# To Label, or Not To Label (in Generic Groups)

Mark Zhandry[1,2]
mark.zhandry@ntt-research.com

[1] NTT Research, Sunnyvale, USA
[2] Princeton University, Princeton, USA

**Abstract.** Generic groups are an important tool for analyzing the feasibility and in-feasibility of group-based cryptosystems. There are two distinct wide-spread versions of generic groups, Shoup's and Maurer's, the main difference being whether or not group elements are given explicit labels. The two models are often treated as equivalent. In this work, however, we demonstrate that the models are in fact quite different, and care is needed when stating generic group results:

- We show that numerous textbook constructions are *not* captured by Maurer, but are captured by Shoup. In the other direction, any construction captured by Maurer *is* captured by Shoup.
- For constructions that exist in both models, we show that security is equivalent for "single stage" games, but Shoup security is strictly stronger than Maurer security for some "multi-stage" games.
- The existing generic group un-instantiability results do not apply to Maurer. We fill this gap with a new un-instantiability result.
- We explain how the known black box separations between generic groups and identity-based encryption do not fully apply to Shoup, and resolve this by providing such a separation.
- We give a new un-instantiability result for the *algebraic* group model.

## 1 Introduction

Generic groups [Nec94, Sho97, Mau05] are idealized cryptographic groups where group operations are carried out by making queries to a group oracle, each query incurring unit cost. For both adversaries and constructions, generic groups capture natural *generic* algorithms which do not make any use of the particular features of the group in question, and instead only perform legal group operations.

There are plenty of valid criticisms of generic groups (e.g. [Fis00, KM06]), and like random oracles, generic groups cannot exist in the real world [Den02]. Nevertheless, cryptographic groups are one of the core cryptographic building blocks, and generic groups are critical to our understanding of the feasibility and infeasibility of group-based cryptosystems. The best *practical* attacks on many cryptosystems built from appropriate cryptographic groups are often generic. What's more, many of the most efficient schemes have only generic group proofs. As just one example, adaptive security is usually straightforward with generic groups. In contrast, standard model proofs of adaptive security often require more complex and less-efficient cryptosystems, such as the dual system methodology [Wat09]. Additionally, even cryptosystems with a standard-model security proof rely on computational assumptions, and for groups, there are many. When new such assumptions are made, they are often accompanied by a generic group proof of hardness, which at least demonstrates the lack of obvious flaws.

Moreover, generic groups are critical for black box separations, showing barriers to achieving various cryptographic objects from groups. Such barriers are important for the design of cryptosystems, even if one objects to using generic groups in security proofs. Examples of objects separated from generic groups include identity-based encryption [PRV12, SGS21], order revealing encryption [ZZ18], types of delay functions [RSS20], and accumulators [SGS20]. An impossibility relative to generic groups helps guide protocol design by showing what kinds of techniques will be required. While non-black box techniques can sometimes overcome such impossibilities — famously, IBE from the Diffie-Hellman assumption [DG17], for example — the use of such techniques almost always makes the results impractical. Thus, generic group impossibilities most likely rule out any practical protocol based on cryptographic groups.

*Two different generic groups.* Since first proposed by Nechaev [Nec94], two different flavors of generic groups have emerged. The first is Shoup's [Sho97], where the group is modeled as a random embedding of the additive group $\mathbb{Z}_p$ into bit strings, with the group operations carried out by making oracle queries.

Later, Maurer [Mau05] proposed a different model that uses pointers instead of a random representation. The oracle initializes a table with various values in $\mathbb{Z}_p$, representing exponents in the group. The adversary cannot access the values directly, but just knows their line numbers in the table. The adversary then outsources linear computations on the table to the oracle.

Shoup and Maurer are the main approaches used in the literature. Numerous works (e.g. [BFF+14, BCFG17, AY20, CL20, CH20]) actually treat the models as identical, simply referring to both as *the* generic group model. Some of these works justify this lack of distinction by pointing to a result of Jager and Schwenk [JS08], which provides some sort of equivalence between Shoup and Maurer. But Maurer, Portmann, and Zhu [MPZ20] find that Maurer's model allows for stronger hardness proofs, seemingly contradicting [JS08]. Recent works of Schul-Ganz and Segev [SGS20, SGS21] briefly argue that the equivalence only applies to problems that are defined "independent of the representation" of the group, and more generally the models are *incomparable.* The relationship between the models is not further explored, and it is not clarified what "independent of the representation" means. But the purpose of the generic group models is precisely to capture algorithms that work in any group, regardless of representation!

The above state of affairs makes it hard to interpret and compare the various positive and negative results using generic groups.

## 1.1 Overview of Results

In this work, we address the important questions above by giving a detailed comparison between the models.

*The Type-Safe Model.* First, we point that many works, primarily those proving black box separations, claiming to use Maurer's model actually do not operate in the model Maurer originally defined. Whereas Maurer's original model has algorithms outsourcing their group computations to a stateful table, many works instead have the algorithms perform the operations locally, but constrain the algorithms to performing only legal group operations through a simple type system. We argue that this is technically a different model than Maurer's original model. However, we observe that Maurer's model is actually poorly suited for the setting of general cryptosystems[1], and that the type-system based model implicitly used in many works is in fact the more appropriate model. We therefore formalize this model, which we call the *Type-Safe* (TS) generic group model.

Likewise, throughout this work, we will refer to Shoup's model as the *Random Representation* (RR) model, to give it a more descriptive name.

*From TS/Maurer to RR/Shoup for cryptosystems (Section 3).* For now we ignore security, and just discuss whether group-based cryptosystems exist in one model or the other. We generalize one direction of the proof of Jager and Schwenk [JS08] from algorithms to general cryptosystems, showing that:

**Theorem 1.1 (Informal).** *Any cryptosystem in the TS/Maurer model also exists in the RR/Shoup model.*

*Separations for cryptosystems (Section 4).* We then observe that the converse does not hold. Concrete cryptosystems that do not work in the TS model include the Blum-Micali PRG [BM82] and the Goldreich-Goldwasser-Micali PRF [GGM84]. Worse, we show that several primitives are simply impossible:

**Theorem 1.2 (Informal).** *Pseudorandom permutations, domain extension for collision resistant hashing, and encryption with additive ciphertext size overhead are each impossible in the TS/Maurer model.*

The applications excluded by Theorem 1.2 follow from textbook techniques that are taught in many introductory cryptography courses. These techniques work in the standard, RR/Shoup, and even random oracle models, the last often being treated as weaker than generic groups.

*Remark 1.3.* The recent works of [RSS20] and [DHH$^+$21] already, perhaps unintentionally, provide separations. [RSS20] proves delay functions impossible in the TS/Maurer model and [DHH$^+$21] proves signatures impossible. However, the RR/Shoup model implies random oracles [ZZ21], and both delay functions and signatures can readily be constructed assuming random oracles. Signatures can even be constructed from cryptographic groups in the *standard* model [Rom90]. We note that the purpose of these works was not to demonstrate limitations of the TS/Maurer model relative to RR/Shoup: [RSS20] argues for the difficulty of

[1] It appears it was never meant to be: Maurer discusses several classes of problems to consider in his model, capturing discrete log, DDH, and more exotic variants. But general cryptosystems are not covered by the classes of problems.

constructing group-based VDFs and time-lock puzzles, and [DHH+21] seeks to explain challenges in efficient group-based signatures. Nevertheless, a separation was a perhaps unintended consequence of their results. Our results show that the case of delay functions and signatures were not isolated incidents, and the inability of cryptosystems to work in the TS/Maurer model is in fact wide-spread.

*Remark 1.4.* The above results (including those of [RSS20] and [DHH+21]) show that the TS/Maurer model is actually *incomparable* to the random oracle model (ROM): public key encryption exists in the TS/Maurer model but not in the ROM [IR89], while the above results give examples that exist in the ROM but not TS/Maurer. This is in contrast to the RR/Shoup model, which is known to be *strictly stronger* than random oracles [ZZ21].

*Security in RR/Shoup vs TS/Maurer (Section 5).* We next turn to discussing whether schemes are *secure* in the models. We give two theorems, which are adaptations of the two directions of the proof of [JS08]:

**Theorem 1.5 (Informal).** *Amongst cryptosystems in TS/Maurer (and hence also RR/Shoup), security in RR/Shoup implies security in TS/Maurer.*

**Theorem 1.6 (Informal).** *Amongst cryptosystems in TS/Maurer, if the security experiment is* **single-stage***, then security in TS/Maurer implies security in RR/Shoup.*

Single-stage means there is a single adversary party; such games capture most of the basic security properties, from one-way functions to public key encryption and more. This is in contrast to *multi-stage* games, which communicate with multiple adversaries that have restricted communication. Multi-stage games include deterministic public key encryption (the message distribution is an additional adversary) and leakage resilience (the leakage function is an adversary).

We complement the above theorems by showing that, in the multi-stage setting, TS/Maurer security does *not* imply RR/Shoup security. Concretely:

**Theorem 1.7 (Informal).** *There exists a deterministic PKE in TS/Maurer (and therefore also in RR/Shoup), which is secure in TS/Maurer, but is insecure in RR/Shoup and insecure in any standard-model instantiation.*

Thus TS/Maurer and RR/Shoup are equivalent for single-stage games (provided the game works in TS/Maurer), but TS/Maurer is strictly less sound than RR/Shoup in the multi-stage setting.

*On the Un-instantiability of Generic Groups (Section 6).* Next, we consider the well-known criticism of generic groups that there are (contrived) schemes secure in the generic group model that cannot be securely instantiated under any group. This was proved by Dent [Den02] by adapting similar results for random oracles due to Canetti, Goldreich, and Halevi [CGH98]. There are now numerous ways to achieve the same result [Nie02, GK03, BBP04, BFM15]. These results typically work by having a branch in the honest algorithms that is completely insecure, say by outputting the secret key in the clear. These branches cannot be triggered in the ideal model, but can be triggered under any instantiation of the group.

However, we observe that the vast majority of these results only apply in RR/Shoup: basically the trigger is detected using the bit representation of group elements. In fact, we show that the paradigm of correlation intractability underlying many of these results cannot be used in TS/Maurer. Existing uninstantiability results that apply in TS/Maurer [BCPR14] correspond to a multi-stage game (extractable OWFs), and require indistinguishability obfuscation, a strong tool not known to be implied by any assumption on groups. Our deterministic PKE scheme is unconditional, though still multi-stage. The prior work therefore leaves open the tantalizing possibility of a standard model group which securely instantiates any single-stage TS/Maurer game. We refute this:

**Theorem 1.8.** *There exists a plain public key encryption (PKE) scheme and one-time message authentication code (MAC)*[2] *in the TS/Maurer model (and therefore also RR/Shoup) which are unconditionally secure in both models, but insecure under any instantiation of the group.*

Our schemes readily adapt to give private-key encryption and MACs in the random oracle model that are insecure under any instantiation, replicating the result of [CGH98]. Our constructions, however, use different ideas, far simpler tools, and are entirely self-contained. Our PKE scheme may also be qualitatively less contrived: as pointed out by [KM06], the prior approach of inserting a branch that causes insecure behavior goes against reasonable cryptographic practice. Our scheme, by contrast, is just ElGamal applied bit-by-bit, together with a single bit of leakage. Our leakage function itself is contrived, but leakage is usually modeled adversarially. An adversary could very well choose a contrived leakage.

*The Impossibility of IBE from Generic Groups (Section 7).* The literature contains two impossibilities for identity-based encryption (IBE) from generic groups: Schul-Ganz and Segev [SGS21] prove a separation in the TS/Maurer model, whereas the original separation due to Papakonstantinou, Rackoff, and Vahlis [PRV12] claims to prove a separation RR/Shoup. However, we observe that the definition of generic groups used in the latter work actually is somewhere between the TS/Maurer and RR/Shoup models. In particular, they make a TS/Maurer-style restriction where algorithms get explicit group elements as input, and are only allowed to make queries on those elements or elements derived from them. This restriction is used at a critical step in their proof, where they show how to eliminate the explicit group elements from a user's secret key.

Restricting algorithms to operating only on explicitly provided group elements makes sense for individual algorithms trying to solve non-interactive problems such as discrete log. But for cryptosystems comprising multiple communicating parts, group elements can easily be transmitted implicitly. One could simply flip all the bits of a group element, and then recover the element by flipping them back. Alternatively, one can secret share a group element into different shares.

[2] Plain PKE and MAC security are single-stage games.

We note the close relationship between IBE and signatures, with IBE immediately giving signatures by re-interpreting user secret keys as signatures. What's more, a crucial part of [PRV12] can be seen as running the verification algorithm of the derived signature scheme. Given the close relationship, and the fact that signatures are possible in RR/Shoup, but impossible after making TS/Maurer restrictions, it is important to understand whether [PRV12] can be overcome in the full RR/Shoup model. We fill in this gap, showing that this is not possible:

**Theorem 1.9.** *IBE does not exist in the RR/Shoup model.*

*The Soundness of the Algebraic Group Model (Section 8).* Fuchsbauer, Kiltz, and Loss [FKL18] propose the Algebraic Group Model (AGM) as a model that lies between the standard model and generic groups. Here, adversaries can see the actual standard-model group elements, but must be able to "explain" any group element it outputs as a linear combination of its input elements.

We first point out some definitional ambiguities in the literature needed to avoid trivially invalidating the model. We argue that the model envisioned by [FKL18] allows exactly the security games which exist in the TS/Maurer model. This means the model inherits the limitations of the TS/Maurer model. The AGM is therefore actually *incomparable* to the RR/Shoup model, and it cannot reason about many textbook techniques[3]. We also resolve an open question raised by [FKL18], showing an un-instantiability result for the AGM:

**Theorem 1.10.** *There exists a one-time message authentication code that is secure in the AGM but insecure in any standard-model instantiation of the group.*

We also take a closer look at the comparison between the AGM and the TS/Mauer model. We do not give any formal results, but argue that the claimed advantages of the AGM are not always supported by existing evidence.

### 1.2 Takeaways

Our work shows that extreme care must be taken when proving security or separations for generic groups. Our equivalence for single-stage *security* justifies the common practice of treating the models as equivalent for positive results in many settings, though the distinction is critical in the multi-stage setting. On the other hand, our separations for *constructions* show that impossibilities in the RR/Shoup vs TS/Maurer models must be interpreted very differently.

We also believe that our work points to significant limitations of black box impossibilities in the TS/Maurer model, as the model excludes numerous textbook (and black-box!) cryptographic techniques, ones that even work for the seemingly weaker random oracle model. On the other hand, these techniques seem to be all captured by RR/Shoup. This shows that impossibilities in RR/Shoup very closely reflect the available black box techniques for groups, whereas TS/Maurer

[3] Note that many works in the AGM starting from [FKL18] sometimes additionally add a random oracle, and these techniques *can* be used on the random oracle.

does not. Nevertheless, TS/Maurer impossibilities may still be useful for guiding cryptosystem design, by showing that non-algebraic (but potentially still generic) techniques making use of the group labels would be necessary.

Our work fills in some important gaps in the literature, showing (1) that IBE is impossible in the fully general RR/Shoup model, and (2) that TS/Maurer is impossible to instantiate in general in the standard model. For (2), we give a new, relatively simple, approach to achieving un-instantiability results. We hope that our result sheds additional light on the plausibility of generic groups.

Finally, we shed some additional light on the algebraic group model, by showing that it is incomparable to the RR/Shoup model and is nevertheless un-instantiable, despite being closer to the standard model.

### 1.3 Organization

Due to limited space and having several different results, we omit a separate detailed technical overview of our results, instead having a brief overview at the beginning of each of our technical sections. Section 2 defines our basic notation. Section 3 defines the TS/Maurer and RR/Shoup models, plus shows when the two can be treated equivalently. Section 4 demonstrates applications which exist in RR/Shoup but not TS/Maurer. Section 5 shows the inequivalence of security for multi-stage games. Section 6 gives our new un-instantiability result for TS/Maurer. Section 7 gives our new impossibility for IBE in the RR/Shoup model. Finally, Section 8 discusses the algebraic group model.

## 2 Preliminaries and Notation

We will use a non-uniform circuit model of computation, though all of our models and results can be translated into the Turing machine setting.

Throughout, let $\lambda > 0$ be a security parameter. An algorithm is therefore a list of circuits $\mathcal{C} = \{C_\lambda\}_{\lambda\in\mathbb{Z}}$, with domains $\mathcal{D}_\lambda$ and range $\mathcal{E}_\lambda$. The circuits comprising an algorithm can either be deterministic or probabilistic. In the later case, there are random coin gates, which generate a random bit.

For interactive algorithms, each circuit $C_\lambda$ is replaced by a sequence of circuits $C_\lambda^{(1)}, C_\lambda^{(2)}, \dots$. The domain of $C_\lambda^{(i)}$ is denoted $\mathcal{S}_\lambda^{(i)} \times \mathcal{I}_\lambda^{(i-1)}$ and the range is $\mathcal{S}_\lambda^{(i+1)} \times \mathcal{O}_\lambda^{(i)}$. Here, $\mathcal{S}_\lambda^{(i)}$ is the space of states that $C_\lambda^{(i)}$ passes to $C_\lambda^{(i+1)}$, $\mathcal{O}_\lambda^{(i)}$ is the space of outgoing messages that the algorithm sends in the $i$th step, and $\mathcal{I}_\lambda^{(i)}$ is the space of incoming messages. For convenience, we will generally suppress the security parameter.

We next consider *complete sets* of interacting algorithms. Each algorithm in the set is an interactive algorithm, sharing the same security parameter. There is a one-to-one correspondence between outgoing and incoming messages. Therefore, a complete set of interacting algorithms taken together yields a single non-interactive algorithm, which maps the initial inputs of each algorithm to the set of algorithms to the set of outputs. We can also consider a subset of a complete set of interacting algorithms, which we call an *incomplete* set. In this case, some of the messages are *internal*, sent amongst algorithms in the set, while other messages are *external*, and sent to and received from outside the set. An incomplete set of interacting corresponds to a single interactive algorithm, whose incoming and outgoing messages are the external messages.

### 2.1 Games and Cryptosystems

A game is given by a probabilistic interactive algorithm $\mathsf{Ch}$, called a challenger, and a function $t : \mathbb{Z}^+ \to [0, 1]$. The challenger is given as input a security parameter $\lambda \in \mathbb{Z}^+$, and interacts with $k$ non-communicating parties $\mathsf{A}_1, \dots, \mathsf{A}_k$. In other words, $(\mathsf{Ch}, \mathsf{A}_1, \dots, \mathsf{A}_k)$ forms a complete set of interacting algorithms, and $\mathsf{A}_1, \dots, \mathsf{A}_k$ forms an incomplete set where all messages are external. Collectively, $\mathsf{A} = (\mathsf{A}_1, \dots, \mathsf{A}_k)$ is called the *adversary.* After the interaction, $\mathsf{Ch}$ outputs a bit $b$; this interaction is denoted $b \leftarrow (\mathsf{A} \rightleftarrows \mathsf{Ch})(\lambda)$. If $b = 1$ we say the adversary wins, and if $b = 0$ we say the adversary looses. In the case $k = 1$, we call $(\mathsf{Ch}, t)$ a *single-stage* game. If $k > 1$, we call $(\mathsf{Ch}, t)$ a *multi-stage* game.

Let $\mathcal{A}$ be a class of adversaries. A game $(\mathsf{Ch}, t)$ is *hard* for $\mathcal{A}$ if, for all $\mathsf{A} \in \mathcal{A}$, there exists a negligible $\epsilon$ such that $\Pr[1 \leftarrow (\mathsf{A} \rightleftarrows \mathsf{Ch})(\lambda)] \leq t(\lambda) + \epsilon(\lambda)$. Typical examples of adversary classes are (1) all algorithms, (2) all polynomial-time (in $\lambda$) algorithms, or (3) all query algorithms making a polynomial number (in $\lambda$) of queries to some oracle. (1) is often referred to as statistical or information-theoretic security, whereas (2) is typically called computational security. For (3), if the algorithms are not restricted, we will call the adversary *query bounded.*

*Cryptosystems.* Abstractly, a cryptosystem is just a set of algorithms. Typically these algorithms will be non-interactive, though their incoming and outgoing messages may contain multiple components that would be sent to different users.

The security of a cryptosystem is usually defined by a game. In the case where the security experiment makes black-box use of the cryptosystem, the game itself is an incomplete set of interacting algorithms: one of these interacting algorithms is a *coordinator*, and the remaining algorithms are all instances of the cryptosystem components. The various instances of the cryptosystem components receive and send messages from the coordinator, who also sends and receives external messages to the one or more adversaries. The game together with the adversaries then forms a complete set of interacting algorithms.

For example, consider the case of one-way functions: the coordinator chooses a random $x$, and sends $x$ to one instance of the function $F$ to get $y$. Then the coordinator sends $y$ externally to the adversary, and receives $x'$ in response. It then sends $x'$ to a second instance of $F$ to get $y'$, and then checks $y = y'$.

### 2.2 Groups

We will write groups multiplicatively, writing $g \times h$ or simply $gh$ to denote group multiplication. We will always assume cyclic groups of prime order $p$. Given a group element $g \in \mathbb{G}$, a matrix of group elements $\mathbf{G} \in \mathbb{G}^{n \times m}$ and matrices

$\mathbf{A} \in \mathbb{Z}_p^{m\times r}, \mathbf{B} \in \mathbb{Z}_p^{s\times n}$, let $g^{\mathbf{A}} \in \mathbb{G}^{m\times r}$, $\mathbf{G}^{\mathbf{A}} \in \mathbb{G}^{n\times r}$ and ${}^{\mathbf{B}}\mathbf{G} \in \mathbb{G}^{s\times m}$ be the matrices defined as:

$$(g^{\mathbf{A}})_{i,j} = g^{\mathbf{A}_{i,j}} \qquad \left(\mathbf{G}^{\mathbf{A}}\right)_{i,j} = \prod_{k=1}^{m} \mathbf{G}_{i,k}^{\mathbf{A}_{k,j}} \qquad \left({}^{\mathbf{B}}\mathbf{G}\right)_{i,j} = \prod_{k=1}^{n} \mathbf{G}_{k,j}^{\mathbf{B}_{i,k}} \ .$$

Observe that for any appropriately-sized matrices $\mathbf{A}, \mathbf{B}$, $g^{\mathbf{A}\cdot\mathbf{B}} = (g^{\mathbf{A}})^{\mathbf{B}} = {}^{\mathbf{B}}(g^{\mathbf{A}})$.

## 3 Different Generic Group Models

Here, we recall Shoup's [Sho97] (which we will also call the *random representation* model) and Maurer's[Mau05] generic group models, as well as propose a *Type Safe* model, formalizing a model implicit in prior work.

### 3.1 Random Representation (RR)/Shoup Model [Sho97]

Let $p \in \mathbb{Z}$ be a positive integer, and let $S \subseteq \{0,1\}^*$ be a set of strings of cardinality at least $p$. We will assume an upper bound is known on the length of strings in $S$. A *random* injection $L : \mathbb{Z}_p \to S$ is chosen, which we will call the *labeling function*. We will think of $L(x)$ as corresponding to $g^x$, where $g$ is a fixed generator of the group. All parties—including the adversary, the cryptosystem, and the challenger—are able to make the following queries:

- **Labeling queries.** The party submits $x \in \mathbb{Z}_p$, and receives $L(x)$.
- **Group operations.** The party submits $(\ell_1, \ell_2, a_1, a_2) \in S^2 \times \mathbb{Z}_p^2$. If there exists $x_1, x_2 \in \mathbb{Z}_p$ such that $L(x_1) = \ell_1$ and $L(x_2) = \ell_2$, then the party receives $L(a_1x_1 + a_2x_2)$. Otherwise, the party receives $\perp$.

All queries incur unit cost. We denote the oracles together as $\mathbb{G}_{\mathrm{RR}}$. For an algorithm $A$ that makes queries to $\mathbb{G}_{\mathrm{RR}}$, we write $A^{\mathbb{G}_{\mathrm{RR}}}$. A game $(\mathsf{Ch}, t)$ in the RR model allows all parties (the challenger and one or more adversaries) to make queries to the generic group. We say that $(\mathsf{Ch}, t)$ is hard in the RR model if it is hard for the class of adversaries whose cost is polynomial in $\log p$.

We will think of $L(x)$ as corresponding to $g^x$, for some fixed generator $g$ of the group. Therefore, if $\ell_1, \ell_2$ correspond to $g^{x_1}, g^{x_2}$, the group operation query computes $g^{a_1x_1+a_2x_2} = (g^{x_1})^{a_1} \times (g^{x_2})^{a_2}$.

### 3.2 Maurer's Model [Mau05]

Again let $p \in \mathbb{Z}$ be a positive integer. An empty table $T$ is initialized. Then all parties are able to make the following queries:

- $\mathsf{Labeling\ queries}$. The party submits $(x, i) \in \mathbb{Z}_p \times \mathbb{Z}$. Row $i$ of $T$ is then set to $x$, potentially overwriting any contents at row $i$. No response is given.
- $\mathsf{Group\ operations}$. The party submits $(i_1, i_2, i_3, a_1, a_2) \in \mathbb{Z}^2 \times \mathbb{Z}_p^2$. If there are entries $x_1, x_2$ in rows $i_1, i_2$, respectively, of $T$, then row $i_3$ is set to $a_1x_1+a_2x_2$, potentially overwriting any contents at row $i_3$. If the contents of $i_1$ or $i_2$ are empty, then nothing is written. No response is given.

– Equality queries. The party submits $(i_1, i_2) \in \mathbb{Z}^2$. If there are entries $x_1, x_2$ in rows $i_1, i_2$, respectively, of $T$, then the party receives 1 if $x_1 = x_2$, and 0 otherwise. If the contents of $i_1$ or $i_2$ are empty, then the party receives $\perp$.

All non-equality queries incur unit cost, and we define hardness analogously to the RR model. We denote the oracles together as $\mathbb{G}_{\mathrm{Ma}}$. For an algorithm $A$ that makes queries to $\mathbb{G}_{\mathrm{Ma}}$, we write $A^{\mathbb{G}_{\mathrm{Ma}}}$. As in the RR model, we will imagine there is a fixed generator $g$, and a row containing $x$ will correspond to the group element $g^x$. The group operation queries therefore take the group elements in two positions, and write the desired combination of them to the third position.

*Remark 3.1.* Our convention of zero-cost equality queries follows Maurer, and better reflects reality: it is easy to show a lower bound of $\Omega(p)$ cost for discrete logarithms when counting equality queries. Yet baby-step-giant-step only takes $\Theta(\sqrt{p})$ cost in the standard and RR/Shoup models, using a data structure that cannot be simulated with just equality queries. With free equality queries, the cost in Maurer's model is the correct $\Theta(\sqrt{p})$. [MPZ20] takes a different approach, refining Maurer's model to allow more sophisticated queries that can implement the required data structure with $\Theta(\sqrt{p})$ queries. Regardless, the number of possible equality queries is at most quadratic in the number of wires, so this convention only makes a polynomial difference, which does not effect our results.

*Challenges of Maurer's Model.* Maurer's model makes sense for reasoning about computational problems, such as discrete logarithms or Diffie-Hellman. Here, the table is initialized to contain the problem instance in the first several rows, with the rest of the table as scratch space for performing computations.

However, the model is potentially problematic when reasoning about cryptosystems, where different components of the cryptosystem (or even the same component run multiple times) are using the group. Since the oracle is stateful, this can cause bad behavior of the algorithms. For example, the security experiment for one-wayness runs the function twice, once to generate the adversary's input, and once to check its output. If running the function causes the table values to change, then the outputs of the function is not deterministic. Worse, the adversary may influence the outputs by writing values to the table, which could make inversion easier. Of course this is not an issue in the real world, but it demonstrates an issue when trying to apply Maurer's model to cryptosystems.

### 3.3 The Type Safe (TS) Model

Here we offer a model that tries to capture the intuitive properties of Maurer's model, while also having a stateless oracle to avoid the issues above. This model is, in fact, implicit in many works claiming to use Maurer's model [RSS20, SGS20, SGS21, DHH+21], but has not to our knowledge been formally written down.

Let $p \in \mathbb{Z}$ be a positive integer. An algorithm $A$ will be given as a circuit. Unlike a standard binary circuit, the circuit for $A$ will have the following features:

– There will be two kinds of wires, *bit* wires and *element* wires. Bit wires take values in $\{0, 1\}$, whereas element wires take values in $\mathbb{Z}_p \cup \{\perp\}$.

– There will be “bit gates” that map bits to bits. These gates *cannot* take element wires as input. Any universal gate set is allowed for the bit gates.
– Additionally, there will be a few special “element gates”, whose inputs and/or outputs include element wires:
  - **Labeling Gate.** This takes as input $\lceil \log_2(p) \rceil$ bit wires, and interprets them as $x \in \mathbb{Z}_p$. Its output is an element wire, containing $x$. This element wire will be thought of as corresponding to the value $g^x$. If the input wires do not correspond to an $x \in \mathbb{Z}_p$, the output wire will contain $\perp$.
  - **Group Operation Gate.** This takes as input $2 \times \lceil \log_2(p) \rceil$ bit wires and 2 element wires. The bit wires are interpreted as $a_1, a_2 \in \mathbb{Z}_p$. Let $x_1, x_2$ be the contents of the element wires. The output wire is an element wire, set to $a_1 x_1 + a_2 x_2$. If any of the bit or element input wires do not correspond to elements of $\mathbb{Z}_p$, then the output wire is set to $\perp$.
  - **Equality Gate.** This takes as input two element wires, and outputs a bit wire. If the input wires both contain the same $x \in \mathbb{Z}_p$, the output wire is set to 1. In all other cases (including $\perp$ inputs) the output is 0.

We somewhat abuse notation, and for an algorithm $A$ in the type safe model, we write $A^{\mathbb{G}_{\mathrm{TS}}}$. Our cost metric for circuits in the TS model will count only labeling and group operation gates, with bit and equality gates being free. Free bit gates corresponds to the other generic group models, where queries are bounded but computation outside of queries is free. Free equality gates are used for the same reason as equality queries in Maurer's model.

A game $(\mathsf{Ch}, t)$ in the TS model allows all parties (the challenger and one or more adversaries) to use labeling and group operation gates, and send both bit and element wires to each other. We define hardness as in the previous models.

In the TS model, we will think of element wires as containing $\log_2 p$ bits. Therefore, if an algorithm has $k_1$ element wires as input and $k_2$ bit wires, its overall input size will be $k_1 \log_2 p + k_2$.

### 3.4 Examples

We now discuss several examples of cryptosystems based on groups, to illustrate the differences between the TS and RR models.

*One-way functions.* Discrete logs give a simple one-way function $f(x) = g^x$. This function easily maps to the RR model as $f(x) = L(x)$, which is evaluated by a single labeling query. The function *also* maps to the Type Safe model, but with some caveats. Namely, $f(x)$ is simply a labeling gate, which outputs an element wire. Thus, in the TS model, $f(x)$ does not map bits to bits, but rather maps bits to *elements*. This makes sense, but it means that the outputs of $f$ cannot be operated on at the bit level, and in particular cannot be fed back into $f$.

*Pseudorandom Generators.* Consider the Blum-Micali [BM82] PRG: on input $x$, let $x_0 = x$. Then define $x_i = g^{x_{i-1}}$ for $i = 1, \ldots, n$, where $n$ is the number of desired outputs. Then for each $i$, output a hardcore bit $b_i$ extracted from $x_i$.

Blum-Micali easily translates to the RR model: just let $x_i = L(x_{i-1})$. The only caveat is that the set of labels must be $\log p$ bits so the domain and range of $x \mapsto g^x$ are essentially identical. On the other hand, Blum-Micali does *not* work in the Type Safe model: $x_1$ is an element wire, and so it cannot be fed into another labeling gate in order to derive $x_2$.

Other standard PRGs work in the TS model. Consider $G(x, y) = (g^x, g^y, g^{xy})$, which is secure under the decisional Diffie-Hellman assumption. To evaluate, compute $xy$ over $\mathbb{Z}_p$ using standard circuit gates, and then apply labeling gates to $x$, $y$, and $xy$. It is straightforward to generalize $G$ to obtain arbitrary stretch.

*Pseudorandom Functions.* Once we have a pseudorandom generator like $G(x, y)$ above, we may hope to build a pseudorandom *function* following Goldreich, Goldwasser, and Micali [GGM84]. This construction takes any length-doubling PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$, and constructs a PRF as follows. Define $G_0$ to be the first $n$ bits of the output of $G$, and $G_1$ to be the second $n$ bits. For a key $k \in \{0,1\}^n$ and input $x \in \{0,1\}^m$, define $F(k, x) = G_{x_m}(G_{x_{m-1}}(\dots G_{x_1}(k)\dots))$.

For a PRG $G$ in the RR model, the PRF of [GGM84] readily also translates to the RR model. However, the same is not true for the Type Safe model. For example, our $G$ from above takes bits as input but outputs *element* wires, and so the outputs cannot be fed back into $G$ as in [GGM84].

Nevertheless, there are PRGs in the TS model, namely Naor-Reingold [NR97]. The secret key consists of $\alpha_0, \dots, \alpha_m$, and $F(k, x) = g^{\alpha_0 \prod_{i=1}^m \alpha_i^{x_i}}$. To evaluate in the TS model, simply compute $\alpha_0 \prod_{i=1}^m \alpha_i^{x_i}$ and then apply a labeling gate.

*Preprocessing attacks.* Several works [CK18, BL22] have studied pre-processing attacks on groups, where an expensive pre-processing stage stores information about the labeling function of the group. These attacks make sense in the RR model, but do not appear to have any meaning in the TS model.

The above examples already begin to show that the Type Safe model does not capture all common cryptographic techniques one may apply to groups. In Section 4, we strengthen these observations to show some concepts that are simply impossible in the TS model, despite there being standard-model constructions from groups. On the other hand, the Random Representation model seems to capture known black-box techniques, and does not suffer from these limitations.

Next, we prove certain positive relationships between the TS and RR model. These results are analogous to the theorem of [JS08], which is often cited as proving the equivalence between Shoup's and Maurer's generic group models. Our results below formalize what this equivalence means, and where it falls short.

### 3.5 Compiling TS To RR

Here we show that any algorithm which exists in the TS model also exists in the RR model. By applying this to cryptosystems and games, we see that any technique which is captured by the TS model is also captured by the RR model.

By applying this to adversaries, we see that security in the RR model implies security in the TS model, amongst games the TS model (and hence in both models). These results are an adaptation of one direction of the proof of [JS08].

In more detail, we show that there is a *canonical translation* of any algorithm (or set of interacting algorithms) into the RR model.

**Definition 3.2.** *Let $A^{\mathbb{G}_{TS}}$ be an algorithm in the TS model. We then define the* canonical translation *of $A^{\mathbb{G}_{TS}}$ into the RR model, which we will denote as $A^{\mathbb{G}_{RR}}$, which is identical to $A^{\mathbb{G}_{TS}}$ except that:*

- *All element wires are replaced by collections of bit wires, which together are interpreted as labels.*
- *Labeling and group operation gates are replaced with the corresponding labeling and group operation queries.*
- *Equality gates are replaced by string comparison sub-routines.*

*For a set of interactive algorithms $A^{\mathbb{G}_{TS}} = (A_1^{\mathbb{G}_{TS}}, ...A_k^{\mathbb{G}_{TS}})$ in the TS model, we define the canonical translation as $A^{\mathbb{G}_{RR}} = (A_1^{\mathbb{G}_{RR}}, \ldots, A_k^{\mathbb{G}_{RR}})$.*

**Theorem 3.3.** *Let $A^{\mathbb{G}_{TS}}$ be a complete set of interactive algorithms in the TS model, whose final output is a set of bit wires. Let $A^{\mathbb{G}_{RR}}$ be its canonical translation. Then the output distributions of $A^{\mathbb{G}_{TS}}$ and $A^{\mathbb{G}_{RR}}$ are identical.*

*Proof.* To see that the distributions of outputs are identical between $A^{\mathbb{G}_{\mathrm{TS}}}$ and $A^{\mathbb{G}_{\mathrm{RR}}}$, we observe that, in the TS model, it is equivalent to consider the element wires as containing $L(x)$ instead of $x$. Each algorithm in $A^{\mathbb{G}_{\mathrm{RR}}}$ then simply replaces each element wire with bit wires, but still containing $L(x)$, and evaluates the equality gates for itself using string comparison. $\square$

Now let $\Pi$ be a protocol and $(\mathsf{Ch}, t)$ an associated security game in the TS model. Let $\Pi', (\mathsf{Ch}', t)$ be the canonical translation into the RR model.

**Theorem 3.4.** *If $(\mathsf{Ch}', t)$ is hard in the RR model, then $(\mathsf{Ch}, t)$is hard the TS model. This holds whether or not $\mathsf{Ch}, \mathsf{Ch}'$ are single-stage games.*

*Proof.* Toward contradiction, let $\mathsf{A} = (\mathsf{A}_1, \ldots, \mathsf{A}_k)$ be an adversary playing the game $\mathsf{Ch}$ in the TS model, and winning with probability $q$, which is non-negligibly greater than $t$. Let $\mathsf{A}'$ be the canonical translation of $\mathsf{A}$. Then $(\mathsf{A}', \mathsf{Ch}')$ is the canonical translation of $(\mathsf{A}, \mathsf{Ch})$, which are complete sets of interacting algorithms. By Theorem 3.3, the output distribution is identical, which contradicts the hardness of $(\mathsf{Ch}', t)$. $\square$

### 3.6 From TS Security to RR Security for Single-Stage Games

Now, we visit the other direction of the equivalence claimed in [JS08], showing that TS security implies RR security *sometimes*, namely in single-stage games.

**Theorem 3.5.** *If $\mathsf{Ch}$ is a* single-stage *game and $\Pi$ is $(\mathsf{Ch}, t)$-secure in the TS model, then $\Pi'$ is $(\mathsf{Ch}', t)$-secure in the RR model, where $(\mathsf{Ch}', t)$ is the canonical translation of $(\mathsf{Ch}, t)$.*

The proof is given in the Full Version [Zha22]. The intuition is that an adversary $\mathsf{A}'$ for $\Pi'$ is compiled into an adversary $\mathsf{A}$ for $\Pi$, where $\mathsf{A}$ lazily simulates the RR labeling function with a table $T$, using its TS gates to ensure consistency.

Note that Theorem 3.5 only applies to single-stage games. If we try applying the proof to a multi-stage adversary, our new adversary has to maintain the table $T$, which must be shared across all the adversaries to maintain consistency. Thus, we actually obtain a single-stage adversary, violating the requirements of the game. Section 5 demonstrates that this limitation is inherent, giving a multi-stage cryptosystem that is secure in the TS model but not in the RR model.

# 4 Further Impossibilities in the Type Safe Model

Here, we give impossibility results in the Type-Safe (TS) generic group model, which nevertheless have standard-model constructions from cryptographic groups and moreover translate to and have security in the Random Representation/Shoup model. These impossibilities are for textbook cryptographic applications, showing a significant weakness for the TS model.

We first state the following lemma, which is implicit in numerous works, and is the main justification for the Algebraic Group Model [FKL18] being implied by the generic group model. The lemma is proved in the Full Version [Zha22].

**Lemma 4.1.** *Consider any deterministic algorithm $A(\mathbf{h}, x)$ in the TS model whose input contains a vector $\mathbf{h}$ of $n$ group elements and which outputs a single group element. Then there is another deterministic algorithm $E(\mathbf{h}, x)$ whose run-time and query complexity is linearly related to $A$, such that $E(x)$ outputs a vector $\mathbf{v} \in \mathbb{Z}_p^{n+1}$ satisfying $A(\mathbf{h}, x) = (g, \mathbf{h})^{\mathbf{v}}$.*

Note that, while Lemma 4.1 discusses deterministic algorithms, it can readily be applied to randomized algorithms by supplying the same random coins to $A$ and $E$, making them deterministic functions of the random coins.

As is typical in the generic group literature, we will also observe that any equality gate can be thought of as a linear test on $(g, \mathbf{h})$: let $\mathbf{v}_1, \mathbf{v}_2$ be the vectors guaranteed by Lemma 4.1 for the two gate inputs. Then the equality gate tests whether $(g, \mathbf{h})^{\mathbf{v}_1 - \mathbf{v}_2} = 1$. We call the linear test *trivial* if $\mathbf{v}_1 = \mathbf{v}_2$, and *non-trivial* if $\mathbf{v}_1 \neq \mathbf{v}_2$. Note that trivial tests will always output 1 (denoting equal), whereas the result of non-trivial tests will depend on the elements in $\mathbf{h}$.

## 4.1 Collision Resistant Domain Extension

Here, we will prove that any hash function $H$ in the TS model must have somewhat large hashing keys. This is in contrast to the standard model, where domain extension allows for constant-sized keys. Intuitively, domain extension typically operates on outputs of $H$ (say, by feeding them back in as inputs), which may be group elements protected by the type system.

**Theorem 4.2.** *Let $H$ be a collision resistant hash function in the TS model with key length $k$, input length $n$, and output length $m$. Let $p$ be the group order. Then $n \leq 1 + m \times (k + \log p)$.*

*Proof.* Suppose $H$ has key space $\mathbb{G}^{k_1} \times \{0,1\}^{k_2}$, domain $\mathbb{G}^{n_1} \times \{0,1\}^{n_2}$, and range $\mathbb{G}^{m_1} \times \{0,1\}^{m_2}$. We assume for simplicity that $p$ is a power of 2; the general case follows from the same arguments, but more care is needed to track parameter sizes. Then $k = k_2 + k_1 \log p$, $n = n_2 + n_1 \log p$, and $m = m_2 + m_1 \log p$. Then our goal is to show that $n \leq 2 + m \times (k + \log p)$. We will prove a stronger statement, namely that: $n \leq 1 + m(k_1 + 1) \log p$.

We first make two simplifying assumption, which we argue are wlog.

**Simplifying Assumption 4.3.** $n_1 = 0$*, meaning* $n = n_2$*.*

In other words, the input consists entirely of bits and no group elements.

**Lemma 4.4.** *For any $H$, there exists a new collision resistant hash function $H'$ with the same domain, range, and key size as $H$, but which satisfies Simplifying Assumption 4.3.*

*Proof.* $H'$ is defined as $H(k,\ (\alpha_1, \ldots, \alpha_{n_1}, x)\ ) = H'(k,\ (g^{\alpha_1}, \ldots, g^{\alpha_{n_1}}, x))$; $H'$ uses the same key sampling algorithm $\mathsf{Gen}$ as $H$. Clearly, any collision for $H'$ can be converted into a collision for $H$, so if $H$ is collision resistant, then so is $H'$. Moreover, this change from $H$ to $H'$ preserves $n = n_2 + n_1 \log p$. □

Let $K_2 \in \{0,1\}^{k_2}$ be the bits of the key, and $K_1 \in \mathbb{G}^{k_1}$ be the group elements. For a key $(K_1, K_2)$, we say that an input $x$ is "good" if all the non-trivial linear equations over $K_1$ queried during evaluating $H(x)$ evaluate to non-zero. We will pick an inverse polynomial $\delta$ (in fact, a constant), to be specified later.

**Simplifying Assumption 4.5.** *Except with negligible probability over the choice of key* $(K_1, K_2) \leftarrow \mathsf{Gen}()$*, a* $1 - \delta$ *fraction of* $x$ *in the domain are good.*

**Lemma 4.6.** *For any $H$ satisfying Simplifying Assumption 4.3, there exists $H'$ satisfying Simplifying Assumptions 4.3* and *4.5, with* $n' = n, m' = m, k_1' \leq k_1$*.*

Lemma 4.6 is proved in the Full Version [Zha22]. The idea is that"bad" inputs yield a linear equation over the exponents of the key. By solving the linear system, one can solve for one element of the hashing key in terms of others, compiling $H$ into a new hash function with fewer bad inputs. One can iterate this process until the fraction of bad inputs becomes sufficiently small. This process expands the bit part of the hashing key ($k_2$), but this is fine since it is independent of the stronger bound $n \leq 1 + m(k_1 + 1) \log p$ that we will prove.

With our simplifying assumptions, we can now finish the proof, which is given in detail in the Full Version [Zha22]. The idea is that for good $x$, we can write the output $(O_1, O_2) \in \mathbb{G}^{m_1} \times \{0,1\}^{m_2}$ as $O_1 = {}^{\mathbf{A}(K_2,x)}(g, K_1) \quad O_2 = F(K_2, x)$ for $\mathbf{A} : \{0,1\}^{k_2} \times \{0,1\}^n \to \mathbb{Z}_p^{m_1 \times (k_1+1)}$ and $F : \{0,1\}^{k_2} \times \{0,1\}^n \to \{0,1\}^{m_2}$. Recall that a left superscript means left multiplication in the exponent. A collision amongst good $x$ for the map $x \mapsto (\mathbf{A}(K_2, x), F(K_2, x))$ therefore yields a collision for the hash function. This means that $(\mathbf{A}(K_2, x), F(K_2, x))$ must be injective for good $x$, since otherwise a collision can be computed inefficiently without making any queries. Since a $1 - \delta$ fraction of $x$ are good, this allows us to lower bound the output length of $(\mathbf{A}(K_2, x), F(K_2, x))$, thereby giving our bound. □

### 4.2 Pseudorandom Permutations

**Theorem 4.7.** *Let $F, F^{-1}$ be an efficient keyed permutation pair in the TS model. Then it is not a secure PRP.*

Theorem 4.7 will be the immediate consequence of the following three lemmas:

**Lemma 4.8.** *Let $F$ be an efficient keyed function in the TS model, such that the output contains at least one bit. Then $F$ is not a secure PRF.*

**Lemma 4.9.** *Let $F$ be an efficient keyed function in the TS model, such that the input contains at least one group element. Then $F$ is not a secure PRF.*

**Lemma 4.10.** *Let $F, F^{-1}$ be an efficient keyed permutation pair in the TS model. Then the number of group elements in the domain and range must be equal.*

Lemmas 4.9, 4.8, and 4.10 are proved in the Full Version [Zha22]; here we sketch the high-level idea. For Lemma 4.8, the idea is to first ignore all the group element outputs and just focus on the bit output. As with Simplifying Assumption 4.3, we can replace all key element wires with bit wires. Then we argue that the resulting function can be computed without element gates at all, and hence insecure against computationally unbounded (but element gate-bounded) adversaries.

For Lemma 4.9, the idea is that any function with group element inputs must be linear in those inputs. But linear functions cannot be pseudorandom.

Finally, for Lemma 4.10, suppose the number of group elements were not equal. By potentially exchanging the roles of $F, F^{-1}$, we assume $F$ has fewer group element outputs than it's inputs. Now consider running $F$ on a random input. Since $F$ is a permutation, the output must information-theoretically encode the input. But since there are now fewer group elements, this means some information about the exponents of the input must now be present in the bit wires of the output. We show how to embed a discrete log challenge into the input, and then extract this information from the bit wires using $F^{-1}$ and Lemma 4.1, resulting in computing the discrete log. But discrete logs are intractable in the TS model.

### 4.3 Efficient CPA-Secure Encryption for Message Strings

We now prove that any CPA-secure encryption scheme in the TS model, whose domain is bits (as opposed to group elements), must have the number of group elements in the ciphertext be approximately at least the bit-length of the message. As group elements are $\log p$ bits, ciphertexts are a $\log p$ factor larger than messages.

**Definition 4.11.** *A CPA-secure encryption scheme consists of a pair of efficient probabilistic algorithms $\mathsf{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}, \mathsf{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ satisfying:*

- **Correctness:** $\forall K \in \mathcal{K}, M \in \mathcal{M}$, $\Pr[\mathsf{Dec}(K, \mathsf{Enc}(K, M)) = M] \geq 1 - \mathsf{negl}$.
- **Chosen Plaintext Security:** *Consider an adversary $\mathsf{A}$ playing the following game with a challenger. The challenger first chooses a random $K \leftarrow \mathcal{K}$, and chooses a random bit $b \in \{0, 1\}$. Then $\mathsf{A}$ makes queries on message pairs $(M_0, M_1)$, and receives $\mathsf{Enc}(k, M_b)$. The adversary outputs a guess $b'$ for $b$. We require that, for any efficient $\mathsf{A}$, $\Pr[b' = b] \leq 1/2 + \mathsf{negl}$.*

**Theorem 4.12.** *Let* $\mathsf{Enc}, \mathsf{Dec}$ *be a CPA-secure encryption scheme which compiles in the TS model. Suppose the message space is bit-strings* $\{0,1\}^n$*. If the ciphertext space is* $\mathbb{G}^{m_1} \times \{0,1\}^{m_2}$*, then* $m_1 \geq \Omega(n/\log\lambda)$*.*

*Proof.* Let $\mathsf{Enc}, \mathsf{Dec}$ be a CPA-secure encryption scheme in the TS model. Note that we can consider the bit portion of the ciphertexts alone as an encryption scheme; there is no correctness, but we can still consider security which is implied by the security of the full $(\mathsf{Enc}, \mathsf{Dec})$. Now suppose the bit portion was not statistically independent of the message conditioned on the secret key $K$. Then by the standard impossibility of statistically secure CPA-secure encryption in the standard model, there is an inefficient query-less attacker that would break CPA-security, thus violating the security of $(\mathsf{Enc}, \mathsf{Dec})$. Therefore we know that the bit portion must be statistically independent of the message.

Now consider decrypting a ciphertext $c \in \mathbb{G}^{m_1}$. $\mathsf{Dec}$ will make a polynomial number $T$ of equation queries over $(1, c)$. The message outputted will be some function of the results and the bit portion of the ciphertext. We can assume without loss of generality that $\mathsf{Dec}$ never makes a query that is linearly dependent on previous queries which returned 0. Indeed, $\mathsf{Dec}$ can correctly predict the output of the query will be 0. Therefore, there will be at most $m_1$ queries that will result in zero, since the dimension of the zero queries is $m_1$.

This means there is $\ll 1$ bit of information about the message in the bit-portion of the ciphertext, and only $\log_2 \binom{T}{m_1} \leq m_1 \log_2 T$ bits of information in the queries, and therefore the message length $n$ is at most this quantity. Hence, $m_1 \geq n/\log_2 T$. Since $T$ is polynomial in the security parameter, this gives the desired lower-bound on the ciphertext size. $\square$

# 5 On the Insecurity of the Type-Safe Model for Multi-stage Games

Here, we show that security in the TS model does *not* imply security in the RR model. We first define deterministic public key encryption, following [BBO07].

**Definition 5.1.** *A deterministic encryption scheme is a triple of efficient algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *where* $\mathsf{Enc}, \mathsf{Dec}$ *are deterministic such that:*

- **Correctness:** *For all* $m$*,* $\Pr[\mathsf{Dec}(\mathsf{sk}, c) = m : {}^{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}()}_{c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)}] \geq 1 - \mathsf{negl}$.
- $\ell$**-Security:** *For any two distributions* $D_0, D_1$ *with min-entropy at least* $\ell$*, and for any efficient probabilistic adversary* $\mathsf{A}$*,* $|\Pr[A(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, D_0)) = 1] - \Pr[A(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, D_1)) = 1]| \leq \mathsf{negl}$*, where* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}()$.

Here, the min-entropy of a distribution $D$ is $H_\infty(D) = \min \log_2 \Pr[x \leftarrow D]^{-1}$. Since $\mathsf{Enc}$ is deterministic, for security to be possible at all, we require $\ell = \omega(\log\lambda)$. For security to be non-trivial, we usually ask that $\ell \ll |m|$.

Note that $\ell$-security is *not* single-stage, since the definition quantifies over $D_0, D_1$ and $A$, and $A$ cannot see the random coins of $D_0, D_1$.

**Construction 5.2.** *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be the following deterministic encryption scheme for messages in* $\mathbb{Z}_p$*:*

- $\mathsf{Gen}^{\mathbb{G}_{TS}}()$*: run* $a_1, \dots, a_n \leftarrow \mathbb{Z}_p$*, and output* $\mathsf{sk} = (a_1, \dots, a_n), \mathsf{pk} = (h_1 = g^{a_1}, \dots, h_n = g^{a_n})$*.*
- $\mathsf{Enc}^{\mathbb{G}_{TS}}(\mathsf{pk}, m)$*: run* $c_0 \leftarrow g^m$ *and* $c_i = h_i^m g^{m_i}$ *for* $i = 1, \dots, n$*.*
- $\mathsf{Dec}^{\mathbb{G}_{TS}}(\mathsf{sk}, c)$*: for* $i = 1, \dots, n$*, compute* $u_i = c_0^{a_i}$*. If* $c_i = u_i$*, set* $m_i = 0$*; otherwise set* $m_i = 1$*.*

It is straightforward that Construction 5.2 is correct: $u_i = g^{ma_i} = h_i^m$. Therefore $c_i = u_i$ if and only if $m_i = 0$.

**Theorem 5.3.** *Construction 5.2 is an* $\ell$*-secure deterministic encryption scheme in the TS model for* $\ell = \log_2 p - 2$*, but is insecure in the RR model or in any standard-model instantiation of the group.*

The proof is given in the Full Version [Zha22]. For *insecurity* in the RR and standard models, the idea is to have $D_0, D_1$ be random messages conditioned on, say, the first bit of $g^m$ being 0 and 1, respectively. These have high min-entropy, but also allow the adversary to trivially distinguish the two cases by looking at the first bit of $c_0$. For *security* in the TS model, we first note that this bit fixing strategy does not apply since the distribution cannot have direct access to the bits of $g^m$. We give a simple proof that *no* distinguishing strategy is possible.

## 6 TS Un-instantiability

Here, we give an example of a protocol in the TS model that is secure under a single-stage game (and hence also in the RR model, by Theorem 3.5) and yet is insecure under any standard-model instantiation of the group.

### 6.1 Overview

We first explain limitations of the existing works on the un-instantiability of idealized models, as well as sketch our solution. The first such un-instantiability result was for random oracles (RO), a predecessor of generic groups, where one models an hash function as a truly random function. Canetti, Goldreich, and Halevi [CGH98] give contrived schemes that are secure in the random oracle model, but which are insecure under any instantiation of the oracle by a concrete hash function. Their impossibility works by identifying a security property of a hash function $H$ called *correlation intractability*, which requires that for any "sparse" relation over input/output pairs, it is computationally infeasible to find an input $x$ such that $(x, H(x))$ satisfies the relation. Correlation intractability is trivially satisfied by random oracles, but [CGH98] show that it is impossible (in certain parameter regimes) for standard-model hash functions. They then build contrived cryptosystems where an input/output pair satisfying the relation causes some clearly insecure behavior: e.g. the secret key holder completely reveals their key, or the encrypter encrypts uses the identity function. In the random oracle model, this will never happen and the system will remain secure. But in the standard model, the attacker simply uses the impossibility for correlation intractability to find such an input. triggering a complete break of the system.

This idea was translated to generic groups by Dent [Den02], who uses similar ideas but where the hash function is replaced by the group labeling function. Another way to obtain a separation is through the recent work of Zhandry and Zhang [ZZ21], who show that the labeling function of a generic group, when properly truncated, gives a random oracle, which in turn is impossible.

However, both of these results crucially rely on the labeling function; that is, they only work in the RR/Shoup model. There are at least a couple reasons why the un-instantiability result appears to not generalize to the TS model. First, [CGH98, Den02] use a random oracle to instantiate Micali's CS proofs [Mic94], which in turn requires a domain-extending Merkle tree; we already showed (Section 4) that domain extension is impossible in the TS model.

More fundamentally, we show in the Full Version [Zha22] that correlation intractability *cannot* separate the TS and standard model, regardless of the construction. Concretely, we give a simple hash function which is correlation intractable (in the standard model) with respect to every evasive relation that exists in the TS model. We note that, the cryptosystems derived from correlation intractability [CGH98] must execute the relation, and therefore the relation must exist in the TS model if we want to use it for an un-instantiability result.

Other un-instantiability results [Nie02, GK03, BBP04, BFM15] have similar issues to the above. Meanwhile, [BCPR14] give an un-instantiability result that works in the TS model, but it is a multi-stage game and requires strong computational assumptions, namely indistinguishability obfuscation.

We fill in this gap, showing that the single-stage TS model is unconditionally un-instantiable, albeit via very different, and arguably simpler, techniques. We start with ElGamal, but where message bits are encrypted bit by bit. This is easily proved secure in the standard model. We then modify the scheme to append a single bit, which is a contrived leakage function of the message and randomness.

In either the TS or RR model, we show this leakage offers minimal advantage to breaking the cryptosystem. However, in the standard model, we show that by encrypting the description of (that is, the code) of the group, the ciphertext, leakage included, easily reveals one bit about the message. This breaks security.

The main challenges are two-fold. First we must make sure the encryption scheme can actually be decrypted in the TS model, which we showed is not trivial in Section 4. A more difficult problem is to maintain security in the generic group models. This is challenging because our leakage function interprets the message as arbitrary code and runs it on some inputs. Without care, this arbitrary code could already break security without having anything to do with the generic group. For example, the code could just be a constant function, in which case the leakage reveals which constant. We give our solution in Section 6.2. In Section 8 we give another un-instantiability result for one-time MACs in the context of the Algebraic Group Model (AGM), which also applies to the TS model.

### 6.2 Our Un-instantiable construction

**Construction 6.1.** *Let* $(\mathsf{Gen}^{\mathbb{G}_{TS}}, \mathsf{Enc}^{\mathbb{G}_{TS}}, \mathsf{Dec}^{\mathbb{G}_{TS}})$ *be the following:*

- $\mathsf{Gen}^{\mathbb{G}_{TS}}(1^\lambda)$*: sample* $\alpha \leftarrow \mathbb{Z}_p \setminus \{0\}$*, and let* $\mathsf{pk} = (g, h = g^\alpha, \lambda)$ *and* $\mathsf{sk} = \alpha$*.*
- $\mathsf{Enc}^{\mathbb{G}_{TS}}(\mathsf{pk}, m)$*: Let* $n$ *be the bit-length of* $m$*. For each* $i \in [n]$*, sample* $r_i \leftarrow \mathbb{Z}_p$ *and let* $c_i = g^{r_i}, d_i = h^{r_i+m_i}$*, where* $m_i \in \{0,1\}$ *is the ith bit of* $m$*. Let* $e \leftarrow L(m, \{r_i\}_i$ *where* $L$ *is defined below. Output output* $c = (\{c_i, d_i\}_{i\in[n]}, e)$*.*
- $\mathsf{Dec}^{\mathbb{G}_{TS}}(\mathsf{sk}, c)$*: for each* $i \in [n]$*, compute* $d'_i = c_i^\alpha$*. If* $d'_i = d_i$*, set* $m_i = 0$*; otherwise set* $m_i = 1$*. Output* $m = m_1 m_2 \cdots m_n$*.*

$L(m, \{r_i\}_i)$ *works as follows: interpret the last* $n - \lambda$ *bits of* $m$ *as the description of a function* $H : \mathbb{Z}_p \to \{0,1\}$ *in some canonical way. Then:*

- *Test if* $H$ *is "balanced" by sampling* $k = 32\lambda$ *random* $s_j \leftarrow \mathbb{Z}_p, j = 1, \ldots, k$*, computing* $b_i \leftarrow H(s_i)$*, and checking that* $\sum_{j=1}^k s_i \in (3k/8, 5k/8)$*.*
- *If* $H$ *is not balanced, sample a random bit* $e \leftarrow \{0,1\}$*. Otherwise, let* $e = H(r_1) \oplus H(r_2) \oplus \cdot \oplus H(r_\lambda)$*. Output* $e$*.*

**Theorem 6.2.** *Construction 6.1 is a secure public key encryption scheme in the TS generic group model (and hence also in the RR model, by Theorem 3.5). However, Construction 6.1 is insecure in the standard model.*

*Proof.* We first show that the construction is correct and secure. Correctness is a straightforward adaptation of the correctness of ElGamal: $d_i = h^{r_i+m_i} = g^{\alpha r_i + \alpha m_i} = c_i^\alpha h^{m_i}$. Therefore $d_i = c_i^\alpha$ if and only if $m_i = 0$.

For security, we first show the followingin the Full Version [Zha22]:

**Lemma 6.3.** *For any message* $m$*, the bit* $e$ *is statistically close to a uniform random bit. Concretely,* $|\Pr[e = 0] - 1/2| \leq 2^{-\lambda}$*.*

The proof idea is $e$ is the XOR of many independent samples from the not-too-biased outputs of $H$. Lemma 6.3 above shows that an adversary cannot break Construction 6.1 just by looking at the bit $e$. We now expand this to consider general adversaries which also get $c_i, d_i$:

**Lemma 6.4.** *Construction 6.1 is secure in the TS generic group model.*

Lemma 6.4 is proved in the Full Version [Zha22]. There we give a simple proof that a TS model adversary essentially must break either the $c_i, d_i$, or the bit $e$, but gains no advantage by considering both together. $c_i, d_i$ is just plain ElGamal, which hides the message; Lemma 6.3 shows that $e$ hides the message as well.

Finally, we show that no matter how the group is instantiated, Construction 6.1 is insecure in the standard model

**Lemma 6.5.** *For any standard-model group, Construction 6.1 is insecure.*

*Proof.* Consider instantiating Construction 6.1 with an arbitrary standard model group scheme. Then consider the following adversary $\mathsf{A}$.

– On input $\mathsf{pk} = (g, h)$, let $\ell$ be the bit-length of group elements.
– Choose a random $t \in \{0,1\}^\ell$, and construct the circuit $H(s) = \langle t, h^s \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the inner product mod 2 of the bit string inputs.
– Send to the challenger the challenge messages $(m_0 = (0^\lambda, H), m_1 = (1^\lambda, H))$. Receive the ciphertext $c = (\{c_i, d_i\}_i, e)$.
– Output $e \oplus \langle t, d_1 \oplus \cdots \oplus d_\lambda \rangle$.

We now analyze $\mathsf{A}$. First, since inner products are good extractors, $H$ is balanced with overwhelming probability. Thus $e = H(r_1) \oplus H(r_2) \oplus \cdot \oplus H(r_\lambda)$.

Suppose $c$ is an encryption of $m_0$. Then $\langle t, d_i \rangle = \langle t, h^{r_i} \rangle = H(r_i)$. Hence, with overwhelming probability, the output of $\mathsf{A}$ is exactly 0. Next suppose $c$ is an encryption of $m_1$. Then $\langle t, d_i \rangle = \langle t, h^{r_i+1} \rangle$. Hence, if we define $W = (h^{r_1} \oplus h^{r_1+1}) \oplus \cdot \oplus (h^{r_\lambda} \oplus h^{r_\lambda+1})$, the output is equal to $\langle t, W \rangle$. The following is proved in the Full Version [Zha22]:

**Lemma 6.6.** $\Pr[W = 0] \leq 1/2$.

Notice that $t$ is independent from the $r_i$, and hence $W$. Hence, if $W \neq 0$, $\langle t, W \rangle$ is a uniform random bit. Therefore the output of $\mathsf{A}$ on $m_1$ is 1 with probability at least $1/4$, giving it a non-negligible advantage. $\square$

Putting together Lemmas 6.4 and 6.5 proves Theorem 6.2. $\square$

## 7 Impossibility of IBE from Generic Groups

Papakonstantinou, Rackoff, and Vahlis [PRV12] give an impossibility of identity-based encryption (IBE) from generic groups. The authors cite Shoup's model, and like Shoup they define the generic group model as a random mapping from $\mathbb{Z}_p$ into bit strings. However, we argue that their definitions and proofs actually lie somewhere between Shoup's and the Maurer/TS model. For example, consider their definition of a *generic algorithm* ([PRV12], page 6):

> A generic algorithm $A$ is a probabilistic algorithm (or with randomness in its input) that takes inputs and produces outputs of the form $(w, g_1, \ldots, g_k) \in (\{0,1\}^* \times \mathbb{G}^k)$ for an arbitrary $k \in \mathbb{N}$. $A$ is given oracle access to $\mathcal{O}$ restricted to sums that have non-zero coefficients only for the elements $g_1, \ldots, \gamma_k$.

Above, $\mathcal{O}$ is their notation for the oracle implementing the generic group. Requiring an algorithm to be explicitly given group elements as input, and then only allowing queries on linear combinations of those explicit group elements, is a TS-style restriction that is not present in the RR/Shoup model. Since all algorithms must declare the type of inputs they work on, this also means that the components of any cryptosystem (public keys, secret keys, ciphertexts, etc) must explicitly delineate between group elements and bits, just as in the TS model.

This restriction on algorithms plays an important role in the impossibility proof. The proof of [PRV12] proceeds in two steps, where in the first step they compile a generic group IBE scheme into one where user secret keys do not contain any group elements. The second step is to show that such a restricted IBE cannot exist in the generic group model. Unfortunately, the distinction between user keys containing group elements or not is only well-defined if the group elements of secret keys are explicitly labeled, as in the TS model. In the RR model, one could imagine trivially hiding group elements by, say, XORing them with an arbitrary string, or secret sharing into different pieces.

Digging deeper, the impossibility proof does the following many times: compute an encryption of a random message, and then promptly decrypt it, collecting all the queries made during the process. We observe that this process is *exactly* how verification works when compiling IBE to a signature scheme in the usual way. However, signatures were shown *impossible* in the TS model by [DHH+21], who also runs the verification procedure many times to collect queries. But we know that signatures *are* possible in the RR model, so the impossibility of [DHH+21] *only* applies in the TS model. Given these similarities between the impossibility proofs, together with the fact that [PRV12] imposes TS-like restrictions, it is unclear whether the IBE impossibility should extend to the full RR model.

*A full impossibility.* Here, we prove a full impossibility of IBE in the RR generic group model, resolving this gap. We first define IBE; we use a key encapsulation variant for simplicity, which is equivalent to the standard notion.

**Definition 7.1.** *An identity-based encryption (IBE) scheme consists of a tuple of efficient probabilistic algorithms* $(\mathsf{Gen}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ *satisfying:*

- **Correctness:** *For all identities* $\mathsf{id}$*,*
  $\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, c) = k : \substack{(\mathsf{msk},\mathsf{mpk})\leftarrow\mathsf{Gen}() \\ \mathsf{sk}_{\mathsf{id}}\leftarrow\mathsf{Extract}(\mathsf{msk},\mathsf{id}) \\ (c,k)\leftarrow\mathsf{Enc}(\mathsf{mpk},\mathsf{id})}\right] \geq 1 - \mathsf{negl}.$
- **Random Identity Security:** *Consider an adversary* $\mathsf{A}$ *playing the following game. The challenger first chooses a random* $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Gen}()$*, and chooses a random bit* $b \in \{0, 1\}$*. The challenger samples* $q+1$ *random identities* $\mathsf{id}_1, \dots, \mathsf{id}_q, \mathsf{id}^*$*. For* $i = 1, \dots, q$*, is compute* $\mathsf{sk}_{\mathsf{id}_i} \leftarrow \mathsf{Extract}(\mathsf{msk}, \mathsf{id}_i)$*. It also computes* $(c^*, k_0^*) \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*)$ *and samples a uniform* $k_1^*$*. It sends* $\{\mathsf{id}_i, \mathsf{sk}_{\mathsf{id}_i}\}_{i\in[q]}, c^*, k_b^*$ *to* $\mathsf{A}$*, which outputs a guess* $b'$ *for* $b$*. We require that, for all efficient* $\mathsf{A}$ *and polynomial* $q$*,* $\Pr[b' = b] \leq 1 + \mathsf{negl}$*.*

**Theorem 7.2.** *There is no random identity-secure IBE scheme in the RR model.*

*Proof.* Our proof, while different than [PRV12], will follow the same basic outline, though it will replace "secret keys contain no group elements" with a related restriction that is well-defined in the RR model. It will also clarify what about IBE makes it impossible in the RR model, where signatures are possible.

Concretely, we will define a type of *signature* scheme, where generic group queries during verification are independent of the signature (but dependent on the message and public key). Such a signature scheme is rather easily shown to be impossible, in *any* idealized model. This replaces the second part of the proof of [PRV12]. By using a simpler object (signatures instead of IBE) we are able to significantly simplify this part of the proof of [PRV12]. It also offers a clear explanation for the gap between general signatures and IBE in the generic group model, since general signatures will not have the special structure.

The bulk of the proof of Theorem 7.2 is showing that any IBE in the RR model can be compiled into such a restricted signature scheme. The idea is to view the IBE as a signature scheme, which is already well-known. However, the resulting signature scheme has a special structure: a first phase that is independent of the signature, and then a second phase that depends on the signature. Importantly, the second phase is decrypting a ciphertext produced in the first phase. We use the correctness of the IBE to argue we can compile out the queries made in the second phase; this compiling-out step crucially uses the linear structure of groups.

We now give the proof. We first define a *restricted signature scheme*:

**Definition 7.3.** *An restricted signature scheme (R-Sig) relative to an oracle $\mathcal{O}$ consists of a tuple of oracle algorithms* $(\mathsf{Gen}^{\mathcal{O}}, \mathsf{Sign}^{\mathcal{O}}, \mathsf{Ver}^{\mathcal{O}})$ *such that:*

- **$\delta$-Correctness:** $\Pr\left[\mathsf{Ver}^{\mathcal{O}}(\mathsf{PK},\sigma) = 1 : {\scriptstyle \begin{array}{c} M \leftarrow \$ \\ (\mathsf{SK},\mathsf{PK}) \leftarrow \mathsf{Gen}^{\mathcal{O}}() \\ \sigma \leftarrow \mathsf{Sign}^{\mathcal{O}}(\mathsf{SK},M) \end{array}}\right] \geq \delta.$
- **Restricted Structure:** $\mathsf{Ver}^{\mathcal{O}}(\mathsf{PK},M,\sigma) = \mathsf{Ver1}(\ \mathsf{Ver0}^{\mathcal{O}}(\mathsf{PK},M)\ ,\sigma)$*, where* $\mathsf{Ver1}$ *is independent of $\mathcal{O}$, but* $\mathsf{Ver0}$ *is independent of $\sigma$.*
- **0-random message security:** *For any query-bounded adversary* $\mathsf{A}$*,* $\Pr[\mathsf{Ver}^{\mathcal{O}}(\mathsf{PK},M,\sigma) = 1 : {\scriptstyle \begin{array}{c} (\mathsf{SK},\mathsf{PK}) \leftarrow \mathsf{Gen}^{\mathcal{O}}() \\ M \leftarrow \$ \\ \sigma \leftarrow \mathsf{A}^{\mathcal{O}}(\mathsf{PK},M) \end{array}}]$ *is negligible.*

**Lemma 7.4.** *For any oracle $\mathcal{O}$ and any constant $\delta > 0$, R-Sigs do not exist.*

*Proof.* Consider choosing an oracle $\mathcal{O}$, a random $M$, and $(\mathsf{SK},\mathsf{PK}) \leftarrow \mathsf{Gen}^{\mathcal{O}}()$, and then fixing them. We will say that $\sigma$ is "good" if $\Pr[\mathsf{Ver}^{\mathcal{O}}(\mathsf{PK},M,\sigma) = 1] \geq \delta/2$, where the probability is taken over the random coins of $\mathsf{Ver}$. By correctness, with probability at least $\delta/2$ over $\mathcal{O}, M, \mathsf{SK}, \mathsf{PK}$, there will exist *at least* one good $\sigma$, namely the output of $\mathsf{Sign}^{\mathcal{O}}(\mathsf{SK},M)$.

Suppose $\mathsf{Ver0}$ was deterministic. Then we could compute $v \leftarrow \mathsf{Ver0}^{\mathcal{O}}(\mathsf{PK},M)$, and consider the oracle-free probabilistic circuit $C(\sigma) = \mathsf{Ver1}(v,\sigma)$. Then an input $\sigma$ is good if and only if $C(\sigma)$ accepts with probability at least $\delta/2$. Since $C$ is oracle-free, we can brute-force search for such a $\sigma$, finding it with probability at least $\delta/2$. The forgery will then be $(M,\sigma)$, which is accepted by the challenger with probability $\delta/2$, giving an overall advantage $\delta^2/4$.

For a potentially randomized $\mathsf{Ver1}$, we have to work slightly harder. For a good $\sigma$, we have that $\Pr_{v \leftarrow \mathsf{Ver0}^{\mathcal{O}}(\mathsf{PK},M)}[\Pr[\mathsf{Ver1}(v,\sigma) = 1] \geq \delta/4] \geq \delta/4$. Meanwhile, we will call a $\sigma$ "bad" if $\Pr_{v \leftarrow \mathsf{Ver0}^{\mathcal{O}}(\mathsf{PK},M)}[\Pr[\mathsf{Ver1}(v,\sigma) = 1] \geq \delta/4] \leq \delta/8$.

For a parameter $t$ chosen momentarily, we let $v_1, \dots, v_t \leftarrow \mathsf{Ver0}^{\mathcal{O}}(\mathsf{PK},M)$, and construct circuits $C_i(\sigma) = \mathsf{Ver1}(v_i,\sigma)$. We then brute-force search for a $\sigma$ such that $\Pr_{i \leftarrow [t]}[\Pr[C_i(\sigma) = 1] \geq \delta/4] \geq 3\delta/8$. By Hoeffding's inequality, any good $\sigma$ will be a solution with probability $1 - 2^{\Omega(\delta^2 t)}$. Meanwhile, any bad $\sigma$ will be a solution with probability $2^{-\Omega(\delta^2 t)}$. By setting $t$ such that $t/\delta^2$ is sufficiently longer than the bit-length of signatures, we can union bound over all bad $\sigma$, showing that there will be no bad solutions except with negligible probability. We will therefore find a not-bad solution with probability at least $\delta/2 - \mathsf{negl} \geq \delta/3$. In this case, with probability at least $\delta/8$ over the choice of $v$ by the verifier, $\Pr[\mathsf{Ver1}(v, \sigma) = 1] \geq \delta/4$. Hence, the overall success probability is at least $(\delta/3) \times (\delta/8) \times (\delta/4) \geq \delta^3/100$. ◻

**Lemma 7.5.** *If there is an IBE scheme in the RR generic group model, then for any constant $\delta$ there exists a restricted signature scheme in the same model.*

*Proof.* Let $(\mathsf{Gen}^{\mathbb{G}_{\mathrm{RR}}}, \mathsf{Extract}^{\mathbb{G}_{\mathrm{RR}}}, \mathsf{Enc}^{\mathbb{G}_{\mathrm{RR}}}, \mathsf{Ver}^{\mathbb{G}_{\mathrm{RR}}})$ be a supposed IBE scheme in the RR model. Now consider the following standard way of constructing a signature scheme from an IBE scheme:

- Key generation is simply $\mathsf{Gen}^{\mathbb{G}_{\mathrm{RR}}}$, with $\mathsf{PK} = \mathsf{mpk}$ and $\mathsf{SK} = \mathsf{msk}$.
- $\mathsf{Sign}^{\mathbb{G}_{\mathrm{RR}}}(\mathsf{SK}, M) = \mathsf{Extract}^{\mathbb{G}_{\mathrm{RR}}}(\mathsf{SK}, M)$, where $M$ is interpreted as an identity.
- $\mathsf{Ver}^{\mathbb{G}_{\mathrm{RR}}}(\mathsf{PK}, M, \sigma)$: Run $(c, k) \leftarrow \mathsf{Enc}^{\mathbb{G}_{\mathrm{RR}}}(\mathsf{PK}, M)$, where again $M$ is interpreted as an identity, and output 1 if and only if $\mathsf{Dec}^{\mathbb{G}_{\mathrm{RR}}}(\sigma, c) = k$.

Notice that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ almost already is restricted: $\mathsf{Ver}^{\mathbb{G}_{\mathrm{RR}}}(\mathsf{PK}, M, \sigma) = \mathsf{Ver1}^{\mathbb{G}_{\mathrm{RR}}}(\ \mathsf{Ver0}^{\mathbb{G}_{\mathrm{RR}}}(\mathsf{PK}, M)\ , \sigma)$ where $\mathsf{Ver0}^{\mathbb{G}_{\mathrm{RR}}}(\mathsf{PK}, M)$ outputs $v = (c, k) \leftarrow \mathsf{Enc}(\mathsf{PK}, \mathsf{id})$ while $\mathsf{Ver1}^{\mathbb{G}_{\mathrm{RR}}}(v, \sigma)$ checks that $\mathsf{Dec}^{\mathbb{G}_{\mathrm{RR}}}(\sigma, c) = k$.

The problem, of course, is that $\mathsf{Ver1}$ likely makes queries to $\mathbb{G}_{\mathrm{RR}}$, so does not have the required structure. We will therefore need to show how to compile out $\mathbb{G}_{\mathrm{RR}}$ from $\mathsf{Ver1}$, which we do in the Full Version [Zha22]. The idea is to provide $\mathsf{Ver1}$ some extra hints (through both $v$ and $\mathsf{PK}$) to help it answer the queries. First, any query made during $\mathsf{Ver0}$ is provided in $v$. Second, during setup, we choose many random messages, which we sign and verify, collecting all queries made during the verification process (both $\mathsf{Ver0}$ and $\mathsf{Ver1}$). We provide these queries in $\mathsf{PK}$. Then $\mathsf{Ver1}$ answers its queries by seeing if the query is in the span of the various queries it has available through $v$ and $\mathsf{PK}$. If so, it knows how to correctly answer the query. If not, it answers with a random label.

By standard arguments, we show that the only way $\mathsf{Ver1}$ can answer incorrectly is if a query corresponds to a "new" equation over labels seen during $\mathsf{Gen}$. By increasing the number of sign/verify trials during setup, we expand the span of queries provided in $\mathsf{PK}$. Since there are only a polynomial number of queries during $\mathsf{Gen}$, we can set the number of trials large enough to ensure any arbitrarily small inverse polynomial correctness error. ◻

Combining Lemmas 7.4 and 7.5, we therefore prove Theorem 7.2. ◻

## 8 On the Algebraic Group Model

Here, we discuss the Algebraic Group Model (AGM) of Fuchsbauer, Kiltz, and Loss [FKL18]. The AGM is proposed as a model lying between the standard and generic group models, striking a compromise between the wide applicability of generic groups and the security conferred by a standard-model proof. For these reasons, the AGM has become a popular model for proving security (e.g. [GRWZ20, BFL20, KLX20, BDFG20, GT21]).

In the AGM, adversaries can see the actual standard model group representation without any type-system constraints. They can therefore perform arbitrary standard-model computations on these elements. However, any time the adversary outputs a group element $h$, it must "explain" that element, by outputting a vector $\mathbf{a}$ such that $h = \prod_i g_i^{a_i}$, where $g_i$ are the input group elements. Because the adversary has unfettered access to the group representation, security cannot hold unconditionally; instead, security is proven by a reduction transforming an algebraic adversary into an algorithm for a hard problem, typically discrete log.

### 8.1 Allowed Games in the AGM

One wrinkle discussed in [FKL18] is that, without further constraints, the model is trivially invalid. Suppose the experiment provides the adversary a group element $h$, but implicitly encoded as a string $s$ by, say, by flipping every bit in the representation of $h$. Then the adversary can turn around and output $h$ (by flipping all the bits back), but it would have no way of producing a representation of $h$ without solving the discrete log of $h$. [FKL18] suggest the following resolution:

> We therefore demand that non-group-element inputs must not depend on group elements.

This demand, however, is never formalized. Fortunately, we can see that the Type Safe mode readily captures the desired intuition. After all, the TS model distinguishes between group and non-group elements, and the type safety guarantee means that once a group element is obtained, nothing can be done with it except for generating new group elements and equality gates. Since equality gates do not depend on the group element itself but just the exponent, no information about the group element can be extracted into bit wires.

We therefore propose that the AGM is restricted to TS model games. In the Full Version [Zha22], we offer a formal definition of the AGM.

### 8.2 AGM Un-instantiability

We now give an construction of a one-time MAC that is secure in the AGM, but insecure in the standard model, resolving an open question from [FKL18]. This result also gives an un-instantiability result for the TS model, which is simpler but somewhat more contrived. We note that our PKE scheme from Section 6 does *not* demonstrate anything about the AGM, since the adversary is not asked to produce any group elements. As such, for the PKE scheme, the AGM is actually *equivalent* to the standard model and hence the scheme is insecure in the AGM.

**Definition 8.1.** *A one-time message authentication code (MAC) is a triple of PPT algorithms* $(\mathsf{Gen}, \mathsf{MAC}, \mathsf{Ver})$ *where:*

- **Correctness:** $\forall m$, $\Pr[\mathsf{Ver}(k,m,\sigma)=1 : {}^{k\leftarrow\mathsf{Gen}()}_{\sigma\leftarrow\mathsf{MAC}(k,m)}] \geq 1-\mathsf{negl}$
- **Security:** *For any adversary* $\mathsf{A}$*, there exists a negligible* $\mathsf{negl}$ *such that* $\mathsf{A}$ *wins the following game with probability at most* $\mathsf{negl}$*: First* $\mathsf{A}$ *produces a message* $m$*; in response it receives* $\sigma \leftarrow \mathsf{MAC}(k,m)$ *for a random key* $k$*; finally it outputs* $m^* \neq m$ *and* $\sigma^*$*. It wins if* $\mathsf{Ver}(k,m^*,\sigma^*)=1$*.*

Note we require MACs for *unbounded-length* messages. Such one-time MACs can readily be built unconditionally. We now give our counter-example:

**Construction 8.2.** *Let* $(\mathsf{Gen}',\mathsf{MAC}',\mathsf{Ver}')$ *be an unconditional one-time MAC. We construct a new MAC* $(\mathsf{Gen},\mathsf{MAC},\mathsf{Ver})$ *using a group* $\mathbb{G}$*. We assume elements in* $\mathbb{G}$ *have bit-length* $\log_2 p$*; we can easily extend to general* $\mathbb{G}$*. We will assume* $\mathbb{G}$ *comes with two generators* $g,h$*, with the discrete log between them unknown*[4]*.*

- $\mathsf{Gen}()$*: run* $k' \leftarrow \mathsf{Gen}'()$ *and sample* $\gamma,\delta \leftarrow \mathbb{Z}_p$*. Output* $k=(k',\gamma,\delta)$*.*
- $\mathsf{MAC}(k,m)$*: first run* $\sigma' \leftarrow \mathsf{MAC}'(k',m)$*. Then interpret* $m$ *as a function* $H$ *whose output length is* $\log_2 p$ *bits. Compute* $u=H(\gamma,\delta)$*. Output* $\sigma=(\sigma',u,g)$*.*
- $\mathsf{Ver}(k,m,\sigma)$*: First run* $\mathsf{Ver}(k',m,\sigma')$*. If it accepts, also accept. Otherwise, write* $\sigma=(\sigma',u,w)$*. Check if* $w=g^\gamma h^\delta$*. If so, accept. Otherwise, reject.*

**Theorem 8.3.** *Construction 8.2 is a secure one-time MAC in the AGM under the discrete log assumption, but insecure under any instantiation of the group.*

*Proof.* We start with standard-model insecurity. We query on the $m$ which will be interpreted as the function $H(\gamma,\delta)=g^\gamma h^\delta$. In the resulting signature, $u$ therefore gives $g^\gamma h^\delta$. It can then sign any message with the signature $(*,*,u)$.

For AGM security, consider an adversary $\mathsf{A}$ breaking security in the AGM. Let $m^*$ be the message it forges, and $\sigma^*=(\sigma',u^*,w^*)$ be the forgery. By the one-time security of $(\mathsf{Gen}',\mathsf{MAC}',\mathsf{Ver}')$, we know that the only way for this signature to pass verification is for $w^*=g^\gamma h^\delta$. Since $\mathsf{A}$ is algebraic, and only previously received two group elements $g,h$ ($u$ was provided as bits), it must therefore explain $w^*$ by producing $\alpha,\beta$ such that $w^*=g^\alpha h^\beta$. There are two cases:

- $(\alpha,\beta)\neq(\gamma,\delta)$. Then we can use the adversary to solve the discrete log of $h$ relative to $g$, contradicting the assumed hardness of discrete logarithms.
- $(\alpha,\beta)=(\gamma,\delta)$. But $\gamma,\delta$ are random strings of total length $2\log_2 p$, and the adversary only gets at most $\log_2 p$ bits of information about them, namely the output of $H$. And yet the adversary is somehow able to recover all $2\log_2 p$ bits. This violates the incompressibility of random strings. $\square$

### 8.3 Is the AGM Superior to Generic Groups?

Since the AGM requires TS games, it is in some ways inferior to the RR model. From now on, we will therefore compare to the TS model. We now argue, however,

[4] $g,h$ *could* be created by $\mathsf{Gen}$, but we would need to get $g,h$ to the adversary before the first query. We could consider a 1-time signature, where $g,h$ would be included in the public key. Alternatively, we could consider a 2-time MAC, which includes $g,h$ as part of each MAC, giving the adversary $g,h$ in time for the second query.

that even though the AGM is “between” the standard and TS models, this does not necessarily demonstrate the AGM to be advantageous to generic groups. We consider two possible perspectives in which to compare the models.

*Attack-oriented perspective.* From an “attack-oriented” perspective, the AGM captures a wider class of attacks than generic groups. In this sense, the model offers a clear advantage when applied to the multiplicative groups over finite fields. The best attacks on such groups are index calculus attacks, which are captured by the AGM but not generic groups.

However, we note that some groups are not susceptible to index calculus attacks. Concretely, elliptic curves without efficient pairings are not, and this is exactly the reason why these curves are often conjectured to have optimal 128-bit security with groups of size $2^{256}$. In fact, essentially the only known attacks on elliptic curves are either generic, rely on a pairing, or the contrived counter-examples to generic groups such as [Den02]. For these groups, there are no known algebraic-but-non-generic adversaries, so it is not obvious that the AGM captures a wider class of adversaries. Thus while not worse than the TS generic group model, it seems that the AGM does not offer significant advantages for pairing-free elliptic curves for this perspective either.

*Security prediction perspective.* Another perspective is that a model is about making predictions about security. For any game $(p, \mathsf{Ch})$ and group $\mathbb{G}$, the standard, algebraic, and TS models will each make a decision about whether the game is hard. The standard model is the ground truth, but it may be infeasible to actually know if a game is secure or not in this model. The algebraic and TS models can be seen as predictions about this ground truth that are easier to reason about by giving more power to the prover, but they will have false-positives.

We now argue that existing work does not demonstrate any benefits of the AGM from this perspective. Concretely, looking at the AGM literature, we can break the known games into two cases:

- Those in which the AGM is trivially equivalent to the standard model. These are cases like public key encryption where the game does not ask for any group elements from the adversary and so the AGM imposes no restrictions over the standard model.
- Those in which the security holds in the AGM if and only if it also holds in the TS model. These include Construction 8.2 and all the positive results about the AGM, as well as trivially easy games in the TS model.

Thus, amongst known games, the AGM offers little predictive power for which games should be secure: in the first case we just stick with the standard model, and in the second case we can just stick with the TS model. So despite having fewer false positives, once we condition on the game, the known examples do not demonstrate any predictive advantages of the AGM over the existing models.

Note that this does not mean the AGM does not offer any predictive advantages, just that the current evidence does not support advantages from this perspective. We leave demonstrating such an advantage, or proving that one cannot exist, as an interesting question for future work.

## References


AY20. Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 13–43. Springer, Heidelberg, May 2020.

BBO07. Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, Heidelberg, August 2007.

BBP04. Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 171–188. Springer, Heidelberg, May 2004.

BCFG17. Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, August 2017.

BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014.

BDFG20. Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081, 2020. https://eprint.iacr.org/2020/081.

BFF$^+$14. Gilles Barthe, Edvard Fagerholm, Dario Fiore, John C. Mitchell, Andre Scedrov, and Benedikt Schmidt. Automated analysis of cryptographic assumptions in generic group models. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 95–112. Springer, Heidelberg, August 2014.

BFL20. Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Heidelberg, August 2020.

BFM15. Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random-oracle uninstantiability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 428–455. Springer, Heidelberg, March 2015.

BL22. Jeremiah Blocki and Seunghoon Lee. On the multi-user security of short schnorr signatures with preprocessing. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 614–643. Springer, Heidelberg, May / June 2022.

BM82. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd FOCS*, pages 112–117. IEEE Computer Society Press, November 1982.

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

CH20. Geoffroy Couteau and Dominik Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 768–798. Springer, Heidelberg, August 2020.

CK18. Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 415–447. Springer, Heidelberg, April / May 2018.

CL20. Alessandro Chiesa and Siqi Liu. On the impossibility of probabilistic proofs in relativized worlds. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 57:1–57:30. LIPIcs, January 2020.

Den02. Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 100–109. Springer, Heidelberg, December 2002.

DG17. Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017.

DHH+21. Nico Döttling, Dominik Hartmann, Dennis Hofheinz, Eike Kiltz, Sven Schäge, and Bogdan Ursu. On the impossibility of purely algebraic signatures. Cryptology ePrint Archive, Report 2021/738, 2021. https://eprint.iacr.org/2021/738.

Fis00. Marc Fischlin. A note on security proofs in the generic model. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 458–469. Springer, Heidelberg, December 2000.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

GK03. Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, October 2003.

GRWZ20. Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2007–2023. ACM Press, November 2020.

GT21. Ashrujit Ghoshal and Stefano Tessaro. Tight state-restoration soundness in the algebraic group model. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 64–93, Virtual Event, August 2021. Springer, Heidelberg.

IR89. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

JS08. Tibor Jager and Jörg Schwenk. On the equivalence of generic group models. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec 2008*, volume 5324 of *LNCS*, pages 200–209. Springer, Heidelberg, October / November 2008.

KLX20. Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 390–413. Springer, Heidelberg, November 2020.

KM06. Neal Koblitz and Alfred Menezes. Another look at generic groups. Cryptology ePrint Archive, Report 2006/230, 2006. https://eprint.iacr.org/2006/230.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

Mic94. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

MPZ20. Ueli Maurer, Christopher Portmann, and Jiamin Zhu. Unifying generic group models. Cryptology ePrint Archive, Report 2020/996, 2020. https://eprint.iacr.org/2020/996.

Nec94. V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

Nie02. Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Heidelberg, August 2002.

NR97. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.

PRV12. Periklis A. Papakonstantinou, Charles W. Rackoff, and Yevgeniy Vahlis. How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653, 2012. https://eprint.iacr.org/2012/653.

Rom90. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.

RSS20. Lior Rotem, Gil Segev, and Ido Shahaf. Generic-group delay functions require hidden-order groups. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 155–180. Springer, Heidelberg, May 2020.

SGS20. Gili Schul-Ganz and Gil Segev. Accumulators in (and beyond) generic groups: Non-trivial batch verification requires interaction. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 77–107. Springer, Heidelberg, November 2020.

SGS21. Gili Schul-Ganz and Gil Segev. Generic-group identity-based encryption: A tight impossibility result. In *Information Theoretic Cryptography*, 2021.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

Wat09. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.

Zha22. Mark Zhandry. To label, or not to label (in generic groups). Cryptology ePrint Archive, Report 2022/226, 2022. https://eprint.iacr.org/2022/226.

ZZ18. Mark Zhandry and Cong Zhang. Impossibility of order-revealing encryption in idealized models. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 129–158. Springer, Heidelberg, November 2018.

ZZ21. Mark Zhandry and Cong Zhang. The relationship between idealized models under computationally bounded adversaries. Cryptology ePrint Archive, Report 2021/240, 2021. https://ia.cr/2021/240.