Ligetron: Zero knowledge on Steroids

Ruihan Wang (Ligero Inc.)
Carmit Hazay (Ligero Inc. / Bar-Ilan University)
Muthu Venkitasubramaniam (Ligero Inc.)

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

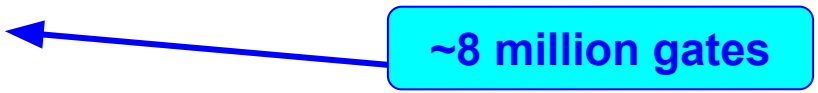
Let's do a demo now!

NP Statement: Prove 1024 copies of the statement that Edit-distance(abc,abcde) < 5

1. Prover —> https://www.yellkey.com/film
2. Hit "Download Proof!"
3. Verifier —> https://www.yellkey.com/myself
4. Load proof.data
5. Voila!

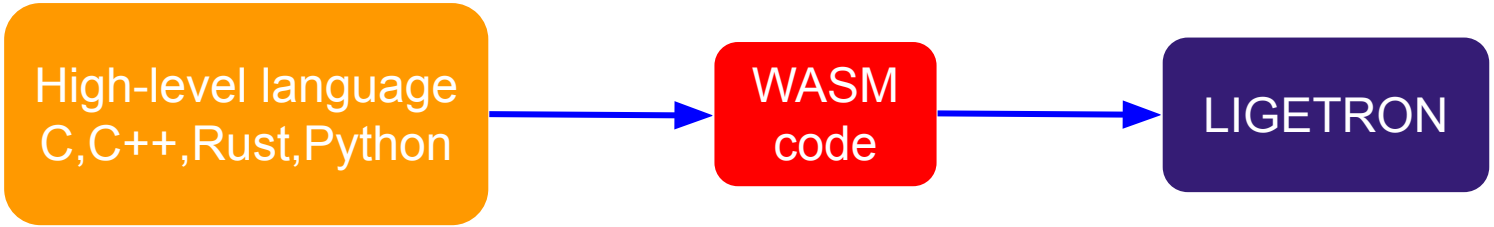
Let's do a demo now!

NP Statement: Prove 1024 copies of the statement that Edit-distance(abc,abcde) < 5



1. https://ligero-inc.com/demo/mobile/prover.html?edit.wasm&abc&abcde
2. Hit "Download Proof!"
3. https://ligero-inc.com/demo/mobile/prover.html?edit.wasm&xxx&xxxxx
4. Load proof.data
5. Voila!

Ligetron Compilation Toolchain



Step 1) Compile high-level languages to WASM

Step 2) Prover and Verifier interpret WASM

Key Idea: Want prover to use roughly same amount of memory as natively evaluating the NP statement

```cpp
extern "C" {

  inline int min(int a, int b) {return a <= b ? a : b;}

  inline int oblivious_if(bool cond, int t, int f) {
      int mask = static_cast<int>((1ULL << 33) - cond);
      return (mask & t) | (~mask & f);
  }

  int minDistance(const char* word1, const char* word2, const int m, const int n) {
      int pre;
      int cur[n + 1];
      for (int j = 0; j <= n; j++) {
        cur[j] = j;
      }
      for (int i = 1; i <= m; i++) {
        pre = cur[0];
        cur[0] = i;
        for (int j = 1; j <= n; j++) {
          int temp = cur[j];
          bool cond = word1[i - 1] == word2[j - 1];
          cur[j] = oblivious_if(cond,
            pre,
            min(pre, min(cur[j - 1], cur[j])) + 1);
          pre = temp;
        }
      }
      return cur[n];
  }

  bool statement(const char *word1, const char* word2, const int m, const int n) {
      return minDistance(word1, word2, m, n) < 5;
  }
```

C code for Edit distance

LLVM compiler

```
(module
  (type (;0;) (func (param i32 i32 i32 i32) (result i32)))
  (import "env" "__linear_memory" (memory (;0;) 0))
  (import "env" "__stack_pointer" (global (;0;) (mut i32)))
  (func $minDistance (type 0) (param i32 i32 i32 i32) (result i32)
    (local i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32)
    global.get 0
    local.tee 4
    drop
    i32.const 0
    local.set 5
    local.get 4
    local.get 3
    i32.const 2
    i32.shl
    i32.const 19
    i32.add
    i32.const -16
    i32.and
    i32.sub
    local.tee 6
    drop
    block  ;; label = @1
      local.get 3
      i32.const 0
      i32.lt_s
      br_if 0 (;@1;)
```

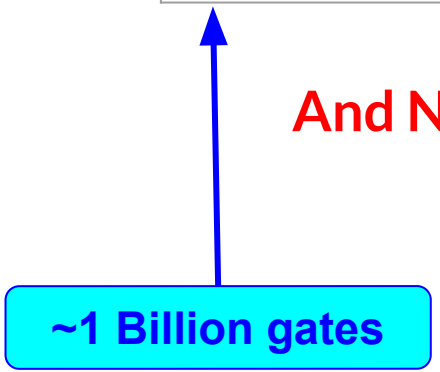
WASM code for Edit distance

Key Insights

- Operational semantics of WASM with Ligero [AHIV17]
  => time/space efficiency
- WASM has only a few set of instructions to compile (<200)
- Numerous compilers from high-level languages to WASM (C,C++,Rust,Python, Solidity,etc)
- Clean sandbox (no I/Os, no system calls, no malloc*)

Performance on the browser (128-bit)

String length	Cons./state	Batch size	Prover speed (end to end)	Verifier Speed (end to end)
30	334820	1024	1.22 µs/g	14.1 ns/g
40	589676	1024	1.21 µs/g	12.9 ns/g
50	915684	1024	1.17 µs/g	11.8 ns/g



And Non-interactive!

And Sublinear!!

And plausibly PQ secure!!

On Steroids
Performance on c6i.8x (32 vcpu,64 GB RAM)

String length	Cons. per state.	Batch size	Prover speed	Verifier Speed	Prover Memory	Verifier Memory	Proof Length
10	40804	16384	34.40 ns/g	2.39 ns/g	130MB	63MB	28MB
15	88282	16384	34.57 ns/g	2.28 ns/g	167MB	107MB	59MB
20	152012	16384	34.93 ns/g	2.41 ns/g	217MB	169MB	100MB

2.5 Billion gates

And Non-interactive!

And Sublinear!!

And plausibly PQ secure!!

Do you want to do another demo now?

NP Statement: Prove 1024 copies of the statement that Count_occur(abc,abcdefabcghiabc) > 2

1. Prover —> https://www.yellkey.com/relate
2. Hit “Download Proof!”
3. Verifier —> https://www.yellkey.com/another
4. Load proof.data
5. Voila!

```
extern "C" {

  inline int oblivious_if(bool cond, int t, int f) {
      int mask = static_cast<int>((1ULL << 33) - cond);
      return (mask & t) | (~mask & f);
  }

  int countFreq(const char *pat, const char *txt, const int M, const int N) {
      int res = 0;

      /* A loop to slide pat[] one by one */
      for (int i = 0; i <= N - M; i++) {
          /* For current index i, check for
             pattern match */
          int count = 0;
          for (int j = 0; j < M; j++) {
              count += oblivious_if(txt[i + j] == pat[j], 1, 0);
          }

          // if pat[0...M-1] = txt[i, i+1, ...i+M-1]
          res += oblivious_if(count == M, 1, 0);
      }
      return res;
  }

  bool statement(const char *pat, const char *txt, const int M, const int N) {
      return countFreq(pat, txt, M, N) > 2;
  }

}
```