The Structured Generic-Group Model

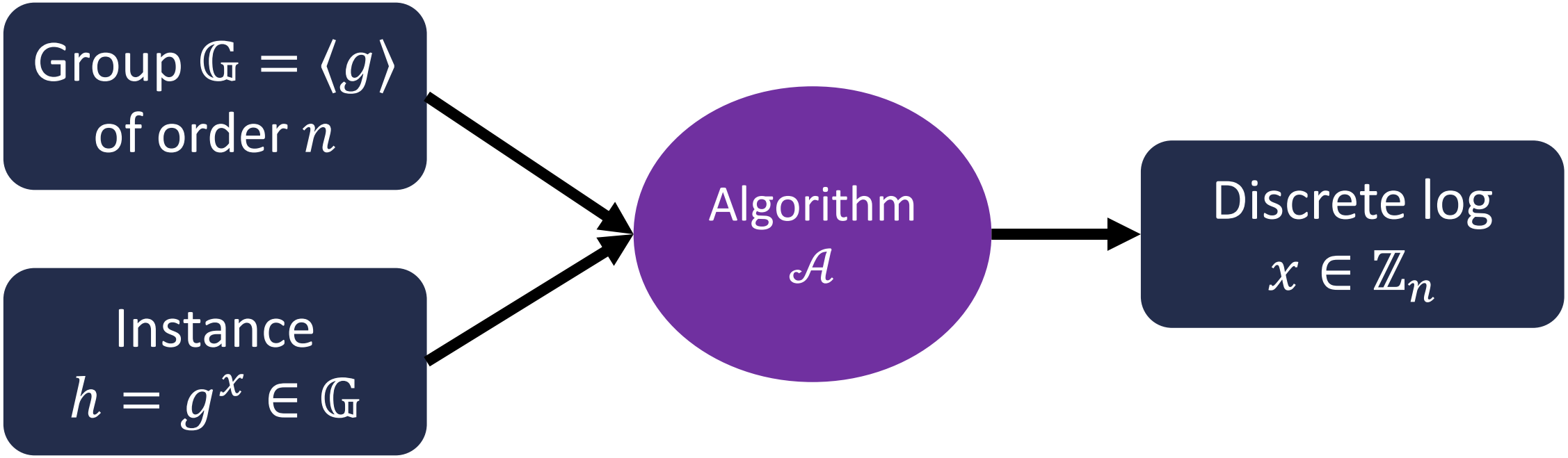
Henry Corrigan-Gibbs, Alexandra Henzinger, and **David Wu**

May 2026

Many slides were adapted from slides by Alexandra

The Discrete Log Problem

[DH76]



New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

Cornerstone of modern cryptography

Is the Discrete Log Problem Hard?

Proving unconditional lower bounds is very difficult…

But… not if we impose some conditions

The generic group model studies restricted classes of discrete log attacks

A "generic" algorithm only has black-box access to the group operation

By restricting the algorithm, we can prove information-theoretic lower bounds

Theorem [Nec94, Sho97, Mau05]
In a group of prime order n, any generic discrete-log algorithm with constant success probability (on a uniformly-random challenge) must run in time $\Omega(\sqrt{n})$.

$\Longrightarrow$ Every generic discrete log algorithm runs in **exponential** time.

The Generic Group Model (GGM)

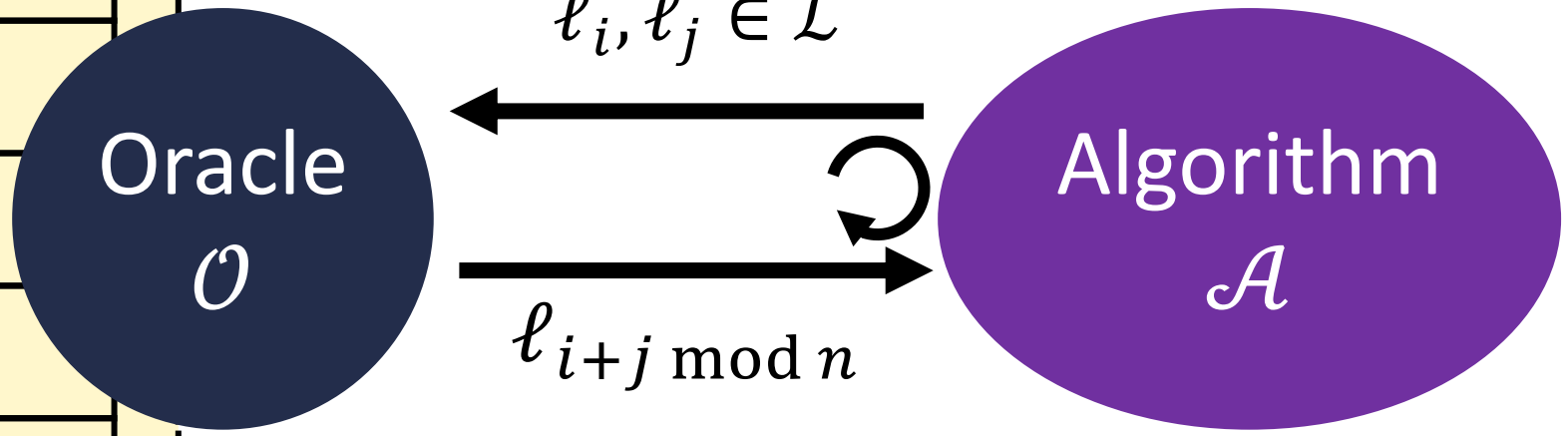
[Sho97]

discrete log

log	label
0	$\ell_0 = \sigma(0)$
1	$\ell_1 = \sigma(1)$
2	$\ell_2 = \sigma(2)$
$\vdots$	$\vdots$
$n-1$	$\ell_{n-1} = \sigma(n-1)$

$\sigma: \mathbb{Z}_n \to \mathcal{L}$

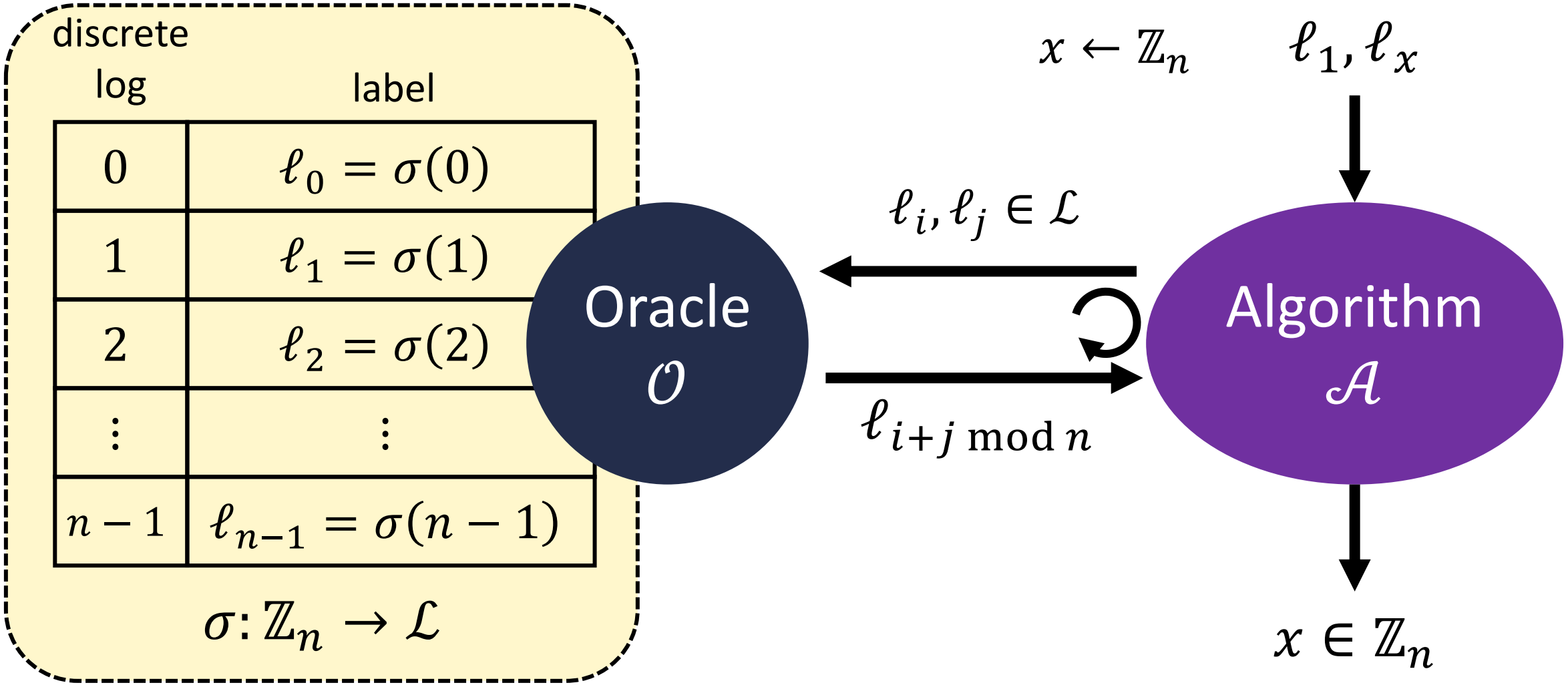
initialized with **random** injective mapping σ



All access to the group operation is only through the generic group oracle

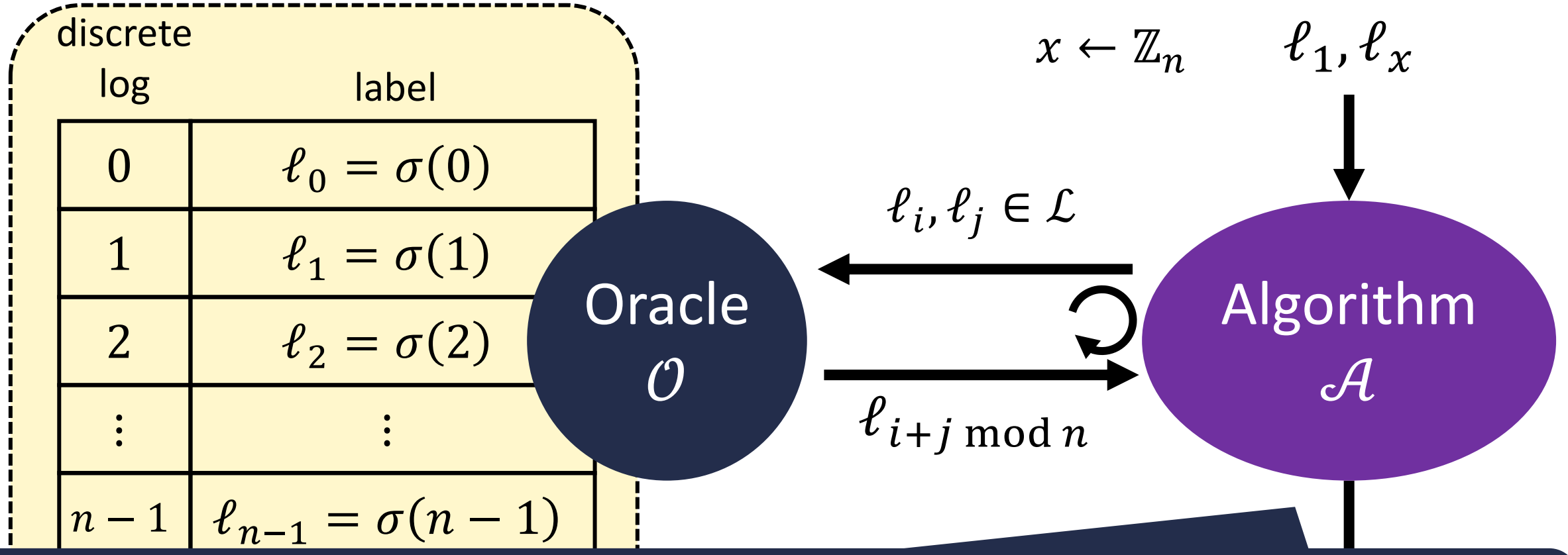
The Discrete Log Problem in the Generic Group Model

[Sho97]



The Discrete Log Problem in the Generic Group Model

[Sho97]



Lower bound: to find discrete log, algorithm must make $\Omega(\sqrt{n})$ oracle queries

Tight: Matches baby-step/giant-step, Pollard's rho, etc.

Attacks on Discrete Log

Generic group model captures generic attacks that cannot take advantage of representation of group elements

But… we have examples of non-generic algorithms over **specific** groups of interest

Over finite fields $\mathbb{F}_n^*$: sub-exponential time [Adl79, Pom87, Gor93, …]

Over small-characteristic fields $\mathbb{F}_{2^n}^*$: quasi-polynomial time [BGJT14]

In these settings, index calculus methods are **much** better than generic algorithms

Important setting: target group in pairing-based cryptography is a finite field $\mathbb{F}_{p^k}^*$

Implication: generic bilinear group model does not accurately capture "best attacks"

This Work: The Structured Generic-Group Model

First model that is

1. Expressive enough to capture sub-exponential-time index calculus attacks
2. Generic enough to prove matching, information-theoretic lower bounds

$\Longrightarrow$ Broad class of index-calculus algorithms are optimal among algorithms exploiting a specific type of group structure.

This Work: The Structured Generic-Group Model

First model that is

1. Expressive enough to capture sub-exponential-time index calculus attacks
2. Generic enough to prove matching, information-theoretic lower bounds

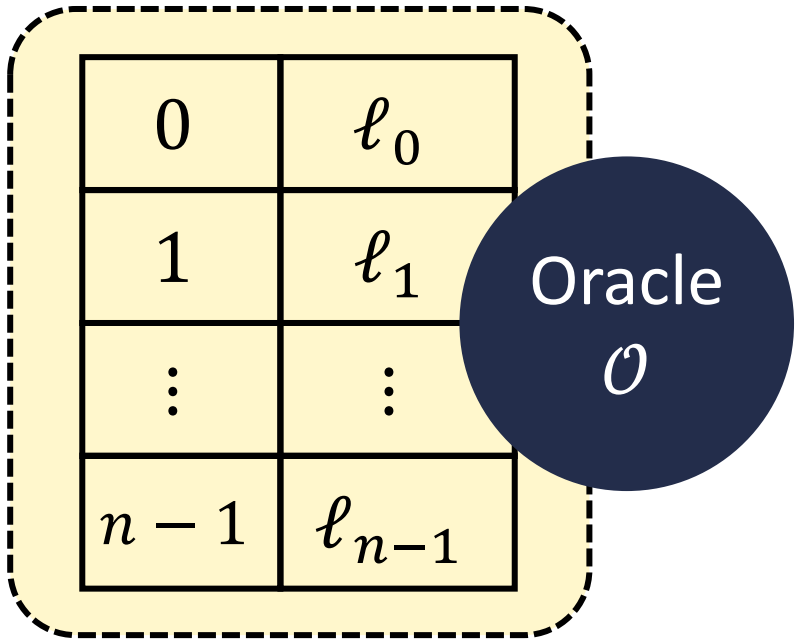
In contrast to other models:

- Algebraic group model [FKL18]
- Generic ring model [AM09]
- Smooth generic group model [Hha25]
- Preprocessing model, bit-fixing model [Unr07, CDG18, CK18, CDGS18]

The Structured Generic-Group Model

Goal: reason about algorithms that exploit some non-generic-group structure

Idea: give algorithm free access to some group-operation outputs



labels will carry some structure

$\star$	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_0	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_1	ℓ_1	ℓ_2	ℓ_3		
ℓ_2	ℓ_2	ℓ_3		ℓ_8	
$\vdots$	$\vdots$		ℓ_8		
ℓ_{n-1}	ℓ_{n-1}				

structure determined by a **partial function** $\star$

The Structured Generic-Group Model

Goal: reason about algorithms that exploit some non-generic-group structure

Idea: give algorithm free access to some group-operation outputs

Assumptions on $\star$:

- Binary operation on label set
- Partial function (output could be undefined)
- Commutative operation
- Induces unique factorization of labels
  - Label ℓ is "prime" if ℓ is not the identity label and there does not exist $\ell_1, \ell_2 \neq \ell$ where $\ell_1 \star \ell_2 = \ell$
  - Each label can be **uniquely** decomposed into a product of primes

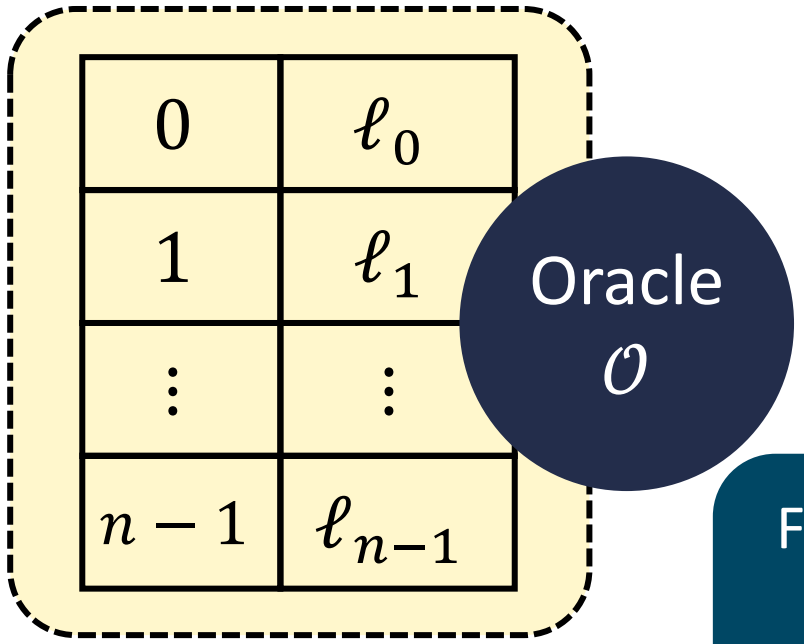
$\star$	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_0	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_1	ℓ_1	ℓ_2	ℓ_3		
ℓ_2	ℓ_2	ℓ_3		ℓ_8	
$\vdots$	$\vdots$		ℓ_8		
ℓ_{n-1}	ℓ_{n-1}				

structure determined by a **partial function** $\star$

The Structured Generic-Group Model

Goal: reason about algorithms that exploit some non-generic-group structure

Idea: give algorithm free access to some group-operation outputs



labels will carry some structure

$\star$	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_0	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_1	ℓ_1	ℓ_2	ℓ_3		
ℓ_2	ℓ_2	ℓ_3		ℓ_8	
			ℓ_8		

For every choice of oracle, it will always be the case that $\mathcal{O}(\ell_0, \ell_2) = \ell_2$.

Such labelings are **"consistent"** with $\star$

e determined by

rtial function $\star$

The Structured Generic-Group Model

Goal: reason about algorithms that exploit some non-generic-group structure

Idea: give algorithm free access to some group-operation outputs

⋆-operator is a parameter of the model – determines how much structure is present within the encoding

$n-1$	ℓ_{n-1}

labels will carry some structure

⋆	ℓ_0	ℓ_1	ℓ_2	⋯	ℓ_{n-1}
ℓ_0	ℓ_0	ℓ_1	ℓ_2	⋯	ℓ_{n-1}
ℓ_1	ℓ_1	ℓ_2	ℓ_3		
ℓ_2	ℓ_2	ℓ_3		ℓ_8	
			ℓ_8		

For every choice of oracle, it will always be the case that $\mathcal{O}(\ell_0, \ell_2) = \ell_2$.

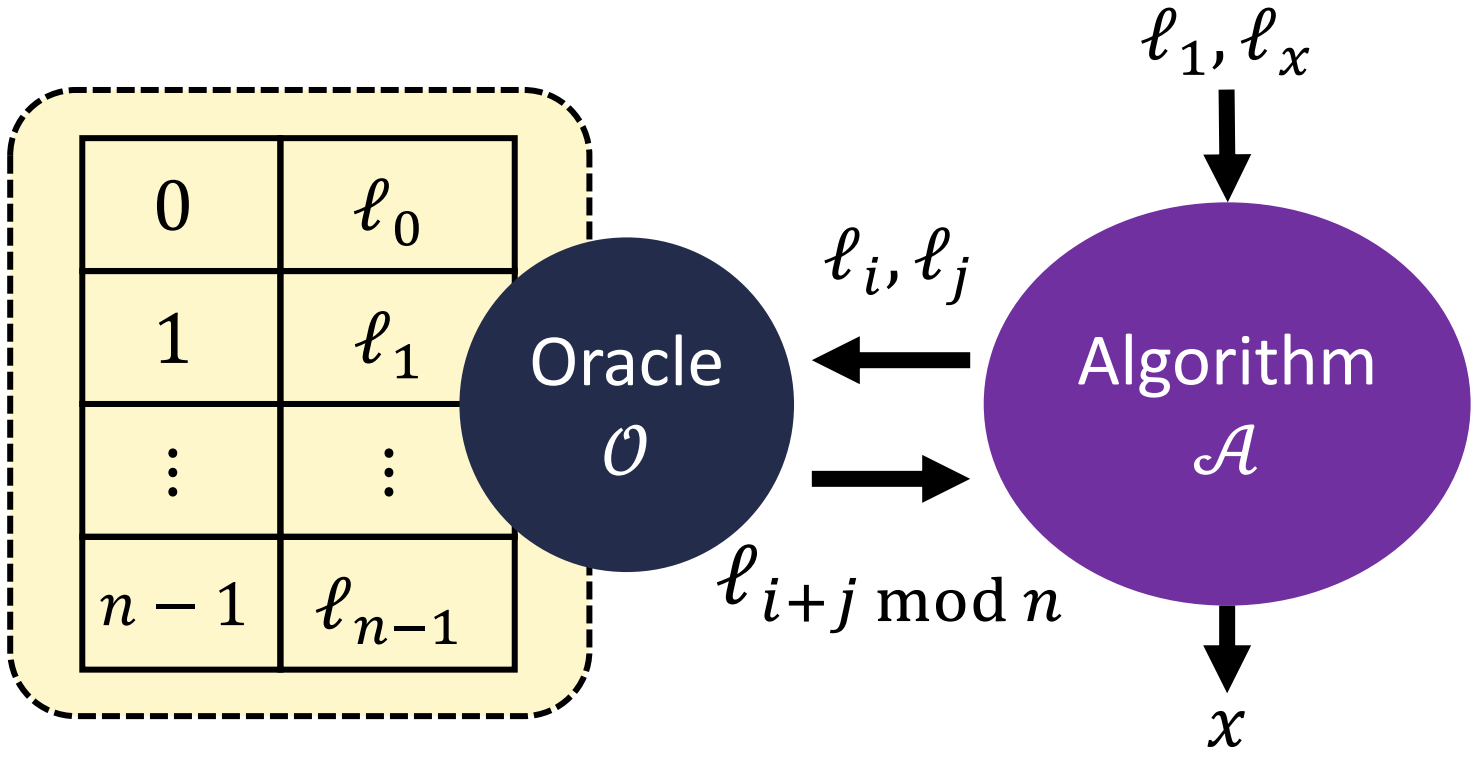
Such labelings are **"consistent"** with ⋆

e determined by

rtial function ⋆

Discrete Log in the Structured GGM

The discrete logarithm problem is hard in the structured GGM if there exists **some** distribution over labels that are consistent with $\star$ for which no fast algorithms exist



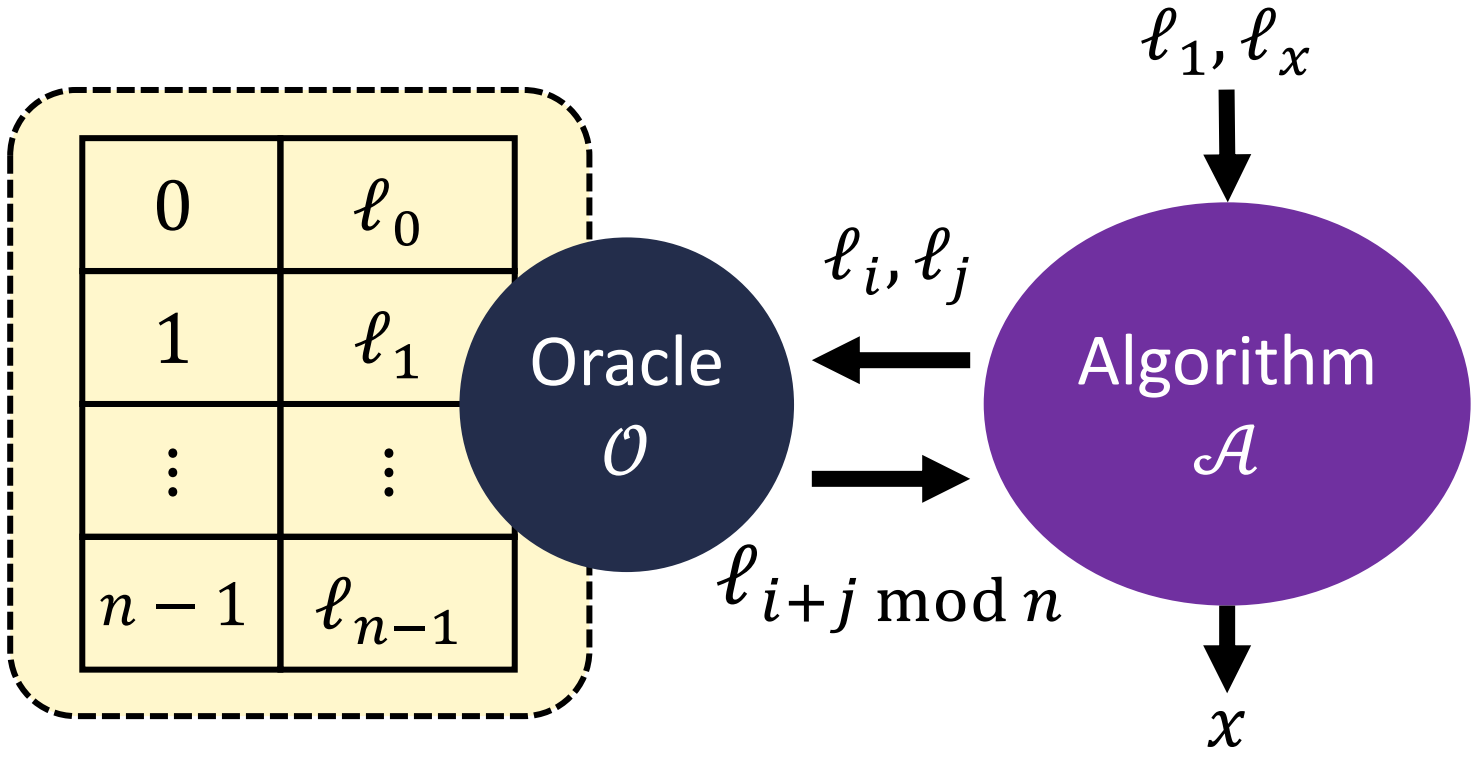
labels will carry some structure

$\star$	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_0	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_1	ℓ_1	ℓ_2	ℓ_3		
ℓ_2	ℓ_2	ℓ_3		ℓ_8	
$\vdots$	$\vdots$		ℓ_8		
ℓ_{n-1}	ℓ_{n-1}				

structure determined by a **partial function** $\star$

Discrete Log in the Structured GGM

The discrete logarithm problem is hard in the structured GGM if there exists **some** distribution over labels that are consistent with $\star$ for which no fast algorithms exist

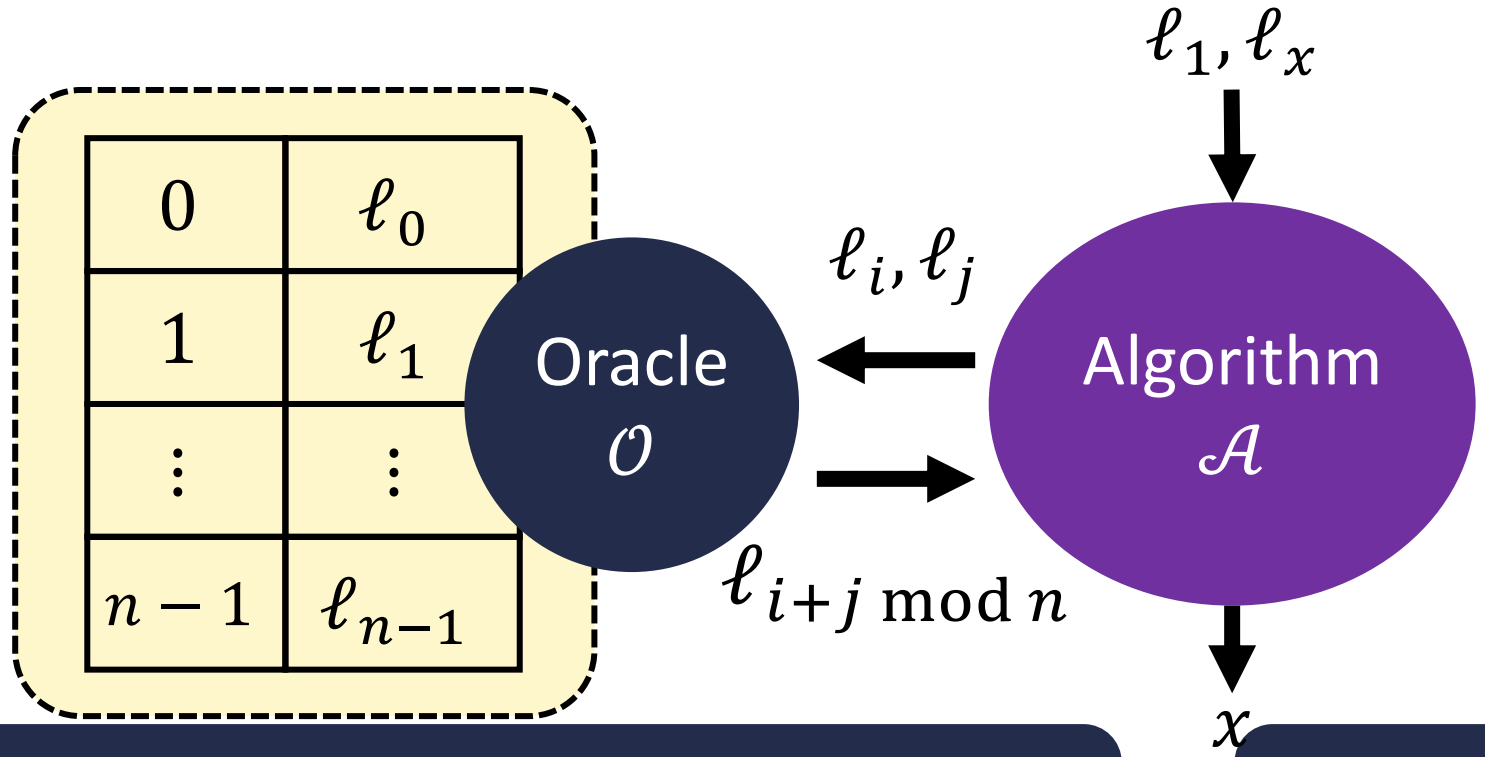


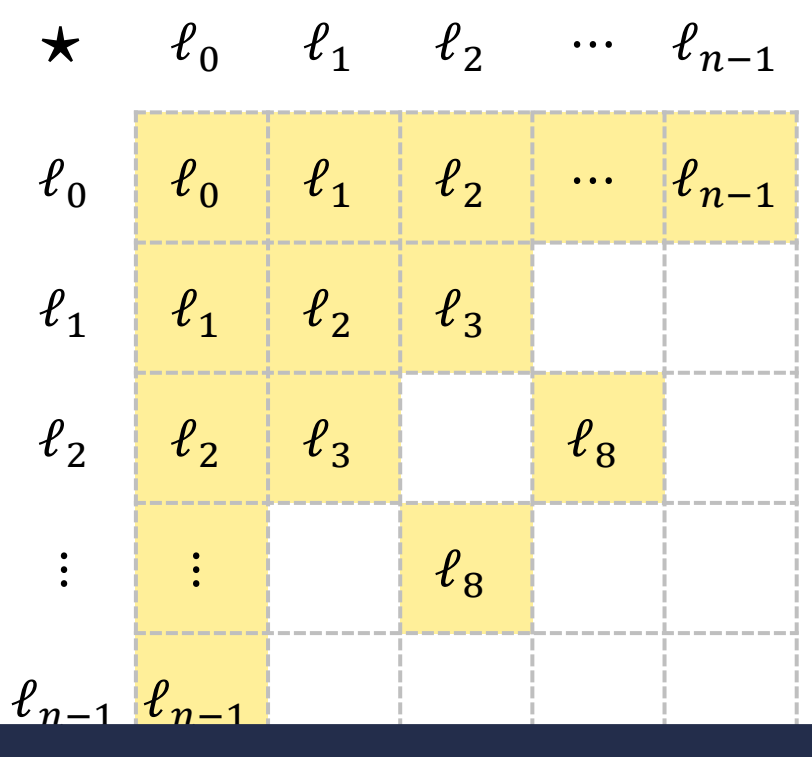
$\star$	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_0	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_1	ℓ_1	ℓ_2	ℓ_3		
ℓ_2	ℓ_2	ℓ_3		ℓ_8	
$\vdots$	$\vdots$		ℓ_8		
ℓ_{n-1}	ℓ_{n-1}				

In the structured GGM, we measure adversary's running time by **number of oracle queries**. Information in $\star$ is **free**.

Discrete Log in the Structured GGM

The discrete logarithm problem is hard in the structured GGM if there exists **some** distribution over labels that are consistent with $\star$ for which no fast algorithms exist





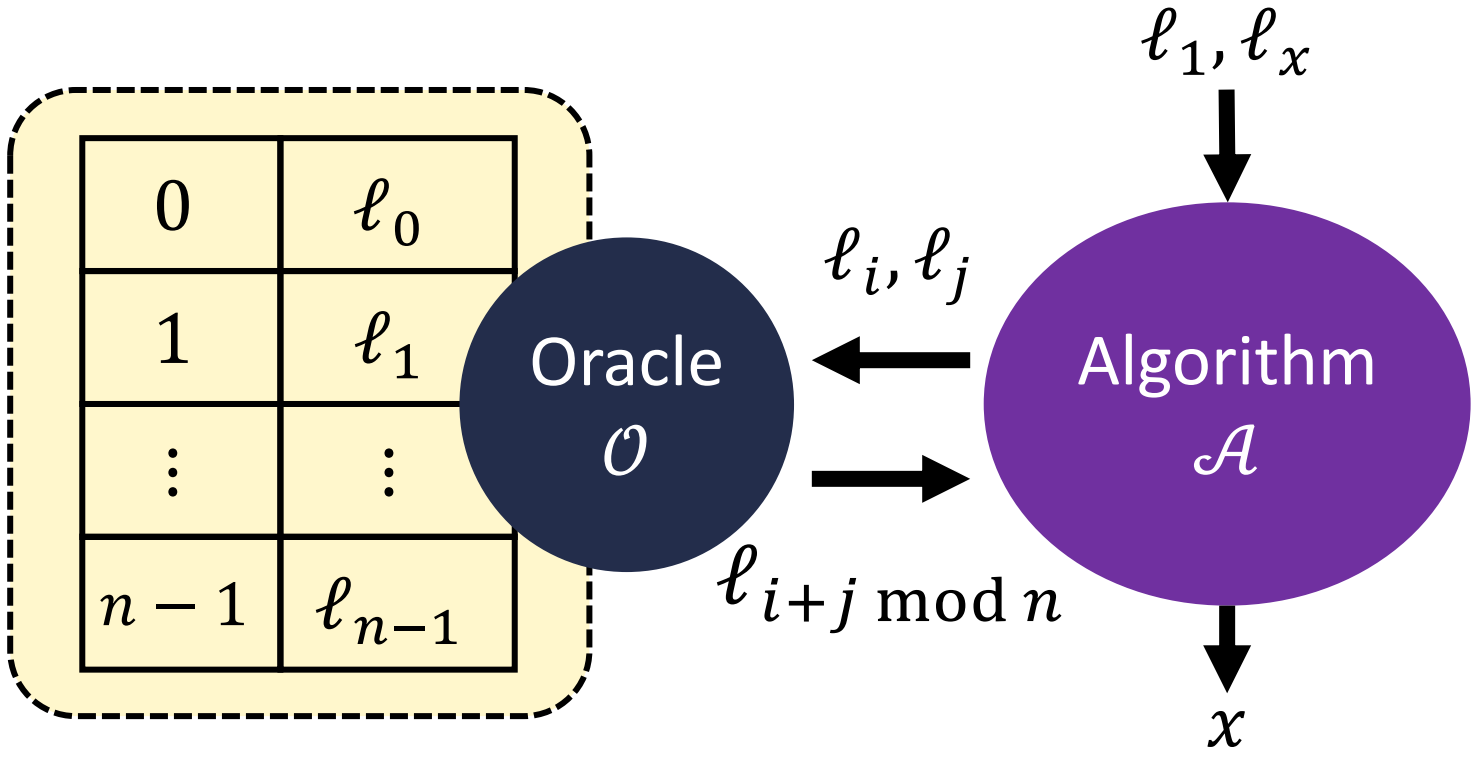
Unlike preprocessing model [CK18, CDG21]: adversary only sees local information on discrete logs

Unlike bit-fixing model [Unr07, CDGS18, CDG21]: adversary does not pick which labels are revealed

Discrete Log in the Structured GGM

Hardness in the structured GGM depends on how $\star$ is defined

Interpolates between Shoup's generic group and a concrete group

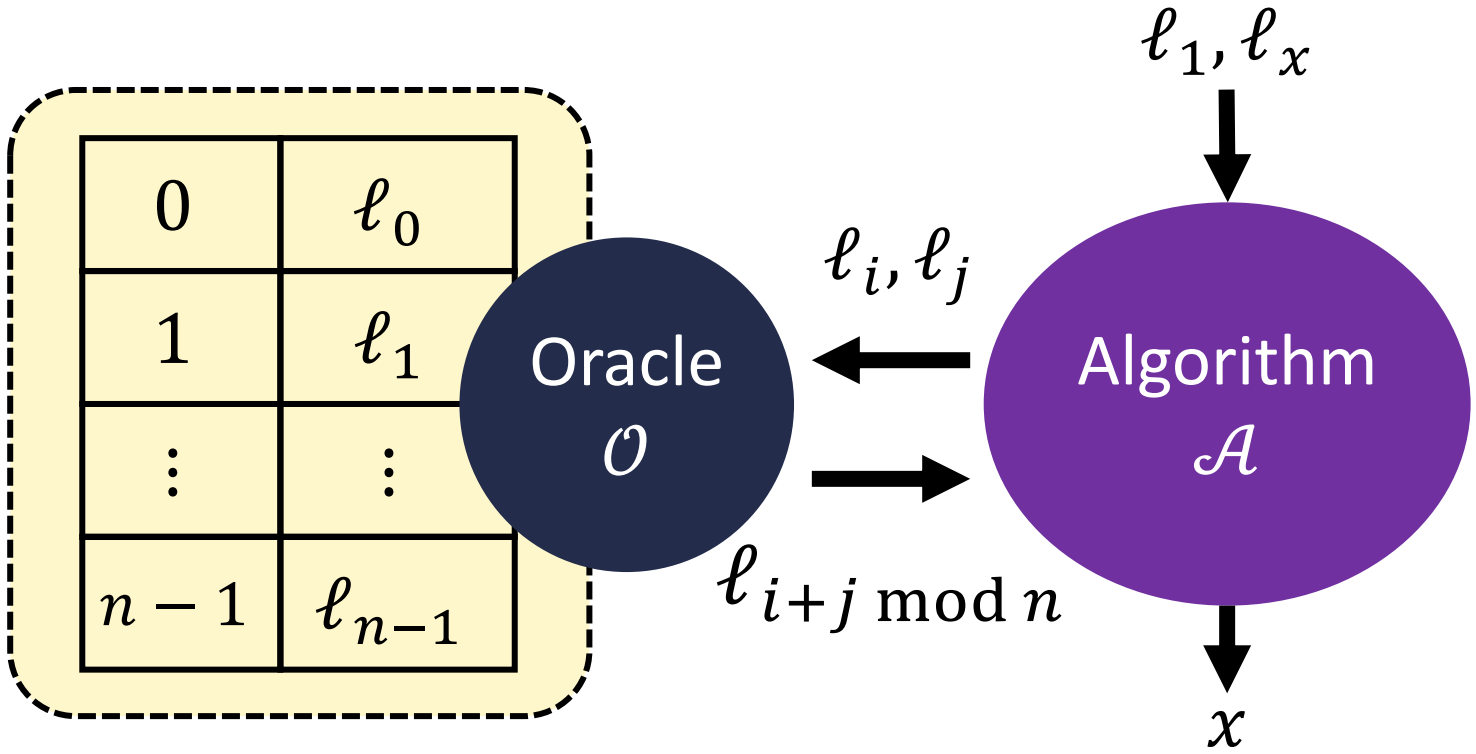


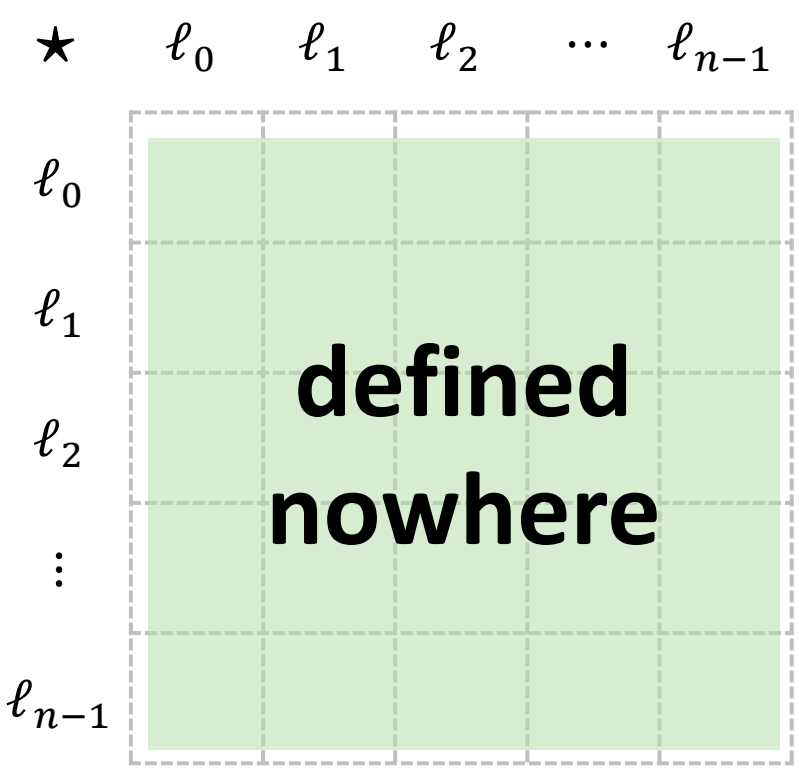
$\star$	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_0	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_1	ℓ_1	ℓ_2	ℓ_3		
ℓ_2	ℓ_2	ℓ_3		ℓ_8	
$\vdots$	$\vdots$		ℓ_8		
ℓ_{n-1}	ℓ_{n-1}				

Discrete Log in the Structured GGM

Hardness in the structured GGM depends on how $\star$ is defined

Parameterization 1: $\star$ defined nowhere $\Rightarrow$ recover Shoup's generic group model



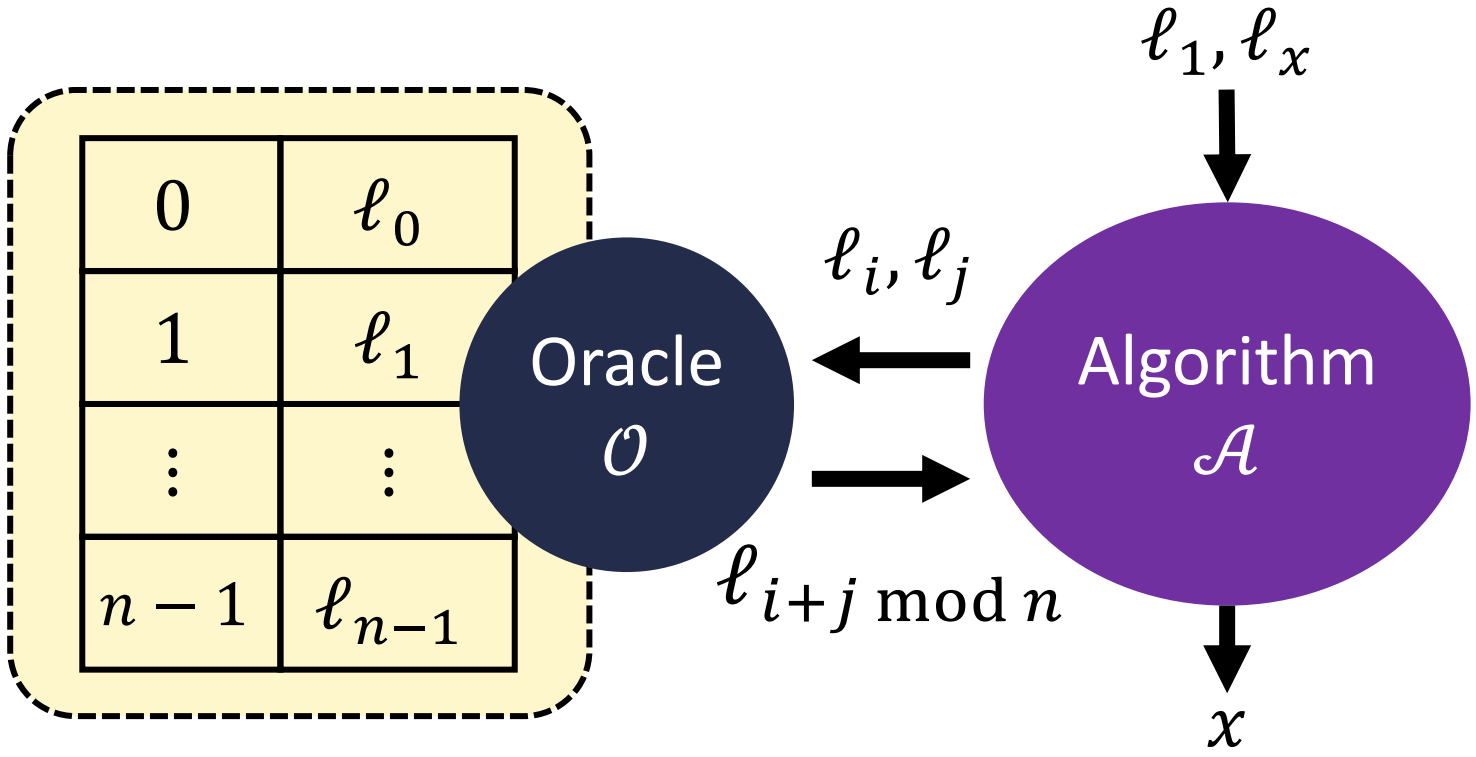


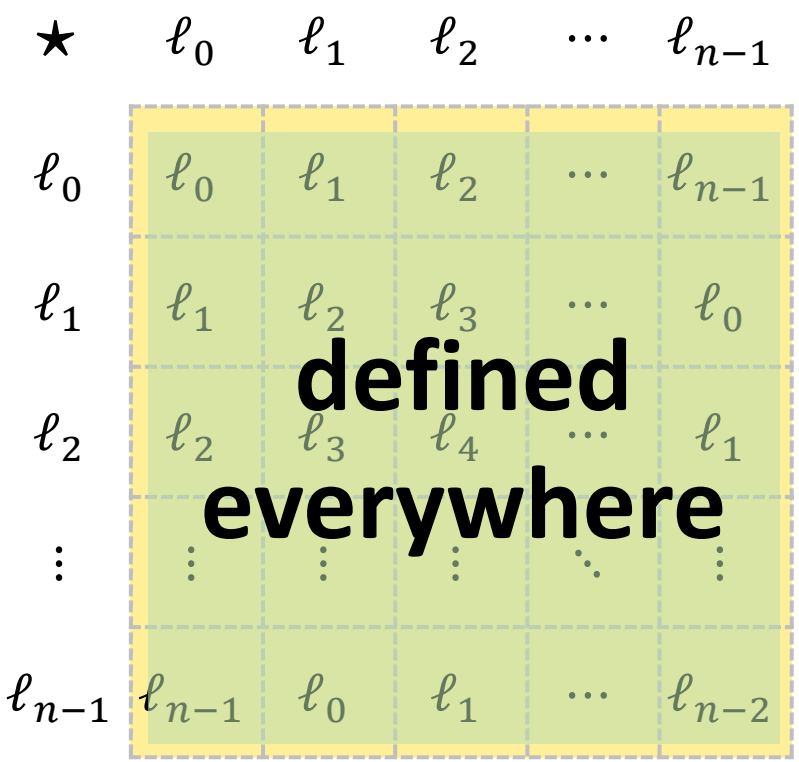
[Sho97]: Discrete log (for the uniform distribution on labels) requires $\Omega(\sqrt{n})$ queries

Discrete Log in the Structured GGM

Hardness in the structured GGM depends on how $\star$ is defined

Parameterization 2: $\star$ defined everywhere $\Rightarrow$ concrete group





Discrete log is information-theoretically easy (requires **0 queries**)

Example: Modeling Integer Factorization

Index calculus over $\mathbb{Z}_n^*$ exploits the ability to **factor smooth integers**

(1) lift group elements in $\mathbb{Z}_n^*$ to the integers $\mathbb{Z}$

(2) try to factor elements over the integers

Example: suppose we want to compute the discrete log of $35 \in \mathbb{Z}_n^*$

Suppose x_i is the discrete log of $i \in \mathbb{Z}_n^*$ with base g

$g^{x_{35}} = 35 \bmod n$ $\quad$ $35 = 5 \cdot 7$ over the integers

$g^{x_5} = 5 \bmod n$ $\quad$ **Therefore:** $g^{x_{35}} = g^{x_5} \cdot g^{x_7} = g^{x_5 + x_7}$

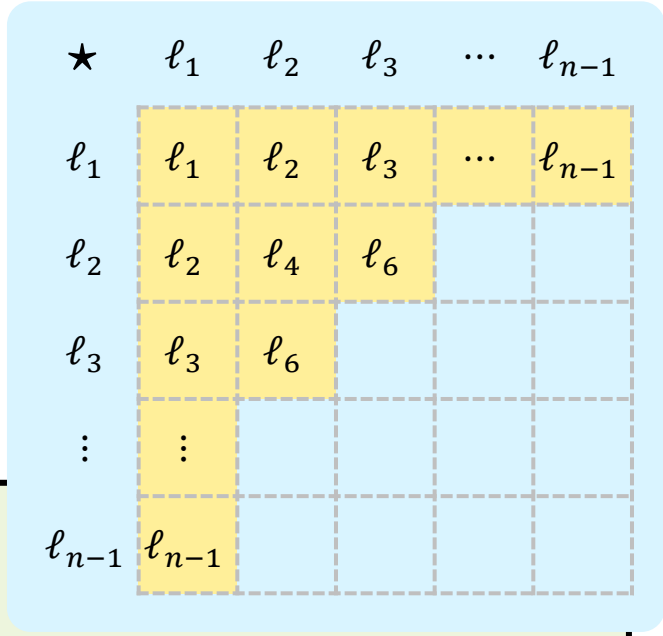
$g^{x_7} = 7 \bmod n$ $\quad$ **Index calculus:** solve for discrete logs of small primes

Example: Modeling Integer Factorization

Index calculus over $\mathbb{Z}_n^*$ exploits the ability to **factor smooth integers**

(1) lift group elements in $\mathbb{Z}_n^*$ to the integers $\mathbb{Z}$

(2) try to factor elements over the integers



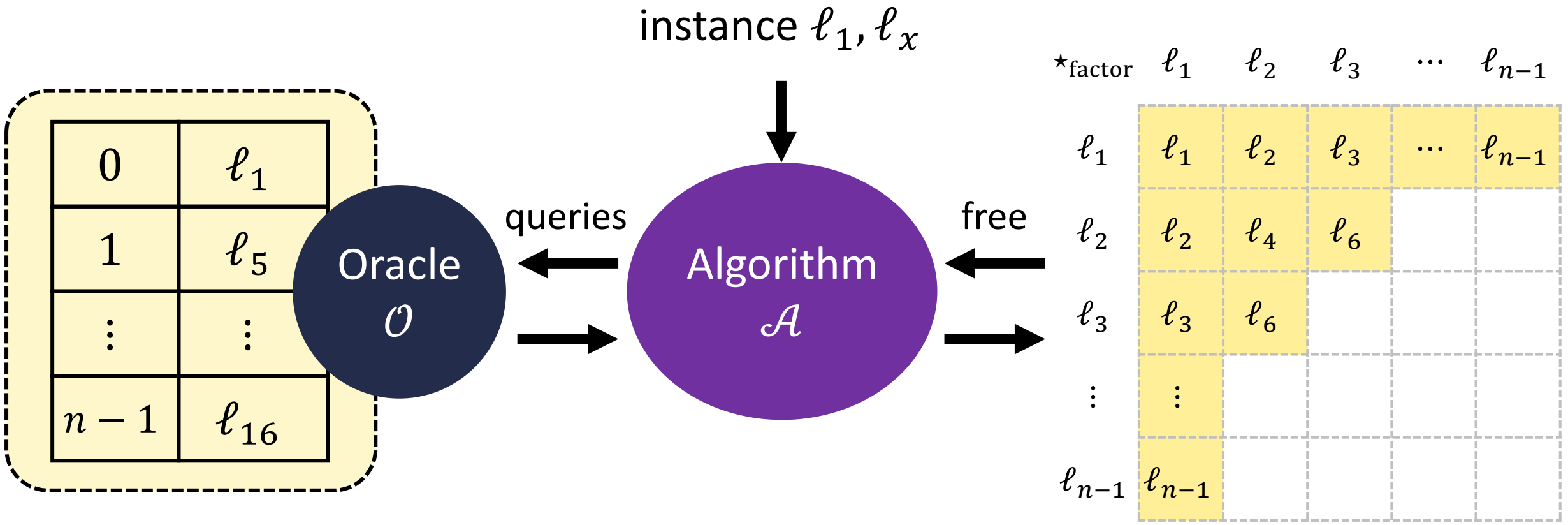
Can capture in the structured GGM with $\star_{\text{factor}}$:

Definition of $\star_{\text{factor}}$:

For a smoothness bound $B \in \mathbb{N}$ and all $i, j < B$, define $\ell_i \star_{\text{factor}} \ell_j = \ell_{i \cdot j}$

$\star_{\text{factor}}$ gives out factorization of all smooth integers for free

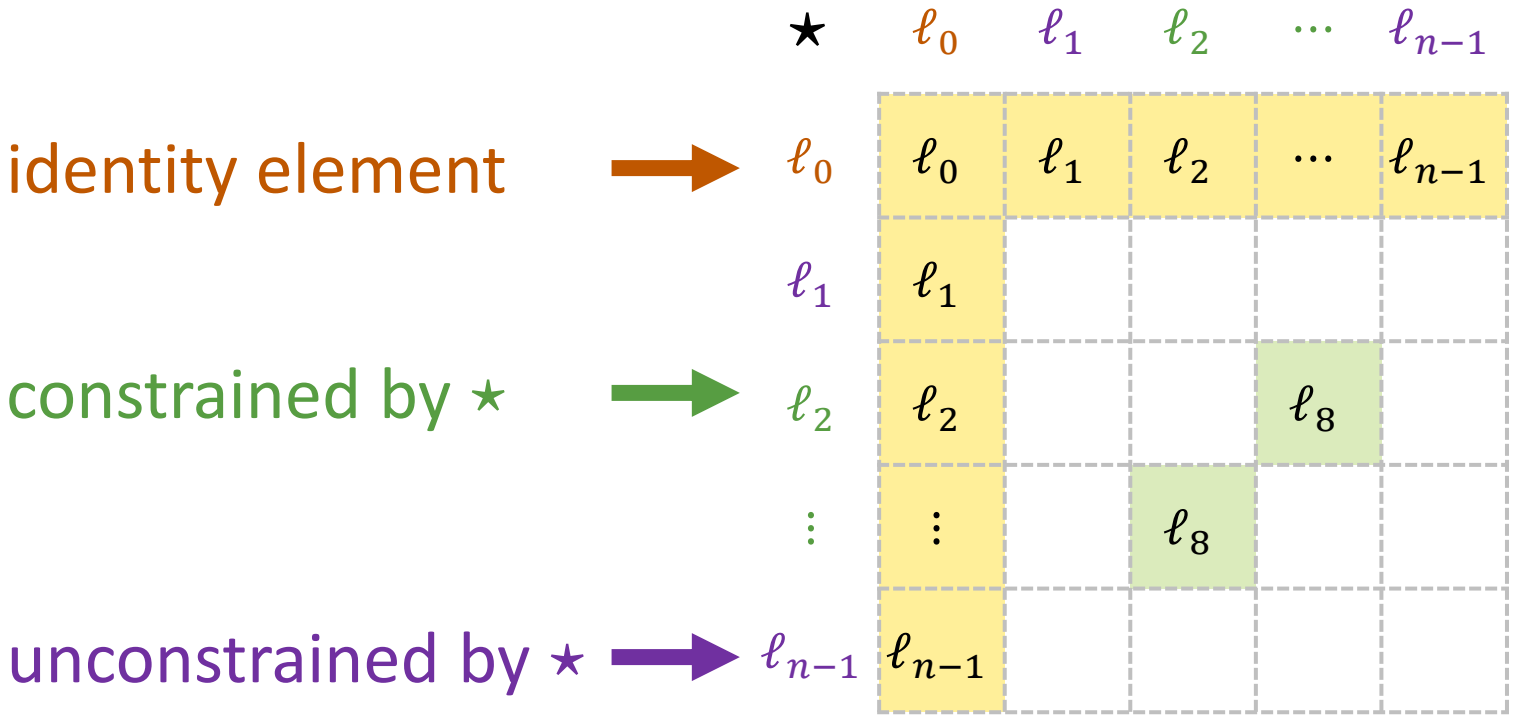
Example: Modeling Integer Factorization



Captures algorithms that are generic but can multiply B-smooth integers for free

Result 1: An Information-Theoretic Lower Bound

Definition. A label ℓ is "constrained by $\star$" if there exists a constraint $\ell_1 \star \ell_2 = \ell_3$ where $\ell \in \{\ell_1, \ell_2, \ell_3\}$ and ℓ_1, ℓ_2, ℓ_3 are not the identity label.



Result 1: An Information-Theoretic Lower Bound

Definition. A label ℓ is "constrained by $\star$" if there exists a constraint $\ell_1 \star \ell_2 = \ell_3$ where $\ell \in \{\ell_1, \ell_2, \ell_3\}$ and ℓ_1, ℓ_2, ℓ_3 are not the identity label.

Theorem. In the structured generic group model of prime order n where a δ-fraction of labels are "constrained by $\star$" (and at least one labeling exists), any algorithm

- that makes up to T oracle queries
- succeeds with probability at most $O(T^2/n + \delta T)$

$\Longrightarrow$ Beating generic algorithms requires exploiting the structure of $\geq 1/\sqrt{n}$-fraction of elements

Result 1: An Information-Theoretic Lower Bound

Theorem. In the structured generic group model of prime order n where a δ-fraction of labels are "constrained by $\star$" (and at least one labeling exists), any algorithm

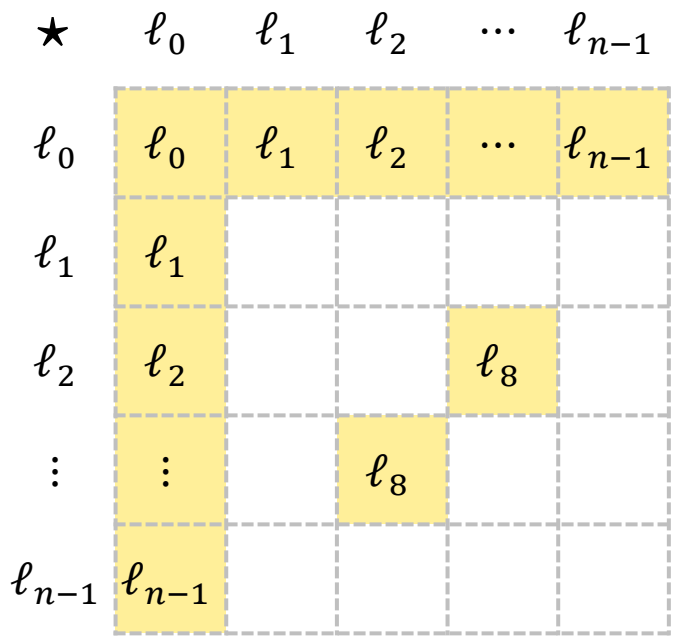
- that makes up to T oracle queries
- succeeds with probability at most $O(T^2/n + \delta T)$

Proof idea: To prove the lower bound, we need to exhibit *some* distribution over labels (that respects $\star$) where no algorithm can succeed

Constructing the hard distribution on labeling functions σ:

1. Take any labeling $\sigma_0 : \mathbb{Z}_n \to \mathcal{L}$ that respects $\star$
2. For every ℓ that is constrained by $\star$, let $\sigma^{-1}(\ell) = \sigma_0^{-1}(\ell)$
3. For remaining labels ℓ, sample $\sigma^{-1}(\ell)$ randomly from unassigned discrete logs

Valid labeling by construction



Result 1: An Information-Theoretic Lower Bound

Hard distribution:

1. Take any labeling $\sigma_0 : \mathbb{Z}_n \to \mathcal{L}$ that respects $\star$
2. For every ℓ that is constrained by $\star$, let $\sigma^{-1}(\ell) = \sigma_0^{-1}(\ell)$
3. For remaining labels ℓ, sample $\sigma^{-1}(\ell)$ randomly from unassigned discrete logs

Hardness follows via a Shoup-style argument

Consider an alternative oracle using the following simulated table

generator ℓ_1 and discrete log challenge ℓ_x

discrete logs for $\star$-constrained labels

remaining discrete logs are lazily sampled as adversary makes queries

1	ℓ_1
X	ℓ_x
5	ℓ_{17}
7	ℓ_6
32	ℓ_{143}

X represents formal variable (for discrete log challenge)

oracle samples the value of X at end of the experiment

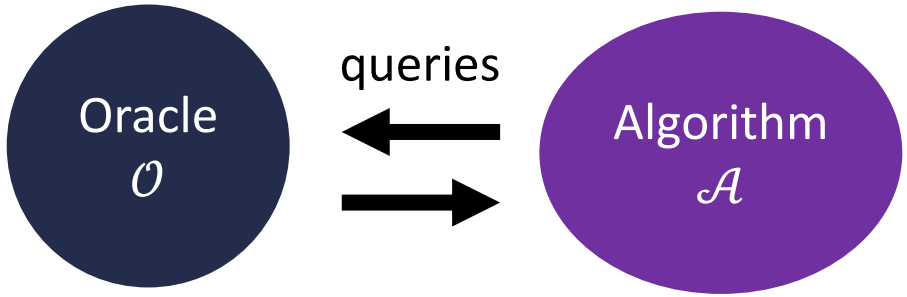


table now maps formal polynomials in X to labels
(distinct polynomials ⇒ discrete labels)

Result 1: An Information-Theoretic Lower Bound

Hard distribution:

1. Take any labeling $\sigma_0 : \mathbb{Z}_n \to \mathcal{L}$ that respects $\star$
2. For every ℓ that is constrained by $\star$, let $\sigma^{-1}(\ell) = \sigma_0^{-1}(\ell)$
3. For remaining labels ℓ, sample $\sigma^{-1}(\ell)$ randomly from unassigned discrete logs

Oracle's table at the end of the experiment:

generator ℓ_1 and discrete log challenge ℓ_x

discrete logs for $\star$-constrained labels

adversary oracle queries

1	ℓ_1
X	ℓ_x
5	ℓ_{17}
7	ℓ_6
32	ℓ_{143}
$4X + 7$	ℓ_{11}
$3X + 15$	ℓ_{209}

these values are all distinct by construction

Perfectly simulates real distribution as long as after instantiating X with $x \leftarrow \mathbb{Z}_n$, each row has a **distinct** discrete log

Result 1: An Information-Theoretic Lower Bound

Hard distribution:

1. Take any labeling $\sigma_0 : \mathbb{Z}_n \to \mathcal{L}$ that respects $\star$
2. For every ℓ that is constrained by $\star$, let $\sigma^{-1}(\ell) = \sigma_0^{-1}(\ell)$
3. For remaining labels ℓ, sample $\sigma^{-1}(\ell)$ randomly from unassigned discrete logs

Oracle's table at the end of the experiment:

generator ℓ_1 and discrete log challenge ℓ_x	1	ℓ_1
	X	ℓ_x
discrete logs for $\star$-constrained labels	5	ℓ_{17}
	7	ℓ_6
	32	ℓ_{143}
adversary oracle queries	$4X + 7$	ℓ_{11}
	$3X + 15$	ℓ_{209}

Perfectly simulates real distribution as long as after instantiating X with $x \leftarrow \mathbb{Z}_n$, each row has a **distinct** discrete log

Two types of collisions:

- Two polynomials share a common value at X
  Probability: $O(T^2)/n$
- Value of polynomial equals a fixed discrete log
  Probability: $O(\delta n T)/n = O(\delta T)$

If neither happens, then adversary succeeds with probability $1/n$

Result 1: An Information-Theoretic Lower Bound

More generally: can extend to groups with composite-order n

Theorem. In the structured generic group model of order n where a δ-fraction of labels are "constrained by $\star$" (and at least one labeling exists) any algorithm

- that makes up to T oracle queries
- succeeds with probability at most $O(T^2/q + \delta nT/q)$, where q is the largest prime factor dividing n

Can also extend to the preprocessing model; see paper for details.

Example: Modeling the Integers

Consider $\mathbb{Z}_n^*$ and let q be the largest prime factor of n

Consider $\star_{\text{factor}}$ that is defined on all ℓ_i, ℓ_j where $i \cdot j < B$

label space $\mathcal{L} = \mathbb{Z}_n^*$

Fact (Density of Smooth Integers) [Gra08]

The fraction of positive integers $\leq x$ that factor into primes $\leq L(x)$ is $1/L(x)^{1/2+o(1)}$, where $L(x) = \exp\left(\sqrt{\log x \log\log x}\right)$.

Take $B = L(q)$. Density of smooth values $\delta = 1/L(q)^{1/2+o(1)}$.

Lower bound in the $\star_{\text{factor}}$-model:

Success probability is $O(T^2/q + \delta nT/q)$

For $q = \Omega(n)$, constant success prob. requires

$$T > \Omega(1/\delta) = \exp\left(\Omega\left(\sqrt{\log q \log\log q}\right)\right)$$

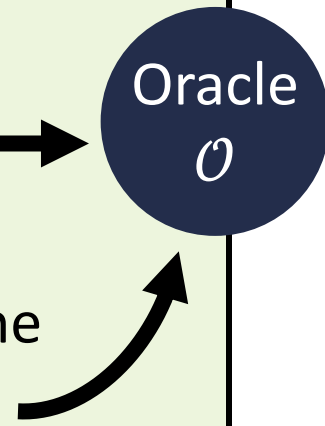
Next: we will derive a corresponding upper bound in the $\star_{\text{factor}}$-structured generic-group model

Result 2: A Tight Index-Calculus Upper Bound

Observation: classic index-calculus attacks are captured by our model [Adl79, Pom87, EG02]

Algorithm sketch:

1. Find a "factor base" in $\star$: a set of group elements $S = \{\ell_1, \ldots, \ell_k\}$ such that a random element ℓ factors over S (i.e., $\ell = \ell_1 \star \cdots \star \ell_t$) with probability γ
2. Collect linear relationships on the discrete logs of the factor base
3. Solve for the discrete logs of the factor base (via Gaussian elimination)
4. Express the target discrete-log as a linear function of the discrete logs of the factor base

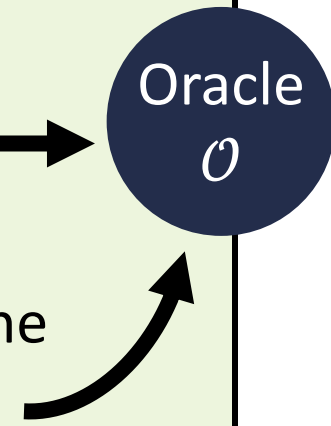


Result 2: A Tight Index-Calculus Upper Bound

Observation: classic index-calculus attacks are captured by our model [Adl79, Pom87, EG02]

Algorithm sketch:

1. Find a "factor base" in $\star$: a set of group elements $S = \{\ell_1, \ldots, \ell_k\}$ such that a random element ℓ factors over S (i.e., $\ell = \ell_1 \star \cdots \star \ell_t$) with probability γ
2. Collect linear relationships on the discrete logs of the factor base
3. Solve for the discrete logs of the factor base (via Gaussian elimination)
4. Express the target discrete-log as a linear function of the discrete logs of the factor base



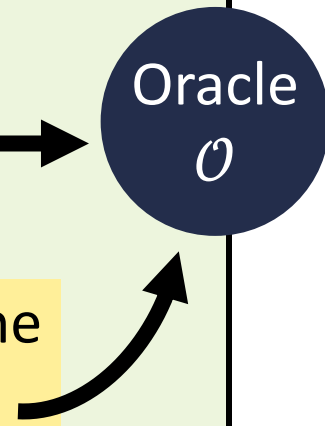
- Let discrete logs of factor base be $y_1, \ldots, y_k$
- Use oracle to compute $\sigma(r)$ for random $r \leftarrow \mathbb{Z}_n$
- If $\sigma(r)$ factors over S, then we obtain a linear combination $r = \sum \alpha_i y_i$

Result 2: A Tight Index-Calculus Upper Bound

Observation: classic index-calculus attacks are captured by our model [Adl79, Pom87, EG02]

Algorithm sketch:

1. Find a "factor base" in $\star$: a set of group elements $S = \{\ell_1, \dots, \ell_k\}$ such that a random element ℓ factors over S (i.e., $\ell = \ell_1 \star \cdots \star \ell_t$) with probability γ
2. Collect linear relationships on the discrete logs of the factor base
3. Solve for the discrete logs of the factor base (via Gaussian elimination)
4. Express the target discrete-log as a linear function of the discrete logs of the factor base



- Use random self-reducibility
- Sample random $r \leftarrow \mathbb{Z}_n$ and check if $\ell_{x'} = \mathcal{O}\big(\ell_x, \sigma(r)\big)$ factors over S
- If z is discrete log of $\ell_{x'}$, then output $x = z - r$

Result 2: A Tight Index-Calculus Upper Bound

Observation: classic index-calculus attacks are captured by our model [Adl79, Pom87, EG02]

Running time of index calculus algorithm depends on the smoothness of $\star$ (i.e., the fraction of elements that factor over S)

If δ-fraction of elements factor over S, then

- algorithm makes $\tilde{O}(|S|/\delta)$ queries
- and succeeds with constant probability

Example: Modeling the Integers

Consider $\mathbb{Z}_n^*$ and let q be the largest prime factor of n

Consider $\star_{\text{factor}}$ that is defined on all ℓ_i, ℓ_j where $i \cdot j < B$

Can also consider algorithms that lift to smooth polynomials

Fact (Density of Smooth Integers) [Gra08]

The fraction of positive integers $\leq x$ that factor into primes $\leq L(x)$ is $1/L(x)^{1/2+o(1)}$, where $L(x) = \exp\left(\sqrt{\log x \log\log x}\right)$.

Take $B = L(q)$. Density of smooth values $\delta = 1/L(q)^{1/2+o(1)}$.

Lower bound in the $\star_{\text{factor}}$-model:

Success probability is $O(T^2/q + \delta nT/q)$

For $q = \Omega(n)$, constant success prob. requires

$$T > \Omega(1/\delta) = \exp\left(\Omega\left(\sqrt{\log q \log\log q}\right)\right)$$

Upper bound in the $\star_{\text{factor}}$-model:

Let S be the primes less than B, so $|S| < L(q)$

Algorithm runs in time $\tilde{O}(|S|/\delta) = \exp\left(O\left(\sqrt{\log q \log\log q}\right)\right)$

Comparison with Other Models

Preprocessing model [CDG18, CK18]:

Generic group model where adversary gets S bits of advice about generic group oracle

$\star$-constraints are ***less*** powerful than arbitrary bits of advice

- $\star$-constraints only encode **local** information about group operation
- each bit of preprocessed advice can be a global function of generic group oracle

If we treat $\star$-constraints as giving out $\approx \delta n$ bits of advice, then preprocessing lower bound would bound the advantage by $\varepsilon < \delta T^2 + T^2/n$

In our model, we can show a **stronger** bound of $\varepsilon < \delta T + T^2/N$

Comparison with Other Models

Bit-fixing model [Unr07, CDG18, CDGS18]:

Adversary is given ability to fix labeling function at S points during the preprocessing step

Bit-fixing lower bounds can be used to derive our lower bound

(e.g., fix the values determined by $\star$)

Algorithms in the bit-fixing model do **not** translate to algorithms in the structured generic group model

- $\star$-constraints are given to the adversary; adversary cannot pick their values
- Our index calculus algorithm works over *any* label space that is smooth

Comparison with Other Models

Algebraic group model [FKL18]:

- Algebraic group model captures **reductions** between problems
- Does not show information-theoretic lower bounds on discrete log by itself (though in some settings, hardness translates to Shoup's generic group model)

Smooth generic-group model [Hha25]:

- Aims to model index calculus algorithms via a "smoothing oracle" that outputs special information if input element falls into a special set (representing smooth values)
- Smooth model is ***inconsistent*** (multiple labels can have the **same** discrete log)

Comparison with Other Models

Algebraic group model [FKL18]:

Algebraic group model captures **reductions** between problems

Does not show information-theoretic lower bounds on discrete log by itself (though in some settings, hardness translates to Shoup's generic group model)

Smooth generic-group model [Hha25]:

Aims to model index calculus algorithms via a "smo
information if input element falls into a special set

Model associates a random discrete log with each small prime – assignment very likely to produce collisions!

Smooth model is ***inconsistent*** (multiple labels can have the **same** discrete log)

Lower bound in smooth model does **not** rule out algorithm that works in all concrete groups: a good algorithm might always fail when there are inconsistent labels, but work in all groups with consistent labels

Comparison with Other Models

Algebraic group model [FKL18]:

Algebraic group model captures **reductions** between problems

Does not show information-theoretic lower bounds on discrete log by itself (though in some settings, hardness translates to Shoup's generic group model)

Smooth generic-group model [Hha25]:

Aims to model index calculus algorithms via a "smoothing oracle" that outputs special information if input element falls into a special set (representing smooth values)

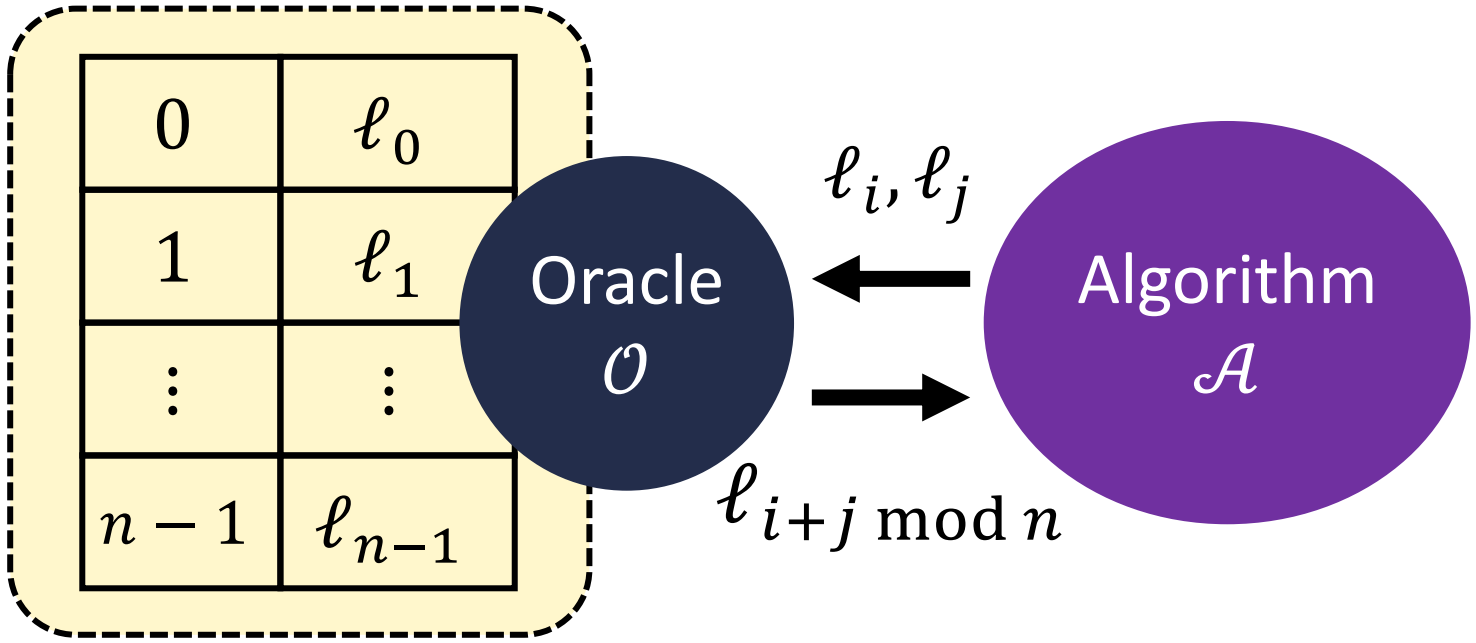
Smooth model is ***inconsistent*** (multiple labels can have the same discrete log)

Upper bound in smooth model does **not** imply algorithm that works in any concrete groups: such an algorithm might critically exploit the fact that multiple labels have the same discrete log

The Structured Generic-Group Model

Goal: reason about algorithms that exploit some non-generic-group structure

Idea: give algorithm free access to some group-operation outputs



$\star$	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_0	ℓ_0	ℓ_1	ℓ_2	$\cdots$	ℓ_{n-1}
ℓ_1	ℓ_1	ℓ_2	ℓ_3		
ℓ_2	ℓ_2	ℓ_3		ℓ_8	
$\vdots$	$\vdots$		ℓ_8		
ℓ_{n-1}	ℓ_{n-1}				

Model expressive enough to capture algorithms like index calculus and prove lower bounds

Open Problems

Extending the model to capture algorithms like the number-field sieve, function-field sieve (e.g., multiple representations + factorizations per label)

Is discrete log hard for an algorithm that can peel off the B-smooth part of any label for free?

Is discrete log hard for an algorithm that can factor *any* label for free?

Thanks!

https://eprint.iacr.org/2026/384.pdf