Towards a Classification of Non-interactive Computational Assumptions in Cyclic Groups*

Essam Ghadafi †[1] and Jens Groth[2]

[1] University of the West of England, Bristol, UK
Essam.Ghadafi@gmail.com
[2] University College London, London, UK
j.groth@ucl.ac.uk

Abstract. We study non-interactive computational intractability assumptions in prime-order cyclic groups. We focus on the broad class of computational assumptions, which we call target assumptions, where the adversary's goal is to compute a concrete group element and investigate the structure of this class.

Our analysis identifies two families of intractability assumptions, the q-Generalized Diffie-Hellman Exponent assumptions and the q-Simple Fractional assumptions that imply all other target assumptions. These two assumptions therefore serve as Uber assumptions that can underpin all the target assumptions where the adversary has to compute specific group elements. We also study the internal hierarchy among members of these two assumption families. We provide heuristic evidence that both families are necessary to cover the full class of target assumptions, and we show that the lowest level in the q-GDHE hierarchy (the 1-GDHE assumption) is equivalent to the computational Diffie-Hellman assumption.

We generalize our results to the bilinear group setting. For the base groups our results translate nicely and a similar structure of non-interactive computational assumptions emerges. We also identify Uber assumptions in the target group but this requires replacing the q-GDHE assumption with a more complicated assumption, which we call the Bilinar Gap Assumption.

Our analysis can assist both cryptanalysts and cryptographers. For cryptanalysts, we propose the q-GDHE and the q-SDH assumptions are the most natural and important targets for cryptanalysis in prime-order groups. For cryptographers, we believe our classification can aid the choice of assumptions underpinning cryptographic schemes and be used as a guide to minimize the overall attack surface that different assumptions expose.

Keywords. Non-Interactive Assumptions, Computational Assumptions, Target Assumptions, Prime-Order Groups, Bilinear Groups.

*The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307937 and EPSRC grant EP/J009520/1.

†The work was done while at University College London.

1 Introduction

Prime-order groups are widely used in cryptography because their clean mathematical structure enables the construction of many interesting schemes. However, cryptographers rely on an ever increasing number of intractability assumptions to prove their cryptographic schemes are secure. Especially after the rise of pairing-based cryptography, we have witnessed a proliferation of intractability assumptions. While some of those intractability assumptions, e.g. the Discrete Logarithm or the Computational Diffie-Hellman assumptions, are well-studied, and considered by now "standard", the rest of the assumption wilderness has received less attention.

This is unfortunate both for cryptographers designing protocols and cryptanalysts studying the security of the underpinning assumptions. Cryptographers designing protocols are often faced with a trade-off between performance and security, and it would therefore be helpful for them to know how their chosen intractability assumptions compare to other assumptions. Moreover, when they are designing a suite of protocols, it would be useful to know whether the different assumptions they use increase the attack surface or whether the assumptions are related. Cryptanalysts facing the wilderness of assumptions are also faced with a problem: which assumptions should they focus their attention on? One option is to go for the most devastating attack and try to break the discrete logarithm assumption, but the disadvantage is that this is also the hardest assumption to attack and hence the one where the cryptanalyst is least likely to succeed. The other option is to try to attack an easier assumption but the question then is which assumption is the most promising target?

Our research vision is that a possible path out of the wilderness is to identify Uber assumptions that imply all the assumptions we use. An extreme Uber assumption would be that anything that cannot trivially be broken by generic group operations is secure, however, we already know that this is a too extreme position since there are schemes that are secure against generic attacks but insecure for any concrete instantiation of the groups [Den02]. Instead of trying to capture all of the generic group model, we therefore ask for a few concrete and plausible Uber assumptions that capture the most important part of the assumption landscape. Such a characterization of the assumption wilderness would help both cryptographers and crypanalysts. The cryptographic designer may choose assumptions that fall under the umbrella of a few of the Uber assumptions to minimize the attack surface on her schemes. The cryptanalyst can use the Uber assumptions as important yet potentially easy targets.

Our contribution. We focus on efficiently falsifiable computational assumptions in prime-order groups. More precisely, we define a target assumption as an assumption where the adversary has a specific target element that she is trying to compute. A well-known target assumption is the Computational Diffie-Hellman assumption over a cyclic group $\mathbb{G}_p$ of prime order p, which states that given $\left(G, G^a, G^b\right) \in \mathbb{G}_p^3$, it is hard to compute the target $G^{ab} \in \mathbb{G}_p$. We define target assumptions quite broadly and also include assumptions where the adversary takes part in specifying the target to be computed. In the q-SDH assumption

[BB08] for instance, the adversary is given $G, G^x, \ldots, G^{x^q}$ and has to output $(c, G^{\frac{1}{x+c}}) \in \mathbb{Z}_p \setminus \{-x\} \times \mathbb{G}_p$. Here c selected by the adversary is part of the specification of the target $G^{\frac{1}{x+c}}$. In other words, our work includes both assumptions in which the target element to be computed is either uniquely determined a priori by the instance, or a posteriori by the adversary. We note that the case of multiple target elements is also covered by our framework as long as all the target elements are uniquely determined. This is because a tuple of elements is hard to compute if any of its single elements is hard to compute.

Our main contribution is to identify two classes of assumptions that imply the security of all target assumptions. The first class of assumptions is the Generalized Diffie-Hellman Exponent (q-GDHE) assumption [BBG05] that says given $(G, G^x, \ldots, G^{x^{q-1}}, G^{x^{q+1}}, \ldots, G^{x^{2q}}) \in \mathbb{G}_p^{2q}$, it is hard to compute $G^{x^q} \in \mathbb{G}_p$. The second class of assumptions, which is a straightforward generalization of the q-SDH assumption, we call the simple fractional (q-SFrac) assumption and it states that given $(G, G^x, \ldots, G^{x^q}) \in \mathbb{G}_p^{q+1}$, it is hard to output polynomials $r(X)$ and $s(X)$ together with the target $G^{\frac{r(x)}{s(x)}} \in \mathbb{G}_p$, where $0 \leq \deg(r(X)) < \deg(s(X)) \leq q$. The assumption that the q-GDHE and q-SFrac assumptions both hold when q is polynomial in the security parameter can therefore be seen as an Uber assumption for the entire class of target assumptions.

Having identified the q-GDHE and q-SFrac assumptions as being central for the security of target assumptions in general, we investigate their internal structure. We first show that q-SFrac is unlikely to be able to serve as an Uber assumption for all target assumptions on its own. More precisely, we show that for a generic group adversary the 2-GDHE assumption is not implied by q-SFrac assumptions. Second, we show that the q-GDHE assumptions appear to be strictly increasing as q grows, i.e., if the $(q+1)$-GDHE holds, then so does q-GDHE, but for a generic group adversary the $(q+1)$-GDHE may be false even though q-GDHE holds. We also analyze the particular case where $q = 1$ and prove that the 1-GDHE assumption is equivalent to the computational Diffie-Hellman (CDH) assumption.

Based on these results we view the q-GDHE and q-SFrac assumptions as a bulwark. Whatever type of target assumptions a cryptographer bases her schemes on, they are secure as long as neither the q-GDHE nor the q-SFrac assumptions are broken.[3] Since the attacker has less leeway in the q-GDHE assumptions, the cryptographer may choose to rely exclusively on target assumpitons that are implied by the q-GDHE assumptions, and we therefore identify a large class of target assumptions that only need the q-GDHE assumptions to hold.

[3] We note the caveat that our reductions are not tight. So it may be for concrete parameters a target assumption can be broken even if q-GDHE and q-SFrac hold for the same parameters. For all reductions, the concrete loss is stated explicitly in our proofs such that a cryptographer can work out a choice of parameters that yields security from the q-GDHE and q-SFrac assumptions.

We also have advice for the cryptanalyst. We believe it is better to focus on canary in a coal mine assumptions than the discrete logarithm problem that has received the most attention so far. Based on our work the easiest target assumptions to attack in single prime-order groups are the q-GDHE assumptions and the q-SFrac assumptions. The class of q-SFrac assumptions allows more room for the adversary to maneuver in the choice of polynomials $r(X)$ and $s(X)$ and appears less structured than the q-GDHE assumptions. Pragmatically, we note that within the q-SFrac assumptions it is almost exclusively the q-SDH assumptions that are used. We therefore suggest the q-GDHE assumptions and the q-SDH assumptions to be the most suitable targets for cryptanalytic research.

Switching from single prime-order groups $\mathbb{G}_p$ to groups with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, a similar structure emerges. For target assumptions in the base groups $\mathbb{G}_1$ and $\mathbb{G}_2$, we can again identify assumptions similar to the q-GDHE and q-SFrac assumptions that act as a joint Uber assumption. In the target group $\mathbb{G}_T$, a somewhat more complicated picture emerges under the influence of the pairing of source group elements. However, we can replace the q-GDHE assumption with an assumption we call the q-Bilinear Gap (q-BGap) assumption, and get that this assumption together with a natural generalization of the q-SFrac assumption to bilinear groups, which we name the q-BSFrac assumption, jointly act as an Uber assumption for all target assumptions in $\mathbb{G}_T$.

A natural question is whether our analysis extends to other assumptions as well, for instance ”flexible” assumptions such as the q-HSDH assumption [BW07], where the adversary can choose secret exponents and therefore the target elements are no longer uniquely determined. Usually assumptions in the literature involve group elements that have discrete logarithms defined by polynomials in secret values in $\mathbb{Z}_p$ chosen by a challenger and/or public values in $\mathbb{Z}_p$. This gives rise to several classes of assumptions:

1. Non-interactive assumptions where the advesary's goal is to compute group elements defined by secret variables chosen by the challenger.
2. Non-interactive assumptions where the adversary's goal is to compute group elements defined by secret variables chosen by the challenger and public values chosen by the adversary.
3. Non-interactive assumptions where the adversary's goal is to compute group elements defined by secret variables chosen by the challenger, and public and secret values chosen by the adversary.
4. Interactive assumptions, where the challenger and adversary interact.

Target assumptions include all assumptions in classes 1 and 2. However, class 3 includes assumptions which are not falsifiable, e.g. knowledge-of-exponent assumptions [BP04]. Since q-GDHE is in class 1 and q-SFrac is in class 2, both of which only have falsifiable assumptions, we cannot expect them to capture non-falsifiable assumptions in class 3. We leave it as an interesting open problem which structure, if any, can be found in classes 3 and 4, and we hope our work will inspire research on this question.

We stress that our aim has been to find concrete and precise reductions and prove separations among the different classes of assumptions which encompass a significant portion of existing assumptions in the literature. Thus, our approach is different from previous works such as [BBG05,Boy08,EHK$^+$13,MRV16] which aimed at defining algebraic frameworks or templates in generic groups to capture some families of assumptions. The closest work to ours is Abdalla et al. [ABP15], which provided an Uber assumption for decisional assumptions in cyclic groups (without a bilinear map, which invalidates many decisional assumptions). Other works, discussed below, have mostly dealt with very specific relations among assumptions, e.g., the equivalence of CDH and square-CDH as opposed to the general approach we take.

Related work. The rapid development of cryptographic schemes has been accompanied by an increase in the number and complexity of intractability assumptions. Cryptographers have been in pursuit to study the relationship among existing assumptions either by means of providing templates which encompass assumptions in the same family, e.g. [Kil01,BCP02,BLMW07], or by studying direct implications or lack thereof among the different assumptions, e.g. [Boe88,Mau94,MW96,BDZ03,JR13].

A particular class of assumptions which has received little attention are fractional assumptions. Those include, for example, the q-SDH assumption [BB08] and many of its variants, e.g. the modified q-SDH assumption [BW07], and the hidden q-SDH (q-HSDH) assumption [BW07]. As posed by, e.g. [KM07], a subtle question that arises is how such class of assumptions, e.g. the q-SDH assumption, relate to other existing (discrete-logarithm related) computational and decisional intractability assumptions. For instance, while it is clear that q-SDH assumption implies the computational Diffie-Hellman assumption, it is still unclear whether the q-SDH assumption is implied by the decisional Diffie-Hellman assumption. Another intriguing open question is if there is a hierarchy between fractional assumptions or the class of assumptions is inherently unstructured.

Sadeghi and Steiner [SS01] introduced a new parameter when defining discrete-logarithm related assumptions they termed *granularity*. They argued that the choice of such a parameter can influence the security of schemes based on such assumptions and showed that such a parameter influences the implications between assumptions.

Naor [Nao03] classified assumptions based on the complexity of falsifying them. Informally speaking, an assumption is *falsifiable* if it is possible to efficiently decide whether an adversary against the assumption has successfully broken it. Very recently, Goldwasser and Kalai [GK16] provided another classification of intractability assumptions based on their complexity. They argued that classifications based merely on falsifiability of the assumptions might be too inclusive since they do not exclude assumptions which are too dependent on the underlying cryptographic scheme/construct they support.

Boneh et al. [BBG05] defined a framework for proving that decisional and computational assumptions are secure in the generic group model [Sho97,Mau05] and formalized an Uber assumption saying that generic group security implies real security for these assumptions. Boyen [Boy08] later highlighted extensions to the framework and informally suggested how some other families which were not encompassed by the original Uber assumption in [BBG05] can be captured. In essence, the Uber assumption encompasses computational and decisional (discrete-logarithm related) assumptions with a fixed unique challenge. Unfortunately, the framework excludes some families of assumptions, in particular, those where the polynomial(s) used for the challenge are chosen adaptively by the adversary after seeing the problem instance. Examples of such assumptions include the q-SDH [BB08], the modified q-SDH [BW07], and the q-HSDH [BW07] assumptions. The statement of the assumption of the aforementioned yield exponentially many (mutually irreducible) valid solutions rather than a unique one. Another distinction, from the Uber assumption is that the exponent required for the solution involves a fraction of polynomials rather than a polynomial. Joux and Rojat [JR13] proved relationships between some instances of the Uber assumption [BBG05]. In particular, they proved implications between some variants of the computational Diffie-Hellman assumption.

Cheon [Che06] observed that if some relation between the group prime order p and the assumption parameter q, in particular, if either $q \mid p-1$ or $q \mid p+1$, holds, the computational complexity of the q-SDH assumption (and related assumptions) is reduced by a factor of $O(\sqrt{q})$ from that of the discrete logarithm problem and hence for such groups one must increase the key sizes accordingly in order to maintain the same level of security. Jao and Yoshida [JY09] proved equivalence between the unforgeability of the Boneh-Boyen signature scheme [BB08] and the q-SDH assumption on which is it based.

Chase and Meiklejohn [CM14] showed that in composite-order groups some of the so-called q-type assumptions can be reduced to the standard subgroup hiding assumption. More recently, Chase et al. [CMM16] extended their framework to cover more assumptions and get tighter reductions.

Barthe et al. [BFF$^+$14] analyzed hardness of intractability assumptions in the generic group model by reducing them to solving problems related to polynomial algebra. They also provided an automated tool that verifies the hardness of a subclass of families of assumptions in the generic group model. More recently, Ambrona et al. [ABS16] improved upon the results of [BFF$^+$14] by allowing unlimited oracle queries.

Kiltz [Kil01] introduced the Poly-Diffie-Hellman assumption as a generalization of the computational Diffie-Hellman assumption. Escala et al. [EHK$^+$13] proposed an algebraic framework as a generalization of Diffie-Hellman like decisional assumptions. Analogously to [EHK$^+$13], Morillo et al. [MRV16] extended the framework to computational assumptions.

Paper Organization. Our research contribution is organized into three parts. In Sec. 3, we define our framework for target assumptions in cyclic groups, and progressively seek reductions to simpler assumptions. In Sec. 4, we study the internal structure and the relationships among the families of assumptions we identify as Uber assumptions for our framework. In Sec. 5, we provide a generalization of our framework in the bilinear setting.

2 Preliminaries

Notation. We say a function f is *negligible* when $f(\kappa) = \kappa^{-\omega(1)}$ and it is *overwhelming* when $f(\kappa) = 1 - \kappa^{\omega(1)}$. We write $f(\kappa) \approx 0$, when $f(\kappa)$ is a negligible function. We will use κ to denote a security parameter, with the intuition that as κ grows we expect stronger security.

We write $y = A(x; r)$ when algorithm A on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r at random and setting $y = A(x; r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from the set S. We will assume it is possible to sample uniformly at random from sets such as $\mathbb{Z}_p$. We write PPT and DPT for probabilistic and deterministic polynomial time respectively.

Quadratic Residuosity. For an odd prime p and an integer $a \neq 0$, we say a is a *quadratic residue* modulo p if there exists a number x such that $x^2 \equiv a \pmod{p}$ and we say a is *quadratic non-residue* modulo p otherwise. We denote the set of quadratic residues modulo p by $\mathbb{QR}(p)$ and the set of quadratic non-residues modulo p by $\mathbb{QNR}(p)$. By Euler's theorem, we have that $|\mathbb{QR}(p)| = |\mathbb{QNR}(p)| = \frac{p-1}{2}$.

2.1 Non-interactive assumptions

In a non-interactive computational problem the adversary is given a problem instance and tries to find a solution. We say the adversary breaks the problem if it has non-negligible chance of finding a valid solution and we say the problem is hard if any PPT adversary has negligible chance of breaking it. We focus on non-interactive problems that are efficiently falsifiable, i.e., given the instance there is an efficient verification algorithm that decides whether the adversary won.

Definition 1 (Non-Interactive Computational Assumption). *A Non-interactive Computational Assumption consists of an instance generator and a verifier* $(\mathcal{I}, \mathcal{V})$.

$(pub, priv) \leftarrow \mathcal{I}(1^\kappa)$**:** *$\mathcal{I}$ is a PPT algorithm that takes as input a security parameter 1^κ, and outputs a pair of public/private information $(pub, priv)$.*

$b \leftarrow \mathcal{V}(pub, priv, sol)$**:** *$\mathcal{V}$ is a DPT algorithm that receives as input $(pub, priv)$ and a purported solution sol and returns 1 if it considers the answer correct and 0 otherwise.*

The assumption is that for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}_\mathcal{A}$ is negligible (in κ), where

$$\mathsf{Adv}_\mathcal{A}(\kappa) := \Pr\left[(pub, priv) \leftarrow \mathcal{I}(1^\kappa); sol \leftarrow \mathcal{A}(pub) : \mathcal{V}(pub, priv, sol) = 1\right].$$

Relations Among Assumptions. For two non-interactive assumptions A and B, we will use the notation $\mathsf{A}{\Rightarrow}\mathsf{B}$ when assumption B is implied (in a black-box manner) by assumption A, i.e. given an efficient algorithm $\mathcal{B}$ for breaking assumption B, one can construct an efficient algorithm $\mathcal{A}$ that uses $\mathcal{B}$ as an oracle and breaks assumption A. The absence of implication will be denoted by $\mathsf{A}{\not\Rightarrow}\mathsf{B}$.

2.2 Non-Interactive Assumptions over Cyclic Groups

We study non-interactive assumptions over prime-order cyclic groups. These assumptions are defined relative to a group generator $\mathcal{G}$.

Definition 2 (Group Generator). *A group generator is a PPT algorithm $\mathcal{G}$, which on input a security parameter κ (given in unary) outputs group parameters $(\mathbb{G}_p, G)$, where*

- *$\mathbb{G}_p$ is cyclic group of known prime order p with bitlength $|p| = \Theta(\kappa)$.*
- *$\mathbb{G}_p$ has a unique canonical representation of group elements, and polynomial time algorithms for carrying out group operations and deciding membership.*
- *G is a uniformly random generator of the group.*

In non-interactive assumptions over prime-order groups the instance generator runs the group setup $(\mathbb{G}_p, G) \leftarrow \mathcal{G}(1^\kappa)$ and includes $\mathbb{G}_p$ in *pub*. Sadeghi and Steiner [SS01] distinguish between group setups with low, medium and high granularity. In the low granularity setting the non-interactive assumption must hold with respect to random choices of $\mathbb{G}_p$ and G, in the medium granularity setting it must hold for all choices of $\mathbb{G}_p$ and a random G, and in the high granularity setting it must hold for all choices of $\mathbb{G}_p$ and G. Our definitions always assume G is chosen uniformly at random, so depending on $\mathcal{G}$ we are always working in the low or medium granularity setting.

We will use $[x]$ to denote the group element that has discrete logarithm x with respect to the group generator G. In this notation the group generator G is $[1]$ and the neutral element is $[0]$. We will find it convenient to use additive notation for the group operation, so we have $[x]+[y] = [x+y]$. Observe that given $\alpha \in \mathbb{Z}_p$ and $[x] \in \mathbb{G}_p$ it is easy to compute $[\alpha x]$ using the group operations. For a vector $\boldsymbol{x} \in \mathbb{Z}_p^n$ we use $[x_1, \ldots, x_n]$ as a shorthand for the tuple $([x_1], \ldots, [x_n])$.

There are many examples of non-interactive assumptions defined relative to a group generator $\mathcal{G}$. We list in Fig. 1 some of the existing non-interactive computational assumptions.

Generic Group Model. Obviously, if an assumption can be broken using the generic group operations, then it is false. The absence of a generic group operations attack on an assumption does not necessarily mean the assumption holds [Den02,JS12] but is a necessary precondition for the assumption to be plausible.

We formalize the generic group model [Nec94,Sho97] where an adversary can only use generic group operations as follows. Given $\mathbb{G}_p$ we let $[\cdot]$ be a random bijection $\mathbb{Z}_p \rightarrow \mathbb{G}_p$. We give oracle access to the addition operation, i.e., $\mathcal{O}([x], [y])$ returns $[x+y]$. We say an assumption holds in the generic group model if an adversary with access to such an addition oracle has negligible chance of breaking the assumption. Note that the adversary gets $\mathbb{G}_p$ as input and hence is capable of deciding group membership. Also, given an arbitrary $[x]$ it can compute $[0] = [px]$ using the addition oracle. A generic adversary might be able to sample a random group element from $\mathbb{G}_p$, but since they are just random encodings

```
$\mathcal{I}_{\mathrm{CDH}}(1^\kappa)$
    $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$; $x, y \leftarrow \mathbb{Z}_p$
    Return $(pub = (\mathbb{G}_p, [1], [x], [y]), priv = (x, y))$

$\mathcal{V}_{\mathrm{CDH}}(pub, priv = (x, y), sol = [z])$
    If $[z] = [xy]$ return 1
    Else return 0
```

Computational Diffie-Hellman (CDH) Assumption

```
$\mathcal{I}_{\mathrm{SCDH}}(1^\kappa)$
    $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$; $x \leftarrow \mathbb{Z}_p$
    Return $(pub = (\mathbb{G}_p, [1], [x]), priv = x)$

$\mathcal{V}_{\mathrm{SCDH}}(pub, priv = x, sol = [z])$
    If $[z] = \left[x^2\right]$ return 1
    Else return 0
```

Square Computational Diffie-Hellman (SCDH) Assumption [MW96]

```
$\mathcal{I}_{q\text{-DHE}}(1^\kappa)$
    $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$; $x \leftarrow \mathbb{Z}_p$
    Return $(pub = (\mathbb{G}_p, [1], [x], \dots, [x^q]), priv = x)$

$\mathcal{V}_{q\text{-DHE}}(pub, priv = x, sol = [z])$
    If $[z] = \left[x^{q+1}\right]$ return 1
    Else return 0
```

q-Diffie-Hellman Exponent (q-DHE) Assumption [ZSNS04]

```
$\mathcal{I}_{q\text{-GDHE}}(1^\kappa)$
    $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$; $x \leftarrow \mathbb{Z}_p$
    $pub = (\mathbb{G}_p, [1], [x], \dots, \left[x^{q-1}\right], \left[x^{q+1}\right], \dots, \left[x^{2q}\right])$
    Return $(pub, priv = x)$

$\mathcal{V}_{q\text{-GDHE}}(pub, priv = x, sol = [z])$
    If $[z] = [x^q]$ return 1
    Else return 0
```

q-Generalized Diffie-Hellman Exponent (q-GDHE) Assumption [BBG05,BGW05]

```
$\mathcal{I}_{q\text{-SDH}}(1^\kappa)$
    $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$; $x \leftarrow \mathbb{Z}_p$
    Return $(pub = (\mathbb{G}_p, [1], [x], \dots, [x^q]), priv = x)$

$\mathcal{V}_{q\text{-SDH}}(pub, priv = x, sol = (c, [z]))$
    If $[z] = \left[\frac{1}{x+c}\right]$ & $c \in \mathbb{Z}_p \setminus \{-x\}$
    return 1 Else return 0
```

q-Strong Diffie-Hellman (q-SDH) Assumption [BB08]

```
$\mathcal{I}_{q\text{-DHI}}(1^\kappa)$
    $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$; $x \leftarrow \mathbb{Z}_p^*$
    Return $(pub = (\mathbb{G}_p, [1], [x], \dots, [x^q]), priv = x)$

$\mathcal{V}_{q\text{-DHI}}(pub, priv = x, sol = [z])$
    If $[z] = \left[\frac{1}{x}\right]$ return 1
    Else return 0
```

q-Diffie-Hellman Inversion (q-DHI) Assumption [MSK02]

```
$\mathcal{I}_{\mathrm{SRDH}}(1^\kappa)$
    $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$; $x \leftarrow \mathbb{Z}_p$
    Return $\left(pub = (\mathbb{G}_p, [1], \left[x^2\right]), priv = x\right)$

$\mathcal{V}_{\mathrm{SRDH}}(pub, priv = x, sol = [z])$
    If $[z] = [\pm x]$ return 1
    Else return 0
```

Square Root Diffie-Hellman (SRDH) Assumption [KMS04]

Fig. 1: Some existing non-interactive intractability assumptions

we may without loss of generality assume she only generates elements as linear combinations of group elements it has already seen.

All the assumptions listed in Fig. 1 are secure in the generic group model.

3 Target Assumptions

The assumptions that can be defined over cyclic groups are legion. We will focus on the broad class of non-interactive computational assumptions where the adversary's goal is to compute a particular group element. We refer to them as *target assumptions*.

The CDH assumption is an example of a target assumption where the adversary has to compute a specific group element. She is given $([1],[x],[y]) \in \mathbb{G}_p^3$ and is tasked with computing $[xy] \in \mathbb{G}_p$.

We aim for maximal generality of the class of assumptions and will therefore also capture assumptions where the adversary takes part in specifying the target element to be computed. In the q-SDH assumption for instance the adversary is given $([1],[x],\ldots,[x^q]) \in \mathbb{G}_p^{q+1}$ and is tasked with finding $c \in \mathbb{Z}_p$ and $[\frac{1}{x+c}] \in \mathbb{G}_p$. Here the problem instance in itself does not dictate which group element the adversary must compute but the output of the adversary includes c, which uniquely determines the target element to be computed.

We will now define target assumptions. The class will be defined very broadly in order to capture existing assumptions in the literature such as CDH and q-SDH as well as other assumptions that may appear in future works.

In a target assumption, the instance generator outputs *pub* that includes a prime order group, a number of group elements, and possibly some additional information. Often, the group elements are of the form $[a(x)]$, where a is a polynomial and x is chosen uniformly at random from $\mathbb{Z}_p$. We generalize this by letting the instance generator output group elements of the form $[\frac{a(\boldsymbol{x})}{b(\boldsymbol{x})}]$, where $a(\boldsymbol{X})$ and $b(\boldsymbol{X})$ are multi-variate polynomials and $\boldsymbol{x}$ is chosen uniformly at random from $\mathbb{Z}_p^m$. We will assume all the polynomials are known to the adversary, i.e., they will be explicitly given in the additional information in *pub* in the form of their coefficients. The adversary will now specify a target group element. She does so by specifying polynomials $r(\boldsymbol{X})$ and $s(\boldsymbol{X})$ and making an attempt at computing the group element $[\frac{r(\boldsymbol{x})}{s(\boldsymbol{x})}]$.

If the target element can be computed using generic group operations on the group elements in *pub*, then the problem is easy to solve and hence the assumption is trivially false. To exclude trivially false assumptions, the solution verifier will therefore check that for all fixed linear combinations $\alpha_1,\ldots,\alpha_n \in \mathbb{Z}_p$ there is a low probability over the choice of $\boldsymbol{x}$ that $\frac{r(\boldsymbol{x})}{s(\boldsymbol{x})} = \sum_i \alpha_i \frac{a_i(\boldsymbol{x})}{b_i(\boldsymbol{x})}$. The solution tuple output by the adversary is $sol = \big(r(\boldsymbol{X}), s(\boldsymbol{X}), [y], sol'\big)$, where sol' is some potential extra information the verifier may need to check, for instance about how the polynomials r and s were constructed.

Definition 3 (Target assumption). *Given polynomials $d(\kappa), m(\kappa)$ and $n(\kappa)$ we say $(\mathcal{I},\mathcal{V})$ is a (d,m,n)-target assumption for $\mathcal{G}$ if they can be defined by a PPT algorithm $\mathcal{I}^{core}$ and a DPT algorithm $\mathcal{V}^{core}$ such that*

$(pub, priv) \leftarrow \mathcal{I}(1^\kappa)$**:** *Algorithm $\mathcal{I}$ proceeds as follows:*

- $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$
- $\left(\left\{\frac{a_i(\boldsymbol{X})}{b_i(\boldsymbol{X})}\right\}_{i=1}^{n}, pub', priv'\right) \leftarrow \mathcal{I}^{core}(\mathbb{G}_p)$
- $\boldsymbol{x} \leftarrow \mathbb{Z}_p^m$ *conditioned on* $b_i(\boldsymbol{x}) \neq 0$
- $pub := \left(\mathbb{G}_p, \left\{\left[\frac{a_i(\boldsymbol{x})}{b_i(\boldsymbol{x})}\right]\right\}_{i=1}^{n}, \left\{\frac{a_i(\boldsymbol{X})}{b_i(\boldsymbol{X})}\right\}_{i=1}^{n}, pub'\right)$
- *Return* $(pub, priv = ([1], \boldsymbol{x}, priv'))$

$0/1 \leftarrow \mathcal{V}\left(pub, priv, sol = \big(r(\boldsymbol{X}), s(\boldsymbol{X}), [y], sol'\big)\right)$**:** *Algorithm $\mathcal{V}$ returns* 1 *if all of the following checks pass and* 0 *otherwise:*

- $r(\boldsymbol{X})\prod_{i=1}^{n} b_i(\boldsymbol{X}) \notin \operatorname{span}\left\{\left\{s(\boldsymbol{X})a_j(\boldsymbol{X})\prod_{i\neq j} b_i(\boldsymbol{X})\right\}_{j=1}^{n}\right\}$
- $[y] = \frac{r(\boldsymbol{x})}{s(\boldsymbol{x})}[1]$
- $\mathcal{V}^{core}(pub, priv, sol) = 1$

We require that the number of indeterminates in $\boldsymbol{X}$ is $m(\kappa)$ and each polynomial $a_1(\boldsymbol{X}), b_1(\boldsymbol{X}), \ldots, a_n(\boldsymbol{X}), b_n(\boldsymbol{X}), r(\boldsymbol{X}), s(\boldsymbol{X})$ has total degree bounded by $d(\kappa)$, both of which can easily be checked by $\mathcal{V}^{core}$.

It is easy to see that all assumptions in Fig. 1 are target assumptions. For CDH for instance, we have $d = 2, m = 2, n = 3$, $a_1(X_1, X_2) = 1, a_2(X_1, X_2) = X_1, a_3(X_1, X_2) = X_2$, and $b_1(X_1, X_2) = b_2(X_1, X_2) = b_3(X_1, X_2) = 1$. Algorithm $\mathcal{V}^{\text{core}}$ then checks that the adversary's output is $r(X_1, X_2) = X_1X_2$ and $s(X_1, X_2) = 1$, which means the adverary is trying to compute the target $[x_1x_2] \in \mathbb{G}_p$.

```
Algorithm $\mathcal{I}_{\mathsf{B}}(1^\kappa)$
- $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$
- $\left(\left\{\frac{a_i(\boldsymbol{X})}{b_i(\boldsymbol{X})}\right\}_{i=1}^{n}, pub'_{\mathsf{A}}, priv'_{\mathsf{A}}\right) \leftarrow \mathcal{I}^{\text{core}}_{\mathsf{A}}(\mathbb{G}_p)$
- For $i = 1, \ldots, n$ set $c_i(\boldsymbol{X}) := a_i \prod_{j\neq i} b_j(\boldsymbol{X})$
- $\boldsymbol{x} \leftarrow \mathbb{Z}_p^m$
- $pub_{\mathsf{B}} = \left(\mathbb{G}_p, \{[c_i(\boldsymbol{x})]\}_{i=1}^{n}, \{c_i(\boldsymbol{X})\}_{i=1}^{n}, pub'_{\mathsf{B}} = \left(\{a_i(\boldsymbol{X}), b_i(\boldsymbol{X})\}_{i=1}^{n}, pub'_{\mathsf{A}}\right)\right)$
- Return $(pub_{\mathsf{B}}, priv_{\mathsf{B}} = ([1], \boldsymbol{x}, priv'_{\mathsf{A}}))$

Algorithm $\mathcal{V}_{\mathsf{B}}\left(pub_{\mathsf{B}}, priv_{\mathsf{B}} = ([1], \boldsymbol{x}, priv'_{\mathsf{A}}), sol_{\mathsf{B}} = \left(t(\boldsymbol{X}), s(\boldsymbol{X}), [y], sol'\right)\right)$
- Check $t(\boldsymbol{X}) \notin \operatorname{span}\{s(\boldsymbol{X})c_1(\boldsymbol{X}), \ldots, s(\boldsymbol{X})c_n(\boldsymbol{X})\}$
- Check $[y] = \frac{t(\boldsymbol{x})}{s(\boldsymbol{x})}[1]$
- Check $t(\boldsymbol{X}) = r(\boldsymbol{X})\prod_{j=1}^{n} b_j(\boldsymbol{X})$ for a polynomial $r(\boldsymbol{X})$ of total degree $d$
- Let $pub_{\mathsf{A}} = (\mathbb{G}_p, \{[c_i(\boldsymbol{x})]\}_{i=1}^{n}, \left\{\frac{a_i(\boldsymbol{X})}{b_i(\boldsymbol{X})}\right\}_{i=1}^{n}, pub'_{\mathsf{A}})$
- Let $priv_{\mathsf{A}} = ([\prod_{j=1}^{n} b_j(\boldsymbol{x})], \boldsymbol{x}, priv'_{\mathsf{A}})$
- Check $\mathcal{V}^{\text{core}}_{\mathsf{A}}\left(pub_{\mathsf{A}}, priv_{\mathsf{A}}, sol_{\mathsf{A}} = (r(\boldsymbol{X}), s(\boldsymbol{X}), [y], sol')\right) = 1$

Adversary $\mathcal{B}\left(\mathbb{G}_p, \{[c_i(\boldsymbol{x})]\}_{i=1}^{n}, \{c_i(\boldsymbol{X})\}_{i=1}^{n}, \{a_i(\boldsymbol{X}), b_i(\boldsymbol{X})\}_{i=1}^{n}, pub'_{\mathsf{A}}\right)$
- $\left(r(\boldsymbol{X}), s(\boldsymbol{X}), [y], sol'\right) \leftarrow \mathcal{A}\left(\mathbb{G}_p, \{[c_i(\boldsymbol{x})]\}_{i=1}^{n}, \left\{\frac{a_i(\boldsymbol{X})}{b_i(\boldsymbol{X})}\right\}_{i=1}^{n}, pub'_{\mathsf{A}}\right)$
- Return $sol_{\mathsf{B}} = \left(t(\boldsymbol{X}) = r(\boldsymbol{X})\prod_{j=1}^{n} b_j(\boldsymbol{X}), s(\boldsymbol{X}), [y], sol'\right)$
```

Fig. 2: Simple target assumption $\mathsf{B} = (\mathcal{I}_{\mathsf{B}}, \mathcal{V}_{\mathsf{B}})$ and adversary $\mathcal{B}$ against it

In q-SDH we have $d = q$, $m = 1$, $n = q + 1$, $a_i(X) = X^{i-1}$, and $b_i(X) = 1$ for $i = 1, \ldots, q + 1$. Algorithm $\mathcal{V}^{\text{core}}$ checks that the target polynomials are of the form $r(X) = 1$ and $s(X) = c + X$ for some $c \in \mathbb{Z}_p$, meaning the adversary it is trying to compute the target $[\frac{1}{x+c}] \in \mathbb{G}_p$.

3.1 Simple Target Assumptions

We now have a very general definition of target assumptions relating to the computation of group elements. In the following subsections, we go through progressively simpler classes of assumptions that imply the security of target assumptions. We start by defining simple target assumptions, where the divisor polynomials in the instance are trivial, i.e., $b_1(\boldsymbol{X}) = \cdots = b_n(\boldsymbol{X}) = 1$.

Definition 4 (Simple Target Assumption). *We say a (d, m, n)-target assumption $(\mathcal{I}, \mathcal{V})$ for $\mathcal{G}$ is simple if the instance generator always picks polynomials $b_1(\boldsymbol{X}) = \ldots = b_n(\boldsymbol{X}) = 1$.*

Next, we will prove that the security of simple target assumptions implies the security of all target assumptions. The idea is to reinterpret the tuple the adversary gets using the random generator [1] to having random generator $[\prod_{i=1}^{n} b_i(\boldsymbol{x})]$. Now all fractions of formal polynomials are scaled up by a factor $\prod_{i=1}^{n} b_i(\boldsymbol{X})$ and the divisor polynomials can be cancelled out.

Theorem 1. *For any (d, m, n)-target assumption $\mathsf{A} = (\mathcal{I}_\mathsf{A}, \mathcal{V}_\mathsf{A})$ there exists a $((n+1)d, m, n)$-simple target assumption $\mathsf{B} = (\mathcal{I}_\mathsf{B}, \mathcal{V}_\mathsf{B})$ such that $\mathsf{B} \Rightarrow \mathsf{A}$.*

Proof. Given an assumption $\mathsf{A} = (\mathcal{I}_\mathsf{A}, \mathcal{V}_\mathsf{A})$ and an adversary $\mathcal{A}$ against it, we define a simple target assumption $\mathsf{B} = (\mathcal{I}_\mathsf{B}, \mathcal{V}_\mathsf{B})$ and an adversary $\mathcal{B}$ (that uses adversary $\mathcal{A}$ in a black-box manner) against it as illustrated in Fig. 2. The key observation is that as long as $\prod_{i=1}^{n} b_i(\boldsymbol{x}) \neq 0$, the two vectors of group elements $([c_i(\boldsymbol{x})], \ldots, [c_n(\boldsymbol{x})])$ and $\left([\frac{a_1(\boldsymbol{x})}{b_1(\boldsymbol{x})}], \ldots, [\frac{a_n(\boldsymbol{x})}{b_n(\boldsymbol{x})}]\right)$ are identically distributed. By the specification of assumption A, it follows that $b_i(\boldsymbol{X}) \not\equiv 0$ for all $\in \{1, \ldots, n\}$. By the Schwartz-Zippel lemma, the probability that $\prod_{i=1}^{n} b_i(\boldsymbol{x}) = 0$ is at most $\frac{dn}{p}$. Thus, if $\mathcal{A}$ has success probability $\epsilon_\mathcal{A}$, then $\mathcal{B}$ has success probability $\epsilon_\mathcal{B} \geq \epsilon_\mathcal{A} - \frac{dn}{p}$. □

3.2 Univariate Target Assumptions imply Multivariate Target Assumptions

We will now show that security of target assumptions involving univariate polynomials imply security of target assumptions involving multivariate polynomials. The following theorem proves that given a multivariate target assumption, there is a univariate target assumption whose intractability implies that of the former.

Theorem 2. *For any (d, m, n)-simple target assumption $\mathsf{A} = (\mathcal{I}_\mathsf{A}, \mathcal{V}_\mathsf{A})$ there exists a $((n+1)d, 1, n)$-simple target assumption $\mathsf{B} = (\mathcal{I}_\mathsf{B}, \mathcal{V}_\mathsf{B})$ where $\mathsf{B} \Rightarrow \mathsf{A}$.*

Proof. Given $\mathsf{A} = (\mathcal{I}_\mathsf{A}, \mathcal{V}_\mathsf{A})$ and an adversary $\mathcal{A}$ with success probability $\epsilon_\mathcal{A}$ against it, we define a simple target assumption $(\mathcal{I}_\mathsf{B}, \mathcal{V}_\mathsf{B})$ with univariate polynomials and construct an adversary $\mathcal{B}$ (that uses $\mathcal{A}$ in a black-box manner) against it as illustrated in Fig. 3.

```
Algorithm $\mathcal{I}_{\mathsf{B}}(1^\kappa)$
- $(\mathbb{G}_p, [1]) \leftarrow \mathcal{G}(1^\kappa)$
- $\left( \{a_i(\boldsymbol{X})\}_{i=1}^n, pub'_{\mathsf{A}}, priv'_{\mathsf{A}} \right) \leftarrow \mathcal{I}_{\mathsf{A}}^{\text{core}}(\mathbb{G}_p)$
- Randomly choose $\boldsymbol{c}(X) \leftarrow (\mathbb{Z}_p[X])^m$ where $\deg(c_i) = n+1$
- For $i = 1, \ldots, n$ let $b_i(X) := a_i(\boldsymbol{c}(X))$
- $pub'_{\mathsf{B}} := \left(\{a_i(\boldsymbol{X})\}_{i=1}^n, \boldsymbol{c}(X), pub'_{\mathsf{A}}\right)$
- $x \leftarrow \mathbb{Z}_p$
- Return $\left(pub_{\mathsf{B}} = \left(\mathbb{G}_p, \{[b_i(x)]\}_{i=1}^n, \{b_i(X)\}_{i=1}^n, pub'_{\mathsf{B}}\right), priv_{\mathsf{B}} = ([1], x, priv'_{\mathsf{A}})\right)$

Algorithm $\mathcal{V}_{\mathsf{B}}\left(pub_{\mathsf{B}}, priv_{\mathsf{B}}, sol_{\mathsf{B}} = \left(r(X), s(X), [y], (r_{\mathsf{A}}(\boldsymbol{X}), s_{\mathsf{A}}(\boldsymbol{X}), sol'_{\mathsf{A}})\right)\right)$
- Check $r(X) \notin \text{span}\{s(X)b_1(X), \ldots, s(X)b_n(X)\}$
- Check $[y] = \frac{r(x)}{s(x)}[1]$
- Let $pub_{\mathsf{A}} = (\mathbb{G}_p, \{[b_i(x)]\}_{i=1}^n, \{a_i(\boldsymbol{X})\}_{i=1}^n, pub'_{\mathsf{A}})$
- Let $priv_{\mathsf{A}} = ([1], x, priv'_{\mathsf{A}})$
- Check $\mathcal{V}_{\mathsf{A}}^{\text{core}}\left(pub_{\mathsf{A}}, priv_{\mathsf{A}}, sol_{\mathsf{A}} = (r_{\mathsf{A}}(\boldsymbol{X}), s_{\mathsf{A}}(\boldsymbol{X}), [y], sol'_{\mathsf{A}})\right) = 1$

Adversary $\mathcal{B}\left(\mathbb{G}_p, \{[b_i(x)]\}_{i=1}^n, \{b_i(X)\}_{i=1}^n, pub'_{\mathsf{B}}\right)$
- $\left(r_{\mathsf{A}}(\boldsymbol{X}), s_{\mathsf{A}}(\boldsymbol{X}), [y], sol'_{\mathsf{A}}\right) \leftarrow \mathcal{A}\left(\mathbb{G}_p, \{[b_i(x)]\}_{i=1}^n, \{a_i(\boldsymbol{X})\}_{i=1}^n, pub'_{\mathsf{A}}\right)$
- Return $\left(r_{\mathsf{A}}(\boldsymbol{c}(X)), s_{\mathsf{A}}(\boldsymbol{c}(X)), [y], sol'_{\mathsf{B}} = (r_{\mathsf{A}}(\boldsymbol{X}), s_{\mathsf{A}}(\boldsymbol{X}), sol'_{\mathsf{A}})\right)$
```

Fig. 3: Univariate simple target assumption B and adversary $\mathcal{B}$ against it

Without loss of generality we can assume $a_1(\boldsymbol{X}), \ldots, a_n(\boldsymbol{X})$ are linearly independent, and therefore the polynomials $r_{\mathsf{A}}(\boldsymbol{X})$ and $s_{\mathsf{A}}(\boldsymbol{X})a_1(\boldsymbol{X}), \ldots, s_{\mathsf{A}}(\boldsymbol{X})a_n(\boldsymbol{X})$ are all linearly independent. By Lemma 1 below this means that with probability $1 - \frac{d(n+1)}{p}$ the univariate polynomials $r(\boldsymbol{c}(X))$ and $s(\boldsymbol{c}(X))a_1(\boldsymbol{c}(X)), \ldots, s(\boldsymbol{c}(X))a_n(\boldsymbol{c}(X))$ output by $\mathcal{B}$ are also linearly independent since the only information about $\boldsymbol{c}(X)$ that $\mathcal{B}$ passes on to $\mathcal{A}$ can be computed from $(x, \boldsymbol{c}(x))$. This means $\mathcal{B}$ has advantage $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}} - \frac{d(n+1)}{p}$ against assumption B.

□

Lemma 1. *Let $a_1(\boldsymbol{X}), \ldots, a_n(\boldsymbol{X}) \in \mathbb{Z}_p[\boldsymbol{X}]$ be linearly independent m-variate polynomials of total degree bounded by d, and let $(x, \boldsymbol{x}) \in \mathbb{Z}_p \times \mathbb{Z}_p^m$. Pick a vector of m random univariate degree n polynomials $\boldsymbol{c}(X) \in (\mathbb{Z}_p[X])^m$ that passes through $(x, \boldsymbol{x})$, i.e., $\boldsymbol{c}(x) = \boldsymbol{x}$. The probability that $a_1(\boldsymbol{c}(X)), \ldots, a_n(\boldsymbol{c}(X))$ are linearly independent is at least $1 - \frac{dn}{p}$.*

Proof. Take n random points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{Z}_p^m$ and consider the matrix

$$M = \begin{pmatrix} a_1(\boldsymbol{x}_1) & \cdots & a_1(\boldsymbol{x}_n) \\ \vdots & & \vdots \\ a_n(\boldsymbol{x}_1) & \cdots & a_n(\boldsymbol{x}_n) \end{pmatrix}.$$

We will argue by induction that with probability $1 - \frac{dn}{p}$ the matrix is invertible. For $n = 1$ it follows from the Schwartz-Zippel lemma that the probability $a_1(\boldsymbol{x}_1) = 0$ is at most $\frac{d}{p}$. Suppose now by induction hypothesis that we have probability $1 - \frac{d(n-1)}{p}$ that the top left $(n-1) \times (n-1)$ matrix is invertible. When it is invertible, the values $a_n(\boldsymbol{x}_1), \ldots, a_n(\boldsymbol{x}_{n-1})$ uniquely determine $\alpha_1, \ldots, \alpha_{n-1}$ such that for $j = 1, \ldots, n-1$ we have $a_n(\boldsymbol{x}_j) = \sum_{i=1}^{n-1} \alpha_i a_i(\boldsymbol{x}_j)$. Since the polynomials $a_1(\boldsymbol{X}), \ldots, a_n(\boldsymbol{X})$ are linearly independent, by the Schwartz-Zippel lemma, there is at most probability $\frac{d}{p}$ that we also have $a_n(\boldsymbol{x}_n) = \sum_{i=1}^{n-1} \alpha_i a_i(\boldsymbol{x}_n)$. So the row $(a_n(\boldsymbol{x}_1), \ldots, a_n(\boldsymbol{x}_n))$ is linearly independent of the other rows, and hence we have M is invertible with at least probability $1 - \frac{d(n-1)}{p} - \frac{d}{p} = 1 - \frac{dn}{p}$.

Finally, picking a vector of m random polynomials $\boldsymbol{c}(X)$ of degree n such that $\boldsymbol{c}(x) = \boldsymbol{x}$ and evaluating it in distinct points $x_1, \ldots, x_n \leftarrow \mathbb{Z}_p \setminus \{x\}$ gives us n random points $\boldsymbol{c}(x_j) \in \mathbb{Z}_p^m$. So the matrix

$$\begin{pmatrix} a_1(\boldsymbol{c}(x_1)) & \cdots & a_1(\boldsymbol{c}(x_n)) \\ \vdots & & \vdots \\ a_n(\boldsymbol{c}(x_1)) & \cdots & a_n(\boldsymbol{c}(x_n)) \end{pmatrix}$$

has at least probability $1 - \frac{dn}{p}$ of being invertible. If $\sum_{i=1}^{n} \alpha_i a_i(\boldsymbol{c}(X)) = 0$, then it must hold in the distinct points $x_1, \ldots, x_n$, and we can see there is only the trivial linear combination with $\alpha_1 = \cdots = \alpha_n = 0$. □

Having reduced target assumptions to simple univariate target assumptions with $m = 1$, we will in the next two subsections consider two separate cases. First, the case where the adversary's polynomial $s(X)$ is fixed, i.e., it can be deterministically computed. Second, the case where the adversary's polynomial $s(X)$ may vary.

3.3 Polynomial Assumptions

We now consider simple target assumptions with univariate polynomials where $s(X)$ is fixed. We can without loss of generality assume this means *priv'* output by the instance generator contains $s(X)$ and the solution verifier checks whether the adversary's solution matches $s(X)$. There are many assumptions where $s(X)$ is fixed, in the Diffie-Hellman inversion assumption we will for instance always have $s(X) = X$ and in the q-GDHE assumption we always have $s(X) = 1$. When the polynomial $s(X)$ is fixed, we can multiply it away as we did for the multivariate polynomials $b_1(\boldsymbol{X}), \ldots, b_n(\boldsymbol{X})$ when reducing target assumptions to simple target assumptions. This leads us to define the following class of assumptions:

Definition 5 (Polynomial Assumption). *We say a $(d, 1, n)$-simple target assumption $(\mathcal{I}, \mathcal{V})$ for $\mathcal{G}$ is a (d, n)* polynomial assumption *if $\mathcal{V}$ only accepts a solution with $s(X) = 1$.*

We leave the proof of the following theorem to the reader.

Theorem 3. *For any $(d, 1, n)$-simple target assumption $\mathsf{A} = (\mathcal{I}_{\mathsf{A}}, \mathcal{V}_{\mathsf{A}})$ for $\mathcal{G}$ where the polynomial $s(X)$ is fixed, there is a $(2d, n)$-polynomial assumption $\mathsf{B} = (\mathcal{I}_{\mathsf{B}}, \mathcal{V}_{\mathsf{B}})$ where $\mathsf{B} \Rightarrow \mathsf{A}$.*

We will now show that all polynomial assumptions are implied by the generalized Diffie-Hellman exponent (q-GDHE) assumptions (cf. Fig. 1) along the lines of [GGPR13]. This means the q-GDHE assumptions are Uber assumptions that imply the security of a major class of target assumptions, which includes a majority of the non-interactive computational assumptions for prime-order groups found in the literature.

Theorem 4. *For any (d, n)-polynomial assumption $\mathsf{A} = (\mathcal{I}_{\mathsf{A}}, \mathcal{V}_{\mathsf{A}})$ for $\mathcal{G}$ we have that $(d+1)$-GDHE $\Rightarrow \mathsf{A}$.*

Proof. Let $\mathcal{A}$ be an adversary against the (d, n)-polynomial assumption. We show how to build an adversary $\mathcal{B}$, which uses $\mathcal{A}$ in a black-box manner to break the $(d+1)$-GDHE assumption. Adversary $\mathcal{B}$ gets $[1], [x], \ldots, \left[x^d\right], [x^{d+2}], \ldots, \left[x^{2d+2}\right]$ from the $(d+1)$-GDHE instance generator and her aim is to output the element $\left[x^{d+1}\right] \in \mathbb{G}_p$. Adversary $\mathcal{B}$ uses algorithm $\mathcal{I}_{\mathsf{A}}^{\text{core}}$ to generate a simulated polynomial problem instance as described below, which she then forwards to $\mathcal{A}$.

```
Adversary $\mathcal{B}\left(\mathbb{G}_p, [1], [x], \ldots, \left[x^d\right], \left[x^{d+2}\right], \ldots, \left[x^{2d+2}\right]\right)$
- $\left(\{a_i(X)\}_{i=1}^n, pub'_{\mathsf{A}}, priv'_{\mathsf{A}}\right) \leftarrow \mathcal{I}_{\mathsf{A}}^{\text{core}}(\mathbb{G}_p)$
- Randomly choose $c(X) \leftarrow \mathbb{Z}_p[X]$ where $\deg(c) = d+1$ and
  no $a_i(X)c(X)$ includes the term $X^{d+1}$
- For all $i$ compute $[a_i(x)c(x)]$ using the $(d+1)$-GDHE tuple
- Run $(r(X), [y], sol') \leftarrow \mathcal{A}\left(\mathbb{G}_p, \{[a_i(x)c(x)]\}_{i=1}^n, \{a_i(X)\}_{i=1}^n, pub'_{\mathsf{A}}\right)$
- Parse $r(X)c(X)$ as $\sum_{i=0}^{2d+1} t_i X^i$
- Return $\frac{1}{t_{d+1}}\left([y] - [\sum_{i \neq d+1} t_i x^i]\right)$
```

Assuming $c(x) \neq 0$, which happens with probability $1 - \frac{d+1}{p}$, the input to $\mathcal{A}$ looks identical to a normal problem instance for assumption A with generator $[c(x)]$. Furthermore, if $\mathcal{A}$ finds a satisfactory solution to this problem, we then have $[y] = r(x)[c(x)]$. By Lemma 2 below there is at most $\frac{1}{p}$ chance of returning $r(X)$ such that $r(X)c(X)$ has coefficient 0 for X^{d+1} and hence using the $(2d+2)$ elements from the $(d+1)$-GDHE tuple, we can recover $[x^{d+1}]$ from $[y]$. We now get that if $\mathcal{A}$ has advantage $\epsilon_{\mathcal{A}}$ against A, then $\mathcal{B}$ has advantage $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}} - \frac{d+2}{p}$ against the $(d+1)$-GDHE assumption. □

Lemma 2 (Lemma 10 from [GGPR13]). *Let $\{a_i(X)\}_{i=1}^n$ be polynomials of degree at most d. Pick $x \leftarrow \mathbb{Z}_p$ and $c(X)$ as a random degree $d+1$ polynomial such that all products $b_i(X) = a_i(X)c(X)$ have coefficient 0 for X^{d+1}. Given $(\{a_i(X)\}_{i=1}^n, x, c(x))$, the probability of guessing a non-trivial degree d polynomial $r(X)$ such that $r(X)c(X)$ has coefficient 0 for X^{d+1} is at most $\frac{1}{p}$.*

3.4 Fractional assumptions

We now consider the alternative case of simple target assumptions with univariate polynomials, where $s(X)$ is not fixed. When $s(X)|r(X)$, we can without loss of generality divide out and get $s(X) = 1$. The remaining case is when $s(X) \nmid r(X)$, which we now treat.

Definition 6 ((d, n)-Fractional Assumption). *We say a $(d, 1, n)$-simple target assumption $(\mathcal{I}, \mathcal{V})$ for $\mathcal{G}$ is an (d, n)-*fractional assumption *if $\mathcal{V}$ only accepts the solution if $s(X) \nmid r(X)$.*

Next we define a simple fractional assumption which we refer to for short as q-SFrac, which says given the tuple $([1], [x], [x^2], \ldots, [x^q])$ it is hard to compute $\left[\frac{r(x)}{s(x)}\right]$ when $\deg(r) < \deg(s)$. The simple fractional assumption is a straightforward generalization of the q-SDH assumption, where $\deg(r) = 0$ since $r(X) = 1$ and $\deg(s) = 1$. We prove in Appendix A that q-SFrac holds in the generic group model.

Definition 7 (q-SFrac Assumption). *The q-SFrac assumption is a simple target assumption where $n = q+1$, $a_i(X) = X^{i-1}$, and $0 \leq \deg(r) < \deg(s) \leq q$.*

We now prove the following theorem.

Theorem 5. *For any (d, n)-fractional assumption $\mathsf{A} = (\mathcal{I}, \mathcal{V})$ for $\mathcal{G}$ we have d-SFrac$\Rightarrow\mathsf{A}$.*

Proof. Let $\mathcal{A}$ be an adversary against a (d, n)-fractional assumption A. We show how to use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against the d-SFrac assumption. Adversary $\mathcal{B}$ gets $\left(\mathbb{G}_p, [1], [x], \ldots, \left[x^d\right]\right)$ and her aim is to output a valid solution of the form $\left(r'(X), s'(X), \left[\frac{r'(x)}{s'(x)}\right]\right)$ where $\deg(r') < \deg(s') \leq d$. Adversary $\mathcal{B}$ uses the instance generator algorithm $\mathcal{I}^{\text{core}}$ of the fractional assumption as described below to generate a problem instance, which she then forwards to $\mathcal{A}$.

```
Algorithm $\mathcal{B}(\mathbb{G}_p, [1], [x], \ldots, [x^d])$
- $\left(\{a_i(X)\}_{i=1}^n, pub', priv'\right) \leftarrow \mathcal{I}^{\text{core}}(\mathbb{G}_p)$
- For all $i$ compute $[a_i(x)]$ using the $d$-SFrac tuple $\left([1]\,[x], \ldots, \left[x^d\right]\right)$
- Run $(r(X), s(X), [y]) \leftarrow \mathcal{A}\left(\mathbb{G}_p, \{[a_i(x)]\}_{i=1}^n, \{a_i(X)\}_{i=1}^n, pub'\right)$
- Let $r'(X) = r(X) \bmod s(X)$ and write $r(X) = t(X)s(X) + r'(X)$
- Return $(r(X)', s(X), [y] - [t(x)])$
```

The advantage of adversary $\mathcal{B}$ against the d-SFrac assumption is the same as that of adversary $\mathcal{A}$ against the fractional assumption A.

□

3.5 The q-SFrac and q-GDHE Assumptions Together Imply All Target Assumptions in Cyclic Groups

We now prove that the q-SFrac and q-GDHE assumptions together constitute an Uber assumption for all target assumptions in prime-order cyclic groups.

Theorem 6. *There is a polynomial $q(d,m,n)$ such that the joint q-SFrac and q-GDHE assumption implies all (d,m,n)-target assumptions.*

Proof. Let A be a (d,m,n)-target assumption. By Theorem 1, for any adversary $\mathcal{A}$ with advantage $\epsilon_{\mathcal{A}}$ against A, we can define a $(d(n+1),m,n)$-simple target assumption A_1 and an adversary $\mathcal{A}_1$ with advantage $\epsilon_{\mathcal{A}_1} \geq \epsilon_{\mathcal{A}} - \frac{dn}{p}$ against it. By Theorem 2, using $\mathcal{A}_1$ against A_1, we can define a $(d(n+1)^2,1,n)$-simple target assumption A_2 and an adversary $\mathcal{A}_2$ against it with advantage $\epsilon_{\mathcal{A}_2} \geq \epsilon_{\mathcal{A}_1} - \frac{d(n+1)^2}{p} \geq \epsilon_{\mathcal{A}} - \frac{d(n+(n+1)^2)}{p}$. We now have two cases as follows:

- With non-negligible probability a successful solution has $s_{\mathsf{A}_2}(X) \nmid r_{\mathsf{A}_2}(X)$. By Theorem 5, we can use $\mathcal{A}_2$ to construct an adversary $\mathcal{A}_3$ against the $(d(n+1)^2)$-SFrac assumption where advantage $\epsilon_{\mathcal{A}_3} \geq \epsilon_{\mathcal{A}_2} \geq \epsilon_{\mathcal{A}} - \frac{d(n+(n+1)^2)}{p}$. Since by defintion $d,n \in \mathsf{Poly}(\kappa)$ and $\log p \in \theta(\kappa)$, it follows that $\frac{d(n+(n+1)^2)}{p}$ is negligible (in κ).
- With overwhelming probability a successful solution uses polynomials where $s_{\mathsf{A}_2}(X)|r_{\mathsf{A}_2}(X)$ which is equivalent to the case where $s_{\mathsf{A}_2}(X) = 1$. By Theorem 3, using $\mathcal{A}_2$ we can define a $(2d(n+1)^2,n)$-polynomial assumption A_3 and an adversary $\mathcal{A}_3$ with advantage $\epsilon_{\mathcal{A}_3} \geq \epsilon_{\mathcal{A}_2} - \frac{4d(n+1)^2}{p} \geq \epsilon_{\mathcal{A}} - \frac{d(n+5(n+1)^2)}{p}$. By Theorem 4, using adversary $\mathcal{A}_3$, we can construct an adversary $\mathcal{A}_4$ against the $(2d(n+1)^2+1)$-GDHE assumption with advantage $\epsilon_{\mathcal{A}_4} \geq \epsilon_{\mathcal{A}_3} - \frac{2d(n+1)^2+2}{p}$. From which it follows that $\epsilon_{\mathcal{A}_4} \geq \epsilon_{\mathcal{A}} - \frac{d(7(n+1)^2+n)+2}{p}$. Since by defintion $d,n \in \mathsf{Poly}(\kappa)$ and $\log p \in \theta(\kappa)$, it follows that $\frac{d(7(n+1)^2+n)+2}{p}$ is negligible (in κ).

□

4 The Relationship between the GDHE and SFrac Assumptions

Having identified the q-GDHE and q-SFrac assumptions as Uber assumptions for all target assumptions, it is natural to investigate their internal structure and their relationship to each other. One obvious question is whether a further simplification is possible and one of the assumption classes imply the other. We first analyze the case where $q \geq 2$ and show that q-SFrac does not imply 2-GDHE for generic algorithms. This means that we need the q-GDHE assumptions to capture the polynomial target assumptions, the q-SFrac assumptions cannot act as an Uber assumption for all target assumptions on their own.

We also look at the lowest level of the q-SFrac and q-GDHE hierarchies. Observe that the 1-SFrac assumption is equivalent to the 1-SDH assumption. We prove that the 1-GDHE assumption is equivalent to the CDH assumption. This immediately also gives us that the 1-SFrac assumption implies the 1-GDHE assumption since the 1-SDH assumption implies the CDH assumption. We summarize the implications we prove in the diagram in Fig. 4.

$$\begin{tikzcd}
q\text{-GDHE} \arrow[r, bend left, "\text{Lemma 4}"] & (q-1)\text{-GDHE} \arrow[l, Rightarrow, "/" marking, "\text{Lemma 5}"] \arrow[r, bend left, "\text{Lemma 4}"] & \dots \arrow[l, Rightarrow, "/" marking, "\text{Lemma 5}"] \arrow[r, bend left, "\text{Lemma 4}"] & 1\text{-GDHE} \arrow[l, Rightarrow, "/" marking, "\text{Lemma 5}"] \arrow[dr, Rightarrow, bend left, "\text{Easy}"] & \\
 & & & & \text{CDH} \arrow[ul, Rightarrow, bend right, "\text{Theorem 8}"] \\
q\text{-SDH} \arrow[uu, Rightarrow, "/" marking, "\text{Theorem 7}"] \arrow[r, Rightarrow, "\text{Easy}"'] & (q-1)\text{-SDH} \arrow[uu, Rightarrow, "/" marking, "\text{Theorem 7}"] \arrow[r, Rightarrow, "\text{Easy}"'] & \dots \arrow[uu, Rightarrow, "/" marking, "\text{Theorem 7}"] \arrow[r, Rightarrow, "\text{Easy}"'] & 1\text{-SDH} \arrow[uu, Rightarrow, "\text{Theorem 8}"] \arrow[ur, Rightarrow, bend right, "\text{Follows [Mau94]}"'] &
\end{tikzcd}$$

Fig. 4: Summary of Reductions

4.1 The SFrac Assumptions Do Not Imply the 2-GDHE Assumption

We prove here that the i-GDHE assumption for $i \geq 2$ is not implied by any q-SFrac assumption for generic adversaries, i.e. q-SFrac$\overset{\mathsf{GGM}}{\not\Rightarrow}$$i$-GDHE for all $i \geq 2$. More presicely, we show that providing an unbounded generic adversary $\mathcal{A}$ against a q-SFrac assumption with a 2-GDHE oracle $\mathcal{O}_{\text{2-GDHE}}$, which on input $([a], [b], [c], [d])$ where $b = az$, $c = az^3$, $d = az^4$ returns the element $\left[az^2\right]$, and returns the symbol $\perp$ if the input is malformed, does not help the adversary.

Theorem 7. *The q-SFrac assumption does not imply the 2-GDHE assumption in generic groups.*

Proof. Consider a generic adversary $\mathcal{A}$ which gets input $([1], [x], \dots, [x^q])$ and is tasked with outputting $\left(\frac{r(X)}{s(X)}, \left[\frac{r(x)}{s(x)}\right]\right)$, where $0 \leq \deg(r) < \deg(s) \leq q$. We give $\mathcal{A}$ access to an oracle $\mathcal{O}_{\text{2-GDHE}}(\cdot,\cdot,\cdot,\cdot)$ as above, which can be queried polynomially many times.

Since $\mathcal{A}$ is generic, the queries $([a], [b], [c], [d])$ she makes to the $\mathcal{O}_{\text{2-GDHE}}$ oracle must be constructed using generic group operations. To have any chance of making a non-trivial query, she must therefore pick the queries as known linear combinations of the elements $[1], [x], \dots, [x^q]$. Thus, we have

$$a = \sum_{j=0}^{q} \alpha_j x^j \qquad b = \sum_{j=0}^{q} \beta_j x^j \qquad c = \sum_{j=0}^{q} \gamma_j x^j$$

for known $\alpha_j, \beta_j, \gamma_j$. Let the corresponding formal polynomials be $a(X)$, $b(X)$ and $c(X)$, respectively. We have that $\deg(a), \deg(b), \deg(c) \in \{0, \dots, q\}$. By definition, for the input to the oracle $\mathcal{O}_{\text{2-GDHE}}$ to be well-formed, we must have $b = az$ and $c = az^3$ for some z. In the generic group model this has negligible probability of holding unless z corresponds to some (possibly rational) function $z(X)$ and we have $b(X) = a(X)z(X)$ and $c(X) = a(X)z(X)^3$ when viewed as formal polynomials.

If the adversary submits a query where $a(X) \equiv 0$ or $b(X) \equiv 0$, the oracle will just return $[0]$, which is useless to the adversary. So from now on let's assume that $a(X) \not\equiv 0$, $b(X) \not\equiv 0$ and $c(X) \not\equiv 0$.

We now have

$$c(X) = a(X)z(X)^3 = a(X)\left(\frac{b(X)}{a(X)}\right)^3 = \frac{b(X)^3}{a(X)^2}.$$

This means $a(X)^2|b(X)^3$, which implies $a(X)|b(X)^2$. The answer returned by the oracle on a well-formed input corresponds to $a(X)z(X)^2 = a(X)\left(\frac{b(X)}{a(X)}\right)^2 = \frac{b(X)^2}{a(X)}$. Since $a(X)|b(X)^2$, the answer corresponds to a proper polynomial.

If $\deg(b) \leq \deg(a)$, we have $2\deg(b) - \deg(a) \leq \deg(b) \leq q$, and if $\deg(b) \geq \deg(a)$, we have $2\deg(b) - \deg(a) \leq 3\deg(b) - 2\deg(a) = \deg(c) \leq q$. Thus, the answer the oracle returns corresponds to a known polynomial of degree in $\{0, \ldots, q\}$ which could have been computed by the adversary herself using generic group operations on the tuple $[1], [x], \ldots, [x^q]$ without calling the oracle. □

Since as we prove later the GDHE assumptions family is strictly increasingly stronger, we get the following corollary.

Corollary 1. *For all $i \geq 2$ it holds that q-SFrac$\overset{\mathsf{GGM}}{\not\Rightarrow}$$i$-GDHE.*

4.2 CDH Implies the 1-GDHE Assumption

Since having access to a CDH oracle allows one to compute any polynomial in the exponent [Mau94] (in fact, such an oracle provides more power as it allows computing even rational functions in the exponent for groups with known order [Mau94]), it is clear that 1-GDHE⇒CDH and hence q-SDH⇒CDH. We prove in this section the implication CDH⇒1-GDHE, which means that the assumptions CDH and 1-GDHE are equivalent. As a corollary q-SFrac ⇒1-GDHE for all q.

We start by proving that the square computational Diffie-Hellman assumption (SCDH) (cf. Fig. 1), which is equivalent to the CDH assumption [BDZ03,JR13], implies the the square root Diffie-Hellman (SRDH) assumption [KMS04] (cf. Fig. 1) [4]. Note that given a SCDH oracle, one can solve any CDH instance by making 2 calls to the SCDH oracle. Let $([1], [a], [b]) \in \mathbb{G}_p^3$ be a CDH instance, then $[ab] = \frac{1}{2}([(a+b)^2] - ([(a-b)^2]))$.

We remark here that Roh and Hahn [RH12] also gave a reduction from the SCDH assumption to the SRDH assumption. However, their reduction relies on two assumptions: that the oracle will always (i.e. with probability 1) return a correct answer when queried on quadratic-residue elements and uniformly random elements when queried on quadratic non-residue elements, and that the prime order of the group p has the special form $p = 2^t q + 1$ where $2^t = O(\kappa^{O(1)})$.

[4] The SRDH assumption differs from the 1-GDHE assumption in that while the former accepts either $[-x]$ or $[x]$ as a valid answer, the latter only accepts $[x]$ as a valid answer.

Our reduction is more general since we do not place any restrictions on t or q and more efficient since it uses $4t + 2|q|$ oracle queries, whereas their reduction uses $O(t(2^t + |q|))$ oracle queries. Later on, we will also show how to boost an imperfect 1-GDHE oracle to get a SCDH oracle.

The Perfect Oracle Case. We prove that a perfect SRDH oracle $\mathcal{O}_{\text{SRDH}}$ which on input a pair $([a], [az]) \in \mathbb{G}_p^2$ returns the symbol QNR (which for convenience we denote by $[0]$) if $z \notin \mathbb{QR}(p)$ and $[\pm a\sqrt{z}]$ otherwise, leads to a break of the SCDH assumption. The role of the exponent a is to allow queries on pairs w.r.t. a different group generator than the default one. Let $p = 2^t q+1$ for an odd positive integer q be the prime order of the group $\mathbb{G}_p$. Note that when $p \equiv 3 \pmod 4$ (this is the case when $-1 \in \mathbb{QNR}(p)$), we have $t = 1$, and in the special case where p is a safe prime, q is also a prime. On the other hand, when $p \equiv 1 \pmod 4$ (this is the case when $-1 \in \mathbb{QR}(p)$), we have $t > 1$.

In the following let $\omega \in \mathbb{QNR}(p)$ be an arbitrary 2^t-th root of unity of $\mathbb{Z}_p^\times$, i.e., $\omega^{2^{t-1}} \equiv -1 \bmod p-1$ and $\omega^{2^t} \equiv 1 \bmod p-1$. Note that there are $\phi(2^t) = 2^{t-1}$ roots of unity and finding one is easy since for any generator g of $\mathbb{Z}_p^\times$, g^q is a 2^t-th root of unity. We observe that all elements in $\mathbb{Z}_p^\times$ can be written in the form $\omega^i\beta$, where β has odd order $k|q$. The quadratic residues are those where i is even, and the quadratic non-residues are the ones where i is odd.

Theorem 8. *Given a perfect $\mathcal{O}_{SRDH}$ oracle, we can solve any SCDH instance using at most $4t + 2|q|$ oracle calls when the group order is $p = 2^t q + 1$ for odd q.*

Proof. Given a SCDH instance $([1], [x]) \in \mathbb{G}_p^2$, our task is to compute $\left[x^2\right] \in \mathbb{G}_p$. The task is trivial when $x = 0$, so let's from now on assume $x \neq 0$. Since any $x \in \mathbb{Z}_p^\times$ can be written as $x = \omega^i\beta$ where $\omega \in \mathbb{Z}_p^\times$ is a 2^t-th root of unity and $\beta \in \mathbb{Z}_p^\times$ has an odd order k where $k|q$, our task is to compute $\left[x^2\right] = \left[\omega^{2i}\beta^2\right]$. In the following, we will first describe an algorithm FindExpi that uses the square-root oracle to determine i. Next, we describe an algorithm Square that computes $[y] = \left[\omega^j\beta^2\right]$ for some j, and then use FindExpi to clean it up to get $\left[x^2\right] = \left[\omega^{2i-j}y\right] = \left[\omega^{2i}\beta^2\right]$. Both of these algorithms are given in Fig. 5.

Recall the perfect SRDH oracle responds with QNR, i.e. $[0]$, whenever it gets a quadratic non-residue as input, i.e., whenever it gets input $([1], [\omega^i\beta])$ for an odd i and β has an odd order. When it gets a quadratic residue as input, i.e., when i is even, it returns $\left[\pm\omega^{\frac{i}{2}}\beta^{\frac{1}{2}}\right]$. Since $\omega^{2^{t-1}} = -1$ this means it returns either $\left[\omega^{\frac{i}{2}}\beta^{\frac{1}{2}}\right]$ or $\left[\omega^{2^{t-1}+\frac{i}{2}}\beta^{\frac{1}{2}}\right]$.

Let the binary expension of the exponent be $i = i_{t-1}i_{t-2}\ldots i_0$. In the FindExpi algorithm we use the oracle to learn the least significant bit and also to right-shift the bits. Consider running the oracle on $([1], [x])$ and on $([1], [\omega^{-1}x])$. If $i_0 = 0$, then i is even and on the first input the oracle returns a new element with exponent $i_t i_{t-1}\ldots i_1$. If $i_1 = 1$ then i is odd, and the oracle returns a new element with exponent $i_t i_{t-1}\ldots i_1$ on the second input. Which call returns a non-trivial group element tells us what i_0 is and in both cases we get a new

```
Algorithm FindExpi($[1], [x], \omega$)
    $[y] \leftarrow [x]$
    $i \leftarrow 0$
    For $j = 0$ to $t-1$:
        $[z] \leftarrow \mathcal{O}_{\mathrm{SRDH}}([1], [y])$
        If $[z] = [0]$ /*$\mathcal{O}_{\mathrm{SRDH}}$ returned QNR*/
            $i \leftarrow i + 2^j$
            $[y] \leftarrow \mathcal{O}_{\mathrm{SRDH}}([1], [\omega^{-1} y])$
        Else
            $[y] \leftarrow [z]$
    Return $i$
```

```
Algorithm Square($[1], [x], \omega$):
    $[y] \leftarrow \mathcal{O}_{\mathrm{SRDH}}([1], [x])$
    If $[y] = [0]$ /*$\mathcal{O}_{\mathrm{SRDH}}$ returned QNR*/
        $[y] \leftarrow \mathcal{O}_{\mathrm{SRDH}}([1], [\omega y])$
    $h \leftarrow \frac{q+1}{2}$
    While $h > 2$:
        If $h$ is even
            $[z] \leftarrow \mathcal{O}_{\mathrm{SRDH}}([1], [y])$
            If $[z] = [0]$ /*$\mathcal{O}_{\mathrm{SRDH}}$ returned QNR*/
                $[y] \leftarrow \mathcal{O}_{\mathrm{SRDH}}([1], [\omega y])$
            Else
                $[y] \leftarrow [z]$
            $h \leftarrow \frac{h}{2}$
        Else /*If h is odd*/
            $[z] \leftarrow \mathcal{O}_{\mathrm{SRDH}}([x], [y])$
            If $[z] = [0]$ /*$\mathcal{O}_{\mathrm{SRDH}}$ returned QNR*/
                $[y] \leftarrow \mathcal{O}_{\mathrm{SRDH}}([x], [\omega y])$
            Else
                $[y] \leftarrow [z]$
            $h \leftarrow \frac{h+1}{2}$
    $i \leftarrow$ FindExpi($[1], [x], \omega$)
    $j \leftarrow$ FindExpi($[1], [y], \omega$)
    Return $[\omega^{2i-j} y]$
```

Fig. 5: Algorithms FindExpi and Square used in proof of Theorem 8

element where the bits have been shifted right and a new most significant bit i_t has been added. Repeating t times allows us to learn all of $i = i_{t-1} \dots i_0$.

Next we describe the Square algorithm. The idea behind this algorithm is that given $[\omega^i \beta]$ we want to compute $[\omega^j \beta^2]$ for some j, but we do not care much about the root of unity part, i.e., what j is, since we can always determine that by calling FindExpi and clean it up later. As a first step one of the inputs $([1],[x])$ or $([1],[\omega x])$ will correspond to a quadratic residue and the square root oracle will return some $[y]=[\omega^j \beta^{\frac{1}{2}}]=[\omega^j \beta^{\frac{q+1}{2}}]$. Let's define $h=\frac{q+1}{2}$, which is a positive integer, so we have $[y]=[\omega^j \beta^h]$.

The idea now is that we will use repeated applications of the SRDH oracle to halve h until we get down to $h=2$. If h is even this works fine. One of the pairs $([1],[\omega^j \beta^h])$ or $([1],[\omega^{j+1}\beta^h])$ will correspond to a quadratic residue and we get a new element of the form $[\omega^{j'}\beta^{h'}]$, where $h'=\frac{h}{2}$.

If h is odd this strategy does not work directly. However, when h is odd we can use $[x]$ as generator instead of $[1]$. One of the pairs $([\omega^i\beta],[\omega^j\beta^h])$ and $([\omega^i\beta],[\omega^{j+1}\beta^h])$ will be a quadratic residue and applying the square-root oracle we get a new element of the form $[\omega^{j'}\beta^{h'}]$, where $h'=\frac{h+1}{2}$.

Repeated application of these two types of calls, depending on whether h is even or odd, eventually gives us an element of the form $[\omega^j\beta^2]$. At this stage we can use the FindExpi algorithm to determine i and j, which makes it easy to compute $[x^2]=\omega^{2i-j}[\omega^j\beta^2]$.

Let's now analyse the time complexity of the algorithms. Algorithm FindExpi makes at most $2t$ oracle calls. Algorithm Square makes at most $2|q|$ oracle calls in addition to two invocations of FindExpi, i.e., at most $4t+2|q|$ oracle calls in total.

□

Using an Adversarial $\mathcal{O}^*{}_{\textbf{1-GDHE}}$ Oracle. In Theorem 8, we assumed the reduction had a perfect $\mathcal{O}_{\mathrm{SRDH}}$ oracle. Here we weaken the assumption used in the reduction and consider an $\mathcal{O}^*{}_{\text{1-GDHE}}$ oracle that returns a correct answer with a non-negligible probability ϵ when queried on a quadratic residue element. More precisely, let $([a],[b])\in\mathbb{G}_p^2$ be the input we are about to query the $\mathcal{O}^*{}_{\text{1-GDHE}}$ oracle on. Since we can easily detect if $b=0$, we can assume that we never need to query to oracle on any input where $b=0$. When queried on $([a],[b])\in\mathbb{G}_p^\times\times\mathbb{G}_p^\times$, the oracle will return either the symbol QNR, i.e. $[0]\in\mathbb{G}_p$ or $[c]\in\mathbb{G}_p$ for some $c\in\mathbb{Z}_p^\times$. Our assumption about correctness is

$$\Pr\Big[(\mathbb{G}_p,[1])\leftarrow\mathbb{G}(1^\kappa);a,z\leftarrow\mathbb{Z}_p^\times:\mathcal{O}^*{}_{\text{1-GDHE}}([a],[az^2])=\pm[az]\Big]\geq\epsilon(\kappa).$$

The oracle can behave arbitrarily when it does not return a correct answer or when the input is not a quadratic residue.

We will now show that we can rectify the adversarial behaviour of the oracle so that it cannot adapt its answer based on the instance input. The idea is to randomize the inputs to be queried to the oracle so that they are uniformly distributed over the input space and we get ϵ chance of getting a correct square-root when the input is a quadratic residue. To check the solution, we then randomize an element related to the answer, which we can use to detect when the oracle is misbehaving. The result is a Monte Carlo algorithm described in Fig. 6, which with probability $\epsilon'\geq\epsilon^2$ returns a correct square-root when queried on a quadratic residue and $\perp$ in all other cases.

```
1 Algorithm MOracle($[a], [b]$):
2     Sample $\alpha, \beta, \gamma, \delta, r, s \leftarrow \mathbb{Z}_p^\times$
3     $[y] \leftarrow \mathcal{O}^*{}_{\text{1-GDHE}}(\alpha[a], \alpha\beta^2[b])$
4     $[z] \leftarrow \mathcal{O}^*{}_{\text{1-GDHE}}(\gamma[a], \gamma\delta^2(r^2[a] + 2rs\frac{1}{\alpha\beta}[y] + s^2[b]))$
5     If $\frac{1}{\gamma\delta}[z] = \pm(r[a] + \frac{s}{\alpha\beta}[y])$
6         Return $\frac{1}{\alpha\beta}[y]$
7     Else
8         Return $\perp$
```

Fig. 6: Monte Carlo Algorithm using $\mathcal{O}^*{}_{\text{1-GDHE}}$

Lemma 3. *Using $\mathcal{O}^*{}_{1\text{-}GDHE}$ which returns a correct answer with probability ϵ, algorithm* MOracle *returns a correct answer with probability $\epsilon' \geq \epsilon^2$ when queried on a well-formed pair $([a], [b]) \in \mathbb{G}_p^\times \times \mathbb{G}_p^\times$ and $\perp$ otherwise.*

Proof. If $\frac{b}{a} \in \mathbb{QR}(p)$, we also have $\frac{\beta^2 b}{a} \in \mathbb{QR}(p)$, and when $\frac{b}{a} \notin \mathbb{QR}(p)$, we also have $\frac{\beta^2 b}{a} \notin \mathbb{QR}(p)$. We have probability ϵ that the answer $[y]$ is a correct answer when $\frac{b}{a} \in \mathbb{QR}(p)$ in which case $y = \pm\alpha\beta a\sqrt{\frac{b}{a}}$, we can thus recover $[\pm a\sqrt{\frac{b}{a}} = \pm\sqrt{ab}]$ by computing $\frac{1}{\alpha\beta}[y]$. Let $y' = \frac{y}{\alpha\beta} = \pm\sqrt{ab}$. We have probability ϵ that $[z]$ is a correct answer. Now let $z' = \frac{z}{\gamma\delta}$.

Note that $r^2a + 2rsy' + s^2b = (ra + sy')^2 + s^2(b - y'^2)$ and if $[z]$ is a correct answer then we have $z' = \pm(ra + sy')$. Thus, we have probability at least ϵ^2 that algorithm MOracle will return a correct square root when the input is well-formed.

We now argue that if $y' \neq \pm\sqrt{ab}$, with overwhelming probability the algorithm will return $\perp$. Let $\tau = a(r^2a + 2rsy' + s^2b) = r^2a^2 + 2arsy' + s^2ab$.

Since $a(r^2a + 2rsy' + s^2b) = (ra + sy')^2 + s^2(ab - y'^2)$, the query to the oracle is determined by a and τ, and there are roughly p^2 pairs (r, s) mapping into a maximum of p choices of τ. Therefore, for the same oracle query there are many possible values s could have. Now if $y' \neq \pm\sqrt{ab}$, i.e. y is an incorrect answer, then the oracle has negligible chance of passing the test $z' = \pm(ra + sy')$ in line 5. If the test passes, then $z'^2 = (ra + sy')^2 = r^2a^2 + 2arsy' + s^2ab = \tau - s^2(ab - y'^2)$. Since s is information theoretically undetermined from a and τ, there is negligible chance over the choice of s that this equality holds unless $ab - y'^2 = 0$. □

Since ϵ is non-negligible (in the security parameter), there must be a constant $c > 0$ such that for infinitely many κ we have $\epsilon' \geq \kappa^{-c}$. We can use repetitions to boost the oracle to give the correct answer with overwhelming probability on these κ values, i.e., on quadratic residues it returns square-roots and on quadratic non-residues it returns $\perp$ or equivalently $[0]$. Chernoff-bounds ensure we only need $\frac{\kappa}{\epsilon^2}$ polynomially iterations of the Monte Carlo algorithm to build a good SRDH oracle.

4.3 The q-GDHE Family Structure

We say a family of assumptions $\{q\text{-}\mathsf{A}\}$ is a *strictly increasingly stronger* family if for all polynomials $q \leq q'$ it holds that $q'\text{-}\mathsf{A} \Rightarrow q\text{-}\mathsf{A}$ but $q\text{-}\mathsf{A} \not\Rightarrow (q'+1)\text{-}\mathsf{A}$.

We prove the following theorem regarding the structure of q-GDHE family.

Theorem 9. *The q-GDHE family is a strictly increasingly stronger family.*

Proof. The following two lemmeta prove the theorem.

Lemma 4. *For all polynomials q, we have $(q+1)$-GDHE$\Rightarrow$$q$-GDHE.*

Proof. Using an adversary $\mathcal{A}$ against the q-GDHE assumption in a black-box manner, we build an adversary $\mathcal{B}$ against the $(q+1)$-GDHE assumption. Adversary $\mathcal{B}$ who is tasked with outputting $[x^{q+1}] \in \mathbb{G}_p$, gets the tuple $([1], [x], \ldots, [x^q], [x^{q+2}], \ldots, [x^{2q+2}]) \in \mathbb{G}_p^{2q+2}$. She can easily test whether $x = 0$ using generic group operations, and if this is the case she returns $[0]$. If $x \neq 0$, she initiates adversary $\mathcal{A}$ on input $\left([x], \ldots, [x^q], \left[x^{q+2}\right], \ldots, \left[x^{2q+1}\right]\right) \in \mathbb{G}_p^{2q}$, i.e. the input tuple given to $\mathcal{A}$ is now w.r.t. the generator $[x]$ rather than $[1]$.

First we argue that since we are working with groups of prime order, $[x]$ is a valid group generator of $\mathbb{G}_p$ and hence the tuple given to $\mathcal{A}$ is a valid q-GDHE tuple. Eventually, when $\mathcal{A}$ halts with her answer $[y]$, $\mathcal{B}$ returns the same answer in her game. If $[y]$ is a valid solution to the q-GDHE problem w.r.t. group generator $[x] \in \mathbb{G}_p$ (on which $\mathcal{A}$ is challenged), it is also a valid solution to the $(q+1)$-GDHE problem w.r.t. group generator $[1] \in \mathbb{G}_p$ (on which $\mathcal{B}$ is challenged). Thus, $\mathcal{B}$ wins her game with at least the same advantage as adversary $\mathcal{A}$.

Lemma 5. *For all polynomials q, we have q-GDHE$\overset{\mathsf{GGM}}{\not\Rightarrow}$$(q+1)$-GDHE.*

Proof. Consider a generic adversary $\mathcal{A}$ against the q-GDHE assumption. Adversary $\mathcal{A}$ gets $\left([1], [x], \ldots, \left[x^{q-1}\right], \left[x^{q+1}\right], \ldots, \left[x^{2q}\right]\right)$ and is tasked with outputting $[x^q]$. We grant $\mathcal{A}$ access to an oracle $\mathcal{O}_{(q\,+\,1)\text{-GDHE}}(\cdot,\cdot,\cdot,\cdot)$ which on input a tuple $\left([a], [b = az], \left[c = az^{q+2}\right], \left[d = az^{q+3}\right]\right)$ returns $\left[az^{q+1}\right]$ if the input is well-formed and the special symbol $\perp$ otherwise, i.e. if the input tuple is not in the above form. Clearly, with the help of this oracle, it is easy to break the $(q+1)$-GDHE assumption.

Since the adversary is generic, the queries $([a], [b], [c], [d])$ she makes to the oracle must be constructed using generic group operations. Thus, the corresponding polynomials are linear combinations of the group elements $\left([1], [x], \ldots, \left[x^{q-1}\right], \left[x^{q+1}\right], \ldots, \left[x^{2q}\right]\right)$ and hence we have

$$a = \sum_{j=0, j \neq q}^{2q} \alpha_j x^j \qquad b = \sum_{j=0, j \neq q}^{2q} \beta_j x^j \qquad c = \sum_{j=0, j \neq q}^{2q} \gamma_j x^j \qquad d = \sum_{j=0, j \neq q}^{2q} \delta_j x^j$$

Let the corresponding formal polynomials be $a(X)$, $b(X)$, $c(X)$ and $d(X)$. We have that $\deg(a), \deg(b), \deg(c), \deg(d) \in \{0, \ldots, 2q\}$ and none of those polynomials have the monomial X^q.

In the generic group model this has negligible probability of holding unless we have we have $b(X) = a(X)z(X)$, $c(X) = a(X)z(X)^{q+2}$ and $d(X) = a(X)z(X)^{q+3}$ as formal polynomials for some (possibly rational) function $z(X)$.

If the adversary submits a query where $a(X) \equiv 0$, $b(X) \equiv 0$, the oracle will just return [0] which is useless to the adversary. So from now on let's assume that $a(X) \not\equiv 0$, $b(X) \not\equiv 0$, $c(X) \not\equiv 0$ and $d(X) \not\equiv 0$.

If the input tuple is well-formed, we have $z(X) = \frac{b(X)}{a(X)}$, where $a(X)$ and $b(X)$ are both proper polynomials, and $c(X) = a(X)z(X)^{q+2} = a(X)\left(\frac{b(X)}{a(X)}\right)^{q+2} = a(X)\frac{b(X)}{a(X)}\left(\frac{b(X)}{a(X)}\right)^{q+1}$ which corresponds to a proper polynomial. We also have $d(X) = a(X)\left(\frac{b(X)}{a(X)}\right)^{q+3} = a(X)\left(\frac{b(X)}{a(X)}\right)^{2}\left(\frac{b(X)}{a(X)}\right)^{q+1}$ which also corresponds to a proper polynomial.

From the above, it is clear that $a(X)^{q+1}|b(X)^{q+2}$ and $a(X)^{q+2}|b(X)^{q+3}$ and $\deg(c), \deg(d) \in \{0, \ldots, 2q\}$ and neither of those polynomials has the term X^q.

Let $t(X) = a(X)\left(\frac{b(X)}{a(X)}\right)^{q+1}$. We have that $\frac{b(X)}{a(X)}t(X)$ is a proper polynomial and so is $\left(\frac{b(X)}{a(X)}\right)^{2}t(X)$ which would not hold if $a(X) \nmid b(X)$ as in the latter case we can write $b(X) = s(X) + \frac{r(X)}{a(X)}$ for some polynomials $r(X)$ and $s(X)$ where $\deg(r) < \deg(a)$ and by substituting $b(X)$, it is easy to see that $c(X)$ and $d(X)$ cannot be both proper polynomials if $a(X) \nmid b(X)$ and hence we must have $a(X)|b(X)$.

The element returned by the oracle corresponds to $\frac{c(X)}{z(X)}$. It follows that $a(X)^q|b(X)^{q+1}$ and hence $\frac{c(X)}{z(X)}$ does not contain the monomial X^q.

Thus, it follows that the element returned by the oracle could have been computed by the adversary herself without calling the oracle.

□

5 Target Assumptions over Bilinear Groups

We now turn our attention to prime order bilinear groups. Our reductions from the cyclic group setting translate into a bilinear framework that captures existing computational bilinear assumptions where the adversary's task is to compute a specific group element in the base groups or the target group. For instance, the bilinear variants of the matrix computational Diffie-Hellman assumption in [MRV16] (which implies the (computational bilinear) k-linear assumptions), the (bilinear) q-SDH assumptions, and the bilinear assumptions studied in [JR13] are all examples of target assumptions in bilinear groups.

Definition 8 (Bilinear Group Generator). *A bilinear group generator is a PPT algorithm $\mathcal{BG}$, which on input a security parameter κ (given in unary) outputs bilinear group parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2)$, where*

- *$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p with bitlength $|p| = \Theta(\kappa)$.*
- *$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ have polynomial-time algorithms for carrying out group operations and unique representations for group elements.*
- *There is an efficiently computable bilinear map (pairing) $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.*
- *G_1 and G_2 are independently chosen uniformly random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e(G_1, G_2)$ generates $\mathbb{G}_T$.*

Again, we will be working in the low/medium granularity setting so we always assume uniformly random generators of the base groups.

According to [GPS08], bilinear groups of prime order can be classified into 3 main types depending on the existence of efficiently computable isomorphisms between the groups. In Type-1, $\mathbb{G}_1 = \mathbb{G}_2$. In Type-2 $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ that is efficiently computable in one direction, whereas in Type-3 no efficient isomorphism between the groups in either direction exists. Type-3 bilinear groups are the most efficient and hence practically relevant and we therefore restrict our focus to this type, although much of this section also applies to Type-1 and Type-2 bilinear groups.

We will use $[x]_1, [y]_2, [z]_T$ to denote the group elements in the respective groups and as in the previous sections use additive notation for all group operations. This means the generators are $[1]_1, [1]_2$ and $e([1]_1, [1]_2) = [1]_T$. We will often denote the pairing with multiplicative notation, i.e., $[x]_1 \cdot [y]_2 = [xy]_T$.

A bilinear group generator can be seen as a particular example of a cyclic group generator generating $\mathbb{G}_1, \mathbb{G}_2$ or $\mathbb{G}_T$. All our results regarding non-interactive computational assumptions therefore still apply in the respective groups but in this section we will also cover the case where exponents are shared between the groups. The presence of the pairing e makes it possible for elements in the base groups $\mathbb{G}_1, \mathbb{G}_2$ to combine in the target group $\mathbb{G}_T$, so we can formulate assumptions that involve several groups, e.g., that given $[1]_1, [1]_2, [x]_T$ it is hard to compute $[x]_1$. In the following sections, we define and analyze non-interactive target assumptions in the bilinear group setting.

5.1 Target Assumptions in Bilinear Groups

We now define and analyze target assumptions, where the adversary's goal is to compute a group element in $\mathbb{G}_j$, where $j \in \{1, 2, T\}$. When we defined the cyclic group target assumption, we gave the adversary group elements of the form $\left[\frac{a(\boldsymbol{x})}{b(\boldsymbol{x})}\right]$. In the bilinear group setting, the adversary may get a mix of group elements in all three groups. We note that if the adversary has $\left[\frac{a^{(1)}(\boldsymbol{x})}{b^{(1)}(\boldsymbol{x})}\right]_1$ and $\left[\frac{a^{(2)}(\boldsymbol{x})}{b^{(2)}(\boldsymbol{x})}\right]_2$ she can obtain $\left[\frac{a^{(1)}(\boldsymbol{x})a^{(2)}(\boldsymbol{x})}{b^{(1)}(\boldsymbol{x})b^{(2)}(\boldsymbol{x})}\right]_T$ via the pairing operation. When we define target assumptions in bilinear groups, we will therefore without loss of generality assume the fractional polynomials the instance generator outputs for the target group $\mathbb{G}_T$ include all products $\frac{a_i^{(1)}(\boldsymbol{X})a_j^{(2)}(\boldsymbol{X})}{b_i^{(1)}(\boldsymbol{X})b_j^{(2)}(\boldsymbol{X})}$ of fractional polynomials for elements in the base groups.

Definition 9 (Bilinear Target Assumption in $\mathbb{G}_j$). *Given polynomials $d(\kappa)$, $m(\kappa)$, $n_1(\kappa)$, $n_2(\kappa)$, and $n_T(\kappa)$ we say $(\mathcal{I}, \mathcal{V})$ is a (d, m, n_1, n_2, n_T)-bilinear target assumption in $\mathbb{G}_j$ for $\mathcal{BG}$ if it works as follows:*

$(pub, priv) \leftarrow \mathcal{I}(1^\kappa)$**:** *There is a PPT algorithm $\mathcal{I}^{core}$ defining $\mathcal{I}$ as follows:*

1. $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [1]_1, [1]_2) \leftarrow \mathcal{BG}(1^\kappa)$; $\mathsf{bgp} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$
2. $\left(\left\{\frac{a_i^{(1)}(\boldsymbol{X})}{b_i^{(1)}(\boldsymbol{X})}\right\}_{i=1}^{n_1}, \left\{\frac{a_i^{(2)}(\boldsymbol{X})}{b_i^{(2)}(\boldsymbol{X})}\right\}_{i=1}^{n_2}, \left\{\frac{a_i^{(T)}(\boldsymbol{X})}{b_i^{(T)}(\boldsymbol{X})}\right\}_{i=1}^{n_T}, pub', priv'\right) \leftarrow \mathcal{I}^{core}(\mathsf{bgp})$
3. $\boldsymbol{x} \leftarrow \mathbb{Z}_p^m$ *conditioned on* $b_i^{(j)}(\boldsymbol{x}) \neq 0$ *for all choices of* i *and* j
4. $pub := \left(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \left\{\left\{\left[\frac{a_i^{(j)}(\boldsymbol{x})}{b_i^{(j)}(\boldsymbol{x})}\right]_j\right\}_{i=1}^{n_j}, \left\{\frac{a_i^{(j)}(\boldsymbol{X})}{b_i^{(j)}(\boldsymbol{X})}\right\}_{i=1}^{n_j}\right\}_{j=1,2,T}, pub'\right)$
5. *Return* $(pub, priv := ([1]_1, [1]_2, \boldsymbol{x}, priv'))$

$b \leftarrow \mathcal{V}\left(pub, priv, sol = \left(r(\boldsymbol{X}), s(\boldsymbol{X}), [y]_j, sol'\right)\right)$**:** *There is a DPT algorithm $\mathcal{V}^{core}$ such that $\mathcal{V}$ returns 1 if all of the following checks pass and 0 otherwise:*

1. $r(\boldsymbol{X}) \prod_{i=1}^{n_j} b_i^{(j)}(\boldsymbol{X}) \notin \operatorname{span}\left(\left\{s(\boldsymbol{X})a_i^{(j)}(\boldsymbol{X}) \prod_{\ell \neq i} b_\ell^{(j)}(\boldsymbol{X})\right\}_{i=1}^{n_j}\right)$
2. $[y]_j = \frac{r(\boldsymbol{x})}{s(\boldsymbol{x})} [1]_j$
3. $\mathcal{V}^{core}(pub, priv, sol) = 1$

We require that the number of variables in $\boldsymbol{X}$ is $m(\kappa)$, the total degrees of the polynomials are bounded by $d(\kappa)$, and that all products of polynomial fractions in $\mathbb{G}_1$ and $\mathbb{G}_2$ are included in the polynomial fractions in $\mathbb{G}_T$.

Also, since the pairing function allows one to obtain the product of any two polynomials from the opposite source groups in the target group, for assumptions where the required target element is in $\mathbb{G}_T$, the degree of the polynomials $r(\boldsymbol{X})$ and $s(\boldsymbol{X})$ the adversary specifies is upper bounded by $2d$ instead of d.

Similarly to the cyclic group case, we can reduce any bilinear target assumption to a simple bilinear target assumption, where all $b_i^{(1)}(\boldsymbol{X}) = b_i^{(2)}(\boldsymbol{X}) = b_i^{(T)}(\boldsymbol{X}) = 1$. Also, we can reduce bilinear target assumptions with multivariate polynomials to bilinear target assumptions with univariate polynomials.

We can then consider two cases depending on whether or not the adversary's polynomial $s(X)$ divides $r(X)$. Just as in the cyclic group case, we get that all bilinear target assumptions can be reduced to the following two assumptions.

Definition 10 (Bilinear Polynomial Assumption in $\mathbb{G}_j$). *We say a $(d, 1, n_1, n_2, n_T)$-simple bilinear target assumption $(\mathcal{I}, \mathcal{V})$ in $\mathbb{G}_j$ for $\mathcal{BG}$ is a (d, n_1, n_2, n_T)-bilinear polynomial assumption in $\mathbb{G}_j$ if $\mathcal{V}$ only accepts solutions where $s(X) = 1$.*

Definition 11 (Bilinear Fractional Assumption in $\mathbb{G}_j$ (BFrac$_j$)). *We say a $(d, 1, n_1, n_2, n_T)$-simple bilinear target assumption $(\mathcal{I}, \mathcal{V})$ in $\mathbb{G}_j$ for $\mathcal{BG}$ is a (d, n_1, n_2, n_T)-bilinear fractional assumption in $\mathbb{G}_j$ if $\mathcal{V}$ only accepts solutions where $s(X) \nmid r(X)$.*

It is straightforward to prove the following theorem, which states that the (d, n_1, n_2, n_T)-BFrac$_i$ assumption can be further simplified to only consider the target group case.

Theorem 10. *If the (d, n_1, n_2, n_T)-BFrac$_T$ assumption over $\mathcal{BG}$ holds, then the assumptions (d, n_1, n_2, n_T)-BFrac$_1$ and (d, n_1, n_2, n_T)-BFrac$_2$ over $\mathcal{BG}$ also hold.*

5.2 Bilinear Target Assumptions in the Base Groups

All the results in this section are assuming a Type-3 bilinear group. Intuitively, bilinear target assumptions in $\mathbb{G}_1$ and $\mathbb{G}_2$ are very similar to the cyclic group case because the generic computations one can do in a base group are not affected by group elements in the other groups. We will now formalize this intuition by generalizing the q-GDHE and the q-SFrac assumptions to the bilinear setting.

Definition 12 (q-Bilinear GDHE Assumption in $\mathbb{G}_1$ (q-BGDHE$_1$)). *The q-BGDHE assumption in $\mathbb{G}_1$ over $\mathcal{BG}$ is that for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [1]_1, [1]_2) \leftarrow \mathcal{BG}(1^\kappa); x \leftarrow \mathbb{Z}_p : \\ [x^q]_1 \leftarrow \mathcal{A}\left(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \left[1, x, \ldots, x^{q-1}, x^{q+1}, \ldots, x^{2q}\right]_1, \left[1, x, \ldots, x^{2q}\right]_2\right)\end{array}\right] \approx 0.$$

The q-BGDHE assumption in $\mathbb{G}_2$ (q-BGDHE$_2$) over $\mathcal{BG}$ is defined similarly.

Definition 13 (q-Bilinear SFrac in $\mathbb{G}_i$ Assumption (q-BSFrac$_i$)). *The q-BSFrac$_i$ for $i \in \{1, 2, T\}$ over $\mathcal{BG}$ is that for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [1]_1, [1]_2) \leftarrow \mathcal{BG}(1^\kappa); x \leftarrow \mathbb{Z}_p; \\ \left(r(X), s(X), [y]_i\right) \leftarrow \mathcal{A}\left(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}, [1, x, \ldots, x^q]_1, [1, x, \ldots, x^q]_2\right) : \\ q \geq \deg(s) > \deg(r) \geq 0 \text{ and } [y]_i = \left[\frac{r(x)}{s(x)}\right]_i\end{array}\right] \approx 0.$$

Similarly to Theorem 4, we can prove that all bilinear polynomial assumptions in the base group $\mathbb{G}_j$ for $j \in \{1, 2\}$ are implied by the q-BGDHE$_j$ assumption. Also, similarly to Theorem 5, we can show that all bilinear fractional assumptions in the base group $\mathbb{G}_j$ for $j \in \{1, 2\}$ are implied by the q-BSFrac$_j$ assumption. For bilinear target assumptions in the base groups, we therefore get a situation very similar to the single cyclic group case, which we express in the following theorem:

Theorem 11. *For $j \in \{1, 2\}$ there is a polynomial $q(d, m, n_1, n_2, n_T)$ such that the joint q-BSFrac$_j$ and q-BGDHE$_j$ assumption imply all (d, m, n_1, n_2, n_T) bilinear target assumptions in $\mathbb{G}_j$.*

The theorem means that the joint q-BGDHE$_j$ and q-BSFrac$_j$ assumption serves as an Uber assumption for all bilinear target assumptions in the base group $\mathbb{G}_j$.

5.3 Bilinear Target Assumptions in the Target Group

We now consider bilinear target assumptions in the target group $\mathbb{G}_T$. Unlike the case of bilinear target assumptions in the base groups where computations in the rest of the groups do not affect the concerned base group, here it is a different story as computations in any of the 3 groups can be mapped into the target group.

Consider a polynomial assumption. We can use matrices $\mathbf{A}$ and $\mathbf{B}$ to represent the polynomials in $\mathbb{G}_1$ and $\mathbb{G}_2$ as $[1, X, \ldots, X^d]_1\mathbf{A}$ and $[X^d, \ldots, X, 1]_2\mathbf{B}$. Given group elements with evaluations of these polynomials one may use the pairing to compute products of the polynomial evaluations in the target group $\mathbb{G}_T$. Observe that $\mathbf{A^T B}$ represents the coefficients of the X^d terms in all the possible pairings of two elements in $\mathbb{G}_1$ and $\mathbb{G}_2$. We will use this to formulate a target group analogue of the q-BGDHE assumptions.

Definition 14 (q-Bilinear Gap Assumption (q-BGap)). *We say the q-BGap assumption holds when for all PPT $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [1]_1, [1]_2) \leftarrow \mathcal{BG}(1^\kappa); (\mathbf{A}, \mathbf{B}) \leftarrow \mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T);\ x \leftarrow \mathbb{Z}_p \\ : \quad [x^q]_T \leftarrow \mathcal{A}\begin{pmatrix} [1, x, \ldots, x^q]_1 \mathbf{A}, [x^q, \ldots, 1]_2 \mathbf{B}, \\ [1, x, \ldots, x^{q-1}]_T, [x^{q+1}, \ldots, x^{2q}]_{\{1,2,T\}} \end{pmatrix} \\ \qquad \textit{and } \mathbf{A}^T\mathbf{B} = \mathbf{0} \textit{ and } \textit{rank}(\mathbf{A}) + \textit{rank}(\mathbf{B}) = q + 1 \end{array}\right] \approx 0 \cdot$$

Let $\mathbf{A} \in \mathbb{Z}_p^{(q+1)\times n_1}$ and $\mathbf{B} \in \mathbb{Z}_p^{(q+1)\times n_2}$ where without loss of generality we can assume their ranks are n_1 and n_2, respectively, and $n_1 + n_2 = q + 1$. Note that for smaller dimensions we can always add a column in the null space of $\mathbf{A^T}$ to $\mathbf{B}$ and still have $\mathbf{A^T B} = \mathbf{0}$.

Theorem 12. *For any (d, n_1, n_2, n_T)-polynomial assumption $\mathsf{A} = (\mathcal{I}_\mathsf{A}, \mathcal{V}_\mathsf{A})$ in $\mathbb{G}_T$ over $\mathcal{BG}$, we have that $(3d + 1)$-BGap$\Rightarrow\mathsf{A}$.*

Proof. Let $\mathcal{A}$ be an adversary against the (d, n_1, n_2, n_T)-polynomial assumption. We build an adversary $\mathcal{B}$ against the $(3d + 1)$-BGap assumption which uses $\mathcal{A}$ in a black-box manner. Adversary $\mathcal{B}$ runs $\mathcal{I}_\mathsf{A}^{\text{core}}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ to get the tuple $\left(\left\{\left\{a_i^{(j)}(X)\right\}_{i=1}^{n_j}\right\}_{j=1,2,T}, pub'_\mathsf{A}, priv'_\mathsf{A}\right)$. Now, $\mathcal{B}$ randomly chooses a polynomial $c(X) \leftarrow \mathbb{Z}_p[X]$ where $\deg(c) \leq 2d+1$ conditioned on all $a_i^{(T)}(X)c(X)X^d$ having coefficient 0 in the term for X^{3d+1}.

Write the polynomials as

$$a_i'^{(1)}(X) = a_i^{(1)}(X)c(X) = \sum_{j=0}^{3d+1} \alpha_{i,j} X^j$$

$$a_i'^{(2)}(X) = a_i^{(2)}(X)X^d = \sum_{j=0}^{2d} \beta_{i,j} X^j$$

$$a_i'^{(T)}(X) = a_i^{(T)}(X)c(X)X^d = \sum_{j=0}^{5d+1} \gamma_{i,j} X^j$$

and define the matrices $\mathbf{A} \in \mathbb{Z}_p^{(3d+2)\times n_1}$ and $\mathbf{B} \in \mathbb{Z}_p^{(3d+2)\times(3d+2-n_1)}$ as follows:

$$\mathbf{A} = \begin{pmatrix} \alpha_{1,0} & \cdots & \alpha_{n_1,0} \\ \vdots & & \vdots \\ \alpha_{1,3d+1} & \cdots & \alpha_{n_1,3d+1} \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} \beta_{1,3d+1} & \cdots & \beta_{n_2,3d+1} & \cdots \\ \vdots & & \vdots & \vdots\, \beta'_{i,j}\, \vdots \\ \beta_{1,0} & \cdots & \beta_{n_2,0} & \cdots \end{pmatrix},$$

where the values $\beta'_{i,j} \in \mathbb{Z}_p$ are chosen as an arbitrary extension to give us $\text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B}) = 3d + 2$ in a way such that we still have $\mathbf{A}^T\mathbf{B} = 0$. Adversary $\mathcal{B}$ now forwards $\mathbf{A}, \mathbf{B}$ to her environment and gets back the tuple $\left[1, x, \ldots, x^{3d+1}\right]_1 \mathbf{A}$, $\left[x^{3d+1}, \ldots, x, 1\right]_2 \mathbf{B}$, and $\left[1, x, \ldots, x^{3d}\right]_T$, $\left[x^{3d+2}, \ldots, x^{6d+2}\right]_{\{1,2,T\}}$ using which she can compute all of the group elements $[a_i'^{(1)}(x)]_1$, $[a_i'^{(2)}(x)]_2$, and $[a_i'^{(T)}(x)]_T$. Adversary $\mathcal{B}$ now starts $\mathcal{A}$ on the tuple

$$pub := \left(\left\{ \left\{ \left[a_i'^{(j)}(\boldsymbol{x}) \right]_j \right\}_{i=1}^{n_j}, \left\{ a_i^{(j)}(\boldsymbol{X}) \right\}_{i=1}^{n_j}, \right\}_{j=1,2,T}, pub'_{\mathsf{A}} \right)$$

and gets $(r(X), [y]_T, sol')$ as an answer, where $r(X) \notin \text{span}\left(\left\{ a_i^{(T)}(X) \right\}_{i=1}^{n_T} \right)$. Parse $r(X)c(X)X^d$ as $\sum_{i=0}^{5d+1} \gamma_i X^i$. Using the tuple available to her, $\mathcal{B}$ can recover $[x^{3d+1}]_T$ by computing $\frac{1}{\gamma_{3d+1}}([y]_T - [\sum_{i \neq 3d+1} \gamma_i x^i]_T)$.

Assume $c(x) \neq 0$, which happens with probability $1 - \frac{2d+1}{p}$, and $x^d \neq 0$, which happens with probability $1 - \frac{d}{p}$. The instance $\mathcal{A}$ sees is indistinguishable from an instance she gets directly from the instance generator with generators $[c(x)]_1$, $[x^d]_2$ and $[c(x)x^d]_T$. By Lemma 2, the probability that $r(X)c(X)X^d$ has a 0 coefficient of X^{3d+1} is $\frac{1}{p}$. Thus, if $\mathcal{A}$ has probability $\epsilon_{\mathcal{A}}$ against assumption A, we have $\epsilon_{\mathcal{B}} = \epsilon_{\mathcal{A}} - \frac{3d+2}{p}$ is the probability of $\mathcal{B}$ against the $(3d + 1)$-BGap assumption.

□

We conclude our analysis of target assumptions in $\mathbb{G}_T$ with the following theorem.

Theorem 13. *There is a polynomial $q(d, m, n_1, n_2, n_T)$ such that the joint q-BSFrac$_T$ and q-BGap assumption imply all (d, m, n_1, n_2, n_T) bilinear target assumptions in $\mathbb{G}_T$.*

References

[ABP15] M. Abdalla, F. Benhamouda, and A. Passelègue. *An algebraic framework for pseudorandom functions and applications to related-key security.* In *Advances in Cryptology – CRYPTO 2015*, vol. 9215 of *Lecture Notes in Computer Science*. Springer, 2015.

[ABS16] M. Ambrona, G. Barthe, and B. Schmidt. *Automated unbounded analysis of cryptographic constructions in the generic group model.* In *Advances in Cryptology – EUROCRYPT 2016*, vol. 9666 of *Lecture Notes in Computer Science*. Springer, 2016.

[BB08] D. Boneh and X. Boyen. *Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology*, 21(2):149, 2008.

[BBG05] D. Boneh, X. Boyen, and E.-J. Goh. *Hierarchical identity based encryption with constant size ciphertext.* In *Advances in Cryptology – EUROCRYPT 2005*, vol. 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

[BCP02] E. Bresson, O. Chevassut, and D. Pointcheval. *The Group Diffie-Hellman Problems.* In *Selected Areas in Cryptography*, vol. 2595 of *Lecture Notes in Computer Science*. Springer, 2002.

[BDZ03] F. Bao, R. Deng, and H. Zhu. *Variations of Diffie-Hellman problem.* In *Information and Communications Security*, vol. 2836 of *Lecture Notes in Computer Science*. Springer, 2003.

[BFF$^+$14] G. Barthe, E. Fagerholm, D. Fiore, J. Mitchell, A. Scedrov, and B. Schmidt. *Automated analysis of cryptographic assumptions in generic group models.* In *Advances in Cryptology – CRYPTO 2014*, vol. 8616 of *Lecture Notes in Computer Science*. Springer, 2014.

[BGW05] D. Boneh, C. Gentry, and B. Waters. *Collusion resistant broadcast encryption with short ciphertexts and private keys.* In *Advances in Cryptology – CRYPTO 2005*, vol. 3621 of *Lecture Notes in Computer Science*. Springer, 2005.

[BLMW07] E. Bresson, Y. Lakhnech, L. Mazaré, and B. Warinschi. *A generalization of DDH with applications to protocol analysis and computational soundness.* In *Advances in Cryptology – CRYPTO 2007*, vol. 4622 of *Lecture Notes in Computer Science*. Springer, 2007.

[Boe88] B. Boer. *Diffie-Hellman is as strong as discrete log for certain primes.* In *Advances in Cryptology – CRYPTO 1988*, vol. 403 of *Lecture Notes in Computer Science*. Springer, 1988.

[Boy08] X. Boyen. *The uber-assumption family.* In *Pairing-Based Cryptography – Pairing 2008*, vol. 5209 of *Lecture Notes in Computer Science*. Springer, 2008.

[BP04] M. Bellare and A. Palacio. *The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols.* In *Advances in Cryptology – CRYPTO 2004*, vol. 3152. Springer, 2004.

[BW07] X. Boyen and B. Waters. *Full-domain subgroup hiding and constant-size group signatures.* In *Public Key Cryptography – PKC 2007*, vol. 4450 of *Lecture Notes in Computer Science*. Springer, 2007.

[Che06] J. Cheon. *Security analysis of the strong Diffie-Hellman problem.* In *Advances in Cryptology – EUROCRYPT 2006*, vol. 4004 of *Lecture Notes in Computer Science*. Springer, 2006.

[CM14] M. Chase and S. Meiklejohn. *Déjà Q: using dual systems to revisit q-type assumptions.* In *Advances in Cryptology – EUROCRYPT 2014*, vol. 8441 of *Lecture Notes in Computer Science*. Springer, 2014.

[CMM16] M. Chase, M. Maller, and S. Meiklejohn. *Déjà Q all over again: Tighter and broader reductions of q-type assumptions.* In *Advances in Cryptology – ASIACRYPT 2016*, vol. 10032 of *Lecture Notes in Computer Science*. Springer, 2016.

[Den02] A. W. Dent. *Adapting the weaknesses of the random oracle model to the generic group model.* In *Advances in Cryptology – ASIACRYPT 2002*, vol. 2501 of *Lecture Notes in Computer Science*. Springer, 2002.

[EHK$^+$13] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. *An algebraic framework for Diffie-Hellman assumptions.* In *Advances in Cryptology –*

CRYPTO 2013, vol. 8043 of *Lecture Notes in Computer Science*. Springer, 2013.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. *Quadratic span programs and succinct nizks without pcps.* In *EUROCRYPT*, vol. 7881 of *Lecture Notes in Computer Science*. 2013.

[GK16] S. Goldwasser and Y. T. Kalai. *Cryptographic assumptions: A position paper.* In *Theory of Cryptography - TCC 2016-A*, vol. 9562. Springer, 2016.

[GPS08] S. D. Galbraith, K. G. Paterson, and N. P. Smart. *Pairings for cryptographers. Discrete Applied Mathematics*, 156(16):3113, 2008.

[JR13] A. Joux and A. Rojat. *Security ranking among assumptions within the Uber assumption framework.* In *Information Security*, vol. 7807 of *Lecture Notes in Computer Science*. Springer International Publishing, 2013.

[JS12] T. Jager and J. Schwenk. *On the analysis of cryptographic assumptions in the generic ring model. Journal of Cryptology*, 26(2):225, 2012.

[JY09] D. Jao and K. Yoshida. *Boneh-Boyen signatures and the strong Diffie-Hellman problem.* In *Pairing-Based Cryptography – Pairing 2009*. Springer, Berlin, Heidelberg, 2009.

[Kil01] E. Kiltz. *A tool box of cryptographic functions related to the Diffie-Hellman function.* In *Progress in Cryptology – INDOCRYPT 2001*, vol. 2247 of *Lecture Notes in Computer Science*. Springer, 2001.

[KM07] N. Koblitz and A. Menezes. *Another look at generic groups. Advances in Mathematics of Communications*, 1(1):13, 2007.

[KMS04] C. Konoma, M. Mambo, and H. Shizuya. *Complexity analysis of the cryptographic primitive problems through square-root exponent. IEICE TRANSACTIONS*, E87-A(5):1083, 2004.

[Mau94] U. M. Maurer. *Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms.* In *Advances in Cryptology – CRYPTO 1994*. Springer, Berlin, Heidelberg, 1994.

[Mau05] U. Maurer. *Abstract models of computation in cryptography.* In *IMA Cryptography and Coding*, vol. 3796. Springer, 2005.

[MRV16] P. Morillo, C. Ràfols, and J. L. Villar. *The kernel matrix Diffie-Hellman assumption.* In *Advances in Cryptology – ASIACRYPT 2016*, vol. 10031. Springer Berlin Heidelberg, 2016.

[MSK02] S. Mitsunari, R. Sakai, and M. Kasahara. *A new traitor tracing. IEICE TRANSACTIONS*, E85-A(2):481, 2002.

[MW96] U. M. Maurer and S. Wolf. *Diffie-Hellman oracles.* In *Advances in Cryptology – CRYPTO*. Springer, 1996.

[Nao03] M. Naor. *On cryptographic assumptions and challenges.* In *Advances in Cryptology – CRYPTO*, vol. 2729. Springer, 2003.

[Nec94] V. I. Nechaev. *Complexity of a determinate algorithm for the discrete logarithm. Mat. Zametki*, 55(2):91, 1994.

[RH12] D. Roh and S. G. Hahn. *The square root Diffie-Hellman problem. Designs, Codes and Cryptography*, 62(2):179, 2012.

[Sho97] V. Shoup. *Lower bounds for discrete logarithms and related problems.* In *EUROCRYPT*, vol. 1233 of *Lecture Notes in Computer Science*. 1997.

[SS01] A. Sadeghi and M. Steiner. *Assumptions related to discrete logarithms: Why subtleties make a real difference.* In *Advances in Cryptology – EUROCRYPT*, vol. 2045 of *Lecture Notes in Computer Science*. Springer, 2001.

[ZSNS04] F. Zhang, R. Safavi-Naini, and W. Susilo. *An efficient signature scheme from bilinear pairings and its applications.* In *Public Key Cryptography – PKC 2004*, vol. 2947 of *Lecture Notes in Computer Science*. Springer, 2004.

A Generic Intractability of the q-SFrac Assumption

Theorem 14. *The q-SFrac assumption holds in the generic group model.*

Proof. The adversary gets the tuple $([1], \ldots, [x^q])$ and if successful she outputs a tuple $(r(X), s(X), [\frac{r(x)}{s(x)}] \in \mathbb{G}_p)$ satisfying $0 \leq \deg(r) < \deg(s) \leq q$. Let $t(X) = \frac{r(X)}{s(X)}$. Since the adversary is generic, she has negligible chance of success unless she computes her output group element as a known linear compbination of her inputs, i.e., $[t(x)]$ for a known polynomial $t(X)$. If the answer of the adversary is correct, we have

$$t(x)s(x) = r(x) \tag{1}$$

First we argue that (1) does not hold identically for formal polynomials, i.e., $t(X)s(X) \neq r(X)$. The group operation queries the adversary is allowed in the game do not result in any polynomials whose degree is not in $\{0, \ldots, q\}$ since they correspond to polynomial additions/subtractions, so $t(X)$ has degree at most q. By the definition of the assumption, we have $0 \leq \deg(r) < \deg(s) \leq q$ thus equality $t(X)s(X) \equiv r(X)$ will only hold if $\deg(t) < 0$ or $\deg(ts) = p - 1$ and $\deg(r) = 0$. Since by definition q is polynomial in the security parameter κ whereas $\log p \in \Theta(\kappa)$, we have that both cases do not happen.

We now bound the probability that the simulation fails. At the end of the game, we have that the number of group elements the adversary has seen is $\ell \leq q + 1 + q_{\mathbb{G}_p}$ where $q_{\mathbb{G}_p}$ is the number of group operation queries made by the adversary. The probability that any two different polynomials corresponding to these group elements evaluate to the same value at a random point $x \leftarrow \mathbb{Z}_p$ is $\epsilon_1 \leq \binom{\ell}{2}\frac{q}{p}$. Also, the probability that (1) holds at a random point $x \leftarrow \mathbb{Z}_p$ is $\epsilon_2 \leq \frac{2q}{p}$. Thus, it follows that

$$\epsilon = \epsilon_1 + \epsilon_2 \leq \frac{\ell^2 q}{p}.$$

From which it follows that the advantage of the generic adversary against the assumption is $\epsilon = O(\frac{q^3 + q^2_{\mathbb{G}_p} q}{p})$. □