
Zero-Knowledge Interval Proofs and Other Cryptographic Protocols Involving Intervals

Berry Schoenmakers
Coding & Crypto group
TU Eindhoven

Outline

$$[0, L) = \{0, 1, \dots, L-1\}$$

1. Zero-knowledge Interval Proofs
 - Prove that committed integer $x \in [0, L)$
 2. Secure Modulo Reduction
 - Determine $x \bmod L$, for given secret x .
 3. Random number generation
 - Bounded numbers: uniform $x \in [0, L)$
 - From random bits, uniform in $\{0, 1\}$
-

1

Zero-knowledge Interval Proofs

- Discrete log setting:
 - $\langle g \rangle = \{1, g, g^2, \dots, g^{q-1}\}$ for prime q , $g^q = 1$
 - $h \in \langle g \rangle$ with $\log_g h$ unknown to anyone
 - Problem:
 - given a Pedersen commitment $C = g^x h^r$, where $x \in [0, L)$ and r is random.
 - prove knowledge of x and r with $x \in [0, L)$.
-

Easy cases

- Given $C = g^x h^r$ with $x \in [0, L)$.
- $L=1$: prove “ $C = h^r$ ” ($x=0$)
 - *Schnorr proof*
- $L=2$: prove “ $C = h^r$ ($x=0$) or $C = g h^r$ ($x=1$)”
 - OR-composition of 2 Schnorr proofs
- $L=2^n$: commit to bits of x ... n times $L=2$ case
 - Using $2n$ Schnorr proofs in total

Arbitrary L : $2^{n-1} < L \leq 2^n$

- Use **intersection** $[0, L) = [0, 2^n) \cap [L-2^n, L)$
 - AND-composition of two length- 2^n intervals
 - Or, use **union** $[0, L) = [0, 2^{n-1}) \cup [L-2^{n-1}, L)$
 - OR-composition of two length- 2^{n-1} intervals
 - Either way: about $2 \cdot 2n = 4n$ Schnorr proofs
 - Optimizations, e.g., if $L = 2^{n-1} + R$, for small R
 - Best case: $R=1$, then reduced to $2n$ Schnorr proofs
 - Worst case: still $4n$ Schnorr proofs
-
- How to do really better?

General approach: case $L = a + b$

- For $x \in [0, L)$, we have:

$$x \in [0, a) \text{ or } x - a \in [0, b)$$

- Then we prove recursively:

“C commits in $[0, a)$ ” OR “ C/g^a commits in $[0, b)$ ”

General approach: case $L = a b$

- For $x \in [0, L)$, we can write (uniquely):

$$x = y b + z \quad \text{with } y \in [0, a), z \in [0, b)$$

- Split $C = g^x h^r$ into commitment D and E:

$$D = g^y h^s \quad \text{with } y = [x/b], \quad \text{random } s$$

$$E = g^z h^t \quad \text{with } z = x \bmod b, \quad \text{random } t$$

subject to $r = s b + t \pmod{q}$.

Then $C = D^b E$, and we prove recursively:

“D commits in $[0, a)$ ” AND “E commits in $[0, b)$ ”

Leads to interesting combinatorics

- Recall: for $L=2^n$ we need $2n$ Schnorr proofs
 - For general L of bit length n we get close to “optimal” of $2n$ Schnorr proofs
 - Exact complexity related to so-called **integer complexity of L**
-

Sequence A005245, $W(L)$ is equal to the complexity of L :

1, 2, 3, 4, 5, 5, 6, 6, 6, 7, 8, 7, 8, 8, 8, 8, 9, 8, 9, 9, 9, 10, 11, 9, 10, 10, 9, 10,
 11, 10, 11, 10, 11, 11, 11, 10, 11, 11, 11, 11, 12, 11, 12, 12, 11, 12, 13, 11, 12,
 12, 12, 12, 13, 11, 12, 12, 12, 13, 14, 12, 13, 13, 12, 13, 13, 14, 13, 14, 13,

The **complexity** of a natural number n is defined as the minimal number of 1's required to write n as a formula composed of 1's, +'s, and *'s (and parentheses).

$$1 = 1$$

$$2 = 1 + 1$$

$$3 = 1 + 1 + 1$$

$$4 = 1 + 1 + 1 + 1$$

$$5 = 1 + 1 + 1 + 1 + 1$$

$$6 = (1 + 1) * (1 + 1 + 1)$$

$$7 = (1 + 1) * (1 + 1 + 1) + 1$$

- 1 corresponds to a Schnorr proof
- + corresponds to OR-composition
- * corresponds to AND-composition

- Introduction
- 'Small' Interval Proofs
- 'Large' Interval Proofs
- Comparison

Bounds for $W(L)$

Upper bound

$$W(L) \leq 2 \log_2 L + \text{'Hamming weight of 'L} \leq 3 \log_2 L$$

Lower bound

$$W(L) \geq 3 \log_3 L$$

So,

$$1.89 \log_2 L \leq W(L) \leq 3 \log_2 L$$

But detailed analysis of sequence $W(L)$ is **notoriously hard**. For example, it is **not** known if $W(2^k) = 2k$ for all k .

But ... FC'2002 paper

- Better and easier approach possible.
 - Helps to split in not-necessarily disjoint intervals.
 - $L=2m$: $[0, L) = [0, m) \cup [m, L)$
 - $L=2m+1$: $[0, L) = [0, m+1) \cup [m, L)$
- Given $x \in [0, L)$, let $b \in \{0, 1\}$ indicate in which half x lies (breaking ties arbitrarily):
 - Let B denote commitment to b
 - Prove that
 - B commits in $\{0, 1\}$, and
 - (recursively) C/B^m commits to $[0, (L+1)/2)$
- Leads to about $2 \log_2 L$ Schnorr proofs

Optimal solution:

- Assume $L=3m$ for simplicity:
 - $L=3m$: $[0, L) = [0, m) \cup [m, 2m) \cup [2m, L)$
- Given $x \in [0, L)$, let $t \in \{0, 1, 2\}$ indicate in which part x lies (breaking ties arbitrarily):
 - Let B denote commitment to t
 - Prove that
 - B commits in $\{0, 1, 2\}$, and
 - (recursively) C/B^m commits in $[0, L/3)$
- Can be extended to work for any L
- Leads to $3 \log_3 L \approx 1.89 \log_2 L$ Schnorr proofs
Better than binary method **even for $L=2^n$**

Optimal solution:

- Assume $L=3m$ for simplicity:
 - $L=3m$: $[0, L) = [0, m) \cup [m, 2m) \cup [2m, L)$
- Given $x \in [0, L)$, let $t \in \{0, 1, 2\}$ indicate in which part x lies (breaking ties arbitrarily):

- Let B denote commitment to t

- P

-

-

- Can

- Lea

- Bet

Optimality:

$f(x) = x \log_x L$ is minimal

for $x = e = 2.71828\dots$ proofs

hence take $x = 3$.

2

(with Bart Mennink & Jorge Guajardo)

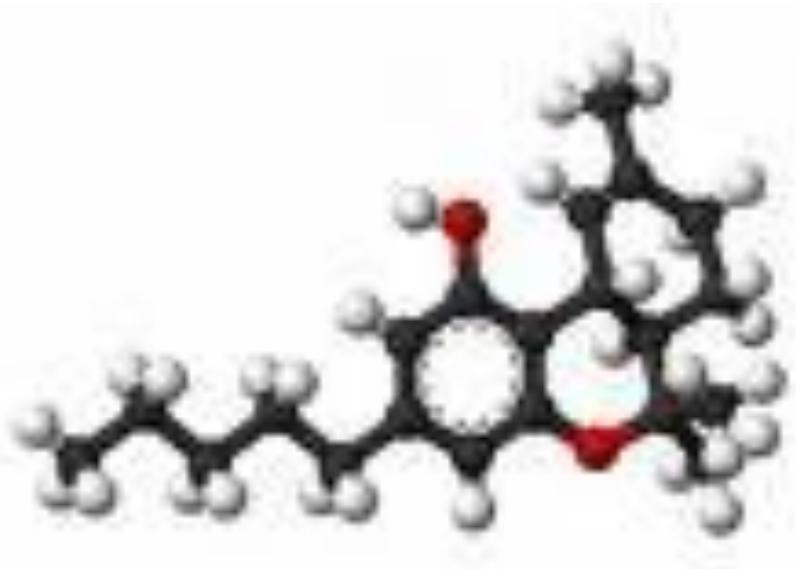
Threshold Homomorphic Cryptosystems

=

Public K

+ **Homomorphic** Enc

+ **Threshold** Decryption



Tetra

Hydro

Cannabinol

=

Public K

+ **Homomorphic** Enc

+ **Threshold** Decryption



THCs

- ElGamal, Paillier, ...
- Homomorphic properties
 - Additive homomorphic: $E[x] * E[y] = E[x+y]$
 - Multiplication by a known constant: $E[x]^c = E[c x]$
 - Random re-randomization: $E[x] * E[0] = E[x]$
- Note: $E[x]$ stands for probabilistic encryption, so $E[x] = E[x, r]$ for some random r .

Modulo reduction: $x \bmod L$ (secret x)

- Efficient gates for addition and multiplication:
 - $E[x], E[y] \rightarrow E[x+y], E[x*y]$
- Efficient gate for least-significant bit [ST06]
 - $E[x] \rightarrow E[x \bmod 2]$
- Here, efficient gate for general L :
 - $E[x] \rightarrow E[x \bmod L]$
- Many applications (e.g. for securely computing statistics such as mean and variance)
- Toy example: $E[x] \rightarrow \text{decide } x < L$
 - Compute $E[x - (x \bmod L)]$ and perform $(=0)$ -test.

Protocol for $E[x] \rightarrow E[x \bmod L]$

- Given $E[x]$
- Jointly generate random $E[r]$ with $0 \leq r < L$
 - Bitwise $E[r_0], \dots, E[r_{n-1}]$
- Jointly generate random $E[s] = E[\sum_i s_i]$
- Threshold decrypt $E[x-r+Ls]$
- Compute $y = (x-r+Ls) \bmod L = (x+r) \bmod L$
- Comparison: $E[c] = E[L-1-y < r]$
- Correction: $E[y+r]/E[c]^L = E[x \bmod L]$

Also solves integer division

- $x = (x \text{ div } L) L + x \text{ mod } L$
 - $E[x \text{ div } L] = (E[x]/E[x \text{ mod } L])^{1/L}$
 - Technical assumption: x is sufficiently small to prevent wrap-around
 - E.g., for Paillier encryption with a 1024-bit modulus, x must be at most a 900-bit number.
-

3

(with Andrey Sidorenko)

Generate secret, uniform $x \in [0,L)$

- Assume random bit gate is available to jointly generate $E[b], b \in_R \{0,1\}$
 - How to generate jointly $E[x], x \in_R [0,L)$?
-

$L = 2^n$ is the easy case

- Write $x \in [0, L)$ binary:

$$x = \sum_{i=0..n-1} x_i 2^i, \quad x_i \in \{0, 1\}$$

- Generate random bits $E[x_i]$
- Put $E[x] = \prod_{i=0..n-1} E[x_i]^{2^i}$,

Random numbers in $[0, L)$, $2^{n-1} < L < 2^n$

- Given a source of (uniform) random bits.
- Two **folklore** algorithms for generating $x \in [0, L)$.

Alg.1: **repeat** $x \in_R \{0,1\}^n$ **until** $x < L$; **return** x

Alg.2: $x \in_R \{0,1\}^{n+k}$; **return** $x \bmod L$

■ Properties:

- Alg.1: perfectly **uniform**; but **wastes up to n bits on average** (worst case $L = 2^{n-1} + 1$); Las Vegas algorithm
- Alg.2: statistical distance **$\Delta < 1/2^k$** ; **wastes k bits exactly**

Our algorithm

- Generate random $x \in [0, L)$ bit by bit, starting from the most significant bit, comparing with most significant bits of $L' = L - 1$.
- Algorithm: let x_i be next random bit
 - if $x_i > L'_i$, start all over “too large”
 - if $x_i = L'_i$, continue with next bit “unsure”
 - if $x_i < L'_i$, complete x with random bits and stop “home free”
- Randomness complexity: n bits plus some waste.
- Question: what's the waste?



coin flip

Our algorithm

- Generate random $x \in [0, L)$ bit by bit, starting from the sign

- Algorithm

Recursive version

rand(L):

- if $L=1$ then return 0 else

- repeat

- $x = 2 \cdot \text{rand}(\lfloor (L+1)/2 \rfloor) + \{0,1\}$

- until $x < L$;

- return x



coin flip

ge"
e"
ree"

ste.

Analysis of randomness complexity

- First computing the exact probability distribution and then the expected value is cumbersome
- We determine expected value E **directly!**

- Example: $S = \sum_{i=0..∞} r^i$

$$S = \sum_{i=0..∞} r^i = 1 + \sum_{i=0..∞} r^{i+1} = 1 + r S, \quad \text{so } S = 1/(1-r)$$

- Example: $T = \sum_{i=0..∞} i r^i$

$$T = \sum_{i=0..∞} (1+i) r^{i+1} = r S + r T, \quad \text{so } T = r/(1-r)^2$$

- By conditioning on the right event, this leads to:

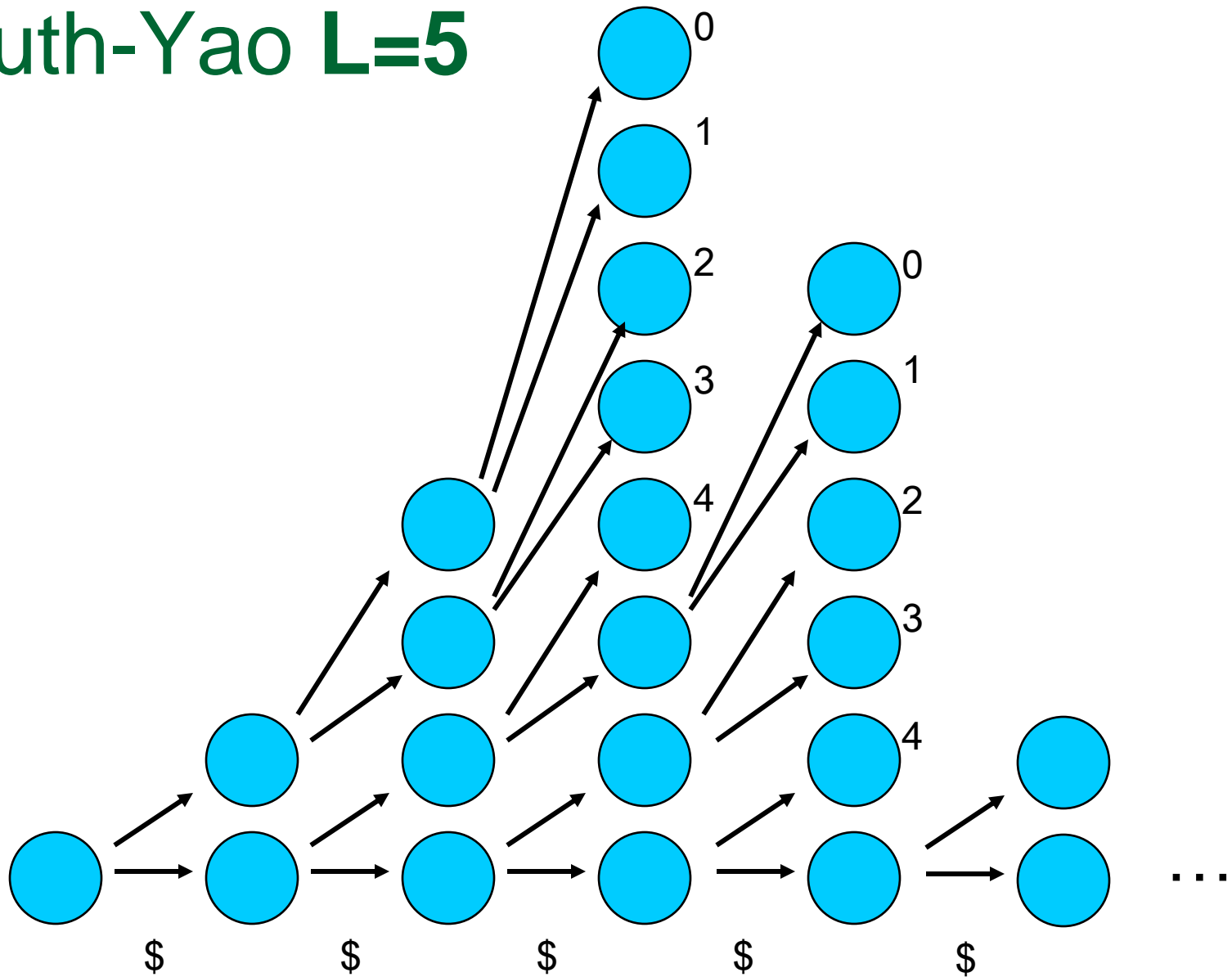
$$E = n + 2^n/L \sum_{i=2..n} (1-(L-1)_{n-i}) i 2^{-i} < n + 3$$

- So, waste is bounded by a small constant!
- Averaged over all L , waste is approx. 1.1 random bits

Knuth-Yao 1976

- Minimize randomness complexity
 - Average wasted random bits for $x \in [0, L)$:
 - ≈ 0.58 random bits averaged over all L
 - < 1 random bits for worst case L
 - Actually, **any probability distribution** can be generated from random bits, with **< 2 bits wasted on average** (beyond entropy).
-

Knuth-Yao $L=5$



Context of secure multiparty computation

- In context of secure multiparty computation:
 - Generating “encrypted” random bits is **expensive**.
 - But, also comparing “encrypted” bits, arithmetic with “encrypted” bits, etc.
- Our algorithm strikes a better balance than Knuth-Yao’s minimal waste algorithm (depending on the setting)
 - cheaper to make our algorithm **oblivious**

“Data independent execution paths”

Various approaches (1/2)

- Naïve:

- each party generates random number in $[0, L)$
- securely add these numbers (bitwise)
- reduce sum modulo L

- Knuth-Yao

- handles bitwise represented integer x , say, using operations like “ $x < L$ ” and “ $x = x - L$ ”
 - computational overhead is large
-

Various approaches (2/2)

- Our protocol:

```
 $r := 0; i := n$   
while  $i > 0$  do  
     $x_i := \text{RAN2}()$   
     $s := \text{MULT}(1 - r, x_i)$   
     $r := r + (1 - r - s) L'_i$   
     $h := \text{REVEAL}(s (1 - L'_i))$   
    if  $h = 1$  then  $i := n$  else  $i := i - 1$   
end while  
return  $x_1, \dots, x_n$ 
```

Note: values in **red** are encrypted or in shared form

Performance comparison (optimized for computational complexity)

| Protocol | Computational complexity | Round complexity |
|----------------------|----------------------------------|------------------|
| Naive | $2n(t + 2 \log_2 t)$ MULTs | $3n \log_2 t$ |
| Generate-and-compare | $4n$ MULTs | $2n$ |
| Generate-and-reduce | $4nk$ MULTs | $2nk$ |
| Knuth-Yao | $6n$ MULTs | $3n$ |
| Bit-by-bit | $n + 3$ MULTs $n + 3$ REVEALS | $n + 3$ |

Conclusion

- We can avoid performance degradations when going from $L=2^n$ to arbitrary $2^{n-1} < L \leq 2^n$ for these interval problems:
 - Zero-knowledge proof for $x \in [0, L)$
 - Secure mod reduction $E[x] \rightarrow E[x \bmod L]$
 - Secure random $E[x], x \in_R [0, L)$
- General research goal:
 - simple protocols yet optimal performance

Author's address

Berry Schoenmakers

Coding and Crypto group
Dept. of Math. and CS
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
Netherlands

berry@win.tue.nl
<http://www.win.tue.nl/~berry/>
