DISE: DISTRIBUTED SYMMETRIC-KEY ENCRYTION

VISA Research

Shashank Agrawal
Payman Mohassel
Pratyay Mukherjee
Peter Rindal

Threshold Cryto

- Has focused on public-key crypto
- Symmetric-key encryption got less attention
  - Symmetric keys don’t stay around long
- Secure communication over internet (TLS)
  - Signing keys are long-term
  - Encryption keys change with every session

Symmetric-key Encryption (SKE)

- Encrypt data at rest
  - AWS, MS Azure, Google Cloud provide client-side, server-side, disk encryption
  - Keys managed by cloud service or client
- Authentication on web, enterprises, ...
  - JSON web tokens, TGT in Kerberos, etc.
- Securing PIN in credit/debit transactions

Threshold SKE

- Threshold PRFs [MS95, NPR99, Nie02, Dod03, DY05, DYY06, BLMR13]

- MPC: Evaluate AES-GCM [DK10, GRRSS16, RSS17]
  - Good: Backward-compability, standard schemes
  - Bad:
    - Communication complexity linear in circuit size, number of parties
    - All parties interact with each other

Build a threshold SKE that works well in practice:

- Fast encryption/decryption
- Requires minimal interactivity
- Provides strong security guarantees

Our Contributions

- Formally study threshold SKE
  - Message privacy & ciphertext integrity in the distributed setting
- Simple and light-weight protocols
  - Initiator sends one message, gets one message (challenge-response style)
  - Support arbitrary threshold t
    - Contact t-1 other parties
    - Resilient to t-1 corruption
- Implement & evaluate
  - A million enc/dec per second, sub millisecond latency with upto 18 parties

Outline

- Security properties
- DiSE: main protocol
- Implementation
- Future work

Threshold SKE

Notation & Model

- n – total number of parties
- Initiator: Party who initiates an enc/dec session
- t – threshold

- Attack model
  - Corrupt t-1 parties maliciously
  - Static model

- Communication model: Point-to-point secure channels

Traditional vs Modern

- Inspired by traditional game-based notions [BN00, KY01, RS06]
- More advanced notions studied for non-threshold [Rog02, RS06, FFL12, PW12 Rog13, GL15, HRRV15, HKR15, BT16, BHT18]
- Extending traditional notions to threshold already non-trivial

Protocols

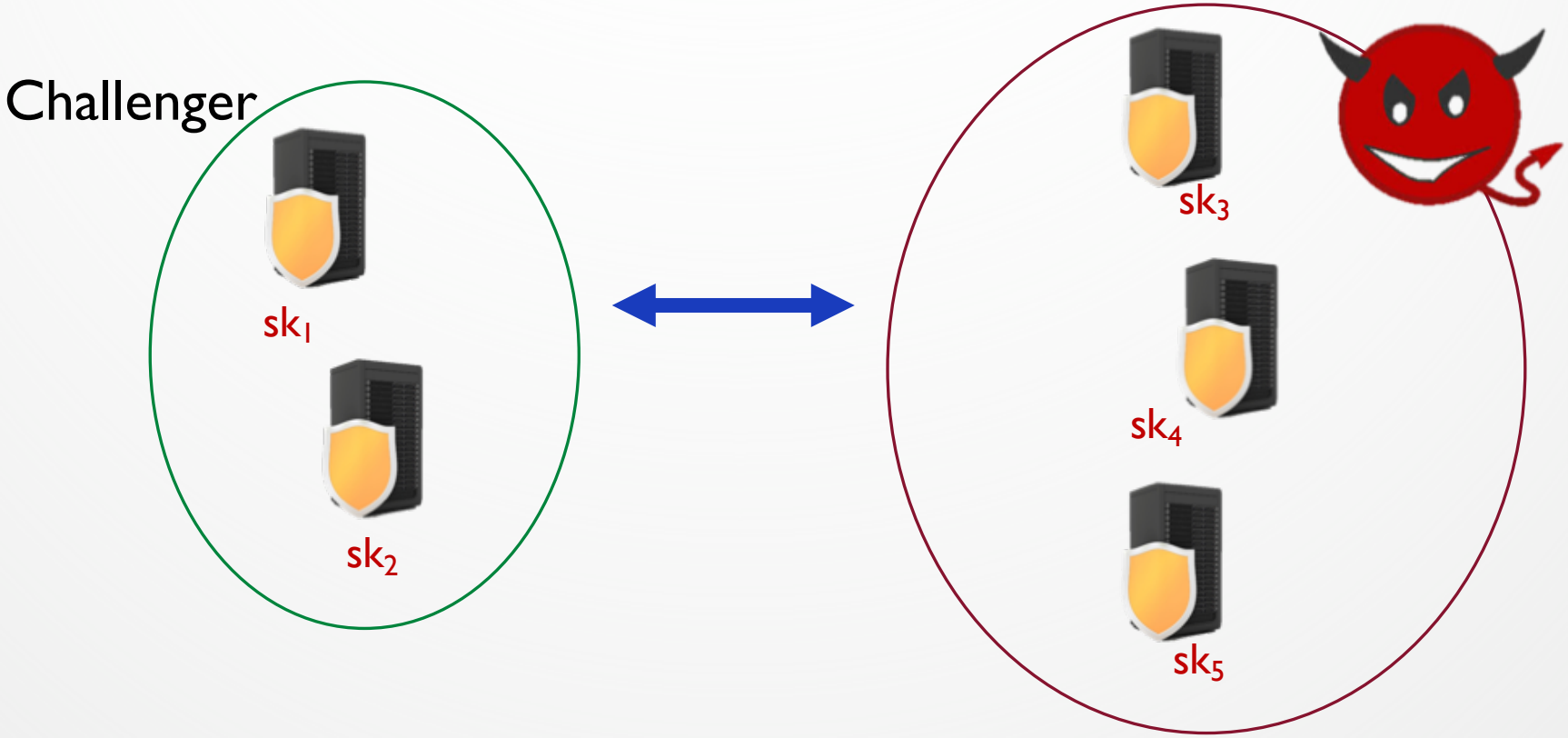
- **Setup** (n, t) → (sk_1, sk_2, ..., sk_n), pp
- **DistEnc** (j, msg, S) → ctxt
  - Parties involved don't learn ciphertext
- **DistDec** (j, ctxt, S) → msg
  - Parties involved don't learn message
- Consistency (all parties honest):
  - DistEnc (j, msg, S) → ctxt
  - DistDec (j*, ctxt, S*) → msg

Correctness

- DistEnc session fails even if initiated by honest party
- DistEnc succeeds but DistDec fails
- **Basic**: *if* DistEnc (msg) → ctxt ≠ ⊥, *then* DistDec (ctxt) → msg or ⊥
- **Strong**: *if* DistEnc (msg) → ctxt ≠ ⊥, *then* DistDec (ctxt) → msg if parties honest

Security Games

- Message privacy & ciphertext integrity
- Games between Challenger *Chal* and Adversary *Adv*



Message Privacy

- Ciphertexts do not reveal message
- Non-threshold: $\mathrm{Enc}(m_0) \approx \mathrm{Enc}(m_1)$
- Adv is allowed to:
  - *Encryption*: Initiated by corrupt/honest party
  - *Decryption*: Initiated by honest party
- *Challenge*: Adv outputs (j, m_0, m_1, S)

Ciphertext Integrity (Authenticity)

- New valid ciphertexts cannot be generated
- Non-threshold: Can keep track of ciphertexts
- **C** – set of corrupt parties
- **g** = t - |C|
- **cnt** – count #messages Adv sends to honest parties
- **L** – list of ciphertexts

Ciphertext Integrity (Authenticity)

- Variables: C, g, cnt, L
- Adv allowed to:
  - (**Encryption**, j, msg, S)
    - j is corrupt: increment cnt by #honest parties in S
    - j is honest: add ctxt to L
  - (**Decryption**, j, ctxt, S)
    - j is corrupt: increment cnt by #honest parties in S
  - (**Targeted Decryption**, j, k, S) with j honest
- Maximum ciphertexts: **cnt / g** (rounded down)

Counter incremented

Decryption!!

Ciphertext Integrity (Authenticity)

- *Forgery*: Adv outputs $(j_1, S_1, ctxt_1), (j_2, S_2, ctxt_2), \ldots, (j_k, S_k, ctxt_k)$
- Adv wins if:
  - $k > cnt / g$
  - Dec sessions output valid messages
- **Basic**: Dec sessions are honest
- **Strong**: Corrupt parties can misbehave

Summary

- Correctness: Basic & Strong
- Message privacy
- Ciphertext integrity: Basic & Strong

DiSE: Threshold SKE Scheme

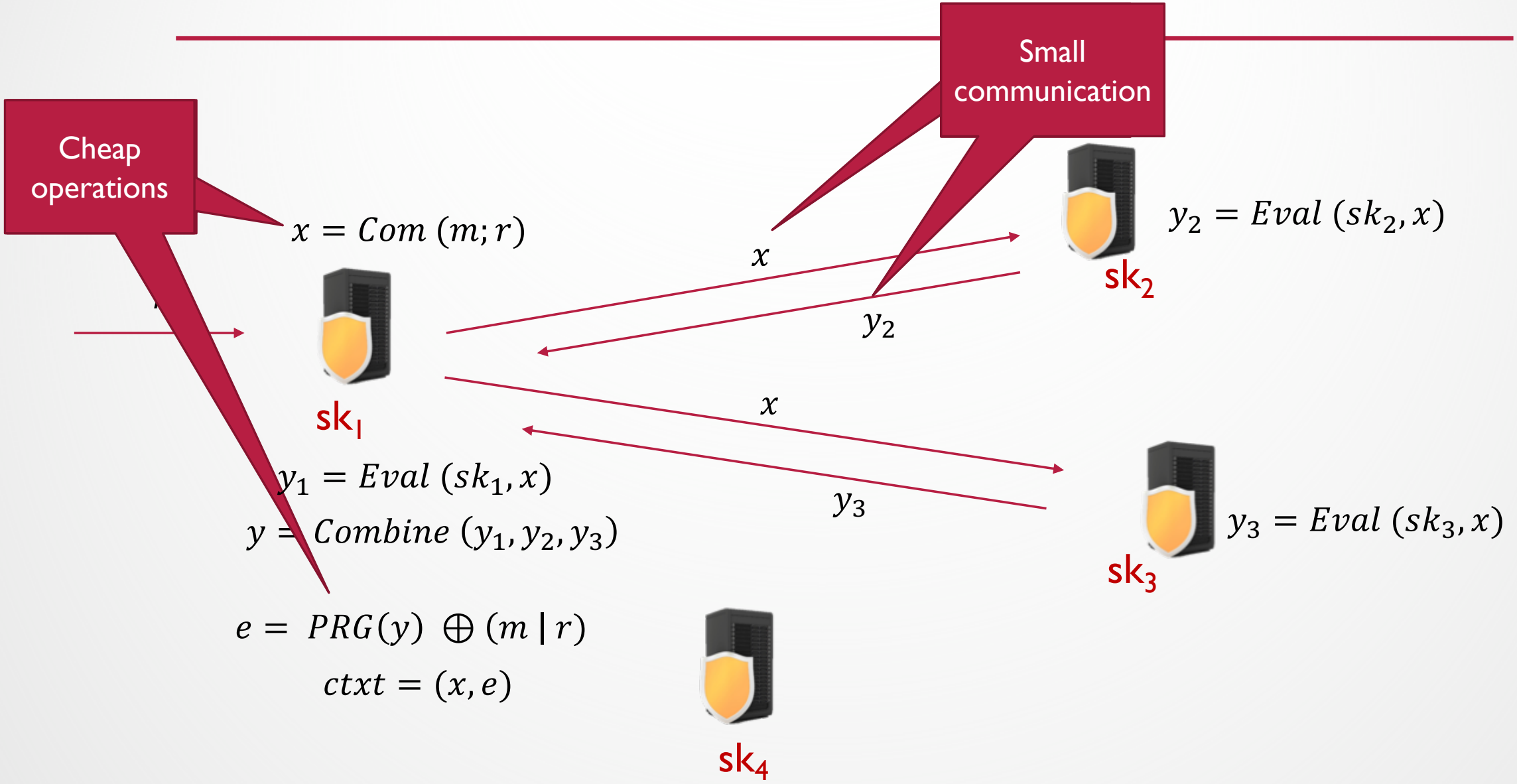
Distributed PRF (DPRF)

- Introduced by Naor et al. [NPR99]
- Several constructions/variations [Nie02, Dod03, DY05, DYY06, BLMR13]
- **Setup** $(n, t) \rightarrow (sk_1, sk_2, \ldots, sk_n)$
- **Eval** $(sk_j, x) \rightarrow y_j$
- **Combine** $(y_1, y_2, \ldots) \rightarrow y$
- *Consistency*: Same output irrespective of the set
- *Pseudorandomness*: Final output should be pseudorandom
- *Correctness*: Final output either correct or $\perp$

Secure

Strongly secure

DiSE


Small communication
Cheap operations
$x = Com\ (m; r)$
x
y_2
$y_2 = Eval\ (sk_2, x)$
sk_2
sk_1
x
y_3
$y_1 = Eval\ (sk_1, x)$
$y = Combine\ (y_1, y_2, y_3)$
$y_3 = Eval\ (sk_3, x)$
sk_3
$e = PRG(y) \oplus (m \mid r)$
$ctxt = (x, e)$
sk_4

Security

- If DPRF is (strongly) secure, then DiSE satisfies
  - (strong) correctness
  - message-privacy
  - (strong) ciphertext-integrity

DPRF instantiations [MS95, NPR99]

- DDH assumption (ROM)
  - Setup $(n, t) \rightarrow (sk_1, sk_2, \ldots, sk_n)$
  - Eval $(sk_j, x) \rightarrow \text{Hash}(x)^{sk_j}$
  - DPRF $(x) = \text{Hash}(x)^{sk}$

- Any PRF like AES
  - Setup → Exponential number of keys
  - DPRF $(x) = \text{PRF}_{k1}(x) \oplus \text{PRF}_{k2}(x) \oplus \text{PRF}_{k3}(x) \oplus \ldots$

Compare

	DDH	PRF
Choice of n, t	Arbitrary	${}^{n}C_{t}$ should be small
Type of operations	Expensive public-key	Cheap symmetric-key
Strong security	Easy	Difficult
Change of n, t	Master key unaffected	Master key affected

Implementation & Evaluation

Implementation

- Three instantiations: PRF, DDH, DDH-NIZK
- Tested on many values of n, but $n = 18$ here
- Tested on both LAN, WAN, but only LAN here
- Choices:
  - Hash function: Blake2
  - PRF/PRG: AES
  - ECC curve: p256k1
- Benchmarking on a single server with two 18-core Intel Xeon CPUs @2.3 GHz, 256GB RAM
- LAN: 10 Gbps bandwidth, 0.1 ms latency

Performance

Throughput (Enc/sec)

Threshold (T)	PRF		DDH		DDH-NIZK	
	Enc/sec	Mbps	Enc/sec	Mbps	Enc/sec	Mbps
2	**1,037,703**	253	553	0.14	226	0.28
6	**45,434**	55	297	0.77	64	0.40
9	**10,194**	20	231	0.45	42	0.50
16	**524,109**	1919	135	0.49	23	0.43

Latency (ms/Enc)

Threshold (T)	PRF	DDH	DDH-NIZK
2	0.1	4.6	9.6
6	0.6	5.4	21.5
9	1.1	8.0	31.3
16	2.2	12.6	55.2

Conclusion & Future Directions

Conclusion

- SKE widely used, secret keys need protection (MPC expensive)
- Formalization of threshold SKE
- New very efficient scheme
- Promising performance

Future Directions

- DiSE lacks concrete security treatment
- Ciphertext integrity definition counts decryption towards encryption
- **ParaDiSE**: Addresses these issues – and more

THANK YOU!

QUESTIONS...