Limits on the Power of Prime-Order Groups: Separating Q-Type from Static Assumptions

George Lu
UT Austin
gclu@cs.utexas.edu

Mark Zhandry
NTT Research
mzhandry@gmail.com

Abstract

Subgroup decision techniques on cryptographic groups and pairings have been critical for numerous applications. Originally conceived in the composite-order setting, there is a large body of work showing how to instantiate subgroup decision techniques in the prime-order setting as well. In this work, we demonstrate the first barrier to this research program, by demonstrating an important setting where composite-order techniques cannot be replicated in the prime-order setting.

In particular, we focus on the case of q-type assumptions, which are ubiquitous in group- and pairing-based cryptography, but unfortunately are less desirable than the more well-understood static assumptions. Subgroup decision techniques have had great success in removing q-type assumptions, even allowing q-type assumptions to be generically based on static assumptions on composite-order groups. Our main result shows that the same likely does *not* hold in the prime order setting. Namely, we show that a large class of q-type assumptions, including the security definition of a number of cryptosystems, cannot be proven secure in a black box way from any static assumption.

1 Introduction

Cryptographic groups and pairings have been some of the most impactful tools for both the theory and practice of cryptography. A cryptographic group $\mathbb{G}$ is a (cyclic) mathematical group with efficient group operations and some hardness, typically at least the hardness of the discrete logarithm problem: computing x from $g, g^x \in \mathbb{G}$. Sometimes these groups come with extra structure, such as pairings where the group additionally has a bilinear map into a different group.

Composite-order groups. Composite-order groups, and especially composite-order pairings, have been very useful extensions of the original prime-order setting. Originally proposed by [BGN05], composite-order groups/pairings have found many applications, from homomorphic encryption [BGN05] to identity-based encryption [LW10, LW11], to adaptive security [Wat09a], and much more.

The key feature of a composite-order group is that it allows for subgroup hiding: the factors of the group order are assumed to be unknown, creating hidden subgroups of the group $\mathbb{G}$. This approach allows for terms to be undetectably moved between the hidden subgroups.

Unfortunately, composite-order groups come with a significant cost, namely the need to have the factors of the group order hidden. This is problematic due to sub-exponential-time factoring algorithms [BLP92].

In order to account for these attacks, parameter sizes must be significantly blown up. Compounded with the fact that pairings are already expensive operations, pairings over composite-order groups are extremely slow. Finding elliptic curves supporting composite-order pairings is also a non-trivial task. More abstractly, factoring is an assumption seemingly unrelated to the "usual" hardness of groups and pairings, where the intuition comes from the hardness of discrete logarithms. In particular, it is consistent with current knowledge that factoring is actually *easy*, while typical assumptions in prime-order groups could remain hard on, say, elliptic curves based groups. Thus, assuming composite-order groups can be thought of as making two very different assumptions.

Mapping composite-order to prime groups. Due to the issues just mentioned, there has been a significant effort made to try to simulate subgroup hiding techniques in prime-order groups [CS03, Gjø04, GS08, OT09, Wat09b, Fre10, Lew12, LM15, SC12]. The basic idea is that the common DDH assumption is equivalent to saying that it is hard to decide if (g^b, g^c) lies in the cyclic group generated by (g, g^a). Thus, DDH implies subgroup hiding in the larger group $\mathbb{G}' = \mathbb{G}^2$. This observation has led to several informal claims that the composite order groups and this simulation using prime-order groups are functionally equivalent[1]. This claim is backed up by numerous examples of where subgroup decision techniques have been successfully carried out in the prime-order setting.

The case of q-type assumptions. While the discrete logarithm problem is essentially always assumed in cryptographic groups and pairings, usually discrete logarithms alone are not enough, and instead much stronger assumptions must be made. Numerous such security assumptions have been made on groups and pairings. In this work, we will be considering two main types:

- Static Assumptions: These are assumptions where the number of group elements in the assumption is fixed, independent of various aspects of the adversary or scheme, such as how many queries the adversary makes during the security experiment. The canonical example of a static assumption is the decisional Diffie-Hellman assumption (DDH), which assumes that (g, g^a, g^b, g^{ab}) is computationally indistinguishable from (g, g^a, g^b, g^c), where a, b, c are independent random elements. Subgroup hiding in composite-order groups is another example.

- q-Type Assumptions: These are assumptions where the number of group elements in the assumption is variable, depending on q. In typical applications, q will be a function of the number of queries the adversary makes to the experiment. For example, the q-Strong decisional Diffie-Hellman assumption assumes that given $g, g^x, g^{x^2}, \cdots, g^{x^q}, g^{x^{q+1}}$ is indistinguishable from a random group element. Other q-type assumptions may often include other terms that depend on x as well as other variables.

The community has a strong preference for static assumptions, with q-type assumptions being much less desirable. This preference for static assumptions arises from several considerations:

- Underlying assumptions cannot be proven unconditionally, given the current status of complexity theory. As such, making an assumption requires a leap of faith, and the fewer assumptions used the better. q-type assumptions are viewed as families of individual assumptions, one for each value of q. As such, assuming a q-type assumption amounts to making infinitely many assumptions, whereas a static assumption is simply one assumption.

[1] [Fre10]: "functionality that can be achieved in composite-order groups under the subgroup decision assumption can also be achieved in prime-order groups under either the DDH or the decision linear assumption." [GS08]: "Assumption [sic.] in composite-order groups, and SXDH can typically be exchanged for one another."

- As q grows, q-type assumptions typically become "less secure" in terms of concrete attack performance [Che06], and become "stronger" (that is, worse) assumptions in terms of reducibility between instances. In order to account for this degradation of the assumption, larger parameter sizes must be used, weakening performance. As q usually depends on the number of queries an adversary makes, q can be quite large.

A major area of study, therefore, has been to eliminate q-type assumptions from cryptographic protocols. It turns out that composite-order groups/subgroup hiding has been very useful for this goal. In [CM14] and follow-up works [CMM16, Wee16], this approach was even adapted into a very useful framework called "Deja Q" for proving q-type assumptions themselves from static assumptions. In particular, by making constant-sized assumptions on composite-order groups, typical q-type assumptions can often be proven in those groups, meaning that schemes based on these q-type assumptions can automatically be based on static assumptions in composite-order groups.

Missing pieces. Despite the wide-spread ability of prime-order variants of subgroup decision to simulate the composite-order version, there are some crucial missing pieces.

- Most notably, the Deja Q framework has never been successfully translated to the prime order setting. In particular, there is no known way to prove the hardness of typical q-type assumptions from any constant-sized assumption over prime-order groups, in contrast to the composite-order case.

- In terms of specific applications, protocols like [BGW05, Del07] for broadcast encryption that rely on q-type assumptions have no equivalent realization in the prime-order setting over a constant-sized assumption. Broadcast encryption protocols that can be proven on prime-order pairings with constant sized assumptions such as [Wee21] have much larger secret keys. Secret keys in [BGW05] are constant-sized.

This work. In this work, we ask whether there is something inherent to the missing pieces outlined above, or whether composite order subgroup-techniques can indeed be always translated to the prime-order setting. In particular, can typical q-type assumptions be proved hard in prime-order groups under fixed-sized assumptions? More generally, can composite-order techniques always be translated to the prime-order setting?

Perhaps surprisingly based on the above discussion, we find that the answer is probably "no." We show a black-box separation between typical q-type assumptions and *every* fixed size assumption on prime-order groups. Our separation can be interpreted as showing the impossibility of a reduction between the assumptions. This result stands in stark-contrast to the composite case, where all mentioned reductions above are black box. Thus, our impossibility shows that the existing intuition regarding subgroup decision in composite vs prime order groups is false in full generality, and there are certain proof techniques that only work in the composite-order setting. Deja Q mentioned above is one example. Our impossibility is quite general, and also applies to many protocols such as [BGW05, Del07] to show that they cannot be based on fixed-sized assumptions.

Along the way, we formalize what it means to be a black-box reduction in a generic group model. We explain that the existing formalizations of generic group models are inadequate for reasoning about reductions. We therefore provide a new formalization that we believe captures the essence of a generic group model reduction.

1.1 Technical Overview

Black Box Reductions in Groups. We consider a family of "black box" reduction in groups. These reductions are black box in two ways:

- First, the reductions must, informally, be independent of the group representation. This means, roughly, that the reduction can make queries to the group operations, but otherwise cannot depend in any way on how the group works.
- Second, the reduction must be black box in terms of the adversary. This means that the reduction must work for *any* adversary, even one that is inefficient. Moreover, the reduction can only make black box use of the adversary, querying it on inputs and observing the outputs. The reduction itself must be efficient, even though the adversary need not be.

This corresponds to the notion of 'fully black box' reductions [IR89, RTV04] for the setting of groups. In Section 1.3, we give a brief overview of why the existing formalism for generic group model reductions is inadequate; we take care of these issues in Section 3. But for the purposes of this high-level overview of our main result, we can put these details aside.

Our main result. Our main result is the following:

Theorem 1.1 (Informal)**.** *There is no black box reduction between any q-type assumption and any true constant-sized assumptions on prime-order groups.*

Here, a q-type assumption is, informally, one where the adversary gets (among other elements) the terms $g^w, g^{wx}, \cdots, g^{wx^q}$ for presumably unknown w, x. Moreover, we informally require that x is somehow tied to the hardness of the assumption, so that if the adversary can find x, then the adversary can break the assumption.

For concreteness, we will assume that the generator g is fixed. Note that uniform vs fixed g can change the nature of the assumption, as explored by [BMZ19]; in our case, we can randomize the generator by raising to w, so this distinction is irrelevant for us.

We now explain how we prove this result, which follows the meta-reduction paradigm [BV98]. Let p be the order of $\mathbb{G}$, and let g be the fixed generator. Consider some fixed-size assumption F which can be used to prove our q-type assumption. This means F provides n group elements $(h_1 = g^{v_1}, \cdots, h_n = g^{v_n})$, as well as potentially some non-group-based information a. Since F is fixed-size, n is a constant.

Now suppose that there is a black box reduction between F and a q-type powers assumption Q. This means that there exists a reduction algorithm R such that, for any potential q and any potential adversary A for Q, $R^{A,\mathbb{G}}$ breaks assumption F. Here, the notation $R^{A,\mathbb{G}}$ indicates that R makes black box use of $\mathbb{G}$ and A (which in turn makes queries to $\mathbb{G}$). Importantly, R must exist for *any* adversary A, even ones that are potentially inefficient (both in computation time and query complexity).

We now claim that if such a reduction exists, then assumption F is actually false. Thus, there is no way to actually justify Q with a black box reduction from any *true* F. Note that we cannot rule out reductions with false assumptions, since a false assumption gives a contradiction, and any contradiction can be used to prove *any* statement.

To prove this claim, let $q > n$. Note that if the *only* terms given out by Q are $g^w, g^{wx}, \cdots, g^{wx^q}$, then the assumption is actually an assumption of size $q + 1$. Moreover, if $w = 1$, then the g^w term is just g, and so the assumption has size only q. This means the assumption can be proven from an assumption of

size $n = q$, namely the assumption itself, via trivial reduction. Therefore, assuming $q > n$ is necessary in general for a lower-bound.

We now design our adversary A. On input $g^{\alpha_0}, \cdots, g^{\alpha_q}$, A performs a brute-force search to recover $\alpha_0, \cdots, \alpha_q$. It then sees if there is a w and x such that each $\alpha_i = wx^i$. If this check fails, A outputs $\perp$. If the check passes, we know that x must be unique (since in particular, $x = \alpha_i/\alpha_{i-1}$). In this case A uses x to efficiently break the assumption Q (recall that we assumed Q to be broken if x is known).

Now consider a supposed black box reduction R which uses A to break F. Note that $R^{A,\mathbb{G}}$ breaks F, but is inefficient due to A needing to brute force search for x. We will use R to design an efficient algorithm M which breaks F efficiently. Algorithm M runs R, which in turn makes queries to A, but we show that M can efficiently answer all queries R makes to A without needing to actually run the inefficient algorithm A.

Consider running R until it makes a query to A. Denote this query by $(g^{\alpha_0}, \cdots, g^{\alpha_q})$ for unknown quantities α_i. Recall that R makes black box use of the group, which means that the only way it can create new group elements is through making queries to the group, and we can observe all such queries. The input elements to R consist of g, together with the challenge provided by F, namely $h_1 = g^{v_1}, \cdots, h_n = g^{v_n}$. For notational consistency, we will let $h_0 = g = g^{v_0}$ where $v_0 = 1$. By tracing the group operations R makes, we can trace each g^{α_i} to the input elements h_j, writing $g^{\alpha_i} = \prod_j h_j^{T_{i,j}}$ for some known values $T_{i,j}$. We can re-write this in matrix-vector notation as $\alpha = \mathrm{T} \cdot \mathrm{v}$, where T is known, but α and v are as of yet unknown.

Our goal is to use knowledge of T to simulate the action of the adversary A, but to do so efficiently. But how can we do this, given that A computes α via brute force, but we do not know anything about the actual values of α and cannot brute force their values due to being efficient?

Our key observation is that, because $q > n$, the matrix $\mathrm{T} \in \mathbb{Z}_p^{(q+1)\times(n+1)}$ has only rank at most $n+1$, meaning the columns of T do not span the entire space of possible α. Thus, we do learn *something* about the vector α since it must be in the column-span of T. It turns out that this is enough information to respond to the query.

In particular, fix any set of $n+2$ distinct elements $x_0, \cdots, x_{n+1} \in \mathbb{Z}_p$. Then, consider the column vectors $(1, x_\ell, x_\ell^2, \cdots, x_\ell^q)$ as ℓ varies. These vectors form an $(n+2) \times (q+1)$ Vandermonde matrix, which must be full rank. Since $q \geq n+1$, the rank is therefore $n+2$, meaning the $n+2$ vectors are linearly independent. As such, these vectors span a subspace of dimension $n+2 > n+1$. This means that the vectors cannot all simultaneously be in the column-span of T, which only has rank (at most) $n+1$.

What we see, then, is that there are at most $n+1$ distinct values x such that the vector $(1, x, x^2, \cdots, x^q)$ is in the column-span of T. Moreover, we can find this set of x efficiently, as follows. Consider the matrix obtained by concatenating to T the column vector $(1, X, \cdots, X^q)$ for a formal variable X. Then, take the first $n+2$ rows, which forms a square matrix. Compute the determinant of this matrix, which becomes a degree $n+2$ polynomial in X. By solving this polynomial for X over the finite field $\mathbb{Z}_p$ [Ber71, Rab80, CZ81], we can find the $\leq n+1$ solutions for x.

Now that we have a list of candidates for x, we can simulate the brute-force step of A, which is the only inefficient part. We note that, as described above, A expects the actual element x, whereas we only get a list of candidates including x. However, in the setting considered above, we can narrow down the list of candidates to the actual value x. Indeed, for any x, we can test if the exponent vector α has the form $(w, wx, \cdots, wx^q)$ for some x, by checking if $g^{\alpha_{i+1}} = (g^{\alpha_i})^x$ for all i. There will be a unique x such that this is the case.

Thus, we can efficiently simulate any query that R makes to A. Since R itself is efficient, this means the overall running time of our simulation M is efficient, and M has the same input/output behavior as $R^{A,\mathbb{G}}$. In particular, since R is guaranteed to break F, so is M. Thus, F is actually insecure, since there is an

efficient attack.

In the body, we generalize the above discussion to consider both interactive constant-sized assumptions and interactive q-type assumptions, as well as to extend our notion of q-type assumptions to allow for general polynomials in x instead of just powers. We also extend the lower-bound to groups with a pairing, and also consider the case where a group G' that is built from a group G, and the q-type assumption lives in G' while the constant-size assumption lives in G.

1.2 Takeaway: Why Deja Q fails.

Given that Deja Q allows for proving certain q-type assumptions in composite-order groups, it is clear that our meta-reduction must fail in the composite order case. This leads to two questions: what is it about our meta-reduction that fails, and why is it that Deja Q is able to actually achieve a positive result in the composite order setting?

The first question is straightforward: in the step where we efficiently compute x, we needed the ability to find roots of polynomial equations over $\mathbb{Z}_p$. In the composite-order setting with p being a composite with unknown factors, finding such roots is as hard as factoring. The subgroup hiding assumption for composite-order groups in particular implies that factoring is hard. Thus, in the composite-order setting, this step is inefficient, which breaks the meta reduction.

For the second question, the key difference is that, for composite-order groups, it is possible to independently rerandomize all subgroups, which is not possible in the prime-order setting. In more detail: if the group-order is $N = pq$, the two subgroups have order p and q, respectively, and are generated by $g_q = g^p$ and $g_p = g^q$. Suppose we are given h, and we compute h^α for a uniform $\alpha \in \mathbb{Z}_N$. Using the Chinese Remainder Theorem, we can write $h = g_q^{x_q} \times g_p^{x_p}$ where $x_q \in \mathbb{Z}_q$ and $x_p \in \mathbb{Z}_p$. We can also write h^α as $g_q^{x_q\alpha_q} \times g_p^{x_p\alpha_p}$ where $(\alpha_p, \alpha_q) \in \mathbb{Z}_p \times \mathbb{Z}_q$ are in bijection with $\alpha \in \mathbb{Z}_N$. As a result, raising h to the α perfectly and independently rerandomizes both the $\mathbb{Z}_p$ component and $\mathbb{Z}_q$ component of h. Note that the rerandomization happens, even thought the subgroups themselves are hidden.

On the other hand, suppose $\mathbb{G}^2$ is separated into hidden subgroups generated by $g_1 = (g, g^a)$ and $g_2 = (g^{-a}, g)$ [2]. We can likewise write any $h \in \mathbb{G}^2$ as $g_1^{x_1} \times g_2^{x_2}$ where $\times$ is component-wise. However, there is no obvious way, without knowledge of the actual subgroups (that is, knowledge of a) to independently re-randomize both x_1 and x_2. Indeed, the only thing that seems reasonable to do is raise h to a random $\alpha \in \mathbb{Z}_p$. But this gives $h^\alpha = g_1^{x_1\alpha} \times g_2^{x_2\alpha}$. Observe that this fully rerandomizes x_1 and x_2 in isolation, but the results are still correlated. In particular, the ratio x_1/x_2 is preserved under rerandomization.

This independent randomization of each subgroup is crucially used in the Deja Q techniques. Namely, they use subgroup-hiding to place the q-type assumption terms into two subgroups. By rerandomization, these terms end up being independent instances of the q-type assumption. Then they use subgroup hiding again to add the terms from one subgroup to the other. After repeating this several times, the end result is that one has the sum of many independent instances of the q-type assumption, which by some statistical analysis ends up being just a vector of truly random group elements.

If we had the ability to independently rerandomize all subgroup in the prime-order setting, then we would be able to carry out the Deja Q arguments. Our lower bound shows that this is impossible, and thus there is no way to independently rerandomize all subgroups in the prime-order setting.

[2] The second subgroup is arbitrary for the purposes of this overview, but we chose the subgroup that is orthogonal to (g, g^a).

1.3 On Generic Group Models

Here, we briefly discuss how to formalize generic group models and reductions in them. See Section 3 for more details.

There are actually two generic group models in the literature, Shoup's [Sho97] and Maurer's [Mau05]. While both offer some formalization of algorithms that are "independent of the representation", they do so in different ways: Shoup's model assumes the group is given as a random embedding of the additive group $\mathbb{Z}_p$ into bit-strings, whereas Maurer's enforces a type system that prevents the adversary from interacting with the group except in the prescribed way. Zhandry [Zha22] re-interpreted these generic groups and gave them new names: Random Representation (RR) for Shoup's, and Type Safe (TS) for Maurer's. [Zha22] also explored the relationships between these group, finding that they are often equivalent but sometimes different.

A particular setting *not* considered by [Zha22] is that of reductions between assumptions. We observe that it is fairly simple to formalize a notion of reduction following the TS/Maurer style generic group that matches the intuition of a black box generic reduction. However, the obvious way of formalizing a black box RR/Shoup style generic reduction is a bit problematic. Indeed, such a reduction would say that, for any (potentially inefficient) algorithm A interacting with a RR generic group (that is, a random embedding of $\mathbb{Z}_p$) and solving some problem Q, there must exist a reduction algorithm R that makes queries to A and solving some problem F. The issue is that this reduction is only guaranteed to work on algorithms interacting with a random embedding, and the reduction may fail on any specific embedding. For example, maybe the reduction decides to fail if the string representation of the group element g^x has a certain form, say that it is a point on an elliptic curve. A random string is exponentially unlikely to lie on the given curve, so the reduction will succeed in the random representation model. Meanwhile, the reduction would fail when the group G is instantiated with the given elliptic curve. This seems to not capture what we actually want from a reduction, which is that it works *for all* groups.

Therefore, we re-imagine Shoup's model as a Generic Representation (GR) model. An adversary for some assumption in the GR model is one that makes group operation queries to the group, and works *for any realization of the group*, even inefficient ones. This means that the adversary must work both for efficient groups like elliptic curves, but also random representations like those in the RR/Shoup model. Unlike the TS/Maurer model, a GR adversary actually gets the bit representation of group elements, regardless of which group is being considered. In order to prove lower bounds in the GR model, one option is to simply prove a lower bound relative to a random representation; in other words, prove a RR/Shoup lower bound. But proofs do not need to rely on this random structure, either, and other options may be possible.

A black box reduction in the GR model is then a reduction that makes group operations queries to the group, and turns any GR model adversary into another GR model adversary by only making black box use of the adversary.

With this definition of a GR model in hand, we prove that, for "single-stage games", a GR model reduction is equivalent to an RR model reduction. Here, single-stage games are roughly any assumption that is defined by an interactive game between a challenger and a single adversary. This is in contrast to multi-stage games, where the challenger interacts with multiple non-communicating adversaries. Single-stage games capture almost all assumptions made in the literature, and more of the common cryptographic security definitions.

Note that [Zha22] showed that TS and RR *adversaries* for single-stage games are equivalent, which in turn is easily adapted to show that TS and GR adversaries are equivalent for single-stage games. But while a reduction acts as an adversary for a single-stage game, the reduction is also itself interacting with another adversary, and the success criteria for the reduction is a function of both the behavior of that other

adversary as well as the probability it wins its game. Therefore, in this sense, a reduction acts more as an adversary in a multi-stage game, as there is no simple game-based definition of when the reduction is valid. Nevertheless, we are able to show that TS and GR reductions are equivalent for single-stage games, despite adversaries for general multi-stage games not being equivalent.

1.4 On Algebraic Reductions

The algebraic group model (AGM) [FKL18] is another idealized model for groups that seen as lying "between" the standard and generic group models. This model allows the adversary full access to the group (rather than just query access), but requires that for every group element produced by an adversary, the adversary can "explain" that group element as a linear combination of the group elements seen so far. As clarified by [Zha22], this is true for the TS generic group model, but the AGM is actually incomparable to the RR model.

We observe that our separation also applies to reductions in the AGM. Namely, our separation used the fact that the reduction R was generic to write each query made by R as a linear combination its inputs. But this information is simply the "explanation" demanded by the AGM, meaning an algebraic reduction R must supply this information. Thus, even though algebraic reductions have non-black-box access to the group, our impossibility still rules them out.

2 Preliminaries

2.1 Notation

Here we will introduce the notation we will use later in the paper. We will use $x \xleftarrow{R} X$ to denote sampling a random variable x from a set X. For a set of polynomials $\mathcal{P} = \{p_i(x)\}$, $\deg(\mathcal{P})$ will refer to the maximum degree of any polynomial $p_i \in \mathcal{P}$. $\dim(\mathcal{P})$ will refer to the dimension of the polynomials $\{p_i(x)\}$ in the vector space $\mathbb{Z}_p[x]$. We will say quantity $t = t(\lambda) \gg t' = t'(\lambda)$ if $t - t'(\lambda)$ is nonnegligible.

Cryptographic Games. We will use the term cryptographic game and assumption interchangeably. A cryptographic game G is specified by a pair (Chal, t) where Chal is a probabilistic interactive algorithm, typically called a challenger, and $t : \mathbb{Z}^+ \to [0, 1]$ is a function. We will consider what are referred to as "single stage" games, where there is only a single adversary $\mathcal{A}$. Chal gets as input the security parameter λ, and then interacts with $\mathcal{A}$. At the end of the interaction, Chal outputs a bit b. The adversary wins if $b = 1$. The advantage of $\mathcal{A}$, denoted $\mathsf{Adv}_{\mathcal{A}}$, is a function of λ that is equal to the probability $\mathcal{A}$ wins, minus $t(\lambda)$. We say that (Chal, t) is "secure" for a class of algorithms if, for every $\mathcal{A}$ in the class, the advantage of $\mathcal{A}$ is negligible in λ. If we do not specify the class of algorithms, we take the algorithms to be those of polynomial running time.

Our notion of a game captures both assumptions as well as the security of cryptosystems.

Black Box Reductions. Let (Chal_0, t_0) and (Chal_1, t_1) be two games. Here, we formalize what it means to prove the security of (Chal_1, t_1) from the security of (Chal_0, t_0).

Specifically, a black box reduction from (Chal_0, t_0) to (Chal_1, t_1) assigns, to any potential adversary $\mathcal{A}$ with non-negligible advantage against (Chal_1, t_1), a probabilistic polynomial time oracle algorithm $\mathcal{R}$. Importantly, the reduction assigns an $\mathcal{R}$ to every possible adversary, even ones that are inefficient. $\mathcal{R}$ can make oracle queries to $\mathcal{A}$, but $\mathcal{R}$ itself is always efficient, even if $\mathcal{A}$ is not. Since $\mathcal{A}$ is potentially interactive, this takes care to formalize. We can think of an interactive $\mathcal{A}$ as a non-interactive procedure that takes as

input a pair (r, T) and produces a message m. Here, r are the random coins of $\mathcal{A}$, which we can assume are chosen at the beginning of the game, and T is the list of messages received so far by $\mathcal{A}$. m is then the next message produced by $\mathcal{A}$, which is a deterministic function of the messages received and the initial randomness r.

The interface $\mathcal{R}$ sees is the following. There is a private random oracle O, which $\mathcal{R}$ does not see. Instead, $\mathcal{R}$ can make queries of the form (i, T). In response, $\mathcal{R}$ receives $\mathcal{A}(O(i), T)$. We can think of i as indexing an instance of $\mathcal{A}$, with the private random coins of $\mathcal{A}$ being $O(i)$. Then $\mathcal{R}$ can carry out the interaction with $\mathcal{A}$ by repeatedly querying $\mathcal{A}(O(i), \cdot)$ on the various partial transcripts. Note that this formulation captures reductions that can rewind the adversary and run the adversary multiple times on the same randomness but different inputs. We denote the algorithm $\mathcal{R}$ making queries of this form by $\mathcal{R}^{\mathcal{A}}$.

The guarantee of a black-box reduction is that, for any (potentially inefficient) adversary $\mathcal{A}$ with non-negligible advantage against (Chal_1, t_1), then $\mathcal{R}^{\mathcal{A}}$ has non-negligible advantage against (Chal_0, t_0).

3 Generic Group Models

Here, we recall the two typical generic group models, modeled after the works of Maurer [Mau05] and Shoup [Sho97], respectively. We will ultimately show that, for the purposes of this work, the two models can be thought of as equivalent, though as explained by [Zha22] they are not equivalent in all contexts. Since they will be equivalent for us, after this section we will focus exclusively on Maurer's model.

3.1 The Type Safe (TS)/Maurer Model [Mau05]

Here, we define the Type Safe (TS) model, which is a reinterpretation of Maurer's [Mau05] due to Zhandry [Zha22]. This will be the main model used in this work. Later, we will show in a formal sense that the TS model is equivalent to the model we consider based on Shoup's model [Sho97].

TS Algorithms. Much of this paragraph is taken almost verbatim from [Zha22], except that we replace "element gates" with queries. Let $p \in \mathbb{Z}$ be a positive integer. A TS algorithm A will be given as a circuit. Unlike a standard binary circuit, the circuit for A will have the following features:

- There will be two kinds of wires, *bit* wires and *element* wires. Bit wires take values in $\{0, 1\}$, whereas element wires take values in $\mathbb{Z}_p \cup \{\perp\}$.
- There will be "bit gates" that map bits to bits. These gates *cannot* take element wires as input. Any universal gate set is allowed for the bit gates.
- Additionally, we allow circuits to make queries, and the inputs/outputs of these queries may include element wires:
  - **Labeling Query.** This takes as input $\lceil \log_2(p) \rceil$ bit wires, and interprets them as $x \in \mathbb{Z}_p$. The response to this query is an element wire, containing x. This element wire will be thought of as corresponding to the value g^x. If the input wires do not correspond to an $x \in \mathbb{Z}_p$, the output wire will contain $\perp$.
  - **Group Operation Query.** This takes as input $2 \times \lceil \log_2(p) \rceil$ bit wires and 2 element wires. The bit wires are interpreted as $a_1, a_2 \in \mathbb{Z}_p$. Let x_1, x_2 be the contents of the element wires. The query response is an element wire, set to $a_1 x_1 + a_2 x_2$. If any of the bit or element input wires do not correspond to elements of $\mathbb{Z}_p$, then the output wire is set to $\perp$.

– **Equality Query.** This takes as input two element wires, and replies with a bit wire. If the input wires both contain the same $x \in \mathbb{Z}_p$, the output wire is set to 1. In all other cases (including $\perp$ inputs) the output is 0.

Notation. Values encoded in bit wires will be denoted as x, and $[\![x]\!]$ will be used to denote values encoded in element wires.

We somewhat abuse notation, and for an algorithm A in the TS model, we will sometimes write $A^{\mathbb{G}_{\mathrm{TS}}}$. Our cost metric for circuits in the TS model will give each gate and query unit cost. Note that this departs somewhat from [Zha22], which only counts labeling and group operation queries. The cost metric of [Zha22] essentially counts query complexity. Our notion counts all operations, not just queries.

The notion above easily extends to interactive algorithms, where the incoming and outgoing wires can contain both element wires and bit wires.

Note that, for any TS algorithm $A^{\mathbb{G}_{\mathrm{TS}}}$, and any actual cryptographic group G, we can turn A into a "plain model" algorithm A^{G}. Let g be a fixed generator of G. We replace every element wire (input, output, and internal) in $A^{\mathbb{G}_{\mathrm{TS}}}$ containing x with collection of wires containing g^x. We can then replace the labeling, group operation, and equality queries in $A^{\mathbb{G}_{\mathrm{TS}}}$ with the actual implementations of these operations in G. We will somewhat abuse notation, and treat the algorithms $A^{\mathbb{G}_{\mathrm{TS}}}$ and A^{G} interchangeably.

We also allow a TS model algorithm $A^{\mathbb{G}_{\mathrm{TS}}}$ to interact with a non-TS model algorithm B. In this case, the actual interaction is between B and the plain model algorithm A^{G}, according to the above transformation on A.

Type Safe Games. A Type Safe Game (TS Game) is given by a pair $(C^{\mathbb{G}_{\mathrm{TS}}}, t)$, where $C^{\mathbb{G}_{\mathrm{TS}}}$ is a TS algorithm. For any cryptographic group G, a TS Game gives rise to a game defined over that group, given by (C^{G}, t).

Type Safe Adversaries. A Type Safe Adversary (TS Adversary) for a TS Game $(C^{\mathbb{G}_{\mathrm{TS}}}, t)$ is a TS algorithm $A^{\mathbb{G}_{\mathrm{TS}}}$ that interacts with $(C^{\mathbb{G}_{\mathrm{TS}}}, t)$. We say that a game $(C^{\mathbb{G}_{\mathrm{TS}}}, t)$ is "hard" in the TS model if, for all polynomial-time TS adversaries $A^{\mathbb{G}_{\mathrm{TS}}}$, the advantage of $A^{\mathbb{G}_{\mathrm{TS}}}$ interacting with $(C^{\mathbb{G}_{\mathrm{TS}}}, t)$ is negligible.

Type Safe Black Box Reductions. A Type Black Box Safe Reduction (TS-BB Reduction) from TS game $(C_0^{\mathbb{G}_{\mathrm{TS}}}, t_0)$ to TS game $(C_1^{\mathbb{G}_{\mathrm{TS}}}, t_1)$ is a black box reduction for TS algorithms. That is, for each (potentially inefficient) adversary $\mathcal{A}$ with non-negligible advantage ϵ_1 against $(C_1^{\mathbb{G}_{\mathrm{TS}}}, t_1)$, there exists a non-negligible function ϵ_0 and a polynomial-time algorithm $\mathcal{R}$ such that $\mathcal{R}^{\mathbb{G}_{\mathrm{TS}}, \mathcal{A}}$ has advantage ϵ_0 against $(C_0^{\mathbb{G}_{\mathrm{TS}}}, t_0)$. Note that, in the interaction between $\mathcal{R}$ and $\mathcal{A}$, we allow $\mathcal{R}$ to see and respond to the queries $\mathcal{A}$ makes to $\mathbb{G}_{\mathrm{TS}}$; $\mathcal{R}$ could choose to forward the queries to $\mathbb{G}_{\mathrm{TS}}$, but it could also answer them differently.

3.2 Generic Representation (GR)/Shoup Model [Sho97]

Here, we describe our Generic Representation (GR) model. This can be thought of as a reinterpretation of Shoup's model, which was called the Random Representation model by [Zha22]. The difference is that we remove any explicit mention of random representations as used in [Sho97] and [Zha22].

GR Algorithms. Let $p \in \mathbb{Z}$ be a positive integer, and $\ell \geq \log_2(p)$ be an integer. A GR algorithm A will be given as a circuit. Unlike a standard binary circuit which can make the following queries

- **Labeling Query.** This takes as input $\lceil \log_2(p) \rceil$ bits x, and outputs $L \in \{0,1\}^\ell$. Think of L as being equal to g^x.

- **Group Operation Query.** This takes as input $a_1, a_2, h_1, h_2 \in \{0,1\}^{2\times\lceil\log_2(p)\rceil+2\times\ell}$ bits, and replies with a string $L \in \{0,1\}^{\ell}$. We will think of h_1, h_2 being group elements, and $L = h_1^{a_1} \times h_2^{a_2}$.

We somewhat abuse notation, and for an algorithm A in the GR model, we will sometimes write $A^{\mathbb{G}_{\mathrm{GR}}}$. Our cost metric for circuits in the GR model will give each gate or query unit cost. The notion above easily extends to interactive algorithms.

Note that, for any GR algorithm $A^{\mathbb{G}_{\mathrm{GR}}}$, and any actual cryptographic group G whose elements can be represented as bits of length ℓ, we can turn A into a "plain model" algorithm A^{G}. To do so, simply answer each labeling query with the actual value g^x, and answer each group operation query with the actual value $h_1^{a_1} \times h_2^{a_2}$. We will somewhat abuse notation, and treat the algorithms $A^{\mathbb{G}_{\mathrm{TS}}}$ and A^{G} interchangeably.

We also allow a GR model algorithm $A^{\mathbb{G}_{\mathrm{GR}}}$ to interact with a non-GR model algorithm B. In this case, the actual interaction is between B and the plain model algorithm A^{G}, according to the above transformation on A.

Generic Representation Games. A Generic Representation Game (GR Game) is given by a pair $(C^{\mathbb{G}_{\mathrm{GR}}}, t)$, where $C^{\mathbb{G}_{\mathrm{GR}}}$ is a GR algorithm. For any cryptographic group G, a GR Game gives rise to a game defined over that group, given by (C^{G}, t). Note that in this work, we will not need to consider GR Games.

Generic Representation Adversaries. A Generic Representation Adversary (GR Adversary) for a GR Game $(C^{\mathbb{G}_{\mathrm{GR}}}, t)$ is a GR algorithm $A^{\mathbb{G}_{\mathrm{TS}}}$ that interacts with $(C^{\mathbb{G}_{\mathrm{GR}}}, t)$. The advantage of A is the infimum over all (potentially inefficient) groups G of the advantage of A^{G} interacting with (C^{G}, t), where A^{G} and C^{G} are the standard model versions of the algorithms A and C. We say that a game $(C^{\mathbb{G}_{\mathrm{GR}}}, t)$ is "hard" in the GR model if, for all polynomial-time GR adversaries $A^{\mathbb{G}_{\mathrm{GR}}}$, the advantage of $A^{\mathbb{G}_{\mathrm{GR}}}$ against $(C^{\mathbb{G}_{\mathrm{GR}}}, t)$ is at most negligibly more than t.

Generic Representation Black Box Reductions. A Generic Representation Box Safe Reduction (GR-BB Reduction) from GR game $(C_0^{\mathbb{G}_{\mathrm{GR}}}, t_0)$ to GR game $(C_1^{\mathbb{G}_{\mathrm{GR}}}, t_1)$ is a black box reduction for Generic Representation adversaries. That is, for every (potentially inefficient) GR algorithm $\mathcal{A}^{\mathbb{G}_{\mathrm{GR}}}$ for $(C_1^{\mathbb{G}_{\mathrm{GR}}}, t_1)$ with non-negligible advantage ϵ_1, there exists a non-negligible function ϵ_0 and a polynomial-time GR algorithm $\mathcal{R}$ such that $\mathcal{R}^{\mathbb{G}_{\mathrm{GR}},\mathcal{A}}$ has advantage ϵ_0 against $(C_0^{\mathbb{G}_{\mathrm{GR}}}, t_0)$. Note that, in the interaction between $\mathcal{R}$ and $\mathcal{A}$, we allow $\mathcal{R}$ to see and respond to the queries $\mathcal{A}$ makes to $\mathbb{G}_{\mathrm{GR}}$; $\mathcal{R}$ could choose to forward the queries to $\mathbb{G}_{\mathrm{GR}}$, but it could also answer them differently.

3.3 Relationship Between The Models

Here, we show that, for the purposes of this paper, the TS and GR models are equivalent. First, we observe that any TS algorithm $A^{\mathbb{G}_{\mathrm{TS}}}$ can be converted into a GR algorithm $A^{\mathbb{G}_{\mathrm{GR}}}$, by replacing labelling and group operations queries with the corresponding queries in $\mathbb{G}_{\mathrm{GR}}$, and implementing the equality check with an actual string equality subroutine.

There are two equivalences we will consider: first we will consider the equivalence of TS and GR adversaries, and then we will consider the equivalence of TS and GR reductions. Note that [Zha22] show the equivalence of TS and Shoup's model for single-stage games, and this equivalence readily extends to show the equivalence of TS adversaries and GR adversaries. However, [Zha22] does not apply to reductions, which is the main consideration in this work.

Equivalence of Models for Security. Let $(C^{\mathbb{G}_{\mathrm{TS}}}, t)$ be a TS game, which can be converted into a GR game $(C^{\mathbb{G}_{\mathrm{GR}}}, t)$ as described above.

Lemma 3.1 (Adapted from [Zha22])**.** *$(C^{\mathbb{G}_{\mathrm{TS}}}, t)$ is hard in the TS model if and only if $(C^{\mathbb{G}_{\mathrm{GR}}}, t)$ is hard in the GR model for sufficiently large ℓ.*

Proof. We only sketch the proof since it follows straightforwardly from [Zha22]. In one direction, consider a TS adversary $A^{\mathbb{G}_{\mathrm{TS}}}$. We can turn it into a GR adversary $A^{\mathbb{G}_{\mathrm{GR}}}$ as described above. $A^{\mathbb{G}_{\mathrm{GR}}}$ maintains the same advantage as $A^{\mathbb{G}_{\mathrm{TS}}}$.

In the other direction, let $\ell > \log_2(p) + \omega(\log \lambda)$. Consider a GR adversary $A^{\mathbb{G}_{\mathrm{GR}}}$, such that, for any potentially inefficient group G, A^{G} has non-negligible advantage against (C^{G}, t). We will let G be a *random* group that is isomorphic to $\mathbb{Z}_p$. That is, choose a random injection $L : \mathbb{Z}_p \to \{0,1\}^\ell$, and let G be the set of images of L, such that L is an isomorphism between $\mathbb{Z}_p$ and G. Then we know that A^{G} has non-negligible advantage against (C^{G}, t).

We construct an adversary $\mathcal{B}^{\mathbb{G}_{\mathrm{TS}}}$ against $(C^{\mathbb{G}_{\mathrm{TS}}}, t)$. $\mathcal{B}^{\mathbb{G}_{\mathrm{TS}}}$ creates a table T of pairs of element wires and random strings. It then runs A, except that every time A makes a labeling query on input x, $\mathcal{B}^{\mathbb{G}_{\mathrm{TS}}}$ makes a labeling query on x to obtain an element wire containing x. It then uses equality queries to see if there is already an element wire containing x in T. If so, it responds to $\mathcal{A}$ with the associated random string in $\{0,1\}^\ell$. Otherwise, it creates a new string and adds the pair consisting of the element wire containing x and the new random string to T.

For group operation queries, $\mathcal{B}^{\mathbb{G}_{\mathrm{TS}}}$ will look up to see if the two group elements appear as random strings in the table T. If so, it will take the two corresponding element wires, and apply the appropriate group operation query, to get a new output element wire. As with labeling queries, it then looks to see if that element wire already existed in T, and if so give $\mathcal{A}$ the corresponding random string. Otherwise, it creates a new random string and adds the pair consisting of the element wire and random string to T.

$\mathcal{B}^{\mathbb{G}_{\mathrm{TS}}}$ will perfectly simulate $\mathcal{A}$, as long as $\mathcal{A}^{\mathrm{G}}$ never queries on a string y that is in the image of the injection L, without first seeing y as the result of a labeling or group operation query. But since L is injective into a sparse range, this can only occur with negligible probability. Hence, the advantage of $\mathcal{B}^{\mathbb{G}_{\mathrm{TS}}}$ is negligibly close to the advantage of $\mathcal{A}^{\mathrm{G}}$. □

Observe that in the proof of Lemma 3.1, the algorithm $\mathcal{B}^{\mathbb{G}_{\mathrm{TS}}}$ has the form $\mathcal{W}^{\mathbb{G}_{\mathrm{TS}},\mathcal{A}}$, where $\mathcal{W}$ is a "wrapper" algorithm that is independent of $\mathcal{A}$. Here, the notation $\mathcal{W}^{\mathbb{G}_{\mathrm{TS}},\mathcal{A}}$ means that $\mathcal{W}$ is a TS algorithm that additionally interacts with the algorithm $\mathcal{A}$, intercepting not only all communication $\mathcal{A}$ makes with the game, but also all queries to $\mathbb{G}_{\mathrm{GR}}$. We can likewise view the other direction as designing a wrapper algorithm as well. This will be important in the next part.

Equivalence for Reductions. While the equivalence between adversaries follows straightforwardly from [Zha22], the case of reductions was not considered in [Zha22].

Let $(C_0^{\mathbb{G}_{\mathrm{TS}}}, t_0)$, $(C_1^{\mathbb{G}_{\mathrm{TS}}}, t_1)$ be two TS games, which can be converted into GR games $(C_0^{\mathbb{G}_{\mathrm{GR}}}, t_0)$, $(C_1^{\mathbb{G}_{\mathrm{GR}}}, t_1)$.

Lemma 3.2. *There is a TS-BB reduction from $(C_0^{\mathbb{G}_{\mathrm{TS}}}, t_0)$ to $(C_1^{\mathbb{G}_{\mathrm{TS}}}, t_1)$ if and only if there is a GR-BB reduction from $(C_0^{\mathbb{G}_{\mathrm{GR}}}, t_0)$ to $(C_1^{\mathbb{G}_{\mathrm{GR}}}, t_1)$.*

Proof. We prove one direction, the other being a straightforward adaptation of the same idea. Suppose there exists a TS-BB reduction from $(C_0^{\mathbb{G}_{\mathrm{TS}}}, t_0)$ to $(C_1^{\mathbb{G}_{\mathrm{TS}}}, t_1)$. We now construct a GR-BB reduction from $(C_0^{\mathbb{G}_{\mathrm{GR}}}, t_0)$ to $(C_1^{\mathbb{G}_{\mathrm{GR}}}, t_1)$. Let $\mathcal{A}^{\mathbb{G}_{\mathrm{GR}}}$ be a (potentially inefficient) GR algorithm against $(C_1^{\mathbb{G}_{\mathrm{GR}}}, t_1)$ with non-negligible advantage. By Lemma 3.1, we can turn $\mathcal{A}^{\mathbb{G}_{\mathrm{GR}}}$ into a (potentially inefficient) TS algorithm $\mathcal{B}^{\mathbb{G}_{\mathrm{TS}}}$

with non-negligible advantage against $(C_1^{\mathbb{G}_{TS}}, t_1)$. As observed above, $\mathcal{B}^{\mathbb{G}_{TS}}$ has the form $\mathcal{W}^{\mathbb{G}_{TS},\mathcal{A}}$, running $\mathcal{A}$ as a sub-routine and intercepting all messages sent by $\mathcal{A}$ as well as all queries $\mathcal{A}$ makes to $\mathbb{G}_{GR}$. We then construct the reduction $\mathbb{R}$ guaranteed by the assumption that there is a TS-BB reduction. We then have that $\mathbb{R}^{\mathbb{G}_{TS},\mathcal{W}}$ has non-negligible advantage against $(C_0^{\mathbb{G}_{TS}}, t_0)$. Here, $\mathbb{R}$ intercepts all messages sent by $\mathcal{W}$ as well as all queries $\mathcal{W}$ makes to $\mathbb{G}_{TS}$.

We next apply Lemma 3.1 again to turn $\mathbb{R}$ into an algorithm § which has non-negligible advantage against $(C_0^{\mathbb{G}_{GR}}, t_0)$. Observe again that in the proof of Lemma 3.1, we can write § as $\mathcal{V}^{\mathbb{G}_{GR},\mathbb{R}}$, where $\mathcal{V}$ interacts with $\mathbb{R}$ as a sub-routine, intercepting all messages as well as queries to $\mathbb{G}_{TS}$. Recall that $\mathbb{R}$ in turn interacts with $\mathcal{W}$ by intercepting all communication and queries to $\mathbb{G}_{TS}$, which in turn interacts with $\mathcal{A}$ by intercepting all communication and queries to $\mathbb{G}_{GR}$. We can combine everything outside of $\mathcal{A}$ into a single algorithm $\mathcal{T}^{\mathbb{G}_{GR},\mathcal{A}}$ which intercepts all communication from $\mathcal{A}$ as well as queries to $\mathbb{G}_{GR}$. This $\mathcal{T}$ is then the promised algorithm associated to $\mathcal{A}$ by the reduction. □

3.4 Pairings

We will also consider generalizing the generic group model to generic *bilinear* groups. In the case of the TS model, this will also include target group element wires (denoted $[\![x]\!]_T$), group operation and equality gates which operate on these as element wires, as well as a pairing gate

- **Pairing Gate.** This takes as input 2 element wires $[\![x_1]\!], [\![x_2]\!]$, and outputs the value $[\![x_1 \cdot x_2]\!]_T$ onto a target wire. If either of the inputs are $\perp$, the gate outputs $\perp$ as well.

We can similarly define a generic bilinear GR model. The equivalence between the models established in Lemmas 3.1 and 3.2 both carry over almost immediately to the bilinear setting.

We also observe that any algorithm in the ordinary type safe generic group model is also an algorithm in the the type safe generic bilinear group model. Likewise for the generic representation model..

3.5 On Groups Built From Other Groups

Here, we consider the possibility of building groups from other groups. For example, when viewing DDH as a subgroup decision assumption, the subgroup decision assumption holds in the group $\mathbb{G}^2$, while the DDH assumption that justifies it lies in the group $\mathbb{G}$. However, we argue that any "natural" group build from $\mathbb{G}$ must essentially be the group $\mathbb{G}^k$ for some integer k.

Let $\mathbb{G}'$ be a group built from a group $\mathbb{G}$. Suppose the group $\mathbb{G}'$ is build as follows: each element h^β in $\mathbb{G}'$ consists of a tuple $(g^{f(\beta)})$ where $f(\beta)$ is a function which takes as input an exponent β for $\mathbb{G}'$, and outputs a vector of exponents for $\mathbb{G}$. Here, the notation $g^{f(\beta)}$ means the vector of group elements obtained by raising g to each of the exponents in $f(\beta)$.

Suppose further that to multiply two elements $h^\beta = g^{f(\beta)}$ and $h^\gamma = g^{f(\gamma)}$, the group $\mathbb{G}'$ computes $h^{\beta+\gamma}$ as a linear combination of the the terms in $g^{f(\beta)}$, $g^{f(\gamma)}$, and g. In other words, we have that $f(\beta+\gamma) = A \cdot f(\beta) + B \cdot f(\gamma) + v$ for some known matrices A, B and known vector v.

We then first observe we can assume $A = B$, without loss of generality. This is because $A \cdot f(\beta) + B \cdot f(\gamma) + v = f(\beta+\gamma) = f(\gamma+\beta) = A \cdot f(\beta) + B \cdot f(\gamma) + v$, and therefore $A \cdot f(\beta) + B \cdot f(\gamma) = A' \cdot (f(\beta) + f(\gamma))$, where $A' = (A+B)/2$. Thus, we can always replace A, B with their mean A′.

Now that we have $f(\beta+\gamma) = A(f(\beta) + f(\gamma)) + v$, we explain that can also assume $A = I$. This is because $f(\beta+\gamma+\delta) = A(f(\beta+\gamma) + f(\delta)) + v = A^2 f(\beta) + A^2 f(\gamma) + Af(\delta) + Av + v$. But we also have that $f(\beta+\gamma+\delta) = A(f(\beta) + f(\gamma+\delta)) = Af(\beta) + A^2 f(\gamma) + A^2 f(\delta) + Av + v$. Subtracting gives that $(A - A^2)(f(\beta) - f(\delta)) = 0$. We break into three cases:

1. $\mathrm{A} = \mathrm{I}$. In this case, by shifting the exponents $\mathrm{f}(\beta)$ to $\mathrm{g}(\beta) = \mathrm{f}(\beta) + \mathrm{v}$, we then have that $\mathrm{g}(\beta + \gamma) = \mathrm{f}(\beta + \gamma) + \mathrm{v} = \mathrm{f}(\beta) + \mathrm{f}(\gamma) + 2\mathrm{v} = \mathrm{g}(\beta) + \mathrm{g}(\gamma)$. Now we have that $\mathbb{G}' = \mathbb{G}^k$.

2. $\mathrm{A} \neq \mathrm{I}$ but $\mathrm{A}^2 = \mathrm{A}$. In this case, A cannot have full rank. But since any $\mathrm{f}(\beta)$ can be written as $\mathrm{f}(\beta) = \mathrm{f}(\beta + 0) = \mathrm{A}(\mathrm{f}(\beta) + \mathrm{f}(0))$, we must have each $\mathrm{f}(\beta)$ is in the column-span of A. But then we can write some components of $\mathrm{f}(\beta)$ for any β as linear combinations of other components. In particular, there must exist a full-rank matrix C that is taller than it is wide, such that $\mathrm{f}(\beta) = \mathrm{C}\mathrm{f}'(\beta)$. In particular, C as the same column-span as A. Moreover, since C has full (column) rank, there must exist a matrix D that is wider than it is tall such that $\mathrm{D} \cdot \mathrm{C} = \mathrm{I}$.

   From such a group, we can construct a group $\mathbb{G}''$ where a group element $(h')^\beta$ is just $g^{\mathrm{f}'(\beta)}$. We then multiply elements by observing that $\mathrm{f}'(\beta + \gamma) = \mathrm{DAC}(\mathrm{f}'(\beta) + \mathrm{f}'(\gamma))$. Let $\mathrm{A}' = \mathrm{DAC}$. We observe that $(\mathrm{A}')^2 = \mathrm{A}'$. Thus, we've essentially shown that the group $\mathbb{G}'$ is equivalent to a group $\mathbb{G}''$ over a smaller group of elements. Either A' is full-rank, or we can repeat this process several times until we've arrived at an equivalent group $\mathbb{G}'''$ defined by a matrix A'' that is full rank but have $(\mathrm{A}'')^2 = \mathrm{A}''$. At this point, we are in Case 1.

3. $\mathrm{A}^2 \neq \mathrm{A}$. This means that there is a linear subspace that contains $f(\beta) - f(\gamma)$ for all β, γ. In particular, this means there is an affine space that contains $f(\beta)$ for all β. In other words, we can write $\mathrm{f}(\beta) = \mathrm{g}(\beta) + \mathrm{w}$, where $\mathrm{g}(\beta)$ lies in a subspace. By shifting the exponent $\mathrm{f}(\beta)$ by w, we can construct a different subgroup $\mathbb{G}''$ where the exponents are $\mathrm{g}(\beta)$. Then, just as in the previous case, we can shrink this subgroup down to a smaller group where the exponents are $\mathrm{g}'(\beta)$ and these do *not* lie in any subspace. Iterating several times, eventually we end up with a group that is in Case 1

Thus, we see that by performing affine transformations on the group, we can reduce to the case where the group is just $\mathbb{G}^k$. Thus, it seems that the *only* way to generically build a group out of $\mathbb{G}$ is to simply have the group be $\mathbb{G}^k$.

Note that we have not completely ruled out any other possible way of building the group $\mathbb{G}'$ from $\mathbb{G}$. We could, for example, have the elements of $\mathbb{G}'$ have bit-string parts that do not correspond to group elements. Then the way multiplication combines group elements could depend on the bit-string parts of the input. We could also perform various equality tests on the input elements, and base the multiplication operation on that. Finally, the group operation in $\mathbb{G}'$ could depend on the actual bit-string representation of the group elements in $\mathbb{G}$. However, it is not clear how to use any of these approaches to actually build a group that whose security is tied to the underlying group $\mathbb{G}$.

4 Types of Group-Based Assumptions

Here, we define both static and q-type assumptions. Our definitions will be very general, which makes our impossibility stronger.

Definition 4.1. We say an assumption $\mathrm{G} = (C, t)$ is of *fixed-size n* if the total number of group elements produced by C is $\leq n$ for any adversary.

In order to define q-type assumptions, we first give some additional notation. For an assumption $\mathrm{G} = (C, t)$, we can decompose the logic of C as follows. First, we can assume C samples all of its randomness at the beginning of the experiment. Then we can specify C by a collection of functions $\mathcal{F} = \{f(\cdot) : \{0,1\}^* \to \mathbb{Z}_p\}, \mathcal{H} = \{h(\cdot) : \{0,1\}^* \to \{0,1\}^*\}$, a success function B. Then the game defined by C works as follows:

- C samples randomness $\mathcal{X} \xleftarrow{R} \{0,1\}^*$
- For a polynomial number of rounds $i \in [R]$
  - Let T_i be the transcript of outputs given by $\mathcal{A}$ by round i
  - C sends $[\![f_{T_i}(\mathcal{X})]\!], h_{T_i}(\mathcal{X})$ to $\mathcal{A}$
  - $\mathcal{A}$ sends a response message $m_i = \{[\![m_i^0]\!], m_i^1\}$
- C outputs a bit $b = B(T_R, \mathcal{X})$.

We are now ready to define q-type assumptions.

Definition 4.2. We refer to a hardness assumption G as a q-type assumption if, as part of challenger randomness $\mathcal{X}$, there exists a $x_0 \in \mathbb{Z}_p$. In addition, for all executions $\{T_1, T_2, \ldots\}$ of G, there exists a function $f'(\mathcal{X})$, as well as a set of polynomials $S_{f'} = \{s_i(x_0)\}$ where:

1. $\deg(S_{f'}) \in \mathsf{poly}(\lambda)$
2. $\dim(S_{f'}) \geq q$
3. $s_i(x_0) \cdot f'(\mathcal{X}) = f_{T_j}(\mathcal{X})$ for some transcript T_j

Furthermore, there exists an (efficient) algorithm $\mathcal{A}_{\mathsf{Break}}$ such that in the following game:

- C samples randomness $\mathcal{X} \xleftarrow{R} \{0,1\}^*$, including $\vec{x} \xleftarrow{R} \mathbb{Z}_p^n$
- For a polynomial number of rounds $i \in [R]$
  - Let T_i be the transcript of outputs given by $\mathcal{A}_{\mathsf{Break}}$ by round i
  - C sends $[\![f_{T_i}(\mathcal{X})]\!], h_{T_i}(\mathcal{X})$ to $\mathcal{A}_{\mathsf{Break}}$
  - If $dim(S_{f'}) \geq q$ for the first time, send an x^* to $\mathcal{A}_{\mathsf{Break}}$
  - $\mathcal{A}_{\mathsf{Break}}$ sends a response message $m_i = \{[\![m_i^0]\!], m_i^1\}$
- C outputs a bit $b = B(T_R, \mathcal{X})$.

If the probability $\Pr[x^* = x_0]$ is noticeable, then the probability C outputs 1 is $\gg t$.

The requirement of $\mathcal{A}_{\mathsf{Break}}$ provides a lower bound on the 'strength' of what is required for an assumption for to be q-type. One can imagine a 'nominally' q-type assumption where an adversary is given

$$g^x \quad , \quad g^y \quad , \quad g^z \quad , \quad g^{z^2} \quad , \quad g^{z^3} \quad \ldots \quad g^{z^q}$$

and tasked with distinguishing g^{xy} from random. This is clearly implied by DDH by embedding the DDH challenge as x, y. By requiring that knowledge of the 'q-type exponent' allows for the assumption to be broken, we rule out these trivial assumptions while still capturing common q-type assumptions such as q-(bilinear) Diffie-Hellman exponent [BBG05, BGW05], q-(bilinear) Diffie-Hellman inversion [MSK02, BB04a], or q-weak/strong Diffie-Hellman [MSK02, BB04b].

5 Results

First, we formalize the key observation that limits the distribution of valid inputs to a q-type assumption conditional on how it is constructed from challenge elements.

Lemma 5.1. *Let $p(\cdot) = (p_1, p_2, \ldots, p_n)$ be a vector of linearly independent univariate polynomials of degree at most $m - 1 \geq n$ over field $\mathbb{F}$. The intersection of $p(\cdot)$ with any $\leq n - 1$ dimensional subspace of $\mathbb{F}^n$ is at most $m - 1$*

Proof. Assume for sake of contradiction there are m unique field elements $x_1, x_2, \ldots x_m \in \mathbb{F}$ contained in some $n - 1$ dimensional subspace of $\mathbb{F}^n$

Let the $m \times n$ matrix $\boldsymbol{M}$ be where $M_{i,j} = p_i(x_j)$ - i.e. the rows of $\boldsymbol{M}$ are exactly the evaluation of $p(\cdot)$ on these elements. Now consider the $m \times m$ Vandermonde matrix

$$\boldsymbol{V} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^{m-1} \end{bmatrix}$$

As $x_1, x_2, \ldots x_m$ are unique, this matrix has rank m. For each polynomial $p_i(x) = c_0 + c_1 x + c_2 x^2 + \ldots c_{m-1} x^{m-1}$, let $\vec{p}_i = [c_0, c_1, \ldots c_{m-1}] \in \mathbb{Z}_p^m$. Since $\{p_1, p_2, \ldots p_n\}$ are linearly independent, as are vectors $\vec{p}_1, \vec{p}_2, \ldots \vec{p}_n$. Observe $\vec{q}_i = \boldsymbol{V} \cdot \vec{p}_i$ is simply the evaluation of p_i on points $x_1, x_2, \ldots x_m$. Since these are linearly independent vectors under an invertible map, they too must be invertible. Therefore, the matrix

$$\boldsymbol{V} \cdot [\vec{p}_1^{\,\intercal} \mid \vec{p}_2^{\,\intercal} \cdots \mid \vec{p}_n^{\,\intercal}]$$

has column rank (and thus rank) n. Thus, they cannot be contained within an $n - 1$ dimensional subspace. □

Now we are ready to state our main theorem - a black box separation between q-type assumptions and assumptions of size $< q - 1$.

Theorem 5.2. *Suppose there exists a reduction from a q-type assumption* Q *as per Definition 4.2 to a fixed size assumption as per* F *Definition 4.1 of size $n < q - 1$. Then there exists a GGM algorithm which breaks assumption* F.

Proof. Let Q be parameterized by polynomial family $\mathcal{F} = \{f_i(\cdot)\}$ and probability threshold t_{Q}. Recall also there exists an x_0-dependent 'breaking adversary' $\mathcal{A}_{\mathsf{Break}}$. Let F be parameterized by probability threshold t_{F}.

Suppose there is a reduction $\mathcal{B}$, when given access to a solver $\mathcal{S}$ which can break assumption Q with noticeable probability, breaks assumption F with noticeable probability. Then there exists an efficient algorithm $\mathcal{A}_{\mathsf{F}}$ which breaks assumption F with noticeable probability, which we describe below. We will write $\mathcal{A}_{\mathsf{F}}$ as 3 separate algorithms $(\mathcal{A}_{\mathcal{B}}, \mathcal{A}_{\mathcal{S}}, \mathcal{A}_{\mathcal{O}})$ with a shared state. $\mathcal{A}_{\mathcal{B}}$ interacts directly with the F challenger via simulating $\mathcal{B}$, while $\mathcal{A}_{\mathcal{S}}$ answers the queries the simulated reduction algorithm $\mathcal{B}$ makes with its solver $\mathcal{S}$ by using $\mathcal{A}_{\mathsf{Break}}$. Finally, $\mathcal{A}_{\mathcal{O}}$ replaces the element gates used by the (simulated) reduction $\mathcal{B}$ and $\mathcal{A}_{\mathsf{Break}}$.

These algorithms will share a dictionary $\mathsf{D} : \mathbb{G} \to \mathbb{Z}_p^{n+1}$ from element wires to bit wires, which $\mathcal{A}$ will initialize as empty. [3]

[3] This can be implemented by in the Maurer/TS GGM model by simply equality checking every entry

- $\mathcal{A}_{\mathcal{B}}$ runs reduction algorithm $\mathcal{B}^{\mathcal{S}(\cdot)}$.
- For rounds $i \in [n]$
  - Receive $[\![x_i]\!]$ and y_i from C_{F}
  - Set $\mathsf{D}[[\![x_i]\!]] = e_i$ (the i^{th} canonical basis vector on $\mathbb{Z}_p^{n+1}$)
  - Forward $[\![x_i]\!]$ and y_i to $\mathcal{B}$, receiving response message m_i
  - Return m_i to C_{F}
- $\mathcal{B}$ outputs a bit β', which $\mathcal{A}_{\mathcal{B}}$ sends to F challenger.

Figure 1: Program $\mathcal{A}_{\mathcal{B}}(\cdot)$.

- $\mathcal{A}_{\mathcal{S}}$ runs breaking adversary $\mathcal{A}_{\mathsf{Break}}$ of Q
- For rounds $i \in [R_{\mathsf{Q}}]$
  - Let T_i be the transcript of outputs thus far.
  - Receive $[\![x_i]\!], y_i$ from $\mathcal{B}$
  - If $\exists S_{f'} : \dim(S_{f'}) \geq q$ (as per in Definition 4.2) for the first time:
    - Let $S_{f'} = \{s_j\}_{j\in[q]}$.
    - Define the matrix $\boldsymbol{M} \in \mathbb{Z}_p^{q\times(n+1)}$ where row $\boldsymbol{M}_j = \mathsf{D}[[\![x_{i_j}]\!]] : f_{T_{i_j}}(\mathcal{X}) = s_j(x_0)\cdot f'(\mathcal{X})$
    - Compute the set of solution $X^\dagger = \{x_i^\dagger\}$ such that

$$(s_1(x_i^\dagger), s_2(x_i^\dagger), \ldots s_q(x_i^\dagger)) \in \mathsf{Span}(\mathsf{col}(\boldsymbol{M}))$$

    - Define set

$$X^* = \{x^* \in X^\dagger \mid \forall j,k \in [q] \ : \ [\![x_{i_j}]\!]/s_j(x^*) = [\![x_{i_k}]\!]/s_k(x^*)\}$$

    - Return a uniform element $x^* \xleftarrow{\mathrm{R}} X^*$ to $\mathcal{A}_{\mathsf{Break}}$.
  - Forward to $\mathcal{A}_{\mathsf{Break}}$, and return response m_i.

Figure 2: Program $\mathcal{A}_{\mathcal{S}}(\cdot)$.

- **Labeling Gate.** Take in wires encoding x. Use regular labeling gate to compute $[\![x]\!]$. Add $\mathsf{D}[[\![x]\!]] = x \cdot e_{n+1}$ to the dictionary and return $[\![x]\!]$.
- **Group Operation Gate.** Take in wires encoding $a_1, a_2, [\![x_1]\!], [\![x_2]\!]$. Use group operation gate to compute $[\![a_1x_1 + a_2x_2]\!]$. Look up $v_1 = \mathsf{D}[[\![x_1]\!]], v_2 = \mathsf{D}[[\![x_2]\!]]$, add $\mathsf{D}[[\![a_1x_1 + a_2x_2]\!]] = a_1v_1 + a_2v_2$ to the dictionary, and return $[\![a_1x_1 + a_2x_2]\!]$.
- **Equality Gate.** Implement as normal.

Figure 3: Replacement Gates $\tilde{O}(\cdot)$.

Lemma 5.3. $|X^\dagger| \leq \deg(S_{f'})$

Proof. Note that since $\boldsymbol{M}$ has $n+1$ columns, we can bound $\dim(\mathrm{col}(\boldsymbol{M})) \leq n+1$. Since $q > n+1$, the number of solutions follows from Lemma 5.1. □

Lemma 5.4. *$\mathcal{A}$ is efficient.*

Proof. First, we can see that $\mathcal{A}_{\mathcal{B}}$ and $\mathcal{A}_{\mathcal{O}}$ both do a constant amount of work for each interaction and group operation performed by $\mathcal{B}$ respectively. Since $\mathcal{B}$ is efficient, it follows that these two are as well. Finally, observe that apart from forwarding responses between $\mathcal{B}$ and $\mathcal{A}_{\mathsf{Break}}$, the group operations of $\mathcal{A}_{\mathcal{S}}$ just include the dictionary lookups to construct $\boldsymbol{M}$ and the division checks to filter X^* from $X^\dagger$. The former is only done q times (once per row of $\boldsymbol{M}$), while the latter also only needs to be checked for the $\deg(S_{f'})$ elements of $X^\dagger$ (from Lemma 5.3), both of which are efficient. □

Lemma 5.5. *$\mathcal{A}$ breaks assumption* F *with probability* $\gg t_{\mathsf{F}}$.

Proof. We will argue the advantage of $\mathcal{A}$ against F through a small sequence of experiments. Let E_i be the event that C_{F} returns 1 in experiment i.

- Exp_0 This is the original security game of assumption F against $\mathcal{A}$.
- Exp_1 In this experiment, $\mathcal{A}_{\mathcal{S}}$ is modified to inefficiently generate X^* without dictionary D.

- $\mathcal{A}^1_{\mathcal{S}}$ runs breaking adversary $\mathcal{A}_{\mathsf{Break}}$ of Q
- For rounds $i \in [R_{\mathsf{Q}}]$
  - Let T_i be the transcript of outputs thus far.
  - Receive $[\![x_i]\!], y_i$ from $\mathcal{B}$
  - If $\exists S_{f'} : \dim(S_{f'}) \geq q$ (as per in Definition 4.2) for the first time:
    - (Inefficiently) recover $x_{i_j} : f_{T_{i_j}} = s_j \cdot f'$ from $[\![x_{i_j}]\!]$
    - Compute set

$$X^* = \{x^* \in \mathbb{Z}_p \mid \exists w \in \mathbb{Z}_p \text{ where } \forall j \;\; w \cdot s_j(x^*) = x_{i_j}\}$$

    - Return a uniform element $x^* \xleftarrow{R} X^*$ to $\mathcal{A}_{\mathsf{Break}}$.
  - Forward $[\![x_i]\!], y_i$ to $\mathcal{A}_{\mathsf{Break}}$, and return response m_i.

Figure 4: Program $\mathcal{A}^1_{\mathcal{S}}(\cdot)$.

Lemma 5.6. $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1]$

Proof. Observe that these experiments only differ in how set X^* is computed (which we will refer to as X^*_0 and X^*_1 respectively). Thus, it suffices to show that $X^*_0 = X^*_1$. The containment $X^*_0 \subseteq X^*_1$ is easy to see, as we can set the term $[\![x_{i_j}]\!]/s_j(x^*)$ to be the w in X^*_1.

Conversely to show $X^*_1 \subseteq X^*_0$, we will first need to show the 'correctness' of our dictionary D.

Claim 5.7. *Let $[\![x_i]\!]$ for $i \in [n]$ be the group elements sent from C_{F}, and let $[\![x]\!]$ be an element wire in $\mathcal{A}$. Then $\mathsf{D}[[\![x]\!]] = \vec{v} \neq \perp$, and moreover, $x = \langle (x_1, x_2, \ldots x_n, 1), \vec{v} \rangle$.*

Proof. We observe that since $\mathcal{A}_{\mathcal{B}}$ and $\mathcal{A}_{\mathcal{S}}$ never produce any new group elements on their own, any $[\![x]\!]$ either comes from the challenger C_{F} or through running $\mathcal{B}$, $\mathcal{A}_{\mathsf{Break}}$. We can verify that in either of these cases, this invariant is maintained for any new group elements produced.

- $[\![x]\!]$ **is received from** C_{F} - In this case, $[\![x]\!] = [\![x_i]\!]$ for some $i \in [n]$. From $\mathcal{A}_{\mathcal{B}}$, we can see $\mathsf{D}[[\![x]\!]] = e_i$, and $\langle (x_1, x_2, \ldots x_n, 1), e_i \rangle$ is indeed x_i.
- $[\![x]\!]$ **is produced by a labeling gate (in** $\mathcal{B}, \mathcal{A}_{\mathsf{Break}}$**)** - In this case, $\vec{v} = x \cdot e_{n+1}$, so we can see $\langle (x_1, x_2, \ldots x_n, 1), x \cdot e_{n+1} \rangle = x$
- $[\![x]\!]$ **is produced by a group operation gate (in** $\mathcal{B}, \mathcal{A}_{\mathsf{Break}}$**)** - Let $a_1, a_2, [\![y_1]\!], [\![y_2]\!]$ be the inputs to this group operation gate. We can inductively reason for $\vec{v_1} = \mathsf{D}[[\![y_1]\!]], \vec{v_2} = \mathsf{D}[[\![y_2]\!]]$, that $\langle (x_1, x_2, \ldots x_n, 1), \vec{v_1} \rangle = y_1$ and $\langle (x_1, x_2, \ldots x_n, 1), \vec{v_2} \rangle = y_2$. Thus,

$$\begin{aligned} \langle (x_1, x_2, \ldots x_n, 1), a_1\vec{v_1} + a_2\vec{v_2} \rangle &= a_1 \langle (x_1, x_2, \ldots x_n, 1), \vec{v_1} \rangle + a_2 \langle (x_1, x_2, \ldots x_n, 1), \vec{v_1} \rangle \\ &= a_1 y_1 + a_2 y_2 \\ &= x \end{aligned}$$

□

Let $x^* \in X_1^*$, and consider the vector

$$\vec{u} = (x_{i_1}/w, x_{i_2}/w, \ldots x_{i_n}/w, 1/w) \in \mathbb{Z}_p^{n+1}.$$

By the above claim, $\langle \vec{u}, \boldsymbol{M}_j = \mathsf{D}[[\![x_{i_j}]\!]]\rangle = x_{i_j}/w = s_j(x^*)$. Thus we can conclude $\boldsymbol{M}\cdot\vec{u}^\intercal = (s_1(x^*), s_2(x^*), \ldots s_q(x^*))^\intercal$, so $x^* \in X^\dagger$. Since $[\![x_{i_j}]\!]/s_j(x^*) = [\![w]\!]$, it will not be filtered out when sampling X_0^*.

□

Lemma 5.8. $\Pr[\mathsf{E}_1] \gg t_{\mathsf{F}}$

Proof.

Claim 5.9. *$\mathcal{A}_{\mathcal{S}}^1$ is an algorithm which breaks the assumption* Q *security game with nonnegligible advantage.*

Proof. Since $\mathcal{A}_{\mathcal{S}}^1$ exactly forwards the inputs from the challenger to and from $\mathcal{A}_{\mathsf{Break}}$, it suffices to check that $x^* = x_0$ with noticeable probability - which by definition means $_{\mathsf{Break}}$ succeeds with noticeable advantage. Recall that since $f_{T_{i_j}}(\mathcal{X}) = f'(\mathcal{X})\cdot s_j(x_0)$, we can take $w = f'(\mathcal{X})$ to see that $x_0 \in X^*$. From Lemma 5.3, x_0 is output with probability $\geq \frac{1}{\deg(S_{f'})}$. From Definition 4.2, $\deg(S_{f'}) \in \mathsf{poly}(\lambda)$, so $\Pr[x^* = x_0]$ is noticeable. □

Since $\mathcal{B}$ is a reduction, from the last claim, $\mathcal{B}[\mathcal{A}_{\mathcal{S}}^1]$ has nonnegligible advantage in the assumption F security game. Since $\mathcal{A}_{\mathcal{B}}$ exactly forwards the input/outputs of $\mathcal{B}$, $\mathcal{A}$ has nonnegligible advantage in this game as well.

□

Taking together Lemmas 5.6 and 5.8, the advantage of $\mathcal{A}$ in assumption F is $\gg t_{\mathsf{F}}$. □

Combining Lemmas 5.4 and 5.5, we get an efficient adversary which breaks assumption F.

□

5.1 Generalizing to Bilinear Groups

We can naturally extend our definition of hardness assumptions to bilinear groups by simply allowing both the challenger and adversary to send target group elements $[\![g_{T_i}(\mathcal{X})]\!]_T, [\![m_i^2]\!]_T$ respectively.

Theorem 5.10. *Suppose there exists a reduction from a* bilinear *q-type assumption* Q *to a fixed size assumption* F *of size* $n : \binom{n+2}{2} < q$. *Then there exists a generic bilinear group algorithm which breaks assumption* F.

Our result and techniques from above readily generalizes to the generic bilinear (and indeed k-linear for fixed k) groups. The presence of the bilinear map mean that our meta-reduction, rather than recording group elements as linear function of the challenge inputs, needs to consider quadratic functions as well. However, we can effectively 'linearize' this by simply mapping all group elements x_i received from C as linear functions in degree ≤ 2 monomials of $(x_1, x_2, \ldots x_n)$. As a result, our polynomial set $\mathcal{S}_{f'}$ is now bound by a $\binom{n+2}{2}$ dimensional subspace, which gives us the parameters of our theorem. For the complete definition and proof of this theorem, see Appendix A.

We remark that this gap from being able to rule out reductions to $O(n)$ sized assumptions in generic groups to $O(n^2)$ in generic bilinear groups is not just a limitation of our proof techniques, and is in fact inherent. Consider the following two assumptions:

Assumption 5.11. Sample $x \xleftarrow{R} \mathbb{Z}_p$. Given

$$A_1 = [\![x]\!], A_2 = [\![x^2]\!] \ldots A_{q-1} = [\![x^{q-1}]\!], B_1 = [\![x^q]\!], B_2 = [\![x^{2q}]\!] \ldots B_{q-1} \left[\!\!\left[x^{q(q-1)}\right]\!\!\right]$$

Distinguish $T = \left[\!\!\left[x^{2q^2}\right]\!\!\right]$ from random.

Assumption 5.12. Sample $x \xleftarrow{R} \mathbb{Z}_p$ Given

$$C_1 = [\![x]\!]_T, C_2 = [\![x^2]\!]_T, C_3 = [\![x^3]\!]_T \ldots C_n = [\![x^{q^2-1}]\!]_T$$

Distinguish $T' = [\![x^{2q^2}]\!]_T$ from random.

These assumptions are both clearly true in the generic group model, but we *can* in fact reduce the security of Assumption 5.12 to Assumption 5.11 by simply setting $C_i = e(B_{\lfloor i/q \rfloor}, A_{i \mod q})$ (where $A_0, B_0 = [\![1]\!]$) and $T' = e(T, [\![1]\!])$.

6 Applications

Our results apply to a broad range of both cryptographic assumptions as well as schemes, some of which we will highlight below.

Corollary 6.1. *There does not exist a generic reduction from q-Strong Diffie Hellman Assumption to a fixed-size assumption.*

Proof. Recall the q-SDH assumption from [BB04b]: given $[\![x]\!], [\![x^2]\!] \ldots, [\![x^q]\!]$, compute tuple $(c, [\![\frac{1}{x+c}]\!])$.

Claim 6.2. q-SDH *is a q-type assumption per Definition 4.2.*

Proof. As a static assumption, we can immediately see that $[\![x]\!], [\![x^2]\!] \ldots, [\![x^q]\!]$ represents a $q-$dimensional set of polynomials which the adversary always receives. In addition, consider the following adversary.

- For rounds $i \in [R]$
  - Receive group element $A_i = [\![x_0^i]\!]$
  - If group element x^* is received, check that $A_1^{x^*} = A_2$. If so, return $(1, [\![1/(x^* + 1)]\!])$, otherwise output $\perp$

Figure 5: Program $\mathcal{A}_{\mathsf{Break}}$.

Observe that when $x_0 = x^*$ (which is checkable via exponentiating provided group elements), $\mathcal{A}_{\mathsf{Break}}$ can simply construct the challenge for an arbitrary c, thus succeeding with probability $\Pr[x_0 = x^*]$, which is nonnegligible.

□

Now that we have shown q-SDH falls within our q-type framework, the corollary follows immediately from Theorem 5.2.

□

Corollary 6.3. *There does not exist a generic reduction from q-Bilinear Diffie Hellman Exponent Assumption to a fixed-size assumption.*

Proof. Recall the q-BDHE assumption from [BB04a]: given $[\![y]\!], [\![x]\!], [\![x^2]\!] \ldots, [\![x^{q-1}]\!], [\![x^{q+1}]\!], \ldots [\![x^{2q}]\!]$, distinguish $[\![yx^q]\!]_T$ from a random target group element.

As before, the result follows immediately from applying our theorem Theorem 5.10, so long as we show the q-BDHE is in fact a q-type assumption.

Claim 6.4. *q-BDHE is a q-type assumption per Definition A.2.*

Proof. As a static assumption, we can immediately see that $[\![x]\!], [\![x^2]\!] \ldots, [\![x^{q-1}]\!], [\![x^{q+1}]\!], \ldots [\![x^{2q}]\!]$ represents a $2q-1$-dimensional set of polynomials which the adversary always receives. In addition, consider the following adversary.

- For rounds $i \in [R]$
  - Receive group element $A_i = [\![x_0^i]\!], B = [\![y]\!]$, challenge element T
  - If group element x^* is received, check that $A_1^{x^*} = A_2$. If not, return $\perp$
  - Construct $A_q = [\![x^{*q}]\!]$, and compute $T^* = e(A_q, B)$. Return 1 iff $T = T^*$.

Figure 6: Program $\mathcal{A}_{\mathsf{Break}}$.

Observe that when $x_0 = x^*$ (which once again is checkable via exponentiating the provided group elements), $\mathcal{A}_{\mathsf{Break}}$ can simply construct the challenge element $[\![x_0^q \cdot y]\!]_T$ using $[\![y]\!]$, thus succeeding with probability $\Pr[x_0 = x^*]$, which is nonnegligible.

□

□

Corollary 6.5. *There does not exist a generic reduction from the security of the broadcast encryption scheme of [BGW05] to a fixed-size assumption.*

Proof. Our framework of q-type assumptions also captures interactive security games, and as such our results extend to these games as well.

We will restate the ecurity of broadcast when instantiated with [BGW05]:

- C samples $x, y \xleftarrow{R} \mathbb{Z}_p$
- $\mathcal{A}$ outputs challenge set $S^* \subseteq [n]$
- C sends public key $[\![x]\!], [\![x^2]\!], \ldots [\![x^n]\!], [\![x^{n+2}]\!], \ldots [\![x^{2n}]\!], [\![y]\!]$ and private keys $[\![y \cdot x^i]\!]$ for $i \notin S^*$.
- $\mathcal{A}$ sends decryption queries which C answers
- $\mathcal{A}$ requests the challenge:
  - $\mathcal{A}$ sends set $S \subseteq S^*$
  - C samples $t \xleftarrow{R} \mathbb{Z}_p$ and sends

$$[\![t]\!], \left[\!\!\left[t \cdot \left(y + \sum_{j \in S} x^{n+1-j} \right) \right]\!\!\right]$$

– C samples a random bit $b \xleftarrow{R} \{0,1\}$, and sends challenge

$$T_b = [\![x^{n+1}]\!]_T \,,\; T_{1-b} \xleftarrow{R} \mathbb{G}_T$$

- $\mathcal{A}$ sends more decryption queries which C answers

Claim 6.6. *Security game of the broadcast encryption when instantiated with [BGW05] is a q-type assumption Definition A.2.*

Proof. Once again, it is easy to see that the public key $\mathcal{A}$ receives contains $[\![x]\!], [\![x^2]\!], \ldots [\![x^n]\!], [\![x^{n+2}]\!], \ldots [\![x^{2n}]\!]$, a $2n-1$-dimensional set of polynomials in x. Now we can construct $\mathcal{A}_{\mathsf{Break}}$ as follows:

- Send challenge set $S^* = [n]$
- Receive public key $[\![x]\!], [\![x^2]\!], \ldots [\![x^n]\!], [\![x^{n+2}]\!], \ldots [\![x^{2n}]\!], [\![y]\!]$
- Receive candidate x^*. Check that $[\![x]\!] \cdot x^* = [\![x^2]\!]$. Otherwise abort.
- Request challenge on $S = [n]$. If $T_0 = [\![x^{*n+1}]\!]_T$ return 0, otherwise return 1.

Figure 7: Program $\mathcal{A}_{\mathsf{Break}}$.

As with our prior assumptions, learning $x^* = x_0$ allows us to directly compute the encapsulated key, and thus breaking the security of the broadcast scheme.

□

Thus, we can conclude via Theorem 5.10 that the security of [BGW05] cannot be reduced to a fixed size assumption.

□

References

[BB04a] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[Ber71] E. R. Berlekamp. Factoring polynomials over large finite fields*. In *Proceedings of the Second ACM Symposium on Symbolic and Algebraic Manipulation*, SYMSAC '71, page 223, New York, NY, USA, 1971. Association for Computing Machinery.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography*, pages 325–341, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 258–275, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[BLP92] J. P. Buhler, H. W. Lenstra, and C. Pomerance. Factoring integers with the number field sieve. 1992.

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 801–830. Springer, 2019.

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking rsa may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 59–71, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[Che06] Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 1–11, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[CM14] Melissa Chase and Sarah Meiklejohn. Déjà q: Using dual systems to revisit q-type assumptions. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 622–639, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[CMM16] Melissa Chase, Mary Maller, and Sarah Meiklejohn. Déjà q all over again: Tighter and broader reductions of q-type assumptions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 655–681, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[CZ81] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, 1981.

[Del07] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, pages 200–215, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 33–62, Cham, 2018. Springer International Publishing.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 44–61, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Gjø04] Kristian Gjøsteen. Subgroup membership problems and public key cryptosystems. 2004. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/249681.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 415–432, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 44–61. ACM, 1989.

[Lew12] Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 318–335, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[LM15] Allison Lewko and Sarah Meiklejohn. A profitable sub-prime loan: Obtaining the advantages of composite order in prime-order bilinear groups. In Jonathan Katz, editor, *Public-Key Cryptography – PKC 2015*, pages 377–398, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC*, 2010.

[LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, 2011.

[Mau05] Ueli Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *Cryptography and Coding*, pages 1–12, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[MSK02] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 85-A(2):481–484, 2002.

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 214–231, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Rab80] Michael O. Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on Computing*, 9(2):273–280, 1980.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[SC12] Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 133–150. Springer, 2012.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 256–266, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[Wat09a] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, 2009.

[Wat09b] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 619–636, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Wee16] Hoeteck Wee. Déjà q: Encore! un petit ibe. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography*, pages 237–258, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[Wee21] Hoeteck Wee. Broadcast encryption with size $n^{1/3}$ and more from k-lin. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 155–178, Cham, 2021. Springer International Publishing.

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Heidelberg, August 2022.

A Generalizing to Bilinear Groups

Here we show the complete definitions and proof of our theorems generalized to bilinear groups.

A.1 Assumptions

Similar to the plain GGM case, we can specify a hardness assumption on bilinear groups by a collection of functions $\mathcal{F} = \{f(\cdot) : \{0,1\}^* \to \mathbb{Z}_p\}, \underline{\mathcal{G} = \{g(\cdot) : \{0,1\}^* \to \mathbb{Z}_p\}}, \mathcal{H} = \{h(\cdot) : \{0,1\}^* \to \{0,1\}^*\}$, a success function B. Then the game defined by C works as follows:

- C samples randomness $\mathcal{X} \xleftarrow{R} \{0,1\}^*$
- For a polynomial number of rounds $i \in [R]$
  - Let T_i be the transcript of outputs given by $\mathcal{A}$ by round i
  - C sends $[\![f_{T_i}(\mathcal{X})]\!], \underline{[\![g_{T_i}(\mathcal{X})]\!]_T}, h_{T_i}(\mathcal{X})$ to $\mathcal{A}$
  - $\mathcal{A}$ sends a response message $m_i = \left\{[\![m_i^0]\!], m_i^1, \underline{[\![m_i^2]\!]_T}\right\}$
- C outputs a bit $b = B(T_R, \mathcal{X})$.

Definition A.1. We say an assumption $\mathsf{G} = (C, t)$ on bilinear groups is of fixed-size n if the total number of group elements (including both the source and target group) produced by C is $\leq n$ for any adversary.

Definition A.2. We refer to a hardness assumption on bilinear groups G as a q-type assumption if, as part of challenger randomness $\mathcal{X}$, there exists a $x_0 \in \mathbb{Z}_p$. In addition, for all executions $\{T_1, T_2, \ldots\}$ of G, there exists a function $f'(\mathcal{X})$, as well as a set of polynomials $S_{f'} = \{s_i(x_0)\}$ where:

1. $\deg(S_{f'}) \in \mathsf{poly}(\lambda)$
2. $\dim(S_{f'}) \geq q$
3. $s_i(x_0) \cdot f'(\mathcal{X}) = f_{T_j}(\mathcal{X})$ $\underline{\text{or} = g_{T_j}(\mathcal{X})}$ for some transcript T_j

Furthermore, there exists an (efficient) algorithm $\mathcal{A}_{\mathsf{Break}}$ such that in the following game:

- C samples randomness $\mathcal{X} \xleftarrow{R} \{0,1\}^*$, including $\vec{x} \xleftarrow{R} \mathbb{Z}_p^n$
- For a polynomial number of rounds $i \in [R]$
  - Let T_i be the transcript of outputs given by $\mathcal{A}_{\mathsf{Break}}$ by round i
  - C sends $[\![f_{T_i}(\mathcal{X})]\!], [\![g_{T_i}(\mathcal{X})]\!]_T, h_{T_i}(\mathcal{X})$ to $\mathcal{A}_{\mathsf{Break}}$
  - If $dim(S_{f'}) \geq q$ for the first time, send an x^* to $\mathcal{A}_{\mathsf{Break}}$
  - $\mathcal{A}_{\mathsf{Break}}$ sends a response message $m_i = \{[\![m_i^0]\!], m_i^1\}$
- C outputs a bit $b = B(T_R, \mathcal{X})$.

If the probability $\Pr[x^* = x_0]$ is noticeable, then the probability C outputs 1 is $\gg t$.

A.2 Theorem for Bilinear Groups

We are now ready to formally prove Theorem 5.10, restated below:

Theorem A.3. *Suppose there exists a reduction from a bilinear q-type assumption* Q *as per Definition A.2 to a fixed size assumption as per* F *Definition A.1 of size* $n : \binom{n+2}{2} < q$. *Then there exists a GGM algorithm which breaks assumption* F.

Proof. Let Q be parameterized by polynomial families $\mathcal{F} = \{f_i(\cdot)\}, \mathcal{G} = \{g_i(\cdot)\}$ and probability threshold t_{Q}. Recall also there exists an x_0-dependent 'breaking adversary' $\mathcal{A}_{\mathsf{Break}}$. Let F be parameterized by probability threshold t_{F}.

Suppose there is a reduction $\mathcal{B}$, when given access to a solver $\mathcal{S}$ which breaks assumption Q with noticeable probability, breaks assumption F with noticeable probability. Then there exists algorithm $\mathcal{A}_{\mathsf{F}}$ which breaks assumption F with noticeable probability as well, which we describe below. We will write $\mathcal{A}_{\mathsf{F}}$ as 3 separate algorithms $(\mathcal{A}_{\mathcal{B}}, \mathcal{A}_{\mathcal{S}}, \mathcal{A}_{\mathcal{O}})$ with a shared state. $\mathcal{A}_{\mathcal{B}}$ interacts directly with the F challenger via simulating $\mathcal{B}$, while $\mathcal{A}_{\mathcal{S}}$ answers the queries the simulated reduction algorithm $\mathcal{B}$ makes with its solver $\mathcal{S}$ by using $\mathcal{A}_{\mathsf{Break}}$. Finally, $\mathcal{A}_{\mathcal{O}}$ replaces the element gates used by the (simulated) reduction $\mathcal{B}$ and $\mathcal{A}_{\mathsf{Break}}$.

These algorithms will share two dictionaries $\mathsf{D} : \mathbb{G}^T \to p^1(x_1, x_2, \ldots x_n)$ and $\mathsf{D}_T : \mathbb{G}^T \to p^2(x_1, x_2, \ldots x_n)$ mapping element wires to linear and quadratic polynomials on random variables $x_1, \ldots x_n$ respectively, which $\mathcal{A}$ will initialize as empty.

We define function $\rho[r_1, \ldots r_n](\cdot)$ from degree 2 polynomials on variables $(x_1, \ldots, x_n)$ to $\mathbb{Z}_p^{\binom{n+2}{2}}$ obtained by mapping the evaluation of each unique monomial term on input $(r_1, \ldots r_n)$ to a unique index $\in \left[\binom{n+2}{2}\right]$ [4]. We will use $\rho(\cdot)$ as shorthand for $\rho[\vec{1}](\cdot)$, which returns a vector of coefficients. We will also define $P_2(x_1, \ldots, x_n)$ to be the quadratic polynomial with coefficient 1 on every monomial (i.e. $\rho(P_2) = \vec{1}$).

- $\mathcal{A}_{\mathcal{B}}$ runs reduction algorithm $\mathcal{B}^{\mathcal{S}(\cdot)}$.
- Start a counter ctr = 1
- For rounds $i \in [R_{\mathsf{F}}]$
  - Receive $[\![w_i]\!], y_i$ and $[\![z_i]\!]_T$ from C_{F}
    - If $[\![w_i]\!] \neq \perp$, set $\mathsf{D}[[\![w_i]\!]] = x_{\mathsf{ctr}}$ and increment ctr+ = 1
    - If $[\![z_i]\!]_T \neq \perp$, set $\mathsf{D}_T[[\![z_i]\!]_T] = x_{\mathsf{ctr}}$ and increment ctr+ = 1
  - Forward $[\![w_i]\!], y_i$ and $[\![z_i]\!]_T$ to $\mathcal{B}$, receiving response message m_i
  - Return m_i to C_{F}
- $\mathcal{B}$ outputs a bit β', which $\mathcal{A}_{\mathcal{B}}$ sends to F challenger.

Figure 8: Program $\mathcal{A}_{\mathcal{B}}(\cdot)$.

[4] We can count the number of unique monomials on n variables of degree d as putting d identical 'balls' in $n+1$ bins (one for each variable and an additional one for 1). This is a well-known combinatorics problem with formula $\binom{n+d}{d}$

- $\mathcal{A}_{\mathcal{S}}$ runs breaking adversary $\mathcal{A}_{\mathsf{Break}}$ of Q
- For rounds $i \in [R_{\mathrm{Q}}]$
  - Let T_i be the transcript of outputs thus far.
  - Receive $[\![w_i]\!], y_i, [\![z_i]\!]_T$ from $\mathcal{B}$
  - If $\exists S_{f'} : \dim(S_{f'}) \geq q$ (as per in Definition A.2) for the first time:
    - Let $S_{f'} = \{s_j\}_{j\in[q]}$.
    - Define the matrix $\boldsymbol{M} \in \mathbb{Z}_p^{q\times\binom{n+2}{2}}$ as follows
      - If $f_{T_{i_j}}(\cdot) = s_j(\cdot)f'(\cdot)$, then set row $\boldsymbol{M}_j = \rho(\mathsf{D}[[\![w_{i_j}]\!]])$ and test element $[\![r_j]\!]_T = e([\![w_{i_j}]\!], [\![1]\!])$
      - If $g_{T_{i_j}}(\cdot) = s_j(\cdot)f'(\cdot)$, then set row $\boldsymbol{M}_j = \rho(\mathsf{D}_T[[\![z_{i_j}]\!]_T])$ and test element $[\![r_j]\!]_T = [\![z_{i_j}]\!]_T$
    - Compute the set of solution $X^\dagger = \{x_i^\dagger\}$ such that
    $$(s_1(x_i^\dagger), s_2(x_i^\dagger), \ldots s_q(x_i^\dagger)) \in \mathsf{Span}(\mathsf{col}(\boldsymbol{M}))$$
    - Define set
    $$X^* = \{x^* \in X^\dagger \mid \forall j,k \in [q] \ : \ [\![r_j]\!]_T / s_j(x^*) = [\![r_k]\!]_T / s_k(x^*)\}$$
    - Return a uniform element $x^* \xleftarrow{\mathrm{R}} X^*$ to $\mathcal{A}_{\mathsf{Break}}$.
  - Forward to $\mathcal{A}_{\mathsf{Break}}$, and return response m_i.

Figure 9: Program $\mathcal{A}_{\mathcal{S}}(\cdot)$.

- **Labeling Gate.** Take in wires encoding w. Use regular labeling gate to compute $[\![w]\!]$. Add $\mathsf{D}[[\![w]\!]] = w$ to the dictionary and return $[\![w]\!]$.
- **Group Operation Gate.** Take in wires encoding $a_1, a_2, [\![w_1]\!], [\![w_2]\!]$. Use group operation gate to compute $[\![a_1w_1 + a_2w_2]\!]$. Add $\mathsf{D}[[\![a_1x_1 + a_2x_2]\!]] = a_1 \cdot \mathsf{D}[[\![x_1]\!]] + a_2 \cdot \mathsf{D}[[\![x_2]\!]]$ to the dictionary, and return $[\![a_1w_1 + a_2w_2]\!]$. Update analogously for Group Operation gate in target group using D_T.
- **Equality Gate.** Implement as normal.
- **Pairing Gate.** Take in wires $[\![w_1]\!], [\![w_2]\!]$. Use pairing gate to compute $[\![w_1 \cdot w_2]\!]_T$. Set $\mathsf{D}_T[[\![w_1 \cdot w_2]\!]_T] = \mathsf{D}[[\![w_1]\!]] \cdot \mathsf{D}[[\![w_2]\!]]$ and return $[\![w_1 \cdot w_2]\!]_T$.

Figure 10: Replacement Gates $\mathcal{O}(\cdot)$.

Lemma A.4. $|X^\dagger| \leq \deg(S_{f'})$

Proof. Note that since $\boldsymbol{M}$ has $\binom{n+2}{2}$ columns, we can bound $\dim(\mathsf{col}(\boldsymbol{M})) \leq \binom{n+2}{2}$. Since $q > \binom{n+2}{2}$, the

number of solutions follows from Lemma 5.1. □

Lemma A.5. *$\mathcal{A}$ is efficient.*

Proof. First, we can see that $\mathcal{A}_{\mathcal{B}}$ and $\mathcal{A}_{\mathcal{O}}$ both do a constant amount of work for each interaction and group operation performed by $\mathcal{B}$ respectively. Since $\mathcal{B}$ is efficient, it follows that these two are as well. Finally, observe that apart from forwarding responses between $\mathcal{B}$ and $\mathcal{A}_{\mathsf{Break}}$, the group operations of $\mathcal{A}_{\mathcal{S}}$ just include the dictionary lookups to construct $\boldsymbol{M}$ and the division checks to filter X^* from $X^\dagger$. The former is only done q times (once per row of $\boldsymbol{M}$, while the latter also only needs to be checked for the $\deg(S_{f'})$ elements of $X^\dagger$ (from Lemma A.4), both of which are efficient. □

Lemma A.6. *$\mathcal{A}$ breaks assumption* F *with probability* $\gg t_{\mathsf{F}}$.

Proof. We will argue the advantage of $\mathcal{A}$ against F through a small sequence of experiments. Let E_i be the event that C_{F} returns 1 in experiment i.

- Exp_0 This is the original security game of assumption F against $\mathcal{A}$.
- Exp_1 In this experiment, $\mathcal{A}_{\mathcal{S}}$ is modified to inefficiently generate X^* without dictionaries D, D_T.

  - $\mathcal{A}^1_{\mathcal{S}}$ runs breaking adversary $\mathcal{A}_{\mathsf{Break}}$ of Q
  - For rounds $i \in [R_{\mathsf{Q}}]$
    - Let T_i be the transcript of outputs thus far.
    - Receive $[\![w_i]\!], y_i, [\![z_i]\!]_T$ from $\mathcal{B}$
    - If $\exists S_{f'} : \dim(S_{f'}) \geq q$ (as per in Definition 4.2) for the first time, for $j \in [q]$
      - If $f_{T_{i_j}}(\cdot) = s_j(\cdot) f'(\cdot)$, set test element $[\![r_j]\!]_T = e([\![w_{i_j}]\!], [\![1]\!])$
      - If $g_{T_{i_j}}(\cdot) = s_j(\cdot) f'(\cdot)$, set test element $[\![r_j]\!]_T = [\![z_{i_j}]\!]_T$
      - (Inefficiently) Recover r_j from $[\![r_j]\!]_T$
      - Compute set

$$X^* = \{x^* \in \mathbb{Z}_p \mid \exists c \in \mathbb{Z}_p \text{ where } \forall j \;\; c \cdot s_j(x^*) = r_j\}$$

      - Return a uniform element $x^* \xleftarrow{\text{R}} X^*$ to $\mathcal{A}_{\mathsf{Break}}$.
    - Forward $[\![w_i]\!], y_i, [\![z_i]\!]_T$ to $\mathcal{A}_{\mathsf{Break}}$, and return response m_i.

Figure 11: Program $\mathcal{A}^1_{\mathcal{S}}(\cdot)$.

Lemma A.7. $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1]$

Proof. Observe that these experiments only differ in how set X^* is computed (which we will refer to as X^*_0 and X^*_1 respectively). Thus, it suffices to show that $X^*_0 = X^*_1$. The containment $X^*_0 \subseteq X^*_1$ is easy to see, as we can set the term $[\![r]\!]_T / s_j(x^*)$ to be the $[\![w]\!]_T$ in X^*_1.

Conversely to show $X^*_1 \subseteq X^*_0$, let $x^* \in X^*_1$, we will first need to show the 'correctness' of our dictionaries D, D_T.

Claim A.8. *Let $[\![w_i]\!], [\![z_i]\!]_T$ be the group elements sent from C_{F}, and let r_{ctr} be the discrete log of the ctr^{th} group element (either w_i or z_i). Consider $[\![w]\!], [\![z]\!]_T$ element wires in $\mathcal{A}$. Then*

- $\mathsf{D}[[\![w]\!]] = p_w(x_1, x_2, \ldots x_n) \neq \perp$
- *p_w is a linear polynomial where $p_w(r_1, r_2, \ldots r_n) = w$*
- $\mathsf{D}_T[[\![z]\!]] = p_z(x_1, x_2, \ldots x_n)$
- *p_z is a quadratic polynomial where $p_z(r_1, r_2, \ldots r_n) = z$*

Proof. We observe that since $\mathcal{A}_{\mathcal{B}}$ and $\mathcal{A}_{\mathcal{S}}$ never produce any new group elements on their own, any $[\![w]\!], [\![z]\!]_T$ either comes from the challenger C_{F} or through running $\mathcal{B}$, $\mathcal{A}_{\mathsf{Break}}$. We can verify that in either of these cases, this invariant is maintained for any new group elements produced.

- **$[\![w]\!]$ is received from C_{F}** - In this case, suppose $[\![w]\!]$ is the ctr^{th} group element received for some $\mathsf{ctr} \in [n]$. From $\mathcal{A}_{\mathcal{B}}$, we can see $\mathsf{D}[[\![w]\!]] = x_{\mathsf{ctr}}$ which is indeed linear, and since r_{ctr} is set to w, this evaluates to w. This holds analogously for target group elements $[\![z]\!]_T$.
- **$[\![w]\!]$ is produced by a labeling gate (in $\mathcal{B}, \mathcal{A}_{\mathsf{Break}}$)** - In this case, $\mathsf{D}[[\![w]\!]]$ is set to a constant w.
- **$[\![w]\!]$ is produced by a group operation gate (in $\mathcal{B}, \mathcal{A}_{\mathsf{Break}}$)** - Let $a_1, a_2, [\![w_1]\!], [\![w_2]\!]$ be the inputs to this group operation gate. We can inductively reason that $\mathsf{D}[[\![w_1]\!]](r_1, \ldots r_n) = w_1$ and $\mathsf{D}[[\![w_2]\!]](r_1, \ldots r_n) = w_2$. Thus,

$$\begin{aligned}\mathsf{D}[[\![w]\!]](r_1, \ldots r_n) &= a_1 \mathsf{D}[[\![w_1]\!]](r_1, \ldots r_n) + a_2 \mathsf{D}[[\![w_2]\!]](r_1, \ldots r_n) \\ &= a_1 w_1 + a_2 w_2 \\ &= w\end{aligned}$$

  Note that a linear combination of polynomials does not increase the maximum degree, so $\mathsf{D}[[\![w]\!]]$ is still linear. This holds analogously for target group elements $[\![z]\!]_T$.

- **$[\![z]\!]$ is produced by a pairing gate (in $\mathcal{B}, \mathcal{A}_{\mathsf{Break}}$)** - Let $[\![w_1]\!], [\![w_2]\!]$ be the inputs to this gate. We can inductively reason that $\mathsf{D}[[\![w_1]\!]](r_1, \ldots r_n) = w_1$ and $\mathsf{D}[[\![w_2]\!]](r_1, \ldots r_n) = w_2$. Thus,

$$\begin{aligned}\mathsf{D}_T[[\![z]\!]_T](r_1, \ldots r_n) &= \mathsf{D}[[\![w_1]\!]](r_1, \ldots r_n) \cdot \mathsf{D}[[\![w_2]\!]](r_1, \ldots r_n) \\ &= w_1 \cdot w_2 \\ &= z\end{aligned}$$

  Since $\mathsf{D}[[\![w_1]\!]], \mathsf{D}[[\![w_2]\!]]$ are both linear polynomials, their product is quadratic.

□

Now consider the vector

$$\vec{u} = \frac{1}{c} \cdot \rho[r_1, \ldots r_n](P_2).$$

Observe that for any polynomial $p(\cdot)$, $\langle \vec{u}, \rho(p) \rangle$ is exactly $\frac{1}{c} p(r_1, \ldots r_n)$ (For each index i in $\mathbb{Z}_p^{\binom{n+2}{2}}$, consider the monomial $c_i x_{i_1} x_{i_2} \in p(\cdot)$ corresponding to that entry. $\vec{u}_i$ is exactly $\frac{1}{c} r_{i_1} r_{i_2}$ by definition of P_2, and $\rho(p)_i$ is exactly $c_i \cdot 1$. Thus, they contribute exactly $\frac{c_i}{c} r_{i_1} r_{i_2}$ to the inner product). Thus, we can

conclude from this observation and our above claim that $\langle \vec{u}, \boldsymbol{M}_j \rangle = r_j/c = s_j(x^*)$. and hence $\boldsymbol{M} \cdot \vec{u}^\intercal = (s_1(x^*), s_2(x^*), \ldots s_q(x^*))^\intercal$ meaning $x^* \in X^\dagger$. Since $[\![r_j]\!]_T / s_j(x^*) = [\![c]\!]_T$, it will not be filtered out when sampling X_0^*.

□

Lemma A.9. $\Pr[\mathsf{E}_1] \gg t_{\mathsf{F}}$

Proof.

Claim A.10. *$\mathcal{A}_S^1$ is an algorithm which breaks the assumption* Q *security game with nonnegligible advantage.*

Proof. Since $\mathcal{A}_S^1$ exactly forwards the inputs from the challenger to and from $\mathcal{A}_{\mathsf{Break}}$, it suffices to check that $x^* = x_0$ with noticeable probability. Recall that since $r_j = f'(X) \cdot s_j(x_0)$, we can take $c = f'(X)$, to see that $x_0 \in X^*$. From Lemma 5.3, x_0 is output with probability $\geq \frac{1}{\deg(S_{f'})}$. From Definition 4.2, $\deg(S_{f'}) \in \mathsf{poly}(\lambda)$, so $\Pr[x^* = x_0]$ is noticeable. □

Since $\mathcal{B}$ is a reduction, from the last claim, $\mathcal{B}[\mathcal{A}_S^1]$ has nonnegligible advantage in the assumption F security game. Since $\mathcal{A}_{\mathcal{B}}$ exactly forwards the input/outputs of $\mathcal{B}$, $\mathcal{A}$ has nonnegligible advantage in this game as well.

□

Taking together Lemmas A.7 and A.9, the advantage of $\mathcal{A}$ in assumption F is $\gg t_{\mathsf{F}}$. □

Combining Lemmas A.5 and A.6, we get an efficient adversary which breaks assumption F.

□