The Algebraic Group Model

Thomas Souliotis

NTUA

June 8, 2023

Overview

- Presentation of the Generic Group Model (GGM) and the Standard Model.
- Presentation and brief analysis of the *The algebraic group model and its applications.* proposed by **Fuchsbauer, G., Kiltz, E., Loss, J.** (2018) ([FKL18]).
- Beyond AGM and potential issues posed by **Katz, J., Zhang, C., Zhou, H. S.** (2022) in their paper: *An analysis of the algebraic group model* ([KZZ22]).

The Standard Model

- Standard model is the model of computation in which the adversary is only limited by the amount of time and computational power available.
- Schemes that can be proven secure using only complexity assumptions (e.g. factorization) are said to be secure in the standard model.
- **Mathematical Abstractions**: The standard model employs mathematical abstractions and idealized assumptions to reason about the security properties of cryptographic protocols.
- **Rigorous Proof Techniques**: It enables rigorous proofs of security properties by formulating precise definitions, security notions, and reductions.

The Generic Group Model (GGM)

- GGM is an idealised cryptographic model, where the adversary is only given access to a randomly chosen encoding of a group, instead of efficient encodings, such as those used by the finite field or elliptic curve groups used in practice.
- GGM includes an oracle that executes the group operation.
- GGM mainly used to analyse computational hardness assumptions.

Idealized Models

- Ideally proof in Standard Model (no simplifications)
- Idealizing via abstraction -> prove in the simplified/idealized model
- Example: ROM $\xrightarrow{\text{idealizes}}$ hash functions, GGM $\xrightarrow{\text{idealizes}}$ cyclic groups

Generic Group Algorithms

Let $\mathbb{G} = < G, \circ, g >$. A is *generic* if it only computes over $\mathbb{G}$ as follows:

- Given $a, b \in \mathbb{G}$ compute $c = a \circ b$.
- Given $a, b \in \mathbb{G}$ check whether $a = b$.

Generic Group Algorithms: Pros

- Work in every cyclic group.
- Information theoretic lower bounds (DLP, CDH, DDH, etc).
- Fitting abstraction for (some) elliptic curves.

Generic Group Algorithms: Cons

- Representaton-based exploits (e.g. Jacobi symbols, index calculus-based attacks, etc).
- Deriving lower bounds is a difficult process (combinatorial arguments).
- Lower bounds are not modular -> new boundaries for a new cryptographic protocol.
- Many algorithms of practical interest are not generic.

Introduction to the Algebraic Group Model

- Weaker model assumptions than GGM.
- $GGM \leq AGM \leq$ Standard Model.
- Reduction based and easy to work with (allows easy proofs).
- Improved abstraction of reality over GGM.
- Results from AGM carry over to GGM.
- Captures a broad spectrum of important algorithms.

Algebraic Algorithms

An algorithm A_{ALG} is algebraic if it fulfills the following:

- Given a list of all group elements $L = (L_1, ..., L_t)$ given to A_{ALG} during its execution
- Whenever A_{ALG} outputs a group element $Z \in \mathbb{G}$, it also outputs a representation $z = (z_1, ..., z_t)$ s.t. $Z = \prod_i L_i^{z_i}$

Basically an Algebraic Algorithm always tells us how it computes new group elements.

AGM in a Nutshell

- All algorithms are modeled as algebraic, i.e., also adversaries in security experiments.
- This gives strictly weaker model assumptions than GGM
- The above is derived from Lemma where every generic algorithm is also an algebraic algorithm.

AGM: Composistion Theorem

If $true \xRightarrow{\text{GGM}} S$ (lower bound for S in the GGM), and $S \xRightarrow{\text{AGM}} T$, then $true \xRightarrow{\text{GGM}} T$, if reduction in AGM is a generic algorithm.

AGM vs GGM proofs

- **GGM**: Lower bounds for algorithms via combinatorial arguments.
- **AGM**: Reduction based proofs.

Reductions in the AGM

- DLP $\xrightarrow{\text{AGM}}$ CDH assumption
- DLP $\xrightarrow{\text{AGM}}$ SDH assumption
- DLP $\xrightarrow{\text{AGM}}$ LRSW assumption
- DDH $\xrightarrow{\text{AGM}}$ ElGamal CCA1
- DLP $\xrightarrow{\text{AGM}}$ Groth's ZK-SNARK
- DLP $\xrightarrow{\text{AGM}}$ BLS signature scheme

AGM Reduction: Example

- CDH: Given g, g^x, g^y, compute g^{xy}
- DLP: Given g^u, compute u.
- Proof that DLP $\xRightarrow{\text{AGM}}$ CDH assumption (breaking CDH algebraically is as hard as solving DLP).

AGM Reduction: Example

- Challenger $U = g^u$ wants to compute u
- Adversary sovles the CDH Problem
- Challenger sends g, g^x, g^y
- Adversary responds with $g^{xy}, z = (a, b, c)$. However, from definition of Algebraic Algorithms: $g^{xy} = (g^x)^a (g^y)^b g^c$. That happens because $L = (L_1, ..., L_t) = (g, g^x, g^y)$ and $Z = g^{xy}$ with representation $z = (z_1, ..., z_t) = (a, b, c)$
- $g^{xy} = (g^x)^a (g^y)^b g^c \overset{x}{\Rightarrow} y = xa + yb + c \ (modp)$. We solve the above for x ($x = \frac{yb+c}{y-a}$) unless $y = a \ (modp)$. Problem is resolved by randomly choosing $g^u = g^x$ or $g^u = g^y$. Problem is always solved with probability $\frac{1}{2}$

AGM Problems

- [KZZ22] poses some potential issues on the AGM.
- They highlight that as the GGM and AGM are currently formalized, this is not true: hardness in the AGM may not imply hardness in the GGM, and a generic reduction in the AGM may not imply a similar reduction in the GGM.
- [KZZ22] focuses on the definition given to the algebraic algorithms by [FKL18] and states that in general algebraic algorithms depend on a specific encoding σ (An encoding $\sigma : Z_p \to \{0,1\}^\ell$ is simply an injective map from Z_p to $\{0,1\}^\ell$).
- [FKL18] poses no restriction on intermediate computations but require that any group elements output by an algorithm must be accompanied by a representation relative to the input ordered set.

AGM Problems (continued)

- The AGM is useless for analyzing games where the algorithm's output is not a group element.
- Whenever the encoding is such that the discrete-logarithm problem can be solved efficiently relative to that encoding, any algorithm can be made algebraic by simply computing a representation of any group elements it outputs e.g.
  1. $r_1, r_2 \leftarrow Z_p$
  2. $s \leftarrow r_1 r_2 \; mod \; p$
  3. Output (s, s)
- Once a particular encoding σ is fixed, it is not immediately well-defined what it means for an algorithm to "be provided with a group element as input" or to "output a group element." E.g a game where given input i it returns the i-th bit of $\sigma(x)$ -> no group element as input but can construct $\sigma(x)$ ([FKL18] tries to mitigate this by enforcing other elements to not depend on any group elements).

Counterexample Via beg Algorithm

Given two games G and H such that: (1) there is a Shoup-generic reduction from H to G; (2) H is hard for Shoup-generic algorithms; but (3) G is easy for Shoup-generic algorithms

- H is the $dlog$ game:
  1. $z \leftarrow Z_p$
  2. $z' \leftarrow A(\sigma(1), \sigma(z))$
  3. Return 1 iff $z' = z$
- G is the beg game:
  1. $z \leftarrow Z_p$
  2. parse $Z = \sigma(z)$ as the bitstring $z_1 ... z_l$
  3. $(X, U_1, ..., U_l) = (\sigma(1), \sigma(z_1), ..., \sigma(z_l))$
  4. $Z' \leftarrow A(X, U_1, ..., U_l)$
  5. Return 1 iff $(Z' = Z)$
- There is a Shoup-generic algorithm A with for G (for each i, A sets $z'_i = 1$ iff $U_i = X$)

Counterexample Via beg Algorithm (continued)

- Then there is a proof by reduction between $dlog, beg$
- Key point of that proof is that the generic algorithm A used cannot be converted to an Algebraic algorithm!
- Proof is the following

Fix an encoding σ. Generic reduction R is given $(\mathbf{X}, \mathbf{Z}) := (\sigma(1), \overset{\text{alg}}{\sigma}(z))$ as input along with oracle access to an algebraic algorithm A; it proceeds as follows:

1. Parse $\mathbf{Z}$ as the bitstring $z_1 \cdots z_\ell$. Set $z_0 := 1$.
2. Request $\mathbf{I} = \sigma(0)$ from the labeling oracle.
3. For $i = 1, \ldots, \ell$ do: if $z_i = 0$ then set $\mathbf{U}_i := \mathbf{I}$; else set $\mathbf{U}_i := \mathbf{X}$.
4. Run $\mathsf{A}(\mathbf{X}, \mathbf{U}_1, \ldots, \mathbf{U}_\ell)$ to obtain output $\mathbf{Z}'$ along with a representation $(x_0, x_1, \ldots, x_\ell)$ such that $\mathbf{Z}' = \mathbf{X}^{x_0} \cdot \mathbf{U}_1^{x_1} \cdots \mathbf{U}_\ell^{x_\ell}$.
5. Output $\sum_{i=0}^{\ell} z_i \cdot x_i \bmod p$.

We now analyze the behavior of R. Let A be an algebraic adversary with $\epsilon = \mathbf{Succ}^{\mathsf{A}}_{\mathbf{beg}_\sigma}$. Observe that when A is run as a subroutine by R in game $\mathbf{dlog}_\sigma$, the input provided to A is distributed identically as in $\mathbf{beg}_\sigma$. Moreover, whenever A succeeds it holds that (1) $\mathbf{Z}' = \mathbf{Z}$ and (2) $z = \sum z_i \cdot x_i \bmod p$. It follows that $\mathbf{Succ}^{\mathsf{R}^{\mathsf{A}}}_{\mathbf{dlog}_\sigma} = \epsilon$. This completes the proof. □

Conclusion

- [KZZ22] construct a clever way to use Algebraic Algorithms and their definition so as to construct reduction proofs.
- On the other hand [KZZ22] state that this definition is not concrete and provide a counterexample.
- Both [FKL18] and [KZZ22] try to formalize their proofs using their systems and assumptions.
- Maybe there are other solutions like Zhandry's in [Zha22] where a new definition for AGM is provided.