Georgia Tech College of Computing

Whitewash: Securely Outsourcing Garbled Circuit Generation

MSR Workshop on Applied Multi-Party Computation
February 2014

Henry “Hank” Carter, Charles Lever,
Patrick Traynor

SMC on mobile devices

- Mobile devices loaded with private and context-sensitive information and applications that use this information
- Highly constrained system resources (memory, power, processing, communication)

Why don't we have mobile SMC?

- The dominant two-party construction, garbled circuits, requires too much memory and processing power
- Special purpose protocols can be optimized, but no efficient general purpose techniques
- Wish: an efficient mobile two-party SFE protocol that generalizes to any function

Head in the clouds

- Given a technique for performing SFE between servers, can we outsource expensive operations to the cloud?
- How trustworthy is the cloud?
- Secure outsourcing requires mechanisms for
  - Hiding inputs and outputs
  - Ensuring the cloud follows the protocol

Setting

- A limited mobile device (Bob) communicating with a web server (Alice). Bob also has access to a cloud service (Cloud).
- Goal: Alice and Bob securely compute a two-party function using garbled circuits.
- Security:
  - Preserve input and output privacy from both the other party and the cloud
  - Security in the malicious setting

Previous work

- Salus Framework (Kamara et al., CCS 2012)
    - First garbled circuit outsourcing scheme
    - Provides malicious and covert secure protocols for outsourcing
- CMTB Outsourcing (USENIX Security 2013)
    - Used outsourced oblivious transfer (OOT) to deliver garbled inputs of phone to the evaluating cloud
    - Phone performs some checks to ensure that the cloud doesn't "lazily" check

Can we do better?

- OT on the phone
  - Reduced, but slow. Bottleneck for parallelization
- Consistency checks
  - Ensure cloud is behaving, but require exponentiations
  - Shown to be very slow on mobile devices
- Restricted collusion model
- Can we improve on these techniques?

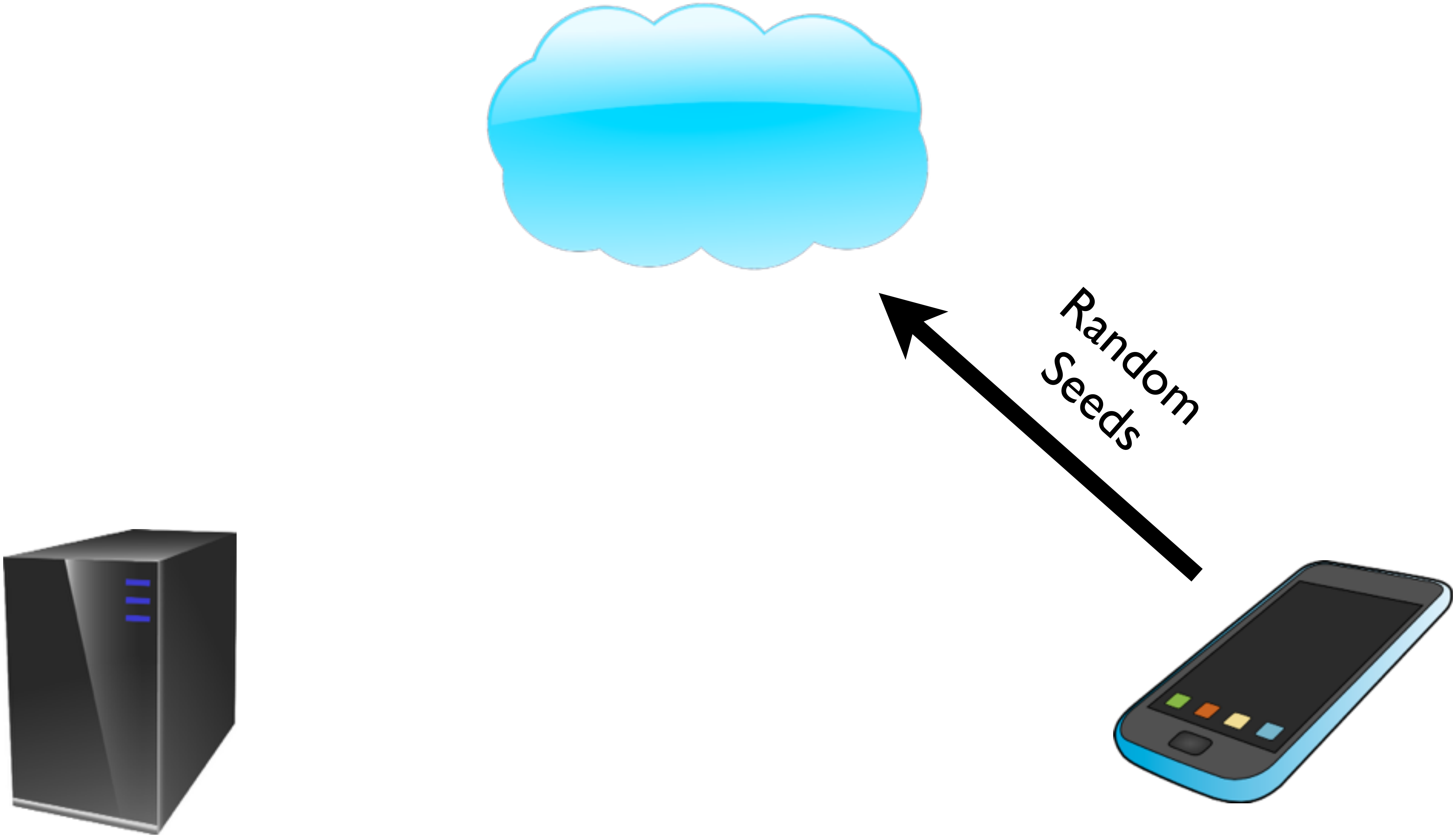
Whitewash

- Consider reversing outsourced party (i.e., outsource generation instead of evaluation)
- Have the mobile device produce random seeds, server generates garbled circuits
- Standard OT/evaluation between servers to garble evaluating server's inputs

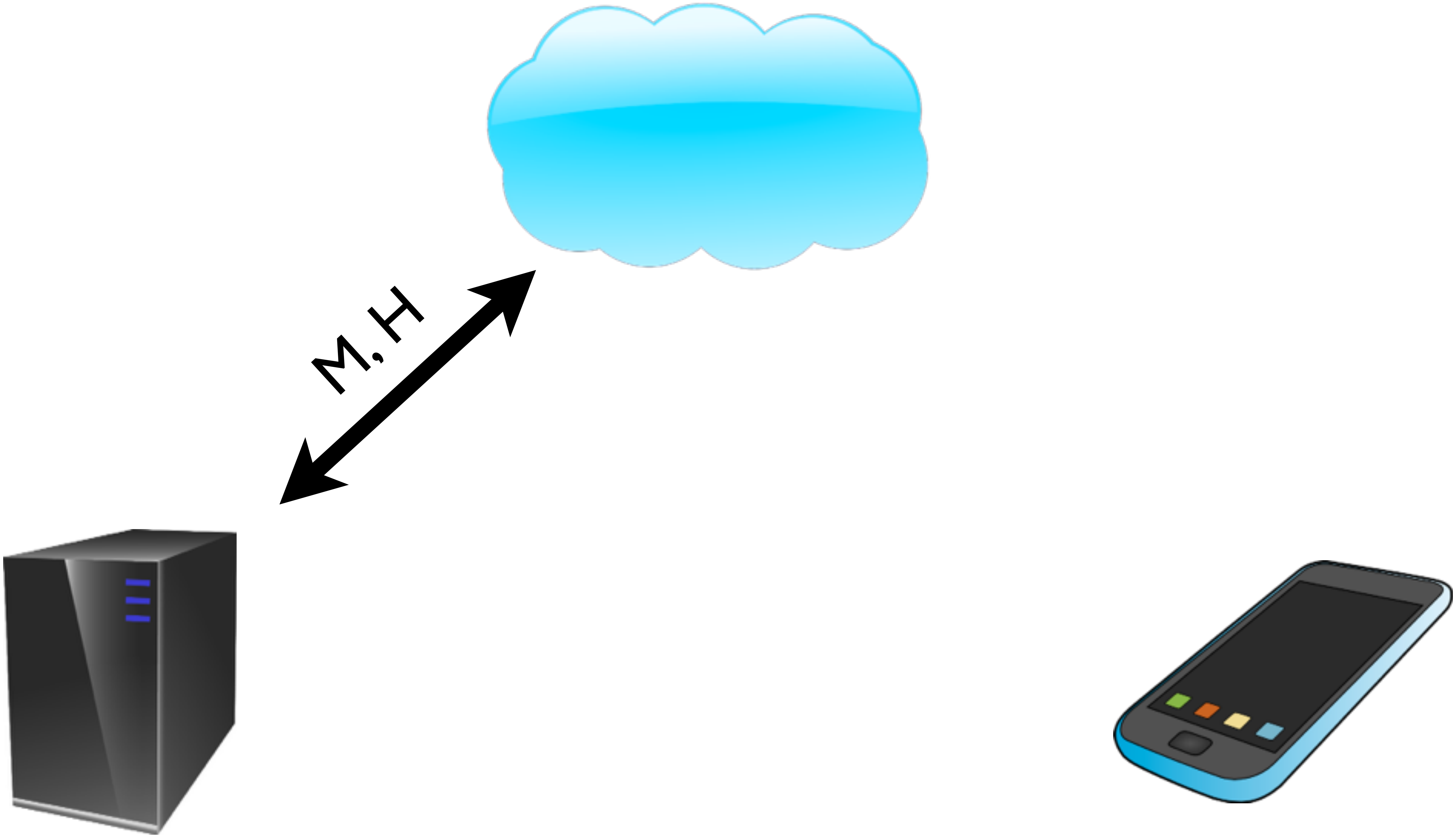
Whitewash

- Built on two garbled circuit advances:
  - shelat-Shen (CCS ’13) Uses only symmetric-key operations (outside of OT)
  - PCF (Kreuter et al. USENIX ’13) compiles smaller circuits with compact program representations
- Improved efficiency for both mobile and servers
- Improved Security for certain types of collusion
- Protocol takes place in 6 phases

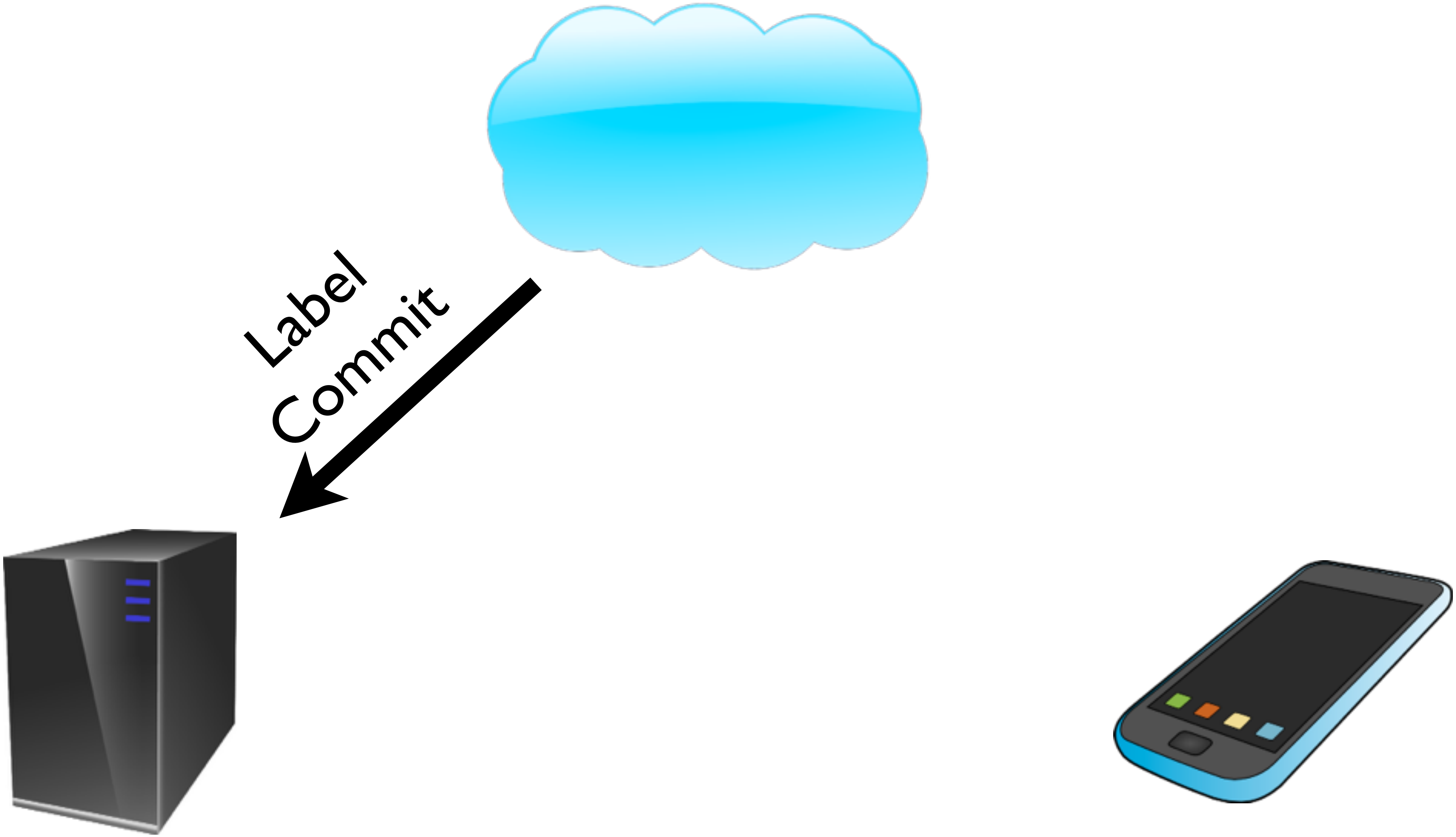
Phase 1-2: Parameter Setup



Phase 1-2: Parameter Setup

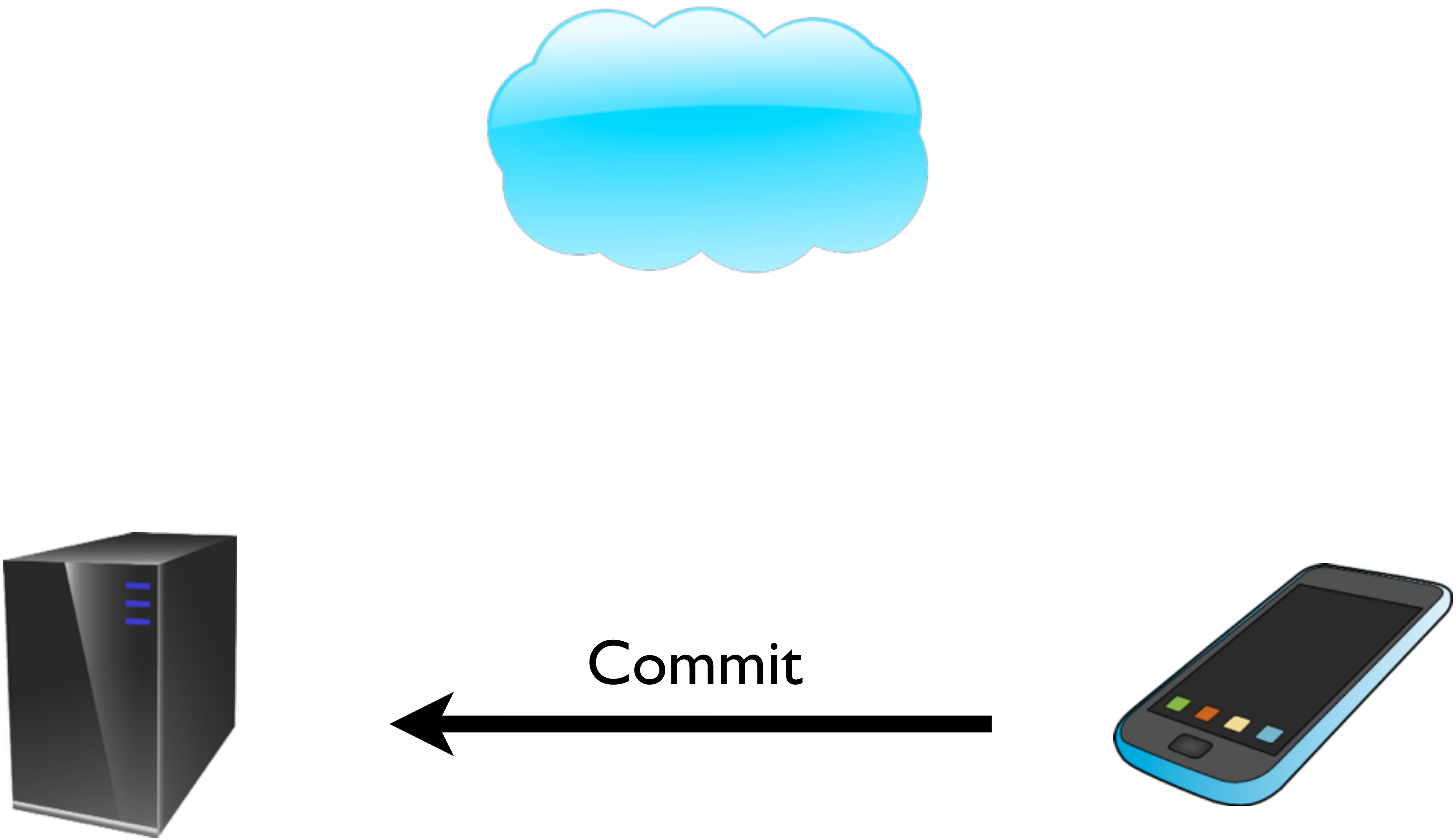


Phase 1-2: Parameter Setup

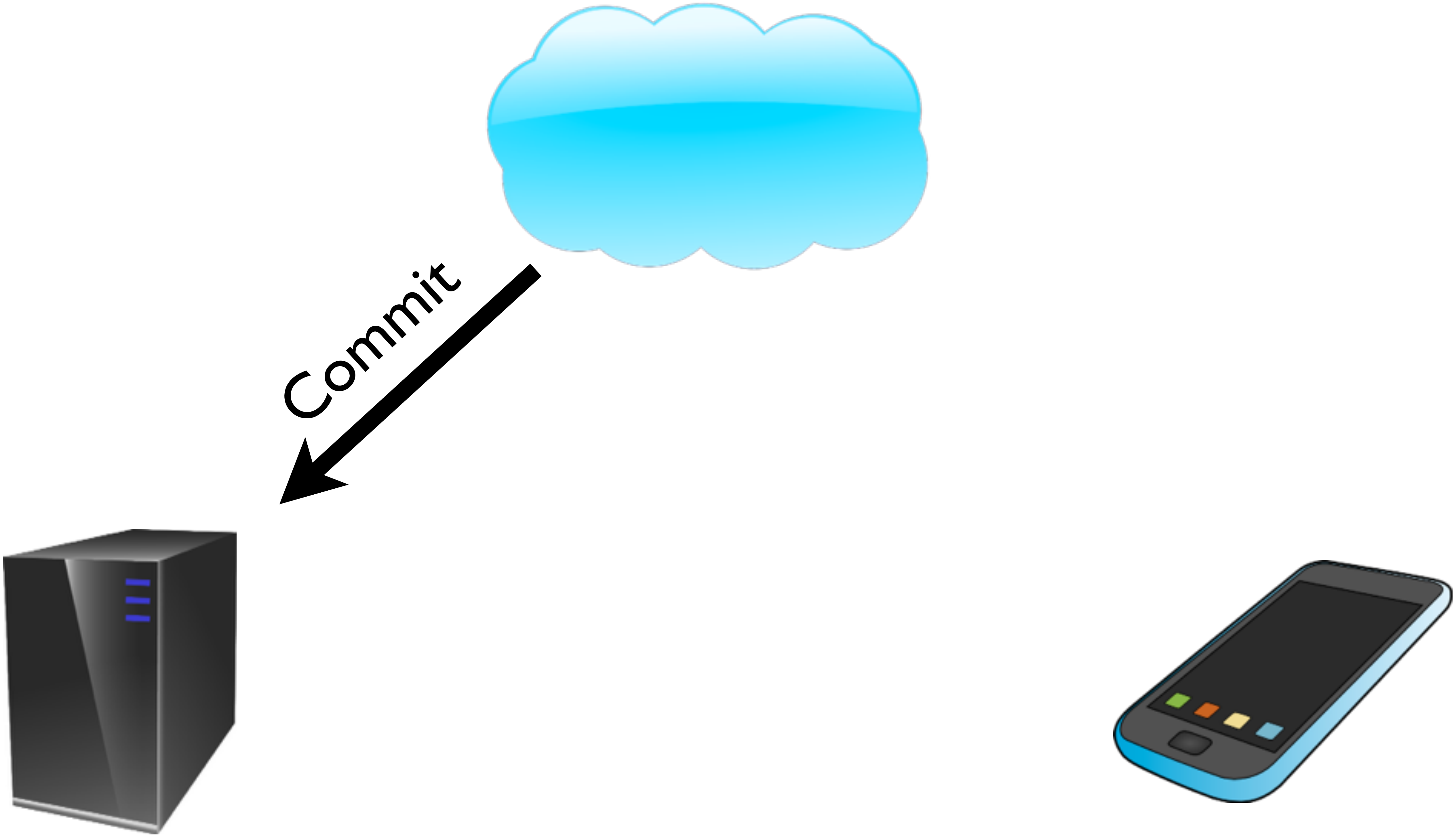


Phase 3: Input Commitment

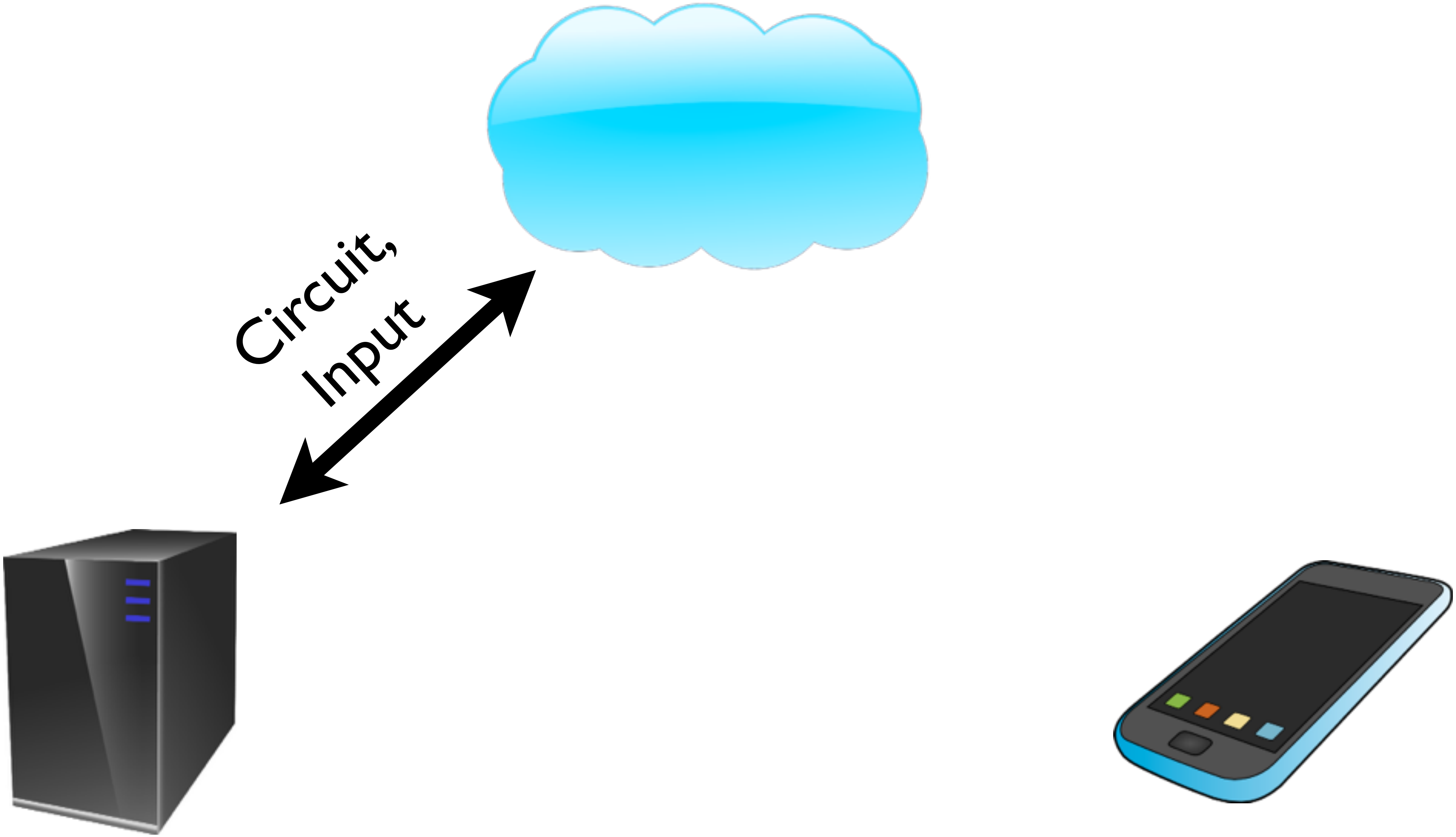
Phase 3: Input Commitment



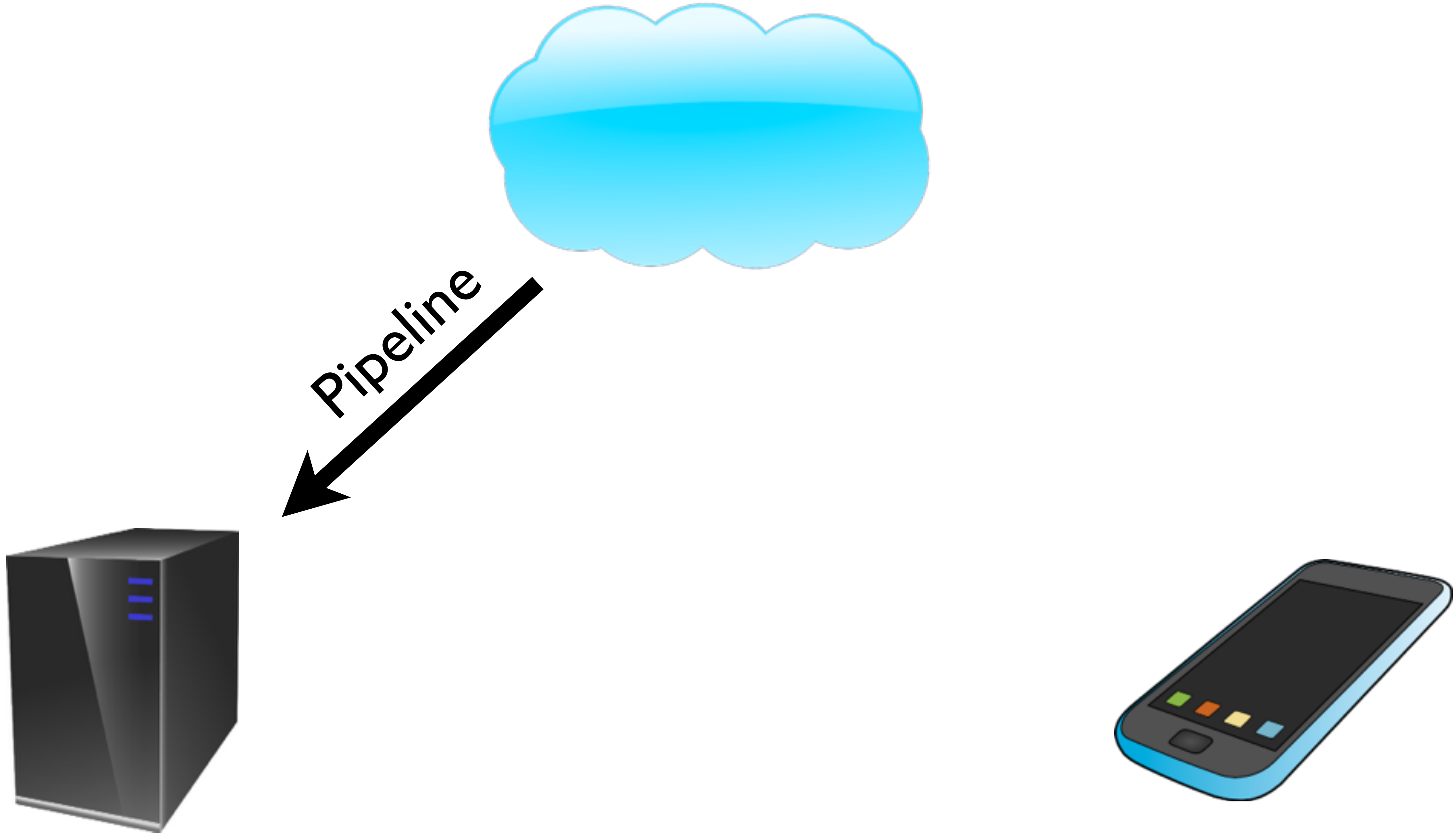
Phase 3: Input Commitment



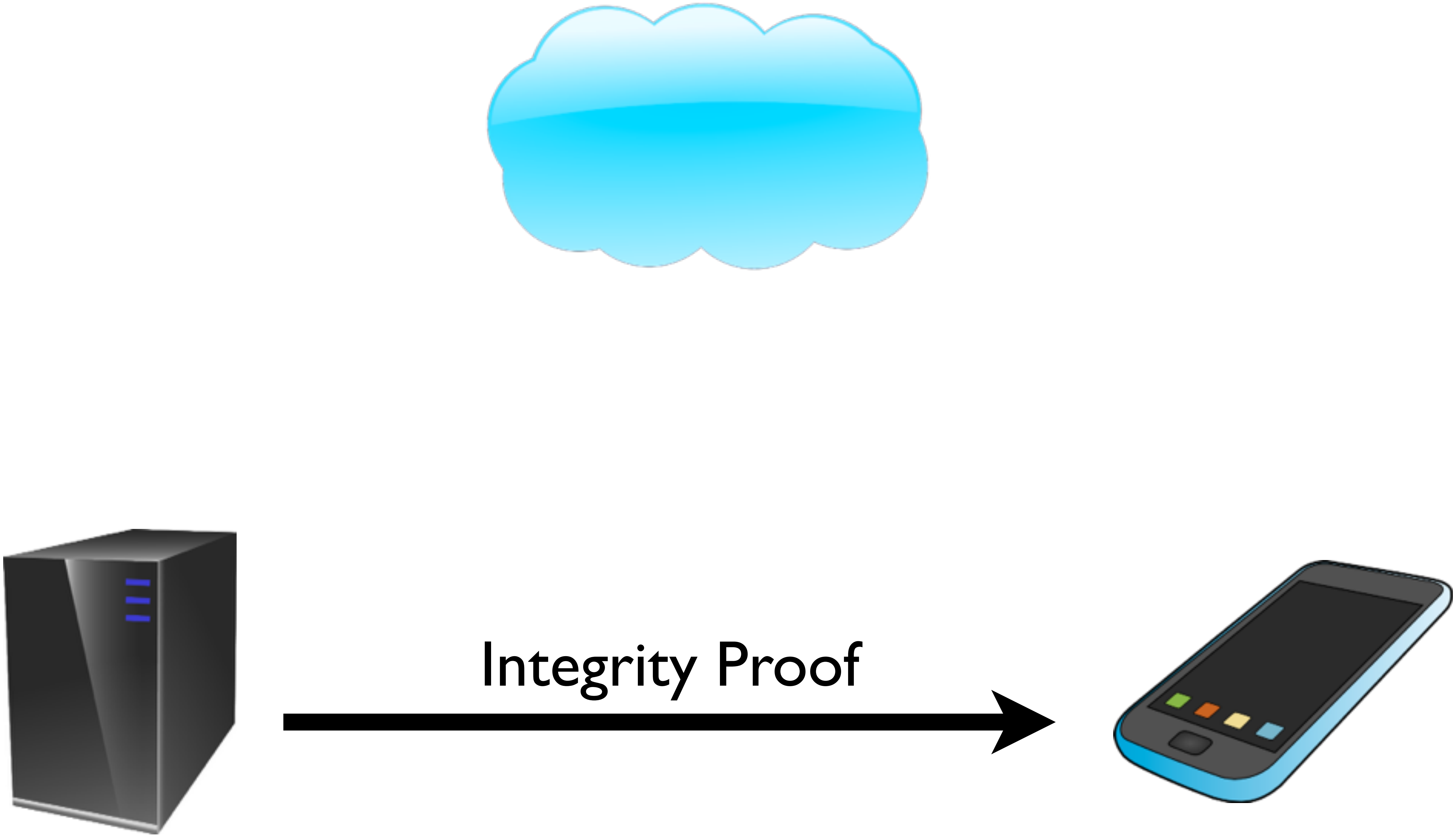
Phase 4: Oblivious Transfers



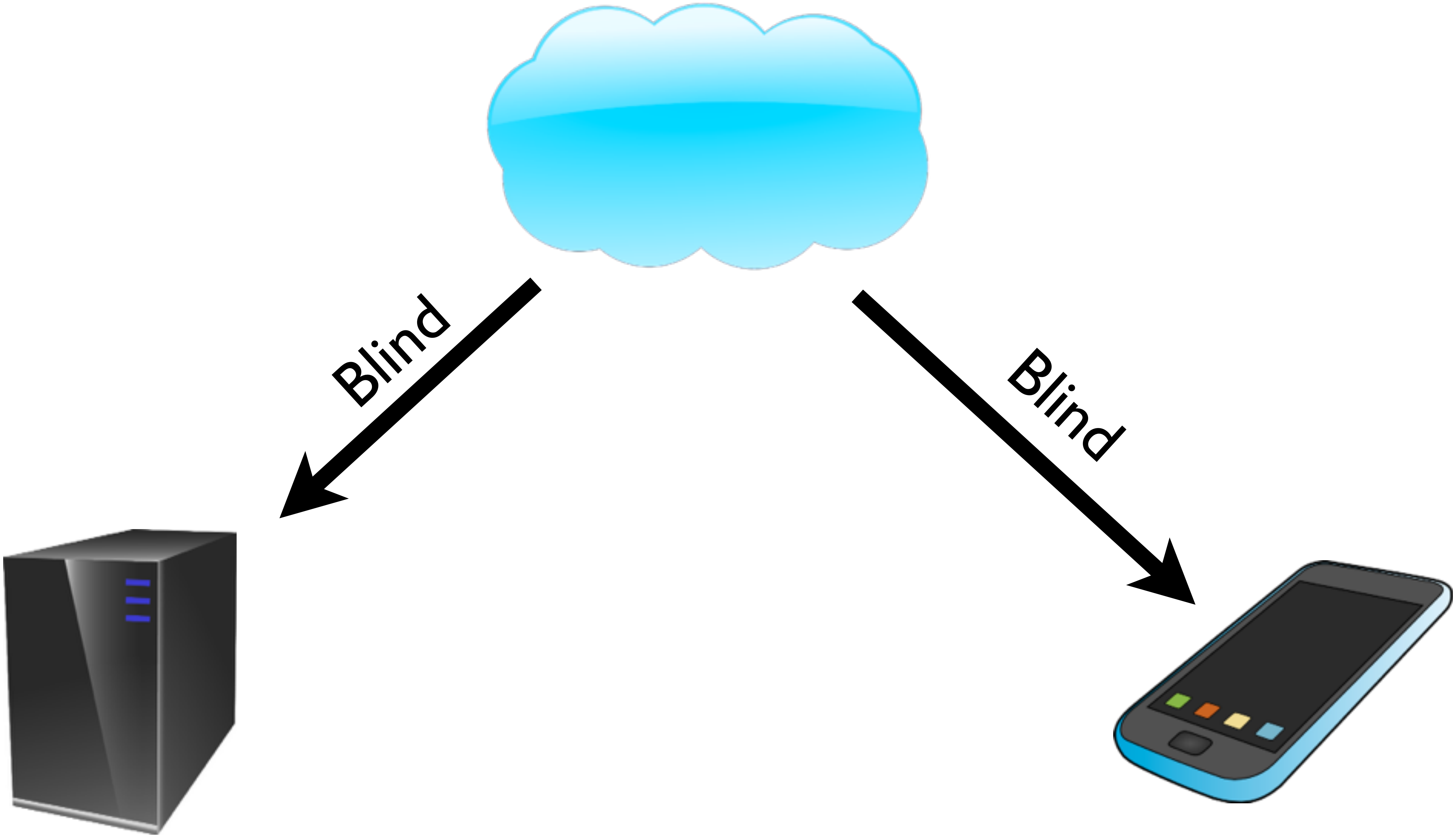
Phase 5: Evaluation



Phase 6: output proof and release



Phase 6: output proof and release



What have we gained?

- No OT on the phone
  - Mobile device generates randomness and input wire labels, so it can garble its own input
- No consistency check on the phone
  - Significantly reduces the number of group algebraic ops required on the device
- Stronger collusion resistance
  - Secure when mobile and cloud collude
- In exchange: randomness generation
  - Can be done a priori to save time.

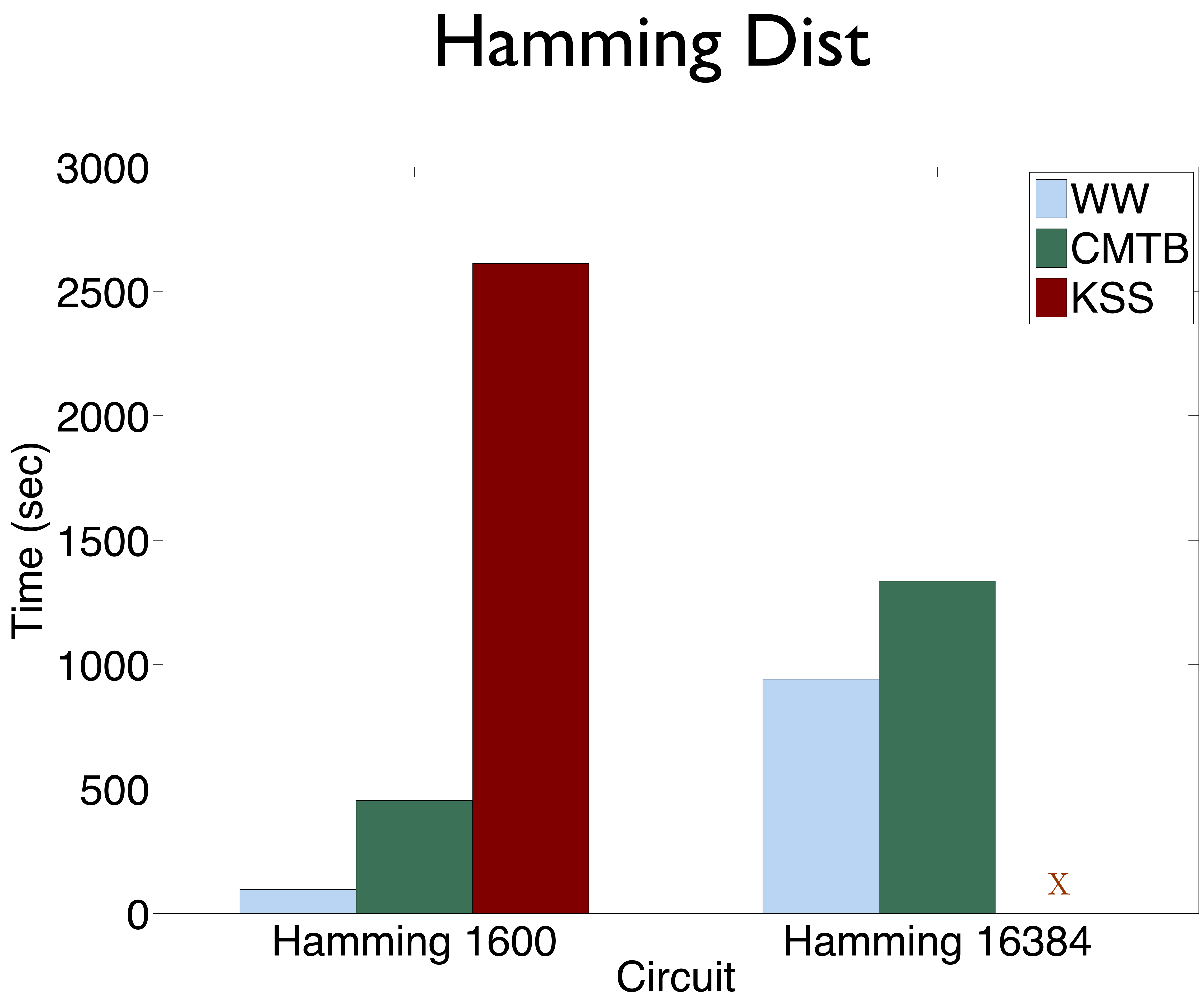
Collusion Assumptions

- Kamara et al. notes that an outsourcing scheme with collusion implies an SFE scheme where one party performs sub-linear work w.r.t. circuit size.
- Previous work assumes NO collusion with the cloud
- Whitewash reduces to shelat-Shen '13 when Bob and Cloud collude
  - Loss of fair release
  - Remains malicious secure
- Realistic scenario: cloud service may collude with the customer

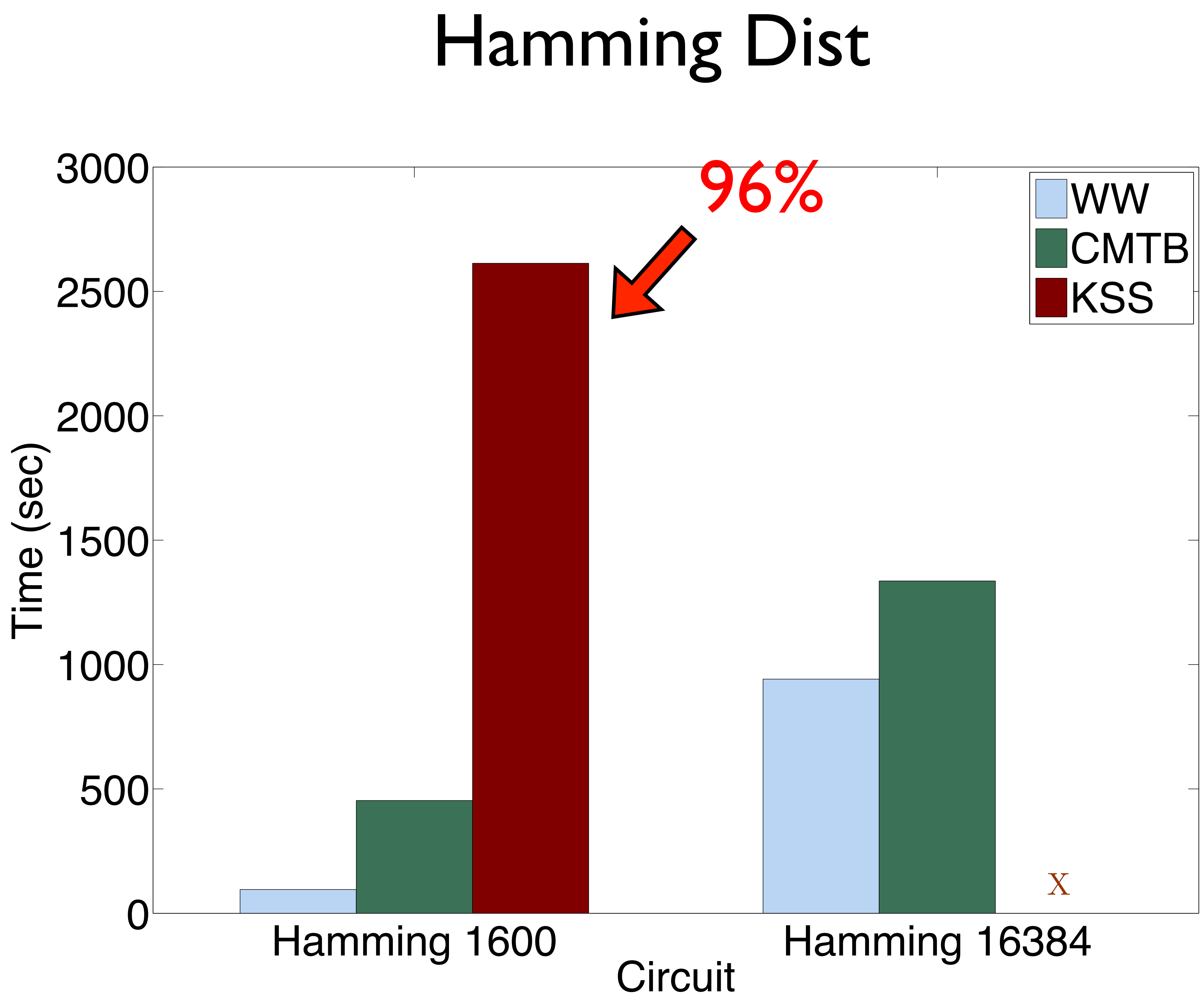
Evaluation

- Server setup
  - 64 core, 1 TB memory
  - 802.11g wireless connection
  - Samsung Nexus One
- Test circuits
  - Hamming Distance
  - Matrix-Multiplication
  - RSA
- Comparison against KsS, CMTB

Improvement: execution time

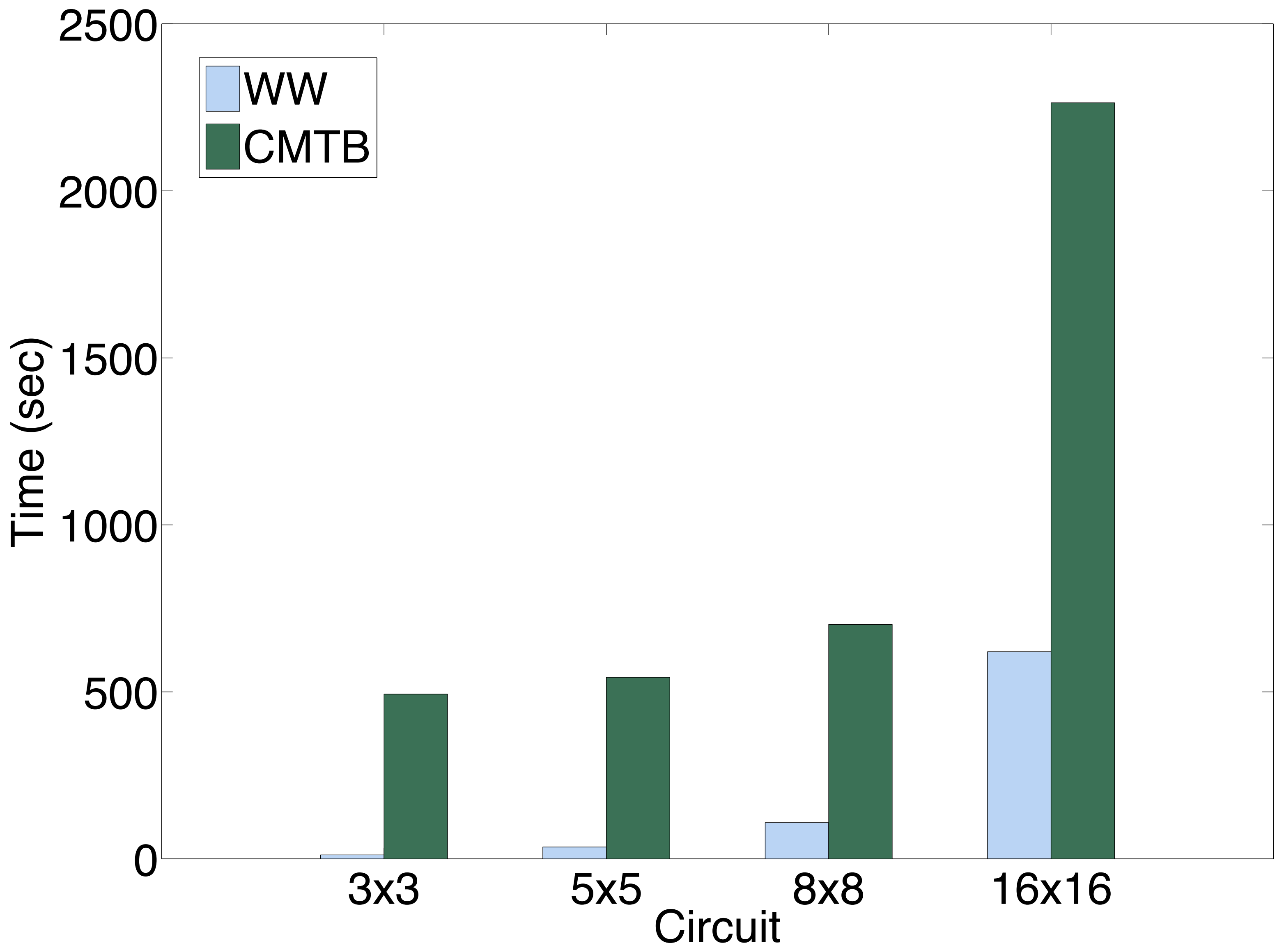


Improvement: execution time

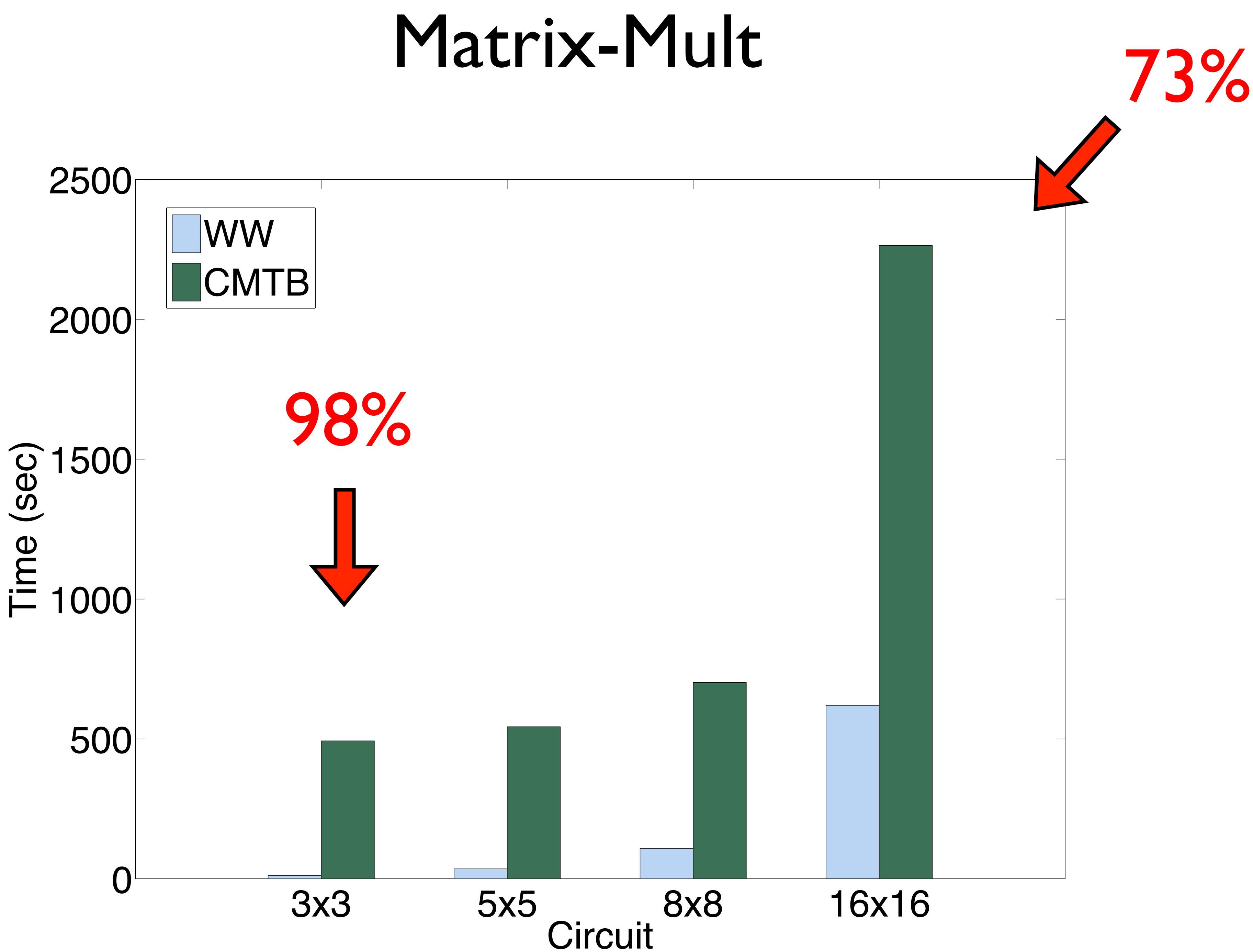


Improvement: execution time

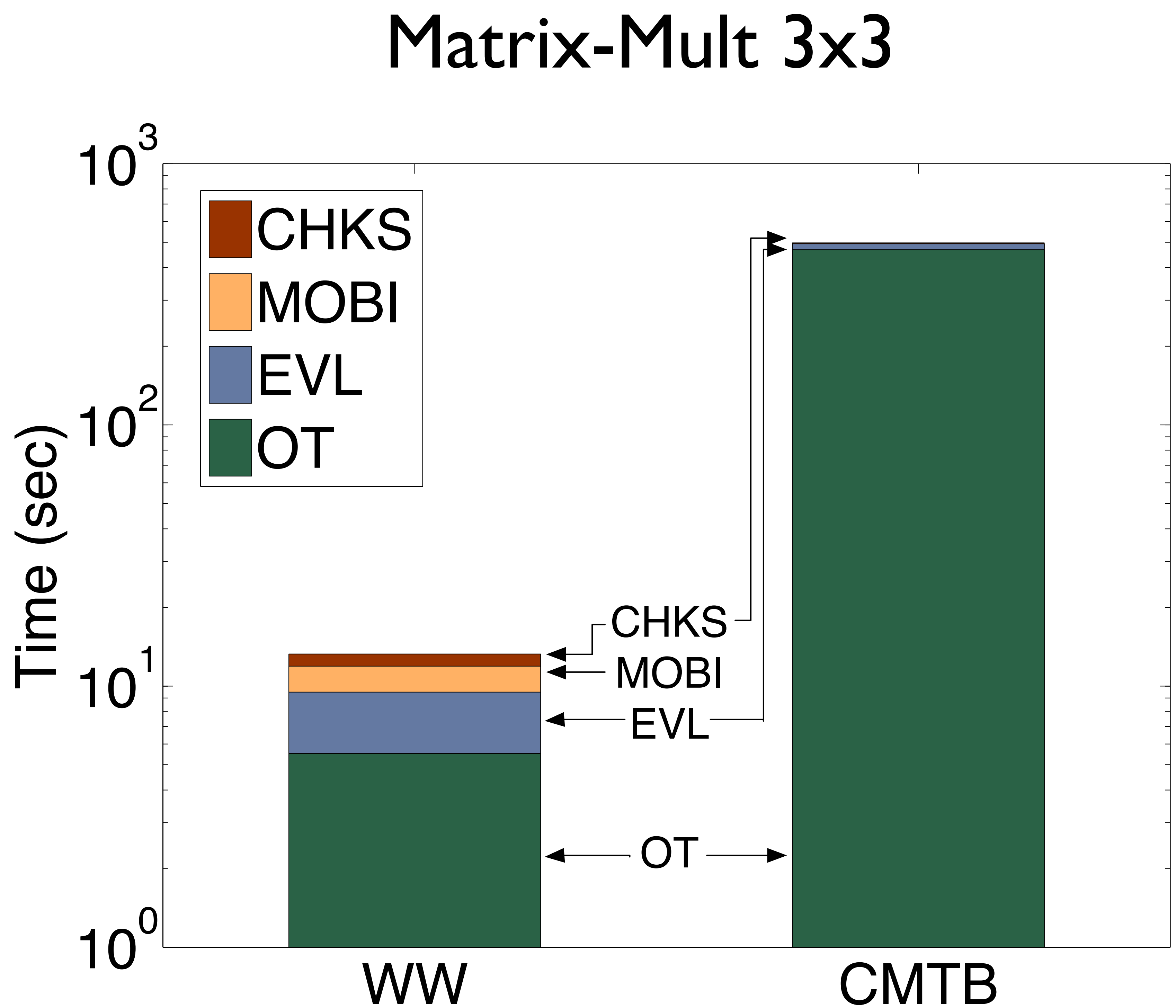
Matrix-Mult



Improvement: execution time



Improvement: execution time



Improvement: Bandwidth

	Bandwidth (MB)			Reduction Over	
Circuit	WW	CMTB	KSS	CMTB	KSS
Hamming (1600)	23.56	41.05	240.33	42.62%	90.20%
Hamming (16384)	241.02	374.03	x	35.56%	x
Matrix (3x3)	4.26	11.50	x	62.97%	x
Matrix (5x5)	11.79	23.04	x	48.82%	x
Matrix (8x8)	30.15	51.14	x	41.05%	x
Matrix (16x16)	120.52	189.52	x	36.41%	x
RSA-256	3.97	x	x	x	x

Improvement: Bandwidth

	Bandwidth (MB)			Reduction Over	
Circuit	WW	CMTB	KSS	CMTB	KSS
Hamming (1600)	23.56	41.05	240.33	42.62%	90.20%
Hamming (16384)	241.02	374.03	x	35.56%	x
Matrix (3x3)	4.26	11.50	x	62.97%	x
Matrix (5x5)	11.79	23.04	x	48.82%	x
Matrix (8x8)	30.15	51.14	x	41.05%	x
Matrix (16x16)	120.52	189.52	x	36.41%	x
RSA-256	3.97	x	x	x	x

Summary

- New protocol for outsourcing garbled circuit generation
- Removes OT and public key operations performed on the mobile device
- Performance evaluations show up to 98% improvement in evaluation time and 63% improvement in bandwidth usage

Questions?

Thanks for your attention!

carterh@gatech.edu