FHE Compiler Using Buildit

Secure Arithmetic Scheduling using BGV and Noise Reduction Techniques

DEVA SUVEDH
CS22BTECH11016

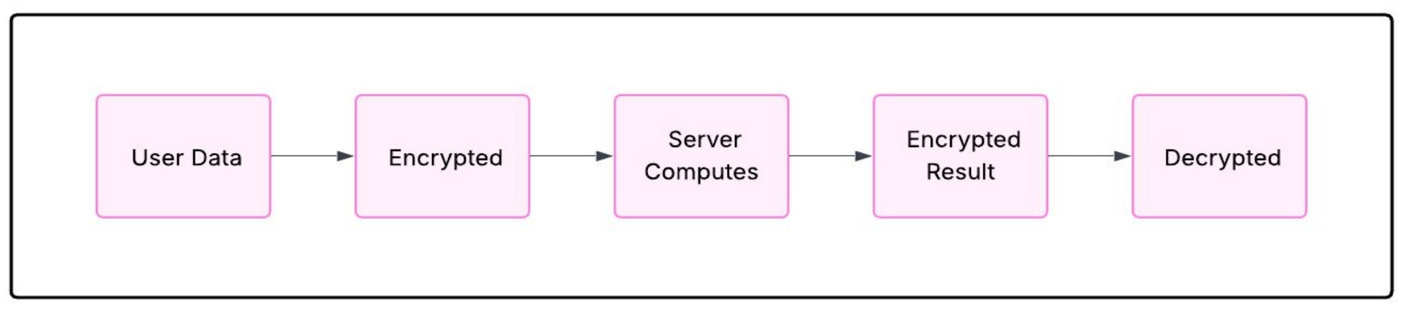
MEDIKONDA SREEKAR
CS22BTECH11037

MENTOR -RAJIV SHAILESH CHITLA

GUIDE-RAMAKRISHNA
UPADRASTA

Introduction

- Homomorphic Encryption allows computation directly on encrypted data.
- BGV scheme supports exact integer operations and logical functions.
- BuildIt separates symbolic scheduling from encrypted execution for optimization.



User Data → Encrypted → Server Computes → Encrypted Result → Decrypted

Problem Statement

- Exploring the compilation techniques for FHE and eventually applying BUILDIT technique to it and applying possible optimizations to it.

Motivation

- Enable secure computation on encrypted data using the BGV scheme.
- Optimize arithmetic expression evaluation by reducing noise growth through techniques like relinearization and NTT.
- Use BuildIt for symbolic scheduling to automate and simplify encrypted computation planning.

Expression Modeling

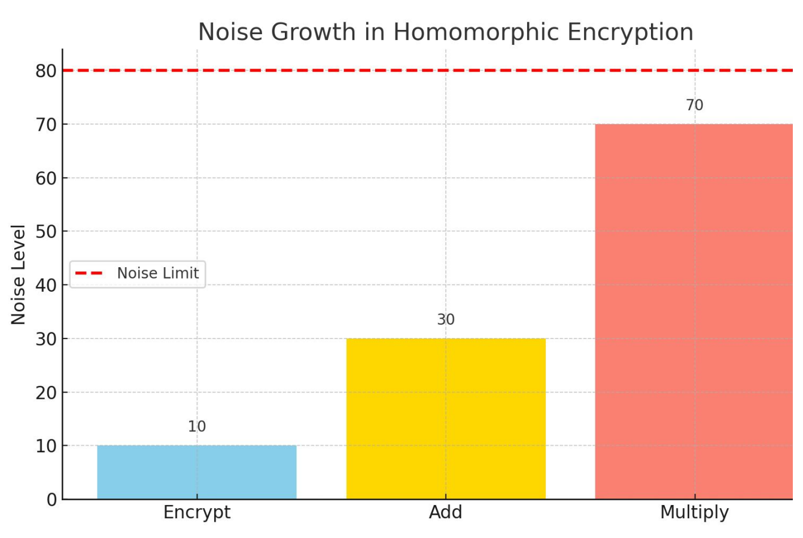
- Expressions parsed into balanced trees using postfix conversion.
- Terms stored as (coefficient, variable) pairs for stack-based evaluation.

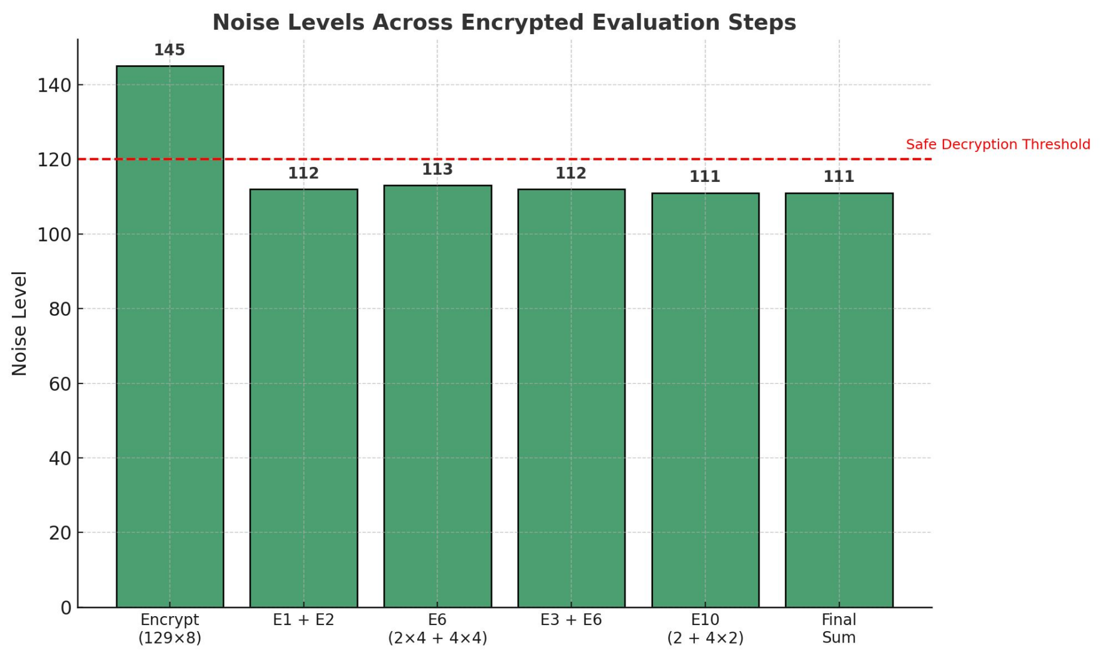
Homomorphic Evaluation (BGV Scheme)

- Operands are encrypted using Microsoft SEAL's BGV scheme
- Arithmetic is performed directly on encrypted integers
- Supports both addition and multiplication

Noise Growth & Its Effects

- Every operation increases ciphertext noise
- Excessive noise leads to decryption failure





Optimizations

- Relinearization and modulus switching
- Rotation for vectorized operations
- Minimize multiplications to reduce noise growth
- BatchEncoder enables SIMD-style parallel ops by packing many values into one ciphertext.

Mathematical Insight:

- Any c decomposed as: $c \cdot x = \left(\begin{smallmatrix}\ell\log_2 c\\ i=0\end{smallmatrix}\right) \cdot b_i\, 2 \cdot g_i \cdot x = \sum_{b_i \in 0} b_i\, (2 \cdot x)$
  - For small c/x: keep circuit shallow $13x = (8+4+1)x = 8x + 4x + x$
  - For large $c=$ too many terms ↑ $\imath v$ $127x = 64x + \ldots + 2x + x$
- For large c: too many terms ↓ noise ↑ $127x = 64x + \ldots + 2x + x$

For $log_2 c \geq 6 \Rightarrow c \equiv 64 \Leftarrow\geq 7$ terms binary sum is efficient ↓ noise ↓

Result:

For scheduled evaluations, $log_2\, c \leq 6$ optimizes decompisotion

$log_2\, c$	Terms in Sum	Efficient?
≤ 6	≤ 7	✓ Yes
> 6	> 7	✗ No

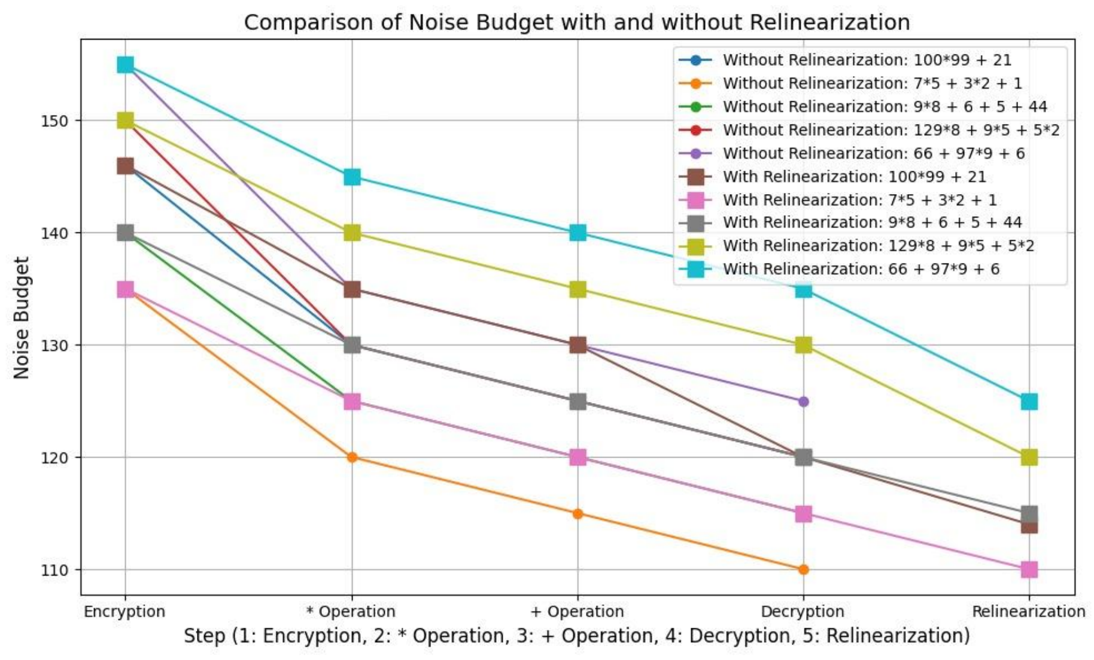
For scheduled evaluations, $log_2\, c \leq 6$ optimizes decomposition.

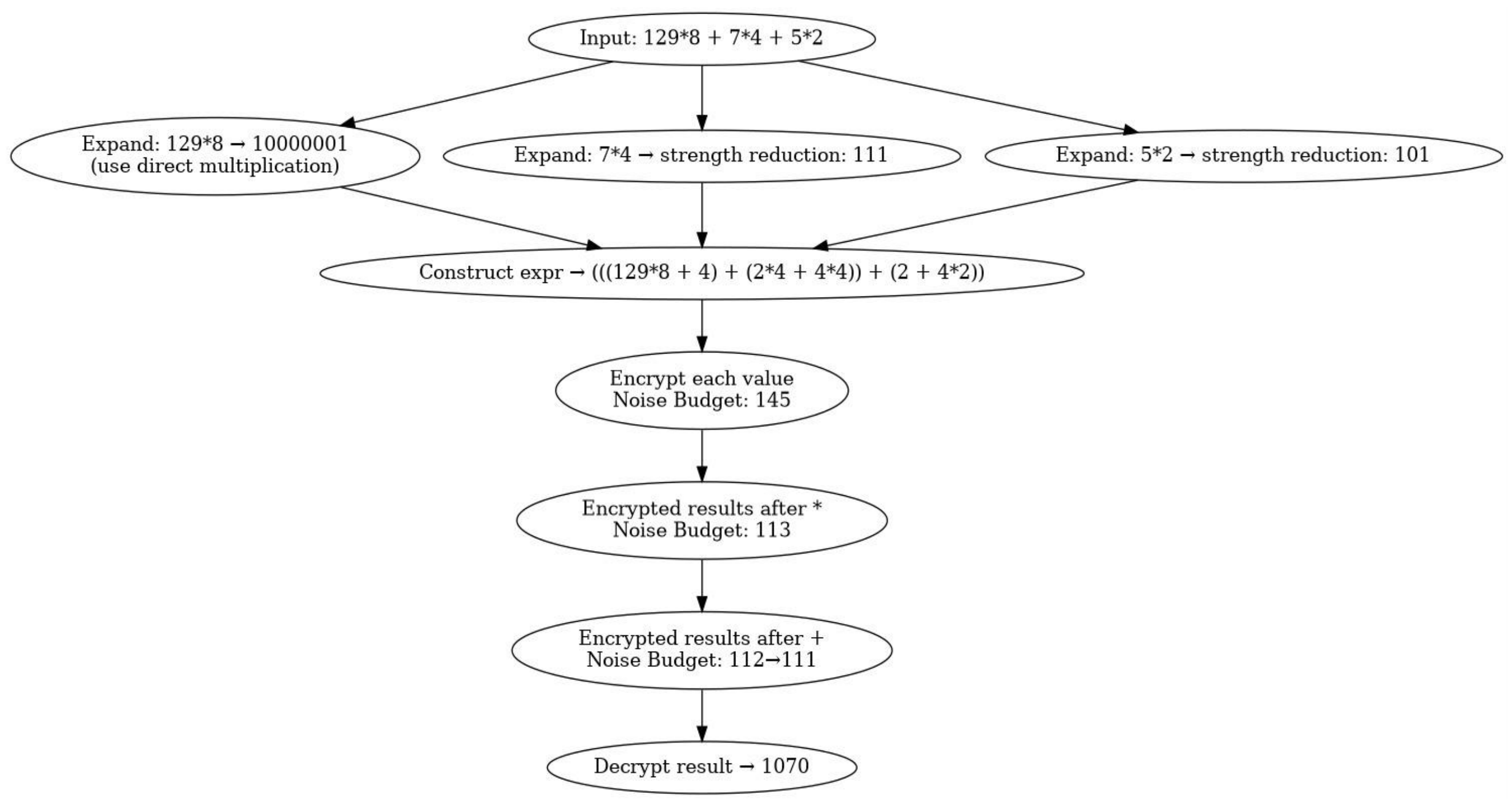
Work Done

- Implemented expression parsing and postfix conversion for structured evaluation.
- Integrated SEAL with BGV scheme to perform encrypted arithmetic operations.
- Applied batching, relinearization, and NTT to optimize encrypted computation.
- Monitored noise budget to ensure successful decryption of result.

Results

Noise with relinearization vs without relinearization





Input: 129*8 + 7*4 + 5*2

Expand: 129*8 → 10000001 (use direct multiplication)

Expand: 7*4 → strength reduction: 111

Expand: 5*2 → strength reduction: 101

Construct expr → (((129*8 + 4) + (2*4 + 4*4)) + (2 + 4*2))

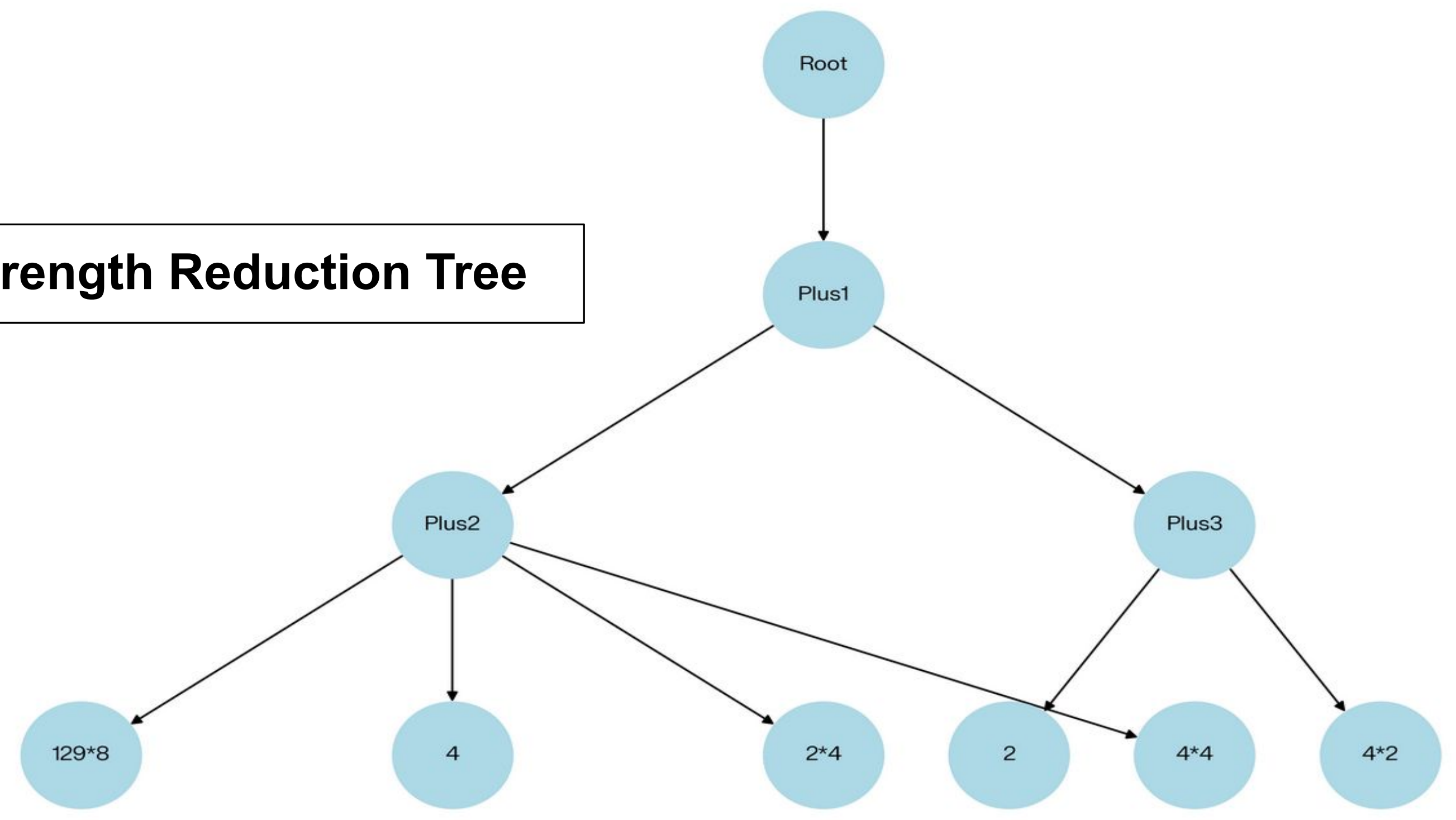
Encrypt each value Noise Budget: 145

Encrypted results after * Noise Budget: 113

Encrypted results after + Noise Budget: 112→111

Decrypt result → 1070

Input ,Output



Strength Reduction Tree

Reference

- Microsoft SEAL (Simple Encrypted Arithmetic Library)
- Gentry, C. "Fully Homomorphic Encryption Using Ideal Lattices", 2009
- Brakerski, Z., Gentry, C., & Vaikuntanathan, V."(Leveled) Fully Homomorphic Encryption without Bootstrapping."
- Buildit Framework for DSLs