CryptoExperts

WE INNOVATE TO SECURE YOUR BUSINESS

From tightPROVE to Tornado: Automatic Generation of Probing-Secure Masked Bitsliced Implementations

Joint works

[Asiacrypt 18] Sonia Belaïd, Dahmun Goudarzi, and Matthieu Rivain
[Eurocrypt 20] Sonia Belaïd, Pierre-Evariste Dagand, Darius Mercadier, Matthieu Rivain, and Raphaël Wintersdorff

Contributions

- tightPROVE: verification in the *bit probing model*
- tightPROVE+: verification in the *register probing model*
- Tornado: global compiler
- Benchmarks of mask-friendly NIST lightweight schemes

CryptoExperts

Brief reminder

- Software implementations are usually protected with *masking*
    - $x \rightarrow (x_0, \ldots, x_t) = [x]$ such that
        - $x_1, \ldots, x_t \leftarrow U$
        - $x_0 \oplus \ldots \oplus x_t = x$
- *t-probing security*: a circuit is t-probing secure if any set of t intermediate variables is independent from the secret

Limitation of previous composition properties

- Previous tools (e.g., maskComp) add a refresh to Circuit 1
- But Circuit 1 was already t-probing secure

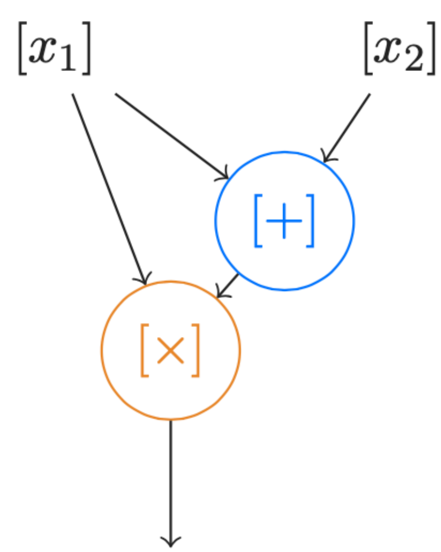


Figure: Circuit 1.

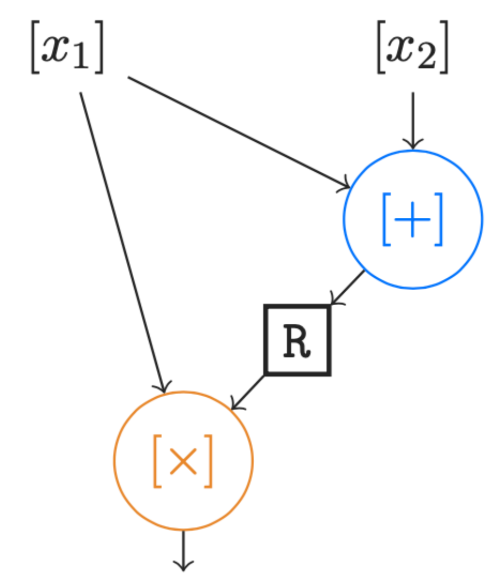


Figure: Circuit 1 after `maskComp`.

> *A circuit is t-probing secure if any set of t intermediate variables is independent from the secret*

CRYPTOEXPERTS

tightPROVE

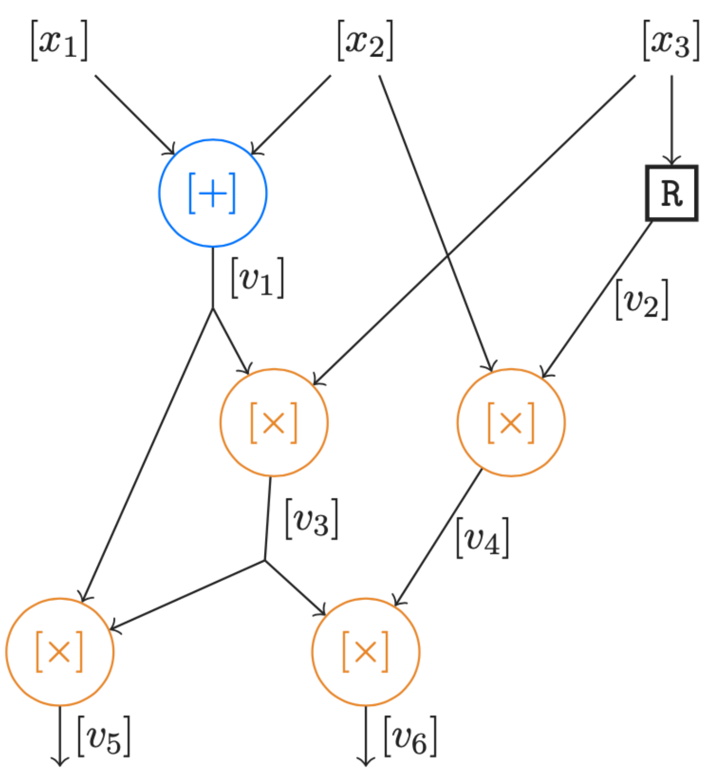
New proposal: tightPROVE

- Apply to tight shared circuits:
  - sharewise additions,
  - ISW-multiplications,
  - ISW-refresh gadgets
- Determine exactly whether a tight shared circuit is probing secure for any order t
  1. Reduction to a simplified problem
  2. Resolution of the simplified problem
  3. Extension to larger circuits
- On $GF(2)$

CryptoExperts

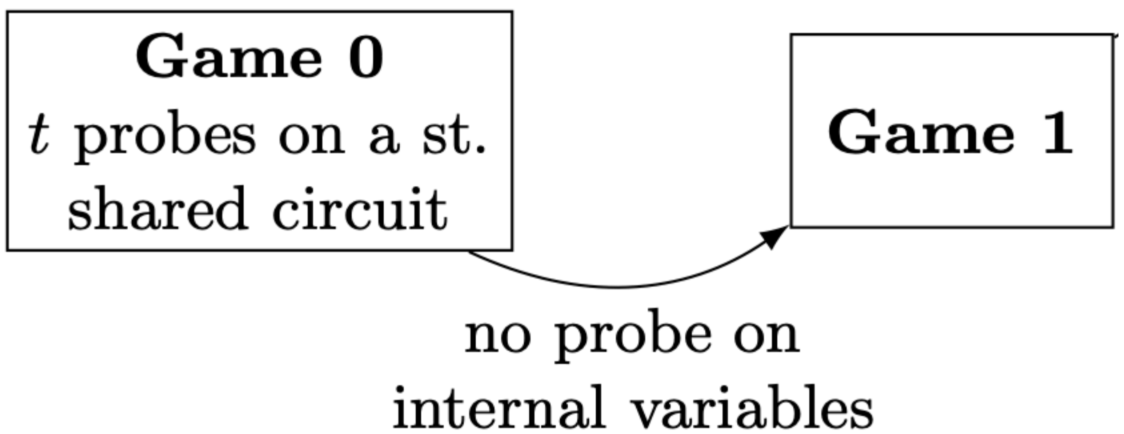
tightPROVE

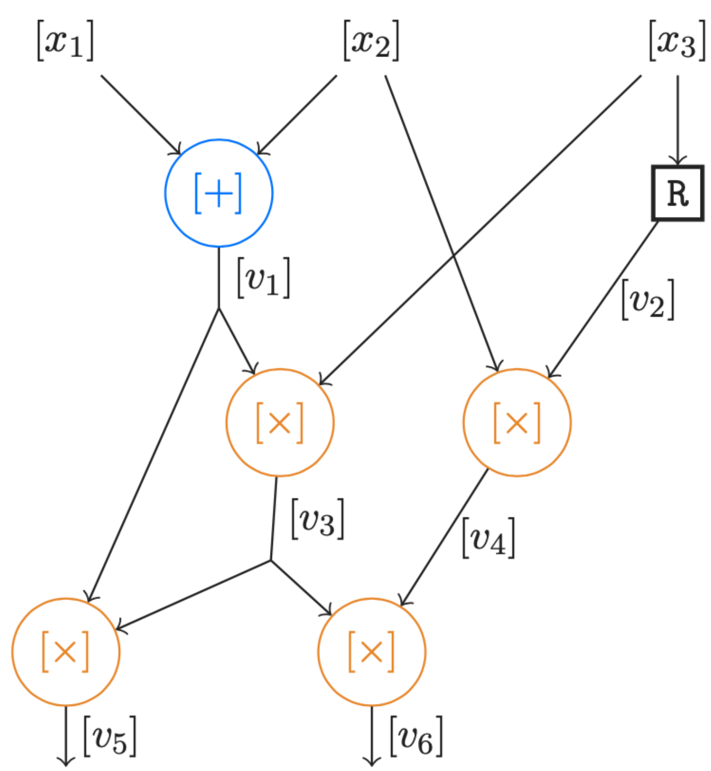
Game 0
t probes on a st. shared circuit



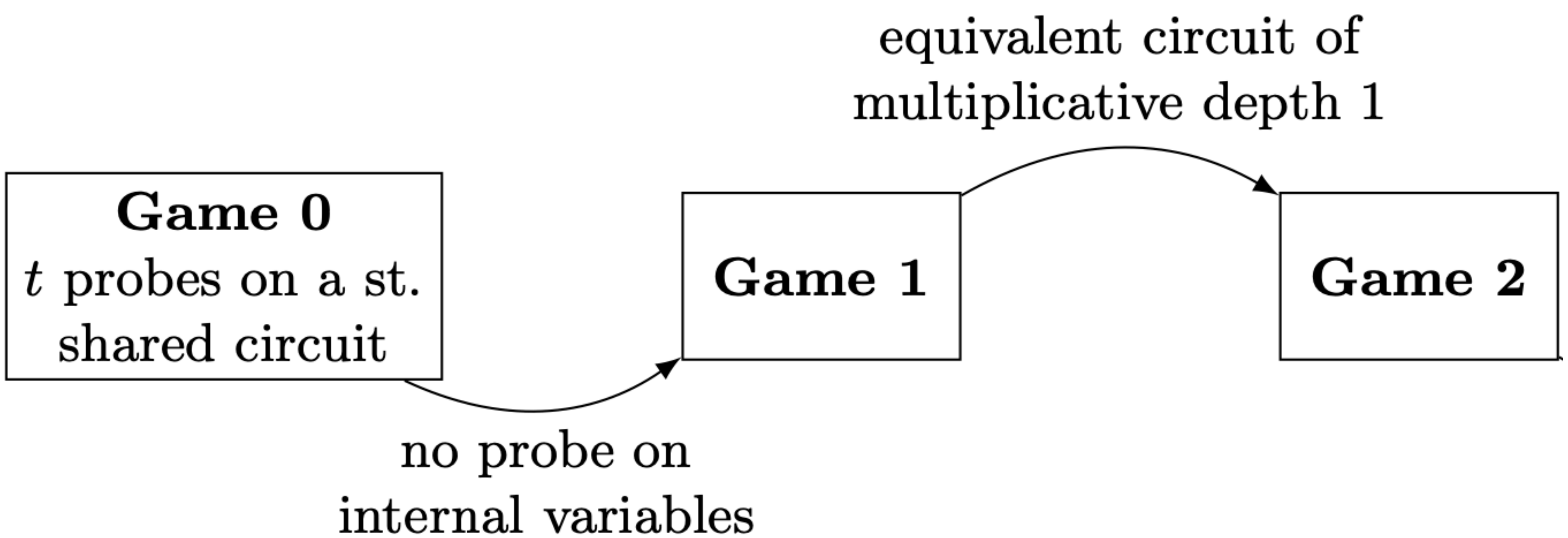
CryptoExperts

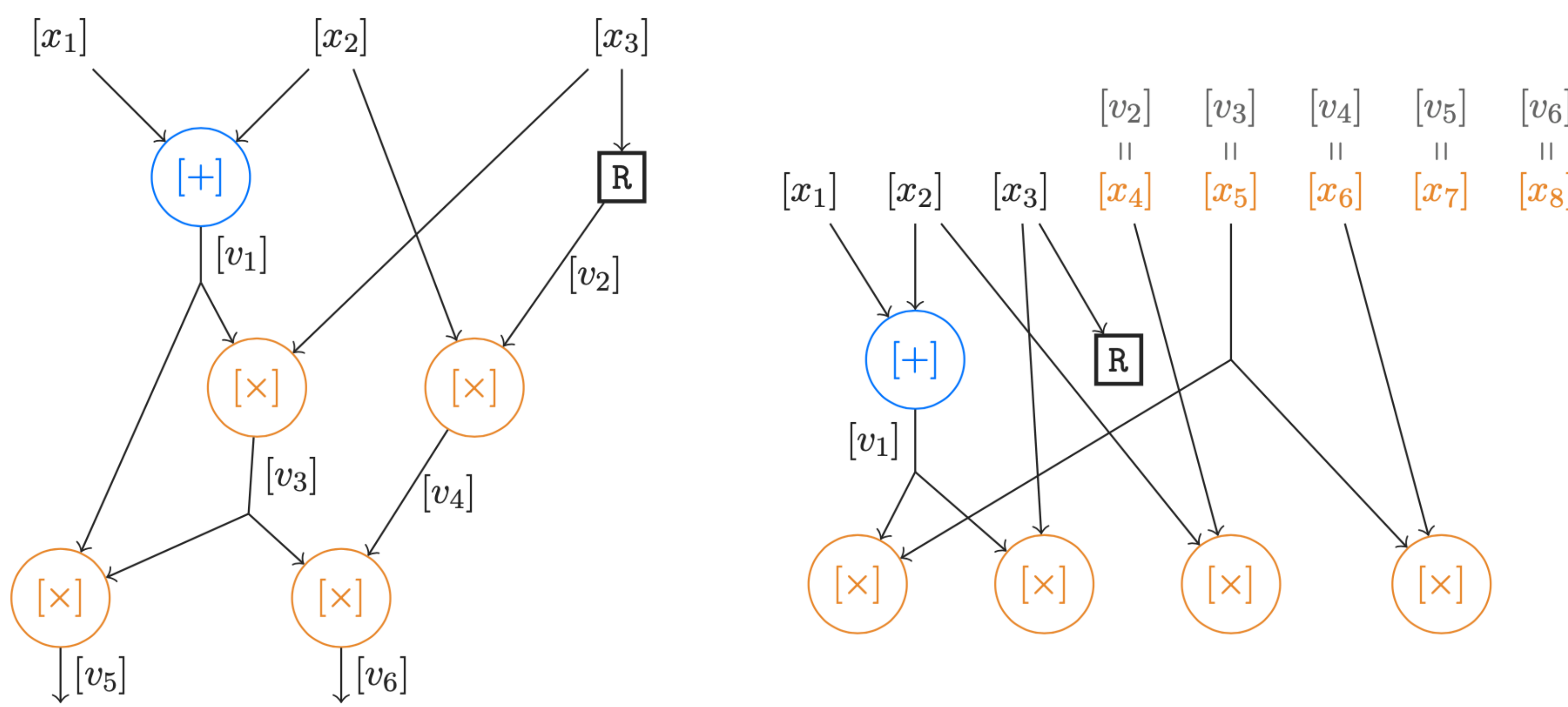
tightPROVE


Game 0
t probes on a st. shared circuit
Game 1
no probe on internal variables

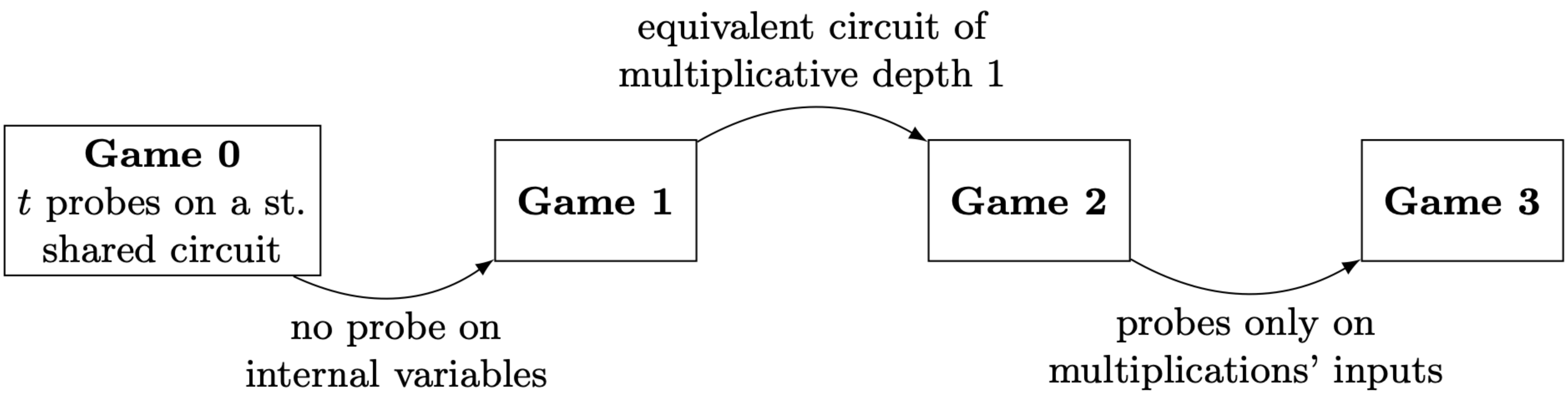

[x1]
[x2]
[x3]
[+]
R
[v1]
[v2]
[×]
[×]
[v3]
[v4]
[×]
[×]
[v5]
[v6]

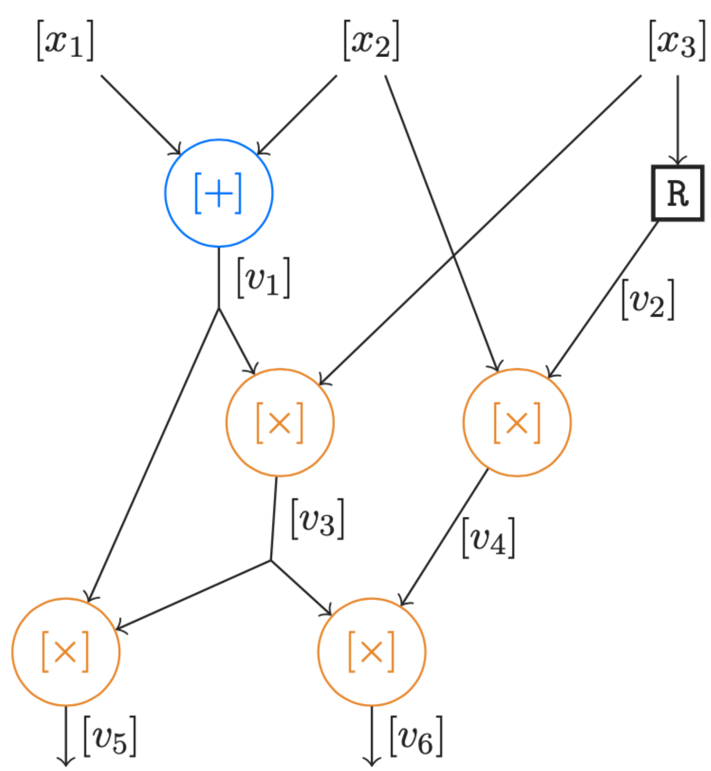
tightPROVE

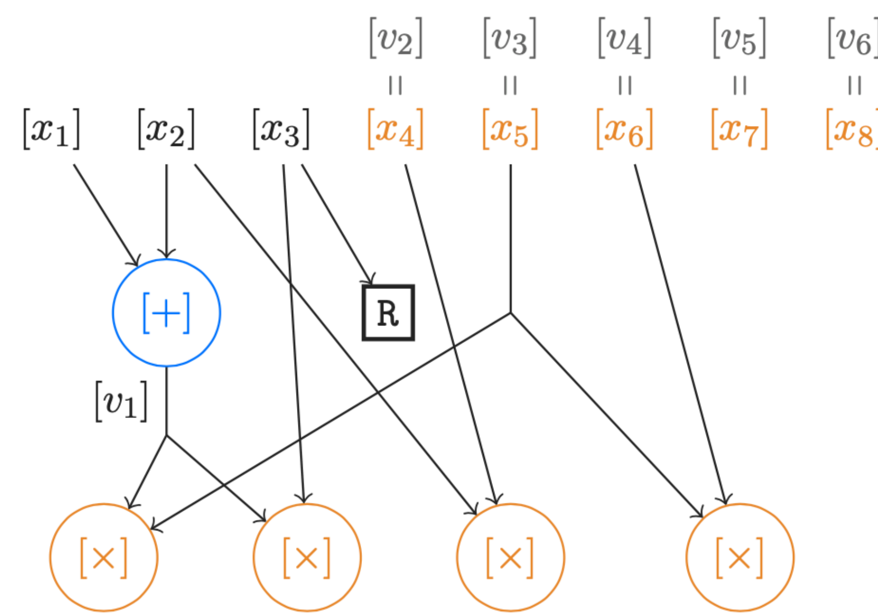

equivalent circuit of
multiplicative depth 1
Game 0
t probes on a st.
shared circuit
Game 1
Game 2
no probe on
internal variables


[x1] [x2] [x3]
[+]
R
[v1] [v2] [v3] [v4] [v5] [v6]
[×]
[v2] [v3] [v4] [v5] [v6]
[x1] [x2] [x3] [x4] [x5] [x6] [x7] [x8]

tightPROVE


Game 0
t probes on a st. shared circuit
no probe on internal variables
Game 1
equivalent circuit of multiplicative depth 1
Game 2
probes only on multiplications' inputs
Game 3

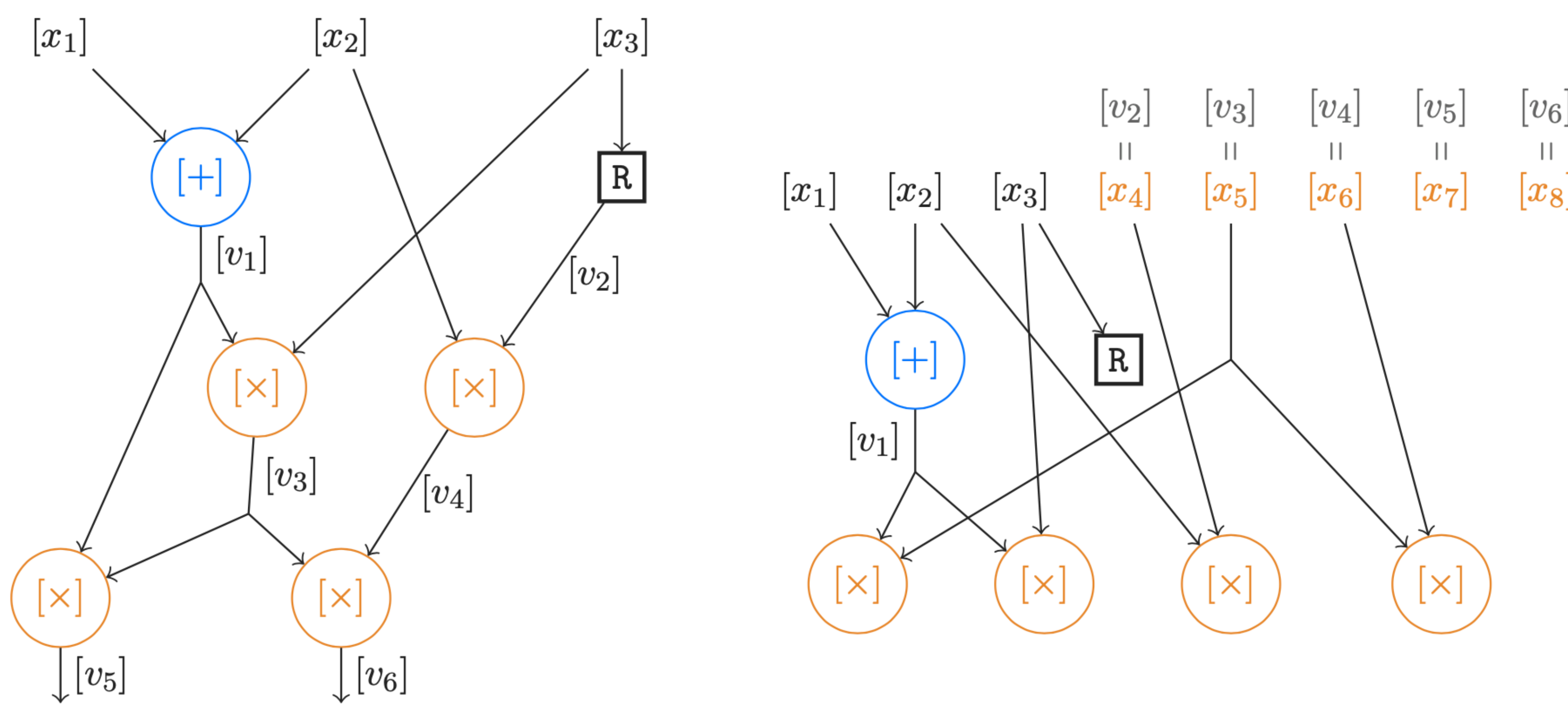

$[x_1]$ $[x_2]$ $[x_3]$
$[+]$
R
$[v_1]$ $[v_2]$ $[v_3]$ $[v_4]$ $[v_5]$ $[v_6]$
$[\times]$


$[v_2]$ $[v_3]$ $[v_4]$ $[v_5]$ $[v_6]$
$[x_1]$ $[x_2]$ $[x_3]$ $[x_4]$ $[x_5]$ $[x_6]$ $[x_7]$ $[x_8]$
$[+]$
R
$[v_1]$
$[\times]$

Linear algebra problem

- The set of probes can be seen as linear combinations of the inputs coordinates (given the share they involve)
  - The probes can be distributed into t+1 matrices $M_0, M_1, \ldots, M_t$
  - The t-probing security of the circuit is equivalent to

$$Im(M_0) \cap Im(M_1) \cap Im(M_t) = \varnothing$$



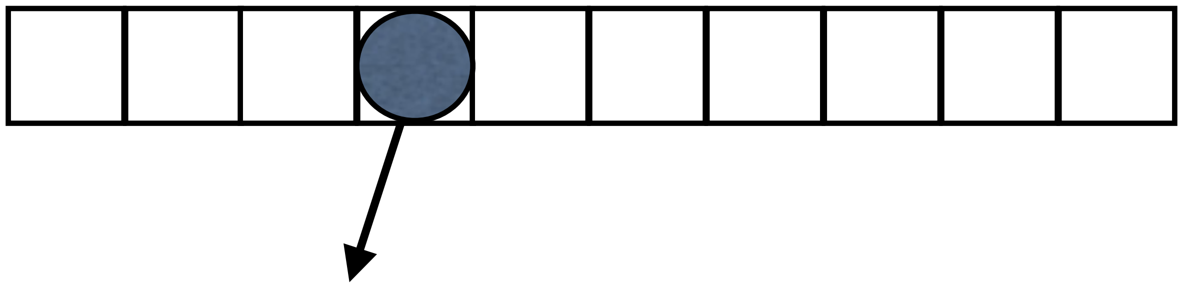
CryptoExperts

tightPROVE+

CryptoExperts

tightPROVE+

- (bit-)probing model

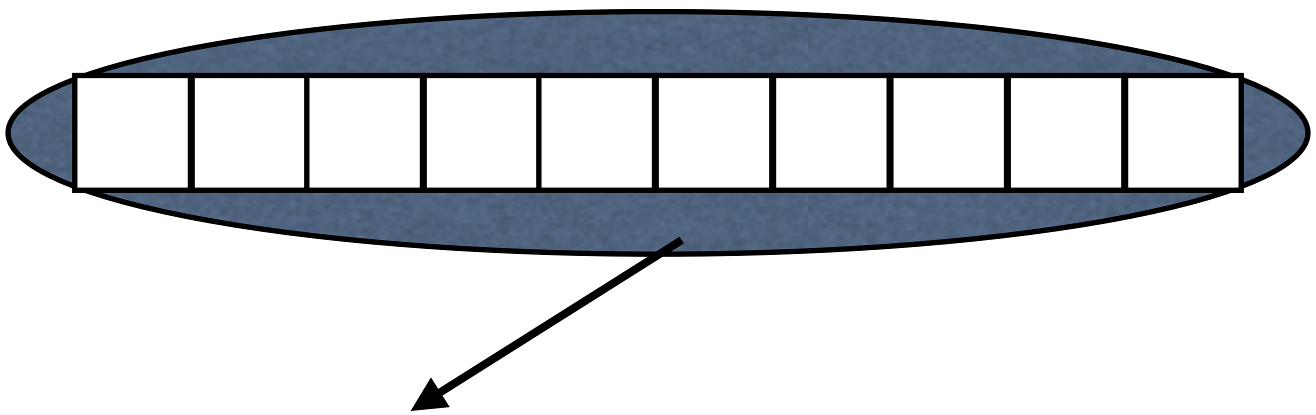


one probe or one observation = one bit

CryptoExperts

tightPROVE+

- register-probing model



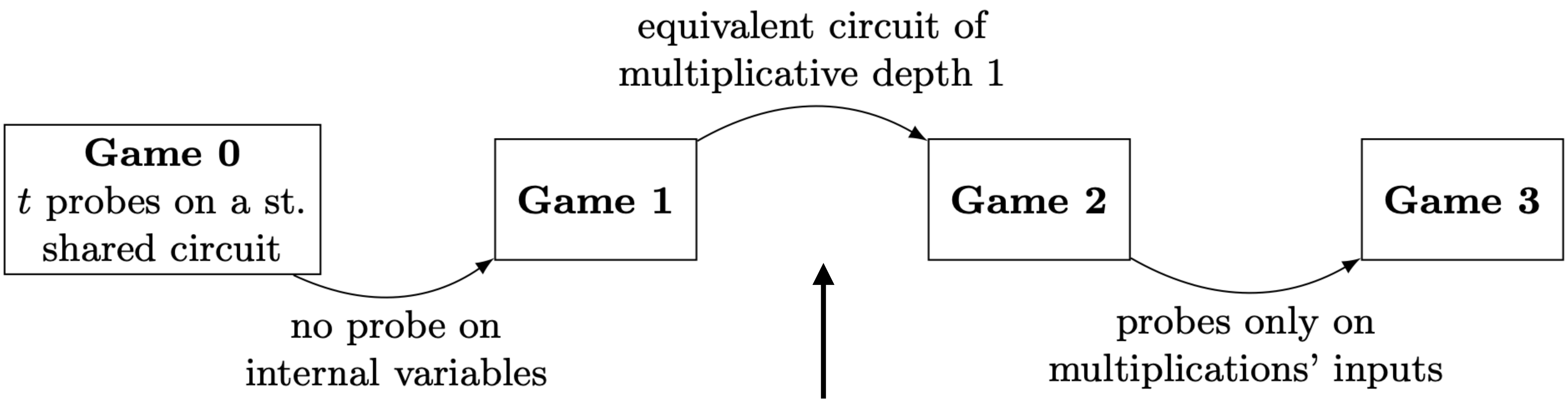
one probe or one observation = m bits

CryptoExperts

tightPROVE+

- Gadgets
  - ISW multiplication gadget
  - ISW refreshing gadget
  - Sharewise addition gadget
  - Sharewise multiplication by a constant
  - Sharewise addition with a constant
  - Sharewise left shift, right shift and rotation gadgets

tightPROVE+



Similar to tightPROVE, still relies on the fact that the output of a multiplication gadget is a uniform Boolean sharing, independent from its inputs (no observation in the gadget)

- Sharewise gadgets are not restricted to the addition anymore
- Multiplication and refresh gadgets are still SNI in the register probing model

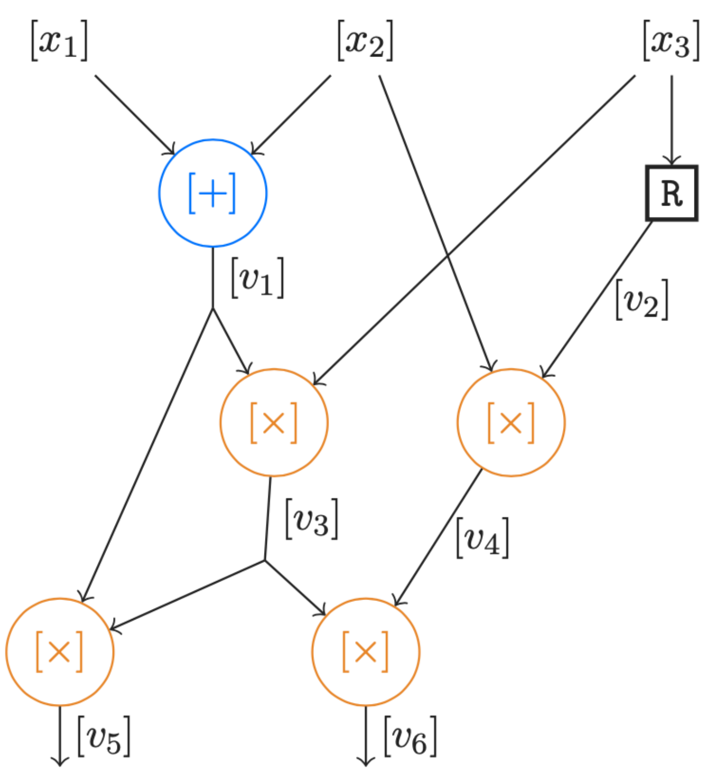
Sub-products of shares at the beginning of multiplications

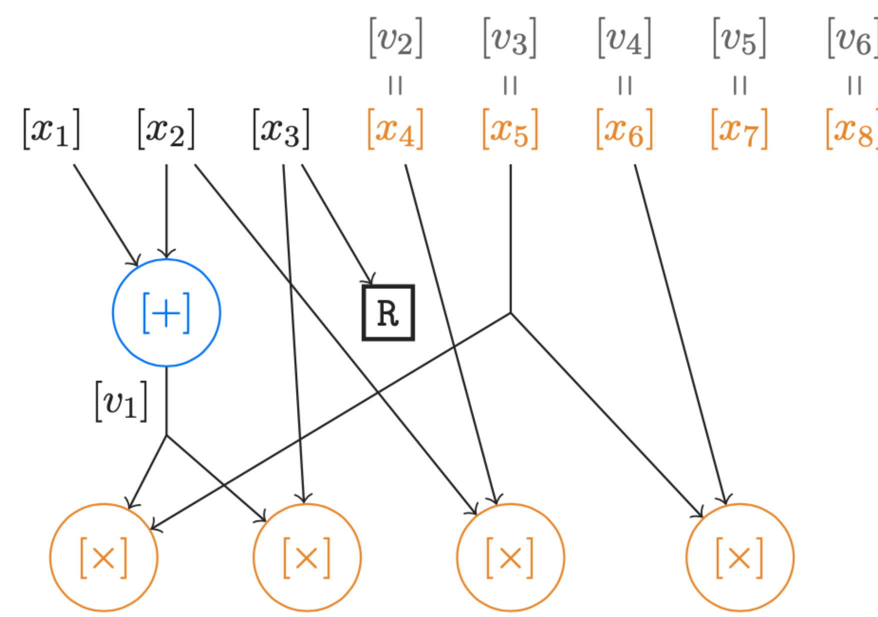
CryptoExperts

Linear algebra problem

- The set of probes can still be seen as linear combinations of the inputs coordinates (given the share they involve)
  - The probes can be distributed into t+1 matrices $M_0, M_1, \ldots, M_t$ (of higher dimensions given the register size)
  - The t-probing security of the circuit is equivalent to

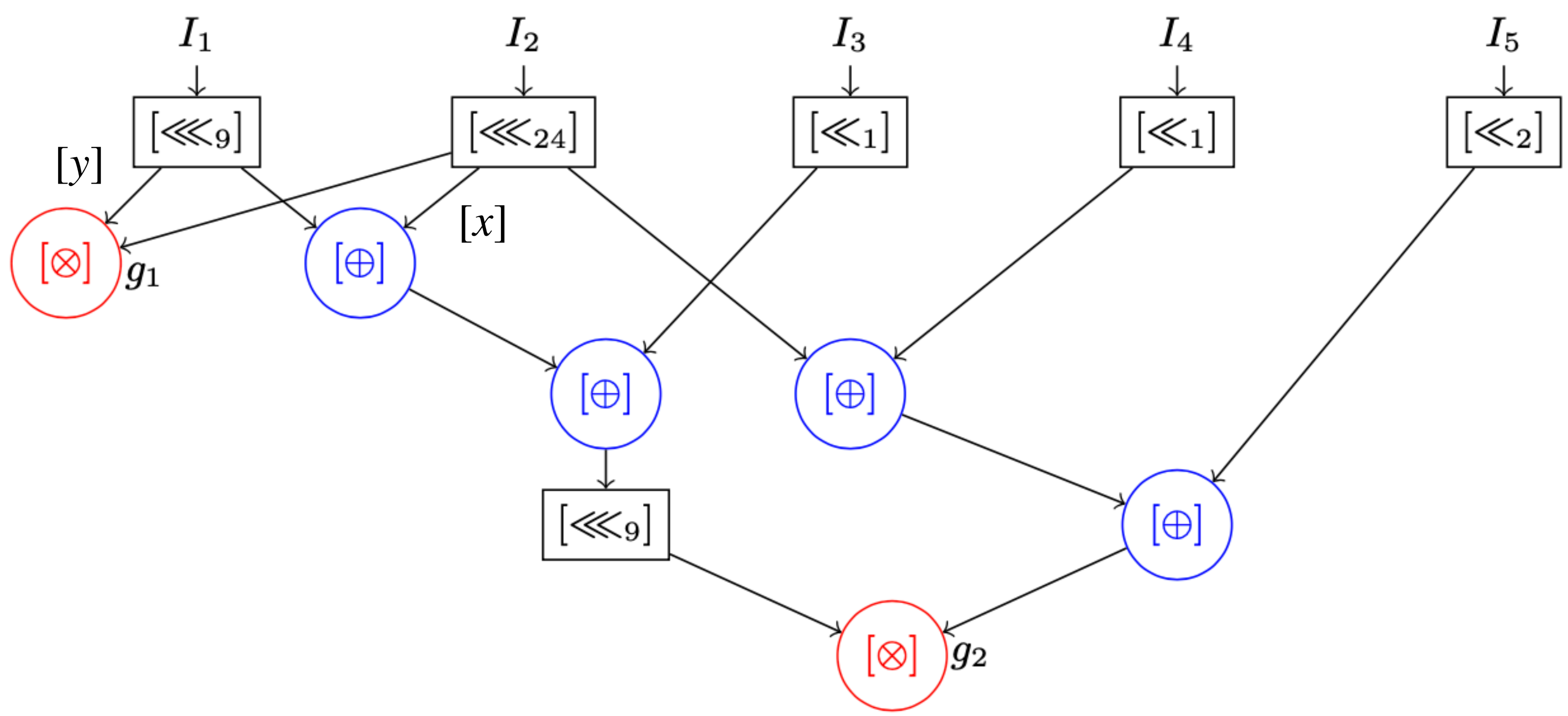
$$Im(M_0) \cap Im(M_1) \cap Im(M_t) = \varnothing$$





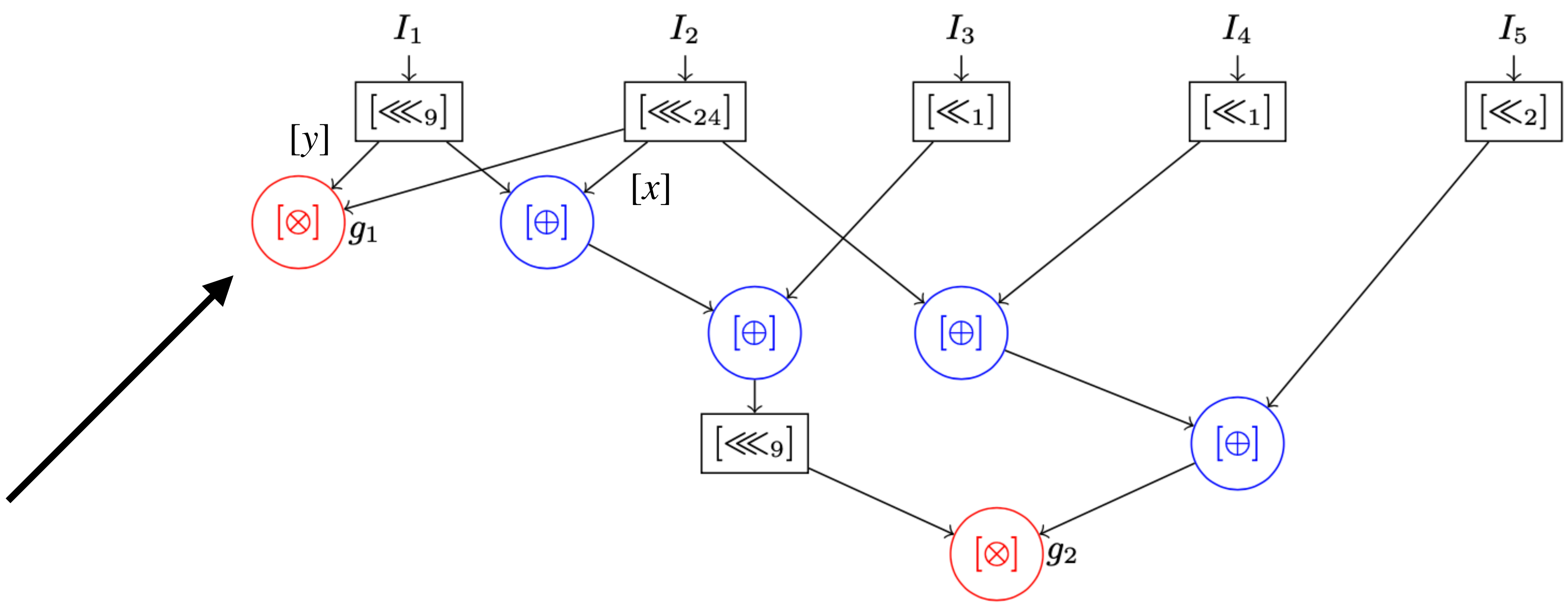
CryptoExperts

Example: Gimli



CryptoExperts

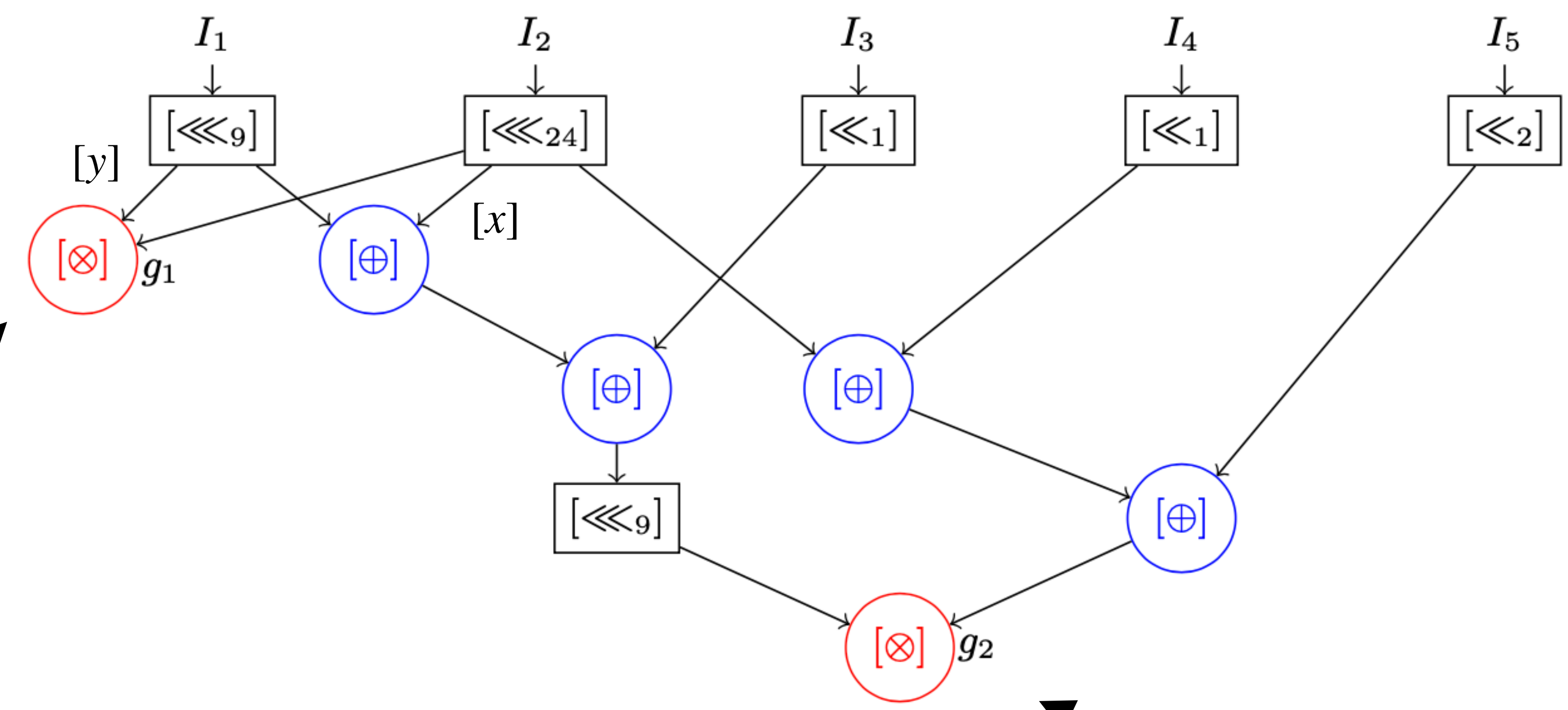
Example: Gimli



We get $x_{32,0}$ and $y_{32,1}$

1 probe

Example: Gimli

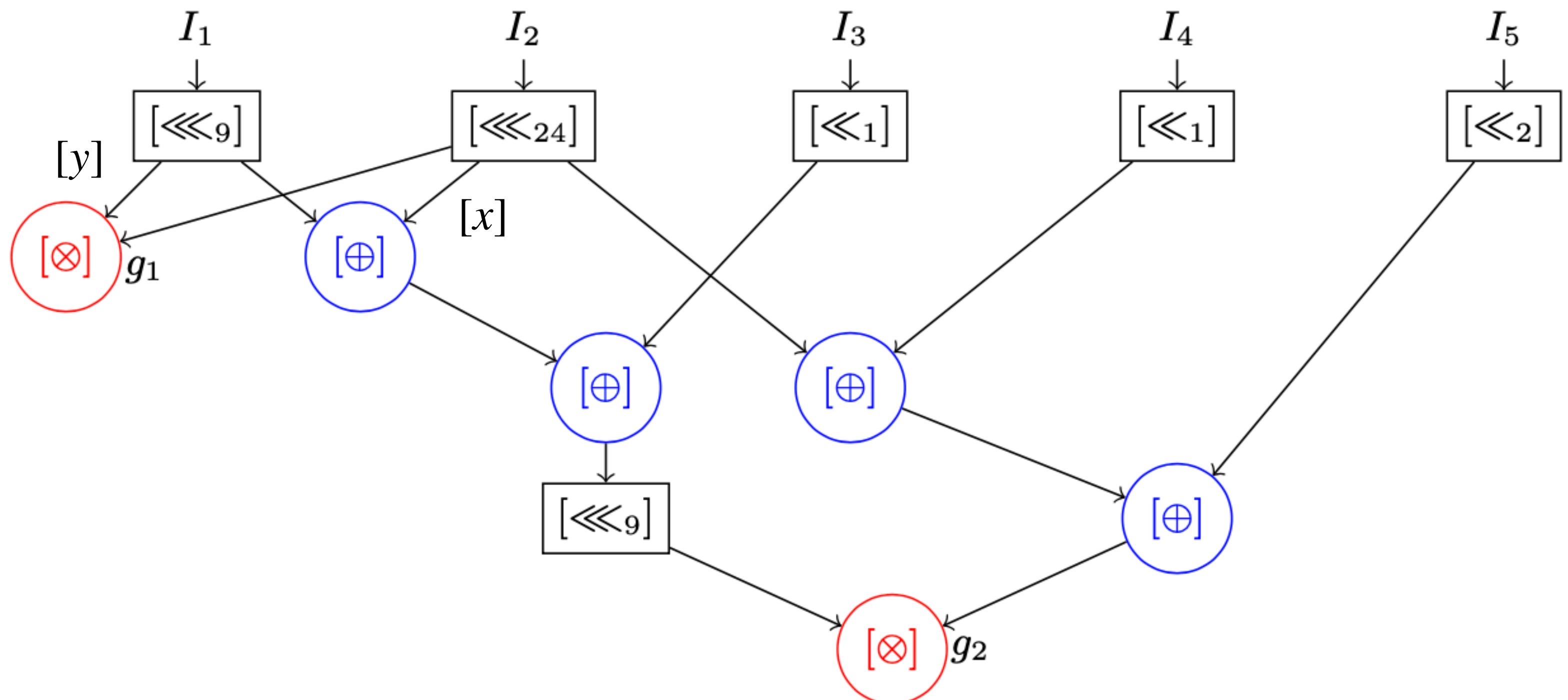


We get $x_{32,0}$ and $y_{32,1}$

We get $x_{32,2}$ and $x_{32,1} \oplus y_{32,1}$

2 probes

Example: Gimli



We get $x_{32,0}$ and $y_{32,1}$

We get $x_{32,2}$ and $x_{32,1} \oplus y_{32,1}$

2 probes $\longrightarrow$ 3 shares

Tornado

Tornado

High-level specification

```
node ascon12(input:u64x5)
         returns (output:u64x5)
vars
    consts:u64[12],
    state:u64x5[13]
let
    consts = (0xf0, 0xe1, 0xd2, 0xc3,
              0xb4, 0xa5, 0x96, 0x87,
              0x78, 0x69, 0x5a, 0x4b);

    state[0] = input;
    forall i in [0, 11] {
        state[i+1] = LinearLayer
                  (Sbox
                  (AddConstant
                  (state[i],consts[i])))
    }
    output = state[12]
tel
```

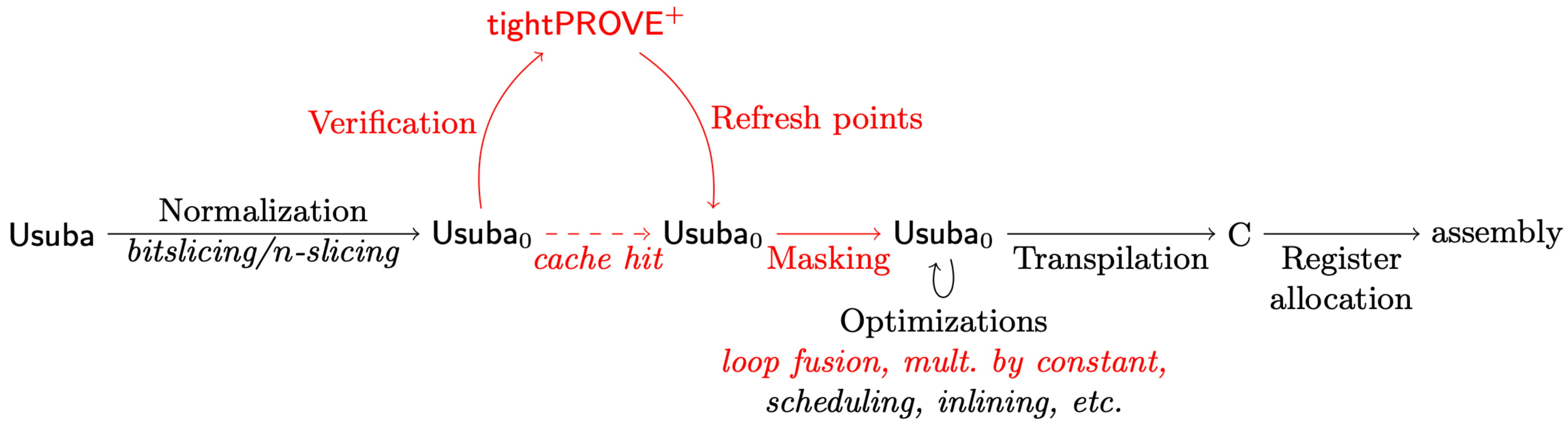
Tornado →

Masked implementation

```
ascon12:
  stmfd sp!, {r4, r5, r6, r7, \
              r8, r9, r10, fp, lr}
  ldmia r0, {r4-r5}
  sub sp, sp, #620
  str r4, [sp, #168]
  str r5, [sp, #172]
  add r5, r0, #8
  ldmia r5, {r4-r5}
  str r4, [sp, #160]
  str r5, [sp, #164]
  add r5, r0, #16
  ldmia r5, {r4-r5}
  str r4, [sp, #192]
  str r5, [sp, #196]
  add r5, r0, #24
  ldmia r5, {r4-r5}
  str r4, [sp, #184]
  ...
```

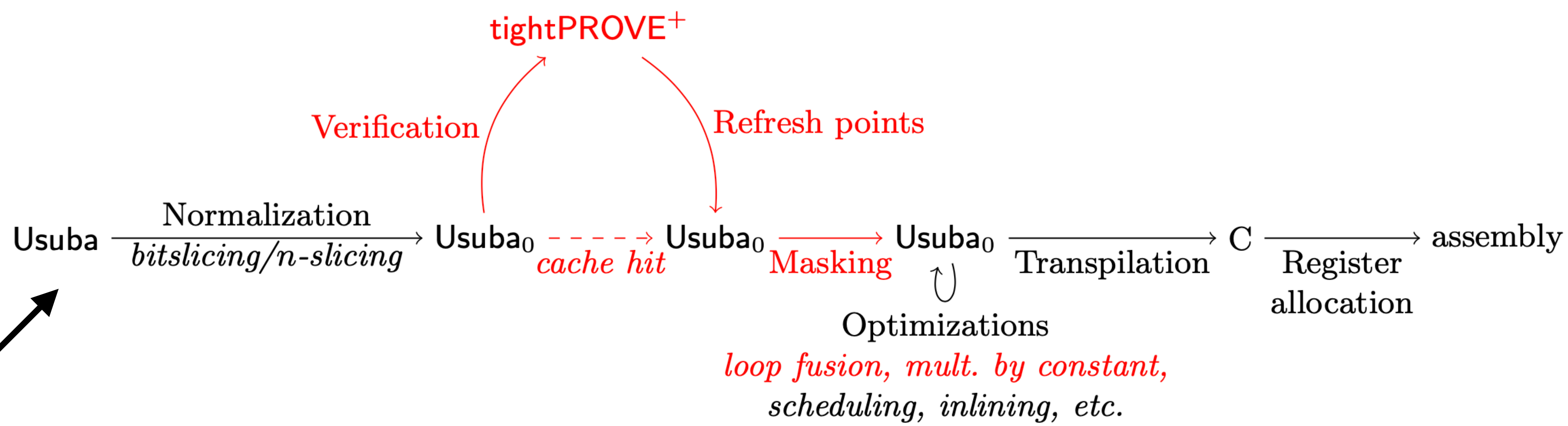

CryptoExperts

Tornado



CryptoExperts

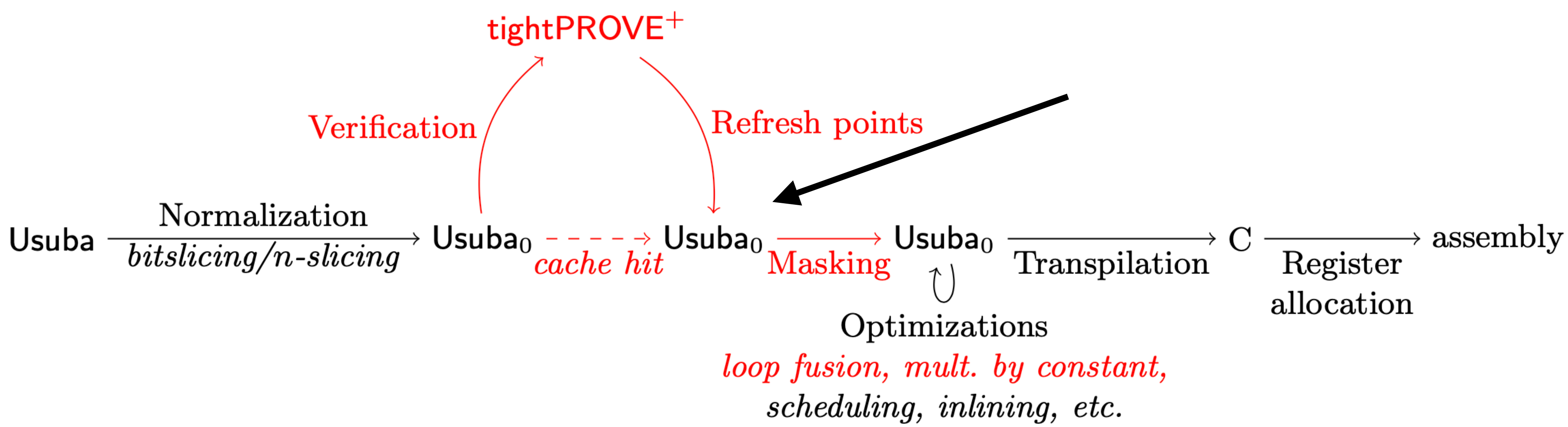
Tornado




```
node f(i1, i2, i3, i4, i5 : u32)
    returns (out : u32)
let
    t1 = (i1 <<< 9) & (i2 <<< 24);
    t2 = (i3 << 1)  ^ (i4 >> 31);
    t3 = i2 ^ t1;
    t4 = t3 & t2;
    t5 = t3 ^ t2;
    t6 = (t2 <<< 3) & t5;
    out = t4 ^ t6;
tel
```

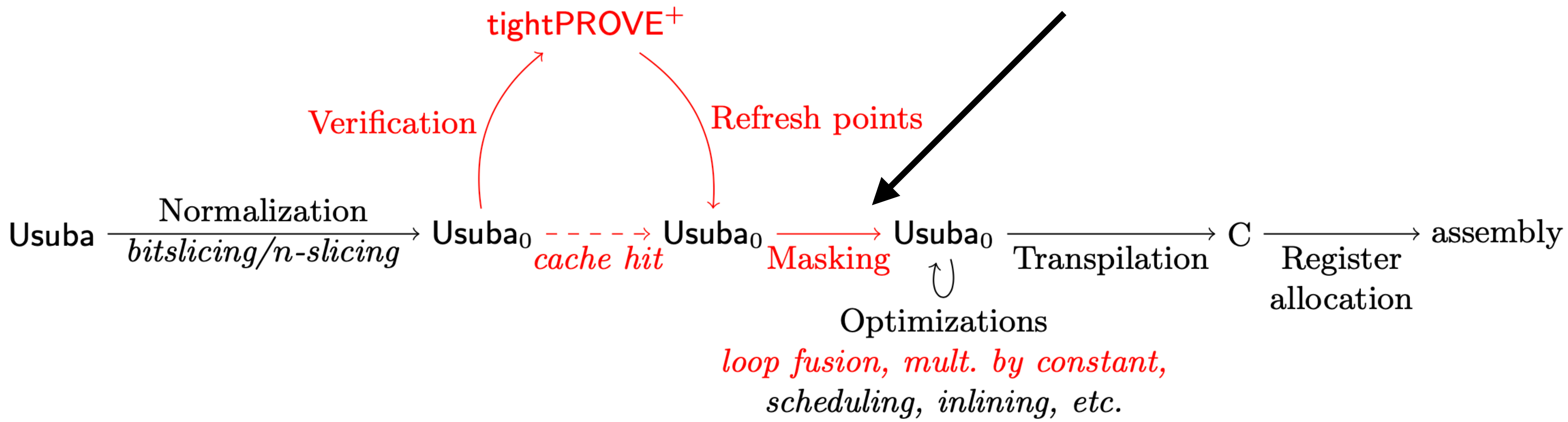

CryptoExperts

Tornado




```
node f(i1, i2, i3, i4, i5 : u32)
    returns (out : u32)
vars tmp1, tmp2, tmp3, tmp4, tmp5, t5_r : u32
let
    tmp1 = i1 <<< 9;
    tmp2 = i2 <<< 24;
    t1 = tmp1 & tmp2;
    tmp3 = i3 << 1;
    tmp4 = i4 >> 31;
    t2 = tmp3 ^ tmp4;
    t3 = i2 ^ t1;
    t4 = t3 & t2;
    t5 = t3 ^ t2;
    t5_r = refresh(t5);
    tmp5 = t2 <<< 3;
    t6 = tmp5 & t5_r;
    out = t4 ^ t6;
tel
```

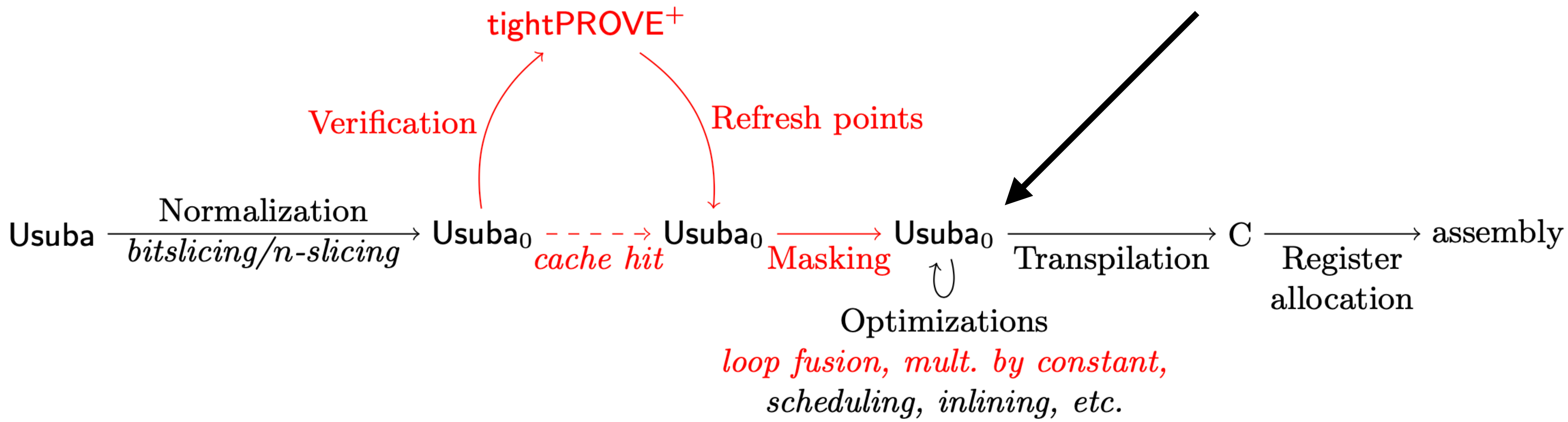
Tornado




```
node f(i1, i2, i3, i4, i5 : u32[NB_S])
    returns (out : u32[NB_S])
vars tmp1, tmp2, tmp3, tmp4, tmp5, t5_r : u32[NB_S]
let
    forall i in [0 .. NB_S-1] { tmp1[NB_S] = i1[NB_S] <<< 9; }
    forall i in [0 .. NB_S-1] { tmp2[NB_S] = i2[NB_S] <<< 24; }
    t1 = MASKED_AND(tmp1, tmp2);
    forall i in [0 .. NB_S-1] { tmp3[NB_S] = i3[NB_S] << 1; }
    forall i in [0 .. NB_S-1] { tmp4[NB_S] = i4[NB_S] >> 31; }
    forall i in [0 .. NB_S-1] { t2[NB_S] = tmp3[NB_S] ^ tmp4[NB_S]; }
    forall i in [0 .. NB_S-1] { t3[NB_S] = i2[NB_S] ^ t1[NB_S]; }
    t4 = MASKED_AND(t3, t2);
    forall i in [0 .. NB_S-1] { t5[NB_S] = t3[NB_S] ^ t2[NB_S]; }
    t5_r = REFRESH(t5);
    forall i in [0 .. NB_S-1] { tmp5[NB_S] = t2[NB_S] <<< 3; }
    t6 = MASKED_AND(tmp5, t5_r);
    forall i in [0 .. NB_S-1] { out[NB_S] = t4[NB_S] ^ t6[NB_S]; }
tel
```


CryptoExperts

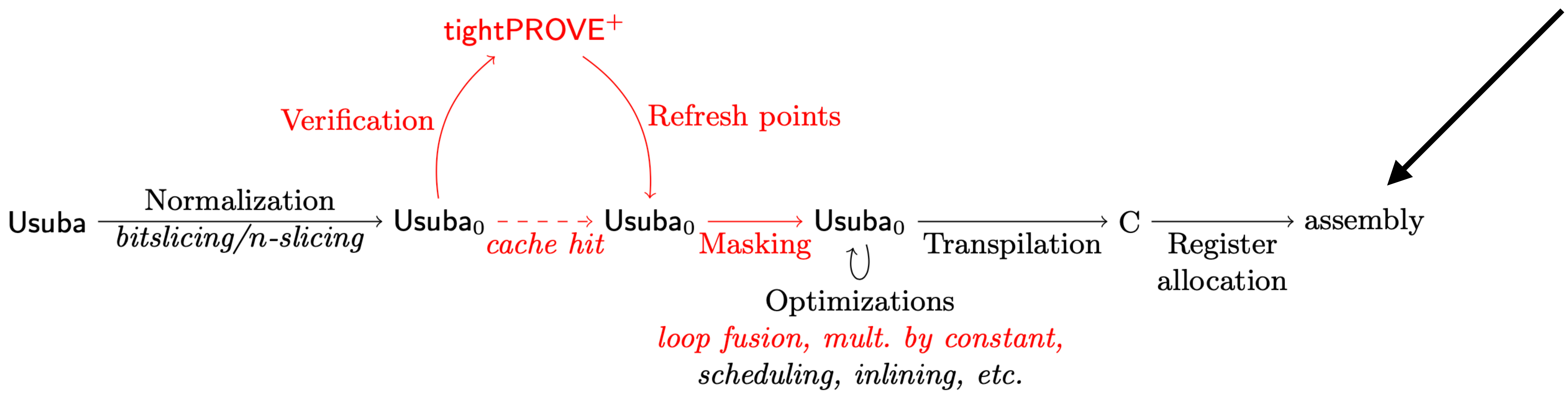
Tornado




```
node f(i1, i2, i3, i4, i5 : u32[NB_S]) returns (out : u32)
vars tmp1, tmp2, tmp3, tmp4, tmp5, t5_r : u32[NB_S]
let
    forall i in [0 .. NB_S-1] {
        tmp1[NB_S] = i1[NB_S] <<< 9;
        tmp2[NB_S] = i2[NB_S] <<< 24;
        tmp3[NB_S] = i3[NB_S] << 1;
        tmp4[NB_S] = i4[NB_S] >> 31;
        t2[NB_S] = tmp3[NB_S] ^ tmp4[NB_S];
        tmp5[NB_S] = t2[NB_S] <<< 3;
    }
    t1 = MASKED_AND(tmp1, tmp2);
    forall i in [0 .. NB_S-1] {
        t3[NB_S] = i2[NB_S] ^ t1[NB_S];
        t5[NB_S] = t3[NB_S] ^ t2[NB_S];
    }
    t4 = MASKED_AND(t3, t2);
    t5_r = REFRESH(t5);
    t6 = MASKED_AND(tmp5, t5_r);
    forall i in [0 .. NB_S-1] { out[NB_S] = t4[NB_S] ^ t6[NB_S]; }
tel
```


CryptoExperts

Tornado




```
f:
  stmfd sp!, {r4, r5, r6, r7, r8, \
            r9, r10, fp, lr}
  sub sp, sp, #5056
  sub sp, sp, #36
  sub r5, r1, #4
  add r1, sp, #2048
  sub r6, r1, #12
  mov fp, r6
  mov r10, r5
  mov r4, #0
  sub r0, r0, #4
  sub r2, r2, #4
  sub r3, r3, #4
  add r9, sp, #4
  add r8, sp, #512
  add r7, sp, #1020
.L19:
  ldr r1, [r2, #4]!
  ldr ip, [r3, #4]!
  mov r1, r1, asl #1
  orr r1, r1, ip, lsr #31
  ldr lr, [r0, #4]!
  ldr ip, [r10, #4]!
  add r4, r4, #1
```

```
  mov lr, lr, ror #23
  mov ip, ip, ror #8
  str r1, [fp, #4]!
  cmp r4, #127
  mov r1, r1, ror #29
  str r1, [r7, #4]!
  str lr, [r9, #4]!
  str ip, [r8, #4]!
  bne .L19
  add r3, sp, #1536
  sub r3, r3, #4
  str r3, [sp]
  ldr r0, [sp]
  add r1, sp, #8
  add r2, sp, #516
  bl  isw_mult
  mov r0, #0
  add r3, sp, #3568
  sub r3, r3, #4
  str r3, [sp, #4]
  ldr r3, [sp]
  sub ip, r3, #4
  ldr r3, [sp, #4]
  add lr, sp, #2544
  sub r4, r3, #4
```

```
.L20:
  ldr r1, [ip, #4]!
  ldr r3, [r5, #4]!
  ldr r2, [r6, #4]!
  eor r3, r3, r1
  add r0, r0, #1
  eor r2, r2, r3
  cmp r0, #127
  str r3, [lr, #4]!
  str r2, [r4, #4]!
  bne .L20
  add r3, sp, #2560
  sub r1, r3, #12
  add r3, sp, #2048
  sub r2, r3, #8
  add r0, sp, #3056
  bl  isw_mult
  add r0, sp, #4064
  ldr r1, [sp, #4]
  add r0, r0, #8
  bl  isw_refresh
  add r2, sp, #4064
  add r0, sp, #4544
  add r1, sp, #1024
  add r0, r0, #36
  add r2, r2, #8
```

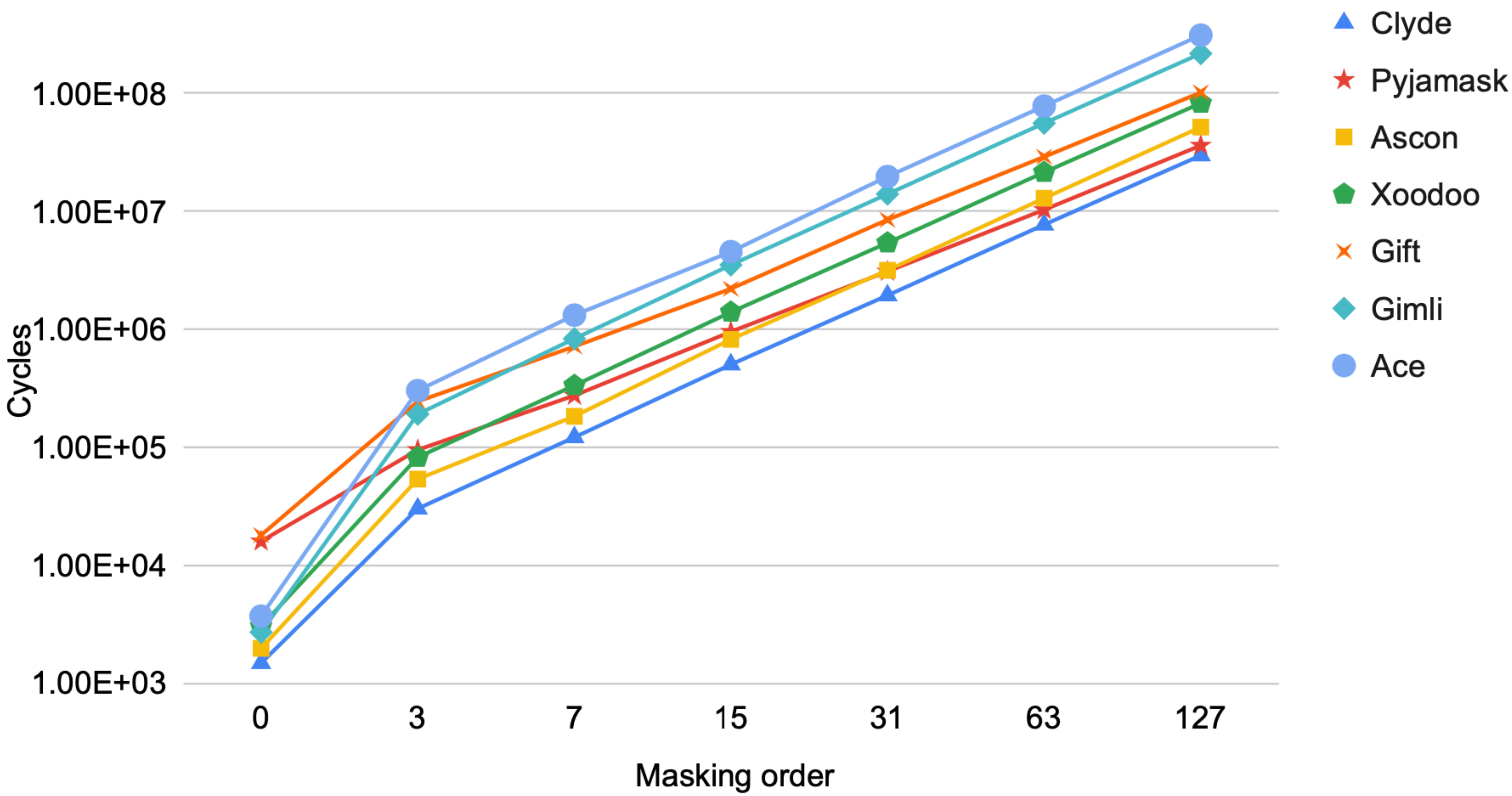
```
  bl  isw_mult
  add r3, sp, #5120
  add r3, r3, #12
  ldr r3, [r3]
  add ip, sp, #4544
  sub r0, r3, #4
  add r3, sp, #3056
  sub r1, r3, #4
  add lr, r3, #504
  add ip, ip, #32
.L21:
  ldr r3, [r1, #4]!
  ldr r2, [ip, #4]!
  cmp r1, lr
  eor r3, r3, r2
  str r3, [r0, #4]!
  bne .L21
  add sp, sp, #5056
  add sp, sp, #36
  @ sp needed
  ldmfd sp!, {r4, r5,\
    r6, r7, r8, r9,\
    r10, fp, lr }
  bx  lr
```

Benchmarks

tightPROVE+

submissions	primitive	time (bitslice)	bit probing security	register size	time (n-slice)	register probing security
block ciphers						
GIFT-COFB, HYENA, SUNDAE-GIFT	GIFT-128	55 H 40 min	✓	32	2 H 15 min	✓
Pyjamask	Pyjamask-128	30 min	✓	32	6 min	✓
SKINNY, ROMULUS	SKINNY-128-256	10 H	✓	-	-	-
Spook	Clyde-128	10 min	✓	32	32 s	✗
permutations						
ACE	ACE	54 H 30 min	✓	32	10 min	✗
ASCON	p^{12}	1 H 45 min	✓	64	1 H 13 min	✓
Elephant	SPONGENT-π[160](1 round)	6 s	✓	-	-	-
Elephant	SPONGENT-π[160](10 rounds)	20 min 40 s	✓	-	-	-
Gimli	Gimli-36	22 H 45 min	✓	32	1 H 10 min	✗
ORANGE, PHOTON-BEETLE	PHOTON-256	2 H	✓	-	-	-
Xoodyak	Xoodoo[12]	2 H 50 min	✓	32	4 H 5 min	✓
others						
Subterranean	blank(8)	17 min	✓	-	-	-

Tornado



CryptoExperts

Thank you

Tornado tool:

https://github.com/CryptoExperts/Tornado/