CAPnet: A Defense Against Cache Accounting Attacks on Content Distribution Networks

Ghada Almashaqbeh[1], Kevin Kelley[2], Allison Bishop[1,3], Justin Cappos[4]

[1]Columbia, [2]CacheCash, [3]Proof Trading, [4]NYU

IEEE CNS 2019, DC, USA

Outline

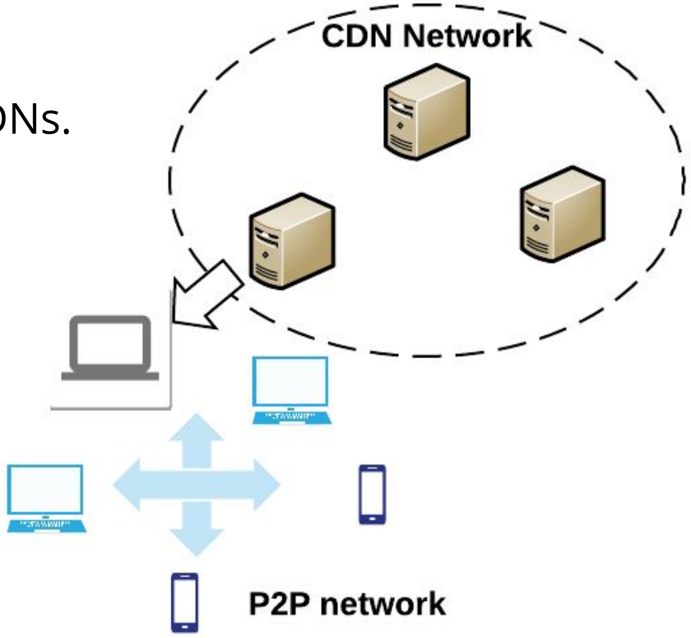
- Background.
- Motivation and problem statement.
- CAPnet design.
- Security analysis.
- Performance evaluation.
- Conclusion.

Online Content Distribution

- Dramatic growth over the past decade.
    - Video streaming accounts for ~60% of today's Internet traffic, projected to exceed 80% by 2022.
- Usually, infrastructure-based content delivery networks (CDNs) are used to distribute the load.
    - Through CDN providers, e.g., Akamai.
- **Drawbacks:**
    - Impose costly and complex business relationships.
    - Require overprovisioning bandwidth to handle peak demands.
    - Issues related to reachability, delays to set up new service, etc.

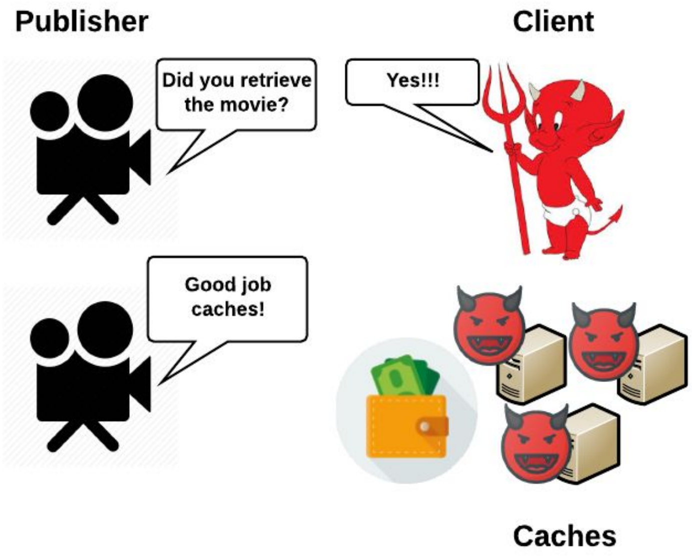
Peer-Assisted CDNs

- Utilize peer-to-peer data transfers to supplement traditional CDNs.
- Allow anyone to join and distribute content to others.
- Advantages:
  - Offer a lower service cost.
  - Create robust and flexible CDN service.
  - Extend network coverage of traditional CDNs.
  - Scale easier with demand.



But … Cache Accounting Attacks

- Clients collude with caches pretending to be served.
- This allows caches to collect rewards without doing any actual work.
- Also, causes problems in network resource management.
- Confirmed by an empirical study on the Maze file system and Akamai Netsession.

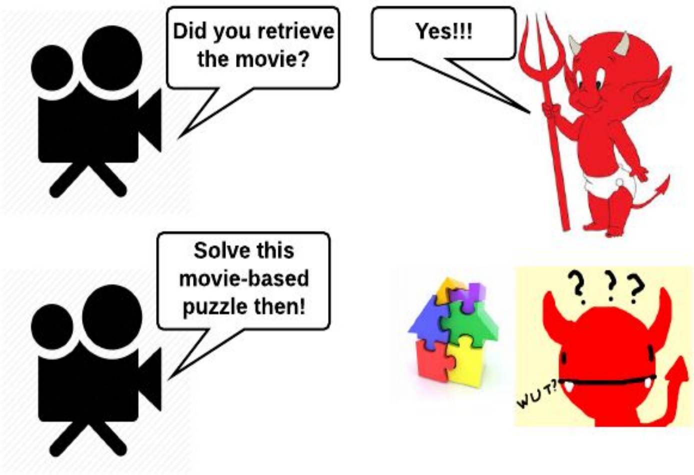


Previous Solutions

- Do not work in typical P2P networks where untrusted, anonymous nodes serve as caches.
  - Rely on activity reports originated by the peers themselves.
    - Such logs can be fabricated.
  - Assume the knowledge of the peer computational power and link delay.
    - Caches cannot be trusted to report such data correctly.
  - Require all nodes who owns a copy of the content to solve a puzzle.
    - Do not work with static content.

Our Solution - CAPnet

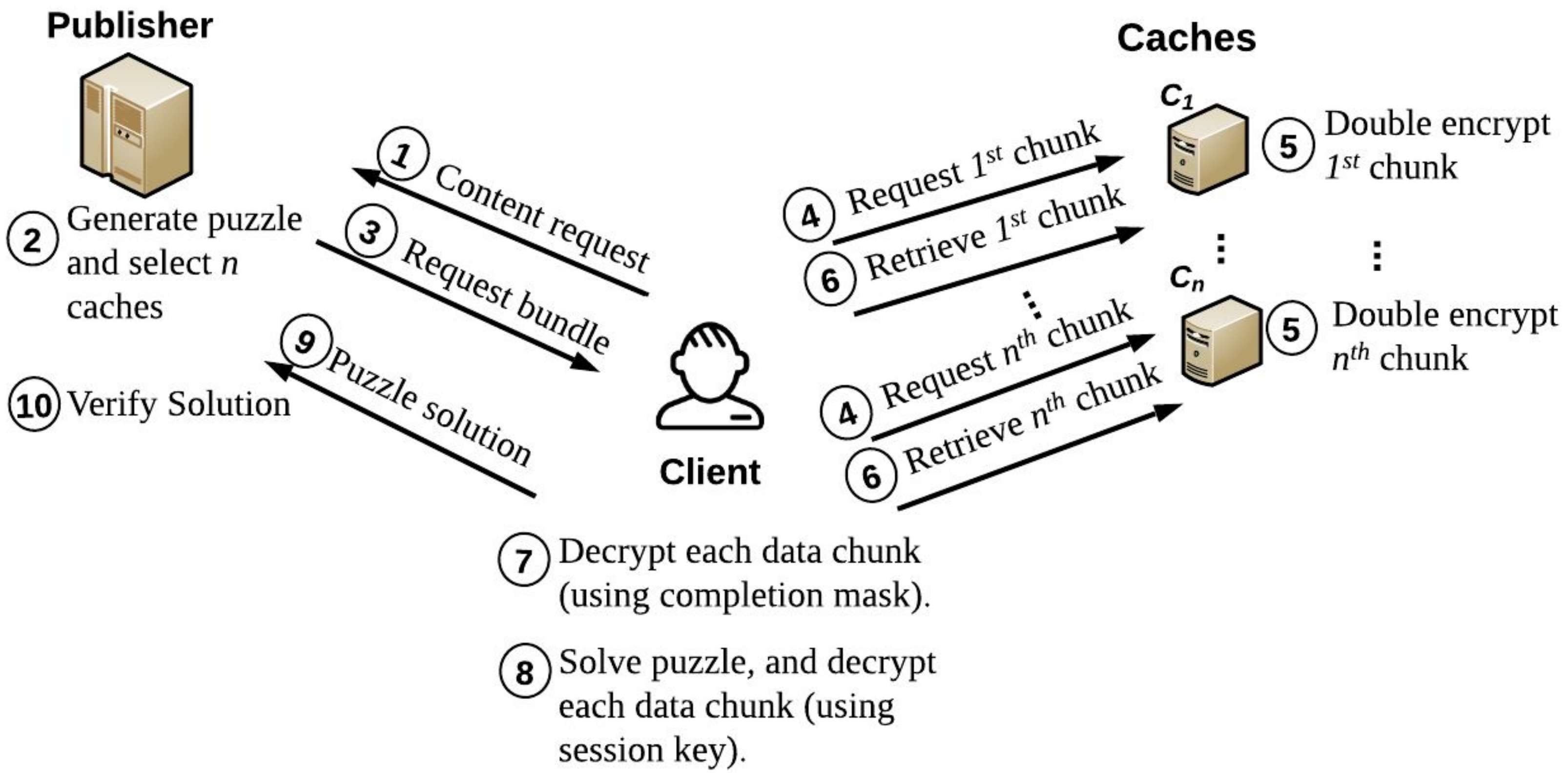
- Lets untrusted caches join peer-assisted CDNs.
- Introduces a novel lightweight cache accountability puzzle that must be solved using the retrieved content.
- Allows a publisher to set a bound on the amount of bandwidth an attacker must expend when solving the puzzle.



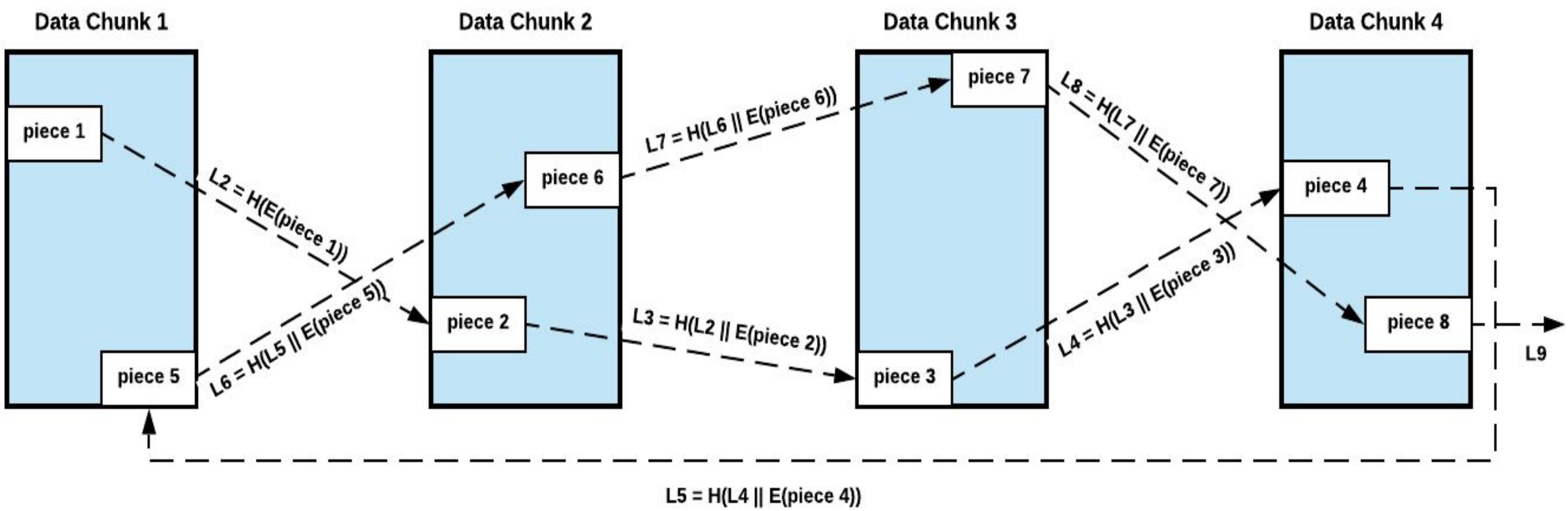
System and Threat Model

- Target peer-assisted CDNs consisting of publishers, clients, and caches.
  - A publisher acts as dispatcher assigning caches to serve content requests.
- When a cache joins a publisher's network:
  - It obtains a full copy of the content, which is divided into data chunks of equal size.
  - It shares a master secret key with the publisher.
- A client can request *n* chunks per request.
  - Hence, CAPnet's puzzle is solved over only *n* chunks (not the whole object).
- A publisher monitors caches' IPs to detect Sybils.
- We work in the random oracle model and in the ideal cipher model.

CAPnet Design



Cache Accountability Puzzle Design



Puzzle challenge = $H(L_9)$ Puzzle solution = L_9

Puzzle Solving and Verification

- **Puzzle Solving.**
  - Same as generation, however, a client does not know the starting piece.
  - It tries pieces from the first data chunk until the solution is found.
- **Puzzle verification.**
  - A publisher can generate a secret token using a secret PRF.
  - Encrypt this token using the puzzle solution, and send ciphertext to client.
  - A client decrypts once it solves the puzzle and send the token back to the publisher.

Security Analysis I

- Define a δ-bound, which is ratio between the number of pieces a puzzle solver retrieve and the total number of pieces in the requested chunks.
    - E.g., 0.9-bound means that a solver would expends a bandwidth cost sufficient to retrieve 90% of the content before solving the puzzle.
- A publisher can configure the number of puzzle rounds to achieve a specific bound.
    - Also, needs to configure the piece size.

Security Analysis II

- A client is colluding with a set of malicious caches, Cm, of size m < n.
  - The goal is to solve the puzzle while retrieving the least amount of data.
- We have a two-entity model:
  - The client always retrieve data chunks from honest caches.
  - A malicious cache pools data from other caches in Cm.
  - One will be the puzzle solver and one will be the piece provider.
- We assume a strong adversary that knows the frequency distribution of all pieces in all data chunks.
- Set piece size <= *hash size/m*
- Using simulation, we determine the number of puzzle rounds based on the desired δ-bound.

Parameter Setup - An Example

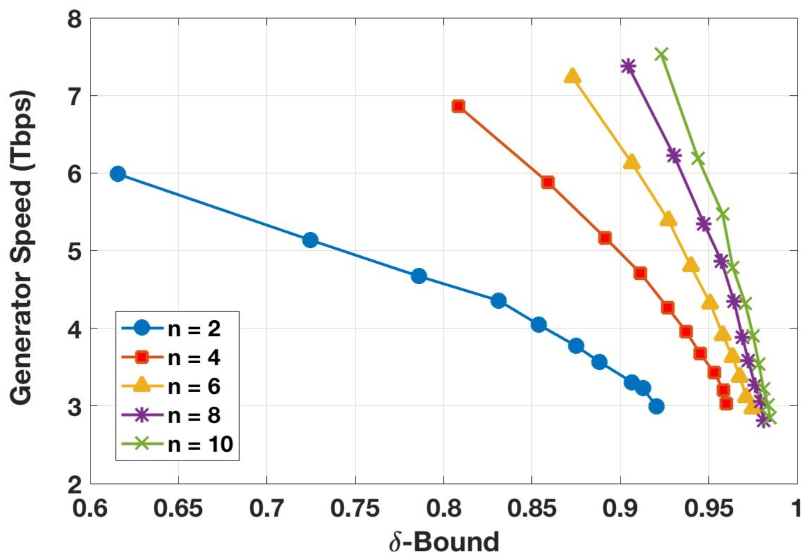
- 1 MB chunk size, 16-byte piece size, n = 6 caches.

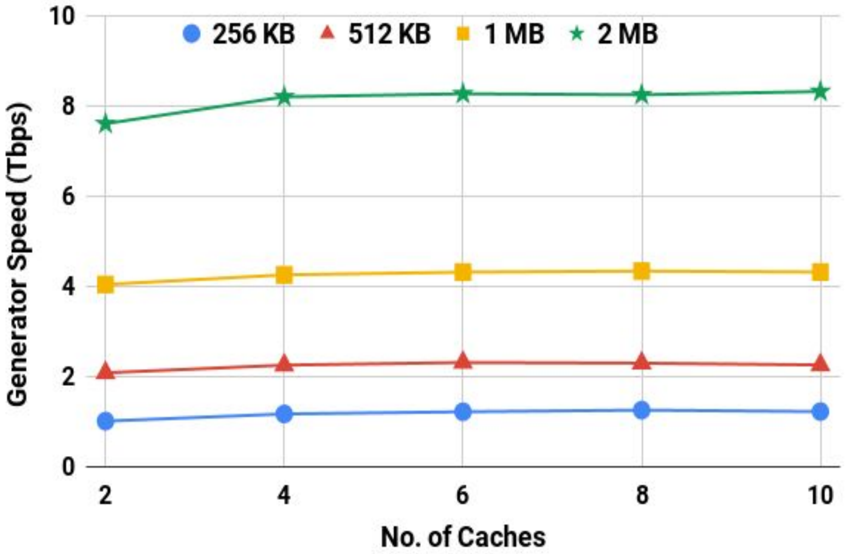
	Client as solver			Either	Cache as solver		
R \ m	**0**	**1**	**2**	**3**	**4**	**5**	**6**
1	1	0.87±0.03	0.78±0.06	0.71±0.08	0.45±0.06	0.21±0.03	0
2	1	0.91±0.04	0.86±0.06	0.82±0.08	0.52±0.06	0.24±0.04	0
3	1	0.93±0.04	0.9±0.05	0.87±0.07	0.57 ±0.05	0.26±0.04	0
4	1	0.94±0.03	0.92±0.05	0.91±0.06	0.59±0.05	0.28±0.03	0
5	1	0.95±0.03	0.94±0.04	0.93±0.04	0.6±0.05	0.29±0.03	0
6	1	0.96±0.03	0.95±0.04	0.94±0.04	0.61±0.04	0.29±0.03	0
7	1	0.96±0.02	0.95±0.02	0.95±0.04	0.62±0.04	0.3±0.03	0
8	1	0.97±0.02	0.96±0.03	0.95±0.03	0.63±0.03	0.3±0.02	0
9	1	0.97±0.02	0.97±0.03	0.96±0.03	0.63±0.03	0.3±0.02	0
10	1	0.97±0.02	0.97±0.03	0.97±0.03	0.64±0.03	0.31±0.02	0

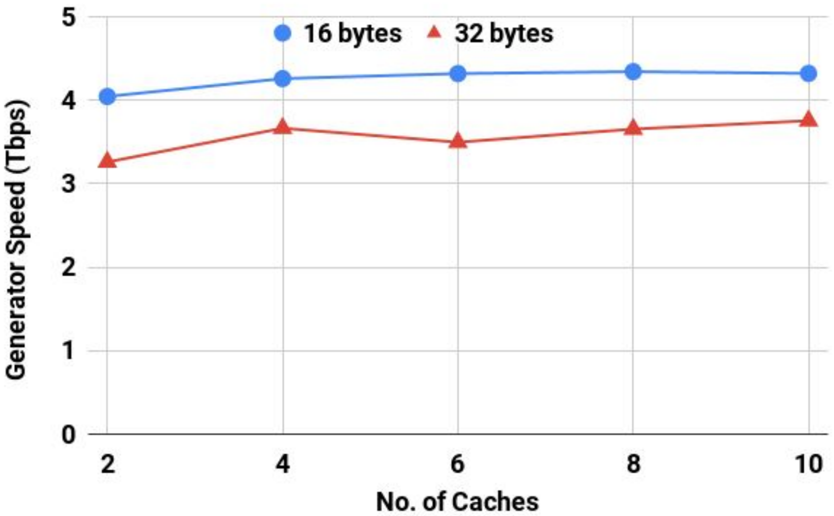
Performance Evaluation

- Benchmarks to evaluate puzzle generation and solving rate.
  - Represented in terms of content bitrate.
- Study the effect of puzzle rounds (or δ bound), chunk size, and piece size.

CAPnet Efficiency - Generator

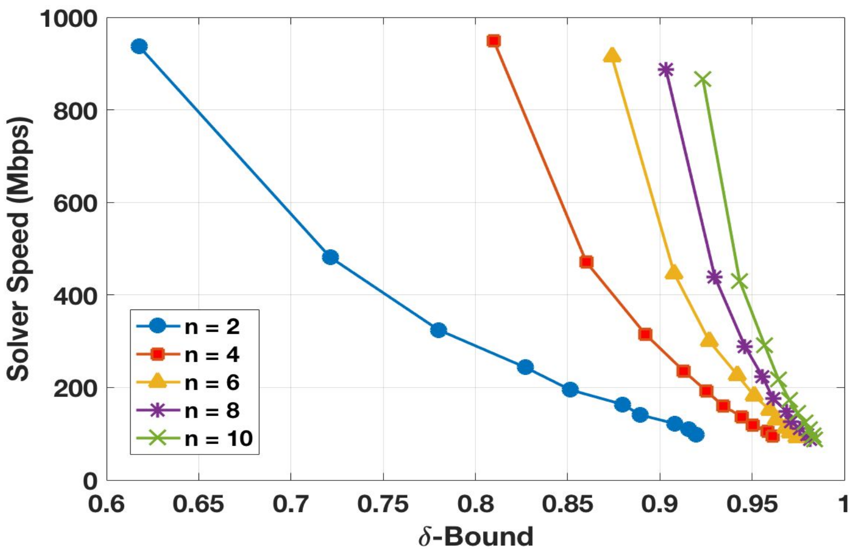


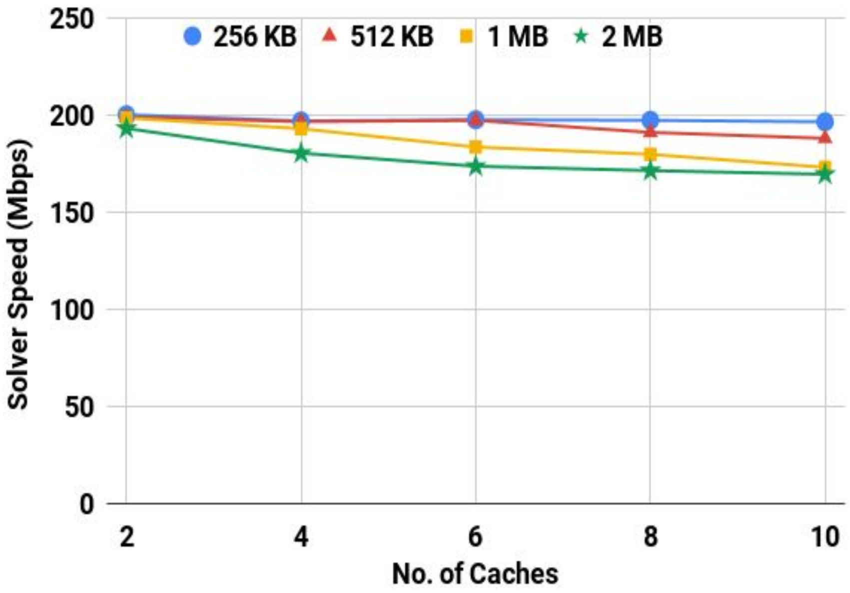


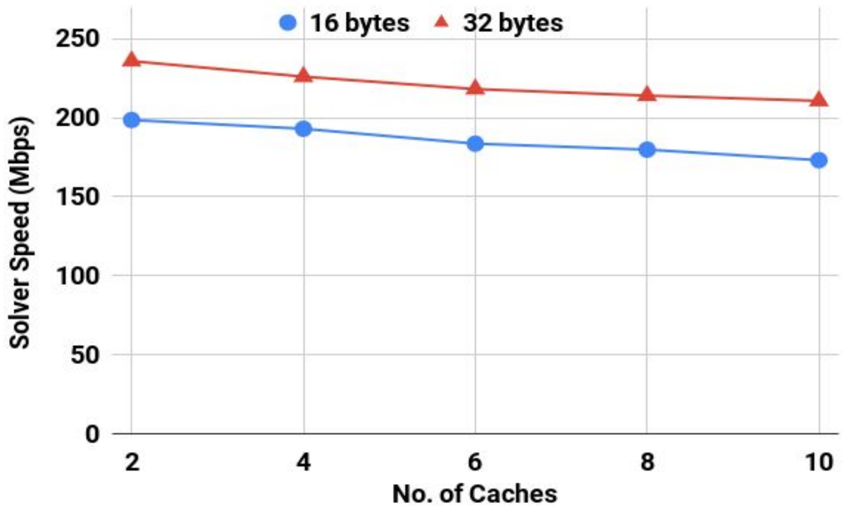


A publisher can generate puzzles sufficient to serve 870,000 clients watching the same 1080p video concurrently.

CAPnet Efficiency - Solver







A client can solve puzzles sufficient to retrieve 34 1080p videos concurrently.

Conclusion

- CAPnet is a low-overhead defense mechanism against cache accounting attacks.
- Its core module is a cache accountability puzzle that clients solves before caches are given credit.
    - Publishers process small number of pieces, while clients process large amount of the content (based on the δ-bound).
- Highly efficient, it allows publishers to serve content, and clients to retrieve content, at a high bitrate.


THANK YOU