Beyond Uber: Instantiating Generic Groups via PGGs

Balthazar Bauer[1], Pooya Farshim[2], Patrick Harasser[3(✉)], and Adam O'Neill[4]

[1] IRIF, CNRS, France
balthazar.bauer@ens.fr

[2] IOHK and Durham University, UK
pooya.farshim@gmail.com

[3] Technische Universität Darmstadt, Germany
patrick.harasser@tu-darmstadt.de

[4] Manning College of Information and Computer Sciences, University of Massachusetts Amherst, USA
adamo@cs.umass.edu

Abstract. The generic-group model (GGM) has been very successful in making the analyses of many cryptographic assumptions and protocols tractable. It is, however, well known that the GGM is "uninstantiable," i.e., there are protocols secure in the GGM that are insecure when using any real-world group. This motivates the study of standard-model notions formalizing that a real-world group in some sense "looks generic."

We introduce a standard-model definition called *pseudo-generic group (PGG)*, where we require exponentiations with base an (initially) unknown group generator to result in random-looking group elements. In essence, our framework delicately lifts the influential notion of Universal Computational Extractors of Bellare, Hoang, and Keelveedhi (BHK, CRYPTO 2013) to a setting where the underlying ideal reference object is a generic group. The definition we obtain simultaneously generalizes the Uber assumption family, as group exponents no longer need to be polynomially induced. At the core of our definitional contribution is a new notion of *algebraic unpredictability*, which reinterprets the standard Schwartz–Zippel lemma as a restriction on sources. We prove the soundness of our definition in the GGM with auxiliary-input (AI-GGM).

Our remaining results focus on applications of PGGs. We first show that PGGs are indeed a generalization of Uber. We then present a number of applications in settings where exponents are not polynomially induced. In particular we prove that simple variants of ElGamal meet several advanced security goals previously achieved only by complex and inefficient schemes. We also show that PGGs imply UCEs for split sources, which in turn are sufficient in several applications. As corollaries of our AI-GGM feasibility, we obtain the security of all these applications in the presence of preprocessing attacks.

Some of our implications utilize a novel type of hash function, which we call *linear-dependence destroyers* (LDDs) and use to convert standard into algebraic unpredictability. We give an LDD for low-degree sources, and establish their plausibility for all sources by showing, via a compression argument, that random functions meet this definition.

1 Introduction

1.1 Background

IDEALIZED MODELS. A useful tool in cryptography are so-called idealized models of computation, which include the random-oracle, random-permutation, ideal-cipher, and generic-group models. In such models, all algorithms work relative to oracles that serve to implement some information-theoretically random reference object. Later, when a scheme defined in an idealized setting is used in practice, the oracles are heuristically instantiated by appropriate public, efficiently computable functions. On the one hand, idealized models are powerful because they limit the adversary's capabilities and make proofs tractable. On the other, they are subject to well-known uninstantiability results, which show the existence of (contrived) schemes that are secure in the idealized model, but provably insecure under any possible instantiation (see, e.g., [15, 27, 33, 39]). This indicates that idealized models are not sound in general, yet "natural" applications (with the oracles appropriately instantiated) have withstood years of scrutiny.

THE GENERIC-GROUP MODEL. In this work, we mainly focus on the generic-group model (GGM), where a generic group is an idealization of a finite cyclic group. It was first defined by Nechaev [46] and later refined by Shoup [50], who considered random encodings of group elements.[5] More specifically, for a cyclic group $(\mathsf{G}, \circ)$ of order p, Shoup's model considers a random injection $\tau\colon \mathbb{Z}_p \to \mathsf{S}$, where $\mathsf{S} \subseteq \{0,1\}^*$ with $|\mathsf{S}| \geq p$. All algorithms run on input p and encodings of application-specific group elements. To perform group operations, algorithms can query a τ oracle and an operation oracle op defined as $\mathsf{op}(h_1, h_2) \coloneqq \tau(\tau^{-1}(h_1) + \tau^{-1}(h_2))$ if $h_1, h_2 \in \mathsf{Rng}(\tau)$, and $\mathsf{op}(h_1, h_2) \coloneqq \perp$ otherwise.

INSTANTIATING GENERIC GROUPS. In practice, a generic group is typically instantiated via an appropriate elliptic curve group. Indeed, for such groups no algorithms for solving discrete-logarithm-like problems more efficiently than the generic ones are known. Addressing the above-mentioned mismatch between idealized and instantiated schemes, we investigate what assumptions are being made when carrying out such an instantiation. Note that an indistinguishability-based approach formalizing the idea of "behaving like a generic group" would suffer from the same shortcomings known for random oracles [27].

This line of work has been carried out with considerable success for the random oracle model (ROM), where the ideal reference object is a random function (see, e.g., [10,24,54]). For generic groups on the other hand, the most compelling formulation so far of what assumption is being made when instantiating them is given by the so-called Uber assumption [18,20]. At a high level, the Uber assumption speaks to the hardness of distinguishing the exponentiation $g^{T(\mathbf{s})}$ of a polynomial evaluation $T(\mathbf{s})$ from random, given exponentiations $g^{R_1(\mathbf{s})}, \ldots, g^{R_n(\mathbf{s})}$ of other polynomial evaluations. This condition holds in generic groups, and must

[5] An alternative formulation of the GGM is given by Maurer [44]; we follow Shoup's presentation in this paper. Relations and comparisons between different flavors of the GGM are discussed in the recent work [55].

therefore be satisfied by any concrete group that aims to "faithfully" instantiate them. However, the Uber assumption is far from the most general (standard-model) property that might hold in generic groups and thus should also be satisfied by their real-world counterparts.

Indeed, we observe that in a wide range of advanced cryptographic protocols and primitives (such as security under bad randomness, deterministic encryption, leakage resilience, and code obfuscation to name a few), inputs may not be uniformly distributed and polynomially related, but follow distributions that are, for example, only assumed to have high entropy. The Uber assumption can fall short of providing means to prove security of practical schemes in such settings. Accordingly, the main question we ask is:

Are there standard-model properties that generalize the Uber assumption and allow instantiating generic groups in a broad range of applications?

We emphasize that our treatment is practice-oriented in that we aim for a notion that captures standard-model properties of groups that can be used to establish the standard-model security of *existing*, *practical* protocols in a variety of models. Further, the new definitions should combine, as far as possible, standard-model analyses with the ease of use offered by the GGM.

In order to develop the core ideas one step at a time, in this work we treat the case of simple (non-bilinear) groups[6] and focus on decisional problems. There are indeed multiple directions in which our work can be extended; we will briefly discuss some of these at the end of the Introduction.

1.2 Our Approach

Our approach is inspired by an existing framework that bridges the standard and idealized models of computation: Universal Computational Extractors (UCEs) of Bellare, Hoang, and Keelveedhi (BHK) [10], a security notion for hash functions which, at a high level, requires indistinguishability from a random oracle under unpredictable inputs. Indeed, their motivation is in some sense conceptually analogous to ours. To that end, we seek to extend UCEs to structured ideal primitives, and call the resulting security notion in the case of cyclic groups *pseudo-generic groups* (PGGs). Before presenting PGGs, we give a brief overview of UCEs and refer to Section 2 for formal definitions.

Universal Computational Extractors. Let $\mathsf{H}\colon K \times D \to R$ be a keyed hash function. The UCE notion is defined via a game played by a source $\mathcal{S}$ and a distinguisher $\mathcal{D}$: Sample a challenge bit $b \twoheadleftarrow \{0,1\}$, a hash key $hk \twoheadleftarrow K$, and a RO $\rho\colon D \to R$. Then, $\mathcal{S}$ runs with access to an oracle Hash which, when queried on $x \in D$, returns $\mathsf{H}(hk, x)$ if $b = 1$, and $\rho(x)$ otherwise. Eventually, $\mathcal{S}$ outputs leakage L which is passed to $\mathcal{D}$, who is also given the hash key hk but no access to the hashing oracle. Distinguisher $\mathcal{D}$ must then guess b, and the requirement for H is that the advantage of every PPT $(\mathcal{S}, \mathcal{D})$ in this game is small.

[6] Similar work on the algebraic-group model (discussed later) was first carried out in simple groups [36] and later in bilinear ones [5].

Notice that for this definition to be meaningful, some restriction must be placed on $\mathcal{S}$ and $\mathcal{D}$: Without any additional requirement, $(\mathcal{S}, \mathcal{D})$ can win with overwhelming advantage by having $\mathcal{S}$ query HASH on any $x \in D$ with answer y, leak the pair (x, y) to $\mathcal{D}$, and then have $\mathcal{D}$ (who knows the hash key hk) check whether $y = \mathsf{H}(hk, x)$. To avoid such generic attacks, one requires $\mathcal{S}$ to be *unpredictable*, a notion formalized by asking that any predictor $\mathcal{P}$ have small advantage in the following game: Source $\mathcal{S}$ runs with access to a RO and produces leakage L. Then $\mathcal{P}$ runs on input L and wins if it can guess any of $\mathcal{S}$'s queries.

OUR NEW NOTION: PGG. To port the UCE definition to the context of cyclic groups, our first idea is to let a random group generator g play the role of the hash key, and to use exponentiation with base g in place of the hash.[7] The PGG security game for a group $(\mathsf{G}, \circ)$ then follows the UCE framework: Sample a secret bit $b \twoheadleftarrow \{0,1\}$, a random generator $g \twoheadleftarrow \mathsf{G}$, and a generic-group encoding $\sigma\colon \mathbb{Z}_p \to \mathsf{G}$. Then, a source $\mathcal{S}$ interacts with an exponentiation oracle EXP which, on input $x \in \mathbb{Z}_p$, returns the real group element g^x if $b = 1$, and a generic element $\sigma(x)$ otherwise. The source can pass some leakage L to a distinguisher $\mathcal{D}$, who is also given the generator g but loses access to the oracle and has to guess b. As for UCEs, the requirement for G is that every such $(\mathcal{S}, \mathcal{D})$ has a small advantage in this game. Thus, the PGG notion captures the intuition that if an adversary does not know the random generator g of G, exponentiation with base g looks like it returns random elements from G.

As before, for this notion not to be void we must put restrictions on the queries that $\mathcal{S}$ is allowed to make. First, observe that $\mathcal{S}$ must be unpredictable, because without any such requirement $(\mathcal{S}, \mathcal{D})$ can mount the attack for UCEs sketched above. We argue now that due to the presence of a group structure on G that was missing in the UCE setting, further conditions are needed.

ALGEBRAIC UNPREDICTABILITY. An important question now is for what sources is PGG achievable in principle, meaning there are no "trivial" attacks. Recall that for UCEs the answer was unpredictable sources. In our context, unpredictability alone is not sufficient: Consider a source $\mathcal{S}$ that samples $x_1, x_2 \twoheadleftarrow \mathbb{Z}_p$, queries $h_i \leftarrow \text{EXP}(x_i)$, and computes $x_3 \leftarrow x_1 + x_2$ and $h'_3 \leftarrow h_1 \circ h_2$. It then queries $h_3 \leftarrow \text{EXP}(x_3)$ and passes the bit $(h_3 = h'_3)$ to $\mathcal{D}$, who simply returns it. The advantage of $(\mathcal{S}, \mathcal{D})$ in the PGG game is almost 1, even though $\mathcal{S}$ is unpredictable since x_1 and x_2 are random. The issue is that $\mathcal{S}$ can place unpredictable queries that satisfy a known linear relation and distinguish by checking if the corresponding relation holds for the oracle replies. Excluding this trivial/generic attack motivates a more refined notion of unpredictability which we call *algebraic unpredictability*. In the corresponding game, the source $\mathcal{S}$ runs with access to the ideal exponentiation oracle while querying $x_1, \ldots, x_q$, and produces leakage L. Predictor $\mathcal{P}$ runs on input L and must guess a linear combination of the queries, i.e., outputs $(\alpha_0, \alpha_1, \ldots, \alpha_q)$ not all zero and wins if $\sum_{i=1}^{q} \alpha_i x_i = \alpha_0$.

[7] Recently, Bartusek, Ma, and Zhandry (BMZ) [4] studied the "fixed-generator" and "random-generator" settings in group-based assumptions. We necessarily work in the latter since, as we shall see, otherwise attacks arise.

This condition excludes the attack above, since $\mathcal{P}$ can output $(0, 1, 1, -1)$ to win the game. One might try to modify the attack and let the source leak (x_3, h_3) to $\mathcal{D}$, who, given g, can compute g^{x_3} and compare it to h_3. But this also contradicts algebraic unpredictability, with a predictor returning $(x_3, 1, 1)$.

As we shall see in Section 3, due to the existence of obfuscation-based attacks (similar to those for UCEs [21]), algebraic unpredictability must be *statistical* in nature; that is, we allow the algebraic predictors to run in unbounded time.

Parallel structure. It turns out that algebraic unpredictability by itself is not sufficient to rule out all generic attacks. Indeed, consider a source $\mathcal{S}$ that samples $x \twoheadleftarrow \mathbb{Z}_p$, queries $h_1 \leftarrow \textsc{Exp}(x)$ and $h_2 \leftarrow \textsc{Exp}(x^2)$, then computes $h_2' \leftarrow h_1^x$ and passes the bit $(h_2 = h_2')$ as leakage to $\mathcal{D}$, who decides accordingly. Again, the advantage of $(\mathcal{S}, \mathcal{D})$ in the PGG game is almost 1, and now $\mathcal{S}$ is even algebraically unpredictable. The issue here is that $\mathcal{S}$'s queries satisfy a linear relation with coefficients that are themselves *unpredictable but known to* $\mathcal{S}$ (in this case, $x \cdot x - 1 \cdot x^2 = 0$), an attack vector not ruled out by algebraic unpredictability.

To address this problem, we consider *parallel sources*. Loosely speaking, this means that $\mathcal{S}$'s Exp queries are made in parallel by single-query sources $\mathcal{S}_i(\mathsf{st})$ which, other than receiving a common initial state, do not pass state among each other. Indeed, the attack above was possible because $\mathcal{S}$ could learn more than one oracle reply. Note that, in this example, although the queries x and x^2 are allowed, the equality check $h_2 = h_2'$ requires knowledge of h_2 and h_2' (related to different queries), and hence violates the definition of a parallel source.

Restricted post-processing. Surprisingly, even considering only parallel sources does not rule out all trivial attacks. Indeed, one can modify the source $\mathcal{S}$ from above and make it parallel by setting $\mathsf{st} \leftarrow x$, having $\mathcal{S}_1(\mathsf{st})$ compute h_1 and h_2', and letting $\mathcal{S}_2(\mathsf{st})$ compute h_2. Leakage (h_2', h_2) is passed to $\mathcal{D}$, which returns the bit $(h_2 = h_2')$. This attack works because each $\mathcal{S}_i(\mathsf{st})$ still allows arbitrary post-processing of its oracle response (here, computing the exponentiation of h_1). Accordingly, we further restrict the class of sources and consider algebraically unpredictable *masking* sources, which are parallel sources where each $\mathcal{S}_i(\mathsf{st})$ is allowed only *structured post-processing* of its oracle replies (e.g., no post-processing or at most one group operation).

Simplification. The nature of the Exp oracle allows us to both strengthen and simplify our notion: We consider a definition of PGG whereby the distinguisher no longer receives the random generator g, and accordingly modify algebraic unpredictability to hold with $\alpha_0 = 0$ only. This new version implies the old one, as g can be obtained by querying 1. Second, algebraic unpredictability holds for this source, as the non-simplified version allows for non-zero α_0.

Generalizing Uber. We give a formal definition of the resulting notion in Section 3. Note that our notion can indeed be seen as a generalization of Uber, whereby the exponents are no longer evaluations of polynomials but may come from arbitrarily correlated distributions, as long as they adhere to the requirements set above. In particular, linear independence between polynomially induced exponents is now generalized to algebraically unpredictable sources.

GGM FEASIBILITY. Analogously to BHK who showed a RO is a UCE, we show the soundness of our definition by proving that a generic group is PGG for algebraically unpredictable masking sources. (We adopt Shoup's model for generic groups here [49].) This turns out to be significantly more involved than in BHK. Typically, GGM proofs appeal to the Schwartz–Zippel lemma to carry out a lazy sampling of group elements. In our proof, we no longer use this lemma and instead rely on the *algebraic unpredictability* of sources to carry out a consistent lazy sampling. Here we use a weaker notion of *computational* algebraic unpredictability. (There is no contradiction with obfuscation-based attacks, as generic groups do not have compact representations.) A second feature of our proof is that we allow our sources to depend on the entire function table of the group encoding. This choice more accurately models computationally unbounded sources in the standard model, widens the applicability of PGGs, and due to the existence of arbitrary leakage from source, also captures the effects of preprocessing attacks (aka. auxiliary information) on the definition. We use the recent technique of decomposition of high-entropy distribution due to Coretti, Dodis, and Guo [28] to handle unbounded sources.

Our GGM feasibility result, beside showing that PGGs do not suffer from structural weaknesses exposed by generic attacks, places PGGs below the GGM in the hierarchy of assumptions on groups (cf. the so-called "layered approach" to security explained in [10])[8]. Indeed, using this result, one can establish security of an application in the GGM by first proving it secure under an appropriate PGG assumption (in the standard model), and then lifting the result to the GGM using the result above.

Finally, equipped with GGM feasibility, it is reasonable to conjecture that appropriate elliptic curve groups are indeed PGGs, thus allowing the framework to be applied to a variety of practical cryptosystems built using such groups.

AVOIDING UNINSTANTIABILITY. We note that PGGs circumvent a variety of uninstantiability techniques. Notably, the classical CGH-type uninstantiability results [26,33] are avoided due to the fact that the group elements are computed wrt. *high-entropy* exponents. Furthermore, attacks due to the existence of various forms of obfuscation are avoided by requiring that the algebraic unpredictability notion be *statistical*. An analogous approach has been used in works on UCEs to avoid uninstantiability [21].

We also note that Zhandry's recent AGM uninstantiability result [55] inherently relies on the fact that an algebraic adversary has to return a representation of the forged tag (which then either breaks DLP or compresses random strings). This does not carry over to PGGs because adversaries are not required to be algebraic in our setting.

[8] The idea is to have assumptions and models organized into a hierarchy, where higher levels justify lower ones and, conversely, proving a scheme secure at some level shows that it meets higher ones as well. This allows us to identify precisely how strong an assumption is needed for a given application. Moreover, proving security of a scheme at a lower level typically gives more insight into its inner workings.

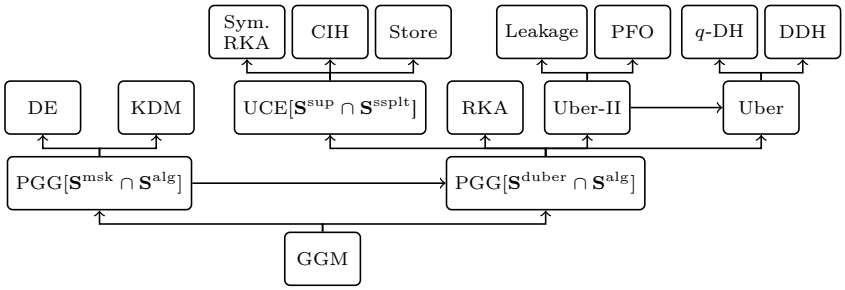

Fig. 1. Implications of PGGs. Here $\mathbf{S}^{\mathrm{ssplt}}$ denotes the class of simple split sources (see Section 6.2), CIH stands for correlated input hashing, and store for storage auditing protocols (see [10]). Results on DE (deterministic encryption), RKA and KDM or for ElGamal. Results for DE, RKA and UCE use LDDs if considering general sources.

1.3 Applications of Pseudo-Generic Groups

We demonstrate the applicability of our definition in three ways. First, we show that the Uber assumption holds in PGGs, thereby allowing us to recover all its applications within the PGG framework. For our second set of results, recall that there are several "advanced" security models for encryption, many of which have only been obtained via novel and often inefficient schemes. We demonstrate that PGGs enable proving (simple variants of) the classical *ElGamal encryption scheme* secure in a number of such advanced security models. According to the discussion above, this means that these notions can be safely assumed when ElGamal is implemented using suitable elliptic curve groups. Third, we show how to construct UCEs in PGGs. As before, this allows us to recover all their applications within the PGG framework. We refer to Figure 1 for a schematic overview of our results.

PGGs generalize Uber. We prove that broad and even novel formulations of the Uber assumption hold in PGGs. The Uber assumption family [18, 20] is an umbrella assumption that generalizes many hardness assumptions used to study the security of group-related schemes. Although it is commonly considered in bilinear groups, as previously mentioned, here we focus on simple groups. Nevertheless, proving that PGGs satisfy the Uber assumption allows us to recover all its applications within the PGG framework. For instance, all constructions whose security relies on the hardness of DDH (such as Diffie–Hellman key exchange, ElGamal encryption, and efficient PRFs [45]) or one of a number of closely related problems (e.g., q-DDHI, strong DDH, square DDH, and divisible DDH), or a randomized version of the recently introduced "Assumption 3" from [4], can be instantiated with PGGs. We further demonstrate that the Uber-II assumption, a variation of Uber with non-uniform exponents [24], also holds for PGGs. Specific instances of Uber-II have been used to build (composable) point-function obfuscation [14, 25] and leakage-resilient PKE schemes [31].

We believe that PGGs better highlight the types of problems one expects to be hard in groups, as it places no restriction on how the exponents are sampled beyond the fact that certain trivial attacks are ruled out.

Linear Dependence Destroyers. For some of our further implications below, we require a particular type of hash function with domain and range $\mathbb{Z}_p$ we call *linear dependence destroyer* (LDD). LDDs are defined via a game played by a source $\mathcal{S}$ and a predictor $\mathcal{A}$. Source $\mathcal{S}$ specifies a tuple of hash inputs $(x_1, \ldots, x_q)$ and state information st without seeing the random hash key hk, whereas $\mathcal{A}$ gets st and hk and returns a tuple of coefficients $(\alpha_0, \alpha_1, \ldots, \alpha_q)$. Adversary $(\mathcal{S}, \mathcal{A})$ wins if $\sum_{i=1}^{q} \alpha_i \cdot \mathsf{H}(hk, x_i) = \alpha_0$, and H is an LDD if every PPT $(\mathcal{S}, \mathcal{A})$ with $\mathcal{S}$ statistically unpredictable wins this game with negligible probability.

We show that the function $\mathsf{H}(hk, x) \coloneqq 1/(x + hk)$ implicit in the work of Goyal, O'Neill, and Rao [38] is an LDD when $\mathcal{S}$ is a *low-degree source*. These are sources that compute their outputs as evaluations of low-degree polynomials on points with sufficient entropy. This result, in turn, enables proofs of security for applications that use LDDs for low-degree sources. The main step in our proof is that different polynomials with random constant terms (given by the hash key) are likely to be coprime. When this is the case, the numerator of the fraction $\sum_{i=1}^{q} \alpha_i/(P_i(s_1, \ldots, s_m) + hk) - \alpha_0$ is non-zero no matter the choice of $(\alpha_0, \alpha_1, \ldots, \alpha_q)$. Winning the LDD game is thus equivalent to $(s_1, \ldots, s_m)$ being a root of this numerator, which is unlikely by the Schwartz–Zippel lemma.

In fact, we conjecture that H is an LDD for *all* statistically unpredictable sources, not just for low-degree ones. To further lend plausibility to this notion, we also prove that a random function is an LDD, under mild restrictions on $\mathcal{S}$. To this end, we apply the compression technique originating from Gennaro and Trevisan [37] in a setting where *two* independent parties have full access to the code of the ideal object. The compression technique is commonly used in cryptography, and our extension may be of wider interest.

UCEs for split sources. A natural question is how PGGs relate to the notion of UCEs. It seems that PGGs are harder to build because they have more structure. In other words, generic groups, which PGGs instantiate, seem stronger than random oracles, which UCEs instantiate. As our first application we show that, indeed, UCEs can be constructed from PGGs for dUber sources and LDDs. The constructed UCE is for statistically unpredictable split sources. A number of applications of UCEs, such as proofs of storage, correlated-input secure hashing, and RKA security for symmetric encryption, only rely on UCE for split sources. We note that a benefit of building UCEs from PGGs is that the construction may enjoy useful algebraic properties that constructions from symmetric-key primitives do not. Once again, in the generic-group model, we show security against preprocessing attacks.

Key-dependent message security for ElGamal. Second, we show that PGGs enable proof of key-dependent message (KDM) security for a slightly tweaked version of ElGamal [16, 23]. KDM security for ElGamal does not seem to be feasible using Uber (though less efficient constructions do exist, e.g., [3,19]).

Application	PGG Source	Other Assumptions
Uber & Uber-II	dUber	–
RKA for ElGamal	dUber	LDD
KDM for ElGamal	Mask	–
Low-degree DE ElGamal	Mask	–
UCE for split sources	dUber	LDD
General DE for ElGamal	Mask	LDD

Table 1. Overview of applications of PGGs.

The KDM notion that we prove does not allow for adaptive queries, but it permits deriving key-dependent messages in an inefficient way. Furthermore, when combined with our GGM feasibility, we obtain KDM security against preprocessing attacks in the GGM.

Hash-then-ElGamal deterministic PKE. Moving on, we prove that ElGamal admits full instantiation of its corresponding random-oracle-model Encrypt-With-Hash (EwH) deterministic encryption scheme [6], which replaces the coins in encryption with the hash of the message. Here we need that the hash function is an LDD. Preprocessing attacks are also accounted for in our definition and analysis. Note that a prior result of BHK [10] also implies security of ElGamal-based EwH, but uses an assumption on the hash that makes the result arguably tautological. It is also known how to instantiate EwH for schemes meeting "lossiness" assumptions [9, 40]. Our result is the first that does not require such an assumption as it shifts the security assumption with non-uniform inputs from the hash function to the underlying group.

Related-key security. We also show that ElGamal offers a form of related-key attack (RKA) security, whereby secret keys (and their corresponding public keys) are generated from related random coins. RKA security was systematically studied by Bellare, Cash, and Miller [8] for PKEs. Under PGGs, and assuming LDDs (which for polynomially induced sources we show to exist) we can handle unpredictable related-key deriving functions that are claw-free (or more generally as long as the repetition pattern of secret keys does not affect unpredictability).

We summarize the above applications in Table 1. For each application, we record what type of source class is used in the reduction and whether the additional assumption of LDD is needed. For applications requiring LDDs we note that our results are modular wrt. the underlying source class. This means that for whatever source class we achieve LDDs, we also obtain an end application wrt. a corresponding source class. For example, low-degree LDDs (which we achieve unconditionally) translate to instantiations of UCEs and deterministic encryption wrt. low-degree sources and RKA security for low-degree related keys. The latter includes affine functions, which are often considered in the RKA literature.

We envision that several other security goals are also feasible under PGGs, of which we consider only a representative sample. Examples include security under bad randomness [7], joint RKA and KDM security [17], randomness-dependent

message security [13], related-randomness security [47], and more generally application scenarios whereby the input distributions are not necessarily random and only guaranteed to come from high entropy distributions.

1.4 Other Related Work and Discussions

Public-seed pseudorandom permutations. Soni and Tessaro [51] define a UCE-like, standard-model notion for random (two-sided) permutations called public-seed pseudorandom permutations (psPRPs). They provide constructions of UCEs from psPRPs (for a variety of sources) by showing, for example, that the five-round Feistel [51] and the more efficient Naor–Reingold construction [52] yield psPRPs when the round functions are UCEs. Our work continues these lines of research by extending the UCE approach to defining security from random oracles and random permutations to generic groups.

Algebraic group model. An intriguing notion that has recently received considerable attention is the algebraic group model (AGM) [5,36]. We observe that the AGM places restrictions on adversaries that are qualitatively different compared to PGGs: Algebraic adversaries must output a representation of returned group elements, which makes the AGM a powerful and useful model since this additional information allows to carry out certain reductions.[9] Restrictions on PGG adversaries on the other hand are of standard-model type.

Nevertheless, it would be interesting to study the relation between the two notions and also to knowledge assumptions. Following work on instantiating UCEs [22] and on constructing groups in which the AGM can be realized and the Uber assumption holds [1,2,41], another goal for future work is to construct PGGs from well-known assumptions (such as iO, dual-mode NIZKs, FHE, etc.).

Extensions of PGGs. In this work, we develop the necessary techniques and set the stage for the pseudo-generic approach to group-related assumptions. In doing so, we leave a number of directions for future research.

A natural extension to our work would be to formulate analogous PGG-type notions for bilinear groups (as considered by Boyen for the Uber assumption [20] and extended via matrix DDH in [34]) or multi-linear groups. We anticipate further applications of this notion, as in the bilinear setting a host of schemes are only known to have a proof in the GGM and may be provable in PGGs.

The matrix DDH assumption (MDDH) [34] considers matrix-vector multiplication in the exponent in multi-linear groups, where a matrix is sampled from a general distribution and the vector is uniform. However, this assumption is only studied for polynomially induced distributions. As such, MDDH is not a generalization of Uber in the sense of PGG to arbitrary distributions.

Certain applications require assumptions that lie beyond the reach of PGGs as currently formulated. PGGs do not capture applications where exponents may depend on a group generator that is not random (as, for example, in recent work

[9] We note that our understanding of the role played by the AGM in assessing the hardness of group-related assumptions is evolving in light of recent works [42,55].

on non-malleable point-function obfuscation [4, 35, 43]). The PGG framework also does not capture interactive [1] or knowledge-type [12] assumptions.

1.5 Structure of the Paper

In Section 2 we define the basic notation and recall the definition of UCEs. Section 3 contains our definitional contributions, where we define pseudo-generic groups, algebraically unpredictable and masking sources, and discuss the choices made in devising these notions. In Section 4 we prove that a generic group is a PGG, and then introduce LDDs and a candidate construction in Section 5. Section 6 contains the applications of PGGs. In Section 6.1 we show that an entropic variant of the decisional Uber assumption (and thus many implications thereof) holds in PGGs. Afterwards, we show how to apply PGG directly to the analysis of cryptosystems, by proving that PGGs and LDDs can be used to build UCEs for (simple) split sources. Further applications of PGGs are presented in the full version of the paper.

2 Preliminaries

Basic notation. If $n \in \mathbb{N}$, we write $[n]$ for the set $\{1, \ldots, n\}$. Unless otherwise stated, an integer $p \in \mathbb{Z}$ is assumed to be prime, and we let $\mathbb{Z}_p$ denote the field of integers modulo p. We denote the set of all bit strings of finite length by $\{0,1\}^*$, and the empty string by ε. We use boldface characters $\mathbf{x} \coloneqq (x_1, \ldots, x_n)$ to denote vectors and write $\mathbf{x}[i]$, with $i \in [n]$, to denote the ith entry x_i of $\mathbf{x}$. By $x \twoheadleftarrow \mathcal{S}$ we mean sampling x according to distribution $\mathcal{S}$. Similarly, $x \twoheadleftarrow S$ means sampling x uniformly at random from a finite set S. The cardinality of a set S is denoted $|S|$. We let $L \leftarrow [\,]$ denote initializing an ordered list to empty, and $L : x$ denote appending an element x to the list L. A table T is a list of pairs (x, y), and we write $T[x] \leftarrow y$ to mean that the pair (x, y) is appended to T. We let $\mathsf{Dom}(T)$ denote the set of all values x such that $(x, y) \in T$ for some y, and similarly $\mathsf{Rng}(T)$ denote the set of all values y such that $(x, y) \in T$ for some x. For two sets D and R we denote by $\mathrm{Fun}(D, R)$ and $\mathrm{Inj}(D, R)$ the set of all functions and the set of all injections from D to R, respectively. When $|D| = |R|$, an injection is also a bijection.

Min-entropy. The min-entropy of a random variable $\mathcal{X}$ over a domain D is $\mathbf{H}_\infty(\mathcal{X}) \coloneqq -\log \max_{x \in D} \Pr[\mathcal{X} = x]$. $\mathcal{X}$ is called a k-source if $\mathbf{H}_\infty(\mathcal{X}) \geq k$.

Polynomials and rational functions. We let $\mathbb{F}[X_1, \ldots, X_m]$ be the ring of polynomials in $m \in \mathbb{N}$ variables over a field $\mathbb{F}$, and $\mathbb{F}(X_1, \ldots, X_m)$ be the field of rational functions of the form $R(X_1, \ldots, X_m) = \hat{R}(X_1, \ldots, X_m)/\check{R}(X_1, \ldots, X_m)$, with $\hat{R}, \check{R} \in \mathbb{F}[X_1, \ldots, X_m]$ and $\check{R} \neq 0$. Here, as usual, $\hat{R}$ is called the numerator and $\check{R}$ the denominator of R. If $0 \neq R \in \mathbb{F}[X_1, \ldots, X_m]$ is a polynomial, we denote its total degree by $\deg(R)$. We extend this notation to rational functions via $\deg(R) \coloneqq \deg(\hat{R}) - \deg(\check{R})$ for every $0 \neq R = \hat{R}/\check{R}$. Observe that the degree

of a rational function is well-defined, since it does not depend on its representation as a fraction of polynomials. Finally, if $R_1, \ldots, R_n \in \mathbb{F}(X_1, \ldots, X_m)$ such that $R_i \neq 0$ for every $i \in [n]$, we let $\deg(R_1, \ldots, R_n) \coloneqq \max_{i \in [n]}\{\deg(R_i)\}$.

Linear dependence. Let $R_1, \ldots, R_n, T \in \mathbb{F}(X_1, \ldots, X_m)$. We say that T is linearly dependent on $R_1, \ldots, R_n$ (over $\mathbb{F}$) if there exist $a_1, \ldots, a_n \in \mathbb{F}$ such that $T(X_1, \ldots, X_m) = \sum_{i=1}^{n} a_i \cdot R_i(X_1, \ldots, X_m)$.

Hash function families. A hash function family is a tuple of PPT algorithms $\mathsf{H} \coloneqq (\mathsf{H.Setup}, \mathsf{H.KGen}, \mathsf{H.Eval})$. Here, algorithm $\mathsf{H.Setup}(1^\lambda)$ outputs a tuple π containing the descriptions of valid domain and range points D and R, as well as a key space K and other system-wide parameters. Algorithm $\mathsf{H.KGen}(\pi)$ is the hash key generation algorithm which returns a key $hk \in K$. The evaluation algorithm $\mathsf{H.Eval}(\pi, hk, x)$, called on a hash key hk and a domain point $x \in D$, outputs a point $y \in R$. To help readability, by slight abuse of notation we will simply write $\mathsf{H}(hk, x)$ in place of $\mathsf{H.Eval}(\pi, hk, x)$.

Remark. Our definition of hash function families augments the usual syntax with a setup algorithm $\mathsf{H.Setup}$. Accordingly, we will extend the UCE definition to incorporate system parameters. Overloading notation, we allow $\mathsf{H.Setup}$ to alternatively take the description of a domain D and a range R as inputs, and let it return corresponding parameters π.

Universal computational extractors [10]. Let H be a hash function family. The advantage of a pair of PPT adversaries $(\mathcal{S}, \mathcal{D})$ (called UCE source and UCE distinguisher) in the UCE game for H is defined as

$$\mathrm{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) \coloneqq 2 \cdot \Pr[\mathrm{UCE}^{\mathcal{S},\mathcal{D}}_{\mathsf{H}}(\lambda)] - 1\,,$$

where the UCE game is defined in Figure 2 (left). We say that H is UCE[$\mathbf{S}$] secure, if the advantage of any PPT $(\mathcal{S}, \mathcal{D})$ with $\mathcal{S} \in \mathbf{S}$ in the UCE game for H is negligible. This is usually written as $\mathsf{H} \in \mathrm{UCE}[\mathbf{S}]$.

Without any restriction on the class of sources $\mathbf{S}$, the UCE notion of security is unachievable [10]. BHK exclude trivial attacks by requiring that the source be *unpredictable*, meaning that it is hard to predict any of its oracle queries when observing the leakage L. Due to the obfuscation-based attack of [21], the flavor of unpredictability needs to be statistical. We recall the formal definition below.

(Statistically) unpredictable sources [10, 21]. Let H be a hash function family and $\mathcal{S}$ a UCE source. We define the advantage of a (possibly unbounded) adversary $\mathcal{P}$ (called predictor) in the predictability game against $(\mathsf{H}, \mathcal{S})$ as

$$\mathrm{Adv}^{\mathrm{pred}}_{\mathsf{H},\mathcal{S},\mathcal{P}}(\lambda) \coloneqq \Pr[\mathrm{Pred}^{\mathcal{P}}_{\mathsf{H},\mathcal{S}}(\lambda)]\,,$$

where the game Pred is defined in Figure 2 (center). A source $\mathcal{S}$ is called statistically unpredictable if the above advantage is negligible for any (possibly unbounded) predictor $\mathcal{P}$. The class of all statistically unpredictable sources is denoted $\mathbf{S}^{\mathrm{sup}}$. We say that H is UCE secure if it is UCE[$\mathbf{S}^{\mathrm{sup}}$] secure.

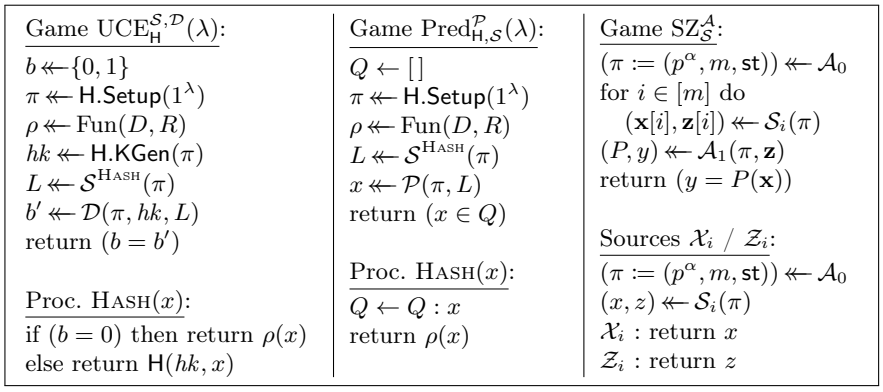

Fig. 2. Left: The UCE game. **Center**: The unpredictability game. **Top right**: The Schwartz–Zippel game. **Bottom right**: The sources $\mathcal{X}_i$ and $\mathcal{Z}_i$.

Schwartz–Zippel Lemma. We now recall the Schwartz–Zippel Lemma [32,48, 57], a simple yet powerful tool to bound the probability of finding a root of a non-zero polynomial when evaluating it at a random point. We also generalize the standard Schwartz–Zippel lemma and obtain a more general and game-based version of this result. In this variant, the points can be chosen according to distributions with enough min-entropy, and the polynomial picked given some leakage. This version may be more suitable for use in a cryptographic setting. A proof of the game-based Schwartz–Zippel lemma can be found in the full version of the paper.

Lemma 1 (Schwartz–Zippel). *Let $\alpha, p \in \mathbb{N}$ with p prime, $S \subseteq \mathbb{F}_{p^\alpha}$, and let $0 \neq P(X_1, \ldots, X_m) \in \mathbb{F}_{p^\alpha}[X_1, \ldots, X_m]$. Then*

$$\Pr_{x_1,\ldots,x_m \twoheadleftarrow S}[P(x_1, \ldots, x_m) = 0] \leq \frac{\deg(P)}{|S|}\,.$$

Lemma 2 (Game-based Schwartz–Zippel). *Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be a two-stage algorithm, where $\mathcal{A}_0$ takes no input and returns a set of public parameters $\pi := (p^\alpha, m, \mathsf{st}) \in \mathbb{N}^2 \times \{0,1\}^*$ with $\alpha, p \in \mathbb{N}$ and p prime, and $\mathcal{A}_1$ takes π and values $z_1, \ldots, z_m \in \{0,1\}^*$ as input and returns a non-constant polynomial $P(X_1, \ldots, X_m) \in \mathbb{F}_{p^\alpha}[X_1, \ldots, X_m]$ with $\deg(P) \leq d \in \mathbb{N}$ and a value $y \in \mathbb{F}_{p^\alpha}$. Let $\mathcal{S} := \{\mathcal{S}_i\}_{i\in\mathbb{N}}$ be a family of sources, each taking π as input and returning values $(x, z) \in \mathbb{F}_{p^\alpha} \times \{0,1\}^*$. Then*

$$\Pr[\mathrm{SZ}^{\mathcal{A}}_{\mathcal{S}}()] \leq d \cdot \mathop{\mathbb{E}}_{\substack{\pi \twoheadleftarrow \mathcal{A}_0,\\ (x_1,z_1) \twoheadleftarrow \mathcal{S}_1(\pi),\ldots,(x_m,z_m) \twoheadleftarrow \mathcal{S}_m(\pi)}} \left[\frac{1}{2^{\min_{i\in[m]}\{\mathbf{H}_\infty(\mathcal{X}_i \mid (\mathcal{A}_0=\pi)\wedge(\mathcal{Z}_i=z_i))\}}}\right],$$

where the game $\mathrm{SZ}^{\mathcal{A}}_{\mathcal{S}}()$ is defined in Figure 2 (top right), and the sources $\mathcal{X}_i$ and $\mathcal{Z}_i$ are given in Figure 2 (bottom right).

Observe that if $m = 1$, the expectation above is the prediction probability of $\mathcal{X}$ given $\mathcal{A}_0$ and $\mathcal{Z}$. In general, the minimum cannot be taken out of the expectation, because it reflects $\mathcal{A}$'s choices of which variables appear in P.

3 Pseudo-Generic Groups

We now formally define pseudo-generic groups (PGGs), where group elements are required to be indistinguishable from random, as long as their exponents satisfy a specific unpredictability condition. PGGs lift the definition of UCEs of Bellare, Hoang, and Keelveedhi [10] from the setting of hash functions to that of groups. In other words, the underlying ideal object in the PGG definition, from which a concrete group is supposed to be indistinguishable, is a generic group rather than a random oracle. We start by giving some background on computational group schemes and generic groups, and then proceed to defining PGGs.

Computational group schemes [30]. A computational group scheme is a randomized algorithm Γ which, on input the security parameter 1^λ, outputs group parameters π consisting of a group operation $\circ$, an arbitrary group generator g, and a prime group order $p \in [2^{\lambda-1}, 2^\lambda)$. Implicit in these parameters is a set G such that $(\mathsf{G}, \circ)$ forms a cyclic group of order p with generator $g \in \mathsf{G}$. We write the sampling of group parameters as $(\pi := (\circ, g, p)) \twoheadleftarrow \Gamma(1^\lambda)$, with the understanding that π implicitly defines the underlying set G. As usual, the group operation gives rise to an exponentiation algorithm $\mathsf{exp}(h, x)$ whose output is denoted as h^x. We will often omit explicitly writing the operation $\circ$.

Generic groups [44, 46, 50]. Given a group $(\mathsf{G}, \circ)$ of prime order p, the generic group on G is the uniform distribution over $\mathrm{Inj}(\mathbb{Z}_p, \mathsf{S})$, where $\mathsf{S} \subseteq \{0,1\}^*$ with $|\mathsf{S}| \geq p$. Recall that every map $\tau \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{S})$ allows one to define an associated group operation $\mathsf{op}\colon \mathsf{S} \times \mathsf{S} \to \mathsf{S} \cup \{\perp\}$ via $\mathsf{op}(h_1, h_2) := \tau(\tau^{-1}(h_1) + \tau^{-1}(h_2))$ if $h_1, h_2 \in \mathsf{Rng}(\tau)$, and $\mathsf{op}(h_1, h_2) := \perp$ otherwise. The generic group model is a model of computation in which all parties, honest or otherwise, are run on inputs p and encodings of application-specific elements, and have oracle access to a random encoding $\tau \in \mathrm{Inj}(\mathbb{Z}_p, \mathsf{S})$ and its associated operation oracle op.

Pseudo-generic groups. Let Γ be a computational group scheme. We define the advantage of a pair of adversaries $(\mathcal{S}, \mathcal{D})$ (called PGG source and PGG distinguisher) in the PGG game for Γ as

$$\mathrm{Adv}^{\mathrm{pgg}}_{\Gamma,\mathcal{S},\mathcal{D}}(\lambda) := 2 \cdot \Pr[\mathrm{PGG}^{\mathcal{S},\mathcal{D}}_{\Gamma}(\lambda)] - 1\,,$$

where the PGG game is defined in Figure 3 (left).[10] We say that Γ is PGG[$\mathbf{S}$] secure if the advantage of any $(\mathcal{S}, \mathcal{D})$ with $\mathcal{S} \in \mathbf{S}$ and $\mathcal{D}$ a PPT algorithm in the PGG game is negligible. We denote this as $\Gamma \in \mathrm{PGG}[\mathbf{S}]$.

Recall from our earlier discussion that, similarly to UCEs and psPRPs, this notion of security is not achievable without restrictions on the class of PGG

[10] Note that σ can be lazily sampled, so that the game runs in polynomial time.

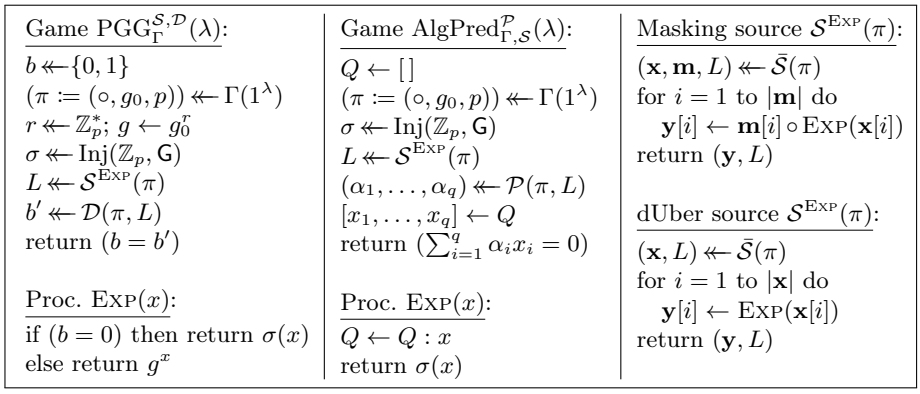

Fig. 3. Left: The PGG game. **Center**: The algebraic unpredictability game. **Top right**: A generic masking source. **Bottom right**: A generic dUber source.

sources $\mathbf{S}$. As a first step towards excluding trivial attacks, we introduce the notion of *algebraic unpredictability*, the core definition which allows us to extend UCE-type security notions beyond unstructured primitives (like hash functions and permutations). We require that no predictor be able to guess a non-trivial linear combination between the points queried by the source, as formalized below.

Algebraically unpredictable sources. Let Γ be a computational group scheme and $\mathcal{S}$ a PGG source, and assume that the leakage L produced by $\mathcal{S}$ encodes the number of Exp queries made by $\mathcal{S}$. We define the advantage of a (possibly unbounded) algorithm $\mathcal{P}$ (called predictor) in the algebraic unpredictability game against $(\Gamma, \mathcal{S})$ as

$$\mathrm{Adv}^{\text{alg-pred}}_{\Gamma,\mathcal{S},\mathcal{P}}(\lambda) := \Pr[\mathrm{AlgPred}^{\mathcal{P}}_{\Gamma,\mathcal{S}}(\lambda)]\,,$$

where the game AlgPred is defined in Figure 3 (center). We require that the output of $\mathcal{P}$ be different from the trivial all-zero tuple. A source $\mathcal{S}$ is called statistically algebraically unpredictable if the above advantage is negligible for any (possibly unbounded) predictor $\mathcal{P}$. We denote the class of all statistically algebraically unpredictable sources by $\mathbf{S}^{\mathrm{alg}}$. Observe that any such source must output distinct points (with high probability).

Masking and dUber sources. Algebraic unpredictability turns out to be insufficient to rule out all trivial attacks, as explained earlier. We thus restrict the set of sources for which we require PGG security even further and consider the class $\mathbf{S}^{\mathrm{msk}}$ of *masking* sources. These are sources $\mathcal{S}$ for which there exists a (possibly unbounded) auxiliary algorithm $\bar{\mathcal{S}}$ with polynomially bounded output, such that $\mathcal{S}$ takes the form in Figure 3 (top right). Here, $\bar{\mathcal{S}}$ returns vectors $\mathbf{x} \in \mathbb{Z}_p^q$ and $\mathbf{m} \in \mathsf{G}^q$ of the same length, and leakage L. Source $\mathcal{S}$ then queries Exp on all entries of $\mathbf{x}$, and multiplies the replies with the corresponding elements from $\mathbf{m}$. We also define the subclass $\mathbf{S}^{\mathrm{duber}} \subseteq \mathbf{S}^{\mathrm{msk}}$ of *distributional Uber* (dUber) sources, as shown in Figure 3 (bottom right), where we require $\mathbf{m} = (1_{\mathsf{G}}, \ldots, 1_{\mathsf{G}})$. To

simplify notation, we define sources $\mathcal{S}$ in these classes via their corresponding auxiliary algorithms $\bar{\mathcal{S}}$, and call them auxiliary masking (resp., dUber) sources. Notice that masking and dUber sources always reveal the number of Exp queries through their leakage via the length $|\mathbf{y}|$ of $\mathbf{y}$.

Focusing on masking sources, and dUber sources in particular, provides a new perspective to our contribution. Indeed, dUber sources generalize the adversary in the Uber assumption insofar as oracle queries are no longer obtained as polynomial evaluations on a product distribution, but from a general distribution. To avoid trivial attacks, the target polynomial in the Uber assumption must be linearly independent from the other ones, a requirement covered by algebraic unpredictability in this setting.

PGG security. We say that a computational group scheme Γ is PGG secure if it is PGG$[\mathbf{S}^{\mathrm{alg}} \cap \mathbf{S}^{\mathrm{msk}}]$ secure. In order to establish confidence in this notion and show that it hides no other obvious structural weaknesses, we prove in Section 4 that PGG security is indeed achievable in the generic-group model. However, we note that, for specific applications, PGG security with respect to subclasses of $\mathbf{S}^{\mathrm{alg}} \cap \mathbf{S}^{\mathrm{msk}}$ may be sufficient. On the other hand, there may exist larger, or even incomparable source classes for which PGG security is also feasible.[11]

Definitional choices. Observe that, in the PGG game, the randomized group generator g plays the role of the hash key hk in the UCE game. The fact that g remains hidden from the source prevents it from trivially winning the PGG game by sampling $x \twoheadleftarrow \mathbb{Z}_p$, querying $h \leftarrow \mathrm{Exp}(x)$, and checking if $g^x = h$. Similarly, g (or r) cannot be given to the distinguisher $\mathcal{D}$, since the source could query $h \leftarrow \mathrm{Exp}(1)$, leak h to $\mathcal{D}$, who then checks if $g = h$ (resp., $g_0^r = h$).

Note also that the random injection σ that the game samples has G (the real group), and not some larger set S, as its range. This is needed because the source can check group elements for validity (e.g., using exponentiation to power $p-1$, or directly via an element validity algorithm if such a procedure is available).

Also observe that the source does not get oracle access to the operation op defined by σ. The reason is that, with such access, once again trivial attacks arise: The source samples two random group elements, then multiplies them first using the op oracle and then again locally using the input group operation $\circ$, and finally checks if the results match.[12] Removing access to the operation oracle from $\mathcal{S}$ does not restrict our ability to prove security results in the PGG model, as we shall see in Section 6.

Computational algebraic unpredictability. In [21], Brzuska, Farshim and Mittelbach demonstrate an attack against UCEs with respect to a *computational* notion of unpredictability. The types of sources that we consider for PGG are analogous to the so-called split UCE sources. As BFM discuss, their iO-based attack does not extend to such sources. However, under the existence of

[11] For instance, one could allow for more expressive forms of post-processing. However, we have not yet been able to find applications of this wider class of sources.

[12] On the other hand, it is unclear how to rule this attack out using an extended notion of algebraic unpredictability that takes operation queries into account.

a plausible form of obfuscation, attacks arise. In more detail, if the function mapping x to the obfuscation of the circuit $C[x]\colon h \mapsto h^x$ is one-way, the following attack emerges: The dUber source picks $x \twoheadleftarrow \mathbb{Z}_p$, defines $\mathbf{x} \leftarrow (1, x)$, and sets L to be an obfuscation of $C[x]$. The distinguisher then returns $(\mathbf{y}[2] = L(\mathbf{y}[1]))$. For this reason we focus on statistical algebraic unpredictability.

Despite this attack, there *is* a benefit in considering a computational notion of algebraic unpredictability when it comes to the analysis in idealized models. Indeed, as we show, PGG with respect to this wider class of sources is achievable in the GGM, and thus a wider class of applications can be proven secure in the GGM. This does not contradict potential security in the standard model since PGG with respect to computational algebraic unpredictability may still exist for sources that take specific forms.

Multi-base PGGs. For the UCE and psPRP notions, BHK and ST respectively considered multi-key extensions to cover a wider range of applications. These notions are not known to be equivalent to their simpler single-key counterparts. For pseudo-generic groups, on the other hand, a simple generator re-randomization argument shows that the multi-base and single-base notions are equivalent. We thus focus on the (single-base) PGG version above.

4 Generic Groups are PGGs

In this section we show the feasibility of PGGs in the generic-group model. The importance of this result is that it rules out generic attacks against the PGG notion, thus forming a check on the soundness of the definitional framework. Furthermore, it automatically lifts the security of each of the applications of PGGs that we consider to the GGM, as long as algebraically unpredictable, masking sources are used. This is similar to the Uber assumption family, where one relies on a specific assumption within Uber, and *reuses* the GGM hardness proved once for the whole family. As discussed above, we show GGM hardness of PGGs for a *computational* notion of algebraic unpredictability, which widens its applicability.

Definitional choices. Before stating our result, we clarify what it means for a generic group to be PGG secure. The PGG and AlgPred games in the GGM for a group of size p with target set S are presented in Figures 4 (left) and 4 (center), masking sources are given in Figure 4 (right). We stress that the oracles τ and op defining the generic group and its operation are *independent* of the injection σ used to define PGG security—only the ranges of the two encodings coincide, since, as in the standard model, σ must take values in the group (which is given by $\mathsf{Rng}(\tau)$ in the GGM). Advantage terms are defined as usual.

Recall that masking sources can be unbounded, which means that they are allowed an unlimited amount of group operations. Following [4], we mirror this in the GGM by giving $\mathcal{S}$ the entire function table of τ. Distinguisher $\mathcal{D}$ on the other hand is bounded, which means that it is only given oracles for τ and op and that the leakage L must be short. This choice of modeling more accurately

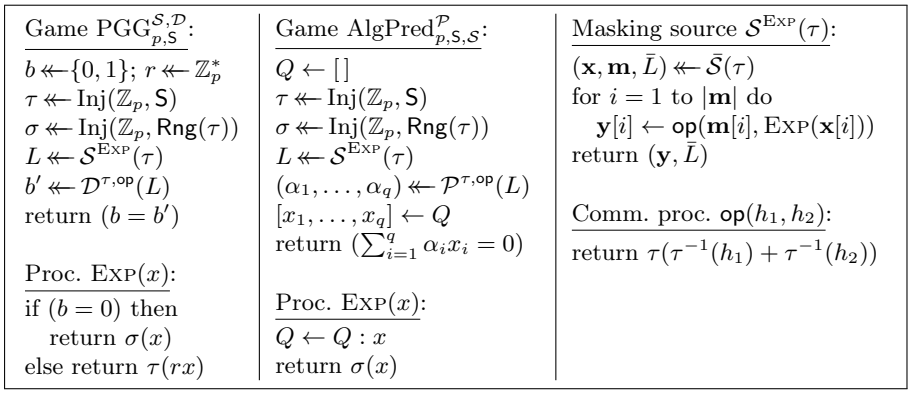

Fig. 4. Left: The PGG game in the GGM. **Center**: The AlgPred game in the GGM. **Right**: A masking source $\mathcal{S}$ in the GGM. In all games, $|\mathsf{S}| \geq p$, and without loss of generality, all algorithms know p.

reflects unbounded sources in the GGM by allowing an arbitrary number of group operations. Furthermore, it allows us to derive security in the presence of preprocessing attackers, as our sources can leak information about τ to the distinguisher.[13]

We are now ready to state the main result of this section. We only give an overview of the proof here, and provide the formal details in the full version.

Theorem 1 (GGM feasibility). *Let $(\mathsf{G}, \circ)$ be a group of order p, and S a set with $|\mathsf{S}| \geq p$. Then the generic group on G is $\mathrm{PGG}[\mathbf{S}^{\mathrm{alg}} \cap \mathbf{S}^{\mathrm{msk}}]$ secure. More precisely, for every adversary $(\mathcal{S}, \mathcal{D})$ in the PGG game with $\mathcal{S} \in \mathbf{S}^{\mathrm{alg}} \cap \mathbf{S}^{\mathrm{msk}}$, there exists a predictor $\mathcal{P}$ in the game AlgPred such that*

$$\mathrm{Adv}^{\mathrm{pgg}}_{p,\mathsf{S},\mathcal{S},\mathcal{D}} \leq \mathcal{O}\left(T^2 \cdot \mathrm{Adv}^{\mathrm{alg}}_{p,\mathsf{S},\mathcal{S},\mathcal{P}} + \sqrt{\frac{ST^2}{p}}\right),$$

where $S := 2q_{\mathcal{S}}(\lfloor \log p \rfloor + 1) + \ell + \lfloor \log q_{\mathcal{S}} \rfloor + \lfloor \log \ell \rfloor + 2$ and $T := q_{\mathcal{S}} + q_{\mathcal{D},\tau} + q_{\mathcal{D},\mathsf{op}}$. Here $q_{\mathcal{S}}$, $q_{\mathcal{D},\tau}$, and $q_{\mathcal{D},\mathsf{op}}$ are upper bounds on the number of queries made by $\mathcal{S}$ and $\mathcal{D}$ to their respective oracles, ℓ is an upper bound on the length of the leakage $\bar{L}$ returned by $\mathcal{S}$, and we assume $T \leq \sqrt{Sp}$.

Proof Overview. Fix any $(\mathcal{S}, \mathcal{D})$ as in the statement of the theorem. Without loss of generality, assume that $\mathcal{S}$ always returns exactly S bits (this can be achieved by padding the leakage $\bar{L}$ returned by $\mathcal{S}$). We use the game-playing framework and consider the following sequence of games:

[13] In particular, this model allows a restricted class of sources that leak arbitrary information (without any unpredictability requirements), as long as the sampling of the exponents is unpredictable (e.g., random, as is the case for the DLP).

Game_0 is the PGG game for $(\mathcal{S}, \mathcal{D})$ with respect to $b = 1$, i.e., where the oracle Exp uses the generic group injection τ.

Game_1 is the same as Game_0, but we additionally require that the queries $\mathcal{S}$ makes to the Exp oracle be all distinct. The distinguishing probability can be bounded by reducing to the algebraic unpredictability of a predictor that picks two coordinates i and j of $\mathbf{y}$ at random, and returns the zero vector with ± 1 at positions i and j.

Game_2 is the same as Game_1, but we use the "bit-fixing lemma" [4, Lemma 9] (with $\gamma := 1/p$) to resample τ right after the execution of $\bar{\mathcal{S}}$, whose output is treated as leakage. Also, we do not sample the new injection all at once, but implement it via lazy sampling. The loss in this transition is given by Lemma 9 in [4].

Game_3 is the same as Game_2, but we replace the randomly chosen exponent $r \twoheadleftarrow \mathbb{Z}_p^*$ with a formal variable R. We also start to lazy sample the new encoding σ with the values returned by Exp. The last two games are only different if, at the end of the execution of $\mathcal{D}$ in Game_3, $\mathcal{D}$ has queried two different polynomials that coincide when evaluated on a random $r \in \mathbb{Z}_p^*$, or if either $\mathcal{S}$ or $\mathcal{D}$ have made a query that belongs to the set of fixed points. All these events are bounded by the Schwartz–Zippel lemma: It is at this step that we use the fact that PGG is defined wrt. a random group generator.

Game_4 is the same as Game_3, but for every Exp query $\mathbf{x}[j]$, instead of saving an entry for $R\mathbf{x}[j]$ to the encoding table, we index it with a different and independent variable Z_j. This game is indistinguishable from the preceding one unless, at the end of the execution of $\mathcal{D}$ in Game_4, $\mathcal{D}$ has queried two distinct polynomials that coincide when evaluated on $(R\mathbf{x}[1], \ldots, R\mathbf{x}[q])$. Any such collision yields a non-trivial linear relation among the entries of $\mathbf{x}$. It is at this step that we appeal to the algebraic unpredictability of the source $\mathcal{S}$.

Game_5 is the same as Game_4, but we evaluate the variables Z_j at random values $c_j \twoheadleftarrow \mathbb{Z}_p$. The two games are close up to a Schwartz–Zippel term.

Game_6 is the same as Game_5, but we insist that the values c_j be pairwise distinct. The distinguishing probability can be bounded by a collision argument.

Game_7 is the same as Game_6, but we do not populate the encoding table in oracle calls to Exp. By construction, the last two games are indistinguishable.

Game_8 is the same as Game_7, but when we lazily sample replies to Exp queries, we do so consistently with σ rather than τ. Notice that the Exp oracle now only depends on σ and is completely decoupled from τ. The two games are close because the sets from where we sample have a large overlap.

Game_9 is the same as Game_8, but we undo lazy sampling of σ and instead sample it all at once. Also, we again use the bit-fixing lemma [4, Lemma 9] to undo resampling of τ. Since we are essentially reverting the second game hop from above, the distinguishing advantage can be bounded as before.

Game_{10} is the same as Game_9, but we remove the constraint that all queries $\mathcal{S}$ makes are pairwise different. Doing so, we have obtained the PGG game with $b = 0$, i.e., where the oracle Exp uses an independent encoding σ. Again, the distinguishing advantage here is the same as in the first game hop.

Collecting the terms above, we obtain

$$\mathrm{Adv}^{\mathrm{pgg}}_{p,\mathsf{S},\mathcal{S},\mathcal{D}} \leq 3T^2 \cdot \mathrm{Adv}^{\mathrm{alg}}_{p,\mathsf{S},\mathcal{S},\mathcal{P}} + \frac{26T^2}{p} + 36T\left(\frac{S}{P} + \frac{P}{p}\right),$$

where $\mathcal{P}$ is a predictor against algebraic unpredictability of $\mathcal{S}$ and $P \in \mathbb{N}$ an arbitrary number (coming from the application of the bit-fixing lemma). Setting $P :\approx \sqrt{Sp}$ to minimize the term on the right, we obtain the claimed result. □

Uber with preprocessing. Looking ahead, we observe that Uber and Uber-II sources are statistically algebraically unpredictable (even in the GGM) and, in particular, their algebraic unpredictability bound does not depend on the number of queries made by the predictor to the generic group oracles. This in turn implies that when the above theorem is applied to the Uber and Uber-II sources (with q polynomials of degree at most d and at most T generic group and operation queries) we obtain $\mathrm{Adv}^{\text{dua-ii}}_{p,\mathcal{A}} \leq \tilde{\mathcal{O}}\left(d(T+q)^2/p + \sqrt{S(T+q)^2/p}\right)$ in the GGM. When setting $q = 4$ and $d = 2$, the bound matches that established for the DDH problem [28, 29].

Discussion on DDH-II. As a second corollary, we obtain the hardness of the r-DDH-II assumption[14] in the GGM (here "r" stands for randomized generator). This result was also established by Bartusek, Ma, and Zhandry (BMZ) [4, Theorem 12]. Our proof, besides establishing the hardness of a winder class of assumption, is more modular and also avoids asymptotics. Furthermore, since our feasibility only relies on *computational* algebraic unpredictability, it can be applied in a setting where some group elements are directly leaked to the distinguisher.

5 From Simple to Algebraic Unpredictability: LDDs

We define a new type of hash function family called *linear-dependence destroyer* (LDD) that is useful for building schemes secure in PGGs. Intuitively, LDDs are hash functions with domain and range $\mathbb{Z}_p$ that remove, in a statistical sense, any linear dependence among a list of distinct but potentially correlated values.

Linear-Dependence Destroyers (LDDs). Let H be a hash function family with domain and range $\mathbb{Z}_p$ for some prime p. We define the advantage of a pair of adversaries $(\mathcal{S}, \mathcal{A})$ in the LDD game for H as

$$\mathrm{Adv}^{\mathrm{ldd}}_{\mathsf{H},\mathcal{S},\mathcal{A}}(\lambda) \coloneqq \Pr[\mathrm{LDD}^{\mathcal{S},\mathcal{A}}_{\mathsf{H}}(\lambda)],$$

where the LDD game is defined in Figure 5 (top left). We require that the outputs of $\mathcal{S}$ be pairwise distinct and the output of $\mathcal{A}$ be different from the all-zero tuple. We say that H is LDD[$\mathbf{S}$] secure if the advantage of any $(\mathcal{S}, \mathcal{A})$ in the LDD game is negligible, with $\mathcal{S} \in \mathbf{S}$ and $\mathcal{A}$ a PPT machine. We write this as $\mathsf{H} \in \mathrm{LDD}[\mathbf{S}]$.

[14] Distinguish (g, g^x, g^y, g^{xy}) from (g, g^x, g^y, g^z) for a random generator g, unpredictable x, and random y and z.

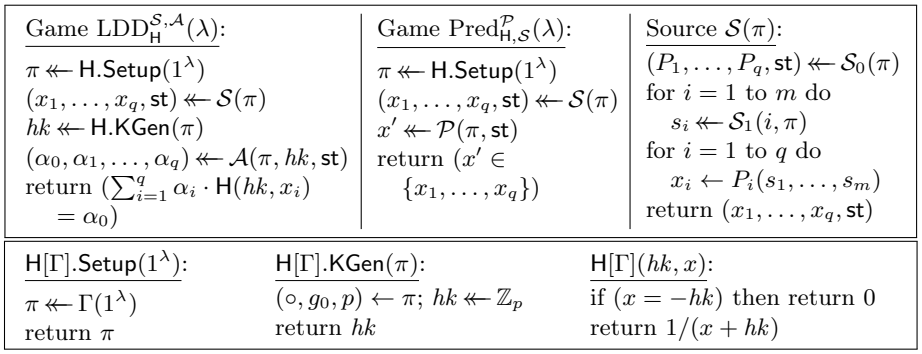

Fig. 5. Top left: The LDD game. **Top center**: The predictability game. **Top right**: Structure of a low-degree LDD source. **Bottom**: Candidate construction of an LDD family $\mathsf{H}[\Gamma]$ from a computational group scheme Γ.

We call an LDD source $\mathcal{S}$ statistically unpredictable if

$$\mathrm{Adv}^{\mathrm{pred}}_{\mathsf{H},\mathcal{S},\mathcal{P}}(\lambda) \coloneqq \Pr[\mathrm{Pred}^{\mathcal{P}}_{\mathsf{H},\mathcal{S}}(\lambda)]$$

is negligible for any (possibly unbounded) predictor $\mathcal{P}$, where the game Pred is defined in Figure 5 (top center). We denote the class of all statistically unpredictable LDD sources by $\mathbf{S}^{\mathrm{sup}}$. We say that H is an LDD if it is LDD$[\mathbf{S}^{\mathrm{sup}}]$ secure.

In the full version of this work we show that, for a computational group scheme Γ, the hash function family $\mathsf{H}[\Gamma]$ with domain and range $\mathbb{Z}_p$ defined in Figure 5 (bottom) is an LDD for the class of *low-degree sources* $\mathbf{S}^{\mathrm{low}}$. These are sources that compute their output as evaluations of low-degree polynomials on high-entropy points, as in Figure 5 (top right). We present an informal statement of our theorem in the following, and refer the reader to the full version for formal definitions and proofs.

Theorem 2 ((Informal) LDD for low-degree sources). *Let Γ be a computational group scheme, and let $\mathsf{H}[\Gamma]$ be the hash function family defined in Figure 5 (bottom). Then $\mathsf{H}[\Gamma] \in \mathrm{LDD}[\mathbf{S}^{\mathrm{low}}]$.*

We were unable to prove that this construction is an LDD for *all* unpredictable sources, though we have not been able to break it either. We conjecture that LDDs exist for *all* statistically unpredictable sources, and not just for low-degree ones. More strongly, we conjecture that the hash function $\mathsf{H}[\Gamma]$ defined in Figure 5 (bottom) is LDD secure for all statistically unpredictable sources. We emphasize that LDD is an *information-theoretic* notion and thus unconditional constructions (as for randomness extractors) may exist. We note that positive results for smaller classes of sources are also meaningful, as they would translate, via our constructions and proofs, into deterministic PKE, UCEs, and RKA-secure encryption.

As evidence towards the first conjecture, we can easily prove that a random oracle $\rho\colon \mathbb{Z}_p^2 \to \mathbb{Z}_p$ is an LDD, if all algorithms only get polynomially bounded

oracle access to ρ (rather than the entire function table). Assume that $\mathcal{A}$ makes at most n queries, and let E denote the event that $\mathcal{A}$ queries ρ on one of the points $(hk, x_1), \ldots, (hk, x_q)$. Then we can build a predictor $\mathcal{P}$ such that $\Pr[E] \leq n \cdot \Pr[\mathrm{Pred}^{\mathcal{P}}_{\rho,\mathcal{S}}(p)]$, which is small by unpredictability of $\mathcal{S}$. If E does not occur, then the equation $\sum_{i=1}^{q} \alpha_i \cdot \mathsf{H}(hk, x_i) = \alpha_0$ is satisfied with probability at most $1/p$, because at least one coefficient α_j, $1 \leq j \leq q$, is non-zero, and thus the random-looking value $\mathsf{H}(hk, x_j)$ is determined by the winning condition.

We provide a stronger feasibility result for LDDs by showing that random functions are LDDs for any unpredictable source, even when *both* the source $\mathcal{S}$ and the adversary $\mathcal{A}$ have *full access* to the table of the random function.[15] This result would thus establish the existence of LDDs, similarly to that for other information-theoretic objects such as randomness extractors. At a very high level, we prove this result in two steps: First, we decompose arbitrary high-entropy sources into a convex combination of flat sources (i.e., sources that are uniform on subsets of the support of the distribution). This is a standard technique in the study of randomness extractors [53]. Second, we apply a *compression-style* argument [37] to show that any predictor that has a high LDD-advantage against unpredictable sources can be used to compress the random function. The complete proof is rather technical, and we refer to the full version for a more detailed overview as well as the formal details.

6 Applications of PGGs

We now explore some examples of how PGGs can be used to prove the hardness of group-based assumptions and the security of practical cryptosystems under a variety of notions. As our first application, we prove that the decisional Uber assumption (DUA) family holds in PGGs. In doing so we also capture all of its implications. We then turn our attention to applications which do not seem to fall under the umbrella of the DUA. In the interest of space, we show here only how to construct UCEs from PGGs and LDDs. Further applications, namely KDM-CPA and RKA-CPA security of (modified versions of) ElGamal, and security of the ElGamal-with-Hash deterministic encryption scheme, are discussed in the full version. Interestingly, all these applications enjoy reductions under PGGs which furthermore retain to a large extent the simplicity of proofs in the GGM. Standard-model constructions of such schemes under Uber (for example, the KDM-secure PKE scheme of Boneh et al. [19]) are often substantially more complex and less efficient.

6.1 Uber Assumptions in PGGs

The Uber assumption family [18, 20] is an umbrella assumption that generalizes many hardness assumptions used to analyze the security of concrete cryptosystems. It has been formalized for both simple and bilinear groups, and has been

[15] Accordingly, we also impose a statistical notion of unpredictability on sources by giving predictors access to the full table.

Game DUA-II$_\Gamma^{\mathcal{A}}(\lambda)$:
$d \twoheadleftarrow \{0,1\}$; $(\pi \coloneqq (\circ, g_0, p)) \twoheadleftarrow \Gamma(1^\lambda)$; $r \twoheadleftarrow \mathbb{Z}_p^*$; $g \leftarrow g_0^r$
$(R_1, \ldots, R_n, T, \mathsf{st}) \twoheadleftarrow \mathcal{A}_0(\pi)$; $R_{n+1} \leftarrow T$
for $i = 1$ to m do $\mathbf{s}[i] \twoheadleftarrow \mathcal{A}_1(i, \pi)$
if $(\exists i \in [n+1])(\check{R}_i(\mathbf{s}) = 0)$ then return true
for $i = 1$ to n do $h_i \leftarrow g^{R_i(\mathbf{s})}$
if $(d = 0)$ then $r' \twoheadleftarrow \mathbb{Z}_p$ else $r' \leftarrow T(\mathbf{s})$
$h \leftarrow g^{r'}$; $d' \twoheadleftarrow \mathcal{A}_2(\pi, h_1, \ldots, h_n, h, \mathsf{st})$; return $(d = d')$

Fig. 6. The decisional Uber assumption II (DUA-II) game. Here, m is an upper bound on the number of variables of the R_i, $i \in [n+1]$. The (ordinary) decisional Uber assumption (DUA) is a special case of DUA-II where $\mathcal{A}_1(i, \pi)$ is the uniform distribution over $\mathbb{Z}_p$ for all $i \in [m]$ and all π.

shown to hold in (bilinear) generic groups [18]. In this work we focus on simple (i.e., non-bilinear) groups and show that Uber assumptions for them fall within the PGG framework. More precisely, we show that non-interactive, generator-independent Uber assumptions hold for PGGs.

We present an entropic generalization of the decisional version of the Uber assumption, which we call DUA-II, and show that it holds for PGGs. Loosely speaking, DUA-II extends DUA by sampling the inputs to the polynomials from independent, high-entropy distributions, rather than uniformly at random. Restricted versions of DUA-II and applications thereof have previously appeared in the literature. (See, for example, Canetti's DDH-II assumption [24].)

Decisional Uber Assumption II (DUA-II). Let Γ be a computational group scheme. We define the advantage of an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ in the DUA-II game for Γ as

$$\mathrm{Adv}^{\text{dua-ii}}_{\Gamma,\mathcal{A}}(\lambda) \coloneqq 2 \cdot \Pr[\text{DUA-II}^{\mathcal{A}}_\Gamma(\lambda)] - 1\,,$$

where the DUA-II game is defined in Figure 6. Here, $\mathcal{A}_0$ and $\mathcal{A}_1$ can be unbounded with polynomially bounded output, and $\mathcal{A}_2$ is PPT. We require that T be linearly independent from $R_1, \ldots, R_n$, and that $\mathbf{H}_\infty(\mathcal{A}_1(i, \pi)) = \omega(\log \lambda)$ for every $i \in \mathbb{N}$ and every $\pi \twoheadleftarrow \Gamma(1^\lambda)$. We say that Γ is DUA-II secure if, for any $\mathcal{A}$ as above, the advantage of $\mathcal{A}$ in the DUA-II game for Γ is negligible.

Notice that we can assume without loss of generality that the rational functions $R_1, \ldots, R_n$ returned by $\mathcal{A}_0$ are linearly independent. Indeed, linear (in)dependence can be checked by computing $D \coloneqq \mathrm{lcm}(\check{R}_1, \ldots, \check{R}_n)$, then writing a generic linear combination $\sum_{i=1}^n a_i \cdot R_i D = 0$, and then solving the ensuing linear system for $(a_1, \ldots, a_n)$. This yields a nonzero solution if and only if $R_1, \ldots, R_n$ are linearly dependent. Now observe that if R_k is linearly dependent on $R_1, \ldots, R_{k-1}$, then $g^{R_k(\mathbf{x})}$ can be computed directly by $\mathcal{A}_2$ (who knows $g^{R_1(\mathbf{x})}, \ldots, g^{R_{k-1}(\mathbf{x})}$) before guessing d', without having to rely on the challenger.

We now show that the Uber-II assumption holds in pseudo-generic groups. (We thus recover several cryptographic applications that fall under the reach of

Uber and Uber-II.) We only give an informal statement and a proof sketch here, and refer to the full version for the formal statement and a full proof.

Theorem 3 ((Informal) PGG $\Longrightarrow$ DUA-II). *Let* Γ *be a computational group scheme. If* Γ *is* PGG$[\mathbf{S}^{\mathrm{alg}} \cap \mathbf{S}^{\mathrm{duber}}]$ *secure, then it is* DUA-II *secure.*

Proof Overview. Given an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ in the DUA-II game, consider a PGG adversary $(\mathcal{S}, \mathcal{D})$ defined as follows: Source $\mathcal{S}$ runs $\mathcal{A}_0$ and $\mathcal{A}_1$, and queries Exp on $R_1(\mathbf{s}), \ldots, R_n(\mathbf{s}), r'$, where r' is either $T(\mathbf{s})$ or a random value, the choice being made at random. Distinguisher $\mathcal{D}$ then runs $\mathcal{A}_2$ and checks if it did predict the choice made by $\mathcal{S}$. By construction, $\mathcal{S}$ is a dUber source.

By direct inspection, the game $\mathrm{PGG}_{\Gamma}^{\mathcal{S},\mathcal{D}}(\lambda)$ with challenge bit $b = 1$ coincides with the game $\text{DUA-II}_{\Gamma}^{\mathcal{A}}(\lambda)$. On the other hand, when $b = 0$, the probability of $(\mathcal{S}, \mathcal{D})$ winning the PGG game is negligible. This follows from a bad event analysis: We transition to a game where $r' \neq R_i(\mathbf{s})$ for all $i \in [n]$. Given that σ is a random injection, we can then move to a game where the corresponding reply is picked at random, independently of the random choice of $\mathcal{S}$, so that $\mathcal{A}$ has no advantage in this game.

For algebraic unpredictability, let $\mathcal{P}$ be any predictor that returns a linear combination of the queries with coefficients $\alpha_1, \ldots, \alpha_n, \alpha_{n+1}$ given the leakage computed using the ideal Exp oracle. We again transition to a game where the Exp queries are pairwise distinct, and then replace the answers with pairwise different random elements that are independent of $\mathbf{s}$. Winning the algebraic unpredictability game then means that $\mathbf{s}$ (which $\mathcal{P}$ now knows nothing about) is a root of $\alpha_1 R_1 + \cdots + \alpha_n R_n + \alpha_{n+1} r'$, which is unlikely by Schwartz–Zippel. □

6.2 Building UCEs

In this section we show how to construct UCEs based on PGGs and LDDs. We consider UCEs for statistically unpredictable and *split* sources [10], whose definition we recall below; see Figure 7 (left). Split sources are required to make distinct queries to prevent iO-based attacks. BHK use split sources to prove security of a number of applications, including RKA security, point-function obfuscation, and storage-auditing protocols, as well as several other applications that rely on computationally unpredictable split sources.

As can be seen in Figure 7 (left), a split UCE source allows for limited post-processing of the outputs of the hash. This feature of split sources, however, is *not* used in any of its applications: The very simple $\mathcal{S}_1$ that merely returns $\mathbf{y}$ is sufficient for proving the security of the applications that BHK consider for split sources.[16] Our result in this section allows to recover applications of UCEs with respect to split sources (with a trivial $\mathcal{S}_1$) under PGGs and LDDs.[17]

[16] Interestingly, this simplification provides another avenue to circumvent iO-based attacks that exploit repetitions in $\mathbf{x}$.

[17] We note, however, that in iterative constructions of block-ciphers from hash functions [11], or indeed in domain extenders for hash functions [51], adaptive calls to the hash function seem to be necessary.

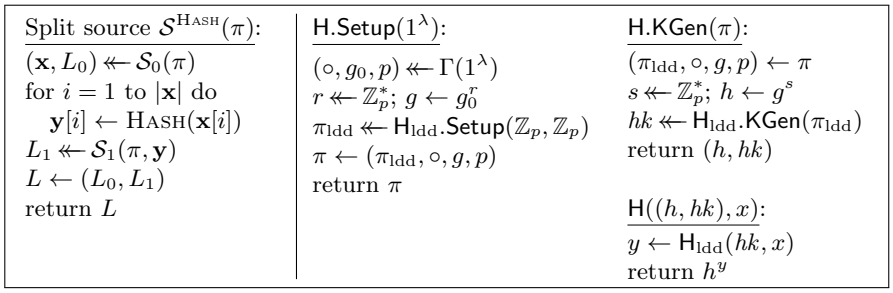

Fig. 7. Left: Structure of the split source $\mathcal{S} = \mathsf{Splt}[\mathcal{S}_0, \mathcal{S}_1]$ associated to $\mathcal{S}_0$ and $\mathcal{S}_1$. In simple split sources, $\mathcal{S}_1$ returns $L_1 = \mathbf{y}$. **Right**: A UCE built from a PGG and an LDD hash function.

Our construction of UCEs from PGGs is inspired by the correlated-input (CI) secure hash of Goyal, O'Neill, and Rao (GOR) [38], where outputs of a hash function are required to look random on high-entropy, but possibly correlated, inputs. GOR show that the hash function which maps $x \mapsto g^{1/(x+hk)}$, where $hk \twoheadleftarrow \mathbb{Z}_p$ is the hash key, is non-adaptively CI secure for polynomially induced correlations under the q-DDH assumption. This assumption falls within Uber and thus together with Theorem 3, we re-obtain this result. This, however, falls short of achieving split UCE security, since the hash inputs are polynomially induced.

We make progress towards building fully secure UCEs from group-based assumptions. We present our construction in a modular way in terms of an underlying LDD as shown in Figure 7 (right). (The GOR hash is that associated with the conjectural LDD $(hk, x) \mapsto 1/(x + hk)$.) Based on the conjectured existence of LDDs for all unpredictable sources, we obtain a fully secure UCE (beyond polynomial sources) for all statistically unpredictable and split sources. As for KDM security, in the GGM, we can account for preprocessing too.

Looking ahead into the proof, there is a close correspondence between the class of sources for which one achieves LDD security and UCE security. That is, if LDDs for a certain (e.g., low-degree) class of sources can be built, this would translate into UCE security for an analogous source class. Thus, we obtain an unconditional result for low-degree sources (since Theorem 2 shows that $1/(x + hk)$ is an LDD for low-degree sources).

Split sources. A UCE source $\mathcal{S}$ is called *split* if there exist PPT algorithms $\mathcal{S}_0$ and $\mathcal{S}_1$ such that $\mathcal{S}$ takes the form in Figure 7 (left). Here, $\mathcal{S}_0$ returns a vector $\mathbf{x}$ whose entries are required to be pairwise distinct, and some leakage L_0. We write $\mathcal{S} = \mathsf{Splt}[\mathcal{S}_0, \mathcal{S}_1]$ if $\mathcal{S}$ is a split source constructed from algorithms $\mathcal{S}_0$ and $\mathcal{S}_1$ as above, and we denote by $\mathbf{S}^{\text{splt}}$ the class of all such split sources. We further define the class $\mathbf{S}^{\text{ssplt}} \subseteq \mathbf{S}^{\text{splt}}$ of *simple split sources*, which are split sources where $\mathcal{S}_1$ simply returns $L_1 = \mathbf{y}$.

Theorem 4 (PGG $\wedge$ LDD $\Longrightarrow$ UCE[$\mathbf{S}^{\text{sup}} \cap \mathbf{S}^{\text{ssplt}}$]). *Let Γ be a computational group scheme, H_{ldd} a hash function family, and H the hash function family based*

Auxiliary dUber source $\bar{\mathcal{S}}'(\pi)$:	PGG distinguisher $\mathcal{D}'(\pi, (\mathbf{y}', L'))$:
$(\circ, g_0, p) \leftarrow \pi$; $r', s' \twoheadleftarrow \mathbb{Z}_p^*$; $g' \leftarrow g_0^{r'}$ $\pi_{\mathrm{ldd}} \twoheadleftarrow \mathsf{H}_{\mathrm{ldd}}.\mathsf{Setup}(\mathbb{Z}_p, \mathbb{Z}_p)$ $\pi' \leftarrow (\pi_{\mathrm{ldd}}, \circ, g', p)$ $hk \twoheadleftarrow \mathsf{H}_{\mathrm{ldd}}.\mathsf{KGen}(\pi_{\mathrm{ldd}})$ $(\mathbf{x}, L_0) \twoheadleftarrow \mathcal{S}_0(\pi')$; $n \leftarrow \lvert\mathbf{x}\rvert$; $L' \leftarrow (L_0, \pi', hk)$ for $i = 1$ to n do $\mathbf{x}'[i] \leftarrow s' \cdot \mathsf{H}_{\mathrm{ldd}}(hk, \mathbf{x}[i])$ $\mathbf{x}'[n+1] \leftarrow s'$ return $(\mathbf{x}', L')$	$n+1 \leftarrow \lvert\mathbf{y}'\rvert$ $h \leftarrow \mathbf{y}'[n+1]$ $(L_0, \pi', hk) \leftarrow L'$ $\mathbf{y} \leftarrow \mathbf{y}'[1..n]$; $L \leftarrow (L_0, \mathbf{y})$ $b' \twoheadleftarrow \mathcal{D}(\pi', (h, hk), L)$ return b'

Fig. 8. Reduction from a UCE adversary $(\mathcal{S}, \mathcal{D})$ to a PGG adversary $(\mathcal{S}', \mathcal{D}')$.

on Γ *and* $\mathsf{H}_{\mathrm{ldd}}$ *as defined in Figure 7 (right). If* Γ *is* $\mathrm{PGG}[\mathbf{S}^{\mathrm{alg}} \cap \mathbf{S}^{\mathrm{duber}}]$ *secure and* $\mathsf{H}_{\mathrm{ldd}}$ *is* $\mathrm{LDD}[\mathbf{S}^{\mathrm{sup}}]$ *secure, then* H *is* $\mathrm{UCE}[\mathbf{S}^{\mathrm{sup}} \cap \mathbf{S}^{\mathrm{ssplt}}]$ *secure. More precisely, for any adversary* $(\mathcal{S}, \mathcal{D})$ *in the UCE game for* H*, there are an adversary* $(\mathcal{S}', \mathcal{D}')$ *in the PGG game for* Γ*, and LDD source* $\mathcal{S}$ *an adversary* $\mathcal{A}_{\mathrm{cr}}$ *such that*

$$\mathrm{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathcal{S},\mathcal{D}}(\lambda) \leq 2 \cdot \mathrm{Adv}^{\mathrm{pgg}}_{\Gamma,\mathcal{S}',\mathcal{D}'}(\lambda) + q(\lambda)^2 \cdot \mathrm{Adv}^{\mathrm{ldd}}_{\mathsf{H}_{\mathrm{ldd}},\mathcal{S},\mathcal{A}_{\mathrm{cr}}}(\lambda) + \frac{q(\lambda)^2}{2^{\lambda-1}}\,.$$

Here, $q(\lambda)$ *is an upper bound on the number of queries made by* $\mathcal{S}$ *to its* Hash *oracle. Algorithm* $\mathcal{S}'$ *makes at most* $q(\lambda)+1$ *queries to its* Exp *oracle and runs in similar time, and algorithm* $\mathcal{D}'$ *runs in similar time to* $\mathcal{D}$*.*

Furthermore, $\mathcal{S}' \in \mathbf{S}^{\mathrm{alg}} \cap \mathbf{S}^{\mathrm{duber}}$ *is algebraically unpredictable. More precisely, for any PGG algebraic predictor* $\mathcal{P}'$ *there is an LDD adversary* $\mathcal{A}$ *for* $\mathsf{H}_{\mathrm{ldd}}$ *such that*

$$\mathrm{Adv}^{\text{alg-pred}}_{\Gamma,\mathcal{S}',\mathcal{P}'}(\lambda) \leq \mathrm{Adv}^{\mathrm{ldd}}_{\mathsf{H}_{\mathrm{ldd}},\mathcal{S},\mathcal{A}}(\lambda) + \frac{q(\lambda)^2}{2} \cdot \mathrm{Adv}^{\mathrm{ldd}}_{\mathsf{H}_{\mathrm{ldd}},\mathcal{S},\mathcal{A}_{\mathrm{cr}}}(\lambda)\,.$$

Moreover, $\mathcal{S} \in \mathbf{S}^{\mathrm{sup}}$*. That is for any predictor* $\mathcal{P}''$*, there is a predictor* $\mathcal{P}$ *against the original UCE source* $\mathcal{S}$ *in the* Pred *game such that*

$$\mathrm{Adv}^{\mathrm{pred}}_{\mathsf{H}_{\mathrm{ldd}},\mathcal{S},\mathcal{P}''}(\lambda) \leq \mathrm{Adv}^{\mathrm{pred}}_{\mathsf{H},\mathcal{S},\mathcal{P}}(\lambda)\,.$$

Proof Overview. Let $(\mathcal{S}, \mathcal{D})$ be PPT adversaries against UCE security of H. We build $(\mathcal{S}', \mathcal{D}')$ against the PGG security of the underlying group as shown in Figure 8.

Advantage bound. Let b denote the challenge bit in the PGG game. Then it is easy to see that

$$\Pr\left[\mathrm{PGG}^{\mathcal{S}',\mathcal{D}'}_{\Gamma}(\lambda) \,\middle|\, b = 1\right] = \Pr\left[\mathrm{UCE}^{\mathcal{S},\mathcal{D}}_{\mathsf{H}}(\lambda) \,\middle|\, b = 1\right]\,.$$

Indeed, when $b = 1$ the exponentiation oracle is implemented via the real group operations. Parameter π' contains a random generator $g' = g_0^{r'}$. Hash values are computed with respect to $h = g_0^{rs'} = g_0^{r' \cdot rs'/r'}$. This means that the exponent of the first element of the hash key is rs'/r', which results in a random group

element. Thus the UCE source and distinguisher are run as they would be in the UCE game with respect to the real hash function.

We next claim that

$$\Pr\left[\mathrm{PGG}_{\Gamma}^{\mathcal{S}',\mathcal{D}'}(\lambda)\,\middle|\,b=0\right] \le \Pr\left[\mathrm{UCE}_{\mathsf{H}}^{\mathcal{S},\mathcal{D}}(\lambda)\,\middle|\,b=0\right] + q(\lambda)^2 \cdot \mathrm{Adv}^{\mathrm{ldd}}_{\mathsf{H}_{\mathrm{ldd}},\mathcal{S},\mathcal{A}_{\mathrm{cr}}}(\lambda) + \frac{q(\lambda)^2}{2^\lambda}\,.$$

This claim follows from the fact that when $b = 0$ the Exp oracle returns random values subject to injectivity. We now transition to a game where Exp implements a random function. Using the random-function/random-permutation switching lemma, we incur an additive loss of $q(\lambda)^2/2^\lambda$. We modify this game further and replace the random function with a forgetful random function. The two games are identical unless there is a collision in the inputs to the random function. We may bound the probability of this event via the collision probability of the LDD, which itself can be bounded in terms of the LDD advantage: consider an adversary $\mathcal{A}_{\mathrm{cr}}$ that picks two indices, sets their coefficients to $+1$ and -1, the rest of the coefficients to 0, and $\alpha_0 = 0$. The LDD source here is identical to the UCE source. Thus any LDD predictor can be converted into a UCE predictor: simply ignore the hash values and run the LDD predictor. This justifies the final inequality in the theorem.

The final game that we arrive at is equivalent to the UCE game with respect to a random oracle (recall that the source outputs distinct inputs).

Algebraic unpredictability. We now show that the PGG source constructed above is algebraically unpredictable. Consider a modified algebraic prediction game whereby Exp returns random group elements, still subject to injectivity but not respecting equality across inputs. These two games are identical unless there is a collision among the inputs. We may bound the probability of collision via the LDD adversary $\mathcal{A}_{\mathrm{cr}}$ as above. This incurs a loss of $q(\lambda)^2/2$ times LDD advantage of $\mathcal{A}_{\mathrm{cr}}$.

We now rely on the LDD security of the hash function to bound the probability of winning the *modified* algebraic predictability. Suppose there exists an algebraic predictor $\mathcal{P}'$ against PGG source $\mathcal{S}'$. We construct an LDD source $\mathcal{S}''$ and an LDD adversary $\mathcal{A}$ as follows. Source $\mathcal{S}$ runs $\mathcal{S}'$, which is itself running $\mathcal{S}$ and hence is identical to $\mathcal{S}$. (This source is unpredictable as shown above.) Adversary $\mathcal{A}$ receives a hash key and leakage, and simulates the group elements that the algebraic predictor $\mathcal{P}'$ in the modified game expects randomly but subject to injectivity. Together with the collision bound above, this establishes the second inequality stated in the theorem. □

Remark. We note that the above proof can be easily extended to *multi-key* UCEs [10, Figure 8] for split sources by generating multiple hash keys and hash public keys via re-randomization. BHK conjectured that UCE and multi-key UCE are in general equivalent, which remains open.

An alternative construction of UCEs from PGGs would first compute g^{rx} and then chop half of the output bits so that group operations on hash outputs

are no longer possible. (This was previously suggested, for example, as a way to build a RO in the GGM by Zhandry and Zhang [56].) An analysis of this construction may be made possible in the PGG framework by defining new sources that permit different forms of post-processing.

Acknowledgments

We thank Sogol Mazaheri for collaborating in the early stages of this work. We also thank anonymous reviewers who helped improve the presentation of our results. Pooya Farshim was supported in part by EPSRC grant EP/V034065/1. Patrick Harasser was funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297. Adam O'Neill is supported in part by a gift from Cisco.

References

1. T. Agrikola and D. Hofheinz. Interactively secure groups from obfuscation. In *PKC 2018, Part II*.
2. T. Agrikola, D. Hofheinz, and J. Kastner. On instantiating the algebraic group model from falsifiable assumptions. In *EUROCRYPT 2020, Part II*.
3. B. Applebaum. Key-dependent message security: Generic amplification and completeness. In *EUROCRYPT 2011*.
4. J. Bartusek, F. Ma, and M. Zhandry. The distinction between fixed and random generators in group-based assumptions. In *CRYPTO 2019, Part II*.
5. B. Bauer, G. Fuchsbauer, and J. Loss. A classification of computational assumptions in the algebraic group model. In *CRYPTO 2020, Part II*.
6. M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO 2007*.
7. M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT 2009*.
8. M. Bellare, D. Cash, and R. Miller. Cryptography secure against related-key attacks and tampering. In *ASIACRYPT 2011*.
9. M. Bellare and V. T. Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In *EUROCRYPT 2015, Part II*.
10. M. Bellare, V. T. Hoang, and S. Keelveedhi. Instantiating random oracles via UCEs. In *CRYPTO 2013, Part II*.
11. M. Bellare, V. T. Hoang, and S. Keelveedhi. Cryptography from compression functions: The UCE bridge to the ROM. In *CRYPTO 2014, Part I*.
12. M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO 2004*.
13. E. Birrell, K.-M. Chung, R. Pass, and S. Telang. Randomness-dependent message security. In *TCC 2013*.
14. N. Bitansky and R. Canetti. On strong simulation and composable point obfuscation. In *CRYPTO 2010*.
15. J. Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In *FSE 2006*.

16. J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *SAC 2002*.
17. F. Böhl, G. T. Davies, and D. Hofheinz. Encryption schemes secure under related-key and key-dependent message attacks. In *PKC 2014*.
18. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT 2005*.
19. D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *CRYPTO 2008*.
20. X. Boyen. The uber-assumption family (invited talk). In *PAIRING 2008*.
21. C. Brzuska, P. Farshim, and A. Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In *CRYPTO 2014, Part I*.
22. C. Brzuska and A. Mittelbach. Using indistinguishability obfuscation via UCEs. In *ASIACRYPT 2014, Part II*.
23. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*.
24. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO'97*.
25. R. Canetti and R. R. Dakdouk. Obfuscating point functions with multibit output. In *EUROCRYPT 2008*.
26. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*.
27. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4), 2004.
28. S. Coretti, Y. Dodis, and S. Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In *CRYPTO 2018, Part I*.
29. H. Corrigan-Gibbs and D. Kogan. The discrete-logarithm problem with preprocessing. In *EUROCRYPT 2018, Part II*.
30. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO'98*.
31. I. Damgård, C. Hazay, and A. Zottarel. Short paper on the generic hardness of DDH-II, 2014.
32. R. A. Demillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4), 1978.
33. A. W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *ASIACRYPT 2002*.
34. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In *CRYPTO 2013, Part II*.
35. P. Fenteany and B. Fuller. Same point composable and nonmalleable obfuscated point functions. In *ACNS 20, Part II*.
36. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *CRYPTO 2018, Part II*.
37. R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st FOCS*.
38. V. Goyal, A. O'Neill, and V. Rao. Correlated-input secure hash functions. In *TCC 2011*.
39. M. D. Green, J. Katz, A. J. Malozemoff, and H.-S. Zhou. A unified approach to idealized model separations via indistinguishability obfuscation. In *SCN 16*.
40. B. Hemenway and R. Ostrovsky. Building lossy trapdoor functions from lossy encryption. In *ASIACRYPT 2013, Part II*.

41. J. Kastner and J. Pan. Towards instantiating the algebraic group model. Cryptology ePrint Archive, Report 2019/1018, 2019.
42. J. Katz, C. Zhang, and H.-S. Zhou. An analysis of the algebraic group model. Cryptology ePrint Archive, Report 2022/210, 2022.
43. I. Komargodski and E. Yogev. Another step towards realizing random oracles: Non-malleable point obfuscation. In *EUROCRYPT 2018, Part I*.
44. U. M. Maurer. Abstract models of computation in cryptography (invited paper). In *10th IMA International Conference on Cryptography and Coding*.
45. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*.
46. V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2), 1994.
47. K. G. Paterson, J. C. N. Schuldt, and D. L. Sibborn. Related randomness attacks for public key encryption. In *PKC 2014*.
48. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. Assoc. Comput. Mach.*, 27(4), 1980.
49. V. Shoup. On fast and provably secure message authentication based on universal hashing. In *CRYPTO'96*.
50. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT'97*.
51. P. Soni and S. Tessaro. Public-seed pseudorandom permutations. In *EUROCRYPT 2017, Part II*.
52. P. Soni and S. Tessaro. Naor-Reingold goes public: The complexity of known-key security. In *EUROCRYPT 2018, Part III*.
53. S. P. Vadhan. *Pseudorandomness*. Now Publishers, 2012.
54. M. Zhandry. The magic of ELFs. In *CRYPTO 2016, Part I*.
55. M. Zhandry. To label, or not to label (in generic groups). In *CRYPTO 2022*.
56. M. Zhandry and C. Zhang. The relationship between idealized models under computationally bounded adversaries. Cryptology ePrint Archive, Report 2021/240, 2021.
57. R. Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and algebraic computation EUROSAM*. Springer, Berlin-New York, 1979.