WaterBear: Asynchronous BFT with Information-Theoretic Security and Quantum Security

Haibin Zhang

Beijing Institute of Technology

Sisi Duan

Tsinghua University and Zhongguancun Laboratory

Liehuang Zhu

Beijing Institute of Technology

Boxin Zhao

Zhongguancun Laboratory

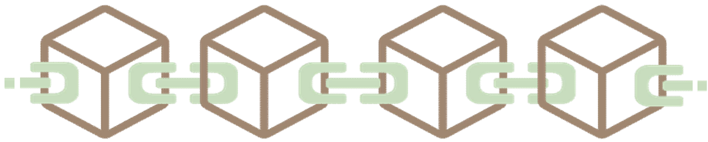
北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

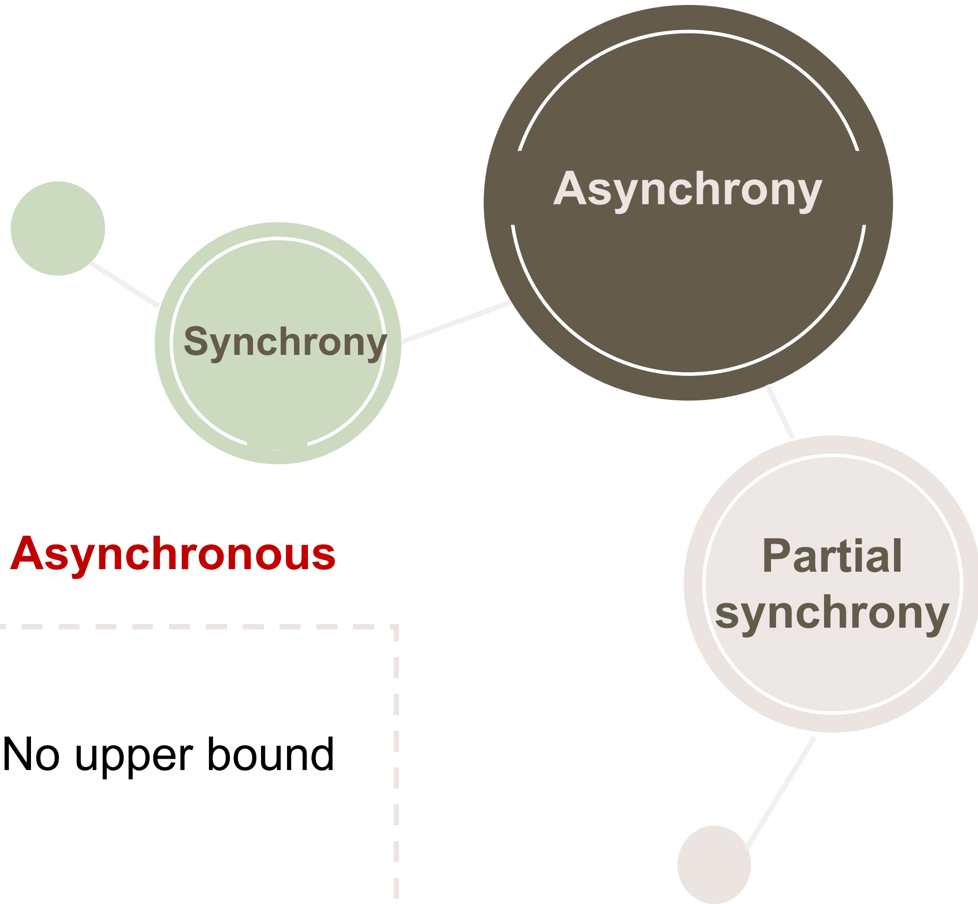
清華大學
Tsinghua University

Usenix Security 2023

Byzantine Fault Tolerance (BFT)

- Building block for blockchains





- Timing assumptions

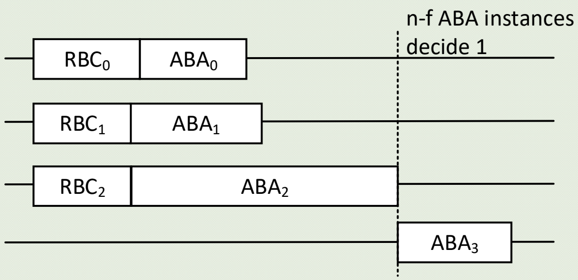
Synchronous	Partially Synchronous	Asynchronous
Known upper bound for message transmission /processing	Unknown upper bound	No upper bound

Background

- **Computational security**
  - The adversary is restricted to probabilisitc polynomial-time
- **Information-theoretic security**
  - The adversary is unbounded
  - Typically assuming secure or authenticated channels
- **Quantum security (no PKC)**
  - No public key cryptography (PKC)

Asynchronous BFT Paradigms

BKR



Ben-Or, Kemler, and Rabin (BKR)
PODC 1994

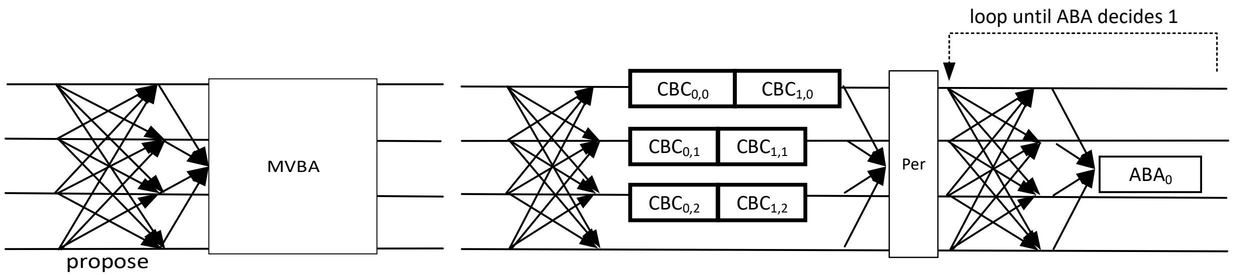
HoneyBadger CCS 2016	BEAT CCS 2018	EPIC DSN 2020	RedBelly S&P 2021

*assumes partial synchrony

$O(n^3)$ message	$O(Ln^2+\lambda n^3 \log n)$ communication	information-theoretic	$O(\log n)$ time
$O(n^2)$ message	$O(Ln^2+\lambda n^3 \log n)$ communication	Not information-theoretic	$O(1)$ time

CKPS

*$O(Ln2+\lambda n2)$ communication can be achieved theoretically

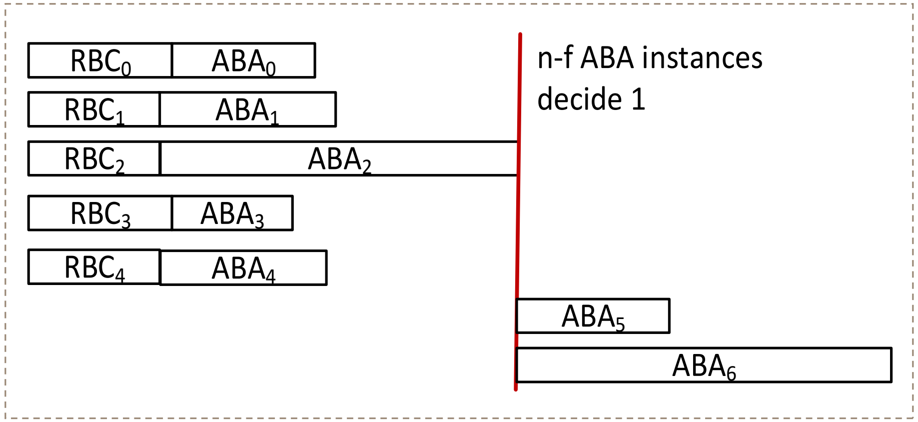


Cachin, Kusawe, Petzold, Shoup (CKPS)
CRYPTO 2001

Dumbo CCS 2020	Speeding Dumbo NDSS 2022

BKR (PODC 1994) -> PACE (CCS 2022)

BKR



ABA becomes the bottleneck

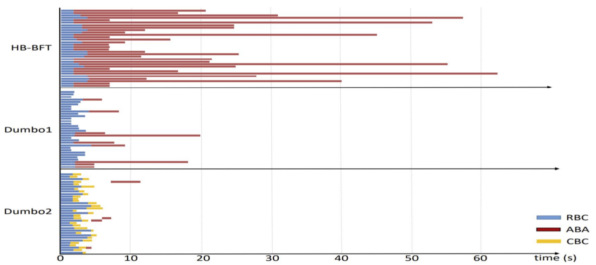
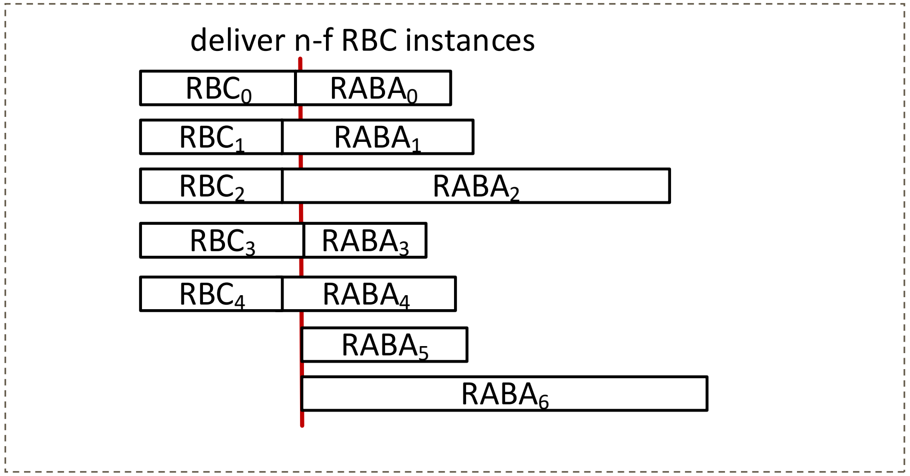


Fig. 5. Running time breakdown of Dumbo1/2 and HoneyBadgerBFT on one random node [6].

Dumbo (CCS 2020)

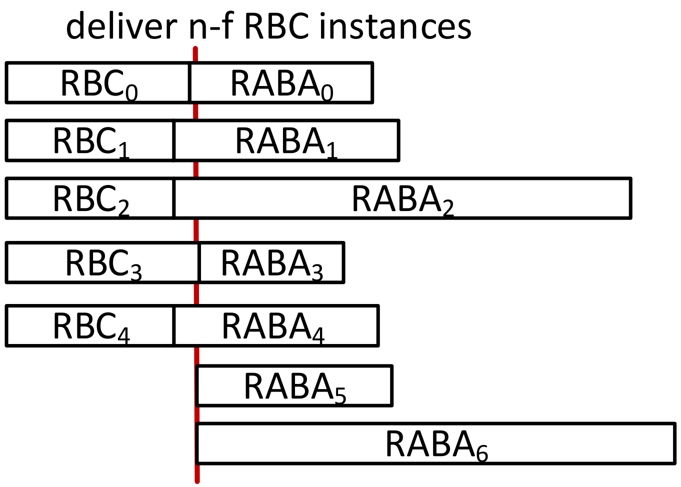
PACE (Zhang and Duan)



Significant performance gain compared to BKR

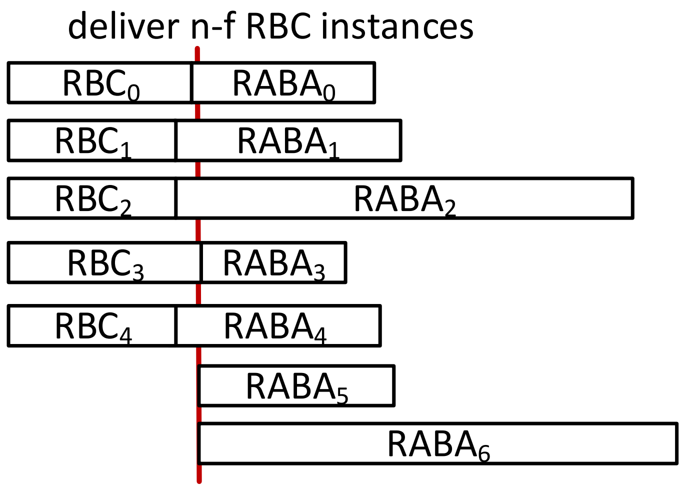
When f=30, the peak throughput of PACE-Pisa is **1.66x** that of Dumbo, **3.6x** that of BEAT (CCS 2018)

A Closer Look at PACE Paradigm



- **Challenges with RBC**
  - Bracha's broadcast (PODC 1984)
    - Information-theoretic
    - carry message payload in every step
    - $O(Ln^2)$ communication; not communication-efficient
    - WaterBear
  - CT RBC (SRDS 2015)
    - Quantum-secure
    - Uses hashes
    - $O(Ln+\kappa n^2 \log n)$ communication
    - WaterBear-QS
  - Can use recent advancement as well, e.g., EFBRB (PODC 2022), CCBRB (PODC 2022)

A Closer Look at PACE Paradigm



- **Challenges with ABA**
  - Most practical ABA rely on common coins
  - Instantiated with threshold signatures or threhsold PRF
- **Our solution**
  - Use ABA with local coins

ABA from Local Coins

```
Initialization
  r ← 0                                                  {round}
func propose(v)
  iv_0 ← v
  vset ← {0,1}           {valid binary values that will be accepted}
  start round 0
round r
  r-broadcast pre-vote_r(iv_r)                         {▷ phase 1}
  upon r-delivering n − f pre-vote_r() such that for each
pre-vote_r(v), v ∈ vset                                {▷ phase 2}
    if there are n − f pre-vote_r(v)
      decide v
      iv_{r+1} ← v
      vset ← {v}
    else
      v ← majority value in the set of pre-vote_r() messages
    r-broadcast main-vote_r(v)
  upon r-delivering n − f main-vote_r() such that for each
main-vote_r(v), v ∈ vset                               {▷ phase 3}
    if there are at least n/2 main-vote_r(v)
      vset ← {v}
    else
      v ← {⊥}
      vset ← {0,1}
    r-broadcast final-vote_r(v)
  upon r-delivering n − f final-vote_r() such that for each
final-vote_r(v), v ∈ vset; for each final-vote_r(∗), vset = {0,1}
    if there are at least 2f + 1 final-vote_r(v)
      decide v
      iv_{r+1} ← v
      vset ← {v}
    else if there are f + 1 final-vote_r(v)
      iv_{r+1} ← v
      vset ← {0,1}
    else
      c ← Random()                              {obtain local coin}
      iv_{r+1} ← c
      vset ← {0,1}
  r ← r+1
```

Figure 10: The Bracha's ABA protocol [13]. The code for p_i.

- **The only known ABA from local coins**
  - Bracha's ABA (PODC 1984)
  - 3 phases of n parallel RBC instances
  - $O(n^3)$ message
  - $O(2^n)$ time complexity due to the use of local coins
- **Our goals**
  - Design more efficient local coin based ABA
  - Avoid querying coins as much as possible
  - Coin-free fast path

Our Local Coin Based ABA

Bracha's ABA
$O(n^3)$ message
9 steps in the fast path

Cubic-ABA
$O(n^3)$ message
5 steps in the fast path

Quadratic-ABA
$O(n^2)$ message
4 steps in the fast path

ABA (local coins)	messages/round	steps/round
Bracha's ABA [14]	n^3	9 to 12
Cubic-ABA (this work)	n^3	5 to 7
Quadratic-ABA (this work)	n^2	4 or 5

Table 3: Local coin based ABA protocols with optimal resilience. We consider the messages and steps in each round. Messages/round and steps/round denote number of messages and steps among all replicas per round.

Our ABAs

- By replacing local coins with **weak common coins** or **comon coins**, we obtain more efficient ABA protocols compared to existing state-of-the-art ABA

ABA (weak common coins)	steps/round	rounds
MMR15 [57, 2nd alg]	9 to 13	$d+1$
Crain [26, 1st alg]	5 to 7	$d+1$
CC-ABA (this work)	4 or 5	$d+1$

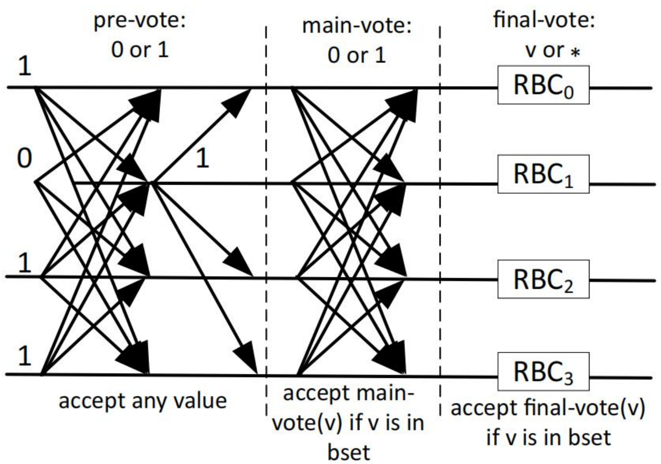
Table 4: ABA protocols using weak common coins. Rounds denote the expected number of rounds. The total number of steps is a product of steps/round and rounds.

ABA (common coins)	steps/round	rounds	good-case-coin-free
MMR15 [57, 2nd alg]	9 to 13	3	yes
Cobalt [53]	3 or 4	4	no
Crain [26, 1st alg]	5 to 7	3	yes
Crain [26, 2nd alg]	2 or 3†	4	no
Pillar [64]	2 or 3	4	no
CC-ABA (this work)	4 or 5	3	yes

Table 5: ABA protocols using perfect common coins. †The second algorithm of Crain relies high threshold common coins and is less efficient than Pillar. Compared to Pillar, CC-ABA has the good-case-coin-free property that is vital for the asynchronous distributed key generation protocol [30].

Our ABAs

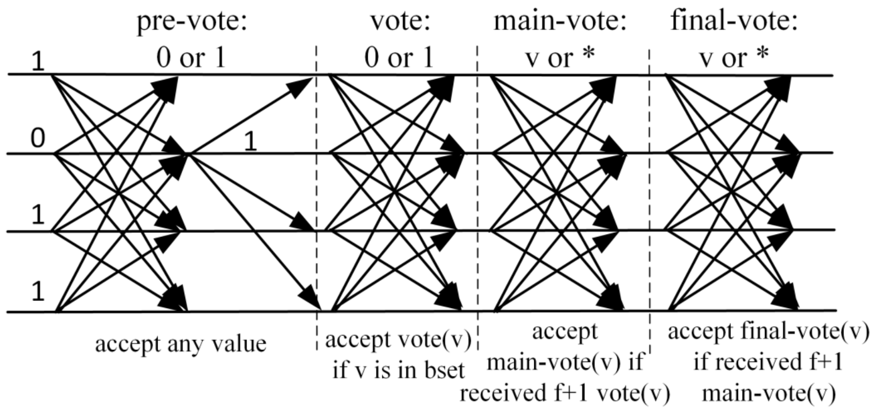
Cubic-ABA



Idea: Use all-to-all communication to replace parallel RBC as much as possible

Bracha's ABA involves 3 phases of n parallel RBC

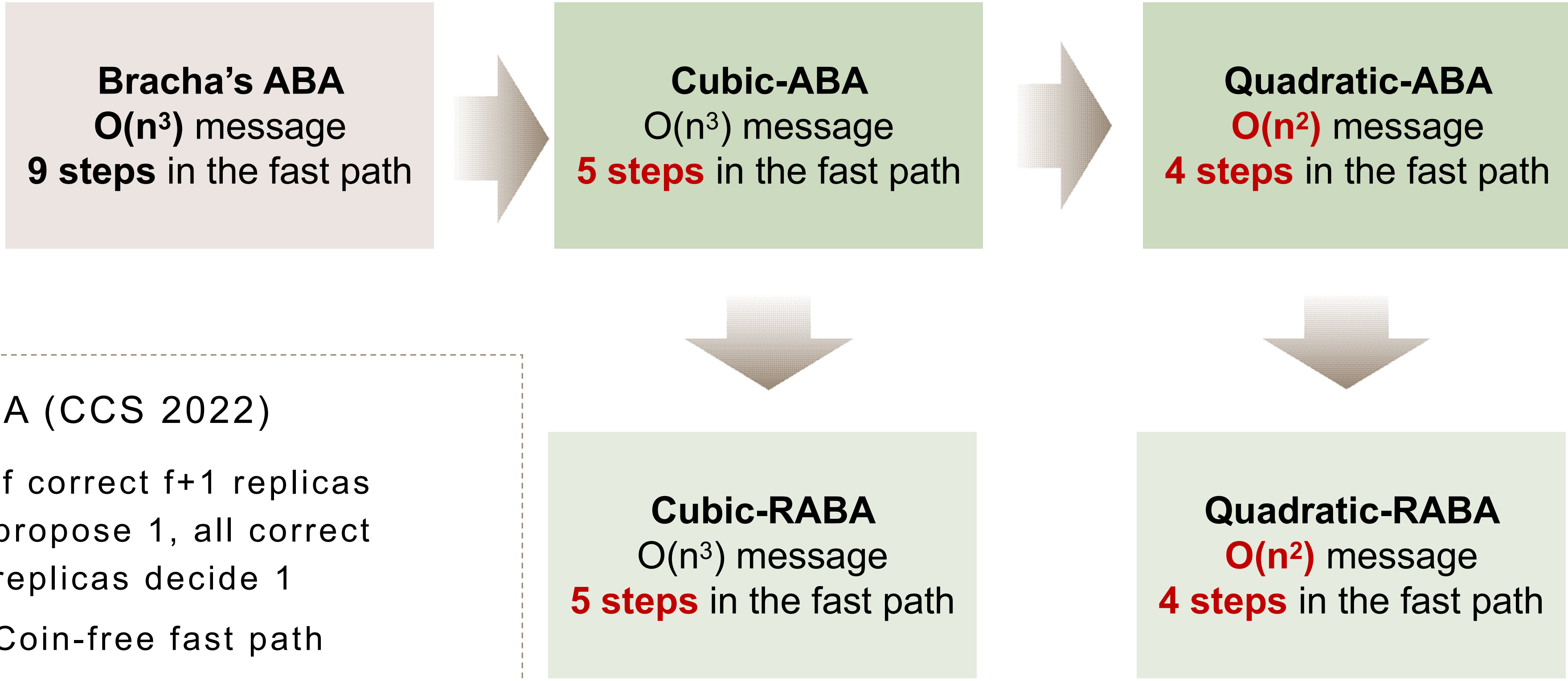
Quadratic-ABA



Idea: Use all-to-all communication only

Any voted value needs to be 'confirmed' by counting the number of votes from the previous step

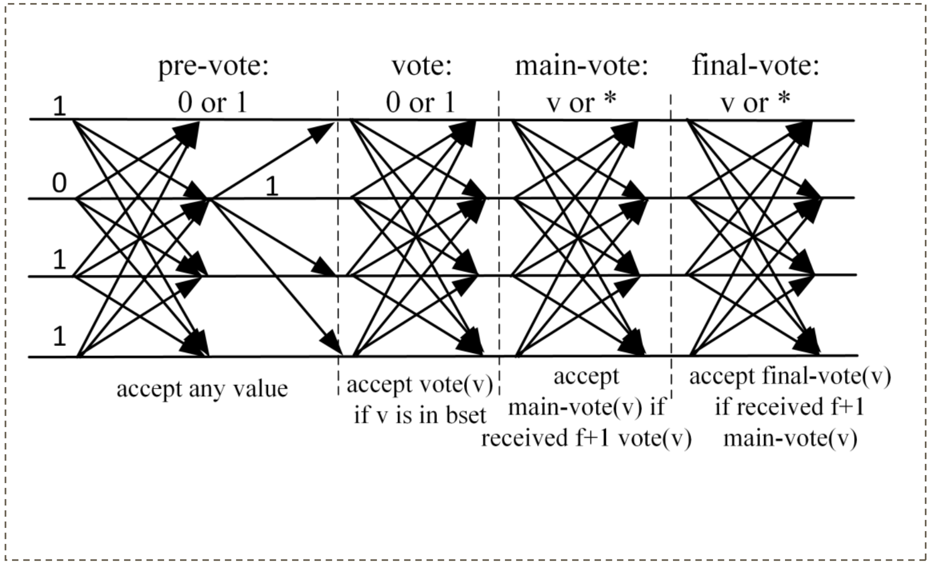
Local Coin Based RABA



- RABA (CCS 2022)
    - if correct f+1 replicas propose 1, all correct replicas decide 1
    - Coin-free fast path

Our RABA

Quadratic-RABA



Idea: Use all-to-all communication only

Any voted value needs to be 'confirmed' by counting the number of votes from the previous step

```
initialization
  r ← 0                                                        {round}
func propose(v)
  broadcast-vote(v)
  start round 0
func repropose(v)
  broadcast-vote(v)
func broadcast-vote(v)
  if pre-vote_0(v) has not been sent, broadcast pre-vote_0(v)
  if v = 1
    bset_0 ← bset_0 ∪ {1}
    if vote_0() has not been sent, broadcast vote_0(1)
    if main-vote_0() has not been sent, broadcast main-vote_0(1)
    if final-vote_0() has not been sent, broadcast final-vote_0(1)
round r
  if r > 0, broadcast pre-vote_r(iv_r)
  upon receiving pre-vote_r(v) from f+1 replicas
    if pre-vote_r(v) has not been sent, broadcast pre-vote_r(v)
  upon receiving pre-vote_r(v) from 2f+1 replicas
    bset_r ← bset_r ∪ {v}
  wait until bset_r ≠ ∅
    if vote_r() has not been sent
      broadcast vote_r(v) where v ∈ bset_r
   upon receiving n − f vote_r() such that for each received
vote_r(b), b ∈ bset_r
    if there are n − f vote_r(v)
      broadcast main-vote_r(v)
    else broadcast main-vote_r(∗)
   upon receiving n − f main-vote_r() such that for each
main-vote_r(v): 1) if r = 0, v ∈ bset_r, 2) if r > 0, at least f+1 vote_r(v)
have been received; for each main-vote_r(∗), bset_r = {0,1}
    if there there are n − f main-vote_r(v)
      broadcast final-vote_r(v)
    else broadcast final-vote_r(∗)
   upon receiving n − f final-vote_r() such that for each
final-vote_r(v), 1) if r = 0, v ∈ bset_r, 2) at least f+1 main-vote_r(v)
have been received; for each final-vote_r(∗), bset_r = {0,1}
    if there there are n − f final-vote_r(v)
      iv_{r+1} ← v, decide v
    else if there are only final-vote_r(v) and final-vote_r(∗)
      iv_{r+1} ← v
    else
      if r = 0, c ← 1                         {coin in the first round is 1}
      else c ← Random()                       {obtain local coin}
      iv_{r+1} ← c
  r ← r+1
```

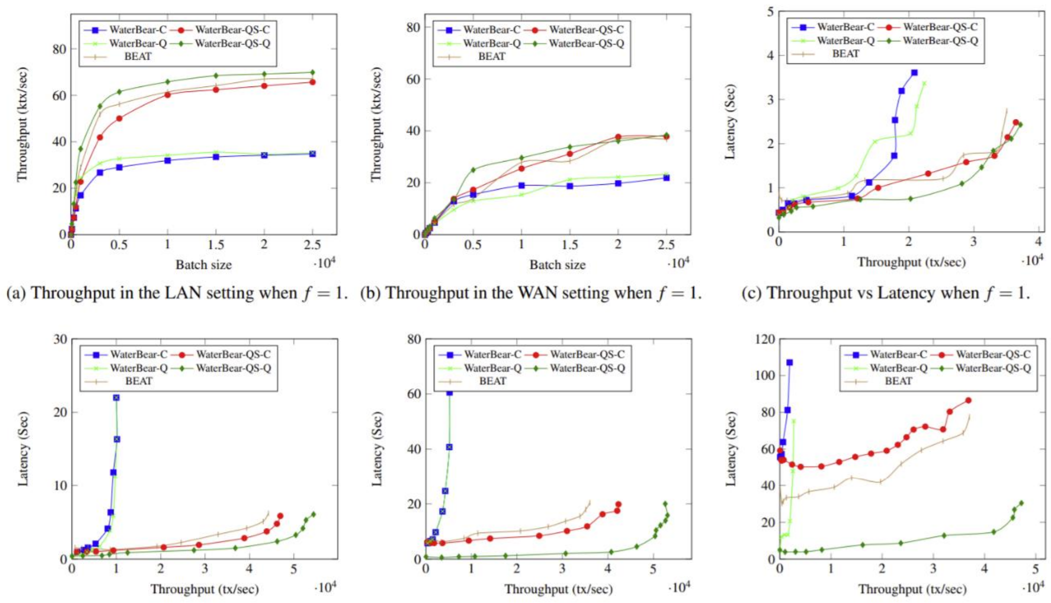
Evaluation

protocol	reference implementation	RBC	RABA	authenticated channels
WaterBear	WaterBear-C	Bracha's RBC [14]	Cubic-RABA (this paper)	HMAC*
	WaterBear-Q	Bracha's RBC [14]	Quadratic-RABA (this paper)	HMAC*
WaterBear-QS	WaterBear-QS-C	CT RBC [18]	Cubic-RABA (this paper)	HMAC
	WaterBear-QS-Q	CT RBC [18]	Quadratic-RABA (this paper)	HMAC

- Golang
- Evaluated 5 protocols in total
  - 4 new ones (WaterBear family)
  - BEAT (CCS 2018)
- AWS m5.xlarge, 4 vCPU, 16GB memory
- up to 61 instances

Results

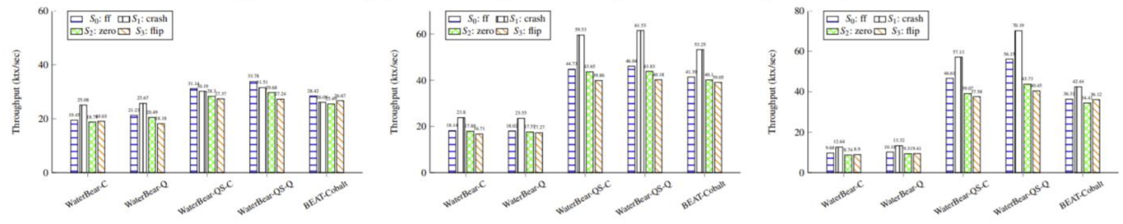
- All WaterBear-QS protcols outperform BEAT
  - n=16, WaterBear-QS-Q has 1/8 latency and 1.23x throughput compared to BEAT
  - Due to the use of PACE framework
- WaterBear-QS protocols consistently outperform WaterBear protocols
  - Communication is important!
- Building efficient quantum-secure asynchronous BFT is possible



(a) Throughput in the LAN setting when $f = 1$. (b) Throughput in the WAN setting when $f = 1$. (c) Throughput vs Latency when $f = 1$.

(d) Throughput vs Latency when $f = 5$. (e) Throughput vs Latency when $f = 10$. (f) Throughput vs Latency when $f = 20$.

Figure 8: Throughput vs latency on m5.xlarge instances for $f = 1$ to $f = 20$.



(a) Throughput for $f = 1$. (b) Throughput for $f = 2$. (c) Throughput for $f = 5$.

Figure 9: Performance of the protocols in failure scenarios.

WaterBear: Asynchronous BFT with Information-Theoretic Security and Quantum Security

Haibin Zhang, Sisi Duan, Boxin Zhao, Liehuang Zhu

- **Quadratic-ABA and Cubic-ABA:** Efficient local-coin based asynchronous binary agreement (ABA) protocols
- **WaterBear Family:** Efficient asynchronous Byzantine fault-tolerant (BFT) protocols with stronger security guarantees

Sisi Duan

Tsinghua University and Zhongguancun Laboratory

duansisi@tsinghua.edu.cn

Usenix Security 2023

清華大學
Tsinghua University