NanoGram: Garbled RAM with $\tilde{O}(\log N)$ Overhead

Andrew Park  
CMU

Wei-Kai Lin  
Northeastern

Elaine Shi  
CMU
Garbled Circuits [Yao82,Yao86]
Garbled Circuits [Yao82, Yao86]
Garbled Circuits [Yao82,Yao86]
Garbled Circuits [Yao82, Yao86]

Garbler

Non Interactive

Large RAM-to-circuit conversion

Evaluator
Garbled RAM [LO13]
Garbled RAM [LO13]
Garbled RAM [LO13]
Garbled RAM [LO13]

Garbler

Input: \{\{i\}\}

Output: \{\{x_i\}\}

Evaluator
Garbled RAM [LO13]

Garbler

Input: \{{i}\}

Output: \{{x_i}\}

Evaluator

Goal: Access takes time sublinear in size of RAM
Garbled RAM Landscape
Garbled RAM Landscape

[LO13, GLO15, HL20]

$\text{poly}(\lambda, \log n)$
Garbled RAM Landscape

[LO13, GLO15, HL20]

poly(λ, log n)

Impractical
Garbled RAM Landscape

[LO13, GLO15, HL20]
\(\text{poly}(\lambda, \log n)\)
Impractical

Epigram
[HKO21]
\(O(\lambda \log^2 n)\)
Garbled RAM Landscape

[LO13, GLO15, HL20]

poly(\(\lambda, \log n\))

Impractical

Epigram
[HKO21]

O(\(\lambda \log^2 n\))

Lower Bound
Garbled RAM Landscape

[LO13, GLO15, HL20]  
\( \text{poly}(\lambda, \log n) \)  
\( \text{Impractical} \)

Epigram  
[HKO21]  
\( O(\lambda \log^2 n) \)

[GO’87, WCS’15]  
\( \Omega(\log n) \)  
\( \text{Lower Bound} \)
Garbled RAM Landscape

[LO13, GLO15, HL20]  
poly(λ, log n)  
Impractical

Epigram  
[HKO21]  
O(λ log^2 n)

[GO’87, WCS’15]  
Ω(log n)  
Interactive

Lower Bound
This Work
This Work

Can we have a (non-interactive) garbled RAM scheme whose asymptotical performance is competitive to the interactive state-of-the-art?
This Work

[LO13, GLO15, HL20]
\[\text{poly}(\lambda, \log n)\]
Impractical

Epigram
[HKO21]
\[O(\lambda \log^2 n)\]

Lower Bound

[GO’87, WCS’15]
\[\Omega(\log n)\]
Interactive
This Work

[LO13, GLO15, HL20]  
$\text{poly}(\lambda, \log n)$  
Impractical

Epigram  
[HKO21]  
$O(\lambda \log^2 n)$

NanoGRAM  
$\tilde{O}(\lambda \log N)$

[GO’87, WCS’15]  
$\Omega(\log n)$  
Interactive

Lower Bound
This Work

Near optimal dependence on N

[LO13, GLO15, HL20]

poly(\(\lambda, \log n\))

Impractical

Epigram

[HKO21]

\(O(\lambda \log^2 n)\)

Interactive

NanoGRAM

\(\tilde{O}(\lambda \log N)\)

[GO’87, WCS’15]

\(\Omega(\log n)\)

Lower Bound
Outline

The Language Translation Problem

Strawman: Garbled ORAM Tree

Our Techniques
Outline

The Language Translation Problem

Strawman: Garbled ORAM Tree

Our Techniques
The Language Translation Problem

Each garbled circuit speaks a time-dependent language
The Language Translation Problem

Each garbled circuit speaks a time-dependent language

$$\text{Encode}(sk, data, L)$$
The Language Translation Problem

T = 0
T = 1
T = 2
The Language Translation Problem

Each gate expects input garbled under time-dependent language
The Language Translation Problem

I want $x_i$ garbled under lang for $t = 1$
The Language Translation Problem

G: doesn’t know i in advance

E: doesn’t know t in advance

T = 0
T = 1
T = 2
The Language Translation Problem

T = 0

G: doesn’t know i in advance

T = 1

E: doesn’t know t in advance

T = 2
The Language Translation Problem

G: doesn’t know $i$ in advance

E: doesn’t know $t$ in advance

Goal: Translate $x_i$ under language $L_t$
Outline

The Language Translation Problem

Strawman: Garbled ORAM Tree

Our Techniques
Strawman: Garbled ORAM Tree
Strawman: Garbled ORAM Tree

Each node has its own local clock

= Garbled Data Structure
Strawman: Garbled ORAM Tree

$$\{\text{labels}\}, \ L_t$$
Strawman: Garbled ORAM Tree

\[ \text{O(log N) labels, one to read from each bucket} \]
Strawman: Garbled ORAM Tree

\{\text{labels}\}, L_t

GOAL: Route $L_t$ to $x_i$ on path

$x_i$
Strawman: Garbled ORAM Tree

{{labels}}, $L_t$

GOAL: Route $L_t$ to $x_i$ on path

Garbled Switch:
Gadget for Language Translation
Garbled Switch [HKO21]
Garbled Switch [HKO21]

Every node has its local clock

T = 5

T = 3

T = 2
Garbled Switch [HKO21]

Every node has its local clock

When invoked, local time increments
Garbled Switch [HKO21]

Every node has its local clock

When invoked, local time increments

Garbled message must speak the local time dependent language of node
Garbled Stacks of Labels [ZRE15]
{{data, leaf_addr}}
{{data, leaf_addr}}

{{L₁}} ←

L₂
L₃
L₄

L₁ →

{{0}}

L₂
L₃
L₄
\{\{data, leaf\_addr\}\}
\{\{\text{data, leaf_addr}\}\} \\
Cost: \(O(\log N), N = \text{Stack Size}\)
Outline

The Language Translation Problem

Construction of [HKO21]

Our Techniques
Two Reasons of Inefficiency
Two Reasons of Inefficiency

Large Switches (i.e. Root Node) = Large Number of Access
Two Reasons of Inefficiency

Large Switches (i.e. Root Node) = Large Number of Access

Passing large payload length (O(log n) labels, \( \lambda \) bits long)
Our Contributions

Large Switches (i.e. Root Node) = Large Number of Access

Passing large payload length (O(log n) labels, λ bits long)
Our Contributions

- Large Switches (i.e. Root Node) = Large Number of Access

- Polylog sized buckets with dynamic finalization (Bucket ORAM)

- Passing large payload length ($O(\log n)$ labels, $\lambda$ bits long)
Our Contributions

- Large Switches (i.e. Root Node) = Large Number of Access
- Polylog sized buckets with dynamic finalization (Bucket ORAM)
- Passing large payload length (O(log n) labels, \(\lambda\) bits long)
- Passing Single Label Using XOR Trick (see paper)
Trick: Break down switches into smaller size

Each node at level $i$ has $\frac{T}{(B \cdot 2^i)}$ copies of GSwitch + GBkt
Trick: Break down switches into smaller size

Each node at level \(i\) has \(\frac{T}{(B \cdot 2^i)}\) copies of GSwitch + GBkt
Trick: Break down switches into smaller size

Each node at level $i$ has $T/(B \cdot 2^i)$ copies of GSwitch + GBkt

Bucket are of size $O(\log N)$
Trick: Break down switches into smaller size

Each node at level $i$ has $T/(B \cdot 2^i)$ copies of GSwitch + GBkt
Trick: Break down switches into smaller size

Each node at level i has \( \frac{T}{(B \cdot 2^i)} \) copies of GSwitch + GBkt

\[
T = \begin{array}{cccccccc}
0 & B & 2B & 3B & 4B & 5B & 6B & 7B \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

= Empty

= Full
Trick: Break down switches into smaller size

Each node at level i has $T/(B \cdot 2^i)$ copies of GSwitch + GBkt

$T = 0, B, 2B, 3B, 4B, 5B, 6B, 7B$

$= Empty$

$= Full$
Trick: Break down switches into smaller size

Each node at level $i$ has $\frac{T}{(B \cdot 2^i)}$ copies of $\text{GSwitch} + \text{GBkt}

Break up $O(N)$-sized switch into $O(\log N)$-sized
Rebuilding the Garbled Buckets
Rebuilding the Garbled Buckets
Rebuilding the Garbled Buckets

Rebuild every \( B \) time steps

\( B = \text{Stash Size} \)
Rebuilding the Garbled Buckets

Rebuild every $B$ time steps

$B = \text{Stash Size}$
Issue: Dynamic Rebuild
Accesses are unknown at garbling time
Issue: Dynamic Rebuild

Accesses are unknown at garbling time
Issue: Dynamic Rebuild

Accesses are unknown at garbling time

Rebuild
Issue: Dynamic Rebuild

Accesses are unknown at garbling time

Local clocks of children have advanced to unknown dynamic value
Solution: Dynamic Finalization

Equip Garbled data structures with Finalize routine
Solution: Dynamic Finalization

Equip Garbled data structures with Finalize routine

Rebuild
Solution: Dynamic Finalization
Equip Garbled data structures with Finalize routine

Rebuild + Finalize
Solution: Dynamic Finalization

Equip Garbled data structures with Finalize routine

Rebuild + Finalize
Additional Optimizations (See Paper)

Avoiding $\lambda$ factor blowup when garbling

Modular framework for garbled algorithm composition

Practical Optimizations
Concrete Performance

![Graph showing performance comparison between Nanogram, Epigram, and Linear scan](image-url)
THANK YOU!