A Review of Idealized Models

Chelsea Komlo

1 Introduction

In this note, we survey several idealized models and their comparison in the literature, summarizing observations made by Zhandry et al. [12,11].

A *generic group* is an idealized cryptographic group where group opeartions are carried out by making queries to oracles that simulate group operations, without revealing the elements of the underlying group directly. In other words, both adversaries and constructions do not make any use of the particular features of a group, and instead only perform legal group operations. What constitutes as a legal group operation is determined by the particular model.

Such generic groups follows similarities with the *Random Oracle Model*, as discussed next.

2 The Random Oracle Model (ROM)

In the ROM, cryptographic hash functions are treated as a uniformly random function that can only be accessed by making (interactive) oracle queries. The ROM is used as a mechanism to prove the security of cryptographic schemes that are otherwise unable to be proven secure using only standard assumptions.

It is well-known that security in the ROM does not imply standard-model security [1,8]. See [5] for an intuitive review of the above results.

3 Shoup's Generic Group Model

As discussed, a generic group model represents a cryptographic group as an idealized representation, where instead of interacting with group elements directly, all parties instead make interactive oracle queries. Shoup [10] introduced one varaint of a generic group model, which Zhandry [11] refers to this model as the "random representation model." Instead of interacting with group elements directly, all parties instead query labeling oracles and group operation oracles, defined as follows.

We now describe Shoup's model in more detail. Let $q \in \mathbb{Z}$ be a positive integer and let $\mathcal{S} \subset \{0,1\}^*$ be the set of strings of cardinality at least q, where some upper bound on $\mathcal{S}$ is assumed to be known.

- $\mathsf{Setup}()$: Perform the following:
  1. Initialize a table $T = \emptyset$.
  2. Define a random injective function $\mathcal{L}(\mathbb{Z}_q) \to \mathcal{S}$ to serve as a "labeling function."

- $\mathsf{Label}(x)$: Accept as input a value $x \in \mathbb{Z}_q$, and do the following:
  1. Derive $\ell \leftarrow \mathcal{L}(x)$
  2. If $T[\ell] = \perp$, set $T[x] = \ell$
  3. Output ℓ
- $\mathsf{GroupOp}(\ell_1, \ell_2, a_1, a_2)$: On input two labels (ℓ_1, ℓ_2) and two scalars $(a_1, a_2) \in \mathbb{Z}_q^2$, perform the following:
  1. If $T[\ell_1] = \perp$ or $T[\ell_2] = \perp$, return $\perp$.
  2. Otherwise, let $x_1 = T[\ell_1]$ and $x_2 = T[\ell_2]$.
  3. Derive $x_3 = a_1 x_1 + a_2 x_2$
  4. Derive $\ell_3 \leftarrow \mathcal{L}(x_3)$
  5. Set $T[x_3] = \ell_3$
  6. Return ℓ_3

Intuition on the Relation of Shoup's GGM to the ROM. Shoup's model implies the ROM, because Shoup's model can be used to perfectly simulate the random oracle model [12]. However, the converse is not true; the ROM does not imply Shoup's GGM. For example generic groups imply key agreement, but random oracles cannot be used for public public-key agreement [6].

4 Maurer's Generic Group Model

Similarly to in Shoup's model, Maurer's generic group model [7] supports labeling queries and group operation queries, but additionally defines an equality query. Also similarly to Shoup's model, each row in the table represents some value g^x with respect to some fixed generator g.

Let $q \in \mathbb{Z}$ be a positive integer.

- $\mathsf{Setup}()$: Perform the following:
  1. Initialize a table $T = \emptyset$.
- $\mathsf{Label}(x, i)$: Accept as input a value $x \in \mathbb{Z}_q$ and $i \in \mathbb{Z}$, and do the following:
  1. Set $T[i] = x$ (potentially overwriting prior entries)
- $\mathsf{GroupOp}(i_1, i_2, i_3, a_1, a_2)$: On input three integers $(i_1, i_2, i_3) \in \mathbb{Z}^3$ and two scalars $(a_1, a_2) \in \mathbb{Z}_q^2$, perform the following:
  1. If $T[i_1] = \perp$ or $T[i_2] = \perp$, return $\perp$.
  2. Otherwise, let $x_1 = T[i_1]$ and $x_2 = T[i_2]$.
  3. Derive $x_3 = a_1 x_1 + a_2 x_2$
  4. Set $T[i_3] = x_3$
- $\mathsf{IsEq}(i_1, i_2)$: On input two integers $(i_1, i_2) \in \mathbb{Z}^2$, perform the following:
  1. If $T[i_1] = \perp$ or $T[i_2] = \perp$, return $\perp$.
  2. Return 1 if $T[i_1] = T[i_2]$
  3. Otherwise, return 0

Challenges with Maurer's Model. As observed by Zhandry [11], potential issues with Maruer's model arise due to its requirement of keeping and overwriting state. More specifically, these issues can arise when different components of a cryptosystem access similar entries in the table, or even if the same component is run multiple times, potentially concurrently.

The same issues arise when proving security in Maurer's model. For example, an adversary might overwrite entries in the table, changing the underlying problem from one that is hard to one that is easy.

5 Zhandry's Type Safe Model

As a fix for the challenges with Maurer's model discussed above, Zhandry [11] introduces the Type Safe (TS) model, noting that the TS model has in fact already been used implicitly in many works claiming security in Maurer's model.

The distinction between the TS model and Maurer's model is that the TS model provides a *stateless* oracle when interacting with the underlying idealized group, whereas Maurer's model requires a stateful oracle. The TS model additionally does not hide the underlying group elements, but instead simply constrains the allowed legal operations that can be performed with respect to these elements. Hence, algorithms perform these computations directly, as opposed to outsourcing computations to an oracle.

Let $q \in \mathbb{Z}$ be a positive integer. The TS model defines algorithms as circuits, defined by wires and gates. The circuit supports two types of wires.

1. *Bit wires.* Denoted as BW; accepts a value b such that $b \in \{0, 1\}$.
2. *Element wires.* Denoted as EW; accepts a value a such that $a \in \mathbb{Z}_q \cup \{\perp\}$.

The TS model supports the following oracles:

1. $\mathsf{BitGate}(b_1, b_2) \to b_3$: Accepts two bit wires $(b_1, b_2) \in \mathsf{BW}^2$ and outputs a bit wire $b_3 \in \mathsf{BW}$.
2. $\mathsf{LabelGate}(\{b_i\}_{i \in [\log_2 q]}) \to e$: Accepts $\log_2 q$ bit wires. and encodes these as an element $x \in \mathbb{Z}_q$, outputting $\perp$ if the encoding fails. Otherwise, outputs an element wire $e \in \mathsf{EW}$.
3. $\mathsf{GroupOp}(\{b_{1i}\}_{i \in [\log_2 q]}, \{b_{2i}\}_{i \in [\log_2 q]}, e_1, e_2) \to e_3$: Accepts two sets of $\log_2 q$ bit wires, and encodes these as two elements $(a_1, a_2) \in \mathbb{Z}_q^2$, outputting $\perp$ if the encoding fails. Let (x_1, x_2) correspond to the element wires (e_1, e_2). Let $x_3 \in \mathbb{Z}_q$ be the value $x_3 = a_1 x_1 + a_2 x_2$, and let $e_3 \in \mathsf{EW}$ be the element wire corresponding to x_3. Output e_3.
4. $\mathsf{EqualityGate}(\{b_{1i}\}_{i \in [\log_2 q]}, \{b_{2i}\}_{i \in [\log_2 q]}) \to b$: Accepts two sets of $\log_2 q$ bit wires, and encodes these as two elements $(x_1, x_2) \in \mathbb{Z}_q^2$. If the encoding is successful and $x_1 = x_2$, outputs a bit wire $b \in \mathsf{BW}$ set to one. Otherwise, outputs a bit wire set to zero.

A game in the TS model allows both adversaries and challengers to interact with gates (modeled as oracles) as described above. Further, these parties can send bit and element wires to one another. We can think of element wires as containing $\log_2 q$ bits.

5.1 Constraints on the TS/Maurer Models

Many works in the literature demonstrate that common cryptosystems cannot be instantiated in the TS/Maurer model, including:

- Pseudorandom generators [11]
- Domain extensions for collision-resistant hash functions [11]
- Encryption with additive ciphertext size overhead [11]; any CPA-secure encryption scheme whose domain is bits in the TS model must have a ciphertext (comprised of group elements) of size at least the bit-length of the message.
- Delay functions [9]; such as time-lock puzzles and verifiable delay functions.
- Signatures [2]
- NIZKs in pairing-free groups [4]

So why use this model? It is an interesting indicator of what *could* be possible using purely algebraic functions. However, impossibility results in the TS/Maurer model need to be taken with a very large grain of salt.

6 Example of a Difference between Models

This example is again given by Zhandry [11].

The Blum-Micali PRG. Let us first introduce the Blum-Micali pseudo-random generator. Let $p \in \mathbb{Z}$ be a prime integer.

1. Pick a seed $x_0 \xleftarrow{\$} \mathbb{Z}_q$ and define a generator $g \in \mathbb{Z}_q$.
2. For $i \in \{1, \ldots, n\}$, define $x_i = g^{x_{i-1}}$.
3. Then, for $i \in \{1, \ldots, n\}$, output a hardcore bit b_i extracted from x_i.

The Blum-Micali PRG Can be Proven in Shoup's Model. Translating the Blum-Micali PRG to Shoup's model is straightforward. Instead of modeling $x_i \in \mathbb{Z}_q$, these values instead are labels $\ell_i \leftarrow \mathsf{Label}(x_i)$. However, each ℓ_i must be $\log_2 p$ bits so that the adversary gains no advantage in viewing the labels than interacting with each element directly.

The Blum-Micali PRG Cannot be Proven in the TS Model. Unfortunately, the Blum-Micali PRG cannot be modeled by the Type Safe Model. Because each $x_i \in \mathsf{EW}$ is an element wire, it cannot be given as input to $\mathsf{LabelGate}$ to derive a subsequent element wire (where $\mathsf{LabelGate}$ can be thought of outputting Y after performing the operation $Y = g^x$).

7 The Algebraic Group Model (AGM)

The Algebraic Group Model (AGM) was introduced by Fuchsbauer, Kiltz, and Loss in 2017 [3]. Intuitively, the AGM requires that any group element output by an adversary must be accompanied by a *representation* relative to an ordered

set of group elements that the adversary has seen previously. In other words, the adversary can see and interact with standard-model group elements directly, but must be able to "explain" any group element that it outputs as a linear combination of any group element it has interacted with previously.

Definition 1 (Algebraic Group Model (AGM) [3]). *An adversary is* **algebraic** *if for any group element $Z \in \mathbb{G}$ that it outputs, it is required to output a representation $\vec{a} = (a_0, a_1, a_2, \ldots)$ such that*

$$Z = g^{a_0} \prod Y_i^{a_i}$$

where $Y_1, Y_2, \ldots$ are group elements the adversary has seen thus far.

Proofs of security in the AGM must be proven by a reduction to a hard problem, as the adversary has unlimited access to the group representation and so security cannot hold unconditionally.

7.1 Required Constraints for Proofs in the AGM

As observed by Zhandry, unless the constructions that are proven in the AGM are constrained, it is trivially invalid, as illustrated by the following example [11].

Example 1. Let $\mathcal{A}$ be an algebraic adversary playing against some experiment $\mathcal{E}^{\mathcal{C}}$, where $\mathcal{E}$ is a security experiment and $\mathcal{C}$ is some cryptographic construction. Let $f(\mathbb{G}) \to \{0,1\}^*$ be an encoding function, and let $g(\{0,1\}^*) \to \{0,1\}^*$ be a function that accepts a bit string and flips each bit, outputting the bit-flipped string.

Suppose at some point during the experiment, $\mathcal{A}$ receives as input y_i', where $y_i' = g(f(Y_i))$. Then $\mathcal{A}$ can output Y_i, without knowing its representation.

As a counterexample Fuchsbauer et al. [3] suggest the following:

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

However, what does this requirement actually mean formally for constructions that are proven in the AGM? For example, does this mean that even hashing operations cannot depend on group elements?

As a means to formalize this requirement, Zhandry [11] proposes the use of the Type-Safe model, and observes that the Type-Safe model readily captures this intuition, the implications of which are summarized in Section 8.

7.2 AGM Un-Instantiability

Zhandry [11] gives the following example and corresponding proof of a one-time message-authentication code (MAC) that is secure in the AGM but completely insecure in the standard model. Recall that in a one-time MAC, each key k can be used to generate exactly one MAC output.

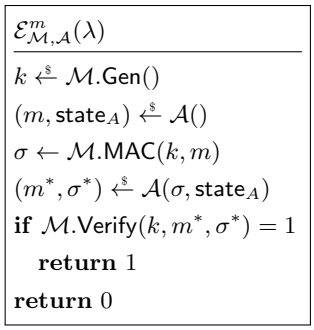

Fig. 1. The security game for a one-time message authentication code, where the key k can be used for exactly one message.

Definition 2. *A* **one-time message authentication code** *(MAC) is a triple of PPT algorithms* $\mathcal{M} = (\mathsf{Gen}, \mathsf{MAC}, \mathsf{Verify})$*, where:*

- *For* **correctness***, we require that*

$$\forall m \in \mathcal{M}, \Pr[\mathsf{Verify}(k, m, \sigma) = 1 \quad : \quad \begin{matrix} k \xleftarrow{\$} \mathsf{Gen}() \\ \sigma \leftarrow \mathsf{MAC}(k, m) \end{matrix}]$$

- *For* **security***, for any adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathcal{A}$ *wins the experiment in Figure 1 with probability at most* $\mathsf{negl}(\mathsf{negl})$*.*

A MAC that is Un-Instantiable in the AGM Let $\mathsf{MAC}' = (\mathsf{Gen}', \mathsf{MAC}', \mathsf{Verify}')$ be a secure one-time MAC. Then, let $\hat{\mathcal{M}} = (\hat{\mathsf{Gen}}, \hat{\mathsf{MAC}}, \hat{\mathsf{Verify}})$ be a MAC that is secure in the AGM but insecure in the standard model. $\hat{\mathcal{M}}$ is defined with respect to the parameters $(\mathbb{G}, g, h)$, where $\mathbb{G}$ is a group of prime order q, and g, h are both generators of the group such that the relation $h = g^{\alpha}$ is unknown. We then define $\hat{\mathcal{M}}$ as follows.

- $\hat{\mathcal{M}}.\mathsf{Gen}()$: Perform the following:
  1. $k \xleftarrow{\$} \mathcal{M}'.\mathsf{Gen}()$
  2. $(\gamma, \delta) \xleftarrow{\$} \mathbb{Z}_q$
  3. Output (k, γ, δ)
- $\hat{\mathcal{M}}.\mathsf{MAC}(k, m)$ On input k and m, do the following:
  1. $\sigma' \leftarrow \mathcal{M}'.\mathsf{MAC}(k', m)$
  2. Define m as the following function: $H(\mathbb{Z}_q, \mathbb{Z}_q) \rightarrow \{0,1\}^{\ell}$
  3. $u \leftarrow H(\gamma, \delta)$
  4. Output (σ', u, g)
- $\hat{\mathcal{M}}.\mathsf{Verify}(k, m, \sigma)$: On input k, message m, and σ, do:
  1. If $\mathcal{M}'.\mathsf{Verify}(k, m, \sigma) \rightarrow 1$: Output 1
  2. Otherwise, parse $(\sigma', u, w) \leftarrow \sigma$
  3. If $w = g^{\gamma} h^{\delta}$: Output 1

4. Otherwise, output 0

Theorem 1 (Zhandry [11]). *$\hat{\mathcal{M}}$ is a secure one-time MAC in the AGM under the DL assumption but insecure under any instantiation of a group.*

Proof Sketch. The scheme is clearly insecure in the standard model, as follows.

Standard Model Insecurity.

1. Pick m arbitrarily
2. Query $\hat{\mathcal{M}}.\mathsf{MAC}(m)$, receive (σ', u, g)
3. Pick σ^* arbitrarily
4. $\hat{\mathcal{M}}.\mathsf{Verify}(\sigma^*, g, u)$ will output 1.

However, the scheme can be proven secure in the AGM, as follows.

AGM Security.

1. Let $(m^*, \sigma^*(\sigma', m^*, w^*))$ be $\mathcal{A}$'s forgery.
2. Because $\mathcal{M}'$ is secure, then $w^* = g^\gamma h^\delta$.
3. Because $\mathcal{A}$ is algebraic, it must additionally output (α, β) such that

$$w^* = g^\alpha h^\beta = g^\gamma h^\delta$$

4. We can define the following two cases:
   (a) Case One: $(\alpha, \beta) \neq (\gamma, \delta)$: Then $\mathcal{A}$ can be used to solve for the discrete log of h with respect to g.
   (b) Case Two: $(\alpha, \beta) = (\gamma, \delta)$: The only mechanism $\mathcal{A}$ has to learn the values of (γ, δ) beyond random guessing is by ovserving the output of H, which outputs $\log_2 p$ bits. However, it is infeasible for $\mathcal{A}$ to recover (γ, δ), which requires recovering $2\log_2 p$ bits. This case then violates the incompressibility of random strings.
5. Hence, an adversary that breaks the security of $\hat{\mathcal{M}}$ in the AGM can be used as a subroutine to solve for the discrete logarithm of a challenge.

8 Relation Between Models

We now summarize results from [12,11] on the relation between security in the discussed models and implications for security in other models.

- Security in the random oracle model implies security in Shoup's model, but the converse is not true.
- Security in Shoup's model implies security in the TS model.
- For single-stage games, security in the TS model implies security in Shoup's model.
- for multi-stage games, security in the TS mode does not imply security in Shoup's model.
- The AGM allows for the same security as the TS model, and so inherits the same limitations as the TS model.

Zhandry [11] defines single-stage games as those which interact with a single adversary. Multi-stage games assume multiple adversaries which require some defined separation of state, capabilities, and means of communication with each other and the environment.

References

1. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In J. S. Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 209–218. ACM, 1998.
2. N. Döttling, D. Hartmann, D. Hofheinz, E. Kiltz, S. Schäge, and B. Ursu. On the impossibility of purely algebraic signatures. In K. Nissim and B. Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part III*, volume 13044 of *Lecture Notes in Computer Science*, pages 317–349. Springer, 2021.
3. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2018.
4. E. Giunta. On the impossibility of algebraic NIZK in pairing-free groups. In H. Handschuh and A. Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 702–730. Springer, 2023.
5. M. Green. What is the random oracle model and why should you care? (part 5), 2020. https://blog.cryptographyengineering.com/2020/01/05/what-is-the-random-oracle-model-and-why-should-you-care-part-5.
6. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In D. S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 44–61. ACM, 1989.
7. U. M. Maurer. Abstract models of computation in cryptography. In N. P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.
8. U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.
9. L. Rotem, G. Segev, and I. Shahaf. Generic-group delay functions require hidden-order groups. In A. Canteaut and Y. Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 155–180. Springer, 2020.

10. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.
11. M. Zhandry. To label, or not to label (in generic groups). In Y. Dodis and T. Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 66–96. Springer, 2022.
12. M. Zhandry and C. Zhang. The relationship between idealized models under computationally bounded adversaries. *IACR Cryptol. ePrint Arch.*, page 240, 2021.