{2,1} \cdot \boldsymbol{D}_{2,2}$, with:

$$\boldsymbol{A}_{2,1} \cdot \hat{\boldsymbol{D}}_{2,2} = s_1 \cdot t_3 \cdot \boldsymbol{A}_{2,3} + \hat{\boldsymbol{F}}_{2,2} \pmod{q}$$

for some small error vector $\hat{\boldsymbol{F}}_{2,2}$, and we can now compute the same $(\boldsymbol{W})_{ij}$ as in (4) but with $\hat{\boldsymbol{D}}_{2,2}$ and $\boldsymbol{D}'_{3,3}$ instead of $\hat{\boldsymbol{C}}_{2,2,i}$ and $\boldsymbol{C}'_{3,3,i}$, where $\boldsymbol{D}'_{3,3}$ is the first column of $\boldsymbol{D}_{3,3}$. More precisely, we compute for all $1 \leq j \leq 2m+2$:

$$\omega_j = \boldsymbol{A}_{2,1} \cdot \hat{\boldsymbol{D}}_{2,2} \cdot \boldsymbol{C}'_{2,3,j} - \boldsymbol{A}_{3,1} \cdot \hat{\boldsymbol{C}}_{3,2,j} \cdot \boldsymbol{D}'_{3,3}$$

which gives as previously:

$$\omega_j = \begin{bmatrix} s_1 & \hat{\boldsymbol{F}}_{2,2} & F_{3,3} & \boldsymbol{D}'_{3,3} \end{bmatrix} \cdot \begin{bmatrix} t_3 \cdot E_{2,3,j} \\ \boldsymbol{C}'_{2,3,j} \\ -t_{2,j} \cdot t_3 \\ -\hat{\boldsymbol{E}}_{3,2,j} \end{bmatrix}\,.$$

This implies that we can replace any row vector $[t_{1,i}\ \hat{\boldsymbol{E}}_{2,2,i}\ E_{3,3,i}\ \boldsymbol{C}'_{3,3,i}]$ in the matrix $\boldsymbol{A}$ by the row vector:

$$\begin{bmatrix} s_1 & \hat{\boldsymbol{F}}_{2,2} & F_{3,3} & \boldsymbol{D}'_{3,3} \end{bmatrix} \tag{5}$$

where $\boldsymbol{D}'_{3,3}$ is the first column of $\boldsymbol{D}_{3,3}$, and $F_{3,3}$ is the first component of $\boldsymbol{F}_{3,3}$. Using the previous technique, we can therefore obtain a linear relation between s_1 and the $t_{1,i}$'s over R. More precisely, with overwhelming probability, such a relation can be put in the form:

$$\mu \cdot s_1 = \sum_{i=1}^{2m+2} \lambda_i \cdot t_{1,i} \tag{6}$$

with $\mu \in \mathbb{Z}$ and $\lambda_1, \ldots, \lambda_{2m+2} \in R$. Indeed, we obtain such a relation by computing the kernel of the matrix analogous to $\boldsymbol{W}$ above in echelon form over

the fraction field of R, which gives the kernel of the corresponding matrix $\boldsymbol{A}$ (assuming that $\boldsymbol{B}$ is invertible). Unless a minor of that matrix vanishes, which happens with only negligible probability, this gives a relation where the coefficient of s_1 is 1 and the other coefficients are in the fraction field $R \otimes_{\mathbb{Z}} \mathbb{Q}$ of R. By clearing denominators, we get an expression of the form (6).

Then, by considering exactly one additional $t_{1,i}$ (say $t_{1,2m+3}$) and carrying out the same computations with indices $i = 2, \ldots, 2m+3$ instead of $i = 1, \ldots, 2m+2$, we get a second relation:

$$\nu \cdot s_1 = \sum_{i=2}^{2m+3} \lambda'_i \cdot t_{1,i} \,.$$

If the integers μ and ν are relatively prime, which happens with significant probability[3], we can apply Bézout's identity to obtain a linear relation in R where the coefficient of s_1 is 1:

$$s_1 = \sum_{i=1}^{2m+3} \alpha_i \cdot t_{1,i} \,. \tag{7}$$

Note that we have the same linear relations for the other components of the vector (5) corresponding to s_1, namely:

$$\hat{\boldsymbol{F}}_{2,2} = \sum_{i=1}^{2m+3} \alpha_i \cdot \hat{\boldsymbol{E}}_{2,2,i}, \quad F_{3,3} = \sum_{i=1}^{2m+3} \alpha_i \cdot E_{3,3,i}, \quad \boldsymbol{D}'_{3,3} = \sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{C}'_{3,3,i} \,. \tag{8}$$

Second step: equivalent private-key. In this second step, we show how to publicly compute an encoding equivalent to $\boldsymbol{D}_{1,1}$, which is private to User 1; this will break the key-agreement protocol. In the first step, we had considered rows 2 and 3 to derive the linear relations (7) and (8); we now consider Row 1. On Row 1, the encodings $\boldsymbol{D}_{1,2}$ and $\boldsymbol{D}_{1,3}$ are public, so we can define as previously the product encoding $\hat{\boldsymbol{D}}_{1,3} = \boldsymbol{D}_{1,2} \cdot \boldsymbol{D}_{1,3}$, which gives:

$$\boldsymbol{A}_{1,2} \cdot \hat{\boldsymbol{D}}_{1,3} = s_2 \cdot s_3 \cdot \boldsymbol{A}_0 + \hat{\boldsymbol{F}}_{1,3} \pmod{q}$$

for some small error vector $\hat{\boldsymbol{F}}_{1,3}$. Recall that the encoding $\boldsymbol{D}_{1,1}$ is private to User 1, with:

$$\boldsymbol{A}_{1,1} \cdot \boldsymbol{D}_{1,1} = s_1 \cdot \boldsymbol{A}_{1,2} + \boldsymbol{F}_{1,1} \pmod{q} \,. \tag{9}$$

Therefore only User 1 can privately compute:

$$\boldsymbol{A}_{1,1} \cdot \boldsymbol{D}_{1,1} \cdot \hat{\boldsymbol{D}}_{1,3} = s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0 + s_1 \cdot \hat{\boldsymbol{F}}_{1,3} + \boldsymbol{F}_{1,1} \cdot \hat{\boldsymbol{D}}_{1,3} \pmod{q} \tag{10}$$

[3] Heuristically, it is the probability that two random elements of R have coprime norms, since the rational integer denominator of an element of the fraction field has the same prime factors as its norm. For $R = \mathbb{Z}[x]/(x^{2^n}+1)$, that probability is close to 3/4: see the full version of this paper [CLLT15].

and extract the high order bits of $s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0 \bmod q$ to generate the session key.

We cannot compute the previous equation since $\boldsymbol{D}_{1,1}$ is private. However since we know a linear relation (7) between s_1 and the $t_{1,i}$'s, and the encodings $\boldsymbol{C}_{1,1,i}$ corresponding to $t_{1,i}$ are public, with:

$$\boldsymbol{A}_{1,1} \cdot \boldsymbol{C}_{1,1,i} = t_{1,i} \cdot \boldsymbol{A}_{1,2} + \boldsymbol{E}_{1,1,i} \pmod q$$

it is then natural to compute:

$$\tilde{\boldsymbol{D}}_{1,1} = \sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{C}_{1,1,i}\,,$$

which gives:

$$\boldsymbol{A}_{1,1} \cdot \tilde{\boldsymbol{D}}_{1,1} = s_1 \cdot \boldsymbol{A}_{1,2} + \sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \pmod q\,. \tag{11}$$

The difference with (9) is that the error term $\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i}$ is not necessarily small since the coefficients α_i can be large. Therefore if we compute:

$$\boldsymbol{A}_{1,1} \cdot \tilde{\boldsymbol{D}}_{1,1} \cdot \hat{\boldsymbol{D}}_{1,3} = s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0 + s_1 \cdot \hat{\boldsymbol{F}}_{1,3} + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{D}}_{1,3} \pmod q \tag{12}$$

then as opposed to (10) this does not reveal the high-order bits of $s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0 \bmod q$. In the following, we show how to derive an approximation of $\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i}$ over R, in order to correct the error in (11) and break the protocol. This is the second part of our attack.

As in the first step of the attack, to get equations over R and not only modulo q, we consider the difference between two rows, this time the difference between rows 1 and 3 (instead of rows 2 and 3). We have the public encodings:

$$\begin{aligned}
\boldsymbol{A}_{1,1} \cdot \boldsymbol{C}_{1,1,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_{1,2} + \boldsymbol{E}_{1,1,\ell} \pmod q \\
\boldsymbol{A}_{1,2} \cdot \hat{\boldsymbol{C}}_{1,3,\ell} &= t_{2,\ell} \cdot t_3 \cdot \boldsymbol{A}_0 + \hat{\boldsymbol{E}}_{1,3,\ell} \pmod q \\
\boldsymbol{A}_{3,1} \cdot \hat{\boldsymbol{C}}_{3,2,\ell} &= t_{2,\ell} \cdot t_3 \cdot \boldsymbol{A}_{3,3} + \hat{\boldsymbol{E}}_{3,2,\ell} \pmod q \\
\boldsymbol{A}_{3,3} \cdot \boldsymbol{C}_{3,3,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_0 + \boldsymbol{E}_{3,3,\ell} \pmod q
\end{aligned}$$

where we let $\hat{\boldsymbol{C}}_{1,3,\ell} := \boldsymbol{C}_{1,2,\ell} \cdot \boldsymbol{C}_{1,3}$, for some small error vector $\hat{\boldsymbol{E}}_{1,3,\ell}$. As previously we can compute over R, restricting ourselves to the first component, where $\hat{\boldsymbol{C}}'_{1,3,j}$ and $\boldsymbol{C}'_{3,3,i}$ are the first columns of $\hat{\boldsymbol{C}}_{1,3,j}$ and $\boldsymbol{C}_{3,3,i}$ respectively:

$$\begin{aligned}
\omega_{ij} &= \boldsymbol{A}_{1,1} \cdot \boldsymbol{C}_{1,1,i} \cdot \hat{\boldsymbol{C}}'_{1,3,j} - \boldsymbol{A}_{3,1} \cdot \hat{\boldsymbol{C}}_{3,2,j} \cdot \boldsymbol{C}'_{3,3,i} \\
&= t_{1,i} \cdot \hat{E}_{1,3,j} + \boldsymbol{E}_{1,1,i} \cdot \hat{\boldsymbol{C}}'_{1,3,j} - t_{2,j} \cdot t_3 \cdot E_{3,3,i} - \hat{\boldsymbol{E}}_{3,2,j} \cdot \boldsymbol{C}'_{3,3,i}\,.
\end{aligned}$$

We can therefore compute over R, using the coefficients α_i from the linear relation (7):

$$\begin{aligned}\Omega_j &= \sum_{i=1}^{2m+3} \alpha_i \cdot \left(\boldsymbol{A}_{1,1} \cdot \boldsymbol{C}_{1,1,i} \cdot \hat{\boldsymbol{C}}'_{1,3,j} - \boldsymbol{A}_{3,1} \cdot \hat{\boldsymbol{C}}_{3,2,j} \cdot \boldsymbol{C}'_{3,3,i} \right) \\ &= \sum_{i=1}^{2m+3} \alpha_i \cdot \left(t_{1,i} \cdot \hat{E}_{1,3,j} + \boldsymbol{E}_{1,1,i} \cdot \hat{\boldsymbol{C}}'_{1,3,j} - t_{2,j} \cdot t_3 \cdot E_{3,3,i} - \hat{\boldsymbol{E}}_{3,2,j} \cdot \boldsymbol{C}'_{3,3,i} \right) .\end{aligned} \tag{13}$$

Using the linear relations (7) and (8), we obtain:

$$\Omega_j = s_1 \cdot \hat{E}_{1,3,j} - t_{2,j} \cdot t_3 \cdot F_{3,3} - \hat{\boldsymbol{E}}_{3,2,j} \cdot \boldsymbol{D}'_{3,3} + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{C}}'_{1,3,j}$$

which gives:

$$\Omega_j = u_j + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{C}}'_{1,3,j} \tag{14}$$

for some small u_j in R. In summary we obtain a large scalar Ω_j because the coefficients α_i in (13) are large, but eventually what makes Ω_j large is only the contribution from $(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i}) \cdot \hat{\boldsymbol{C}}'_{1,3,j}$; namely because of the linear relations (7) and (8) the other terms remain small.

We can now write (14) in vectorial form, where we let $\hat{\boldsymbol{C}}''_{1,3}$ be the square matrix whose columns are the column vectors $\hat{\boldsymbol{C}}'_{1,3,j}$ for $1 \leq j \leq m$; recall that the $\hat{\boldsymbol{C}}'_{1,3,j}$ are the first column vectors of the matrix encodings $\hat{\boldsymbol{C}}_{1,3,j}$. We obtain a row vector $\boldsymbol{\Omega}$ of dimension m, where:

$$\boldsymbol{\Omega} = \boldsymbol{u} + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{C}}''_{1,3} \tag{15}$$

where $\hat{\boldsymbol{C}}''_{1,3}$ is a public square matrix of dimension m.

Now the crucial observation is that because the vector $\boldsymbol{u}$ has small components, we can get an approximation of the vector $\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i}$ by reducing the vector $\boldsymbol{\Omega}$ modulo the matrix $\hat{\boldsymbol{C}}''_{1,3}$, assuming that $\hat{\boldsymbol{C}}''_{1,3}$ is an invertible matrix, which heuristically holds with good probability. This can be done by solving over the fraction field of R the linear system $\boldsymbol{\Omega} = \boldsymbol{y} \cdot \hat{\boldsymbol{C}}''_{1,3}$ and then rounding to R the coefficients of $\boldsymbol{y}$. Heuristically the vector $\boldsymbol{E} = \lfloor \boldsymbol{y} \rceil$ should be a good approximation of $\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i}$; namely letting:

$$\boldsymbol{E}' = \sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} - \boldsymbol{E} \tag{16}$$

we get using $\boldsymbol{y} = \boldsymbol{\Omega} \cdot \hat{\boldsymbol{C}}''^{-1}_{1,3}$:

$$\begin{aligned}\boldsymbol{E}' &= (\boldsymbol{\Omega} - \boldsymbol{u}) \cdot \hat{\boldsymbol{C}}''^{-1}_{1,3} - \boldsymbol{E} \\ &= \boldsymbol{y} - \boldsymbol{E} - \boldsymbol{u} \cdot \hat{\boldsymbol{C}}''^{-1}_{1,3}\end{aligned}$$

and therefore since $\boldsymbol{y} - \boldsymbol{E}$ and $\boldsymbol{u}$ are small, the difference vector $\boldsymbol{E'}$ should be small if the norm of the transpose of the matrix $\hat{\boldsymbol{C}}''^{-1}_{1,3}$ remains small. We know that such a bound holds with probability close to 1 if we model $\hat{\boldsymbol{C}}''_{1,3}$ as a random matrix (*e.g.* Rudelson [Rud08] provides a bound of the form $O(m^{3/2})$), and so we expect $\boldsymbol{E'}$ to be small (compared to q) for randomly generated encodings, since in the GGH15 parameter selection one takes $m = \Theta(\log q)$.

Combining (11) and (16), we get:

$$\boldsymbol{A}_{1,1} \cdot \tilde{\boldsymbol{D}}_{1,1} - \boldsymbol{E} = s_1 \cdot \boldsymbol{A}_{1,2} + \boldsymbol{E}' \pmod q$$

for a small vector $\boldsymbol{E}'$. Note that the previous equation is very similar to the original equation for the private encoding $\boldsymbol{D}_{1,1}$:

$$\boldsymbol{A}_{1,1} \cdot \boldsymbol{D}_{1,1} = s_1 \cdot \boldsymbol{A}_{1,2} + \boldsymbol{F}_{1,1} \pmod q$$

the only difference being the publicly computed correction vector $\boldsymbol{E}$. Therefore the pair $(\tilde{\boldsymbol{D}}_{1,1}, \boldsymbol{E})$ gives us an equivalent of the private encoding $\boldsymbol{D}_{1,1}$, which breaks the protocol. More precisely we can eventually compute from public parameters:

$$\begin{aligned}\left(\boldsymbol{A}_{1,1} \cdot \tilde{\boldsymbol{D}}_{1,1} - \boldsymbol{E}\right) \cdot \boldsymbol{D}_{1,2} \cdot \boldsymbol{D}_{1,3} &= (s_1 \cdot \boldsymbol{A}_{1,2} + \boldsymbol{E}') \cdot \hat{\boldsymbol{D}}_{1,3} \pmod q \\ &= s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0 + \\ &\quad s_1 \cdot \hat{\boldsymbol{F}}_{1,3} + \boldsymbol{E}' \cdot \hat{\boldsymbol{D}}_{1,3} \pmod q .\end{aligned}$$

Since all the error terms are small, this enables to extract the high-order bits of $s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0 \bmod q$, and breaks the protocol.

3.2 Extension to $k \geq 3$ Users

The extension of our attack to $k \geq 3$ users is relatively straightforward and described in the full version of this paper [CLLT15].

4 Cryptanalysis of GGH15 With Safeguards

In [GGH15, Section 5.1] two safeguards for multipartite key agreement based on GGH15 multilinear maps are described:

1. Kilian-style randomization of the encodings, where $\boldsymbol{C}$ is replaced by $\bar{\boldsymbol{C}} := \boldsymbol{R}^{-1} \cdot \boldsymbol{C} \cdot \boldsymbol{R'}$ using the randomizer matrices $\boldsymbol{R}$, $\boldsymbol{R'}$ belonging to two adjacent nodes.
2. Choosing the first encoding matrix in each chain to have large entries.

In the following, we show how to extend our previous attack when those two safeguards are used.

4.1 First Safeguard: Kilian-Style Randomization of the Encodings.

The following safeguard for GGH15 multilinear maps is described in [GGH15], using Kilian-type randomization [Kil88]. For each internal node v in the graph one can choose a random invertible $m \times m$ matrix $\boldsymbol{R}_v$ modulo q, and for the sinks and sources we set $\boldsymbol{R}_v = \boldsymbol{I}$. Then each encoding $\boldsymbol{C}$ relative to path $u \rightsquigarrow v$ is replaced by a masked encoding $\bar{\boldsymbol{C}} := \boldsymbol{R}_u^{-1} \cdot \boldsymbol{C} \cdot \boldsymbol{R}_v$. Concretely, in the GGH15 key-agreement protocol, instead of publishing encodings $\boldsymbol{C}_{i,j}$ with:

$$\boldsymbol{A}_{i,j} \cdot \boldsymbol{C}_{i,j,\ell} = t_{1+(j-i \bmod k),\ell} \cdot \boldsymbol{A}_{i,j+1} + \boldsymbol{E}_{i,j,\ell} \pmod q$$

one would only publish the masked encodings modulo q:

$$\bar{\boldsymbol{C}}_{i,j,\ell} := \boldsymbol{R}_{i,j}^{-1} \cdot \boldsymbol{C}_{i,j,\ell} \cdot \boldsymbol{R}_{i,j+1} \tag{17}$$

with $\boldsymbol{R}_{i,1} = \boldsymbol{R}_{i,k+1} = \boldsymbol{I}$ for all i; the same masking is applied to the encodings $\boldsymbol{D}_{i,j}$. Since the product of encoding on any source-to-sink path remains the same, the same value is eventually extracted. Namely for all i we have:

$$\prod_{j=1}^{k} \bar{\boldsymbol{C}}_{i,j} = \prod_{j=1}^{k} \boldsymbol{C}_{i,j}$$

and therefore exactly the same session-key as before is computed by all users.

4.2 Second Safeguard: First Encodings With Large Entries

The second safeguard described in [GGH15, Section 5.1] consists in choosing the first encodings $\boldsymbol{C}_{i,1}$ in each chain to have large entries modulo q, instead of small entries. Namely the first encoding $\boldsymbol{C}_{i,1}$ does not contribute in the error term when computing the session-key, so it can have large entries.

4.3 Cryptanalysis of GGH15 With Both Safeguards

In this section we show how to extend our attack from Section 3 when both safeguards are used. Note the first step of our attack still applies, since in the first step we are only using product of encodings from source to sink. Namely in Equation (4) exactly the same value $(\boldsymbol{W})_{ij}$ is obtained when using masked encodings. Therefore we can still derive the same linear relation between secret exponents as in (7) and (8).

However the second step of our attack does not apply directly, since our second step requires the knowledge of the matrix $\hat{\boldsymbol{C}}''_{1,3}$ in (15), which is obtained from the first columns of the encodings $\hat{\boldsymbol{C}}_{1,3,j} = \boldsymbol{C}_{1,2,j} \cdot \boldsymbol{C}_{1,3}$. Since these are partial products only, such partial products would be masked by the unknown randomization matrix $\boldsymbol{R}_{1,2}^{-1}$ modulo q, hence the matrix $\hat{\boldsymbol{C}}''_{1,3}$ is unknown.

We can however adapt our second step as follows. For simplicity we keep the same notations as previously, that is we describe our extended attack in term of

the original encodings $\boldsymbol{C}_{i,j,\ell}$, instead of the masked encodings $\bar{\boldsymbol{C}}_{i,j,\ell}$ from (17); in that case we are only allowed to use products of encodings from source to sink. We first start with a slightly different equation from (15):

$$\boldsymbol{\Omega} = \boldsymbol{u} + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{G}}''_{1,3} \tag{18}$$

where $\hat{\boldsymbol{G}}''_{1,3}$ is a matrix whose columns are the first column vectors of $\boldsymbol{D}_{1,2} \cdot \boldsymbol{C}_{1,3,j}$ for $1 \leq j \leq 2m+2$. Note that in (12) the error term that we must estimate to recover the session key is:

$$\boldsymbol{E} = \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{D}}_{1,3} \tag{19}$$

Using a similar approach as in the attack first step, our approach consists in finding a vector $\boldsymbol{x}$ with coefficients in the fraction field $R \otimes_{\mathbb{Z}} \mathbb{Q}$ of R such that:

$$\hat{\boldsymbol{D}}'_{1,3} = \hat{\boldsymbol{G}}''_{1,3} \cdot \boldsymbol{x}$$

where $\hat{\boldsymbol{D}}'_{1,3}$ is the first column vector of $\hat{\boldsymbol{D}}_{1,3}$. Applying the vector $\boldsymbol{x}$ on (18) and rounding in R, we obtain:

$$\lfloor \boldsymbol{\Omega} \cdot \boldsymbol{x} \rceil = \lfloor \boldsymbol{u} \cdot \boldsymbol{x} \rceil + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{D}}'_{1,3}$$

Since the components of $\boldsymbol{u}$ (over R) are small, and moreover the coefficients of $\boldsymbol{x}$ (over $R \otimes_{\mathbb{Z}} \mathbb{Q}$) are heuristically also small, the scalar $\lfloor \boldsymbol{u} \cdot \boldsymbol{x} \rceil$ in R is small compared to q, and therefore we obtain a good estimate of the first component of the error vector $\boldsymbol{E}$ from (19), which enables to recover the first component of the session key and breaks the scheme.[4]

4.4 Detailed Description

First step: linear relations in $\boldsymbol{R}$. The first step of our attack is exactly the same as previously. Namely as mentioned previously the first step of our previous attack still applies, since in the first step we are only using product of encodings from source to sink. More precisely in Equation (4) exactly the same value $(\boldsymbol{W})_{ij}$ is obtained when using masked encodings, and therefore we can still derive the same linear relations as in (7) and (8):

$$\begin{aligned} s_1 &= \sum_{i=1}^{2m+3} \alpha_i \cdot t_{1,i}, \quad \hat{\boldsymbol{F}}_{2,2} = \sum_{i=1}^{2m+3} \alpha_i \cdot \hat{\boldsymbol{E}}_{2,2,i}, \\ F_{3,3} &= \sum_{i=1}^{2m+3} \alpha_i \cdot E_{3,3,i}, \quad \boldsymbol{D}'_{3,3} = \sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{C}'_{3,3,i}. \end{aligned} \tag{20}$$

[4] Other components of the session key can be also obtained analogously.

Note that as opposed to Section 3 we don't know the value of the encodings $\boldsymbol{D}'_{3,3}$ and $\boldsymbol{C}'_{3,3,i}$, since they are masked by the $\boldsymbol{R}_{ij}$ matrices; we only recover the coefficients α_i in R.

Second step: another linear relation. In the second step, our goal is to find a vector $\boldsymbol{x}$ with coefficients in the fraction field $R \otimes_{\mathbb{Z}} \mathbb{Q}$ of R such that:

$$\boldsymbol{D}'_{1,3} = \sum_{i=1}^{2m+2} x_i \cdot \boldsymbol{C}'_{1,3,i}$$

where $\boldsymbol{D}'_{1,3}$ and $\boldsymbol{C}'_{1,3,i}$ are the first column vectors of $\boldsymbol{D}_{1,3}$ and $\boldsymbol{C}_{1,3,i}$ respectively. We show that this can be done using the same approach as in the attack first step.

Namely letting $\hat{\boldsymbol{C}}_{1,2,\ell} := \boldsymbol{C}_{1,1,\ell} \cdot \boldsymbol{C}_{1,2}$ where we let $\boldsymbol{C}_{1,2} := \boldsymbol{C}_{1,2,1}$ corresponding to $t_2 := t_{2,1}$, we obtain:

$$\begin{aligned}
\boldsymbol{A}_{1,1} \cdot \hat{\boldsymbol{C}}_{1,2,\ell} &= t_{1,\ell} \cdot t_2 \cdot \boldsymbol{A}_{1,3} + \hat{\boldsymbol{E}}_{1,2,\ell} \pmod q \\
\boldsymbol{A}_{1,3} \cdot \boldsymbol{C}_{1,3,\ell} &= t_{3,\ell} \cdot \boldsymbol{A}_0 + \boldsymbol{E}_{1,3,\ell} \pmod q
\end{aligned}$$

Similarly letting $\hat{\boldsymbol{C}}_{2,3,\ell} := \boldsymbol{C}_{2,2,\ell} \cdot \boldsymbol{C}_{2,3}$ where $\boldsymbol{C}_{2,3} := \boldsymbol{C}_{2,3,1}$, we get:

$$\begin{aligned}
\boldsymbol{A}_{2,1} \cdot \boldsymbol{C}_{2,1,\ell} &= t_{3,\ell} \cdot \boldsymbol{A}_{2,2} + \boldsymbol{E}_{2,1,\ell} \pmod q \\
\boldsymbol{A}_{2,2} \cdot \hat{\boldsymbol{C}}_{2,3,\ell} &= t_1 \cdot t_{2,\ell} \cdot \boldsymbol{A}_0 + \hat{\boldsymbol{E}}_{2,3,\ell} \pmod q
\end{aligned}$$

We can therefore compute the following matrix elements in R, restricting ourselves as previously to the first component of the vectors:

$$\begin{aligned}
(\boldsymbol{W})_{ij} &= \boldsymbol{A}_{1,1} \cdot \hat{\boldsymbol{C}}_{1,2,i} \cdot \boldsymbol{C}'_{1,3,j} - \boldsymbol{A}_{2,1} \cdot \boldsymbol{C}_{2,1,j} \cdot \hat{\boldsymbol{C}}'_{2,3,i} \\
&= t_{1,i} \cdot t_2 \cdot E_{1,3,j} + \hat{\boldsymbol{E}}_{1,2,i} \cdot \boldsymbol{C}'_{1,3,j} - t_{3,j} \cdot \hat{E}_{2,3,i} - \boldsymbol{E}_{2,1,j} \cdot \hat{\boldsymbol{C}}'_{2,3,i}
\end{aligned}$$

for all $1 \leq i \leq 2m+2$ and $1 \leq j \leq 2m+2$, where $\boldsymbol{C}'_{1,3,j}$ and $\hat{\boldsymbol{C}}'_{2,3,i}$ are the first column vectors of $\boldsymbol{C}_{1,3,j}$ and $\hat{\boldsymbol{C}}_{2,3,i}$ respectively. This gives:

$$(\boldsymbol{W})_{ij} = \begin{bmatrix} t_{1,i}\, t_2 & \hat{\boldsymbol{E}}_{1,2,i} & \hat{E}_{2,3,i} & \hat{\boldsymbol{C}}'_{2,3,i} \end{bmatrix} \cdot \begin{bmatrix} E_{1,3,j} \\ \boldsymbol{C}'_{1,3,j} \\ -t_{3,j} \\ -\boldsymbol{E}_{2,1,j} \end{bmatrix}.$$

Moreover, since the encodings $\boldsymbol{D}_{1,3}$ and $\boldsymbol{D}_{2,1}$ corresponding to s_3 on rows 1 and 2 are public, we can additionally compute the corresponding vector:

$$\begin{aligned}
(\boldsymbol{V})_i &= \boldsymbol{A}_{1,1} \cdot \hat{\boldsymbol{C}}_{1,2,i} \cdot \boldsymbol{D}'_{1,3} - \boldsymbol{A}_{2,1} \cdot \boldsymbol{D}_{2,1} \cdot \hat{\boldsymbol{C}}'_{2,3,i} \\
&= \begin{bmatrix} t_{1,i}\, t_2 & \hat{\boldsymbol{E}}_{1,2,i} & \hat{E}_{2,3,i} & \hat{\boldsymbol{C}}'_{2,3,i} \end{bmatrix} \cdot \begin{bmatrix} F_{1,3} \\ \boldsymbol{D}'_{1,3} \\ -s_3 \\ -\boldsymbol{F}_{2,1} \end{bmatrix}.
\end{aligned}$$

where $\boldsymbol{D}'_{1,3}$ is the first column vector of $\boldsymbol{D}_{1,3}$. Therefore assuming that the matrix $\boldsymbol{W}$ is invertible, we can find $\boldsymbol{x}$ in $R \otimes_{\mathbb{Z}} \mathbb{Q}$ such that:

$$\boldsymbol{W} \cdot \boldsymbol{x} = \boldsymbol{V}$$

which gives as required:

$$\boldsymbol{D}'_{1,3} = \sum_{i=1}^{2m+2} x_i \cdot \boldsymbol{C}'_{1,3,i} \tag{21}$$

Note that the only difference with the linear relations from Step 1 is that we don't require the x_i's to be in R, only in the fraction field $R \otimes_{\mathbb{Z}} \mathbb{Q}$ of R; this implies that heuristically such coefficients should remain small in absolute value.

Third step: estimating the error term. In the third step our goal is to estimate the error term when computing the session-key, as in the second step of the basic attack. We first start with a slightly different equation from (15):

$$\boldsymbol{\Omega} = \boldsymbol{u} + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{G}}''_{1,3} \tag{22}$$

where $\hat{\boldsymbol{G}}''_{1,3}$ is a matrix whose columns are the first column vectors of $\boldsymbol{D}_{1,2} \cdot \boldsymbol{C}_{1,3,j}$ for $1 \leq j \leq 2m+2$. Therefore the only difference with (15) is that we use the matrix $\hat{\boldsymbol{G}}''_{1,3}$ instead of $\hat{\boldsymbol{C}}''_{1,3}$.

To obtain (22) we proceed as follows. Instead of letting $\hat{\boldsymbol{C}}_{1,3,\ell} = \boldsymbol{C}_{1,2,\ell} \cdot \boldsymbol{C}_{1,3}$ as in the basic attack, we let $\hat{\boldsymbol{C}}_{1,3,\ell} = \boldsymbol{D}_{1,2} \cdot \boldsymbol{C}_{1,3,\ell}$. Similarly we let $\hat{\boldsymbol{C}}_{3,2,\ell} := \boldsymbol{D}_{3,1} \cdot \boldsymbol{C}_{3,2,\ell}$. This is possible because on rows 1 and 3 the encodings $\boldsymbol{D}_{1,2}$ and $\boldsymbol{D}_{3,1}$ corresponding to s_2 are public. We obtain:

$$\begin{aligned}
\boldsymbol{A}_{1,1} \cdot \boldsymbol{C}_{1,1,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_{1,2} + \boldsymbol{E}_{1,1,\ell} \pmod q \\
\boldsymbol{A}_{1,2} \cdot \hat{\boldsymbol{C}}_{1,3,\ell} &= s_2 \cdot t_{3,\ell} \cdot \boldsymbol{A}_0 + \hat{\boldsymbol{E}}_{1,3,\ell} \pmod q \\
\boldsymbol{A}_{3,1} \cdot \hat{\boldsymbol{C}}_{3,2,\ell} &= s_2 \cdot t_{3,\ell} \cdot \boldsymbol{A}_{3,3} + \hat{\boldsymbol{E}}_{3,2,\ell} \pmod q \\
\boldsymbol{A}_{3,3} \cdot \boldsymbol{C}_{3,3,\ell} &= t_{1,\ell} \cdot \boldsymbol{A}_0 + \boldsymbol{E}_{3,3,\ell} \pmod q
\end{aligned}$$

As previously we can compute over R, restricting ourselves to the first component, where $\hat{\boldsymbol{C}}'_{1,3,j}$ and $\boldsymbol{C}'_{3,3,i}$ are the first columns of $\hat{\boldsymbol{C}}_{1,3,j}$ and $\boldsymbol{C}_{3,3,i}$ respectively:

$$\begin{aligned}
\omega_{ij} &= \boldsymbol{A}_{1,1} \cdot \boldsymbol{C}_{1,1,i} \cdot \hat{\boldsymbol{C}}'_{1,3,j} - \boldsymbol{A}_{3,1} \cdot \hat{\boldsymbol{C}}_{3,2,j} \cdot \boldsymbol{C}'_{3,3,i} \\
&= t_{1,i} \cdot \hat{E}_{1,3,j} + \boldsymbol{E}_{1,1,i} \cdot \hat{\boldsymbol{C}}'_{1,3,j} - s_2 \cdot t_{3,j} \cdot E_{3,3,i} - \hat{\boldsymbol{E}}_{3,2,j} \cdot \boldsymbol{C}'_{3,3,i} \,.
\end{aligned}$$

We can therefore compute over R, using the coefficients α_i from the linear relations (20):

$$\begin{aligned}
\Omega_j &= \sum_{i=1}^{2m+3} \alpha_i \cdot \omega_{ij} \\
&= \sum_{i=1}^{2m+3} \alpha_i \cdot \left(t_{1,i} \cdot \hat{E}_{1,3,j} + \boldsymbol{E}_{1,1,i} \cdot \hat{\boldsymbol{C}}'_{1,3,j} - s_2 \cdot t_{3,j} \cdot E_{3,3,i} - \hat{\boldsymbol{E}}_{3,2,j} \cdot \boldsymbol{C}'_{3,3,i} \right)
\end{aligned}$$

Using the linear relations in (20), we obtain:

$$\Omega_j = s_1 \cdot \hat{E}_{1,3,j} - s_2 \cdot t_{3,j} \cdot F_{3,3} - \hat{\boldsymbol{E}}_{3,2,j} \cdot \boldsymbol{D}'_{3,3} + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{C}}'_{1,3,j}$$

where $\boldsymbol{D}'_{3,3}$ is the first column vector of $\boldsymbol{D}_{3,3}$. This gives:

$$\Omega_j = u_j + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{C}}'_{1,3,j}$$

for some small u_j in R. Since we have let $\hat{\boldsymbol{C}}_{1,3,j} = \boldsymbol{D}_{1,2} \cdot \boldsymbol{C}_{1,3,j}$ for $1 \leq j \leq 2m+2$, in vectorial form we obtain (22) as required, where $\hat{\boldsymbol{G}}''_{1,3}$ is the matrix whose columns are the first column vectors of $\boldsymbol{D}_{1,2} \cdot \boldsymbol{C}_{1,3,j}$ for $1 \leq j \leq 2m+2$.

Recall that in (12) the error term that we must estimate to recover the session key is:

$$\boldsymbol{E} = \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{D}}_{1,3} \tag{23}$$

where $\hat{\boldsymbol{D}}_{1,3} = \boldsymbol{D}_{1,2} \cdot \boldsymbol{D}_{1,3}$. In the following we will only estimate the first component, so we let $\hat{\boldsymbol{D}}'_{1,3} = \boldsymbol{D}_{1,2} \cdot \boldsymbol{D}'_{1,3}$, where $\hat{\boldsymbol{D}}'_{1,3}$ and $\boldsymbol{D}'_{1,3}$ are the first column vectors of $\hat{\boldsymbol{D}}_{1,3}$ and $\boldsymbol{D}_{1,3}$ respectively.

We now use the vector $\boldsymbol{x}$ computed in the second step. In matrix notation, Equation (21) gives:

$$\boldsymbol{D}'_{1,3} = \boldsymbol{C}''_{1,3} \cdot \boldsymbol{x}$$

where $\boldsymbol{C}''_{1,3}$ is the matrix whose columns are the first column vectors of $\boldsymbol{C}_{1,3,i}$ for $1 \leq i \leq 2m+2$. Using $\hat{\boldsymbol{G}}''_{1,3} = \boldsymbol{D}_{1,2} \cdot \boldsymbol{C}''_{1,3}$, this gives:

$$\hat{\boldsymbol{D}}'_{1,3} = \boldsymbol{D}_{1,2} \cdot \boldsymbol{D}'_{1,3} = \boldsymbol{D}_{1,2} \cdot \boldsymbol{C}''_{1,3} \cdot \boldsymbol{x} = \boldsymbol{G}''_{1,3} \cdot \boldsymbol{x}$$

where $\hat{\boldsymbol{D}}'_{1,3}$ is the first column vector of $\hat{\boldsymbol{D}}_{1,3}$. Applying the vector $\boldsymbol{x}$ on (22), we therefore get:

$$\boldsymbol{\Omega} \cdot \boldsymbol{x} = \boldsymbol{u} \cdot \boldsymbol{x} + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{D}}'_{1,3}$$

We claim that this provides a good estimate of the first component of the error vector $\boldsymbol{E}$ from (23). Recall that the components of $\boldsymbol{x}$ are in $R \otimes_{\mathbb{Z}} \mathbb{Q}$, so by rounding to the nearest integer we can get the following value in R:

$$E' = \lfloor \boldsymbol{\Omega} \cdot \boldsymbol{x} \rceil = \lfloor \boldsymbol{u} \cdot \boldsymbol{x} \rceil + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{D}}'_{1,3} \tag{24}$$

Since the components of $\boldsymbol{u}$ (over R) are small, and moreover the coefficients of $\boldsymbol{x}$ (over $R \otimes_{\mathbb{Z}} \mathbb{Q}$) are also small (heuristically), the scalar $\lfloor \boldsymbol{u} \cdot \boldsymbol{x} \rceil$ in R is small.

Finally, letting as previously:

$$\tilde{\boldsymbol{D}}_{1,1} = \sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{C}_{1,1,i}\,,$$

we obtain:

$$\boldsymbol{A}_{1,1} \cdot \tilde{\boldsymbol{D}}_{1,1} = s_1 \cdot \boldsymbol{A}_{1,2} + \sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \pmod q\,.$$

which gives as previously:

$$\boldsymbol{A}_{1,1} \cdot \tilde{\boldsymbol{D}}_{1,1} \cdot \hat{\boldsymbol{D}}'_{1,3} = s_1 \cdot s_2 \cdot s_3 \cdot A_0 + s_1 \cdot \hat{F}_{1,3} + \left(\sum_{i=1}^{2m+3} \alpha_i \cdot \boldsymbol{E}_{1,1,i} \right) \cdot \hat{\boldsymbol{D}}'_{1,3} \pmod q$$

Therefore combining with (24) we can compute from public parameters:

$$\boldsymbol{A}_{1,1} \cdot \tilde{\boldsymbol{D}}_{1,1} \cdot \hat{\boldsymbol{D}}'_{1,3} - E' = s_1 \cdot s_2 \cdot s_3 \cdot A_0 + s_1 \cdot \hat{F}'_{1,3} - \lfloor \boldsymbol{u} \cdot \boldsymbol{x} \rceil \pmod q$$

Since the terms $s_1 \cdot \hat{F}'_{1,3}$ and $\lfloor \boldsymbol{u} \cdot \boldsymbol{x} \rceil$ are small, this reveals the first component of the secret vector $s_1 \cdot s_2 \cdot s_3 \cdot \boldsymbol{A}_0$, which breaks the scheme.

Acknowledgments. This work has been supported in part by the European Union's H2020 Programme under grant agreement number ICT-644209.

References

[BGH+15] Zvika Brakerski, Craig Gentry, Shai Halevi, Tancrède Lepoint, Amit Sahai, and Mehdi Tibouchi. Cryptanalysis of the quadratic zero-testing of GGH. Cryptology ePrint Archive, Report 2015/845, 2015. Available at `https://eprint.iacr.org/2015/845`.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002.

[CFL+16] Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new CLT multilinear map over the integers. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 509–536. Springer, 2016.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 247–266. Springer, 2015.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, 2015.

[CLLT15] Jean-Sebastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. Cryptology ePrint Archive, Report 2015/1037, 2015. `http://eprint.iacr.org/`. Full version of this paper.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, 2013.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 267–286. Springer, 2015.

[Dev16] The Sage Developers. *Sage Mathematics Software (Version 7.0)*, 2016. `http://www.sagemath.org`.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In Omer Reingold, editor, *FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527. Springer, 2015.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, 2013.

[Hal15] Shai Halevi. Graded encoding, variations on a scheme. Cryptology ePrint Archive, Report 2015/866, 2015. Available at `https://eprint.iacr.org/2015/866`.

[HJ16] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 537–565. Springer, 2016.

[Jou00] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In Wieb Bosma, editor, *ANTS-IV*, volume 1838 of *LNCS*, pages 385–394. Springer, 2000.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *STOC 1988*, pages 20–31. ACM, 1988.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, 2012.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. Cryptology ePrint Archive, Report 2016/147, 2016. Available at `https://eprint.iacr.org/2016/147`.

[PS15] Alice Pellet-Mary and Damien Stehlé. Cryptanalysis of Gu's ideal multilinear map. Cryptology ePrint Archive, Report 2015/759, 2015. Available at `https://eprint.iacr.org/2015/759`.

[Rud08] Mark Rudelson. Invertibility of random matrices: Norm of the inverse. *Annals of Mathematics*, 168(2):575–600, 2008.