Let’s standardize garbled circuits!

Vlad Kolesnikov

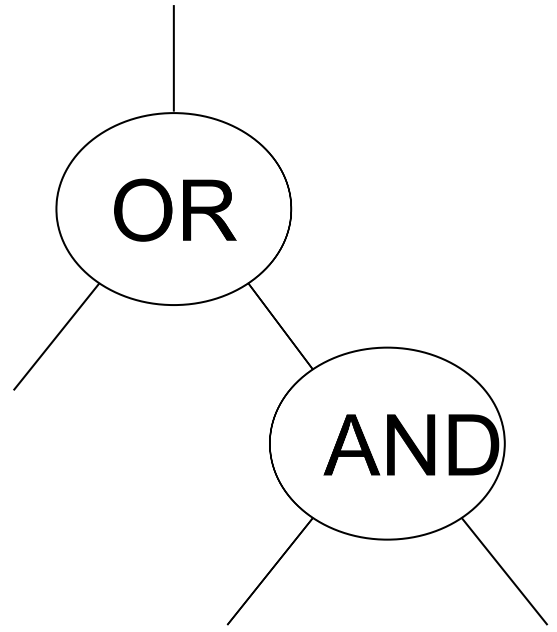
Georgia Tech

Outline

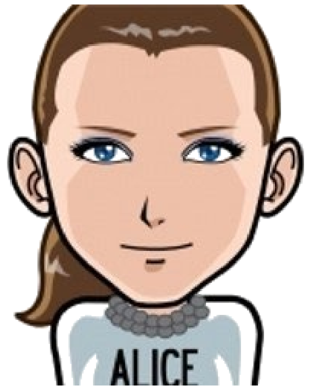
- Garbled Circuits (GC)
- Applications to threshold crypto
- Simplicity and stability
- Many advanced features from basic GC properties

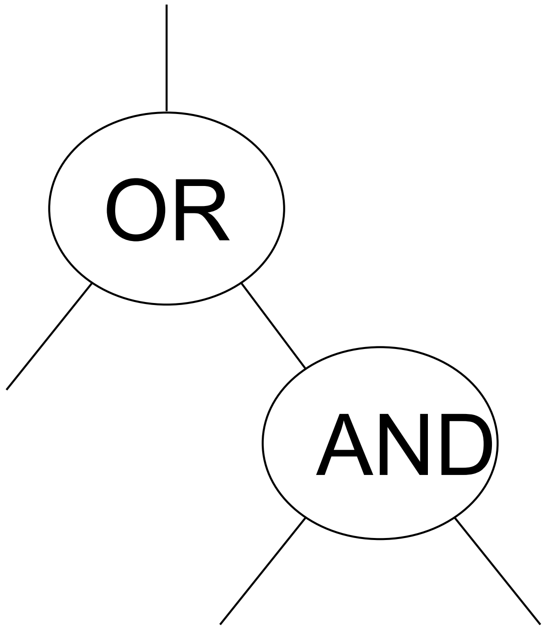
Functions are circuits

$F(x, y) \qquad \rightarrow$

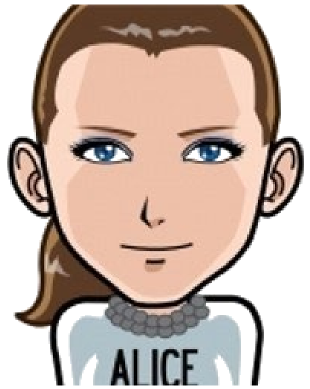


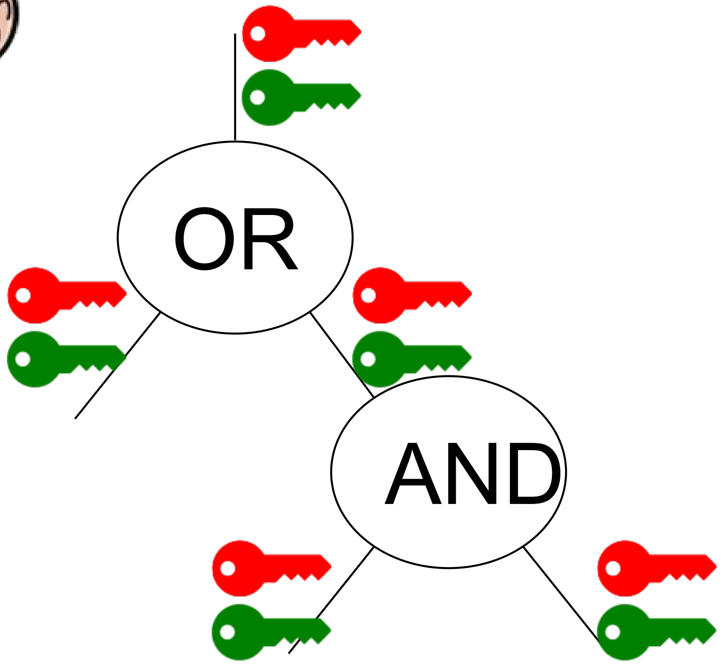
GC intuition: computing on encrypted values



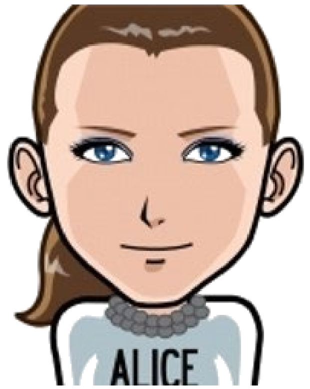


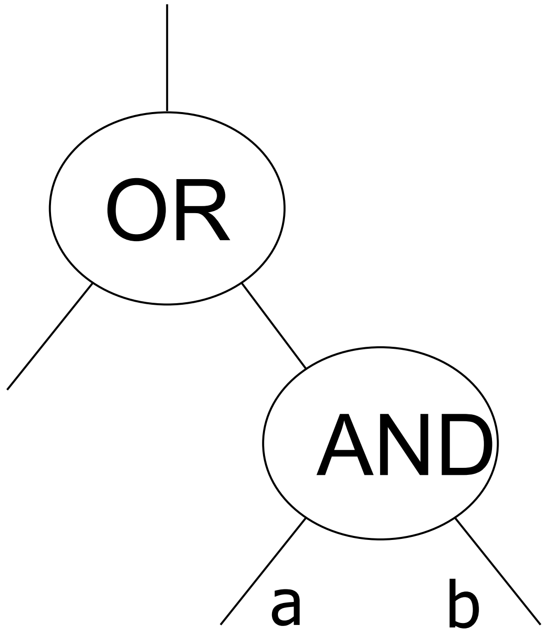
GC intuition: computing on encrypted values



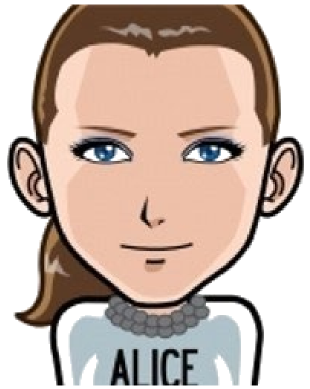


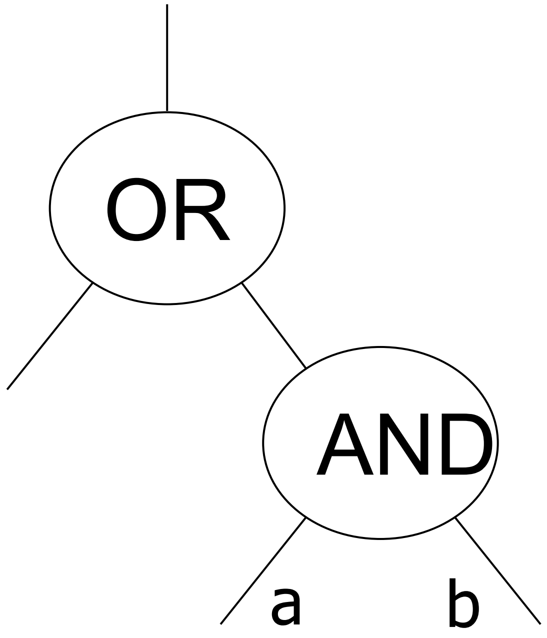
GC intuition: computing on encrypted values





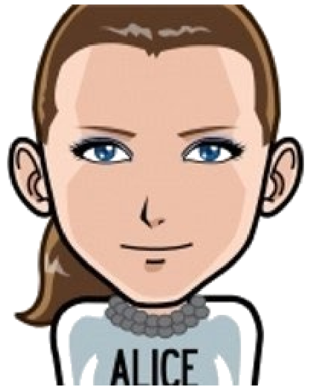
GC intuition: computing on encrypted values

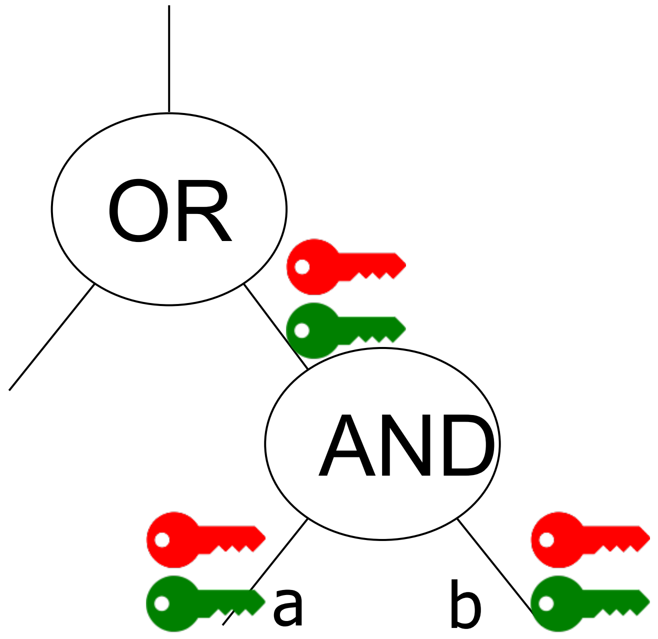




a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

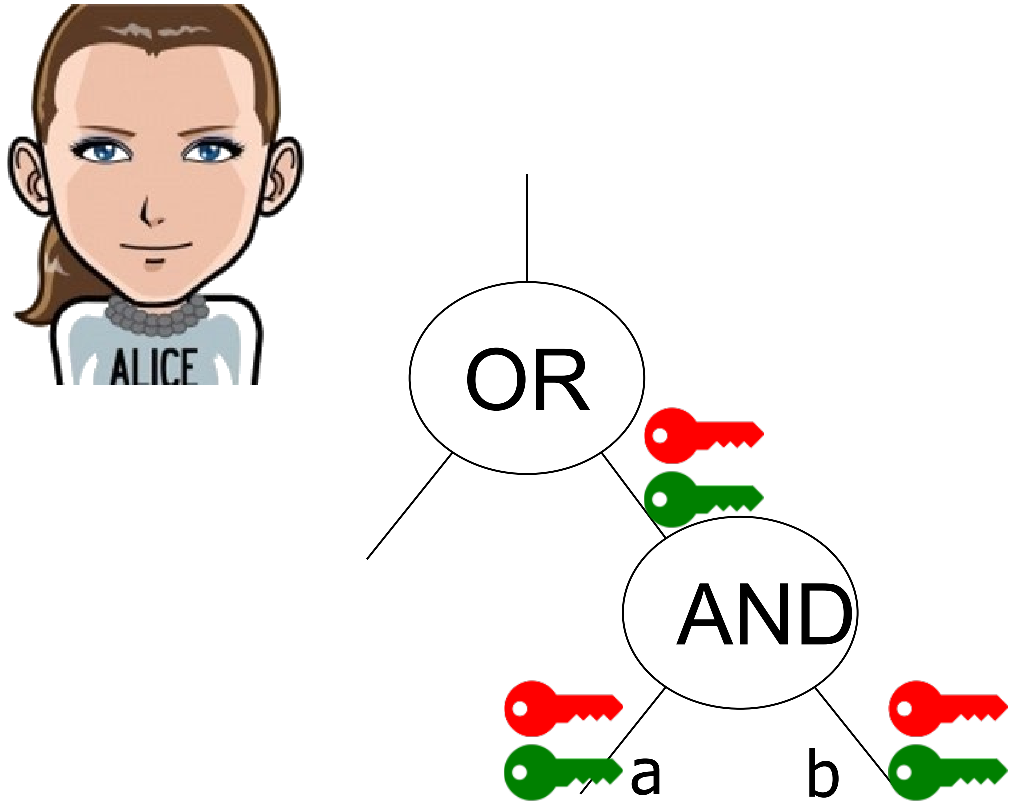
GC intuition: computing on encrypted values

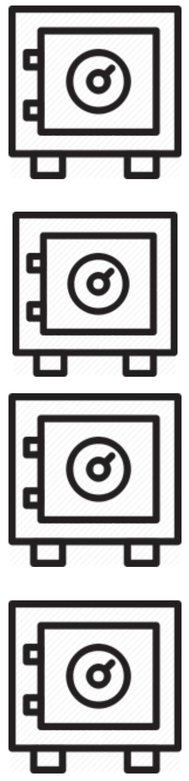




a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

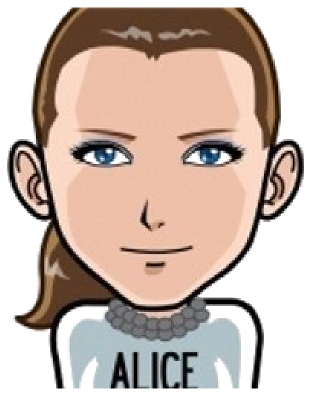
GC intuition: computing on encrypted values

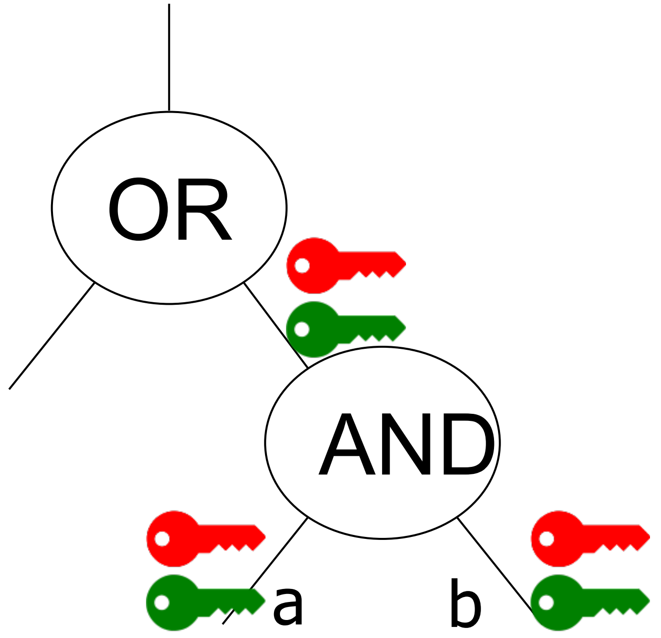


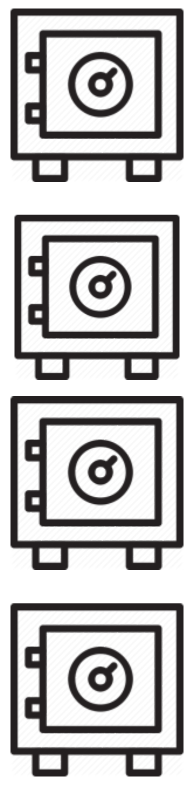


a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

GC intuition: computing on encrypted values

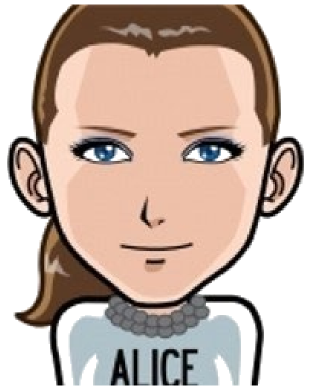


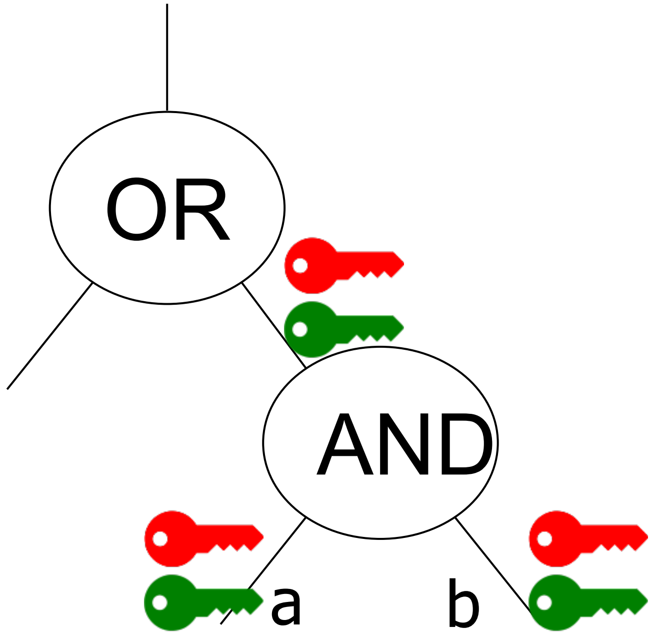


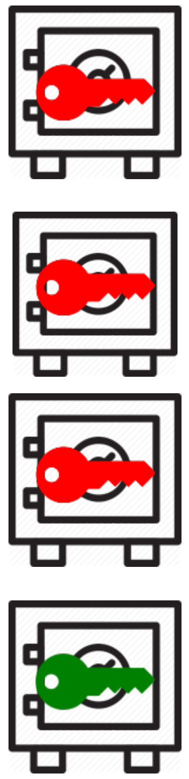


a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

GC intuition: computing on encrypted values

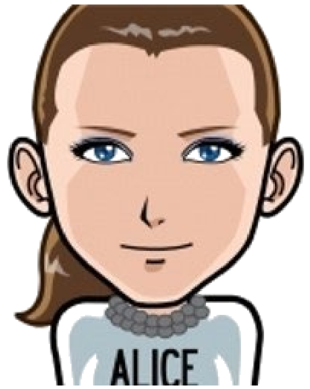


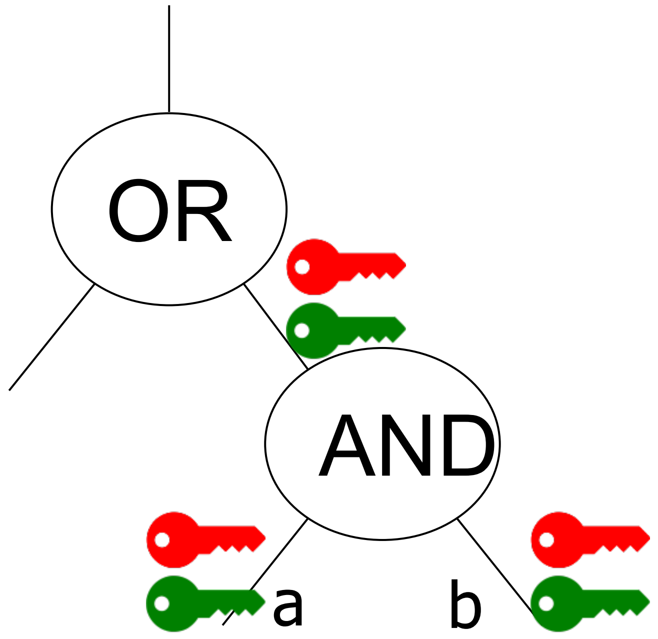


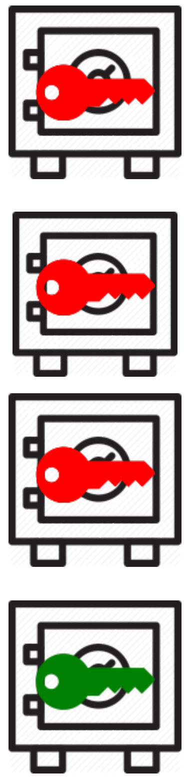


a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

GC intuition: computing on encrypted values

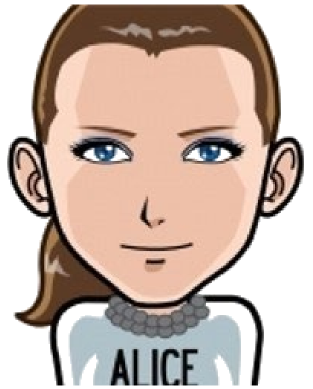


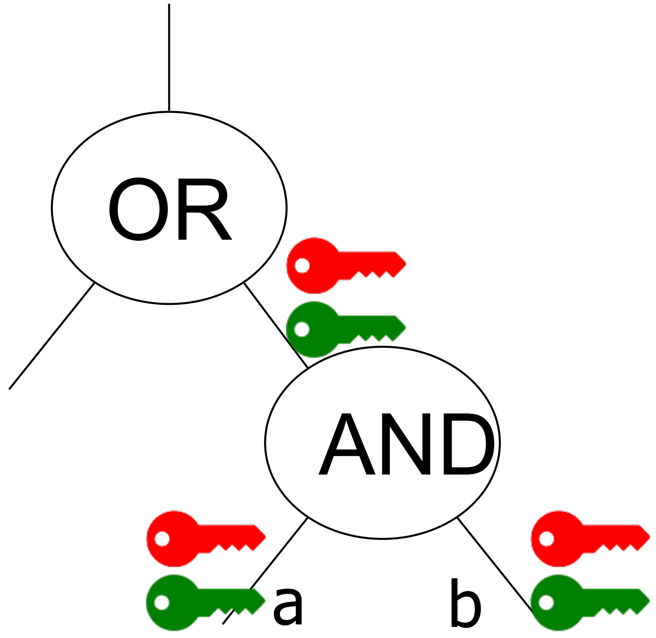


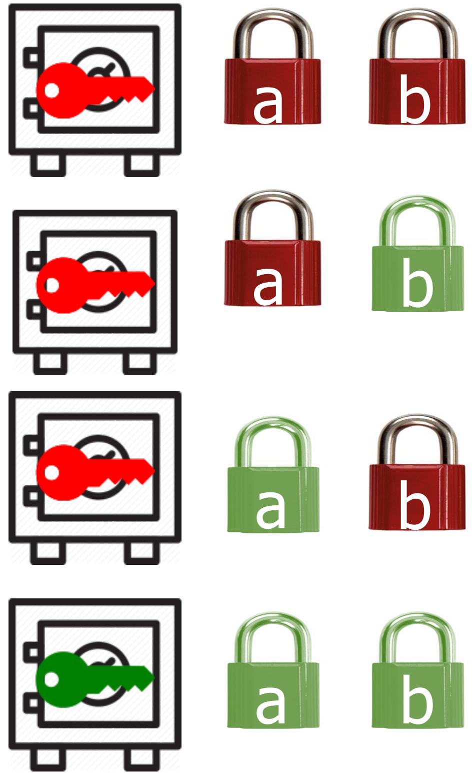


a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

GC intuition: computing on encrypted values

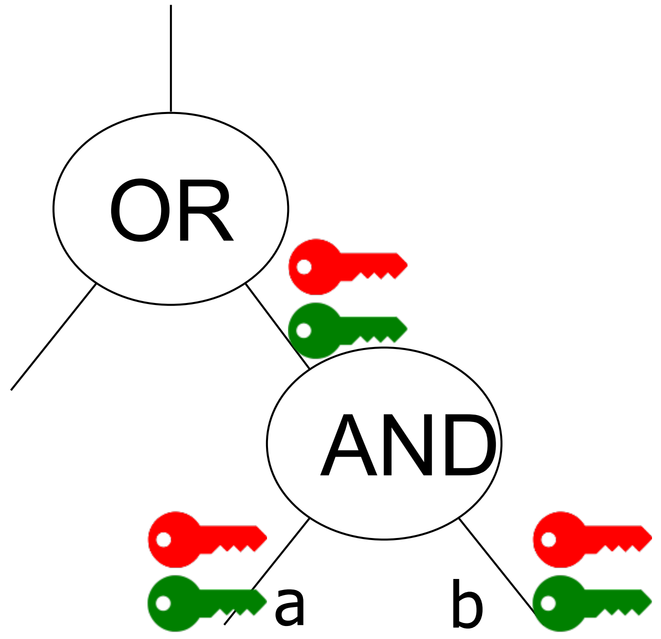


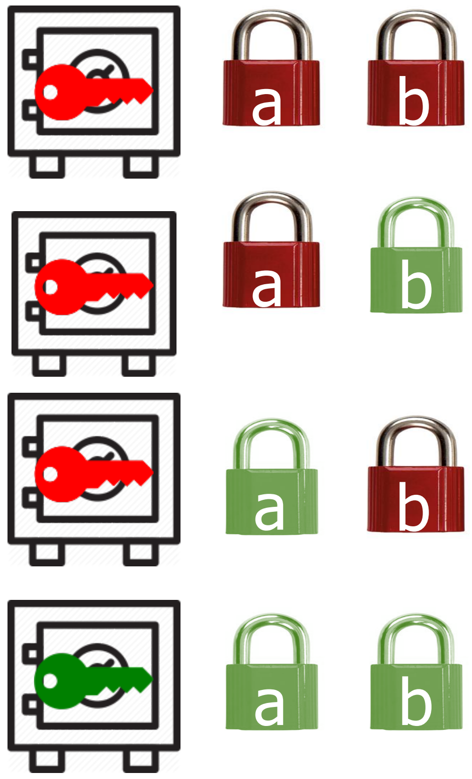




a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

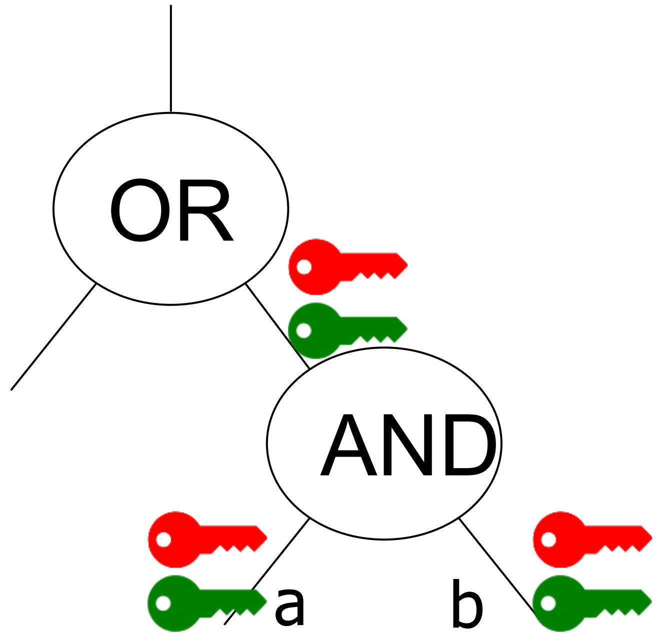
GC intuition: computing on encrypted values

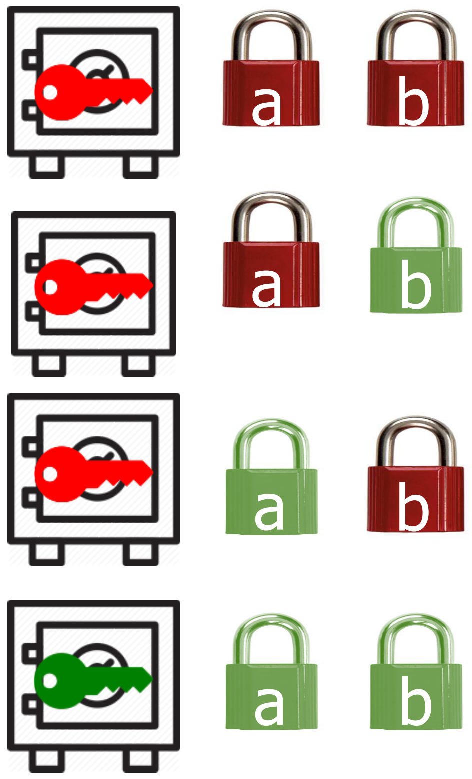




a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

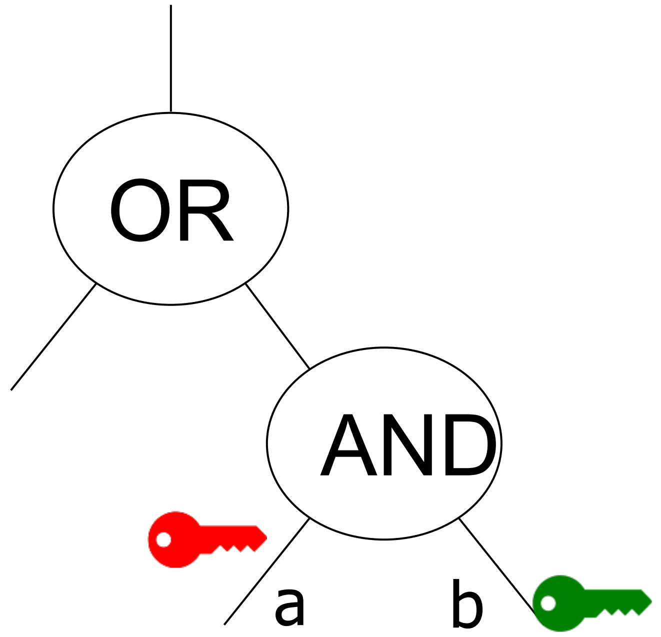
GC intuition: computing on encrypted values

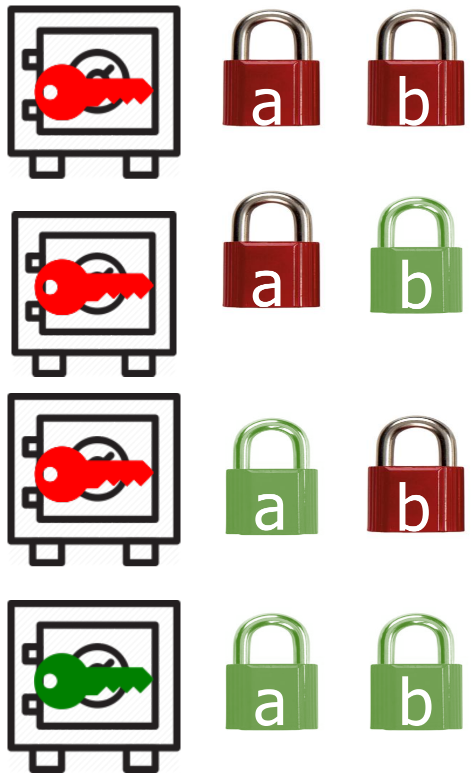




a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

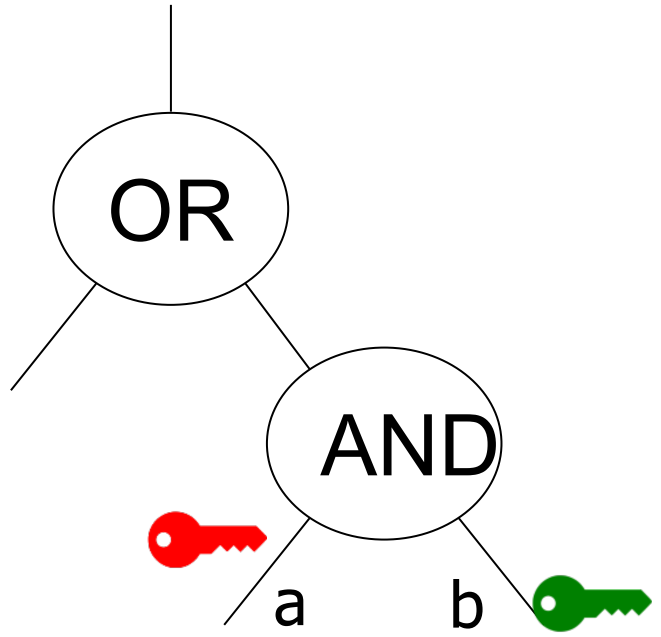
GC intuition: computing on encrypted values

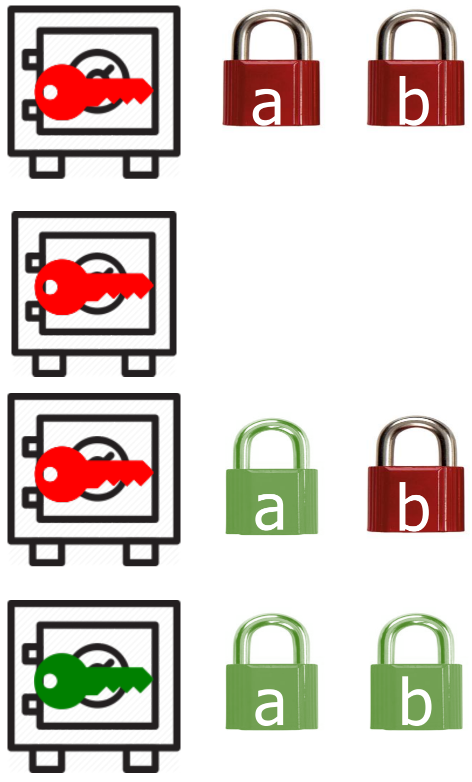




a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

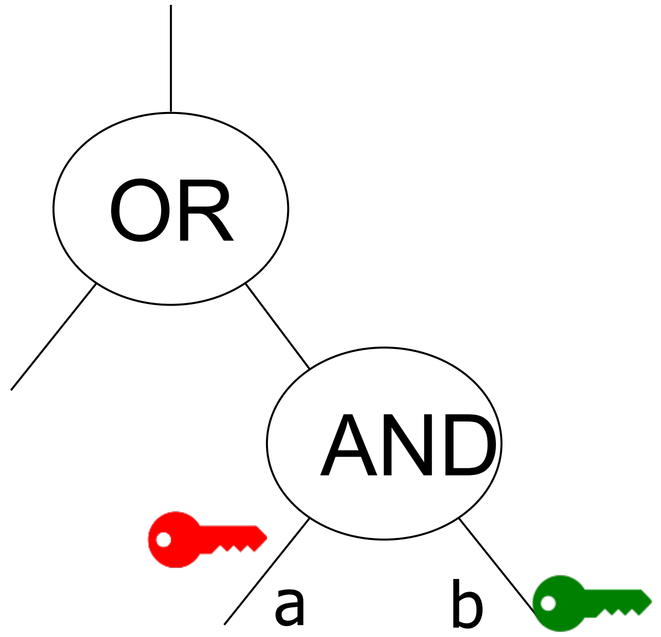
GC intuition: computing on encrypted values

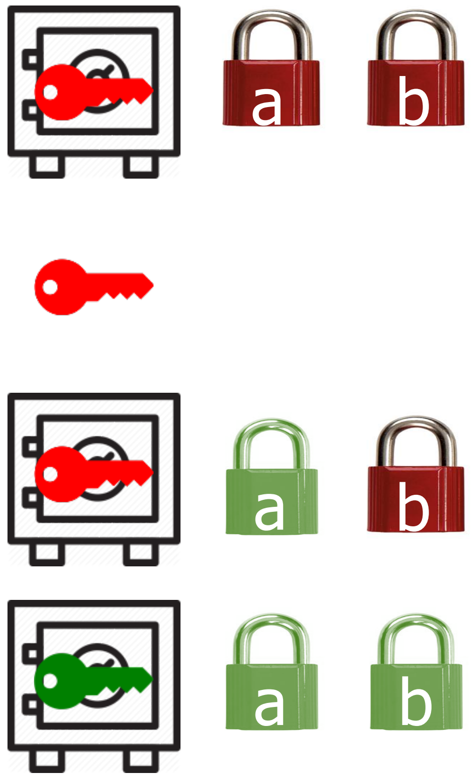




a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

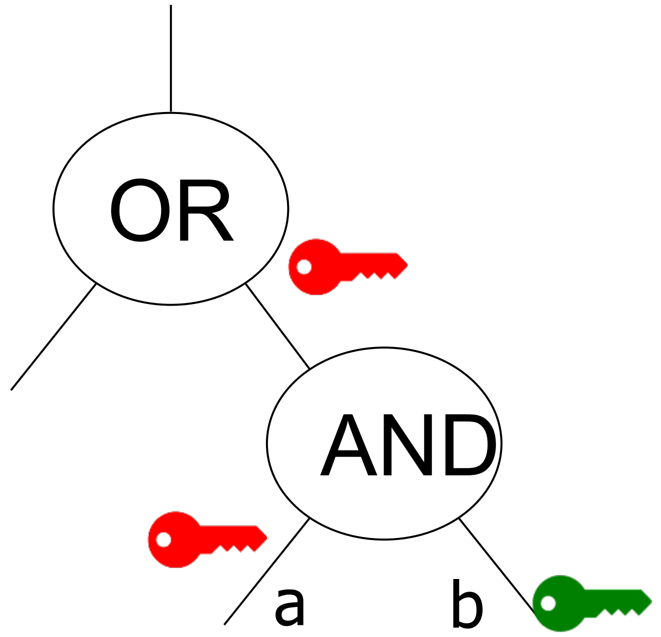
GC intuition: computing on encrypted values

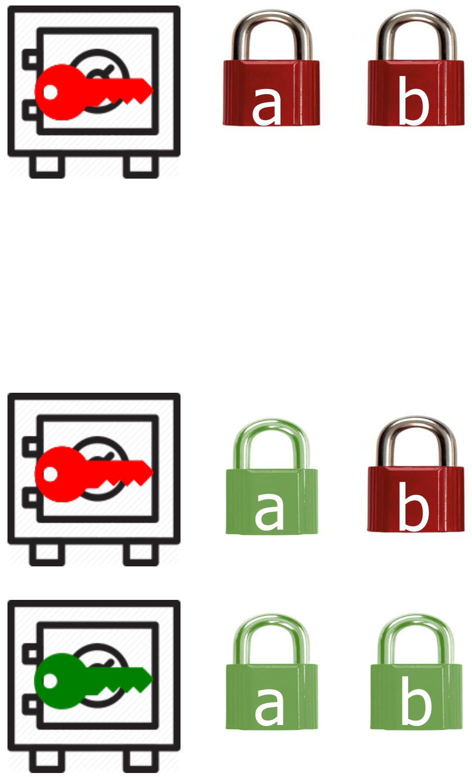




a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

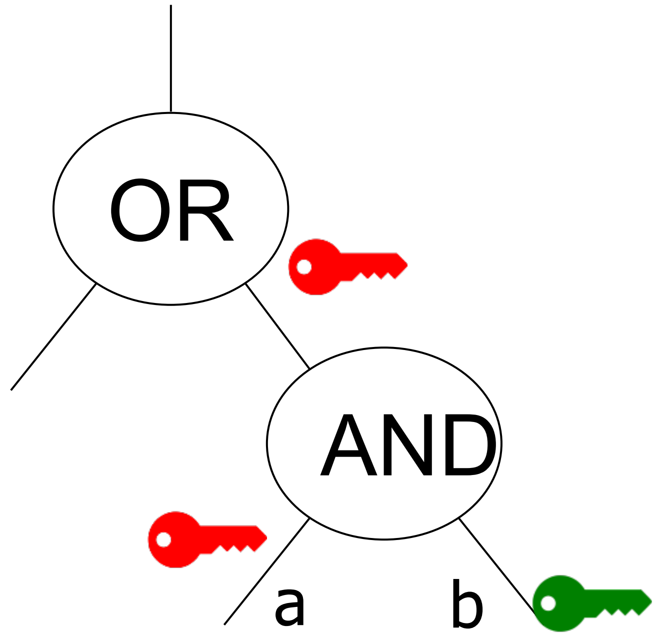
GC intuition: computing on encrypted values





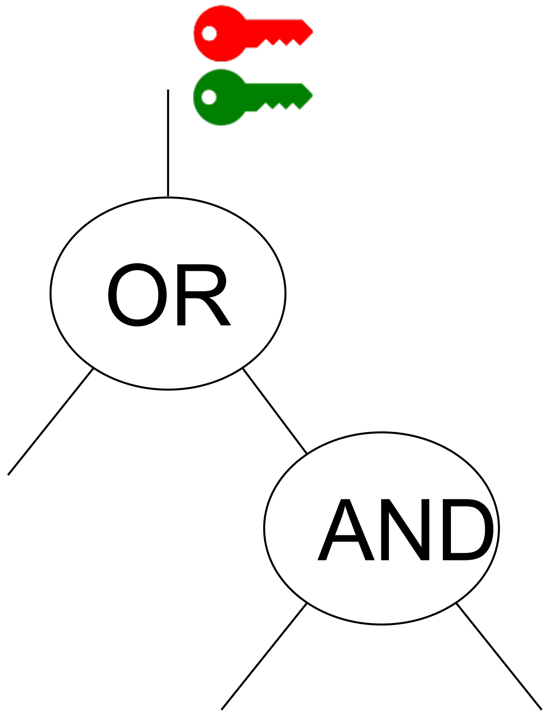
a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

GC intuition: computing on encrypted values



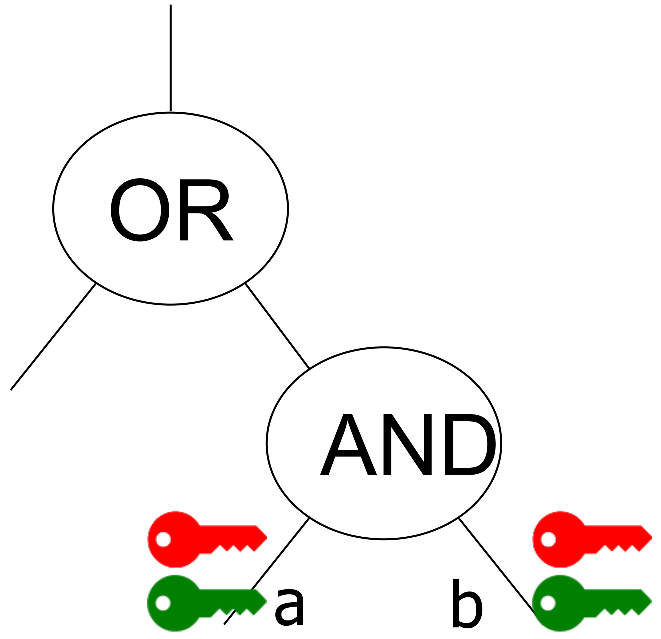
a	b	a^b
0	0	0
0	1	0
1	0	0
1	1	1

GC intuition: decoding encrypted output

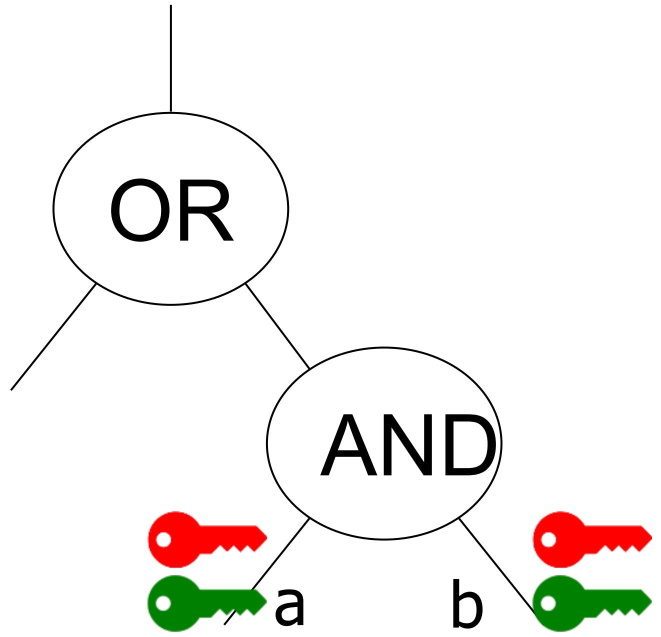


🟥	0
🟩	1

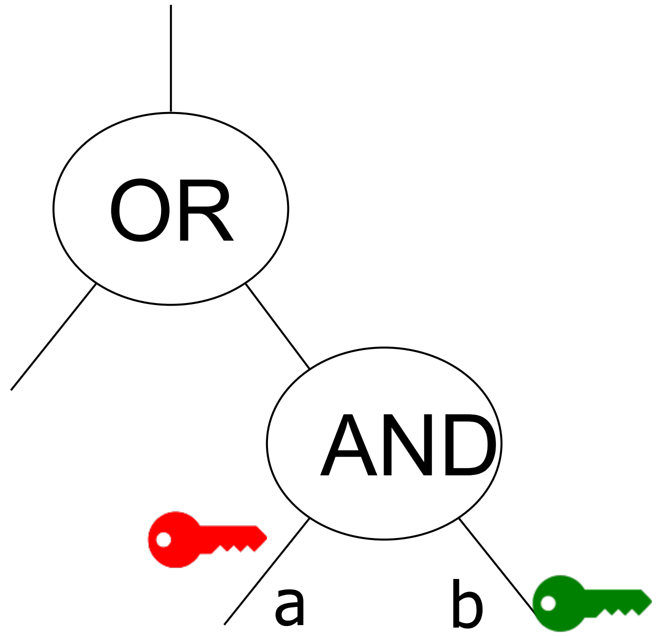
GC intuition: OT for transferring input labels



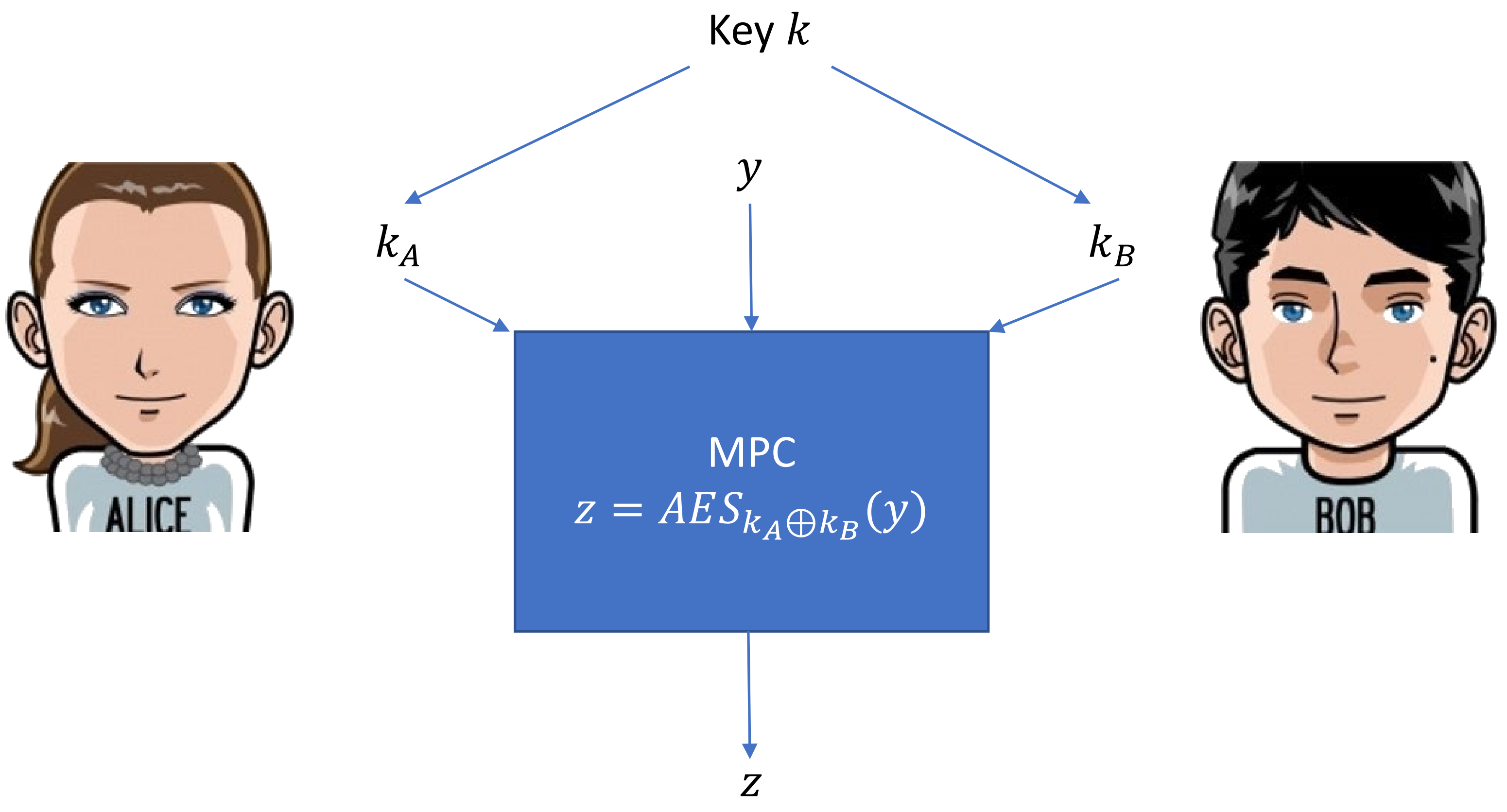
GC intuition: OT for transferring input labels



GC intuition: OT for transferring input labels



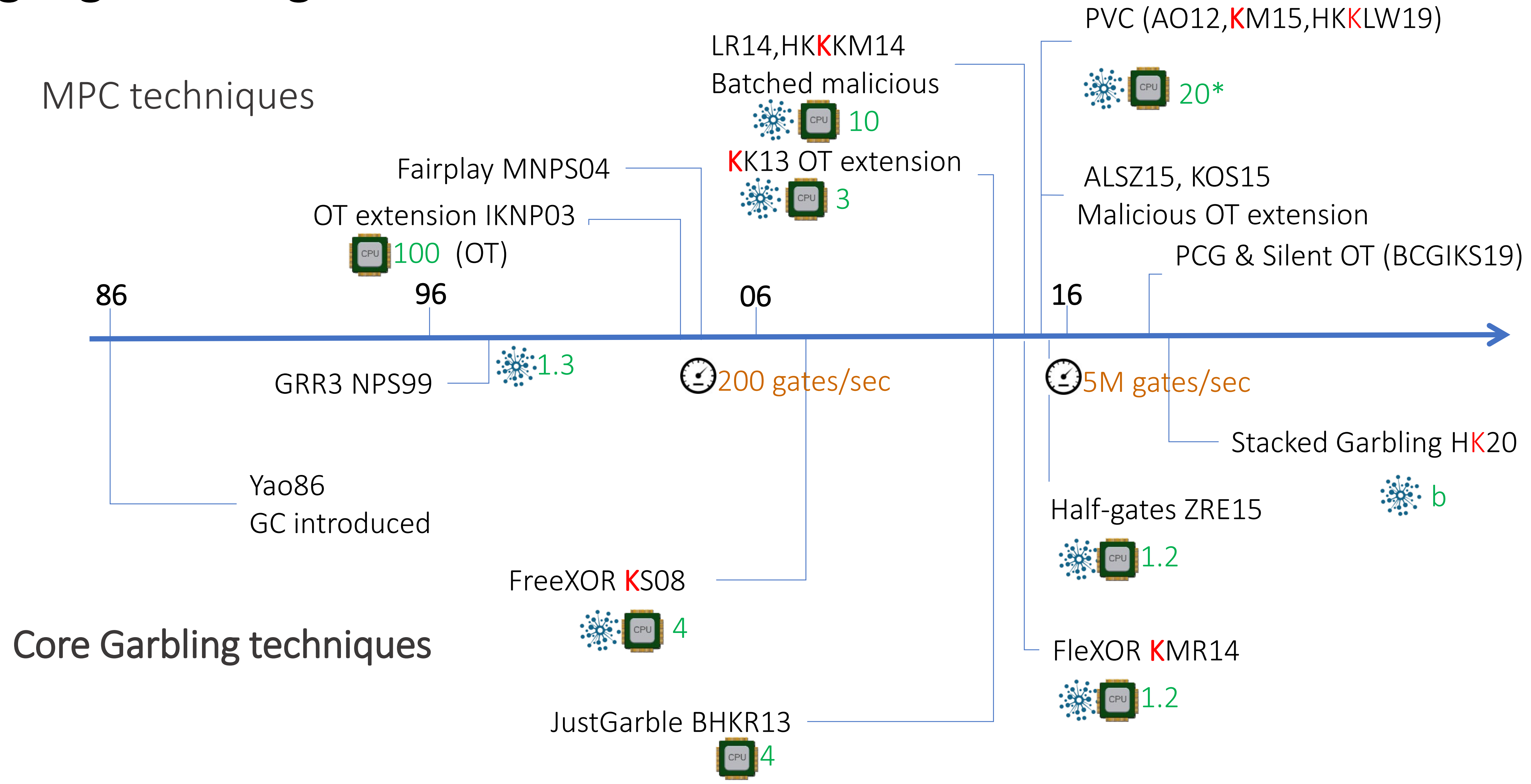
Applications to threshold cryptography



Of course, a number of variations are possible. Efficiency depends mostly on the size of the computed circuit.

Garbled circuits are pretty stable

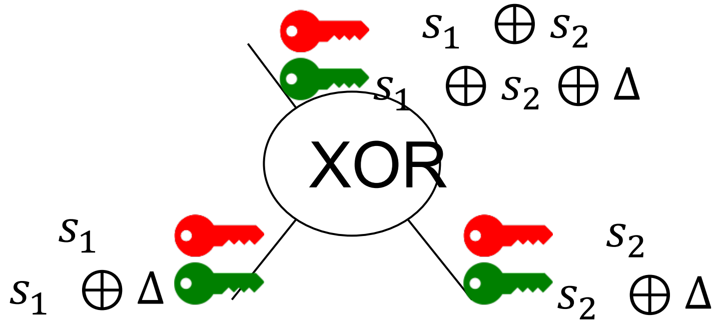
Highlights of algorithmic GC advances



A sample of GC advances

Free XOR [K05,KS08]

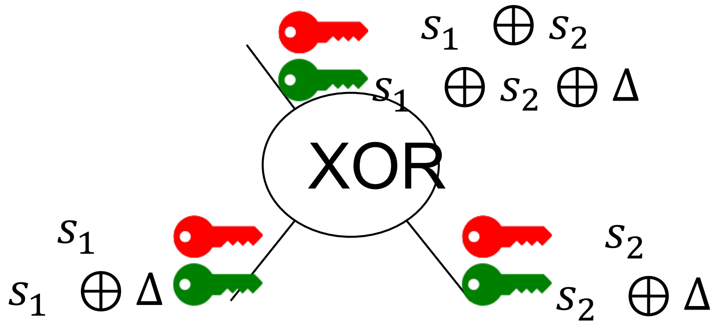
- [K05] Information-theoretic garbled circuit:
  - Based on secret sharing/reconstruction
  - **XOR gates are free (no tables)**
  - Wire secrets are not independent



Free XOR [K05,KS08]

Free XOR [KS08]

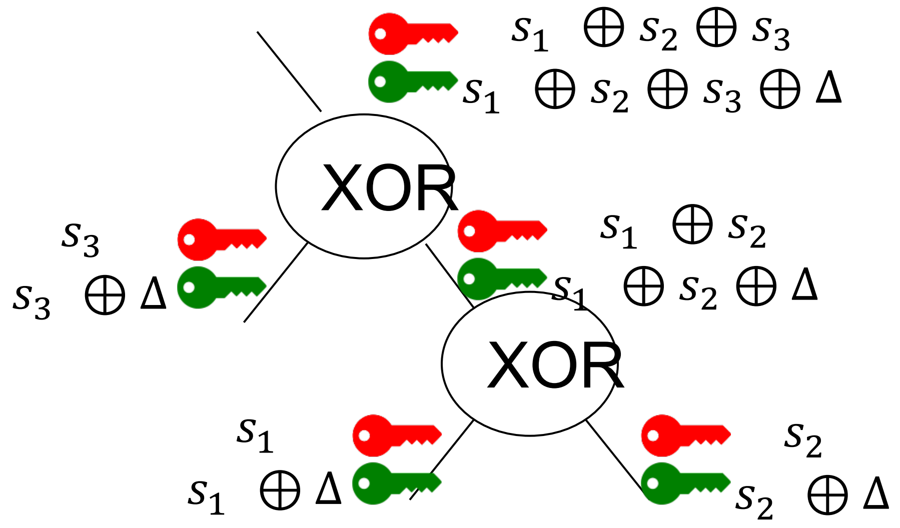
- Choose same Δ for entire circuit
  - Show that OK to have related keys
- All XOR gates free
- Stronger encryption required for other gates



Free XOR [K05,KS08]

Free XOR [KS08]

- Choose same Δ for entire circuit
    - Show that OK to have related keys
- All XOR gates free
- Stronger encryption required for other gates

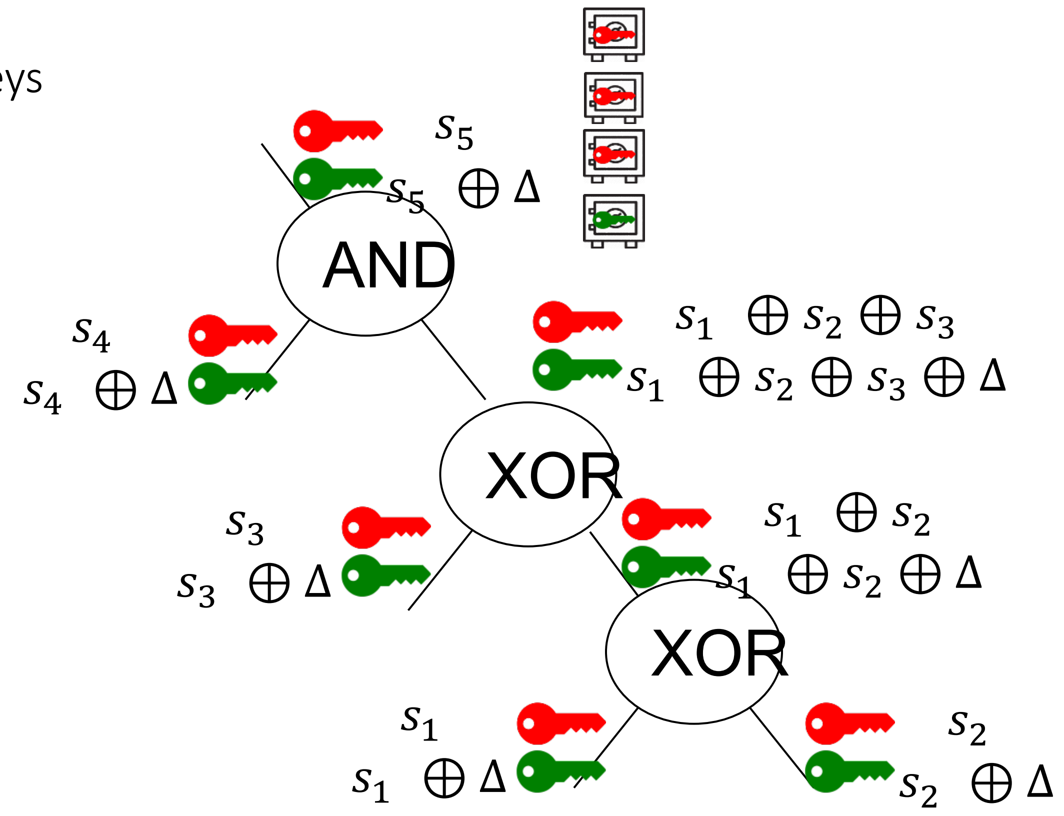


$s_1 \oplus s_2 \oplus s_3$
$s_1 \oplus s_2 \oplus s_3 \oplus \Delta$
XOR
s_3
$s_3 \oplus \Delta$
$s_1 \oplus s_2$
$s_1 \oplus s_2 \oplus \Delta$
XOR
s_1
$s_1 \oplus \Delta$
s_2
$s_2 \oplus \Delta$

Free XOR [K05,KS08]

Free XOR [KS08]

- Choose same Δ for entire circuit
  - Show that OK to have related keys
- All XOR gates free
- Stronger encryption required for other gates



Stacked garbling [HK20]

Stacked garbling [HK20]

- Sequence of works [K18,HK20a,HK20b]

Stacked garbling [HK20]

- Sequence of works [K18,HK20a,HK20b]
- Let’s question the circuit model of computation.

Stacked garbling [HK20]

- Sequence of works [K18,HK20a,HK20b]
- Let’s question the circuit model of computation.
- But not too much..

Stacked garbling [HK20]

- Sequence of works [K18,HK20a,HK20b]
- Let’s question the circuit model of computation.
- But not too much..
- Just consider circuits with conditionals

Stacked garbling [HK20]

- Sequence of works [K18,HK20a,HK20b]
- Let’s question the circuit model of computation.
- But not too much..
- Just consider circuits with conditionals

Let C0, C1 be two arbitrary circuits. The space of circuits is defined as follows:

Stacked garbling [HK20]

- Sequence of works [K18,HK20a,HK20b]
- Let’s question the circuit model of computation.
- But not too much..
- Just consider circuits with conditionals

Let C0, C1 be two arbitrary circuits. The space of circuits is defined as follows:

$$C ::= \mathrm{Netlist}(\cdot) \mid \mathrm{Cond}(C0, C1) \mid \mathrm{Seq}(C0, C1)$$

Stacked garbling [HK20]

$$C ::= \mathsf{Netlist}(\cdot) \mid \mathsf{Cond}(C0, C1) \mid \mathsf{Seq}(C0, C1)$$

HK20: Can evaluate $\mathsf{Cond}(C0, C1)$ while transmitting only one branch

Idea:

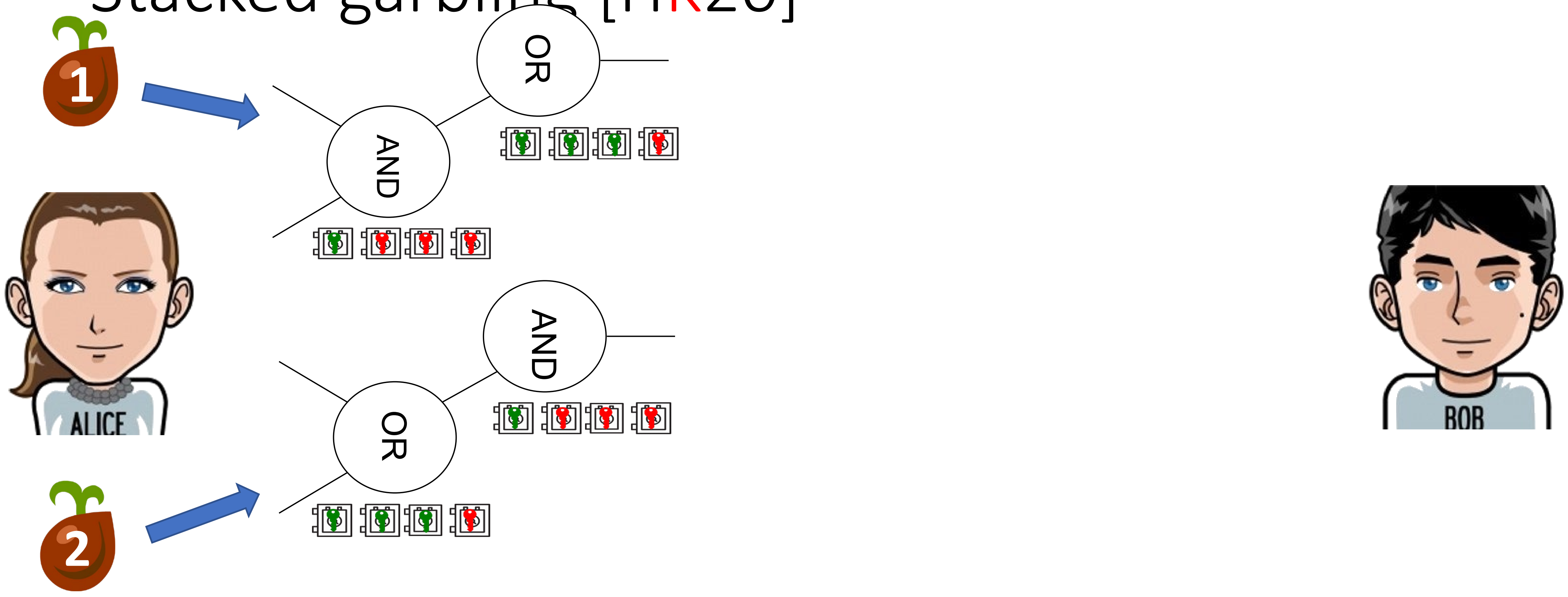
* the same GC *material* M is used for evaluation of $C0$ and $C1$.

* GC outputs a key to Eval which converts material M to a valid GC or to a random-looking string for inactive branch

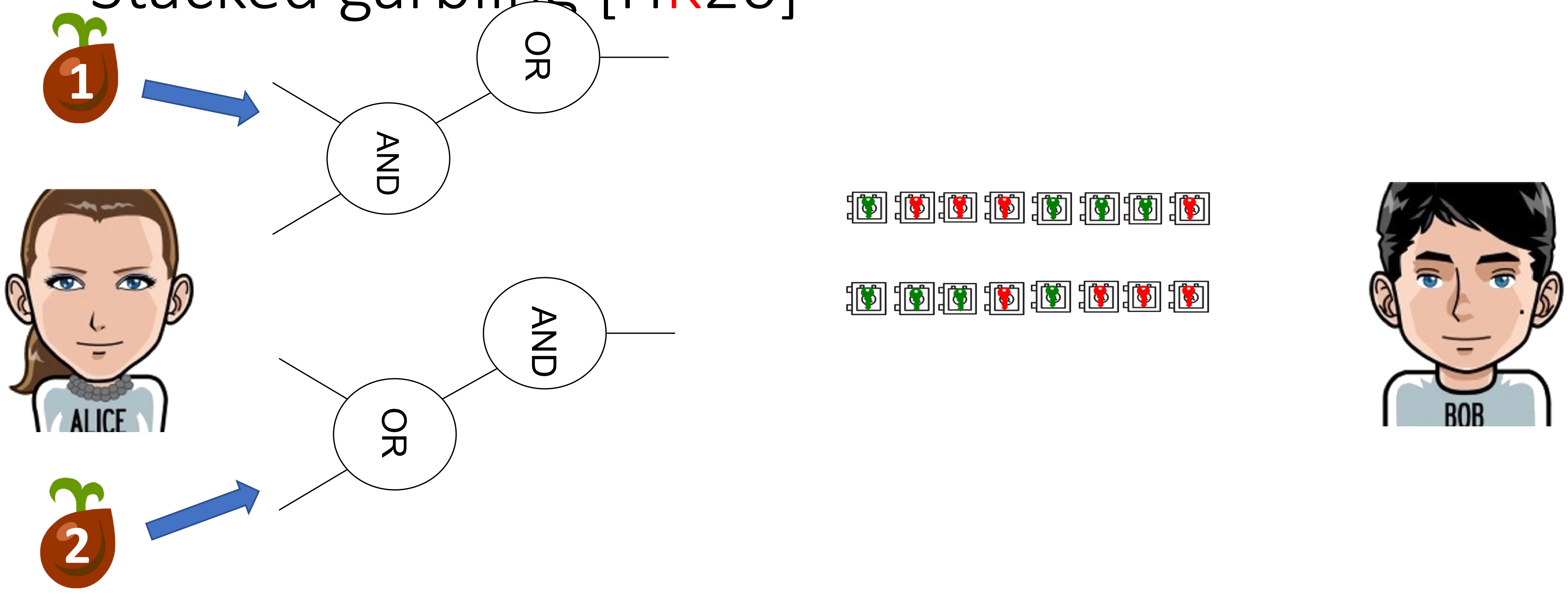
* Eval evaluates both $C0, C1$. One of them will produce garbage labels. They are canceled (garbage-collected) by gadgets constructed by Garbler.

* Material reuse (novel general idea; works for other protocols as well)

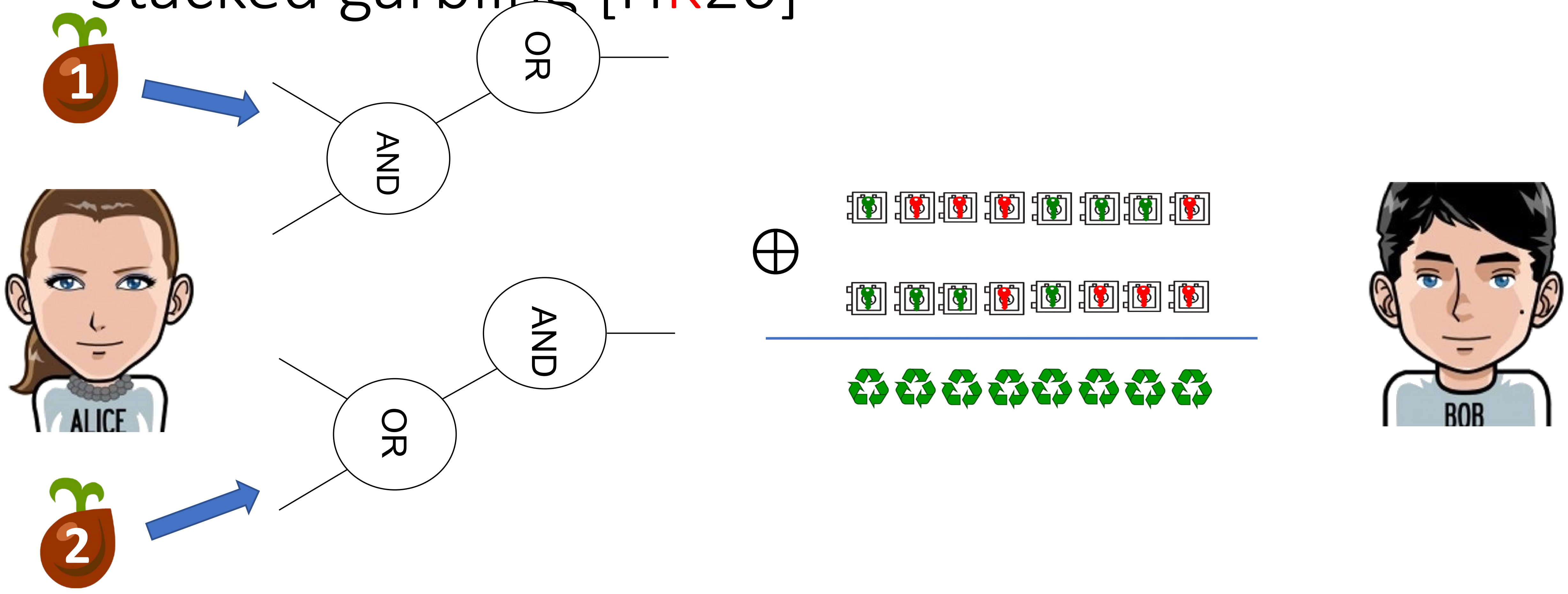
Stacked garbling [HK20]



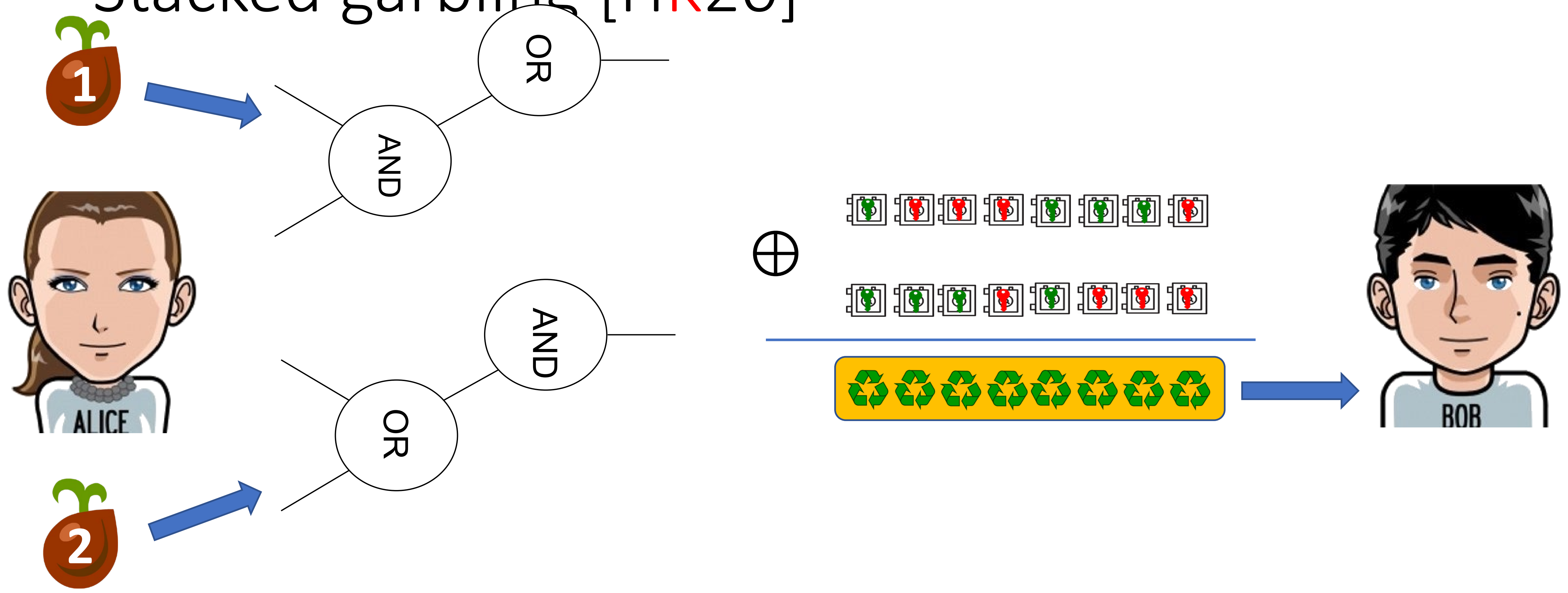
Stacked garbling [HK20]



Stacked garbling [HK20]

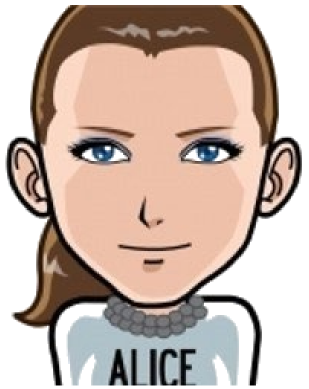


Stacked garbling [HK20]

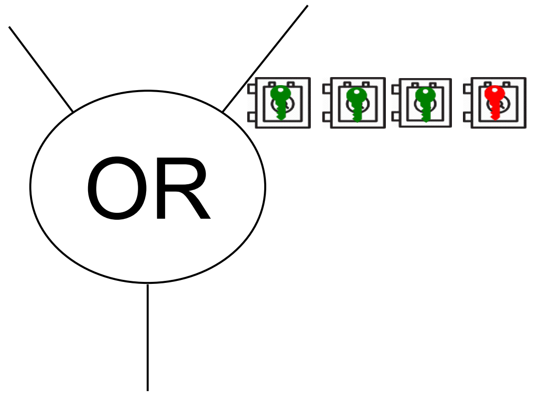


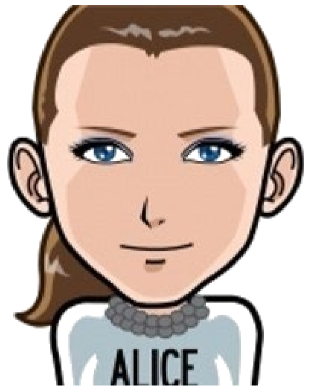
Stacked garbling [HK20]

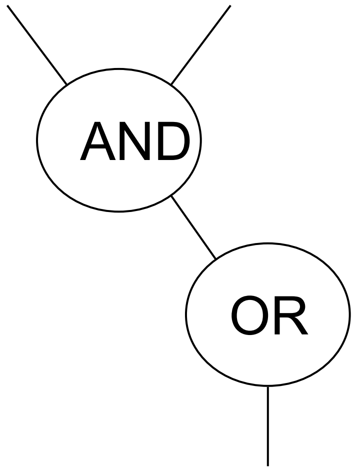
♻♻♻♻♻♻♻♻

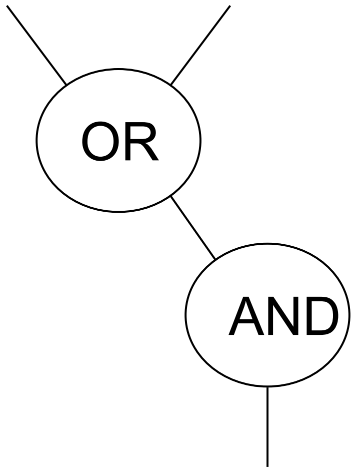


Stacked garbling [HK20]

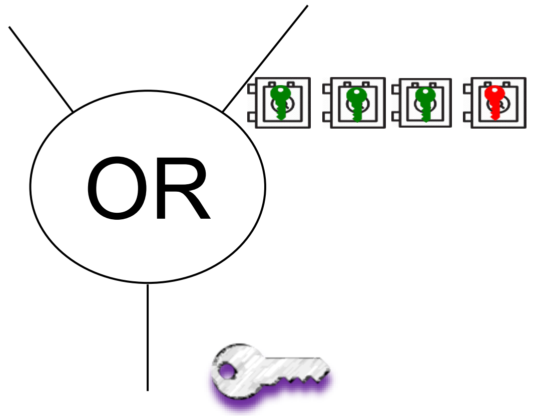


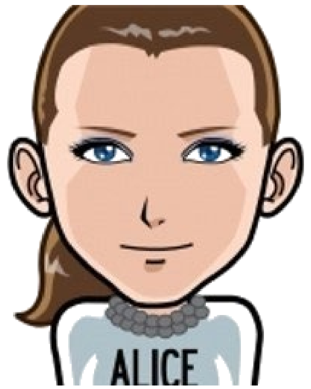


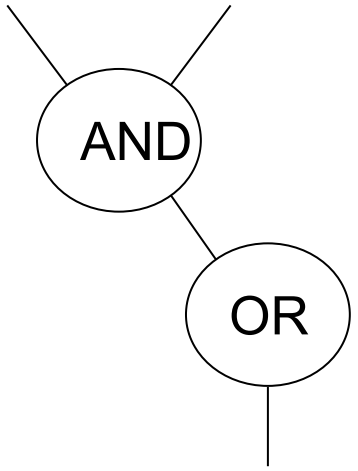


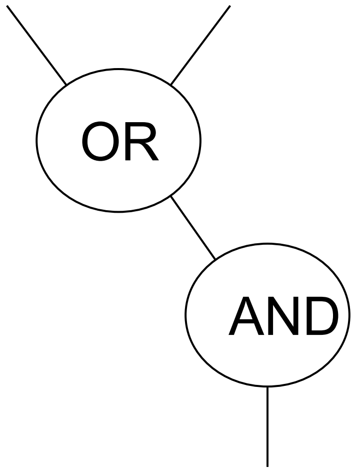


Stacked garbling [HK20]

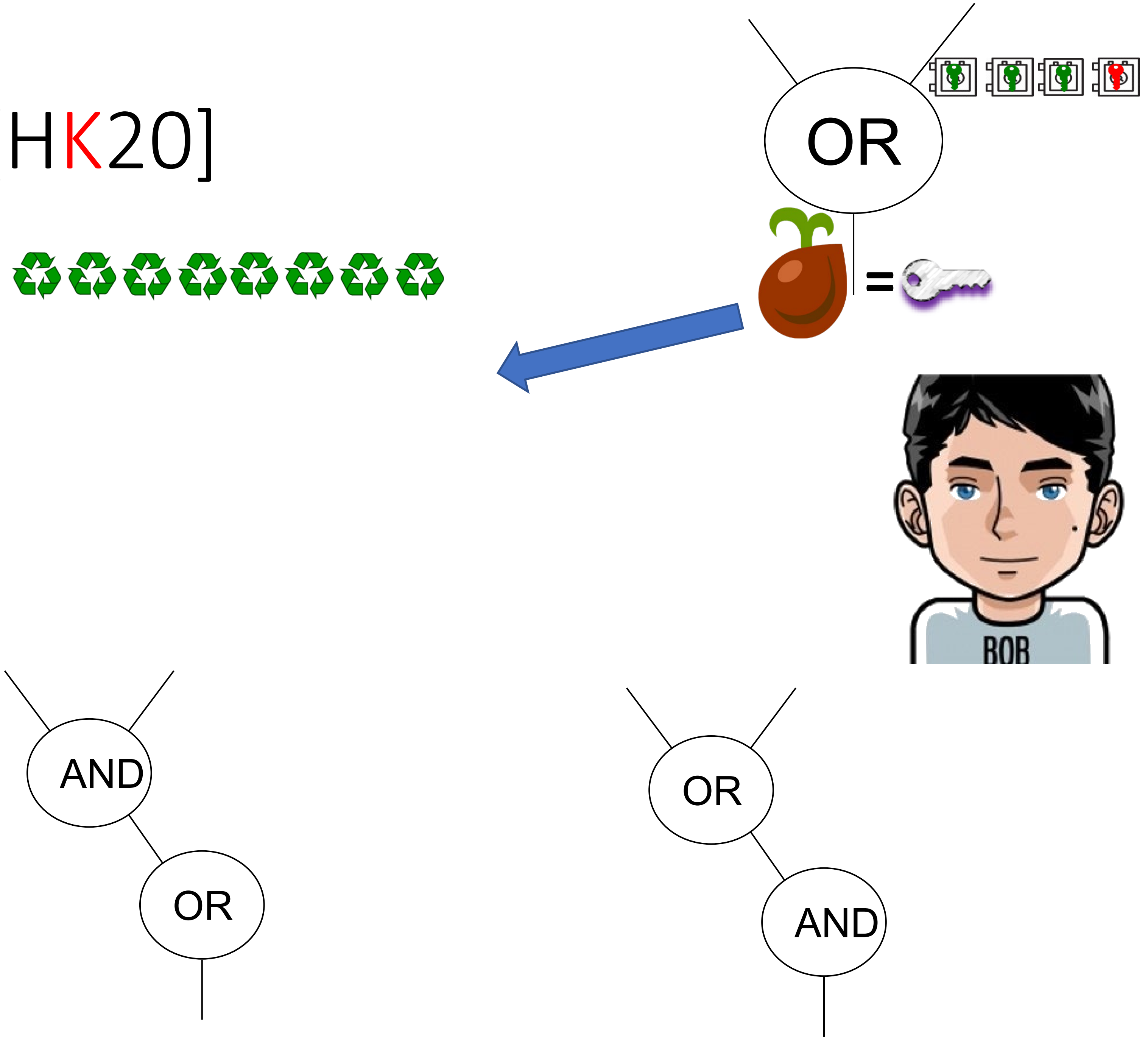


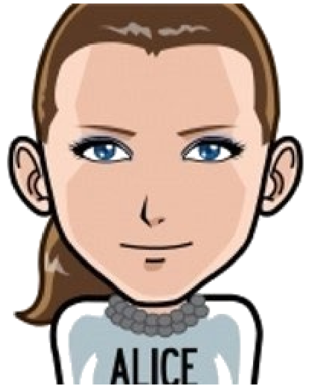




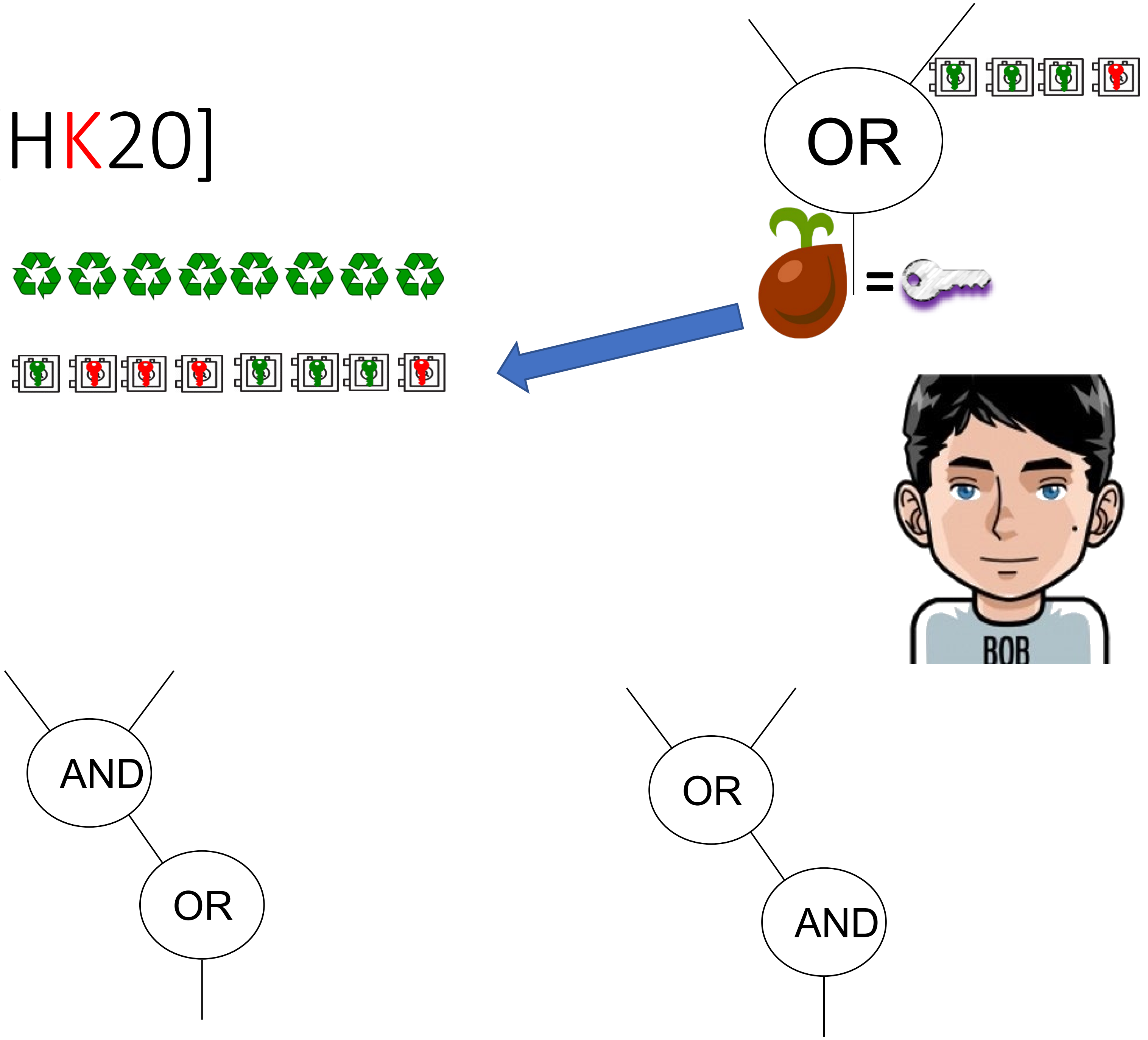


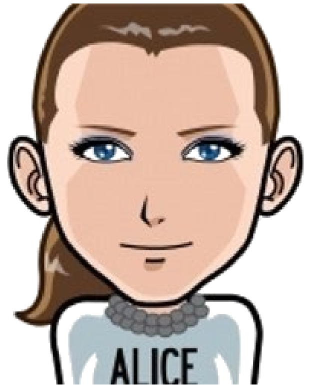
Stacked garbling [HK20]



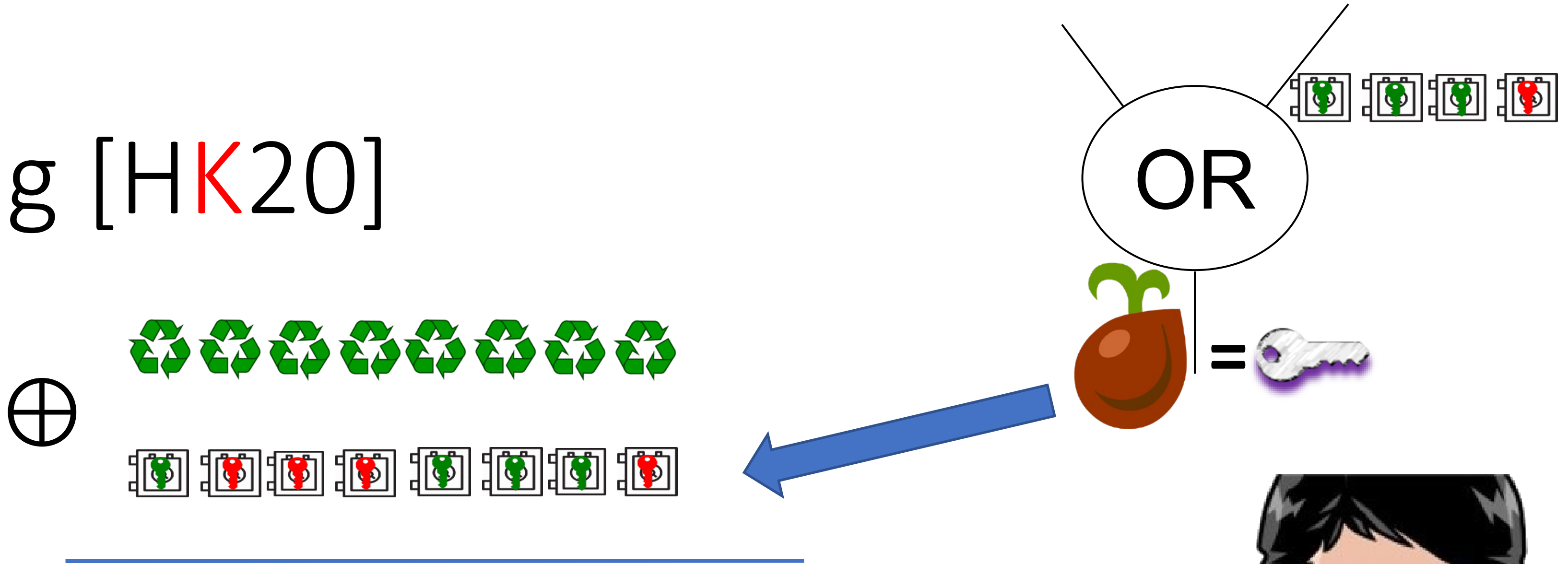


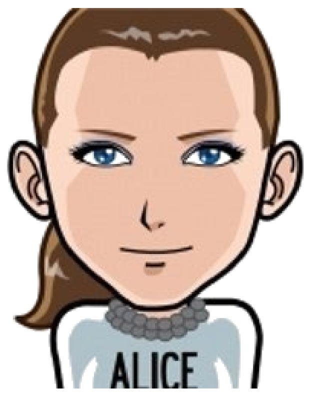
Stacked garbling [HK20]

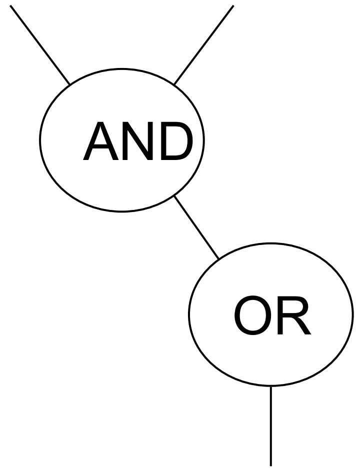


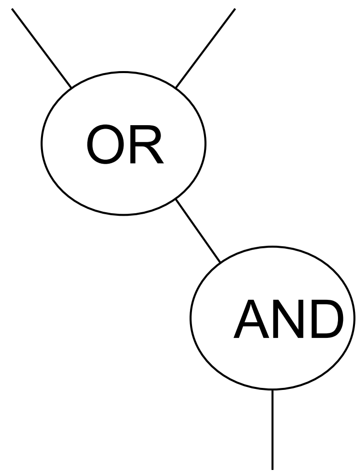


Stacked garbling [HK20]

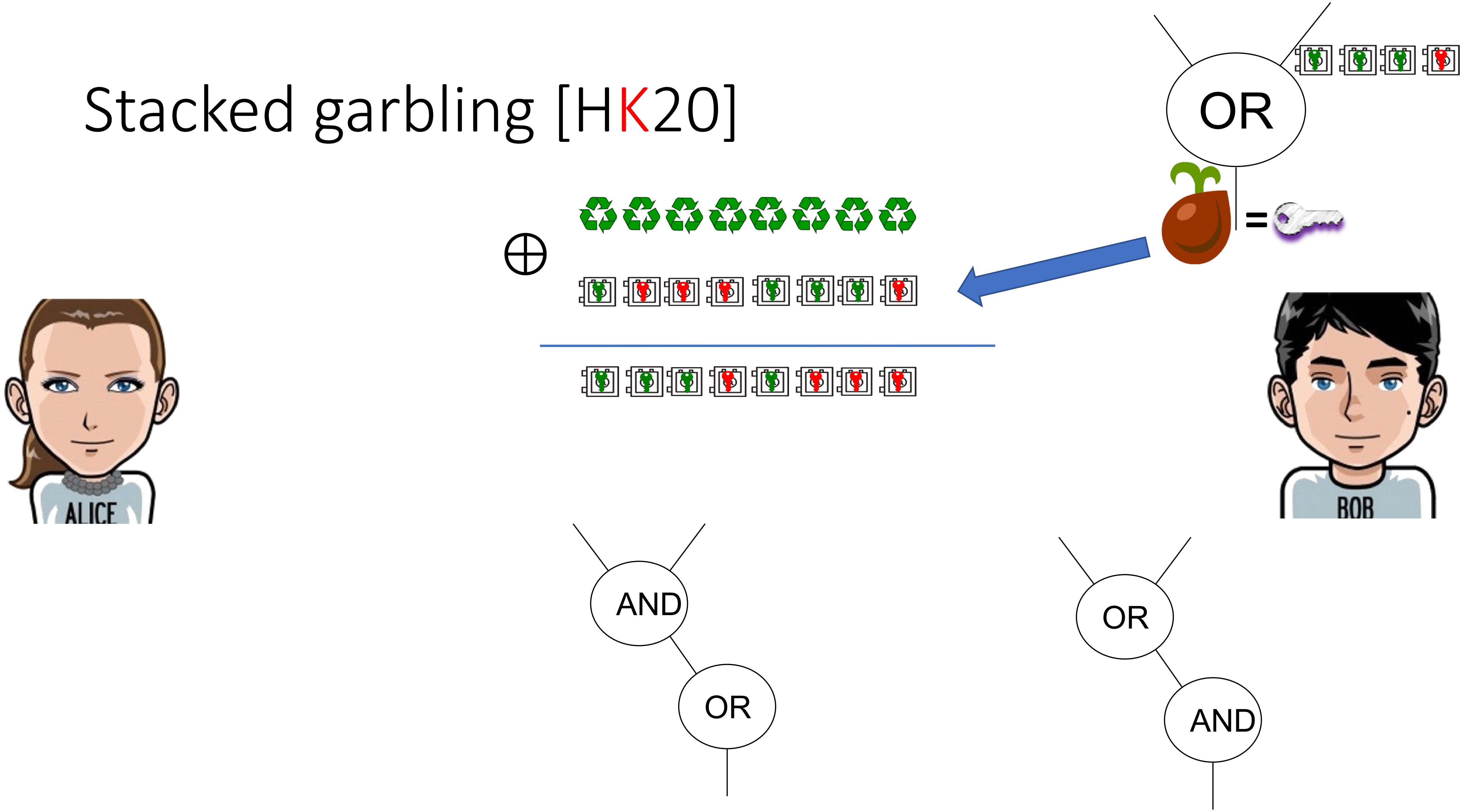




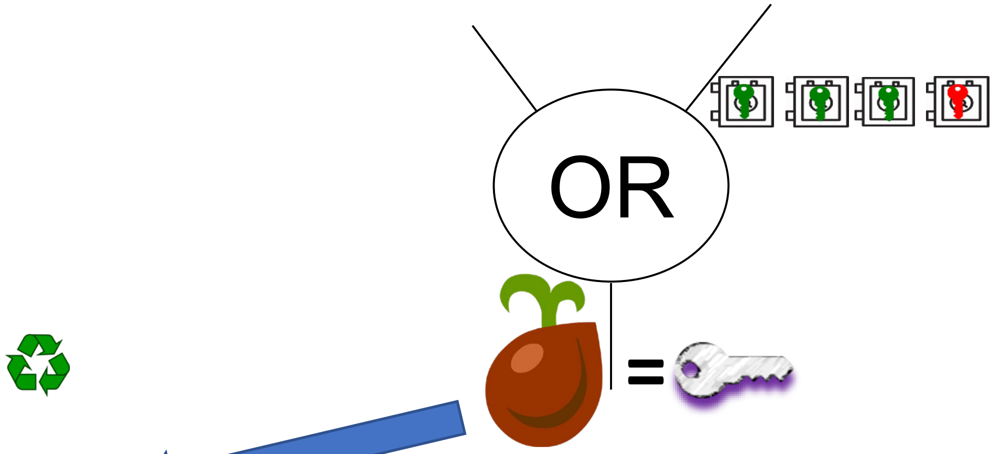


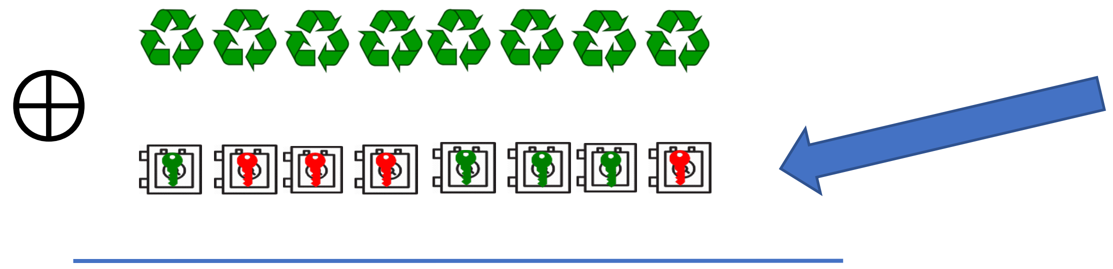


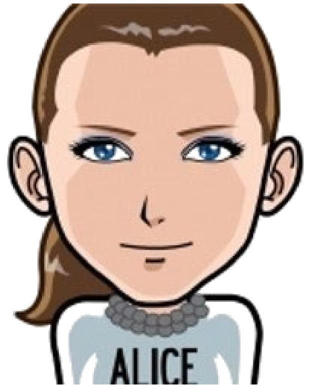
Stacked garbling [HK20]

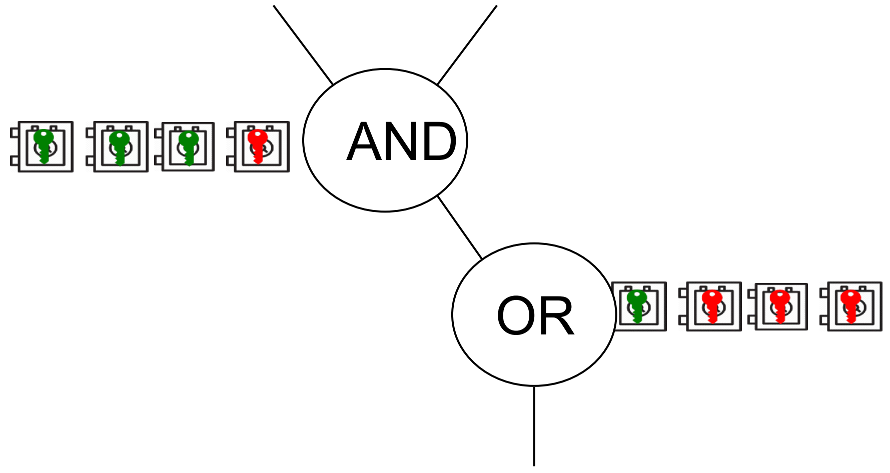


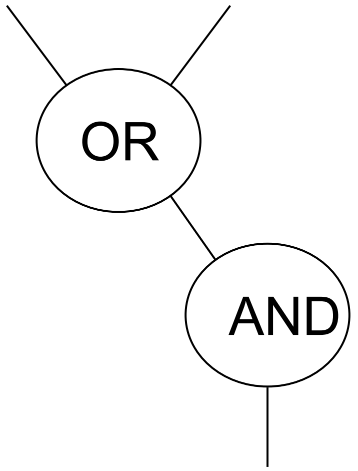
Stacked garbling [HK20]



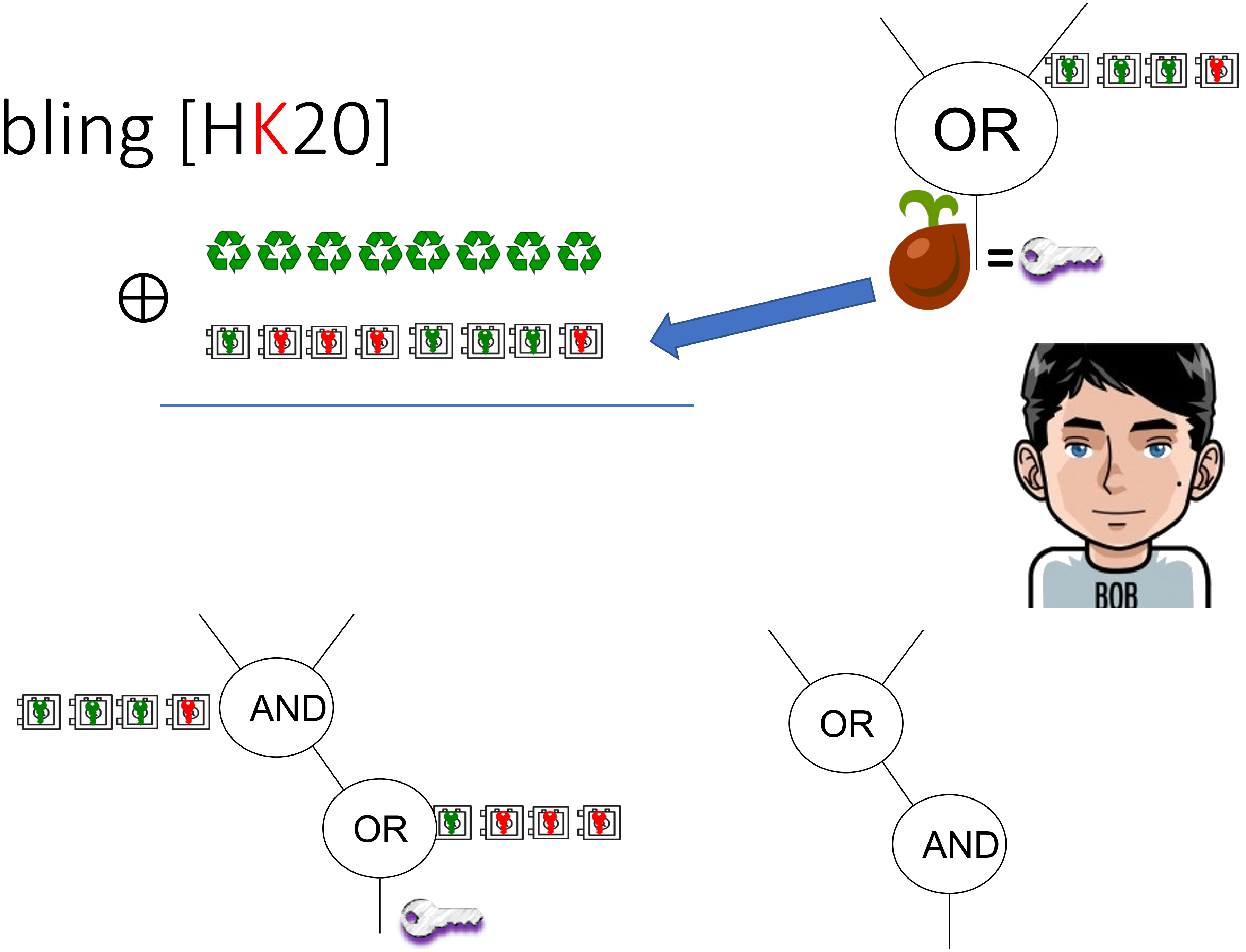




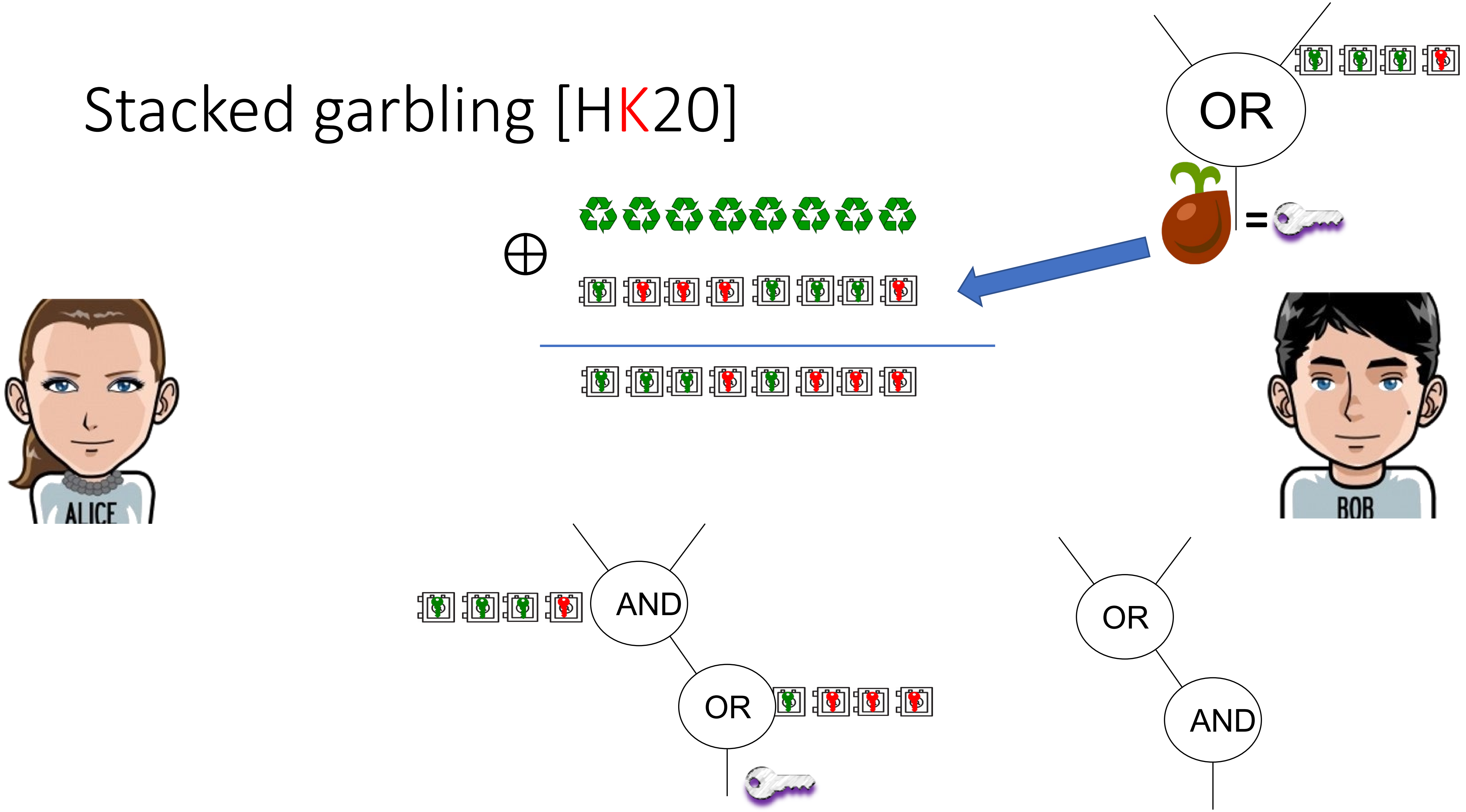




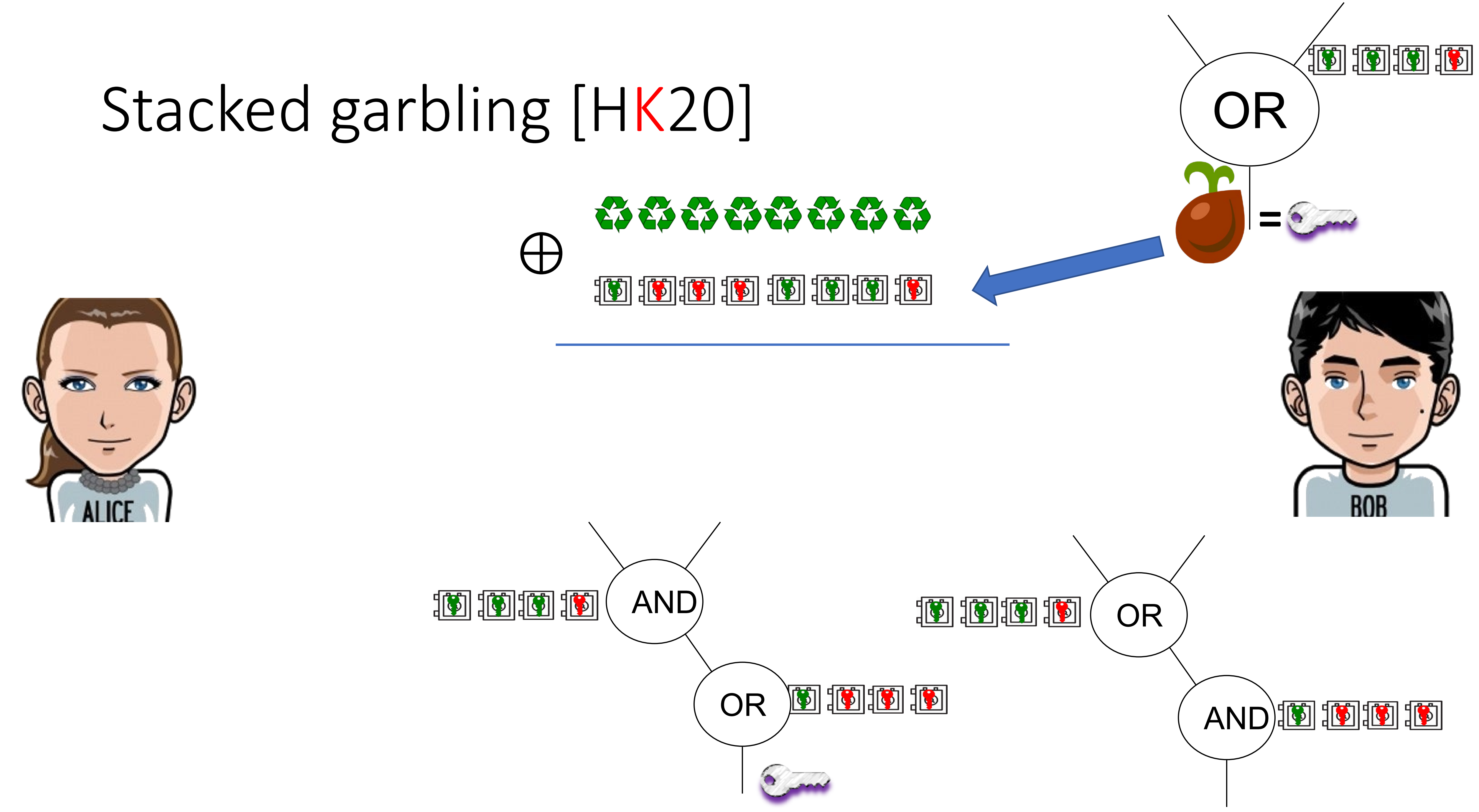
Stacked garbling [HK20]



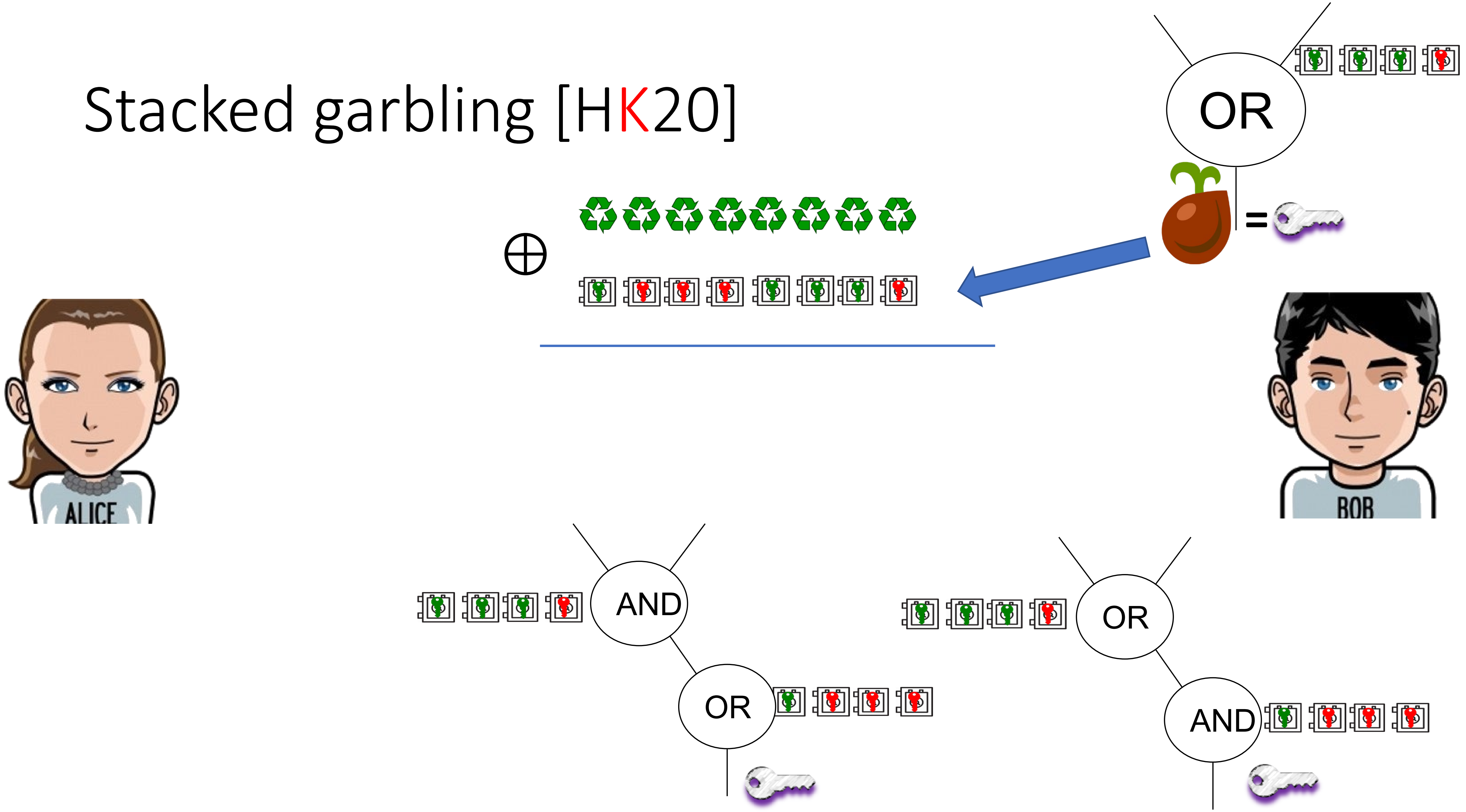
Stacked garbling [HK20]



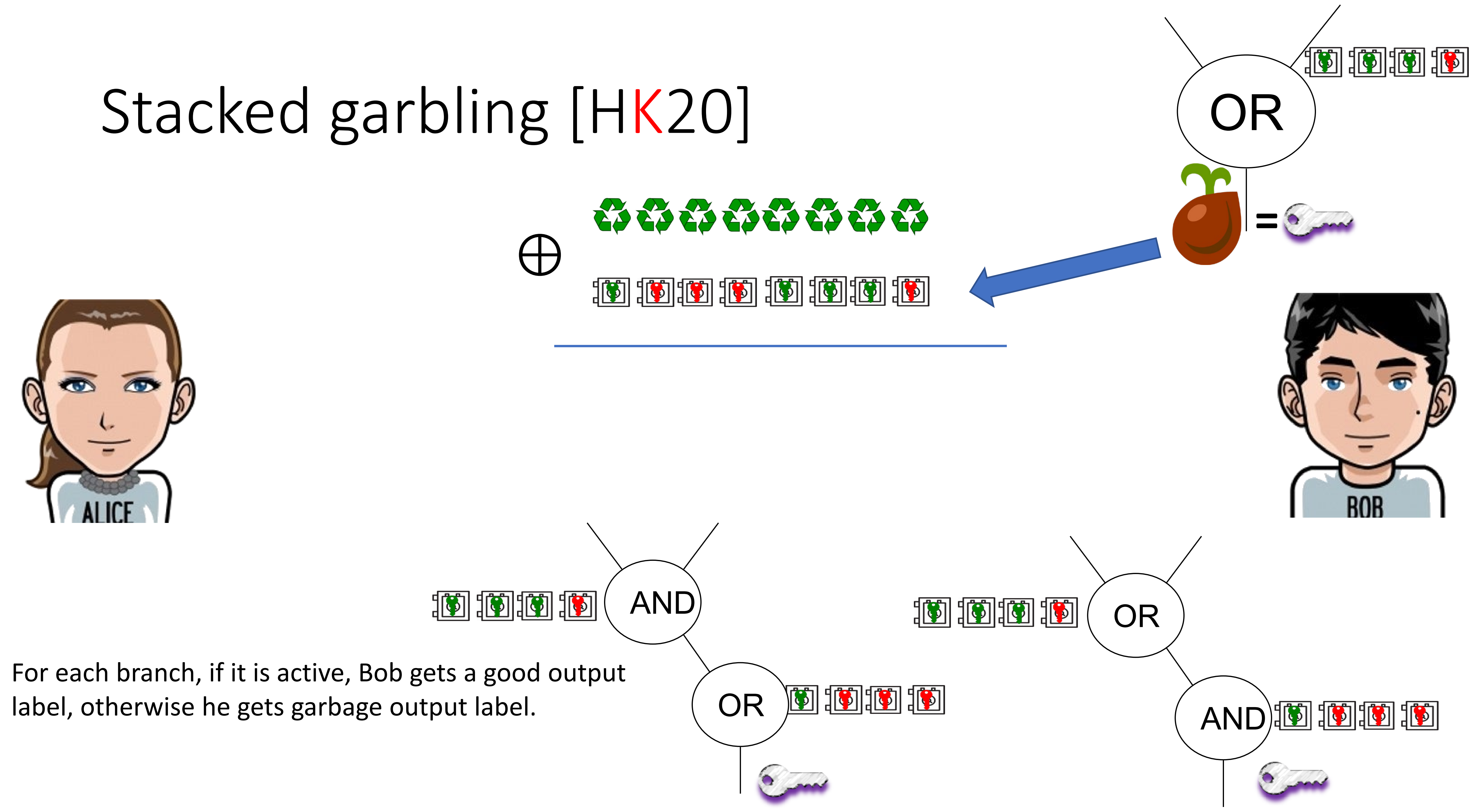
Stacked garbling [HK20]



Stacked garbling [HK20]

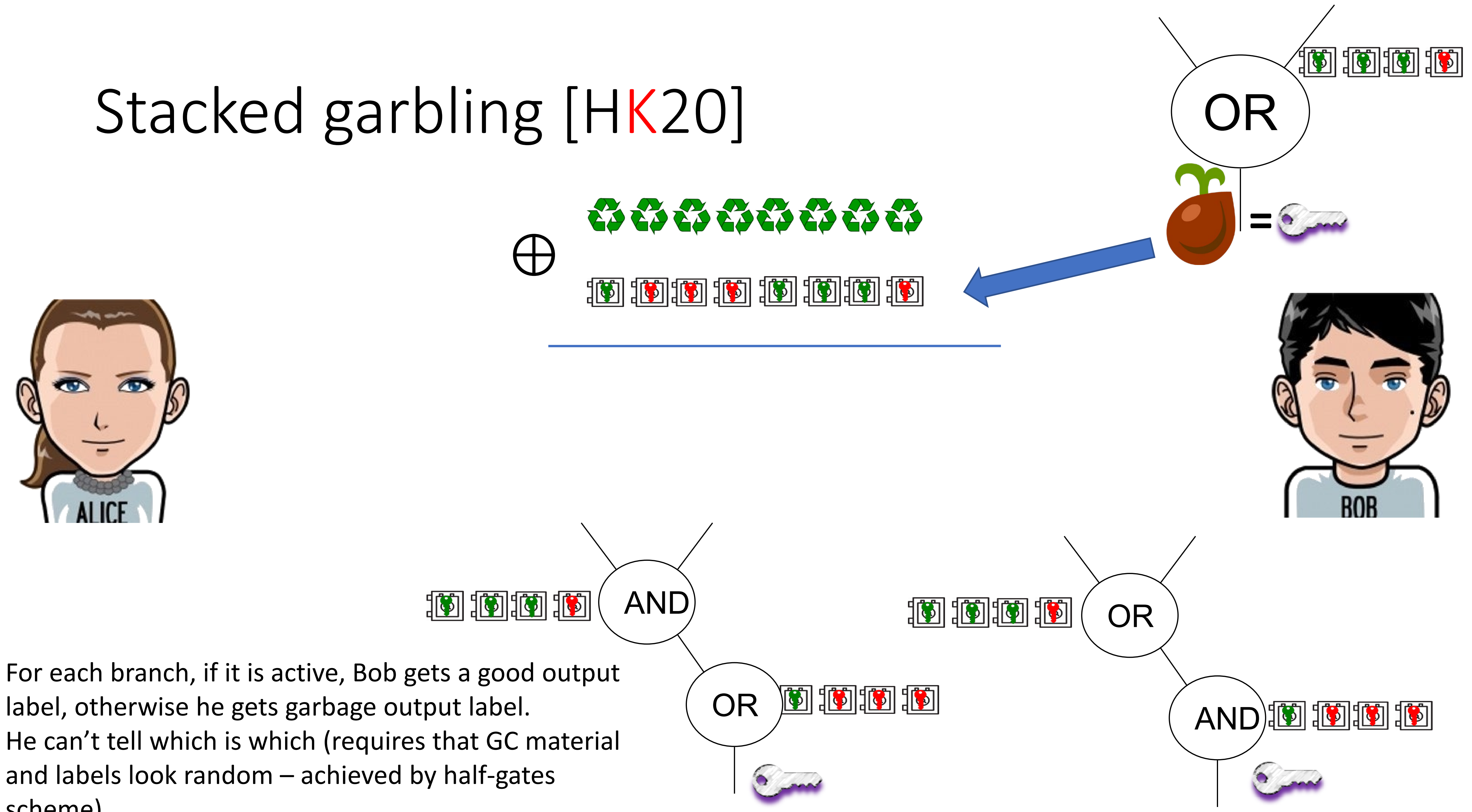


Stacked garbling [HK20]



For each branch, if it is active, Bob gets a good output label, otherwise he gets garbage output label.

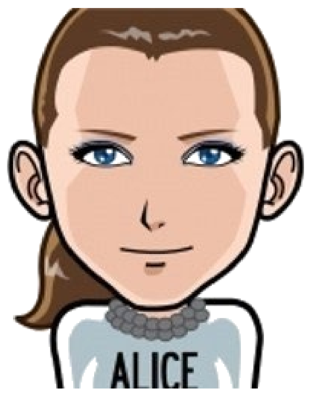
Stacked garbling [HK20]

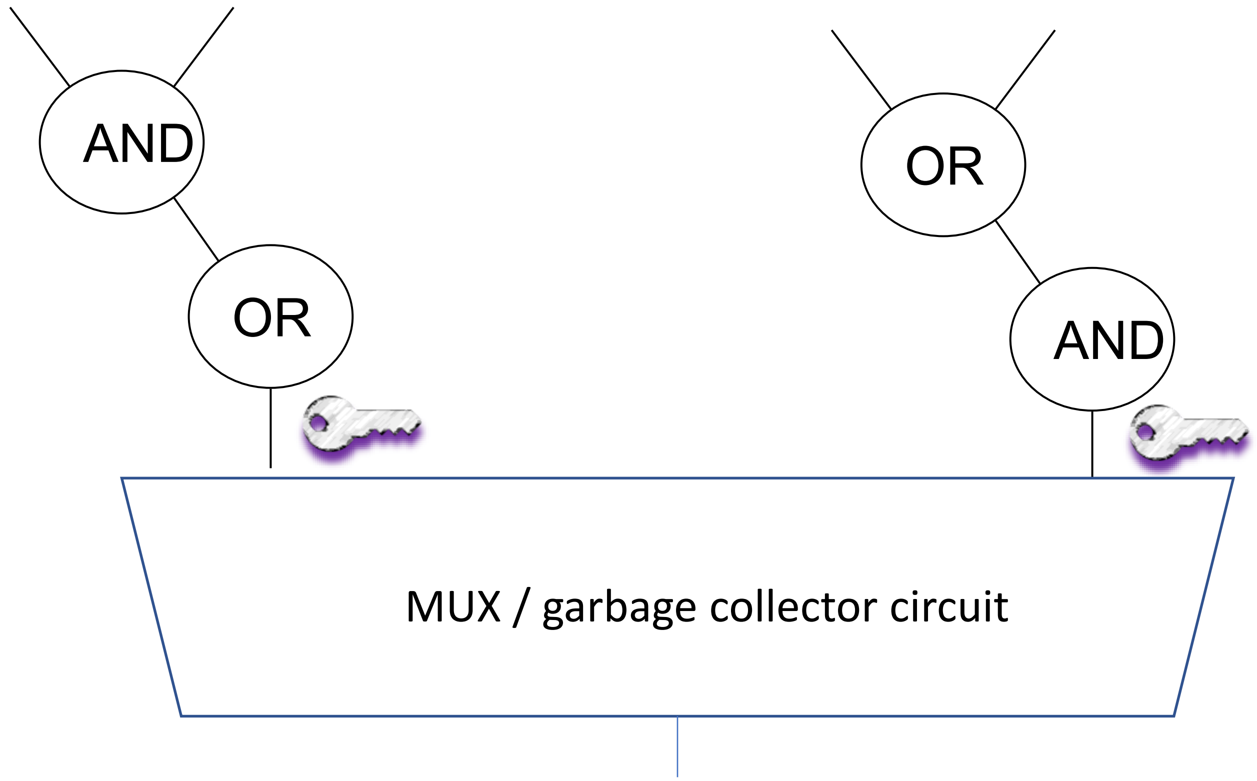


For each branch, if it is active, Bob gets a good output label, otherwise he gets garbage output label.
He can't tell which is which (requires that GC material and labels look random – achieved by half-gates scheme)

Stacked garbling [HK20]

For active branch, Bob gets a valid label, otherwise he gets garbage output label.





We need to obliviously discard garbage.
Key idea: Bob is deterministic and Alice can emulate him and *predict* the possible garbage keys
Then Alice constructs a MUX gadget which collects garbage.

GC is basic

- It is a simple object; it is not a protocol
- Standardizing just GC gives cryptographic object with clean security properties.
- Optional OT/GC usage standardization makes is a secure MPC standard

GC standardization

- Don’t need full generality of GC (such a version of BHR)
  - Half-gates with free XOR is a de-facto standard
- Fix the underlying cipher used for encryption
- Important features (incomplete list):
  - GC is projective (a label corresponds to a wire value)
  - Labels and GC material look random (required for SGC)
  - Perfect correctness (e.g. via point-and-permute)

  - Half-gates meets all these requirements

GC standardization

GC is very stable.

Standardizing basic GC

- Not likely to hinder future algorithmic enhancements
- Will greatly aid in Threshold crypto (mandate of this group),
  - and be a catalyst for MPC development *and adoption.*

So let's go!