Los Alamos
NATIONAL LABORATORY

Private and Verifiable Model Evaluation using Secure Multiparty Computation and Zero-knowledge Proofs

Michael Dixon
A-4: Advanced Research in Cyber Systems
Email: mdixon@lanl.gov

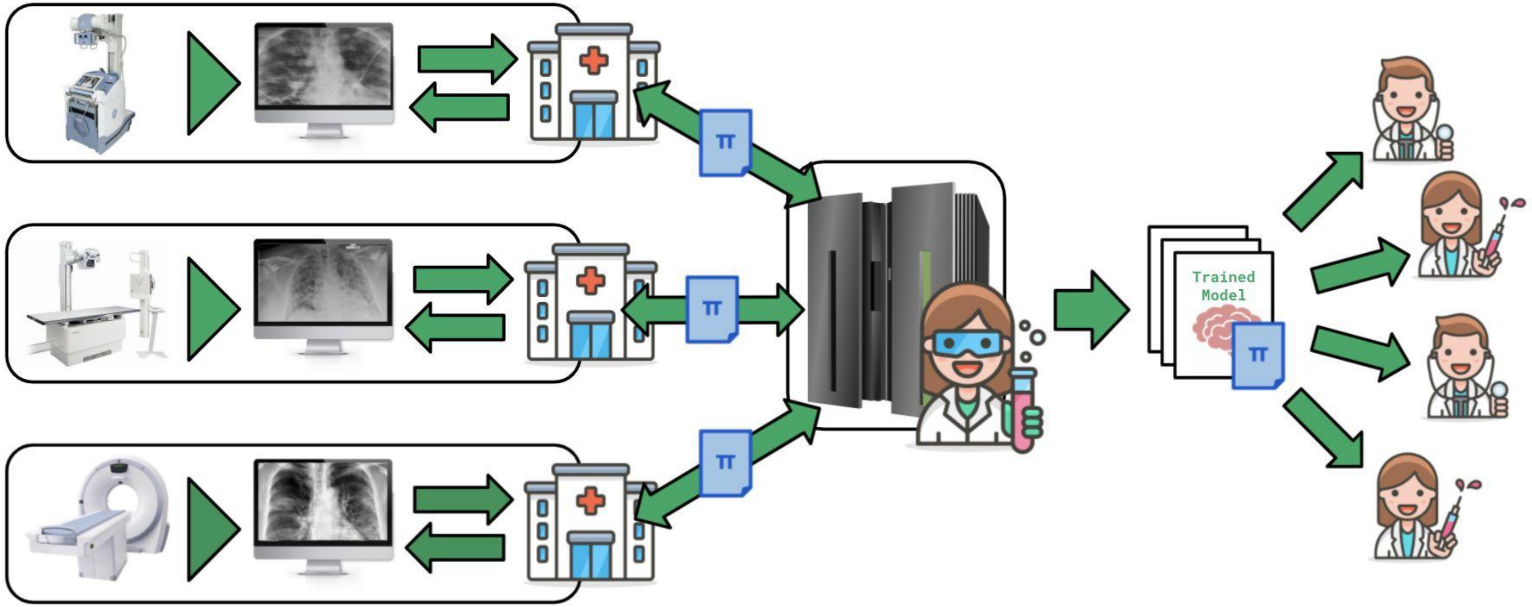
(Joint work with Zachary DeStefano and Dani Barrack)
Co-Mentor: Juston Moore

September 13th, 2021

ISTI Information Science & Technology Institute

LA-UR-21-28963

Prior Work: Privacy-Preserving and Verifiable Federated ML



For FY20 ISTI Novel Compute, we developed an application using zkSNARKs to verify the computational integrity of a federated neural network training process without exposing sensitive input data or revealing model parameters during classification

Goal: Verifying Additional Model Qualities

Provable Guarantees

The model training was *executed* correctly

No training data *tampering* occurred

The model was trained on the *full dataset*

No private data was *leaked* in the process

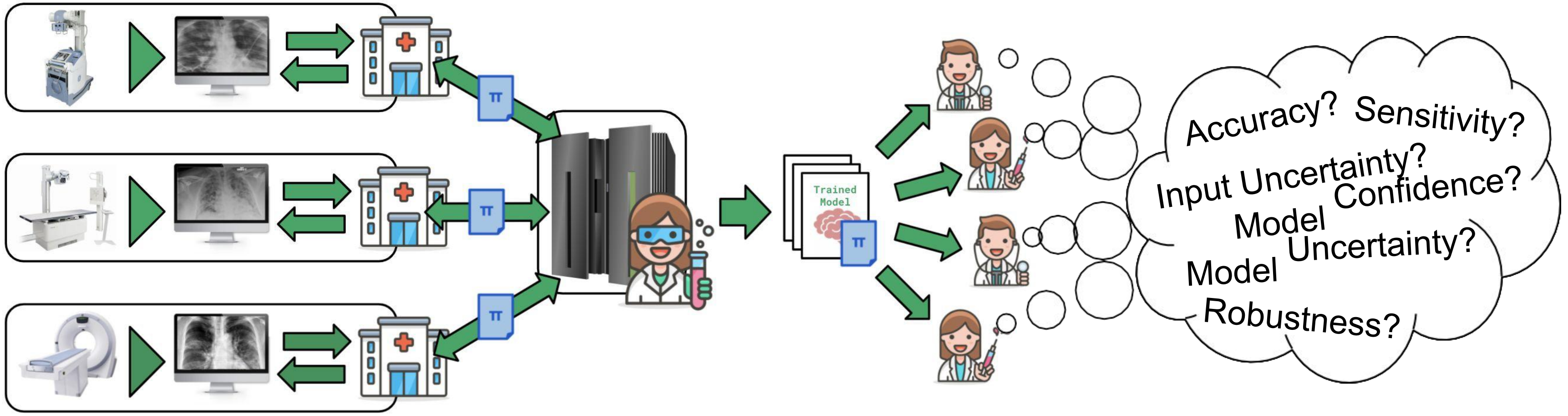
Unknowns (not captured in prior work)

NO guarantee that the model *performs* well

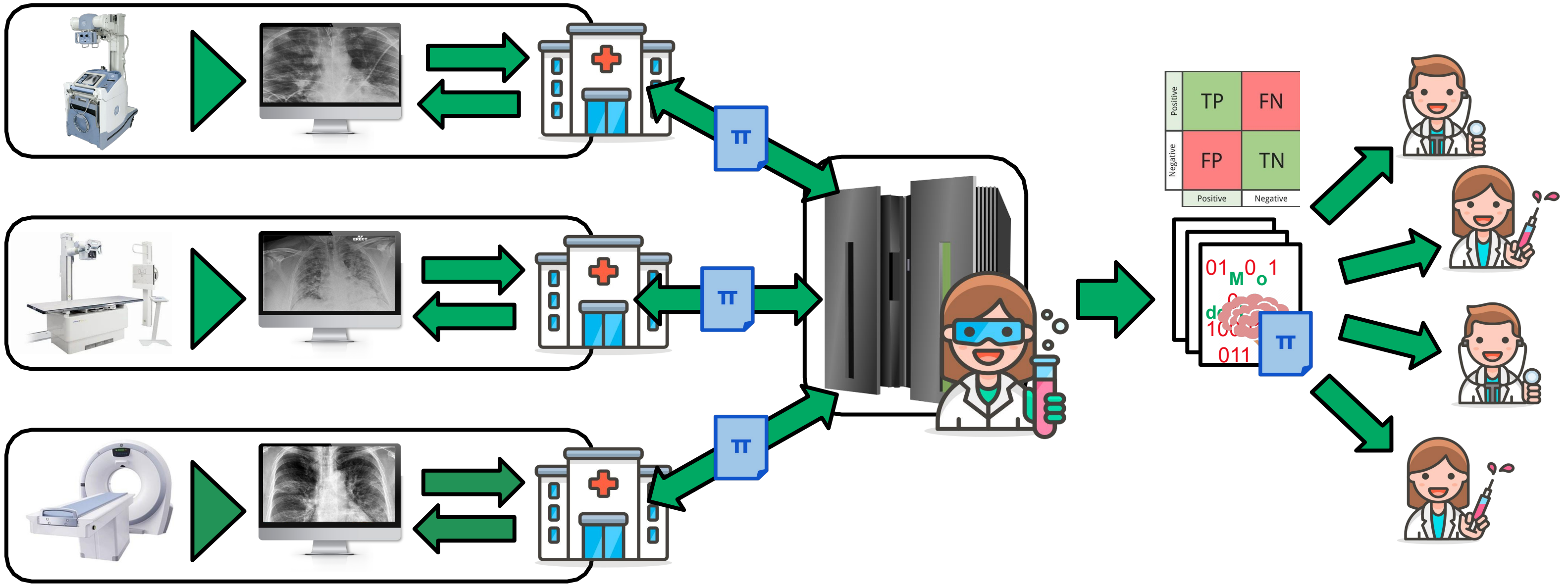
NO quantification of model *uncertainty*

NO evaluation of *risk* or *regret*

NO measure of model *robustness*



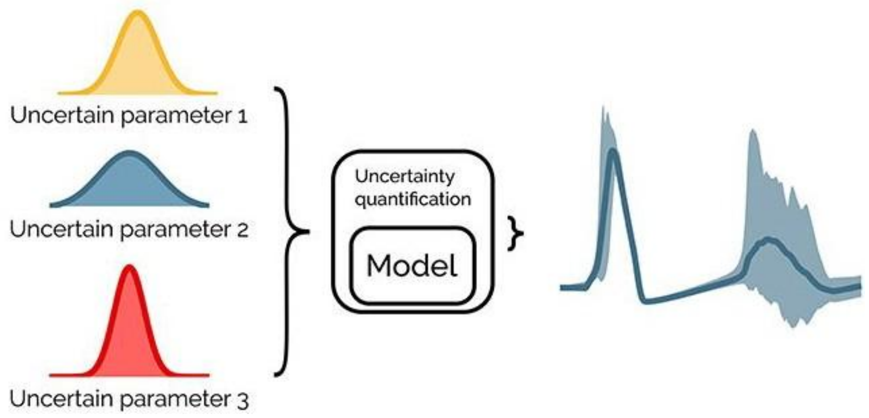
Model Performance and Confusion Matrix



Annotating current framework with additional metadata regarding tests creates a confusion matrix proven consistent with the model by a ZKP.

What about Uncertainty Quantification?

Many approaches to UQ require reasoning about randomness, sampling, and probabilities



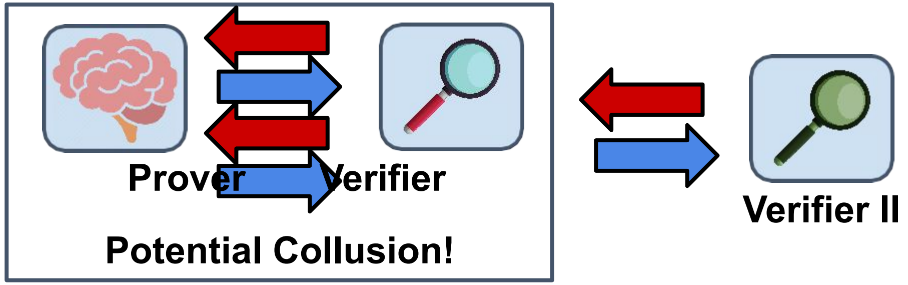
Problem

Non-interactive ZKPs currently only capture *deterministic* computations

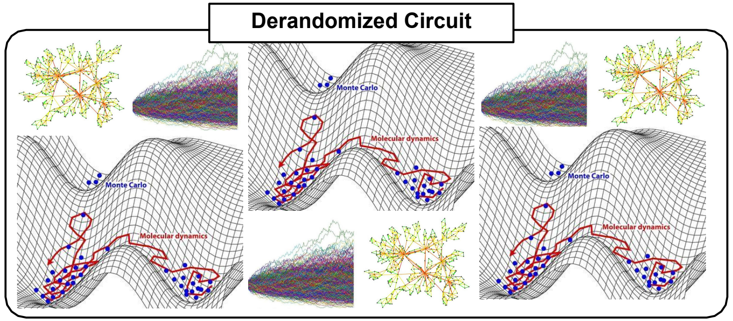
Solution Attempt I

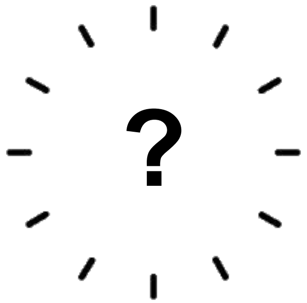
Verifier supplies input randomness
Prover supplies ZKP of computation

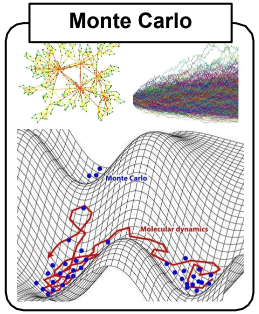
NOTE: This becomes fully interactive and is not recursively composable

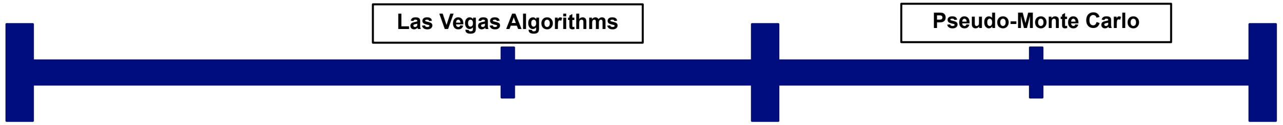


Derandomization, Certainty, and a Goldilocks Solution



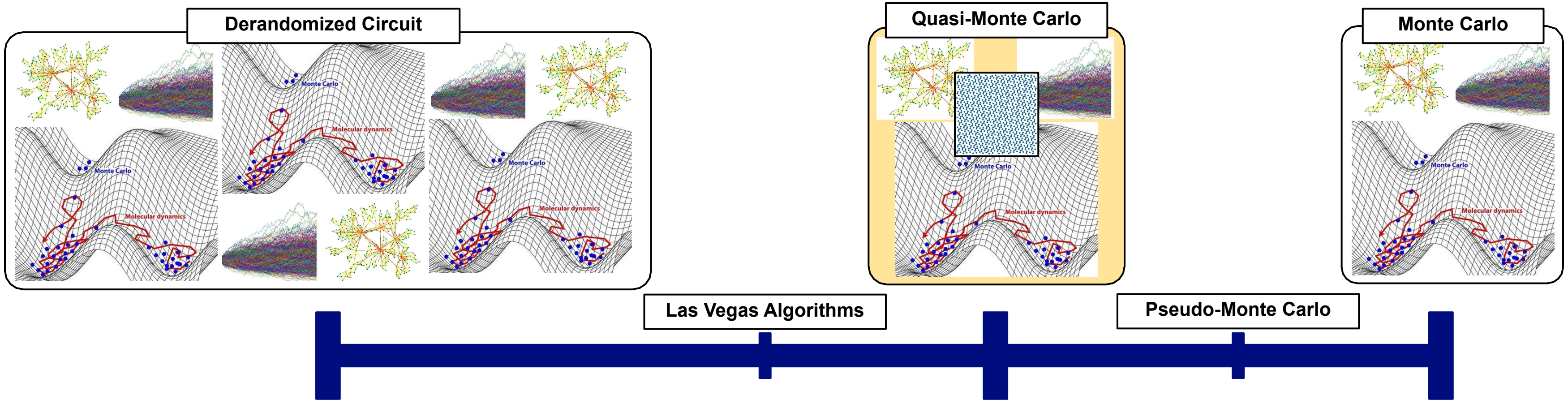






Circuit Creation Overhead	Intractable * (Usually EXPTIME)	...	**Tractable**	...	None
Worst Case Prover Deception	No Error	...	**$0 < \varepsilon < C$**	...	Unbounded

Derandomization, Certainty, and a Goldilocks Solution

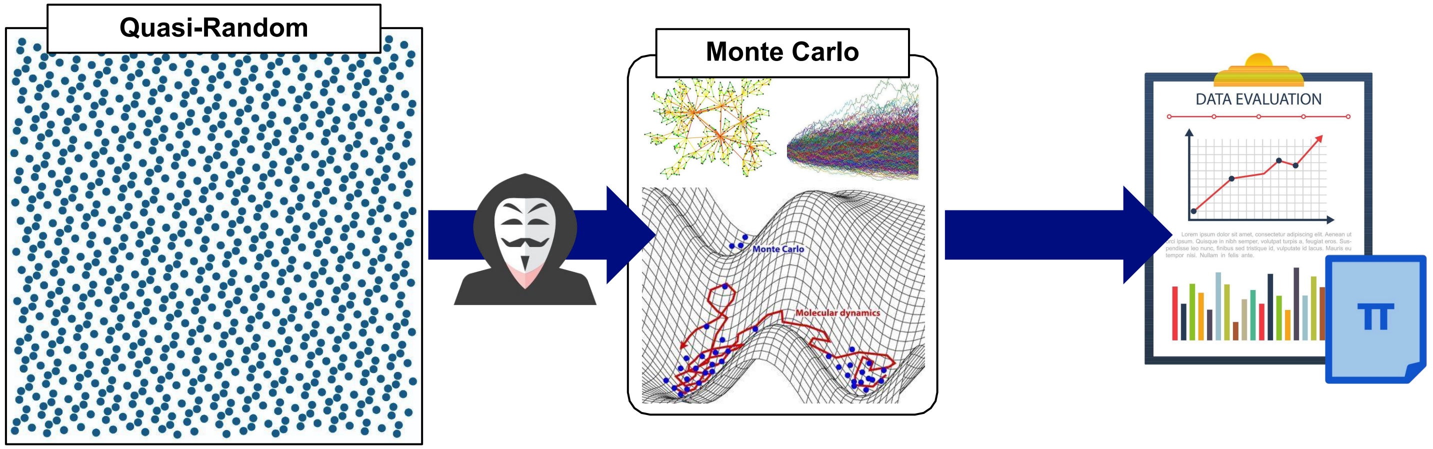


Circuit Creation Overhead	Intractable * (Usually EXPTIME)	…	$O(d)$	…	None
Worst Case Prover Deception	No Error	…	$O\left(\frac{(\log N)^d}{N}\right)$	…	Unbounded

Quasi-Monte Carlo (QMC) and Non-Interactive ZKPs

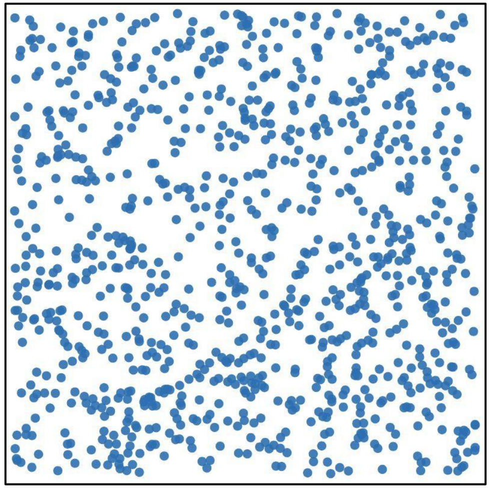
Quasi-Monte Carlo is the application of *quasi-random sequences* in place of *randomness* for Monte Carlo algorithms and simulations.

These are *cheap* ways to provide *provable guarantees* on the *worst case performance* of Monte Carlo techniques which gives us enough *adversarial robustness* to apply them to non-interactive zero-knowledge proof applications

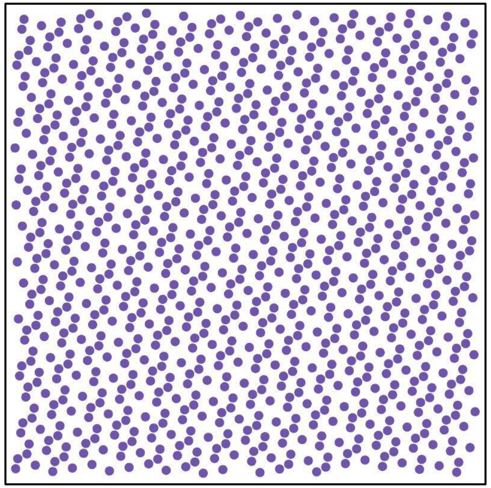


Pseudo-randomness vs Quasi-randomness Sequences

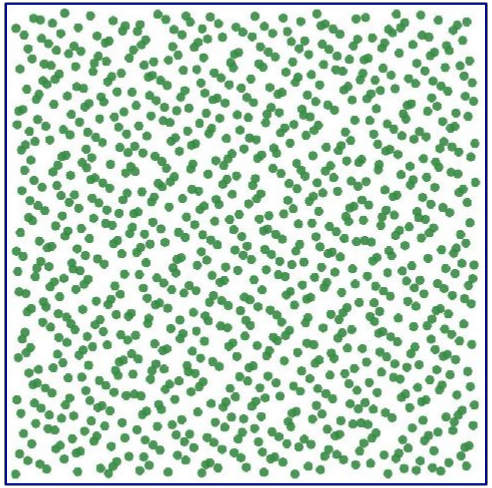
Pseudo-Random



$R(\phi_2)$ **Quasi-Random**

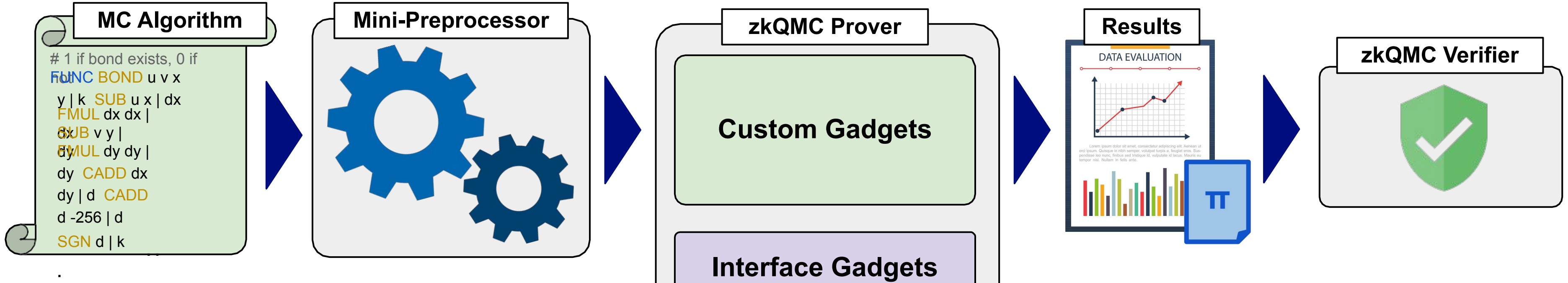


(2,3) Halton Quasi-Random

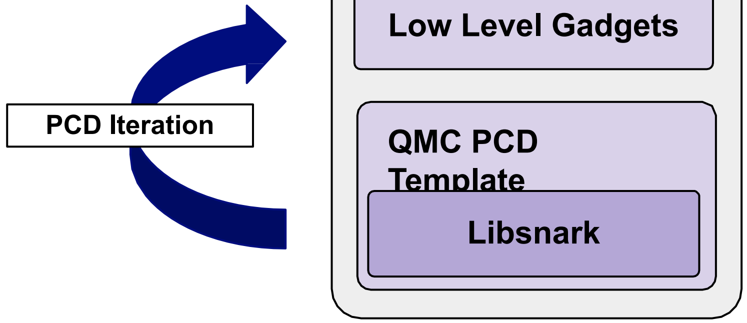


Quasi-randomness is *low discrepancy*, so it provides optimal coverage of the space and *guaranteed convergence*, even in the worst case, making it robust enough against malicious provers in the ZKP setting

High-Level zkSNARK Quasi-Monte Carlo Architecture



```
# 1 if bond exists, 0 if not
FUNC BOND u v x
  y | k  SUB u x | dx
  FMUL dx dx |
  dx SUB v y |
  dy FMUL dy dy |
  dy  CADD dx
  dy | d  CADD
  d -256 | d
  SGN d | k
```



We pack several *QMC iterations* into each *PCD iteration* for performance

zkQMC Prototype Engine

We developed this engine in C++ for the rapid prototyping of QMC applications in zero-knowledge

Demo: Monte Carlo Estimation of π

Monte Carlo "Hello World" in Zero-Knowledge

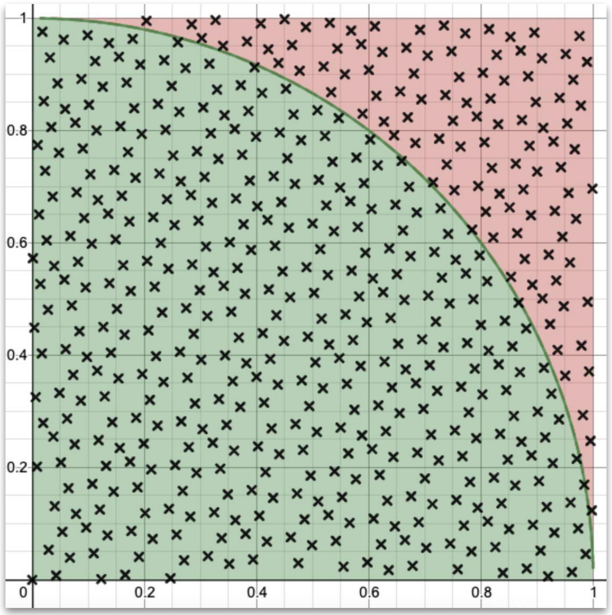
My first Quasi-Monte Carlo application whose outputs can be verified using a zkSNARK estimates the value of π.

Implementation Details

This QMC specification is 8 short lines which are preprocessed into a 78 line C++ file, and compiled with our zkQMC backend

This spec references a variety of lower-level fixed point gadgets and our library completely handles the heavy lifting involved with zkSNARKs, PCD, and QMC

A zero-knowledge proof that **π = 3.133 ± 0.04**



$$\pi \approx \frac{\text{4 x green hits}}{\text{total trials}}$$

Demo: Cluster Integration

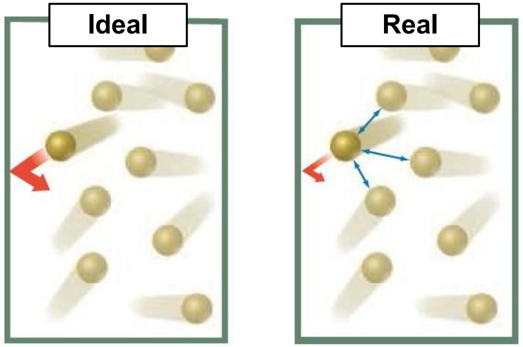
Cluster Integration in Zero-Knowledge

I implemented the algorithms *developed at LANL* and detailed in *Equation of State Calculations by Fast Computing Machines* by Metropolis, et al. to compute *cluster integrals* in 2D and 3D in my zkQMC framework.

Accurate solutions to *high dimensional integrals* are vital for handling *intermolecular interactions* (top right) in real gas *simulations*.

To illustrate the *complexity* of this computation, we have a *diagram* from that paper here (right), and the *integral* corresponding to $A_{5,9}$ (below).

$$A_{5,9} = c \int \cdots \int (f_{14} f_{15} f_{24} f_{25} f_{35}) d\tau_1 \cdots d\tau_5$$



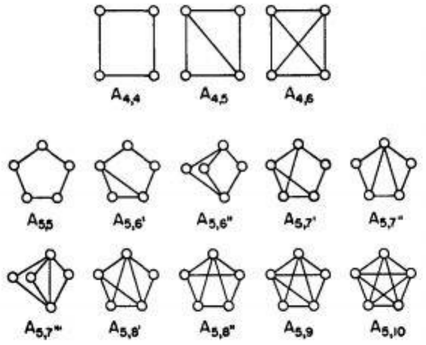
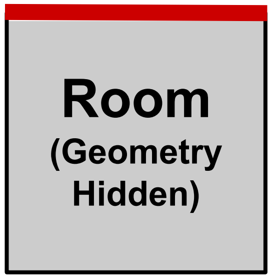
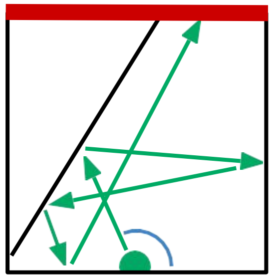


FIG. 6. Schematic diagrams for the various area integrals.

Demo: Simple Particle Exposure Simulation UQ with ZKPs

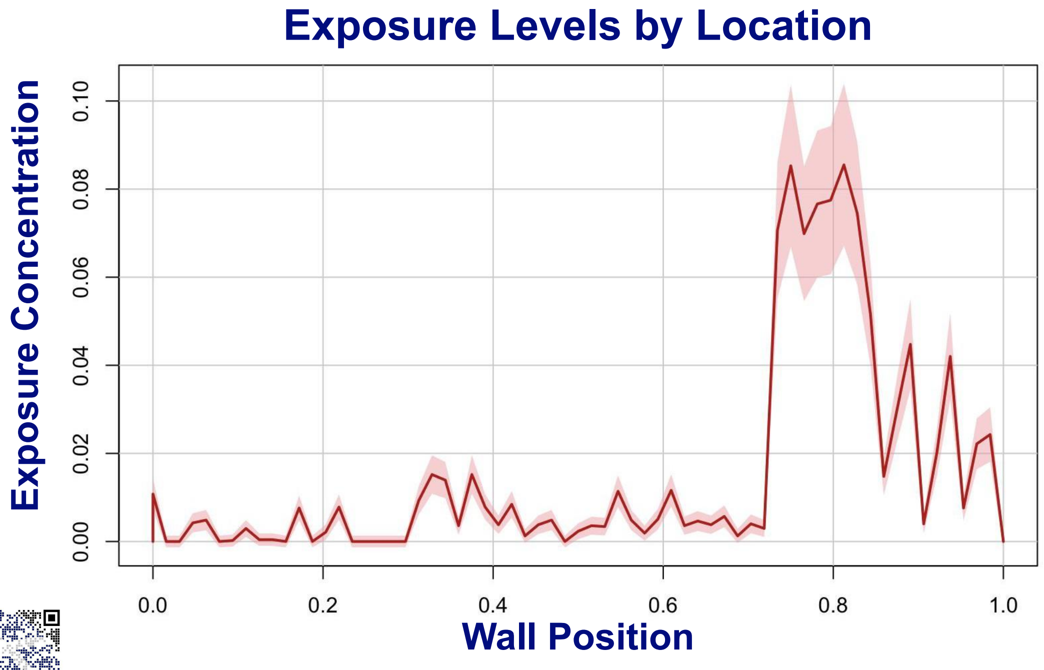




We model exposure levels by a toy QMC simulation.

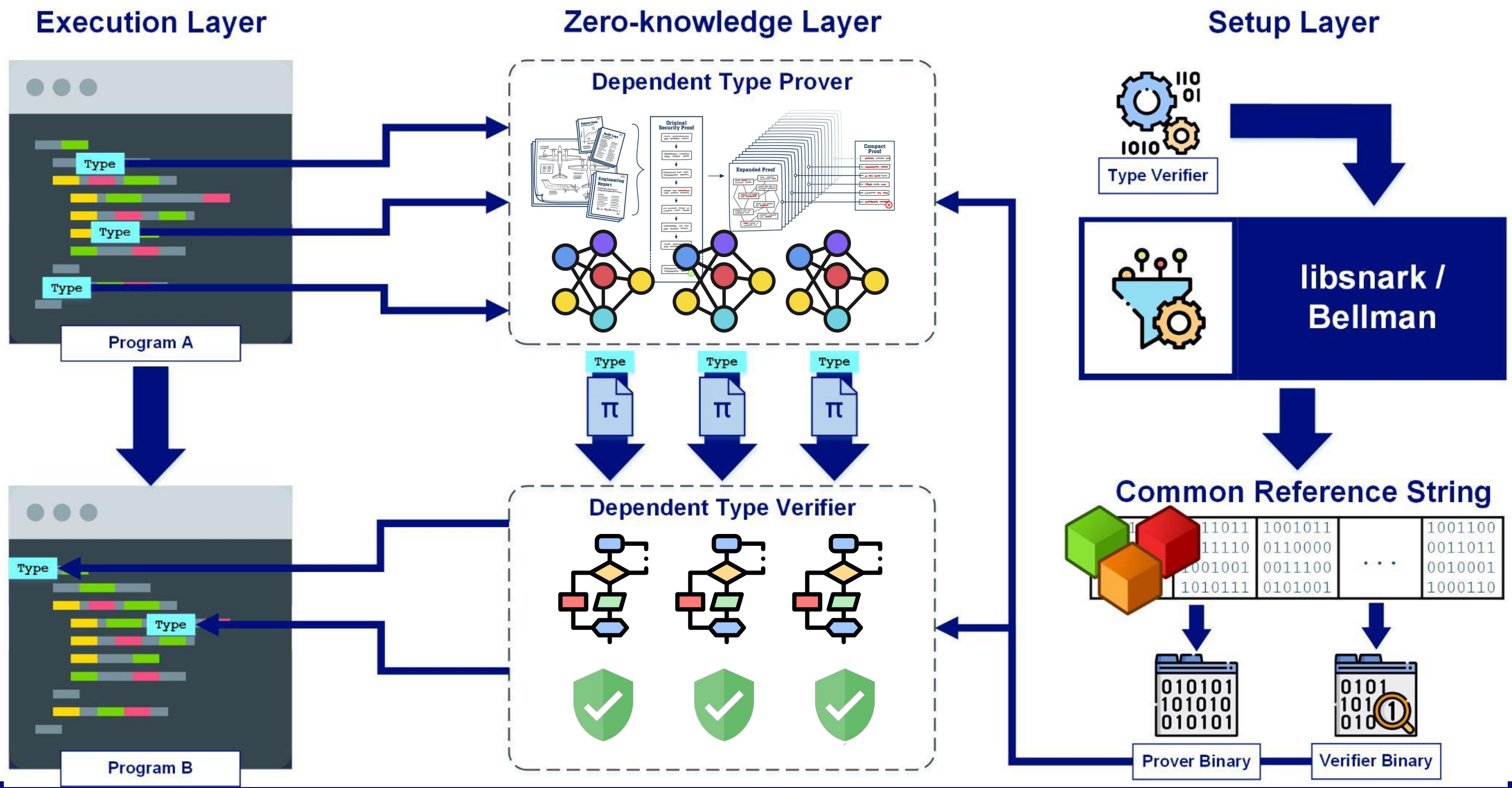
Simulated particles with random initial velocity are emitted from a source. They bounce around the room until it hits the back wall or times out.

Uncertainty is computed from measurements and model



The ZKP (in the QR code on the left) attests to the integrity of the UQ results from the QMC simulation

Dependent Type Replacement by ZKPs



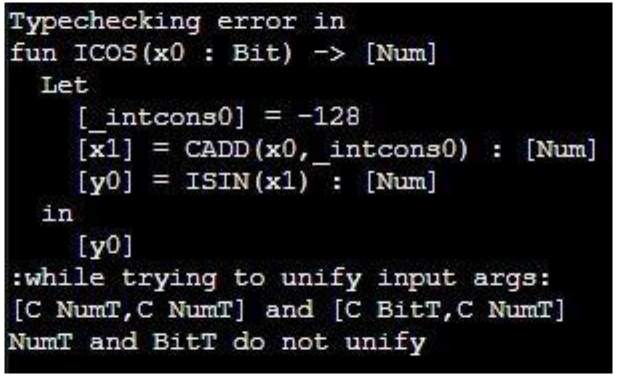
Type Checking Pipeline using ZKPs

A simple, custom DSL is compiled down to:

1. Relevant libsnark gadgets
2. Type-level Haskell

Haskell type checker formally verifies type safety and correctness of ZKP integration.

Hard or impossible to prove properties are enforced using a ZKP that guarantees that data has the proper types or provenance



```
Typechecking error in
fun ICOS(x0 : Bit) -> [Num]
  Let
    [_intcons0] = -128
    [x1] = CADD(x0,_intcons0) : [Num]
    [y0] = ISIN(x1) : [Num]
  in
    [y0]
:while trying to unify input args:
[C NumT,C NumT] and [C BitT,C NumT]
NumT and BitT do not unify
```

A type checker for ZKP gadgets will catch bugs at compile time

This successfully caught a bug in our existing code already!

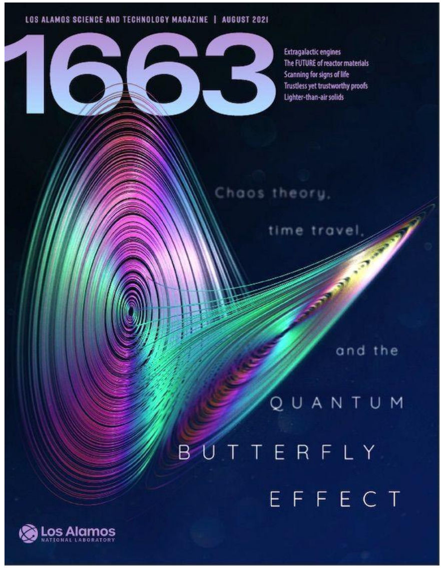
Accomplishments & Future Work

Accomplishments

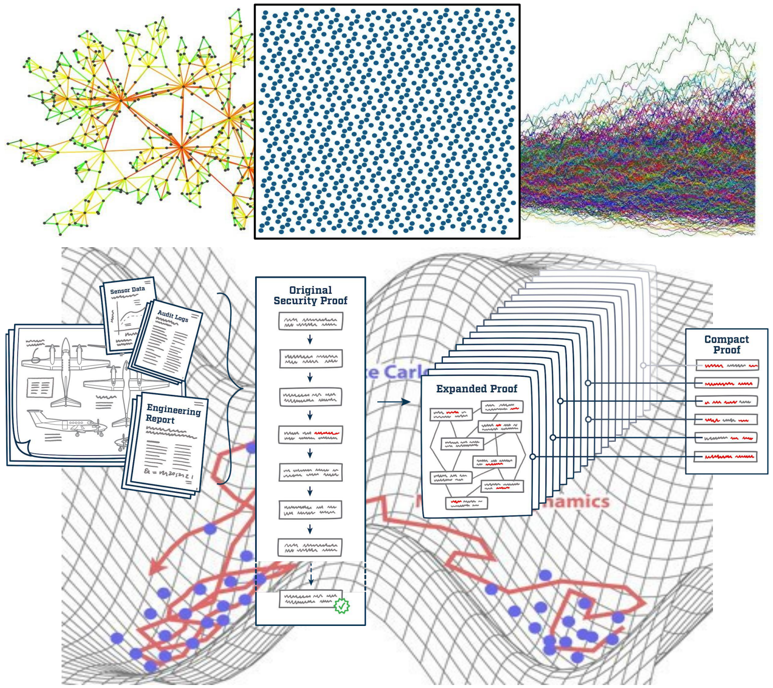
- Novel QMC framework with ZKPs
- Integrated with a formal methods pipeline
- ISTI ZKP work featured in 1663
- On-going business development on derivative capabilities

Future Directions

I. Potential applied use for LANL mission-relevant applications based on Monte Carlo codes
II. Generalize theoretical derandomization trade-off
III. Include ZKPs of ML, UQ, and QMC capabilities into a unified AI decision and optimization framework


LOS ALAMOS SCIENCE AND TECHNOLOGY MAGAZINE | AUGUST 2021
1663
Extragalactic engines
The FUTURE of reactor materials
Scanning for signs of life
Trustless yet trustworthy proofs
Lighter-than-air solids
Chaos theory,
time travel,
and the
QUANTUM
BUTTERFLY
EFFECT
Los Alamos
NATIONAL LABORATORY
August 2021 Issue

Private and Verifiable Model Evaluation using Secure Multiparty Computation and Zero-knowledge



Project Description

We extend the current capabilities of non-interactive zero-knowledge proofs to assure the integrity of Monte Carlo algorithms to enable rich forms of model evaluation such as uncertainty quantification. We also leverage formal methods to prove software correctness and prevent bugs.

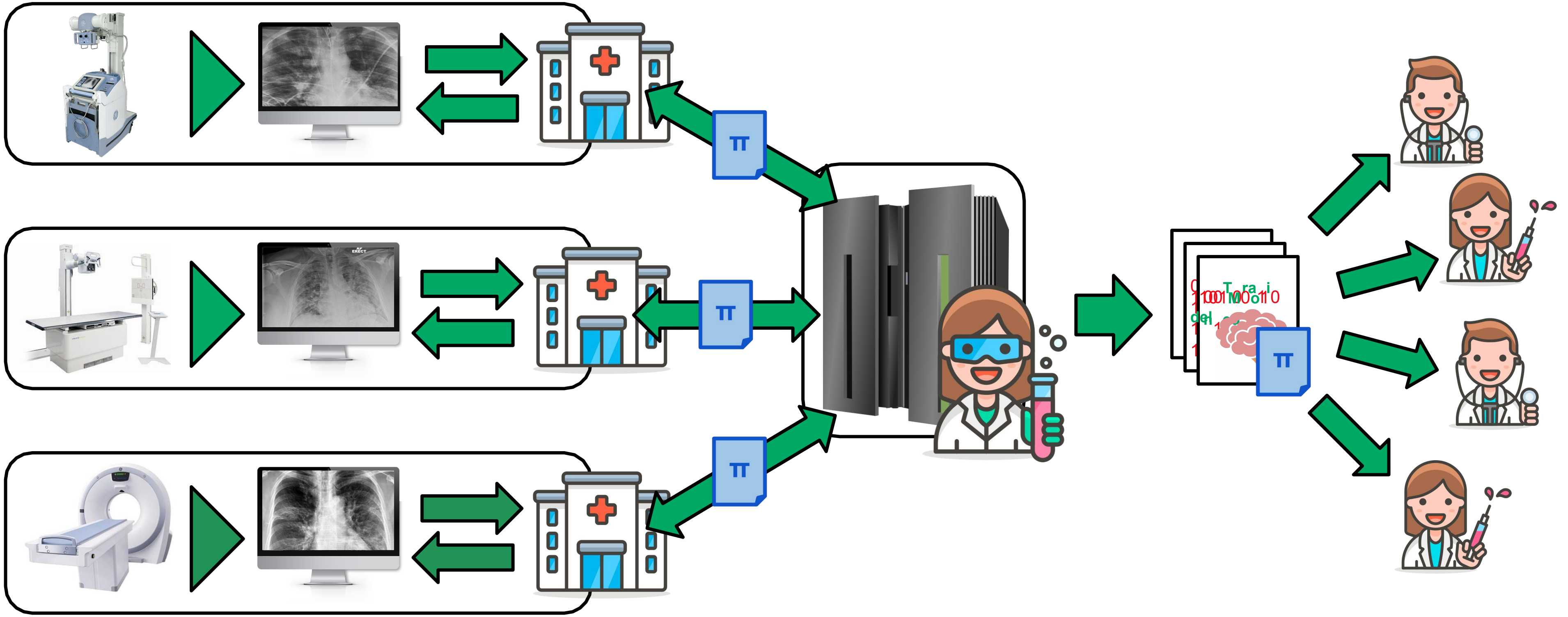
Project Outcomes

- Novel QMC framework with ZKPs
- Integrated with a formal methods pipeline
- ISTI ZKP work featured in 1663
- On-going business development on derivative capabilities

PI: Michael Dixon
Total Project Budget: $60,000
ISTI Focus Area: Computational & Data Integrity

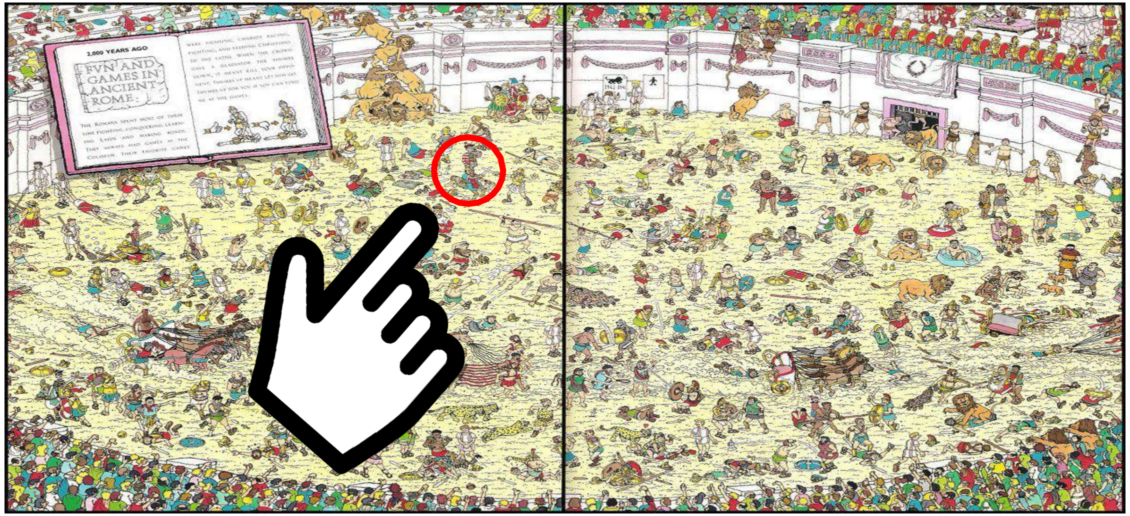
Backup

Prior Work: Privacy-Preserving and Verifiable Federated Machine Learning



Classical Proofs vs. Zero-Knowledge Proofs (II)

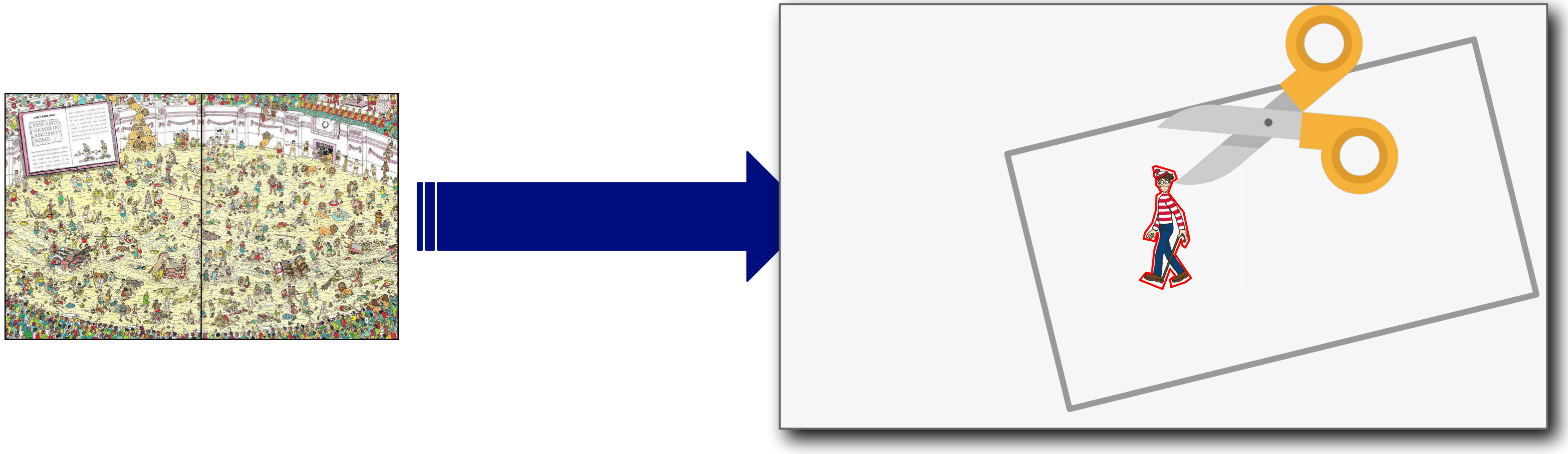
A *classical proof* for "Where's Waldo?"



Step 1: *Point to Waldo and/or draw a circle around him*
Result: This proves that you know where he and reveals his location

Classical Proofs vs. Zero-Knowledge Proofs (II)

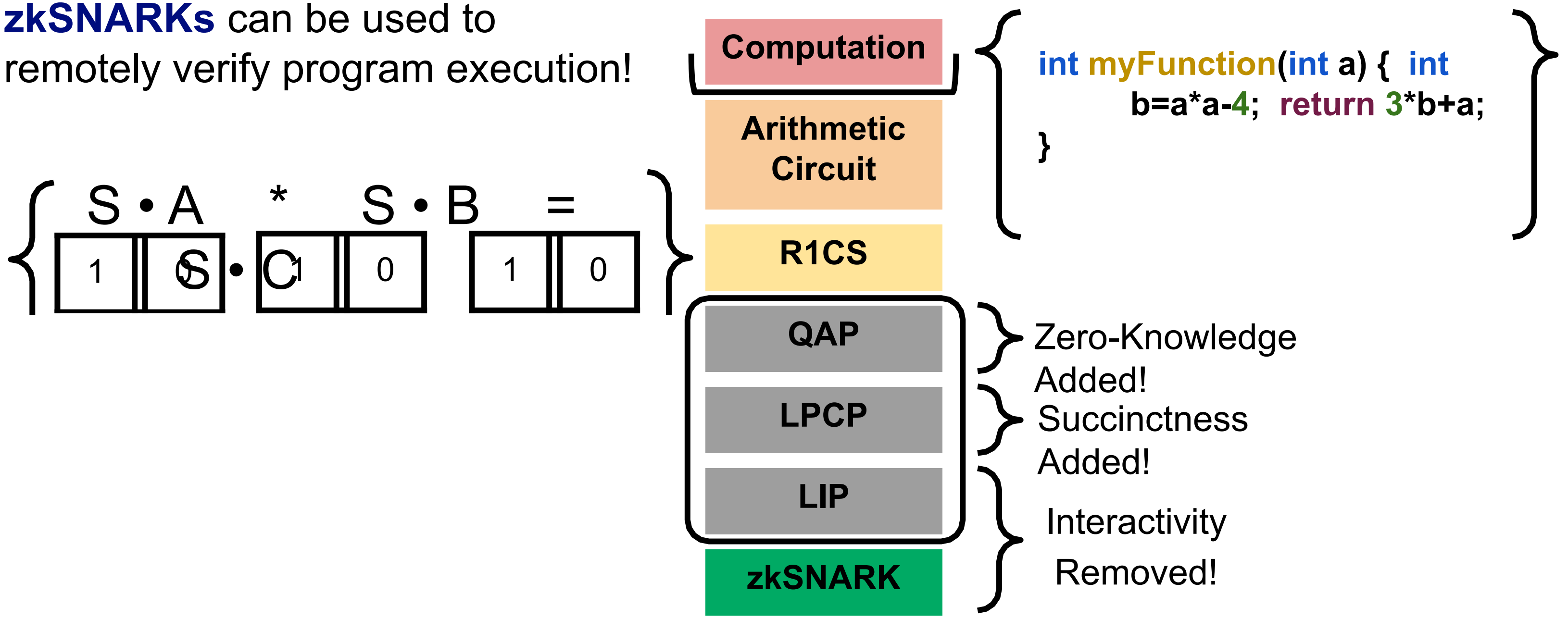
A *zero-knowledge proof* for “Where’s Waldo?”



Step 1: *Cut out a Waldo shaped hole in a large piece of paper*
Step 2: *Align both papers so that the hole is directly over the location of Waldo*
Result: This proves that you know where he is without revealing his location

zkSNARK Construction for Program Verification

zkSNARKs can be used to remotely verify program execution!



```
int myFunction(int a) {  int
        b=a*a-4;  return 3*b+a;
}
```

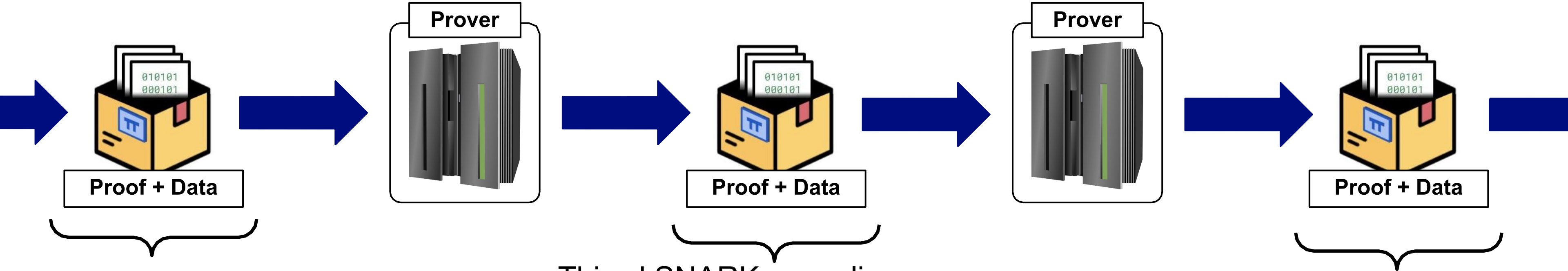
This pipeline is just one example of how to create a zkSNARK from general computation

zkSNARKs and Proof Carrying Data

Proof Carrying Data (PCD)

There is a technique in cryptography called Proof Carrying Data which we can recursively compose zkSNARKs, allowing for one to attest to an entire chain (or tree) of computations

This is particularly useful for distributed, iterative, and dynamic length computations



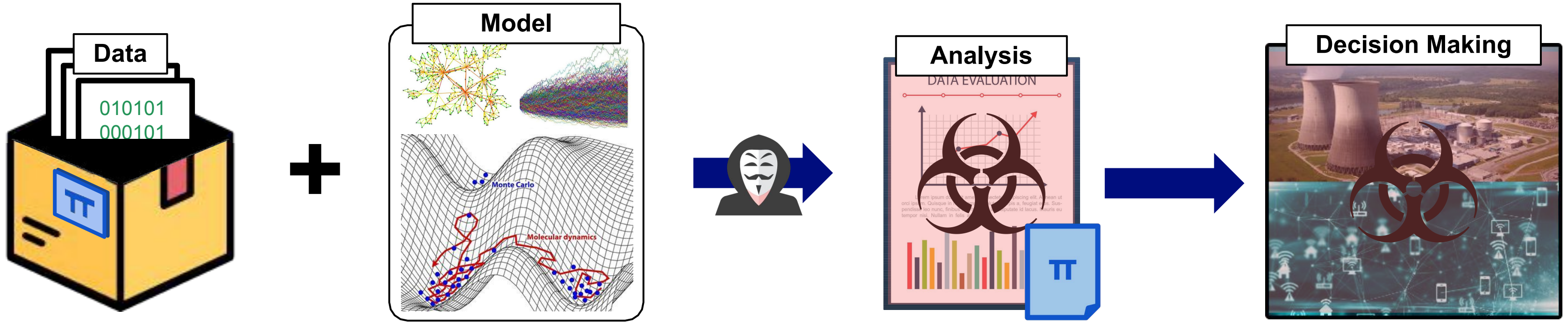
An error here would propagate through the balance of the protocol

This zkSNARK, compliance predicate, and associated witness can be verified or reused in another prover at any time

Verifying this package verifies all of the computations before it

Project Motivation | Uncertainty in Computation

Zero-Knowledge Proofs are excellent tools for remotely verifying the execution of *deterministic computation*; however, in practice, there is a need to capture computations with *uncertainties*, *estimates*, and *probabilities*.

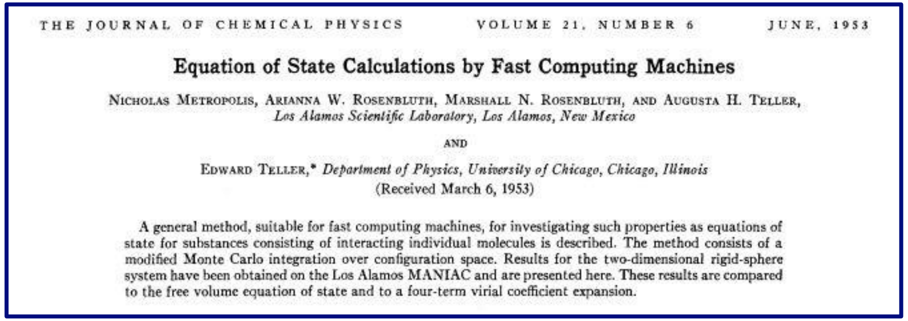


Without proper **uncertainty quantification**, we *cannot* trust computational analysis to inform our *decision making*, particularly when these computations are influenced by *malicious adversaries*!

Monte Carlo (MC) and Non-Interactive ZKPs

Monte Carlo Techniques:

A class of *fast algorithms* which trade *certainty* for *speed and randomness* developed at LANL for the ENIAC in the 1940s and 50s by Metropolis, Ulam, von Neumann, and Teller



THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

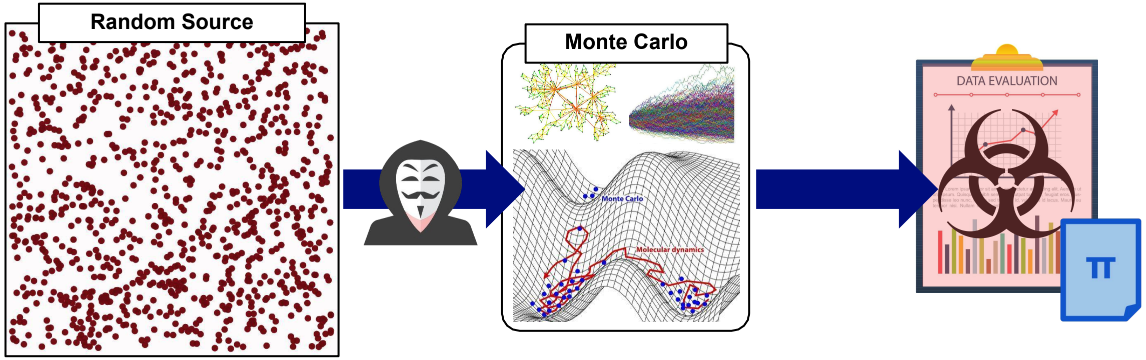
NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.



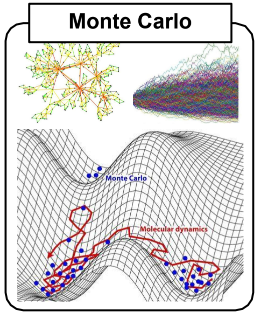
Limitations:

Adversarially chosen randomness *prevents convergence* and poisons results of Monte Carlo algorithms!

Worst case MC performance has an *unbounded error*!

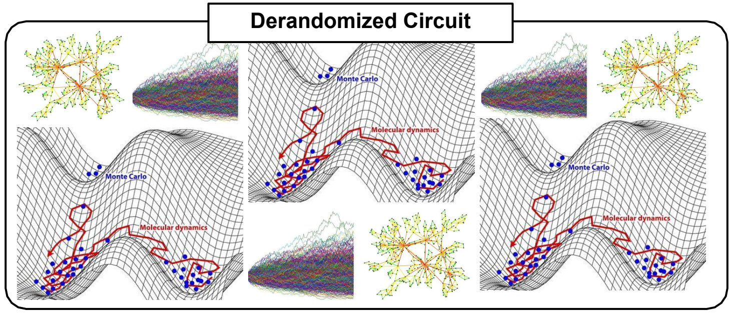
Derandomization and Complexity

Adleman's Theorem



If you can construct a polynomial-sized *randomized circuit* for a problem (with a fixed input size), then there exists a polynomial-sized *deterministic circuit* for the same problem

This is done by duplicating the original circuit several times and replacing the randomness with a deterministic "advice" string



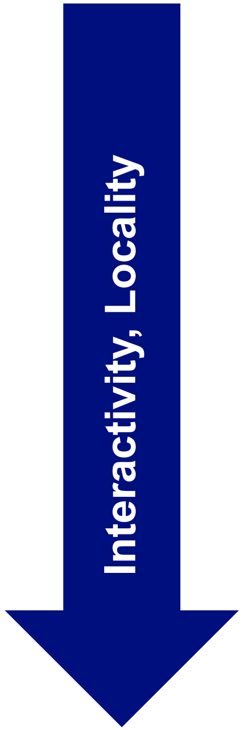
Problem I

This transformation can take an *exponential* amount of time, making it not just *impractical*, but generally *computationally intractable*

Problem II

This transformation can lead to a *polynomial increase* in the size of the circuit making it more *computationally expensive to execute*

Degrees of Scope that ZKPs Capture Properties



1. NIZK proves local property (Proof Replacement)
2. NIZK approximates local property (Prove a less-than-sound measurement; limits to amplifiability)
3. Interactive proof system required to prove local property
4. Interactive proof system required to approximate local property
5. Secure multiparty computation required to prove global property
6. Secure multiparty computation required to approximate global property
7. SMC + FHE + Global datastructure (Blockchain) required to securely evaluate global property
8. Global property cannot be securely computed by any interactive protocol without more exotic crypto primitives

Key Proofs 1 (Niederreiter, 18)

Conclusion

Applying *Quasi-Monte Carlo* allows *non-interactive zero-knowledge proofs* (like zkSNARKs) to soundly capture *randomized algorithms* run by *adversarial provers* by exploiting a *cryptographic Goldilocks zone* between fully *randomized* and fully *derandomized algorithms*

The *algorithms* and *techniques* designed at LANL for nuclear physics, when combined with *modern techniques in cryptography* and *complexity*, provide a working model for *robust uncertainty quantification*, even in interactions with *powerful* and *malicious adversaries*