DAP – Supporting heavy hitters

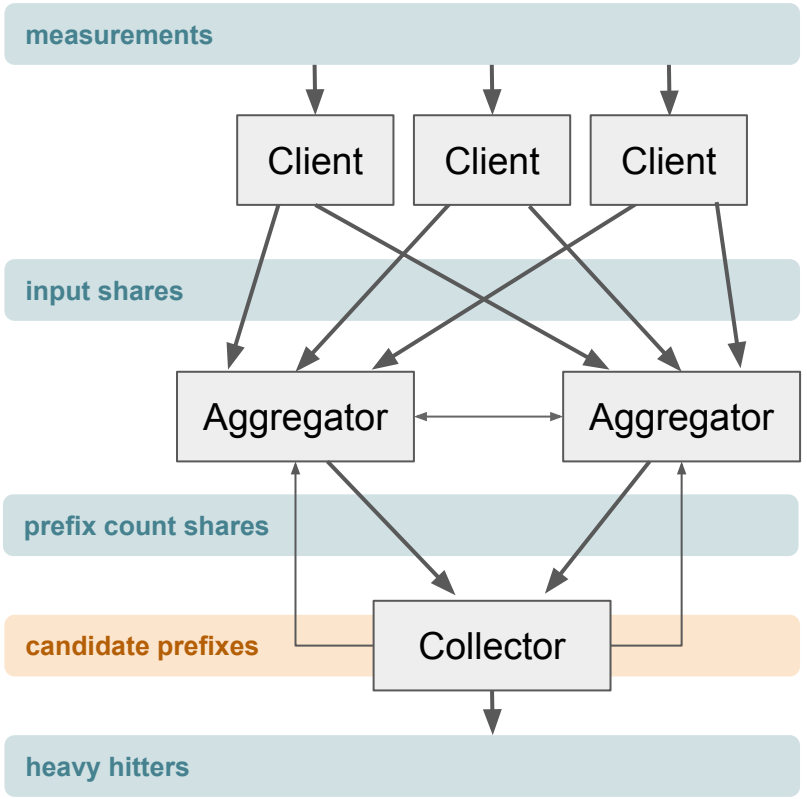
Tim G. and Chris P.

PPM interim – 2024/4/22

Special thanks to David Cook, Hannah Davis, Armando Faz-Hernandez, Simon Friedberger, Brandon Pitman, and Phillipp Schoppmann for spending time on this topic.

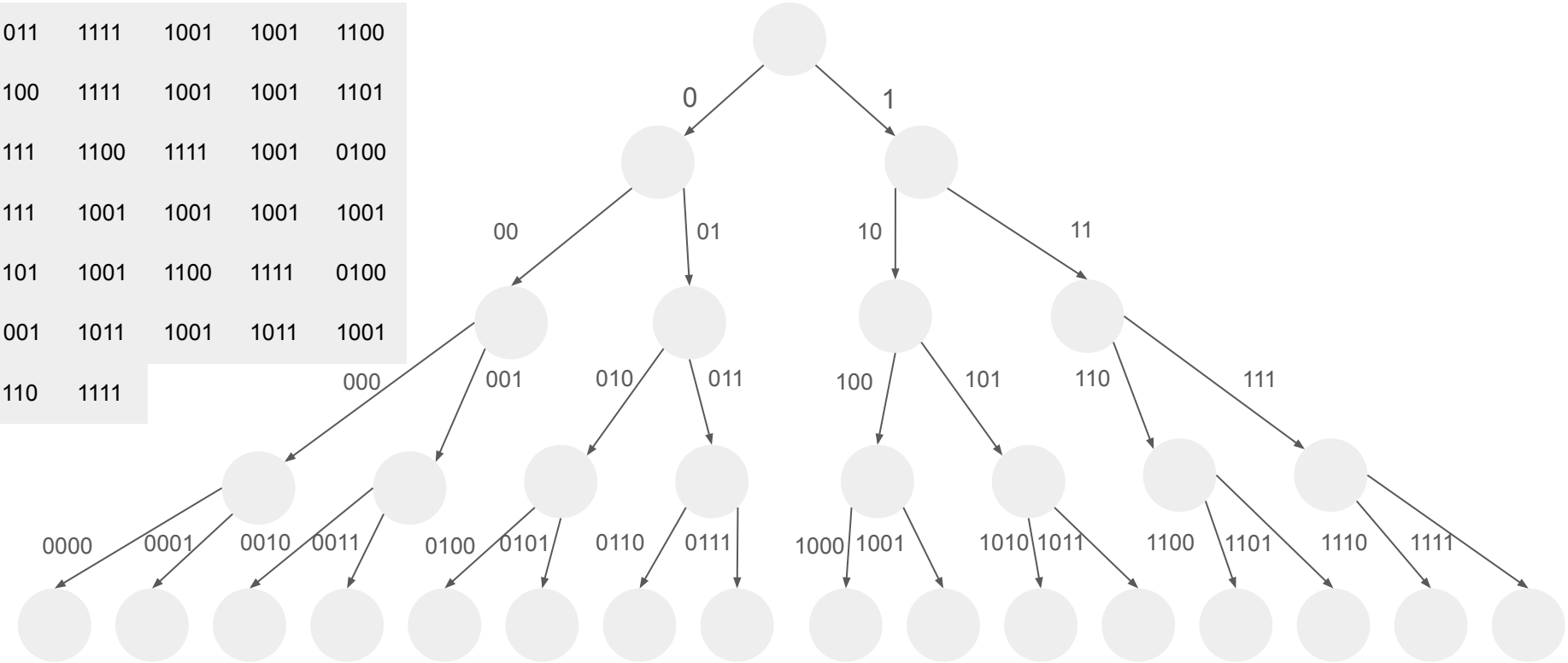
Heavy hitters via Poplar1

- ***t*-heavy-hitters problem**: Each Client uploads a bit string and the Collector wants to learn the subset of strings uploaded at least *t* times
- Solved by Poplar1 (draft-irtf-cfrg-vdaf):
  - IDPF (incremental distributed point function): Aggregators **count** how many inputs begin with each **candidate prefix** specified by the Collector.
  - If a prefix *p* has `prefix_count ≥ t`, then the Collector includes children *p* || 0 and *p* || 1 in the next generation of candidate prefixes.
  - The last generation are the heavy hitters.



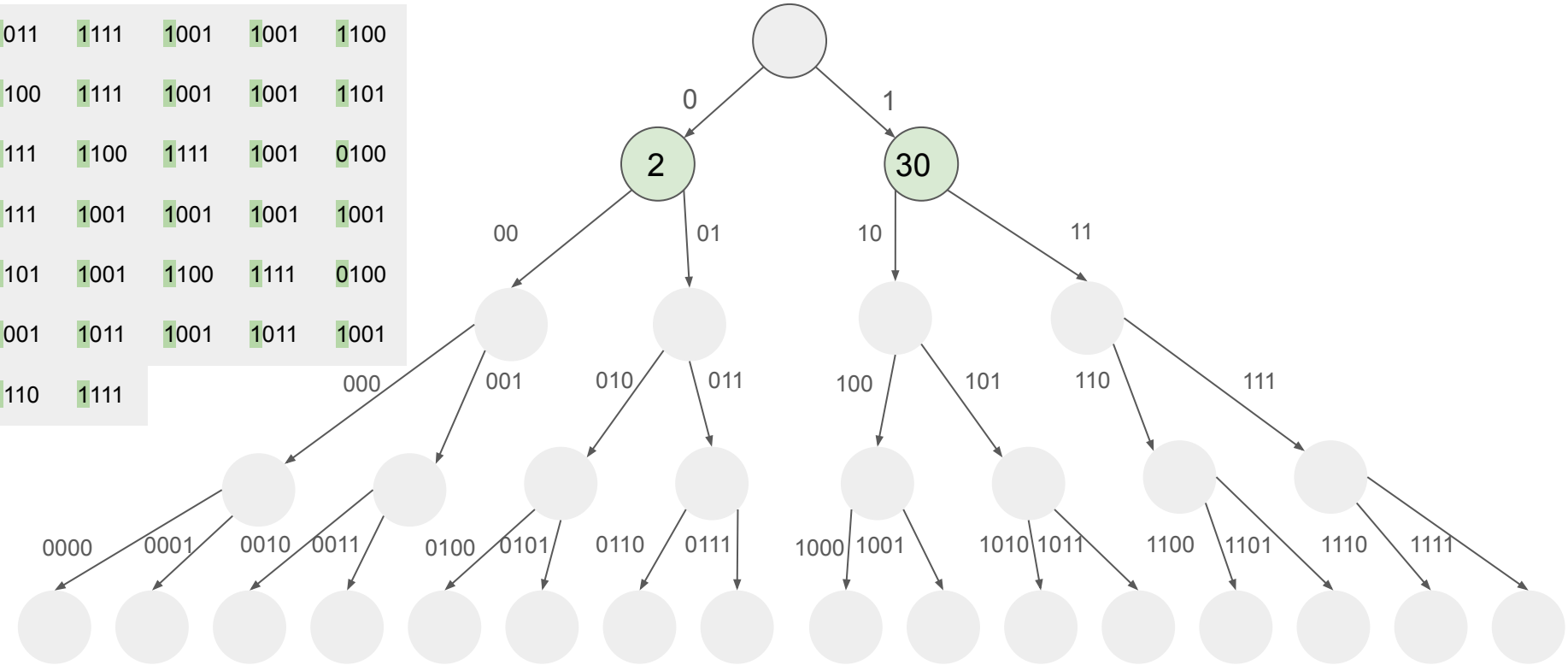
The prefix tree (*t*=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



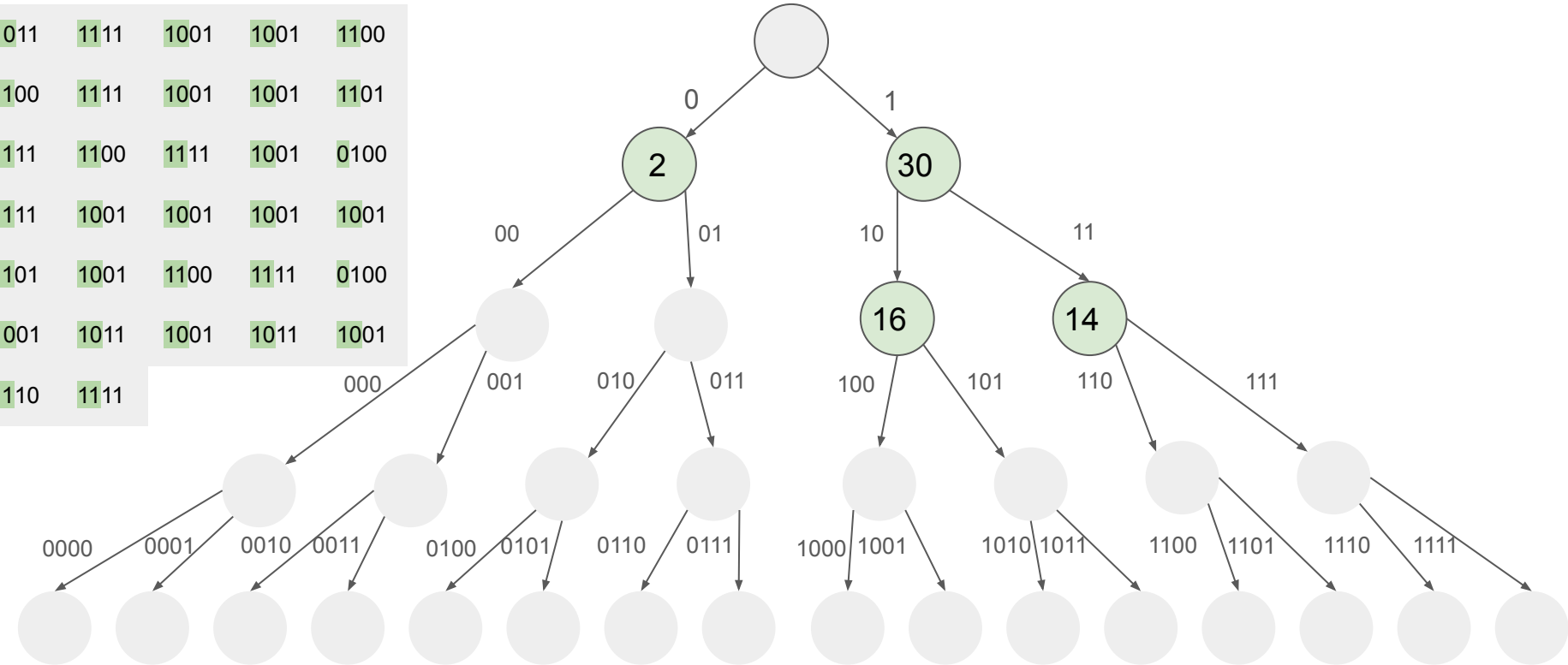
The prefix tree (t=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



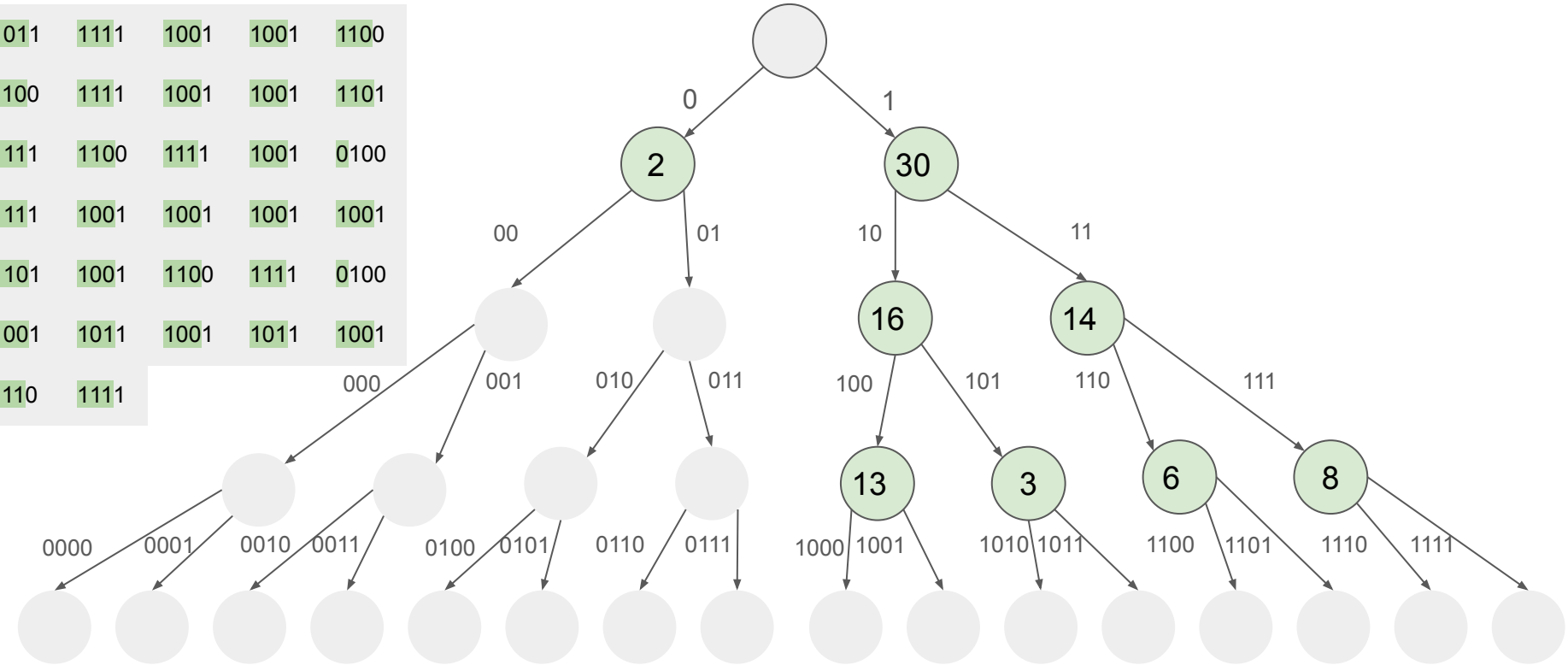
The prefix tree (*t*=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



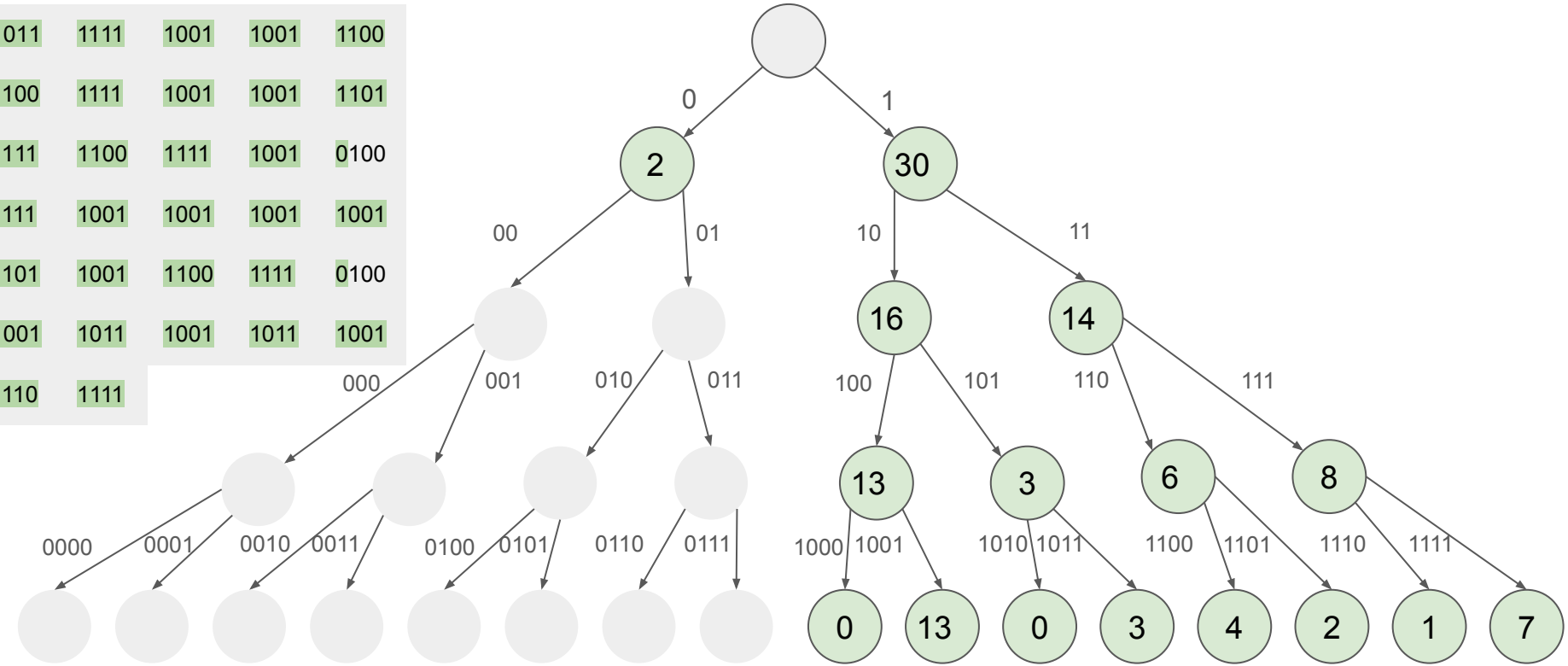
The prefix tree (*t*=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



The prefix tree (*t*=3)

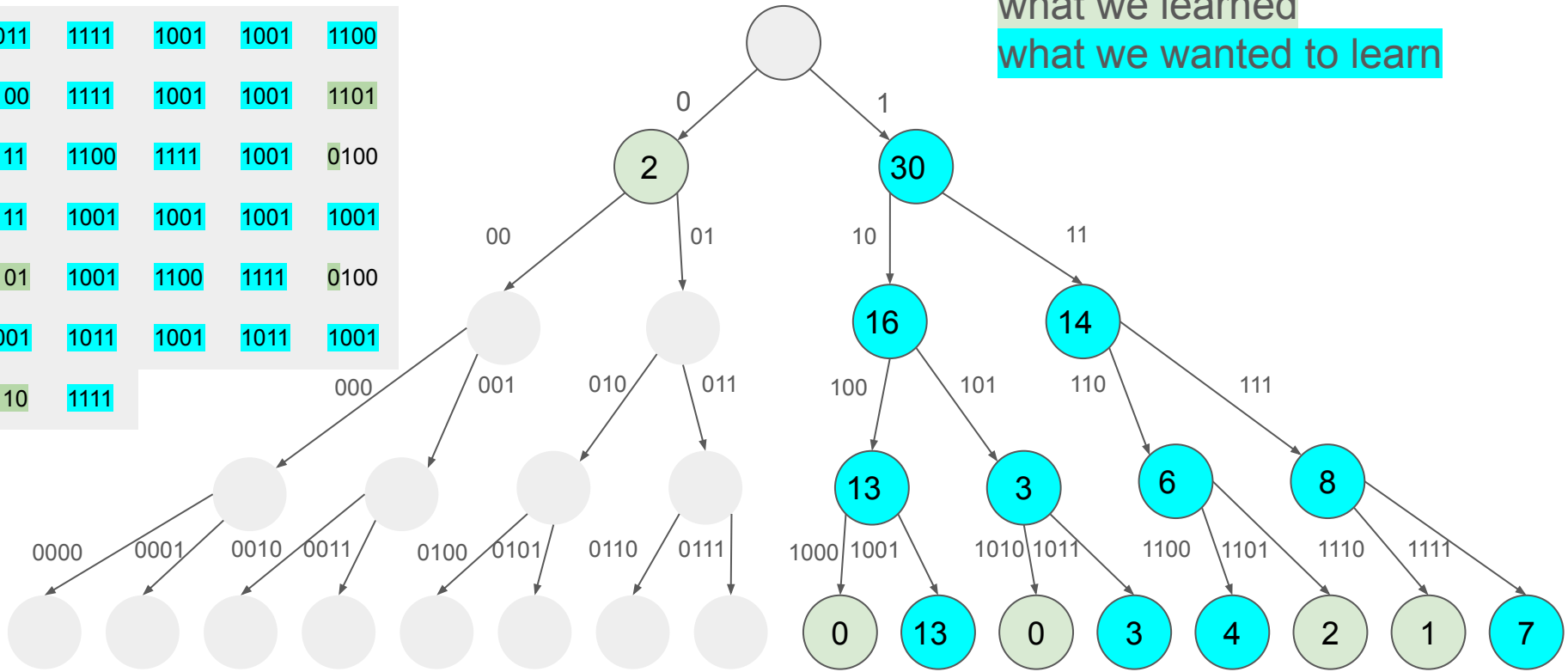
1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



The prefix tree (t=3)

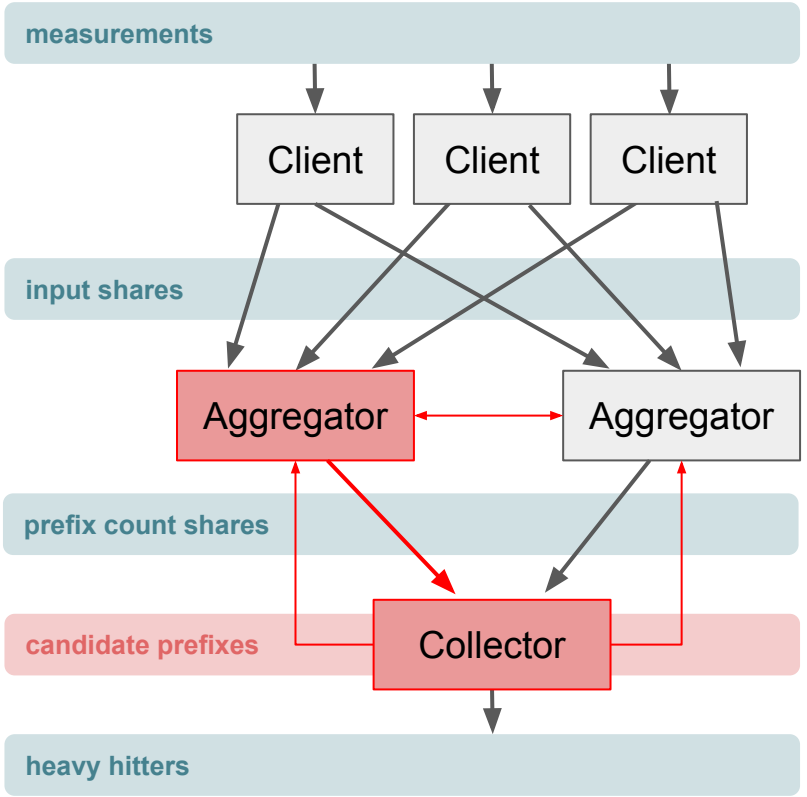
1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			

what we learned
what we wanted to learn



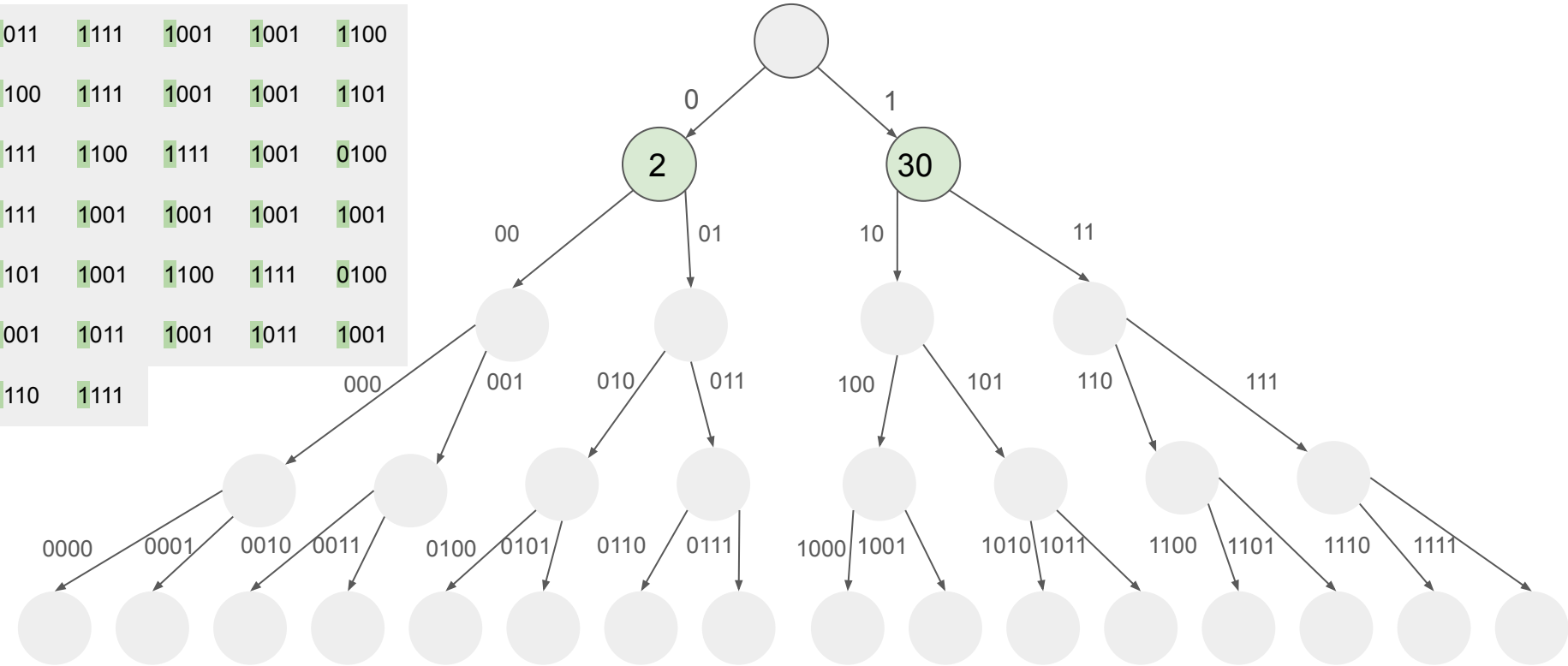
Steering attacks

- **Threat model:** The attacker is active and controls the Collector and one Aggregator ⇒ can **steer** prefix tree traversal towards inputs of interest
- Can be partially mitigated by being stricter about tree traversal:
  - Reveal prefix counts, commit to aggregate shares, …
- Can't mitigate **additive attacks** with IDPFs alone (attacker can always lie about its aggregate share)



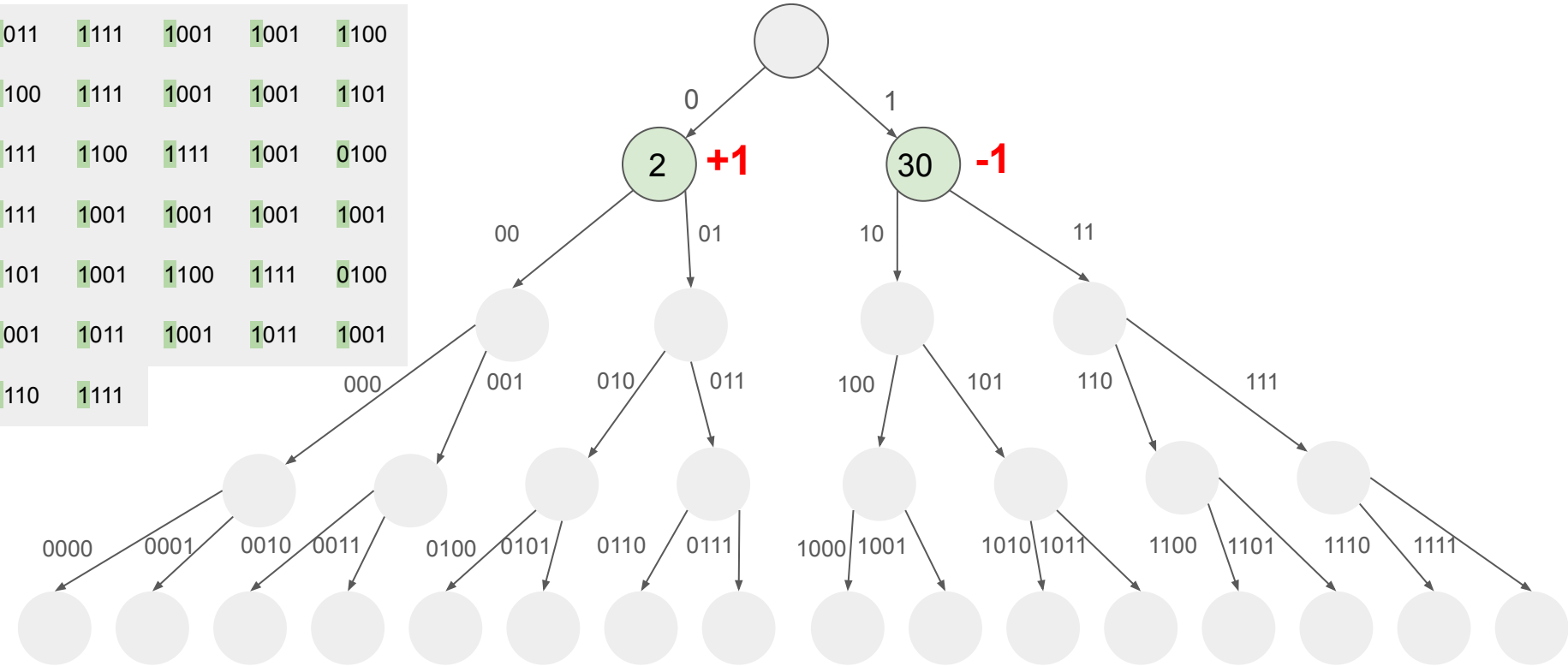
Additive attack (t=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



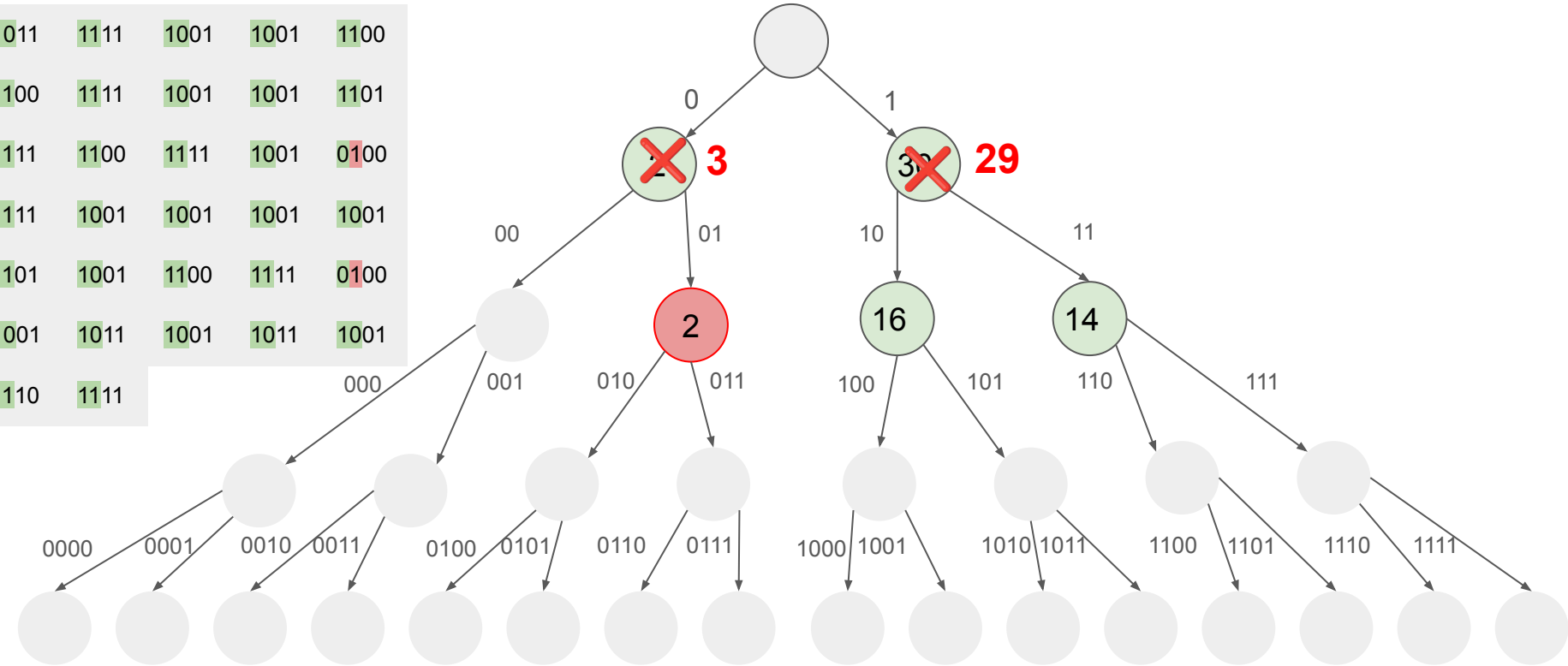
Additive attack (t=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



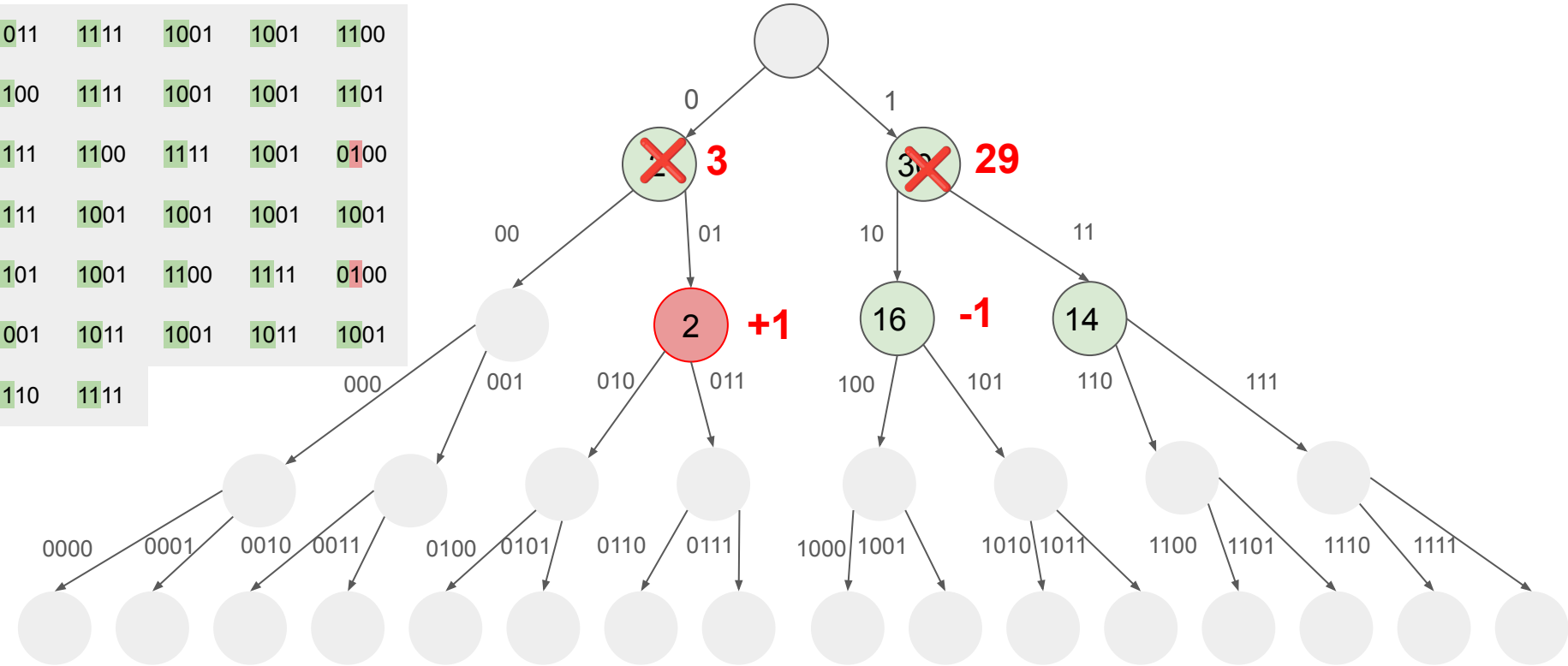
Additive attack (t=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



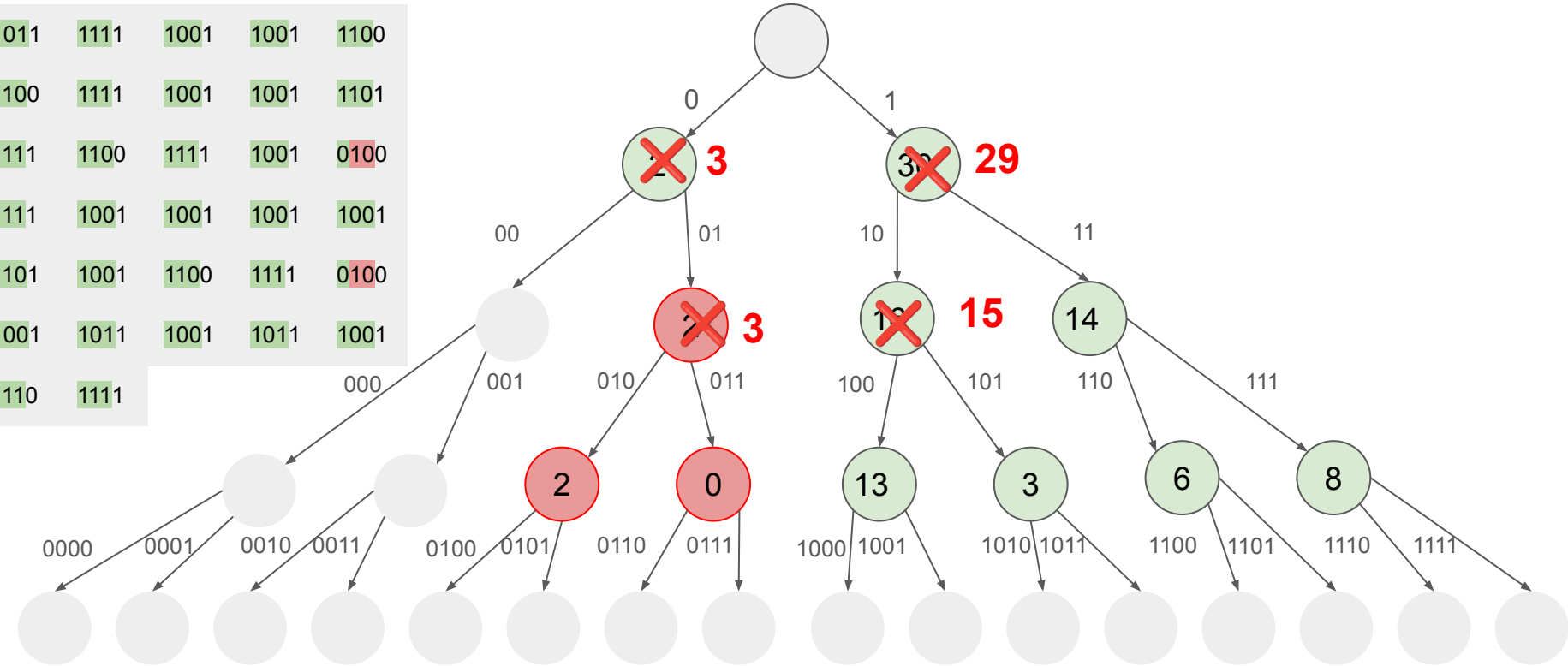
Additive attack (t=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



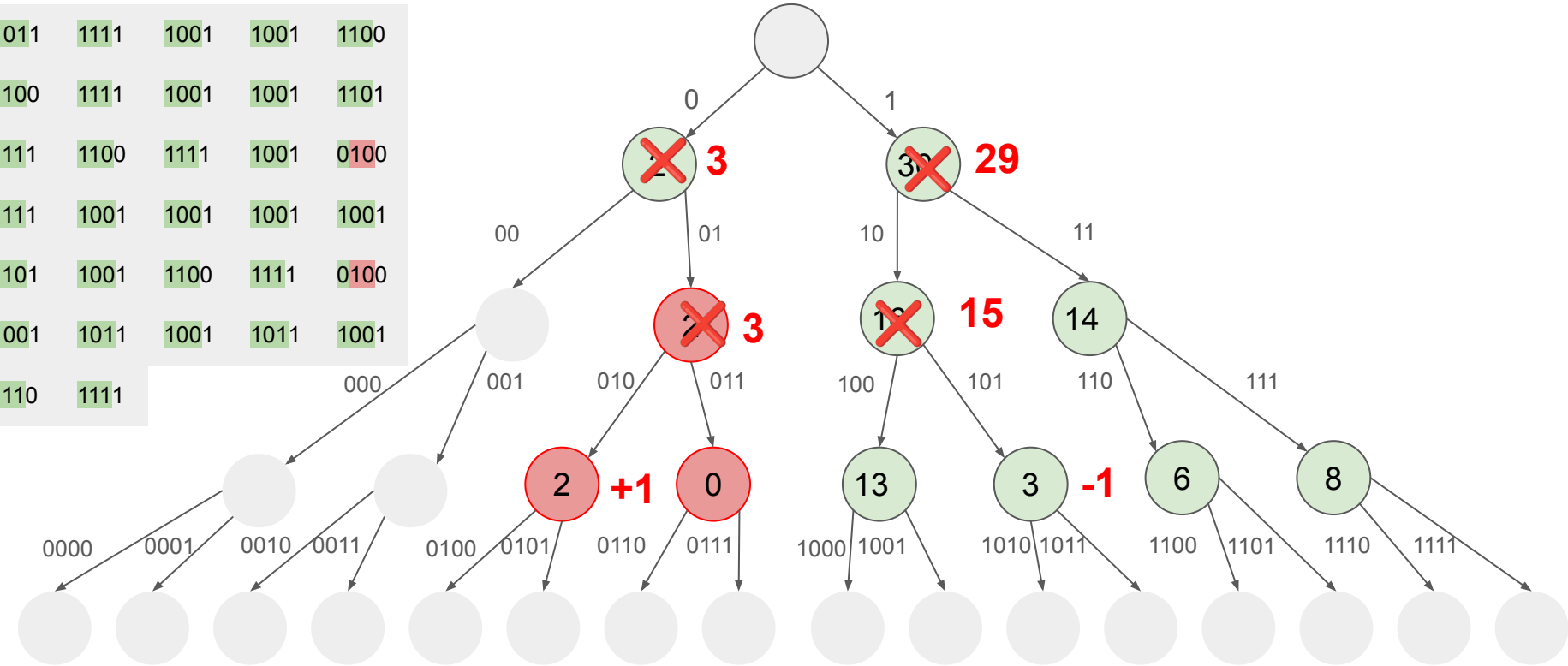
Additive attack (t=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



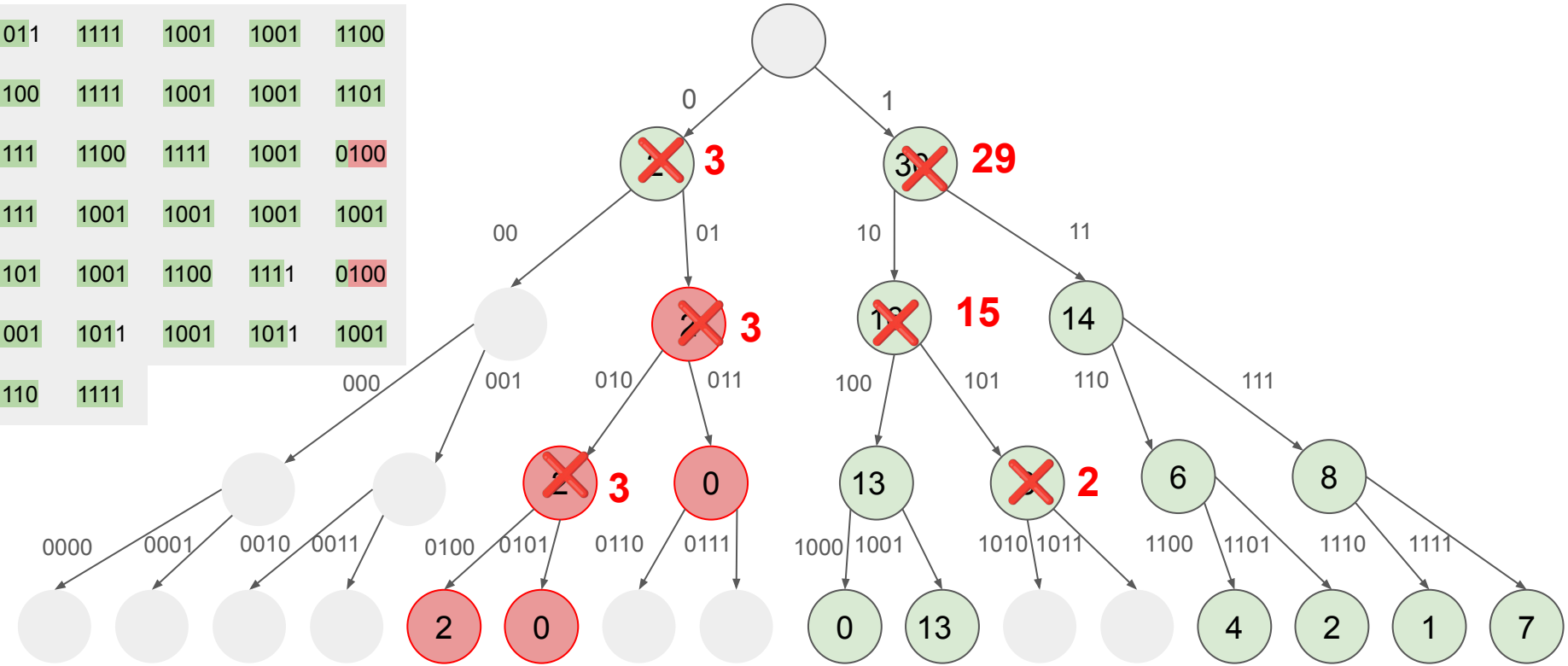
Additive attack (t=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



Additive attack (t=3)

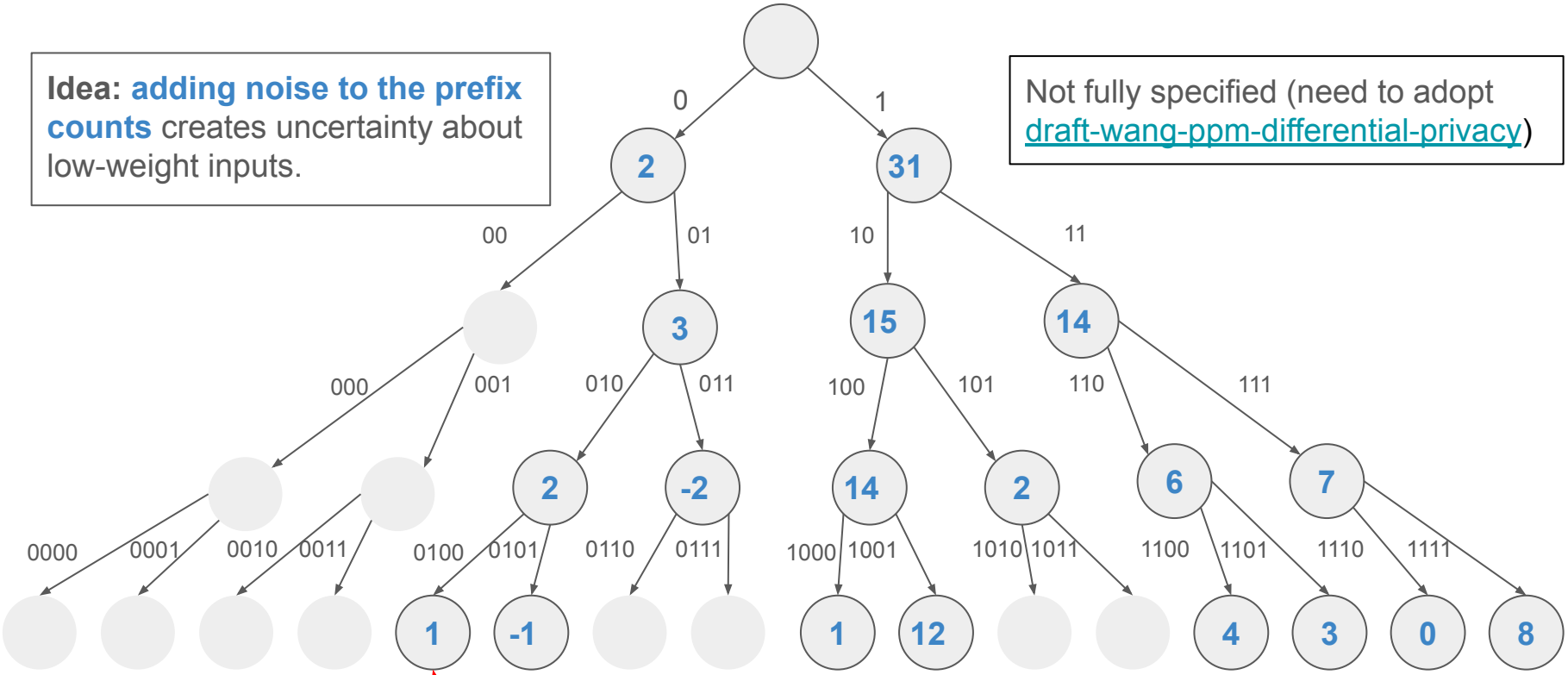
1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111			



Mitigating additive attacks w/ differential privacy

Idea: adding noise to the prefix counts creates uncertainty about low-weight inputs.

Not fully specified (need to adopt draft-wang-ppm-differential-privacy)



Rare input or DP noise?

More crypto?

- With **general purpose MPC**:
  - Securely compute "`prefix_count ≥ t`" over shares of `prefix_count` in the presence of an active attacker ⇒ don't need to reveal prefix counts to the attacker, making steering attacks harder.
  - Sample additive shares of optimal DP noise
  - Probably too expensive in the 2-party setting.
    - What about the 3-party, honest majority MPC framework of IPA?

Different crypto?

- Threshold secret sharing: Clients submit shares of a measurement
  - STAR (draft-dss-star) or its potential successor, POPSTAR (ia.cr/2024/320)
    - Less expensive than DAP/Poplar1, similar threat model
    - Admits a DoS attack we don't yet know how to mitigate efficiently

Q1: Shall DAP support heavy hitters?

- Private heavy hitters is still somewhat of a research problem 🤔
  - DAP/Poplar1 will require DP (not yet specified)
  - Improvements (more or different crypto) might be possible
- Could punt to a future PPM draft that is specialized to heavy hitters
  - The draft would build on or use elements of draft-ietf-ppm-dap
- The current draft is ready to ship with:
  - Prio3 (draft-irtf-cfrg-vdaf) for "all the (linear) things"
  - Mastic (draft-mouris-cfrg-mastic) for attribute-based metrics (aka "drill-down")
  - PINE (draft-chen-cfrg-vdaf-pine) for model training

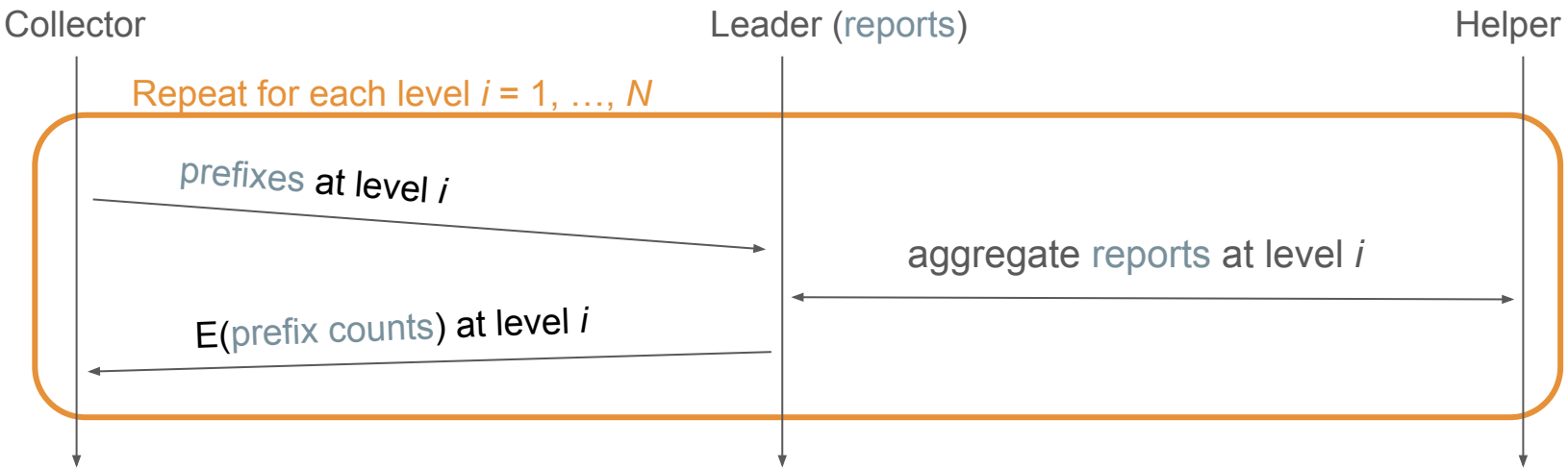
Q1: Shall DAP support heavy hitters?

- If **NO**:
  - Proposal #0: Modify DAP so that each report is aggregated/collected at most once. (Keep aggregation parameter in order to support attribute-based metrics.)
- If **YES**:
  - Proposal #1: Keep current architecture. (Minor changes still required.)
  - Proposal #2: Have the Aggregators compute the prefix tree on their own. (Collector only sees the heavy hitters.)

If **Q1** is **YES** then **Q2**: What to do about steering attacks?

1. Enforce consistent tree traversal: every node must be the child of a previously traversed node. (Prevents the attacker from abandoning one subtree in favor of another.)
2. Reveal prefix counts: the weight of each node must equal the sum of its children. (Prevents the attacker from redistributing weight across subtrees.)
3. Commit to shares of the prefix count: the weight of each node must sum up to the shares committed to by the Aggregators. (Prevents the attacker from re-weighting a node based on its true count.)
    - **Observation**: Taken together, (1), (2), (3) limit the attacker to additive attacks.
4. Require differential privacy: Spell out a DP mechanism to use with DAP/Poplar1. (Mitigates the efficacy of additive attacks.)
    - **Observation**: Some natural DP mechanisms, such as [BBCG+21, Appendix E], make (2) more complicated

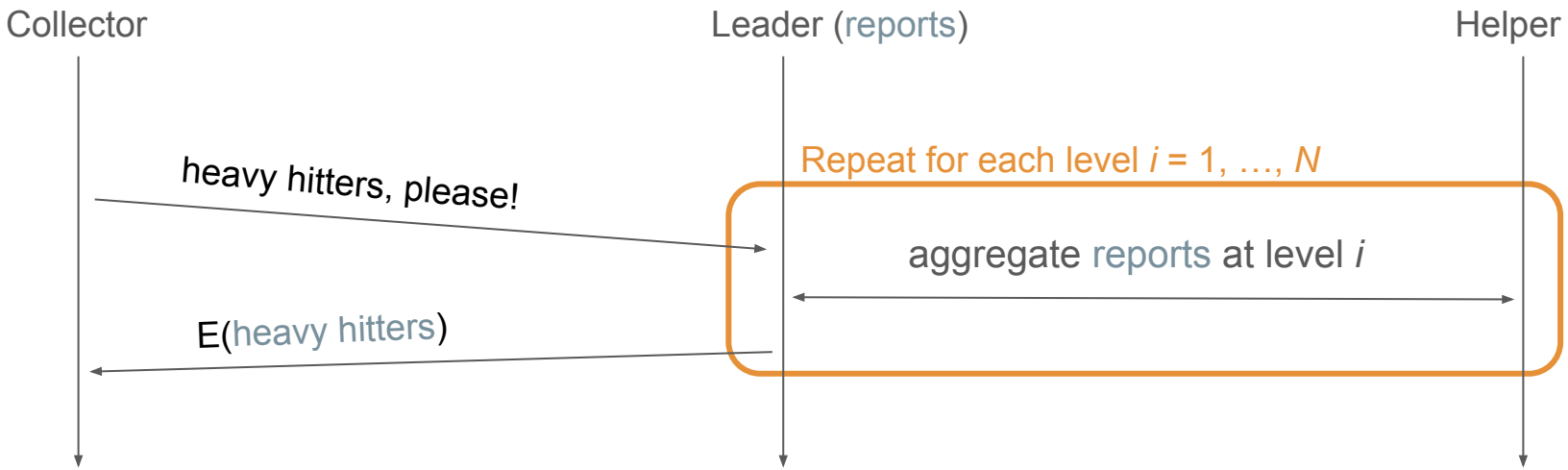
Proposal #1: Collector drives traversal



Spec changes required:

- Store reports across aggregations and add code path for aggregating stored reports
- Clarify early report validation rules

Proposal #2: Aggregators drive traversal



Spec changes required:

- Same as proposal #1, plus…
- Define *Collection Modes* (of which heavy hitters is an instance)
- Aggregators exchange commitment to aggregate shares
- Introduces heavy hitters specific semantics to DAP, breaking the VDAF abstraction

If **Q1** is **YES** then **Q3**: Which proposal shall we adopt?

Property	Proposal #1	Proposal #2
Minimal spec changes	yes	no
Collector learns nothing beyond the heavy hitters	no	yes if both Aggregators are honest
Optimal latency	Collector creates a bottleneck	yes
Generalizes to new VDAFs	yes	any VDAF requiring multiple rounds of collection would need a new *Collection Mode* (in fact, this may be desirable)
Allows application-specific traversal strategies ("level skipping")	yes	Aggregators need to know the strategy

Further reading

- Supporting heavy hitters
  - https://docs.google.com/document/d/1ZjXz-1kGsTDf2Vn2u-fwYqR8BSc3tOYlELVHAYvAfjk
- PPM list discussion
  - https://mailarchive.ietf.org/arch/msg/ppm/jN0udduyP-WP1AmWsbve8THyYlY/

And now for something completely different.

PPM logo?

Credit: Martin Thomson

The prefix tree (t=3)

1011	1111	1001	1001	1100
1100	1111	1001	1001	1101
1111	1100	1111	1001	0100
1111	1001	1001	1001	1001
1101	1001	1100	1111	0100
1001	1011	1001	1011	1001
1110	1111