KU LEUVEN

Robin Geelen and Frederik Vercauteren
COSIC, ESAT, KU Leuven, Belgium
FHE.org Conference, 8 March 2026

Better GBFV Bootstrapping and Faster Encrypted Edit Distance Computation

Better GBFV Bootstrapping

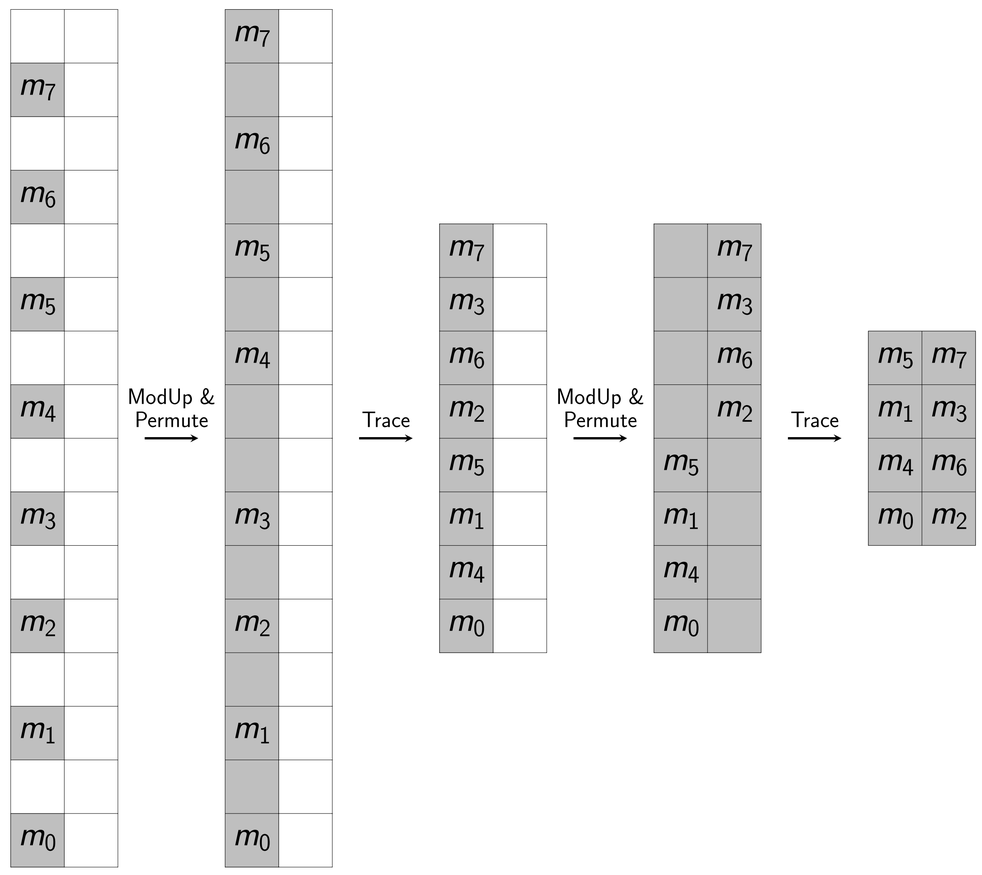
Summary

- GBFV is encoding with lower latency and noise growth than BFV
- Previous bootstrapping: perform SlotToCoeff in BFV domain
- New bootstrapping: SlotToCoeff entirely in GBFV domain
  - Crucial subroutine: convert GBFV encoding to BFV (see figure)
  - SlotToCoeff is faster or has lower noise growth
  - Less than a second to bootstrap at 128-bit security level

Bootstrapping results for 1024 slots and $p = 2^{16} + 1$:

Bootstrapping algorithm		Baseline	Ours	Ours	Ours
Number of stages s		2	2	3	4
Noise (bits)	Initial	317	317	317	317
	Noisy expansion	111	81	96	110
	Digit removal	82	82	82	82
	Remaining	124	154	139	125
Execution time (sec)	Noisy expansion	0.95	1.16	0.61	0.47
	Digit removal	0.34	0.34	0.34	0.34
	Total	1.29	1.50	0.95	0.81

Homomorphically convert GBFV encoding to BFV encoding:



Faster Encrypted Edit Distance Computation

Summary

- Comparison of two strings in the encrypted domain
  - Edit distance: number of edits to transform one string into another
  - We count insertions, deletions and substitutions
- Applications:
  - Privacy-preserving detection of DNA mutations
  - Detection of typos in bank account number
  - Spelling correction

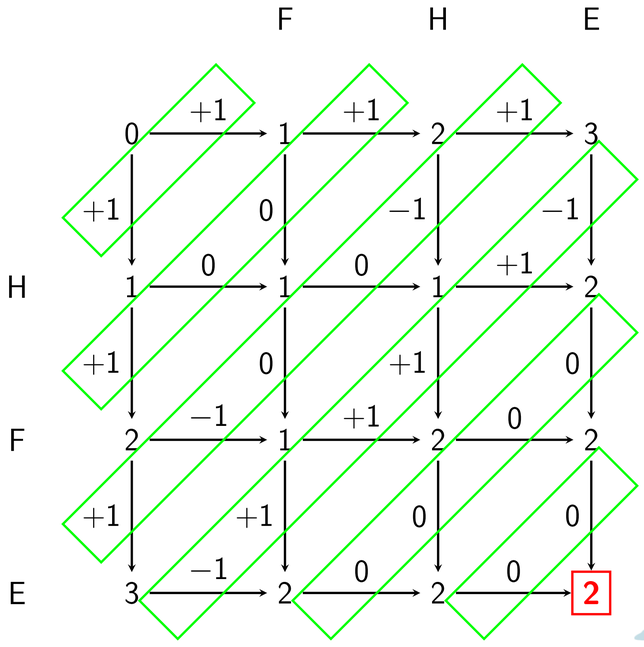
Algorithm Homomorphic edit distance

Require: $\mathsf{ct}_a, \mathsf{ct}_b$ that encrypt input strings **a** and **b** of length m
Ensure: $\mathsf{ct}_{\mathsf{acc}}$ that encrypts Levenshtein edit distance in its first slot

function EditDistance($\mathsf{ct}_a, \mathsf{ct}_b$)
  $\mathsf{ct}_H, \mathsf{ct}_V \leftarrow \mathsf{Enc}([1, 0, \ldots, 0])$ ▷ Length-m vectors
  $\mathsf{ct}_{\mathsf{acc}} = \mathsf{Enc}([m, 0, \ldots, 0])$ ▷ Accumulator for result
  $C = (\mathsf{ct}_1, \ldots, \mathsf{ct}_M) \leftarrow \mathsf{PrecompDeltas}(\mathsf{ct}_a, \mathsf{ct}_b)$
  for $k \leftarrow 2$ to m **do**
    $\mathsf{ct}_{-\delta} \leftarrow \mathsf{ExtractMinusDelta}(C, k)$
    $\mathsf{ct}_{\mathsf{min}} \leftarrow \mathsf{Eval}(g, [\mathsf{ct}_{-\delta}, \mathsf{ct}_H, \mathsf{ct}_V])$ ▷ Minimum using polynomial g
    $\mathsf{ct}_T \leftarrow [1, 1, \ldots, 0] \ominus \mathsf{ct}_V \oplus \mathsf{ct}_{\mathsf{min}}$ ▷ First $k-1$ entries are 1's
    $\mathsf{ct}_V \leftarrow [1, 1, \ldots, 0] \ominus \mathsf{ct}_H \oplus \mathsf{ct}_{\mathsf{min}}$ ▷ First k entries are 1's
    $\mathsf{ct}_H \leftarrow \mathsf{ShiftRight}(\mathsf{ct}_T) \oplus [1, 0, \ldots, 0]$ ▷ Set $H[1] = 1$
  end for
  for $k \leftarrow m+1$ to $2m$ **do**
    $\mathsf{ct}_{-\delta} \leftarrow \mathsf{ExtractMinusDelta}(C, k)$
    $\mathsf{ct}_{\mathsf{min}} \leftarrow \mathsf{Eval}(g, [\mathsf{ct}_{-\delta}, \mathsf{ct}_H, \mathsf{ct}_V])$ ▷ Minimum using polynomial g
    $\mathsf{ct}_T \leftarrow [1, 1, \ldots, 0] \ominus \mathsf{ct}_H \oplus \mathsf{ct}_{\mathsf{min}}$ ▷ First $2m-k+1$ entries are 1's
    $\mathsf{ct}_H \leftarrow [1, 1, \ldots, 0] \ominus \mathsf{ct}_V \oplus \mathsf{ct}_{\mathsf{min}}$ ▷ First $2m-k$ entries are 1's
    $\mathsf{ct}_V \leftarrow \mathsf{ShiftLeft}(\mathsf{ct}_T)$
    $\mathsf{ct}_{\mathsf{acc}} \leftarrow \mathsf{ct}_{\mathsf{acc}} \oplus \mathsf{ct}_T$
  end for
  return $\mathsf{ct}_{\mathsf{acc}} \otimes [1, 0, \ldots, 0]$ ▷ First entry contains actual distance
end function

Timings (sec) for edit distance and comparison to Leuvenshtein (LVS):

Algorithm	LVS	LVS	LVS	Ours	Ours	Ours
String size m	8	100	256	8	128	256
Batch size ℓ/m	–	–	–	512	32	16
Equality tests	–	–	–	5.10	82	167
Bootstrappings	–	–	–	11.1	193	387
Other operations	–	–	–	1.50	26	57
Total latency	2.83	439	2903	17.7	301	611
Amortized time	2.83	439	2903	0.036	9.41	38.2
Speedup	–	–	–	79×	–	76×



Full paper: https://ia.cr/2025/1104