Collaborative, Privacy-Preserving Data Aggregation at Scale

Michael J. Freedman
Princeton University

Joint work with: Benny Applebaum, Haakon Ringberg, Matthew Caesar, and Jennifer Rexford

Problem:
Network Anomaly Detection

Collaborative anomaly detection

- Some attacks look like normal traffic
  - *e.g.*, SQL-injection, application-level DoS [Srivatsa TWEB '08]
- Is it a DDoS attack or a flash crowd? [Jung WWW '02]

Collaborative anomaly detection

- Targets (victims) could correlate attacks/attackers

[Katti IMC ’05], [Allman Hotnets ‘06], [Kannan SRUTI ‘06], [Moore INFOC ‘03]

Problem:
Network Anomaly Detection

Solution:

- Aggregate suspect IPs from many ISPs
- Flag those IPs that appear > threshold τ

Problem:

Distributed Ranking

Solution:

- Collect domain statistics from many users
- Aggregate data by domain

Problem:

…

Solution:

- Aggregate (id, data) from many sources
- Analyze data grouped by id

But what about privacy?

What inputs are submitted?

Who submitted what?

Data Aggregation Problem

- Many participants, each with (key, value) observation
- Goal: Aggregate observations by key

Key	Values
k_1	$\mathcal{A}(v_a, v_b)$
k_2	$\mathcal{A}(v_i, v_j, v_k)$
…	
k_n	$\mathcal{A}(v_x)$

Data Aggregation Problem

- Many participants, each with (key, value) observation
- Goal: Aggregate observations by key

Key	Values
k_1	$F(A(v_a, v_b))$
k_2	$F(A(v_i, v_j, v_k))$
…	
k_n	$F(A(v_x))$

PDA: Only release the value column

CR-PDA: Plus keys whose values satisfy some func

Data Aggregation Problem

- Many participants, each with (key, value) observation
- Goal: Aggregate observations by key

Key	Values
k_1	$\Sigma\,(1, 1) \quad \stackrel{?}{\geq} \tau$
k_2	$\Sigma\,(1, 1, 1) \quad \stackrel{?}{\geq} \tau$
…	
k_n	$\Sigma\,(1) \quad \stackrel{?}{\geq} \tau$

PDA: Only release the value column

CR-PDA: Plus keys whose values satisfy some func

Goals

- Keyword privacy: No party learns anything about keys
- Participant privacy: No party learns who submitted what
- Efficiency: Scale to many participants, each with many inputs
- Flexibility: Support variety of computations over values
- Lack of coordination:
  - No synchrony required, individuals cannot prevent progress
  - All participants need not be online at same time

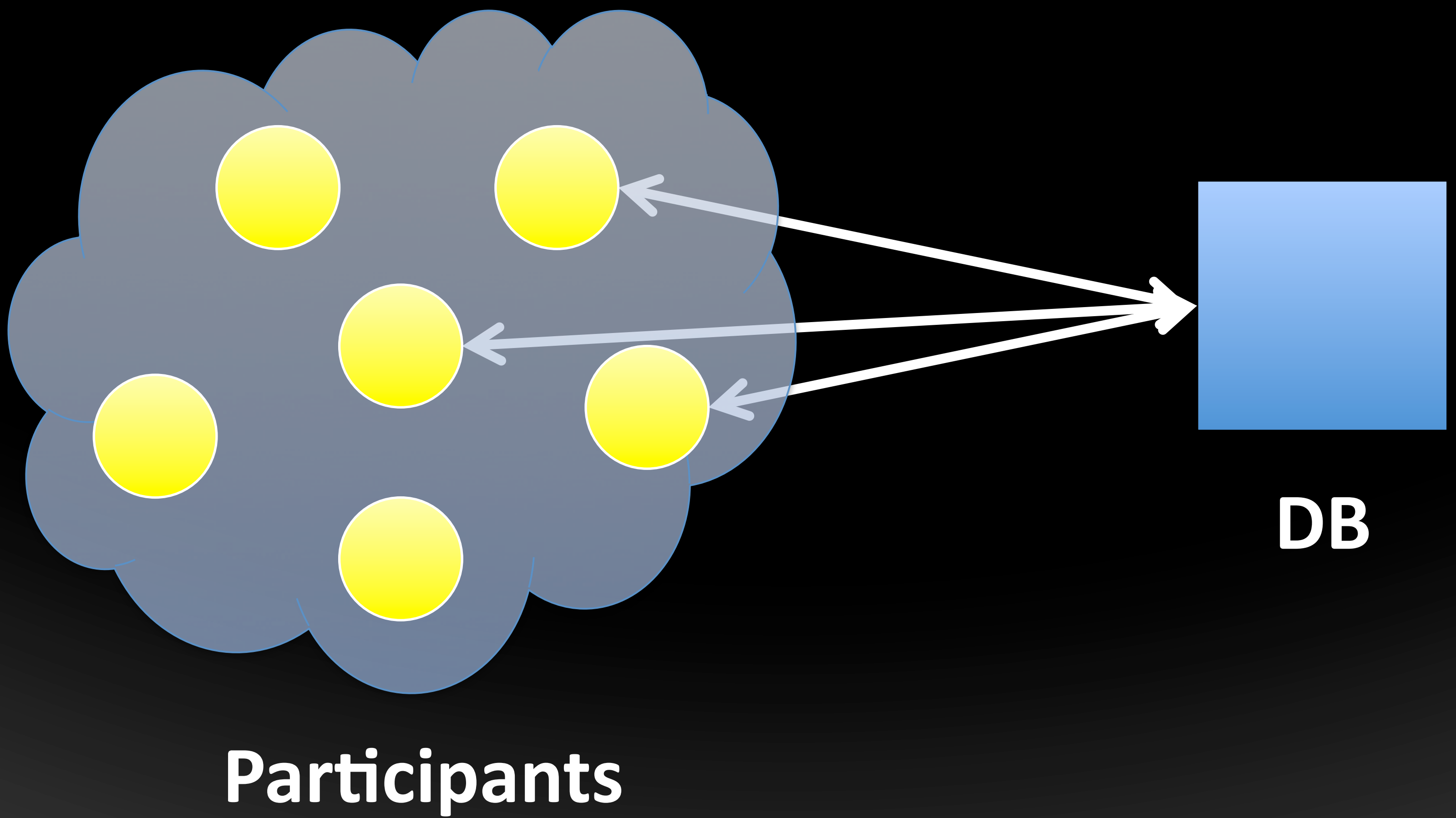
Potential solutions

Decentralized

Approach	Keyword Privacy	Participant Privacy	Efficiency	Flexibility	Lack of Coord
Garbled Circuit Evaluation	Yes	Yes	Very Poor	Yes	No
Multiparty Set Intersection	Yes	Yes	Poor	No	No

- Weaken security assumptions?
  - Assume honest but curious participants?
  - Assume no collusion among malicious participants?
- In large/open setting, easy to operate multiple nodes (so-called “Sybil attack”)

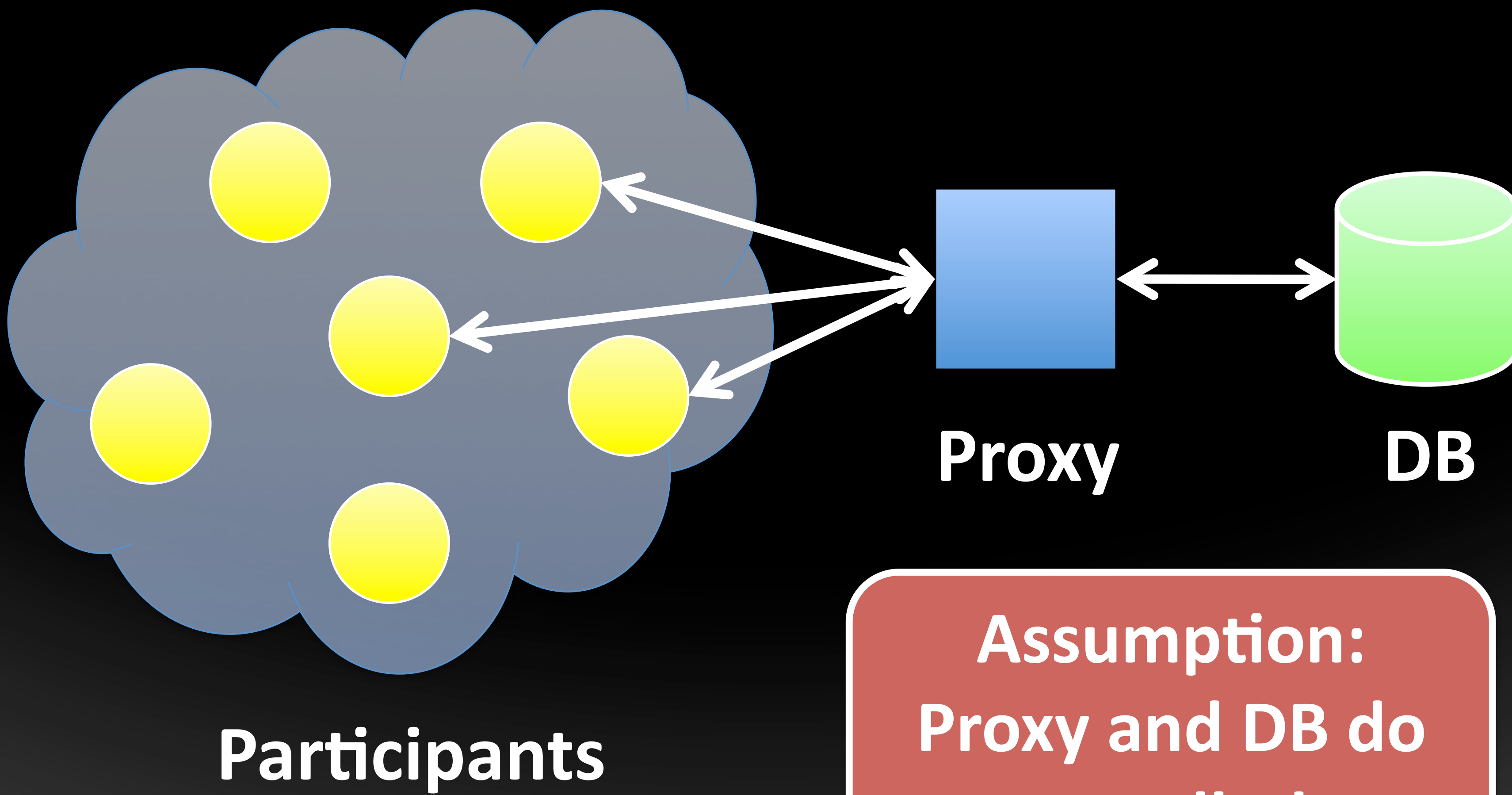
Towards Centralization?



Potential solutions

	Approach	Keyword Privacy	Participant Privacy	Efficiency	Flexibility	Lack of Coord
Decentralized	Garbled Circuit Evaluation	Yes	Yes	Very Poor	Yes	No
	Multiparty Set Intersection	Yes	Yes	Poor	No	No
Centralized	Hashing Inputs	No	No	Very Good	Yes	Yes
	Network Anonymization	No	Yes	Very Good	Yes	Yes

Towards semi-centralization



Assumption: Proxy and DB do not collude

Potential solutions

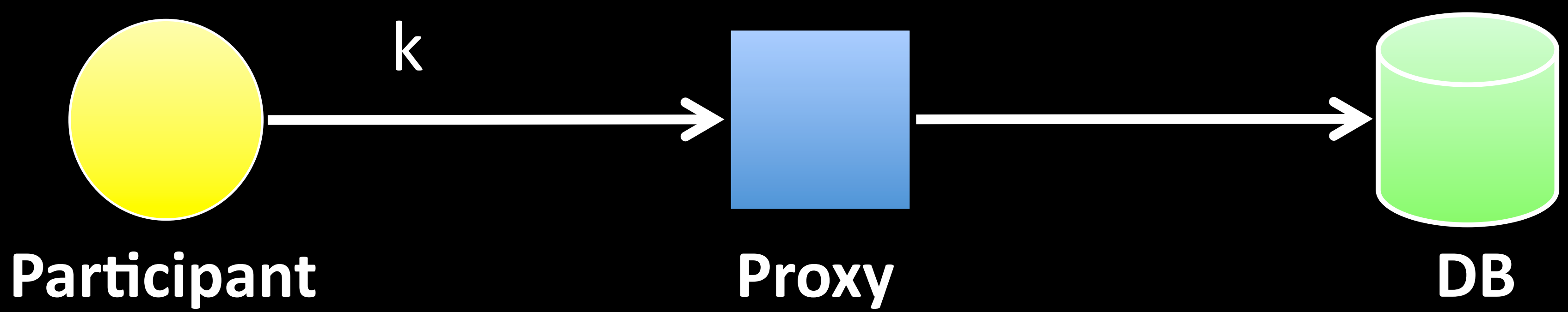
	Approach	Keyword Privacy	Participant Privacy	Efficiency	Flexibility	Lack of Coord
Decentralized	Garbled Circuit Evaluation	Yes	Yes	Very Poor	Yes	No
Decentralized	Multiparty Set Intersection	Yes	Yes	Poor	No	No
Centralized	Hashing Inputs	No	No	Very Good	Yes	Yes
Centralized	Network Anonymization	No	Yes	Very Good	Yes	Yes
	This Work	Yes	Yes	Good	Yes	Yes

Privacy Guarantees

- Privacy of PDA against malicious entities and participants
  - Malicious participant may collude with either malicious proxy or DB, but not both
  - May violate *correctness* in almost arbitrary ways

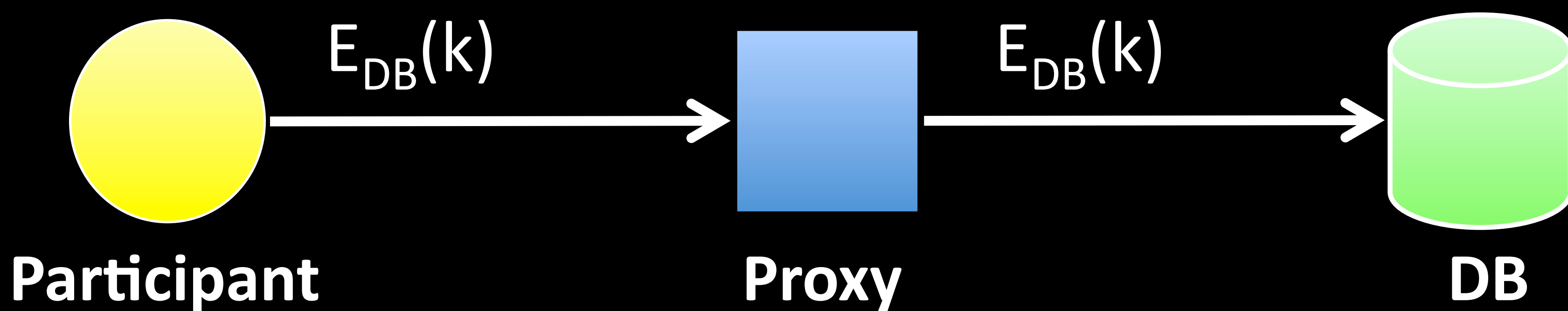
- Privacy of CR-PDA against honest-but-curious entities and malicious participants

PDA Strawman #0



1. Client sends input k

PDA Strawman #1

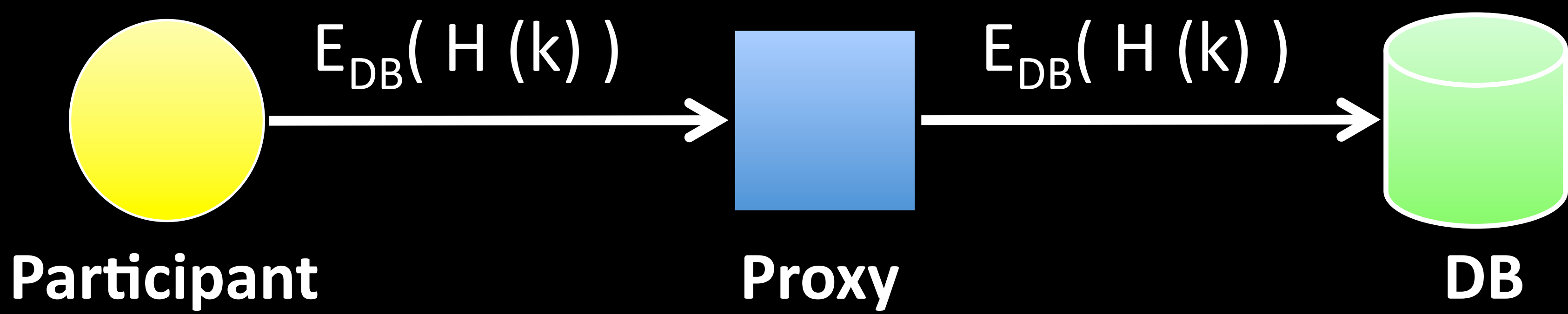


1. Client sends encrypted input k
2. Proxy batches and retransmits
3. DB decrypts input

k	#
1.1.1.1	1
2.2.2.2	9

Violates keyword privacy

PDA Strawman #2

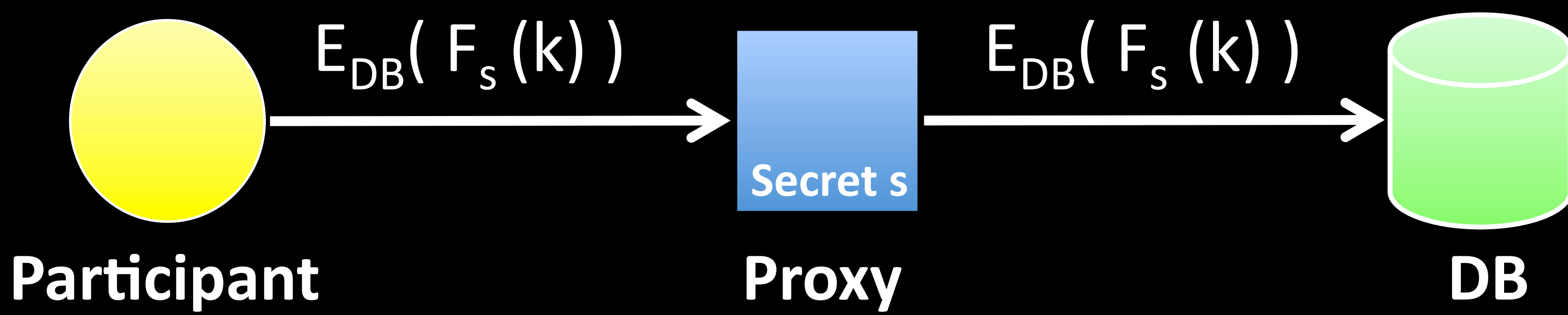


H (k)	#
H(1.1.1.1)	1
H(2.2.2.2)	9

1. Client sends hashes of k
2. Proxy batches and retransmits
3. DB decrypts input

Still violates keyword privacy: IPs drawn from small domains

PDA Strawman #3

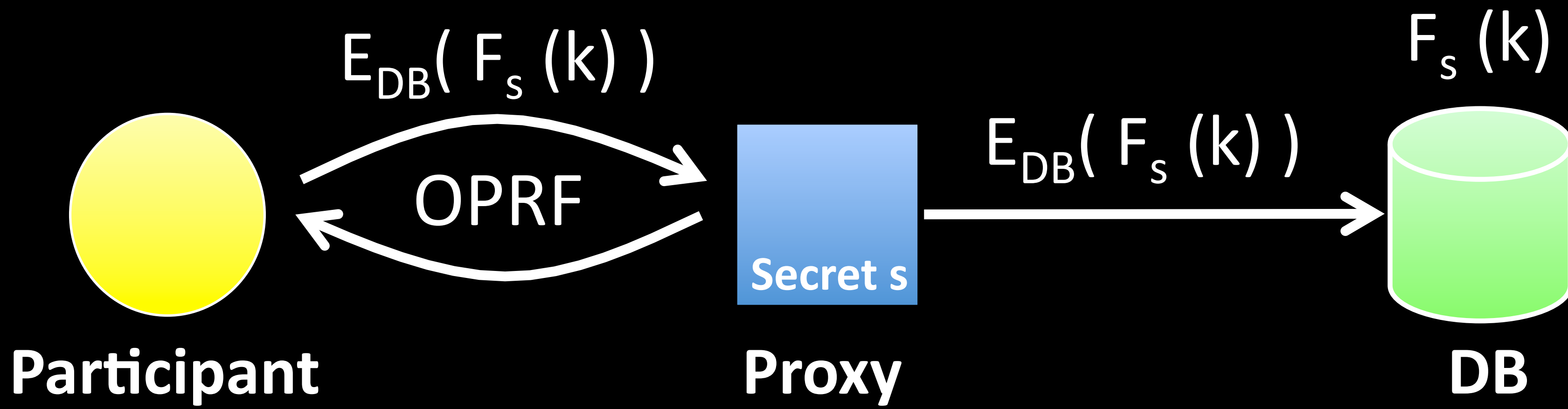


1. Client sends keyed hashes of k
   - Keyed hash function (PRF)
   - Key s known only by proxy

F_s (k)	#
F_s(1.1.1.1)	1
F_s(2.2.2.2)	9

But how do clients learn F_s (IP)) ?

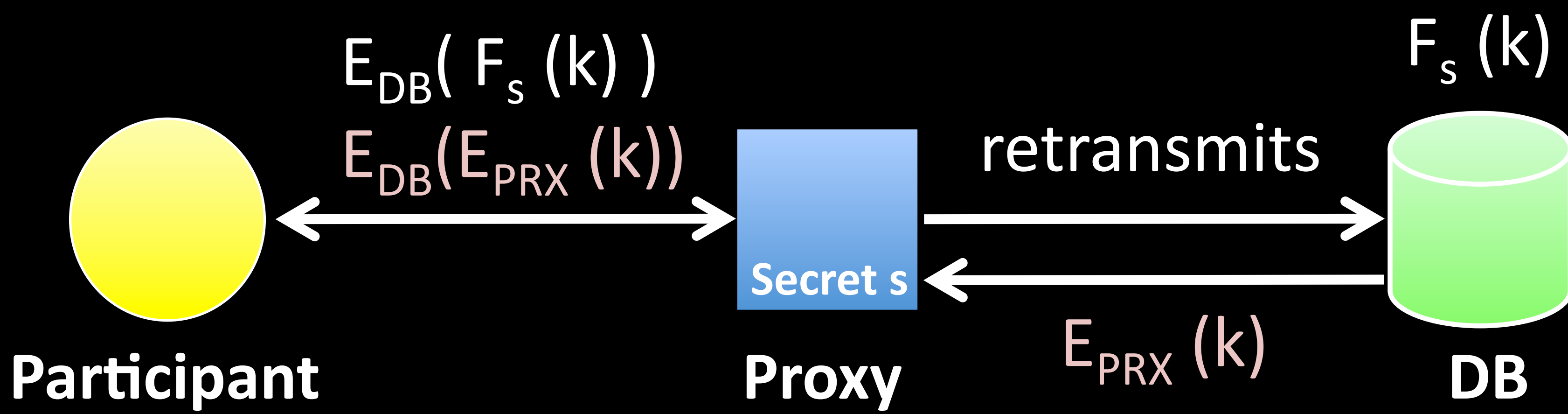
Our Basic PDA Protocol



F_s (k)	#
F_s(1.1.1.1)	1
F_s(2.2.2.2)	9

1. Client sends keyed hashes of k
   – $F_s(x)$ learned by client through Oblivious PRF protocol
2. Proxy batches and retransmits keyed hash
3. DB decrypts input

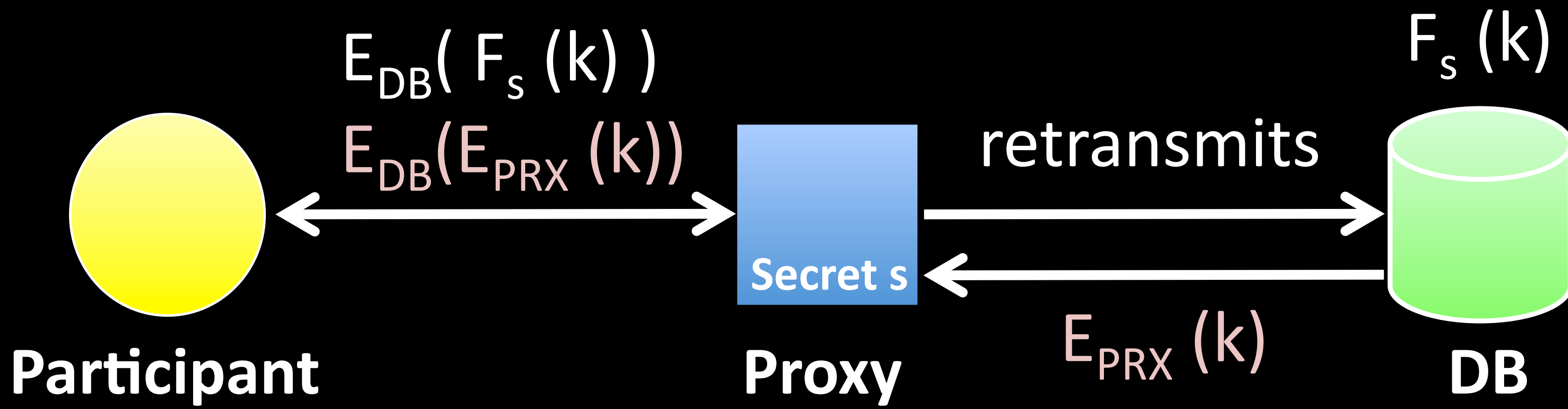
Basic CR-PDA Protocol



F_s (k)	#	Enc'd k
F_s(1.1.1.1)	1	E_{PRX}(1.1.1.1)
F_s(2.2.2.2)	9	E_{PRX}(2.2.2.2)

1. Client sends keyed hashes of k, and encrypted k for recovery
2. Proxy retransmits keyed hash
3. DB decrypts input
4. Identify rows to release and transmit E_{PRX} (k) to proxy
5. Proxy decrypts k and releases

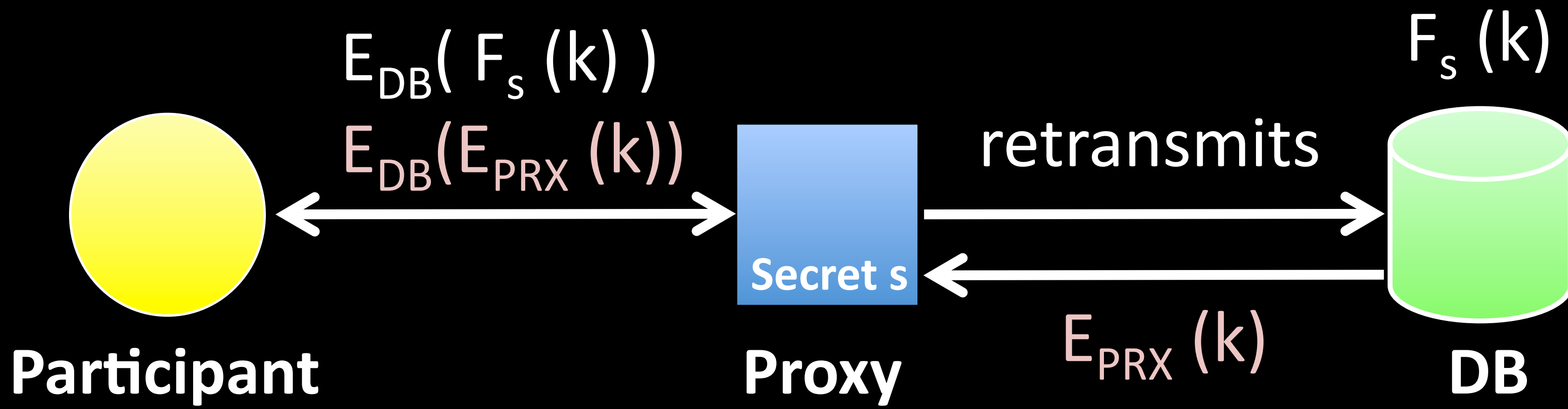
Privacy Properties



- Keyword privacy: Nothing learned about unreleased keys
- Participant privacy: Key ←→ Participant not learned

- Any coalition of HBC participants
- HBC coalition of proxy and participants
- HBC database

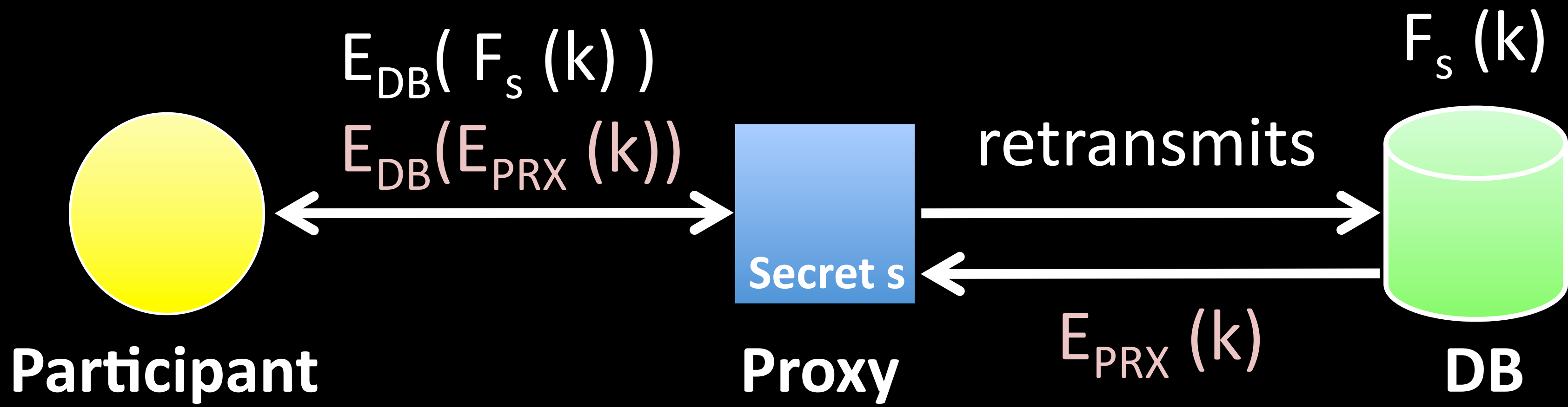
Privacy Properties



- Keyword privacy: Nothing learned about unreleased keys
- Participant privacy: Key ←→ Participant not learned

- Any coalition of ~~HBC participants~~ malicious participants
- HBC coalition of proxy and participants
- ~~HBC database~~ HBC coalition of DB and participants

More Robust PDA Protocol



- ORPF → Encrypted OPRF Protocol
- Ciphertext re-randomization by proxy
- Proof by participant that submitted k's match

- Any coalition of ~~HBC participants~~ malicious participants
- HBC coalition of proxy and participants
- ~~HBC database~~ HBC coalition of DB and participants

Encrypted-OPRF protocol

- Problem: in basic OPRF protocol, participant learns $F_s(k)$
- Encrypted-OPRF protocol:
  - Client learns blinded $F_s(k)$
  - Client encrypts to DB
  - Proxy can unblind $F_s(k)$ “under the encryption”

$$\left(\mathrm{Enc}\left(\left(F_s(k)\right)^{\cancel{r}}\right)\right)^{\cancel{r^{-1}}}$$

El Gamal $\qquad$ $g^{(\pi_{k_i=1} s_i)}$ mod p

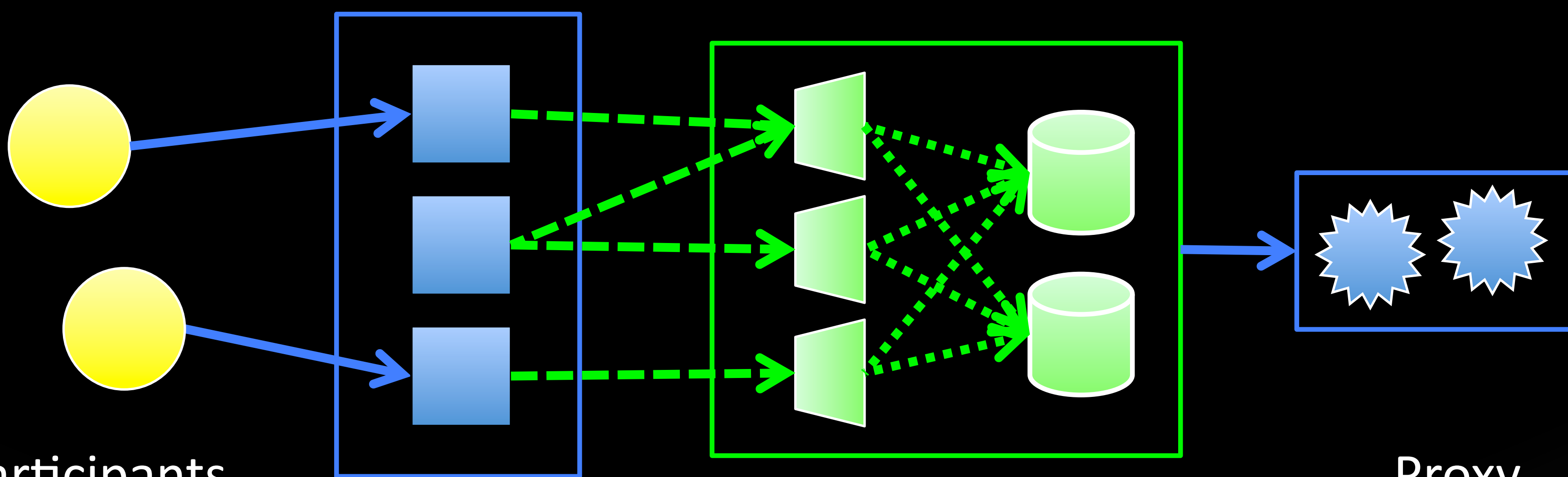
Encrypted-OPRF protocol

- Problem: in basic OPRF protocol, participant learns $F_s(k)$
- Encrypted-OPRF protocol
  - Client learns blinded $F_s(k)$
  - Client encrypts to DB
  - Proxy can unblind $F_s(k)$ "under the encryption"

$$\left(\mathrm{Enc}\left(\left(F_s(k)\right)^{\cancel{r}}\right)\right)^{\cancel{r^{-1}}}$$

- OPRF runs OT protocol for each bit of input k
- OT protocols expensive, so use batch OT protocol [Ishai et al]

Scalable Protocol Architecture



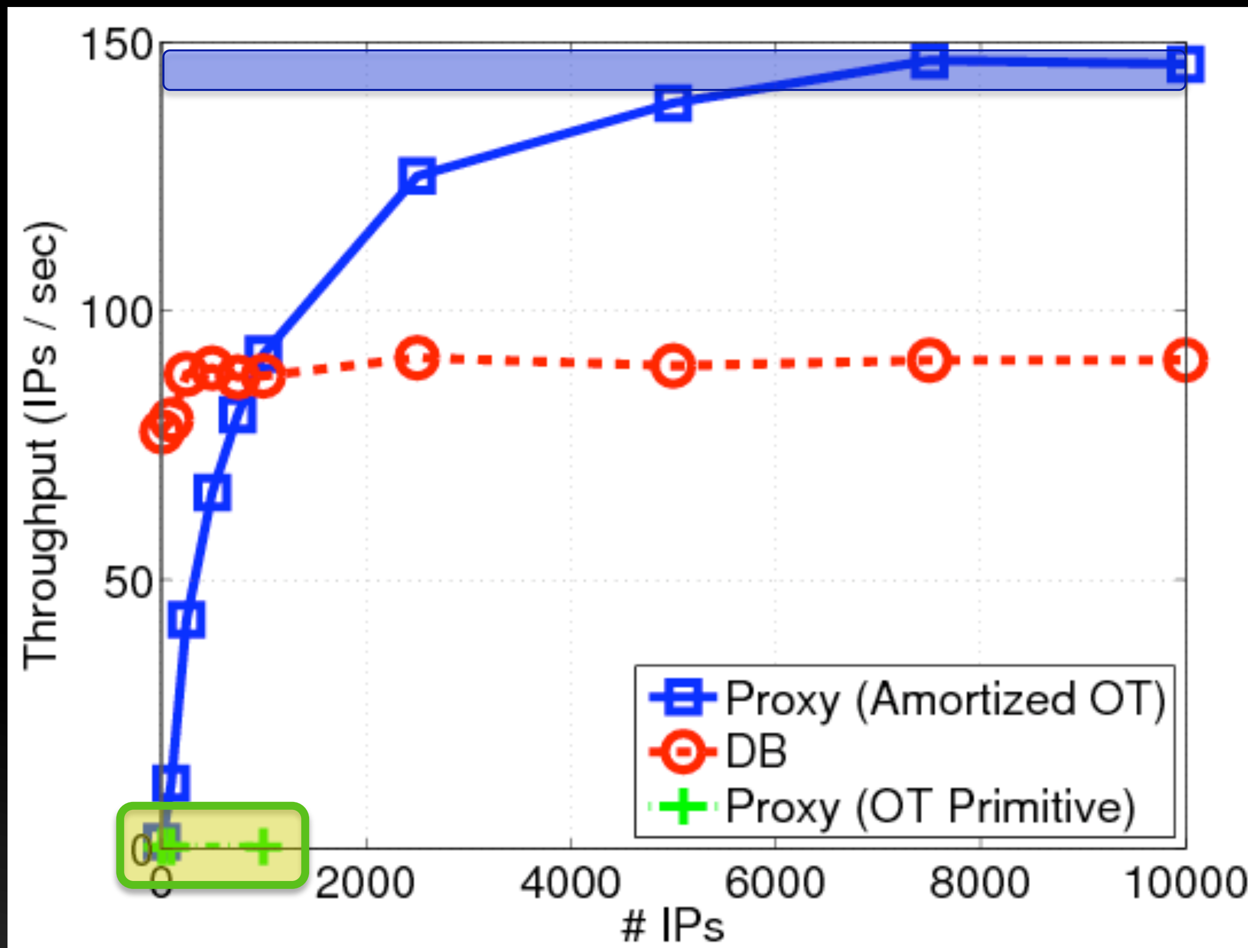
Participants	Client-Facing Proxies	Front-End DB Tier	Back-End DB Storage	Proxy Decryption Oracles
	Share secret *s*	Share DB key	Partition F_s keyspace	Share PRX key

Evaluation

- Scalable architecture implemented
  - Basic CR-PDA / PDA protocol
    + and encrypted-OPRF protocol w/ Batch OT
  - ~5000 lines of threaded C++, GnuPG for crypto
- Testbed of 2 GHz Linux machines

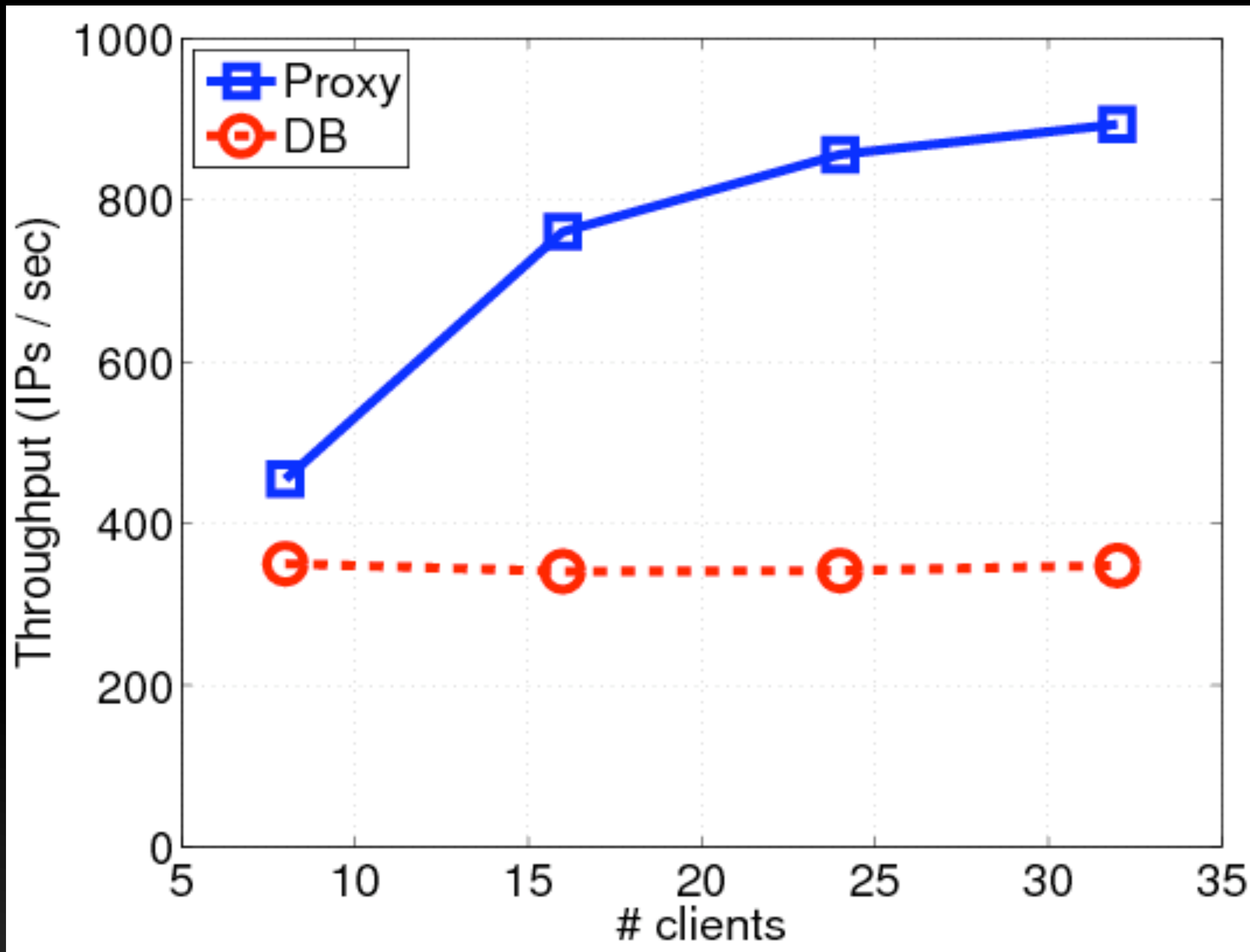
Algorithm	Parameter	Value
RSA / ElGamal	key size	1024 bits
Oblivious Transfer	k	80
AES	key size	256 bits

Throughput vs. participant batch size



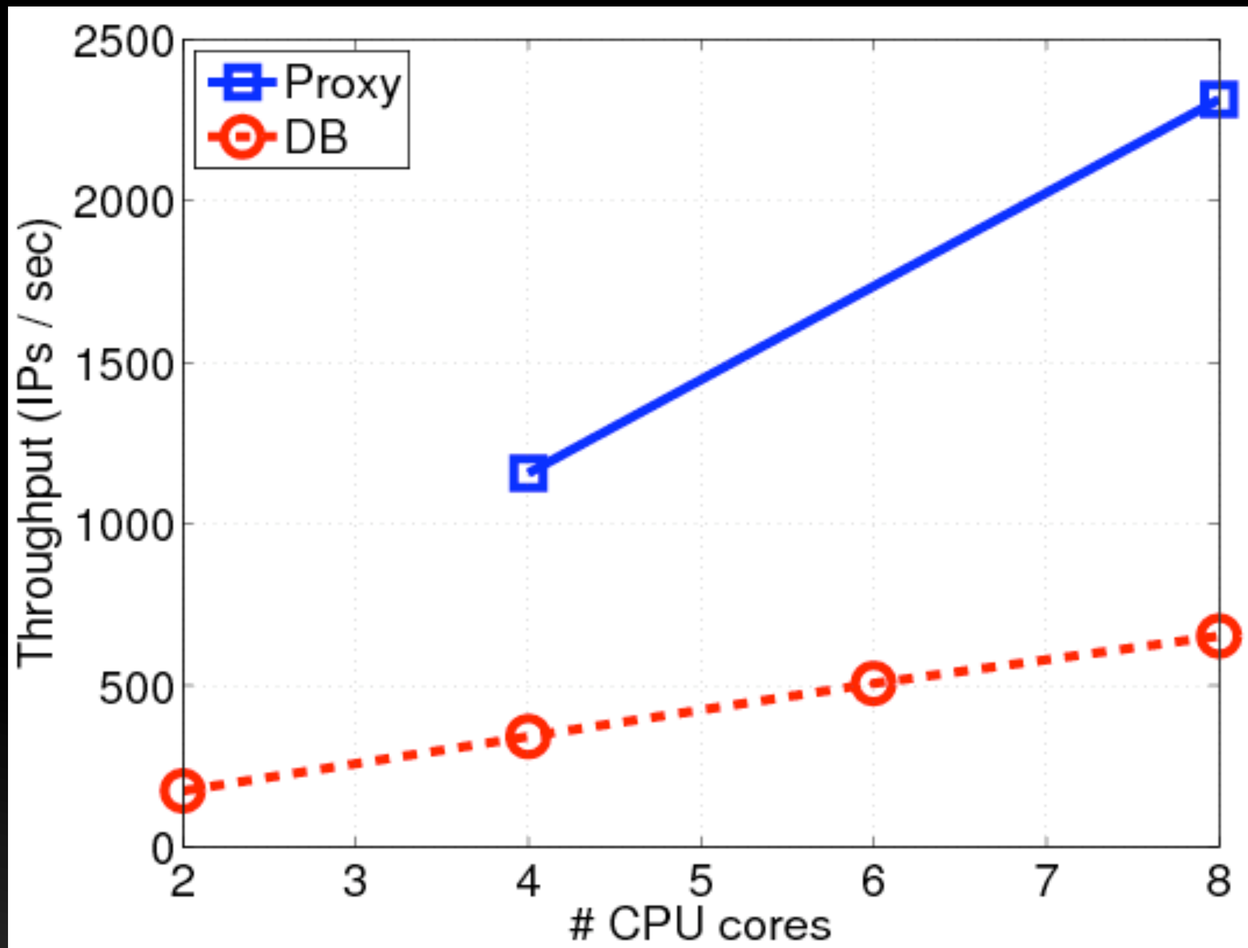
Single CPU core for DB and proxy each

Maximum throughput per server



Four CPU cores for DB and proxy (each)

Throughput scalability



Number CPU cores per DB and proxy (each)

Summary

- Privacy-Preserving Data Aggregation protects:
  - Participants: Do not reveal who submitted what
  - Keywords: Only reveal values / released keys
- Novel composition of crypto primitives
  - Based on assumption that 2+ known parties don't collude
- Efficient implementation of architecture
  - Scales linearly with computing resources
  - Ex: Millions of suspected IPs in hours
- Of independent interest…
  - Introduced encrypted OPRF protocol
  - First implementation/validation of Batch OT protocol