KU LEUVEN

One-Hot Conversion: Towards Faster Table-based A2B Conversion

Jan-Pieter D'Anvers

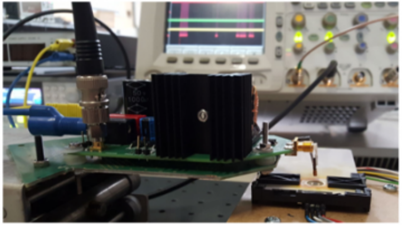
April 24, 2023

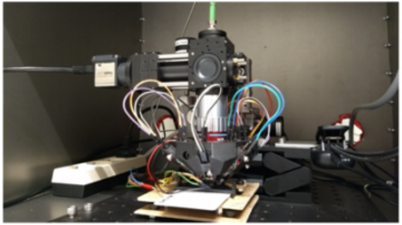
Outline

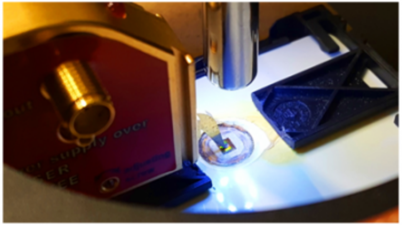
1. Side-Channel protection
2. One-hot conversion
3. Scaling up
4. One-bit-output functions
5. Results

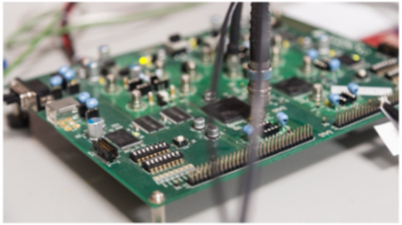
KU LEUVEN

Computer Security and Industrial Cryptography (COSIC)



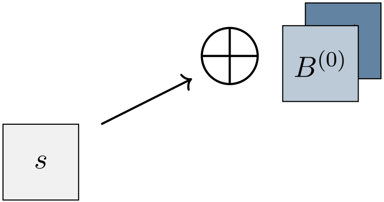




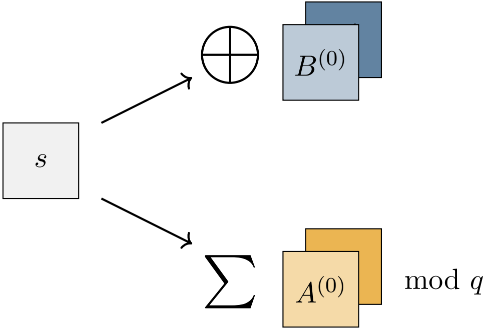


Masking

Masking

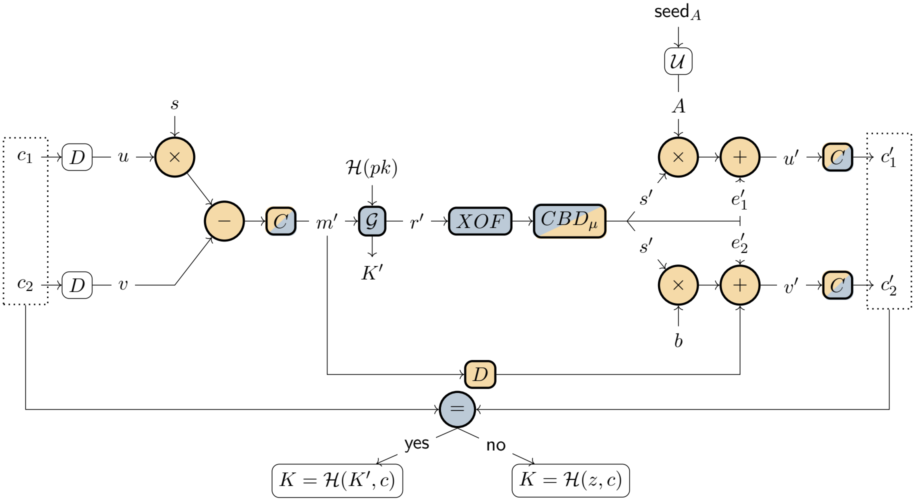


Masking



KU LEUVEN

Masking Kyber



Conversions needed

- Need conversions from arithmetic domain to Boolean domain (A2B)

Conversions needed

- Need conversions from arithmetic domain to Boolean domain (A2B)
- First-order vs. Higher-order

Existing conversion techniques

Circuit based [Gou01, CGV14]

- Write down circuit
- Replace gates w/ masked equivalent

Table-based [CT03, CGMZ21]

- Make (masked) table
- Shuffle table for each input shares
- Final lookup with last share

Existing conversion techniques

Circuit based [Gou01, CGV14]

- Write down circuit
- Replace gates w/ masked equivalent

- Scales relatively well to higher-order masking

Table-based [CT03, CGMZ21]

- Make (masked) table
- Shuffle table for each input shares
- Final lookup with last share

- Efficient in first-order
- Very inefficient in higher-order

Outline

KU LEUVEN

One-hot intermediate representation

- Improvement of table-based methods
- One-hot encoding (instead of table)

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

represents 3

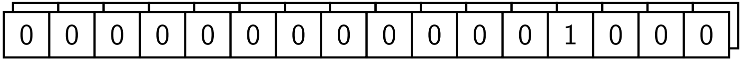
One-hot intermediate representation

- Improvement of table-based methods
- One-hot encoding (instead of table)

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

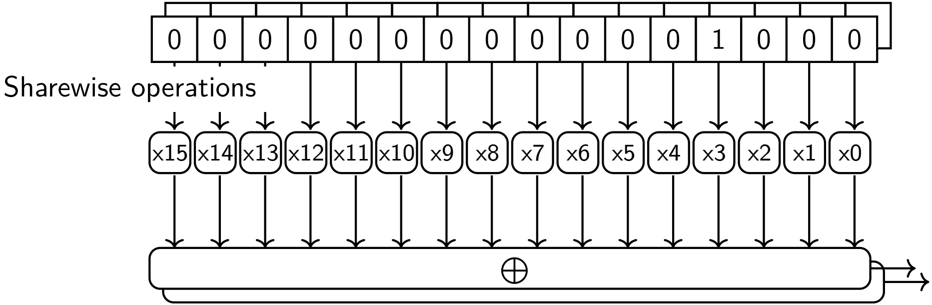
represents 3

- Boolean masked



One-hot to Boolean

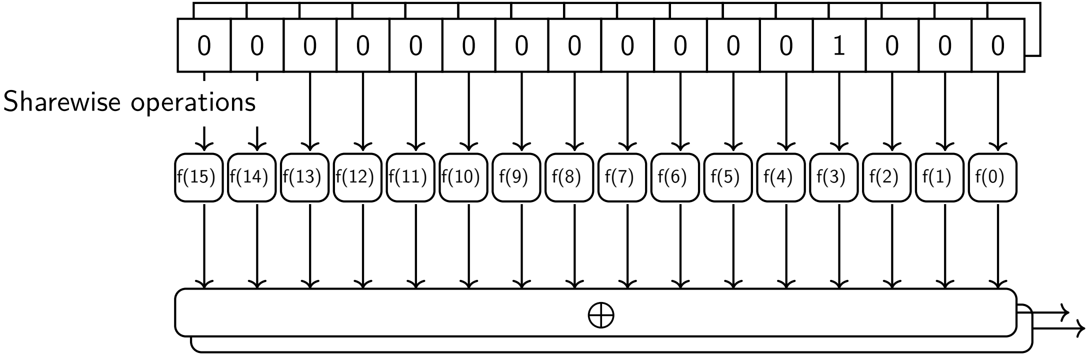
- Convert from one-hot encoding to Boolean domain



- All operations are sharewise!
- The paper describes how to implement this operation more efficiently

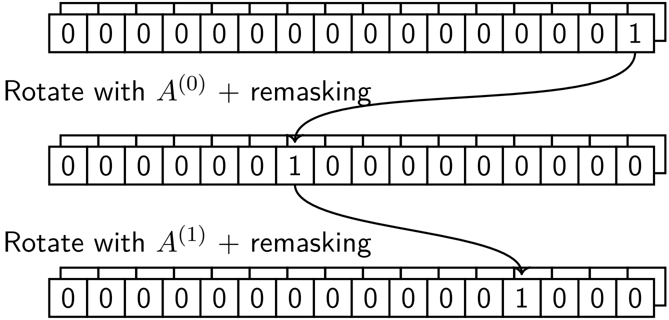
One-hot to Boolean

- We can even apply any function $f()$

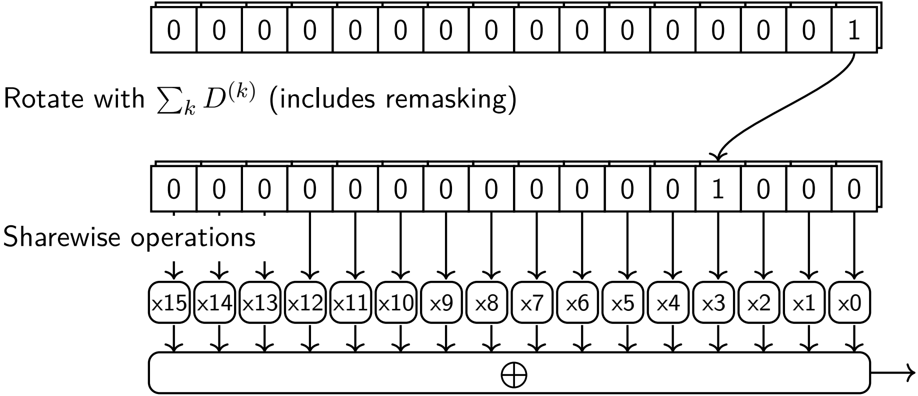


Arithmetic to one-hot

- Use 1-bit table-based method [CGMZ21]
- Adding an arithmetic share = rotating the encoding
- Example $s = 3$, arithmetically shared in $A^{(0)} = 10, A^{(1)} = 9, q = 16$

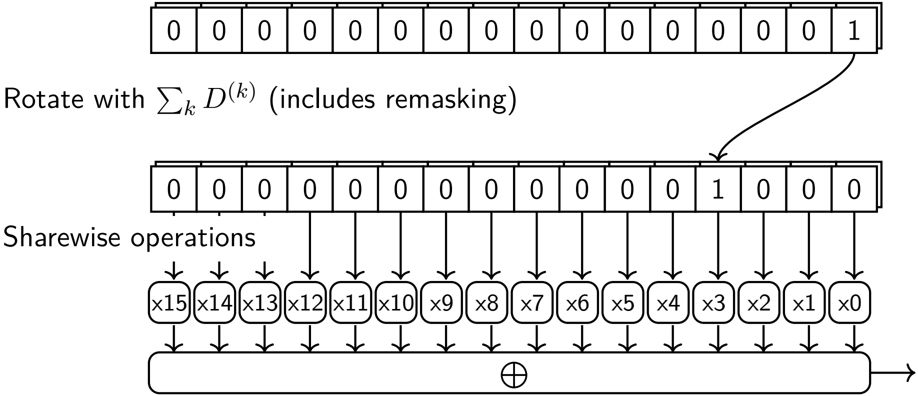


Arithmetic to Boolean



KU LEUVEN

Arithmetic to Boolean



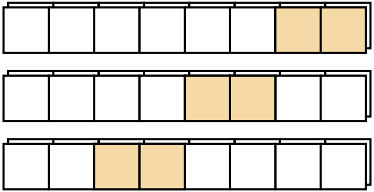
- Does not scale well

Outline

KU LEUVEN

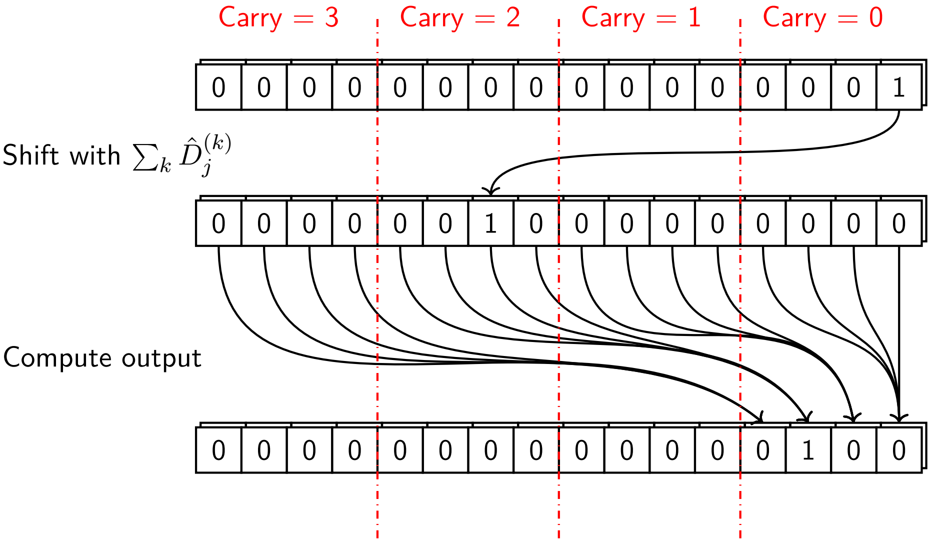
Scaling up

- Divide the input arithmetic share into chuncks of n bits
- Process each chunk iteratively



- Need to take care of carries

Carry propagation



Scaling A2B

- Three building blocks:
  - Arithmetic to one-hot
  - One-hot to Boolean
  - Carry propagation

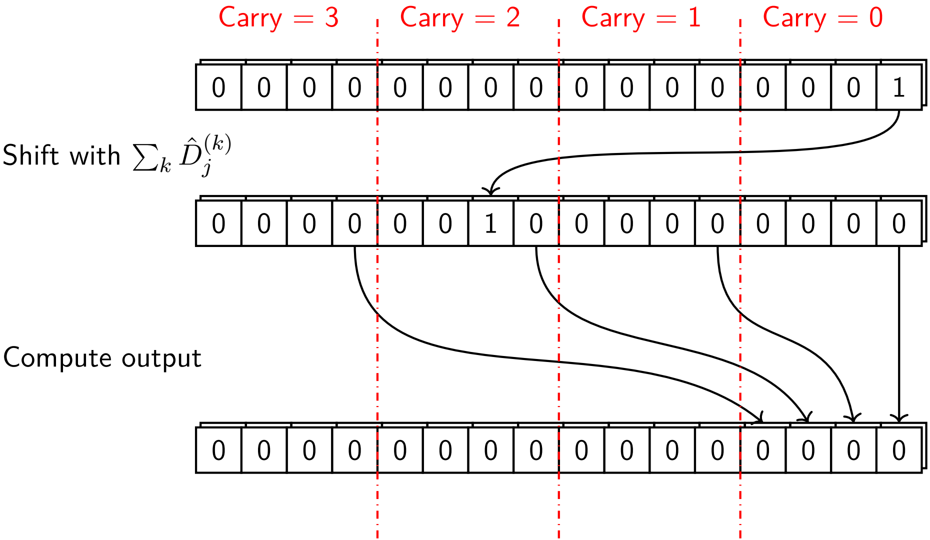
Outline

KU LEUVEN

One-bit output

- One-hot to Boolean part can be ignored for specific one-bit functions
- Notably possible for typical PQ functions:
  - MSB extraction
  - Ciphertext validation

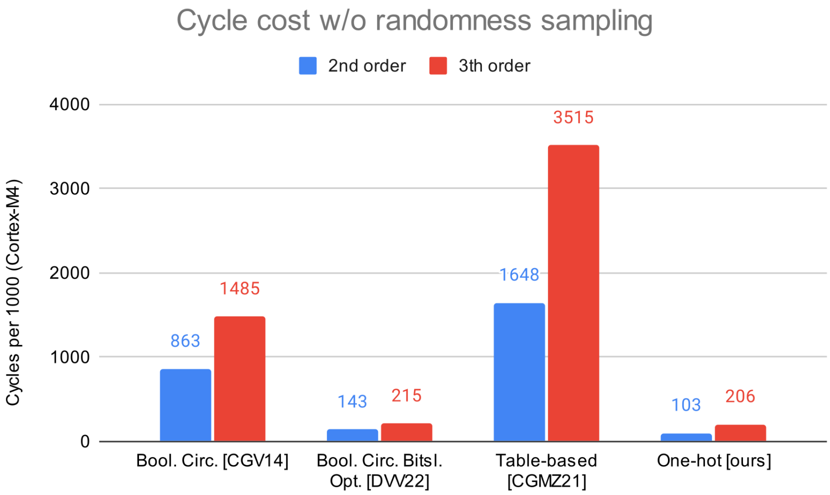
Check if masked value is zero



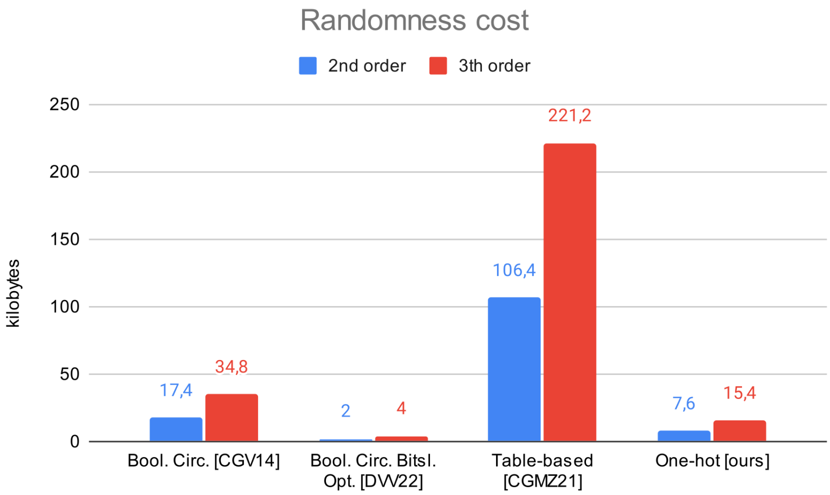
KU LEUVEN

Outline

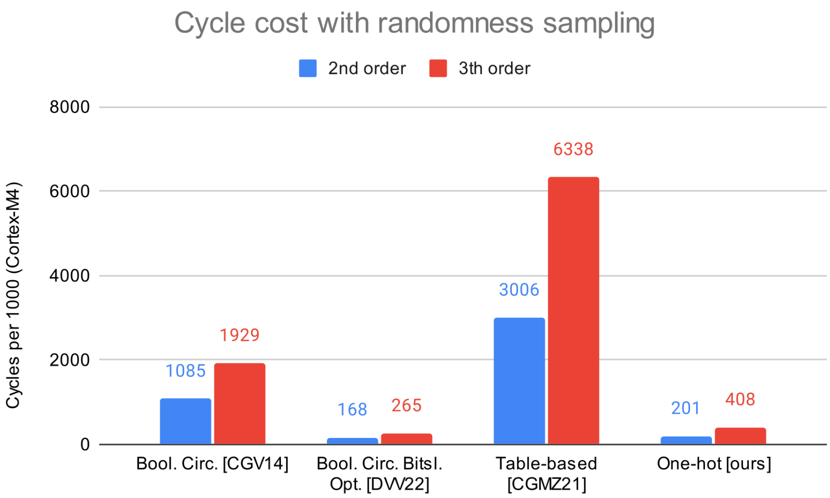
KU LEUVEN


Cycle cost w/o randomness sampling
2nd order
3th order
4000
3000
2000
1000
0
Cycles per 1000 (Cortex-M4)
863
1485
143
215
1648
3515
103
206
Bool. Circ. [CGV14]
Bool. Circ. Bitsl. Opt. [DVV22]
Table-based [CGMZ21]
One-hot [ours]

KU LEUVEN



Randomness cost
2nd order
3th order
kilobytes
250
200
150
100
50
0
17,4
34,8
2
4
106,4
221,2
7,6
15,4
Bool. Circ. [CGV14]
Bool. Circ. Bitsl. Opt. [DVV22]
Table-based [CGMZ21]
One-hot [ours]


Cycle cost with randomness sampling
2nd order
3th order
8000
6000
4000
2000
0
Cycles per 1000 (Cortex-M4)
1085
1929
168
265
3006
6338
201
408
Bool. Circ. [CGV14]
Bool. Circ. Bitsl. Opt. [DVV22]
Table-based [CGMZ21]
One-hot [ours]

KU LEUVEN

Comparison

- Table-based
  - 16x faster
  - 14x less randomness needed
  - Note that both are proof of concept implementations and not optimized

Comparison

- Table-based
  - 16x faster
  - 14x less randomness needed
  - Note that both are proof of concept implementations and not optimized
- Circuit-based
  - We are up to 35% faster when disregarding randomness sampling
  - We are up to 50% slower when including randomness sampling
  - We have 4x higher randomness cost

Comparison

- Table-based
  - 16x faster
  - 14x less randomness needed
  - Note that both are proof of concept implementations and not optimized
- Circuit-based
  - We are up to 35% faster when disregarding randomness sampling
  - We are up to 50% slower when including randomness sampling
  - We have 4x higher randomness cost

Is circuit-based better?

- Not necessarily

Is circuit-based better?

- Not necessarily
- Higher-order circuit-based methods are quite mature [CGV14]
  - Optimized implementations available
- Higher-order table-based methods are newer [CGMZ21]
  - No optimized implementation available yet

Is circuit-based better?

- Not necessarily
- Higher-order circuit-based methods are quite mature [CGV14]
  - Optimized implementations available
- Higher-order table-based methods are newer [CGMZ21]
  - No optimized implementation available yet
- Already caught up in speed, maybe speedup possible?
- Focus point: randomness reduction

Conclusion & Future work

- Compared to table-based A2B:
  - We are 16x faster and need 14x less randomness
- Compared to circuit-based methods
  - We are 1.35x faster if randomness cost is not counted
  - We are 1.5x slower if randomness needs to be sampled on Cortex-M4
  - We need 4x more randomness

Conclusion & Future work

- Compared to table-based A2B:
  - We are 16x faster and need 14x less randomness
- Compared to circuit-based methods
  - We are 1.35x faster if randomness cost is not counted
  - We are 1.5x slower if randomness needs to be sampled on Cortex-M4
  - We need 4x more randomness
- Future work:
  - Randomness reduction
  - Optimized implementation
  - First-order optimized version
  - Constant hamming-weight intermediate representation useful?

Bibliography I

Jean-Sébastien Coron, François Gérard, Simon Montoya, and Rina Zeitoun.
High-order table-based conversion algorithms and masking lattice-based encryption.
Cryptology ePrint Archive, Paper 2021/1314, 2021.
https://eprint.iacr.org/2021/1314.

Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala.
Secure conversion between Boolean and arithmetic masking of any order.
In Lejla Batina and Matthew Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 188–205. Springer, Heidelberg, September 2014.

Jean-Sébastien Coron and Alexei Tchulkine.
A new algorithm for switching from arithmetic to Boolean masking.
In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 89–97. Springer, Heidelberg, September 2003.

Bibliography II

Jan-Pieter D'Anvers, Michiel Van Beirendonck, and Ingrid Verbauwhede.
Revisiting higher-order masked comparison for lattice-based cryptography: Algorithms and bit-sliced implementations.
Cryptology ePrint Archive, Report 2022/110, 2022.
https://eprint.iacr.org/2022/110.

Louis Goubin.
A sound method for switching between Boolean and arithmetic masking.
In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 3–15. Springer, Heidelberg, May 2001.

bits	8-bit		16-bit		32-bit	
order	2	3	2	3	2	3
Bool. circ. [CGV14]	228.7	402.4	442.6	767.1	862.5	1484.7
Bool. circ. (opt. bitsl.) [DBV22]	37.3	55.1	72.3	**108.2**	142.6	214.6
Table-based [CGMZ21]	427.2	916.2	847.2	1806.6	1647.8	3514,8
One-hot [ours]	**27.3**	**51.2**	**54.3**	109.6	**103.3**	**206.4**
When sampling the randomness from the on-chip TRNG generator:						
Bool. circ. [CGV14]	294.1	532.9	560.2	1002.0	1084.5	1928.6
Bool. circ. (opt. bitsliced) [DBV22]	**43.2**	**67.1**	**84.8**	**133.3**	**168.2**	**265.9**
Table-based [CGMZ21]	767.8	1617.4	1524.1	3213.0	3005,8	6338.3
One-hot [ours]	47.0	90.4	103.3	207.5	201.3	408.2

Table: Cost to perform 32 A2B conversions on Cortex M4 in 1000 cycles. The top results ignore randomness sampling using the on-chip TRNG generator, the bottom results include the randomness sampling.

bits	8-bit		16-bit		32-bit	
order	2	3	2	3	2	3
Bool. circ.	5,120	10,240	9,216	18,432	17,408	34,816
Bool. circ. (opt. bitsliced)	**464**	**928**	**976**	**1,952**	**2,000**	**4,000**
Table-based	26,624	55,296	53,248	110,592	106,496	221,184
One-hot [ours]	1,536	3,072	3,840	7,680	7,680	15,360

Table: Randomness cost to perform 32 A2B conversions in bytes.

Order		Cycles w/o TRNG 2	Cycles w/o TRNG 3	Cycles with TRNG 2	Cycles with TRNG 3	Randomness 2	Randomness 3
simple optimized	Kyber	2.5M	4.1M	3.1M	5.3M	48K	100K
streamlined hybrid	Kyber	2.4M	3.4M	3.3M	4.4M	80K	95K
one-hot (ours)	Kyber	2.3M	4.3M	4.6M	8.9M	184K	369K
simple optimized	Saber	1.3M	2.0M	1.6M	2.6M	26K	53K
one-hot (ours)	Saber	1.0M	2.0M	2.2M	4.2M	92K	184K

Table: Cycle and randomness cost of the state-of-the-art higher-order comparison methods