Message-locked Encryption and Deduplication Security

Thomas Ristenpart
University of Wisconsin—Madison

Joint work with:
Mihir Bellare, Sriram Keelveedhi
UC San Diego

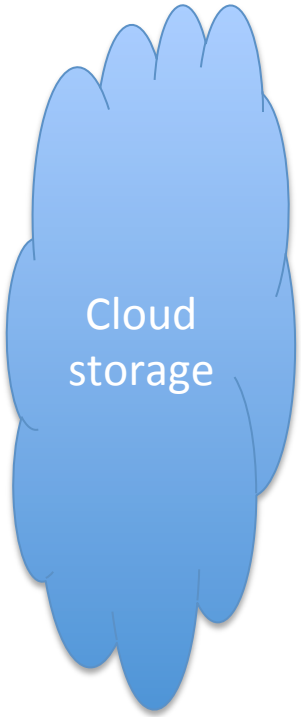
A motivating example

User A

"myFile", 010101…

User B

"theFile", 010101…



User	Filename	Contents
A	myFile	010101…
B	theFile	010101…

Dropbox saves on storage by storing (logically) only one copy of file contents

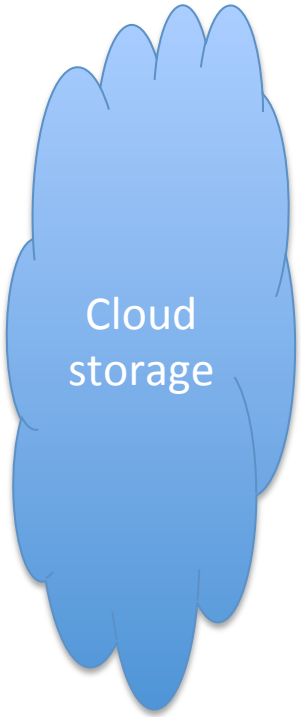
A motivating example

User A

"myFile", 010101…

User B

"theFile", 010101…



User	Filename	Contents
A	myFile	010101…
B	theFile	

Dropbox saves on storage by storing (logically) only one copy of file contents

Deduplication
Find duplicate files and remove redundant copies

[Meyer, Bolosky 2011] ~50% space saved

NetApp

bitcasa
INFINITE STORAGE

Dedup doesn't work with conventional client-side encryption

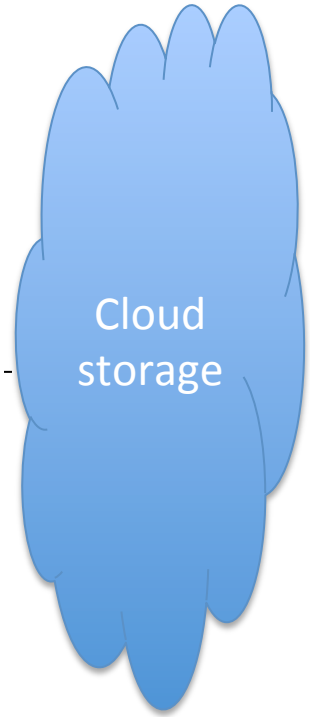
Dropbox

"myFile", M

User A

"theFile", M

User B



Only stores one copy of M

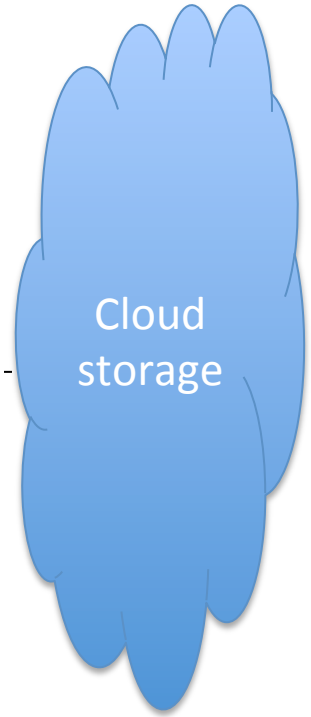
Dedup doesn't work with conventional client-side encryption

Dropbox

$C_A \xleftarrow{\$} E(K_A, M)$

User A

"myFile", C_A



Must store both C_A and C_B

$C_B \xleftarrow{\$} E(K_B, M)$

User B

"theFile", C_B

E is a conventional, possibly randomized encryption algorithm

C_A and C_B are indistinguishable from independent, random bit strings

A big, expensive problem

Dropbox does server-side encryption ***with keys they retain***

TUESDAY, APRIL 12, 2011

How Dropbox sacrifices user privacy for cost savings

Christopher Soghoian,
http://paranoia.dubfire.net/2011/04/how-dropbox-sacrifices-user-privacy-for.html

Companies must encrypt data before storing it or backing it up, preventing deduplication

Can we build secure, ***client-side*** encryption mechanisms that support deduplication?

What's been done here?

(Distributed) storage literature:

[Batten et al. `01]
[Douceur et al. `02]
[Cox et al. `02]
[Cooley et al. `04]
[Killijian et al. `06]
[Wilcox-O'Hearn, Warner `08]
[Storer et al. `08]
… (many more)

Convergent encryption (CE): Encrypt M under H(M)

Systems:

Flud
TahoeFS
Ciphertite
GNUnet

Companies:

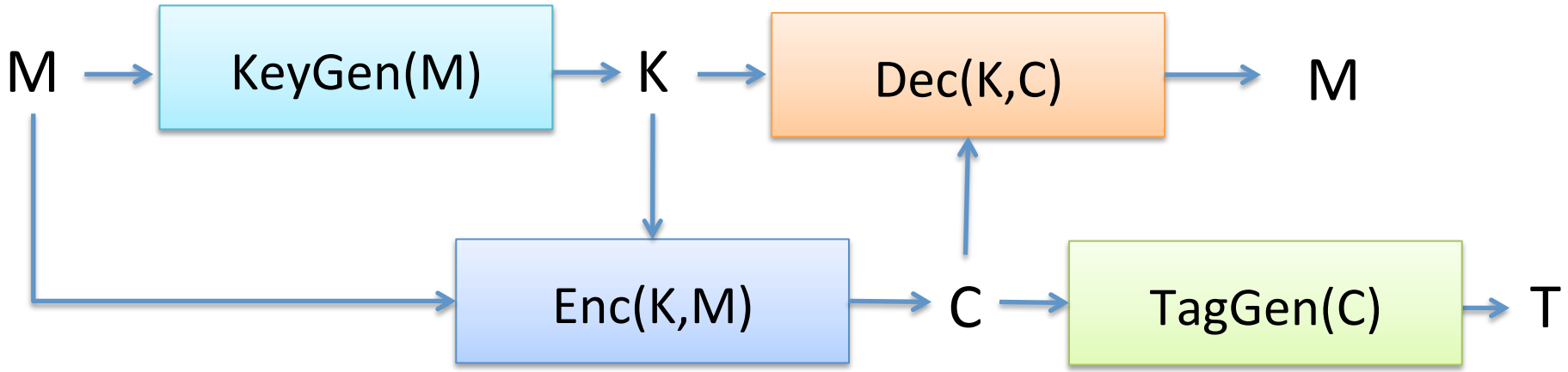
bitcasa

Crypto literature:

(this space intentionally left blank)

The big idea (from CE):
message is "shared secret key material" used to derive keys

We formalize a new cryptographic primitive:
Message-Locked Encryption (MLE)



- Param generates a ***public*** system-wide parameter P given to all algorithms (not shown for brevity)
- Param, Keygen, Encrypt may be randomized
- TagGen, Dec are deterministic

Using MLE with deduped storage

$K_A \xleftarrow{\$}$ KeyGen(M)
$C_A \xleftarrow{\$}$ Enc(K_A,M)

User A

"myFile", C_A →

$K_B \xleftarrow{\$}$ Keygen(M)
$C_B \xleftarrow{\$}$ Enc(K_B,M)

User B

"theFile", C_B →

$M \xleftarrow{\$}$ Dec(K_B,C_A)

User B

Get "theFile" →

← C_A

Storage server:

T ← TagGen(C_A)
T' ← TagGen(C_B)
If T = T' then
    store C_A
else
    store C_A , C_B

K_A , K_B can be encrypted and stored using conventional scheme

Using MLE with deduped storage

Important functionality properties required:

Non-triviality: |K| << |M|

Tag correctness: T = T' for ciphertexts for same message M

Efficient search: *O*(log d) search for duplicate over d ciphertexts

Decryption correctness: any key works for any ciphertext (same M)

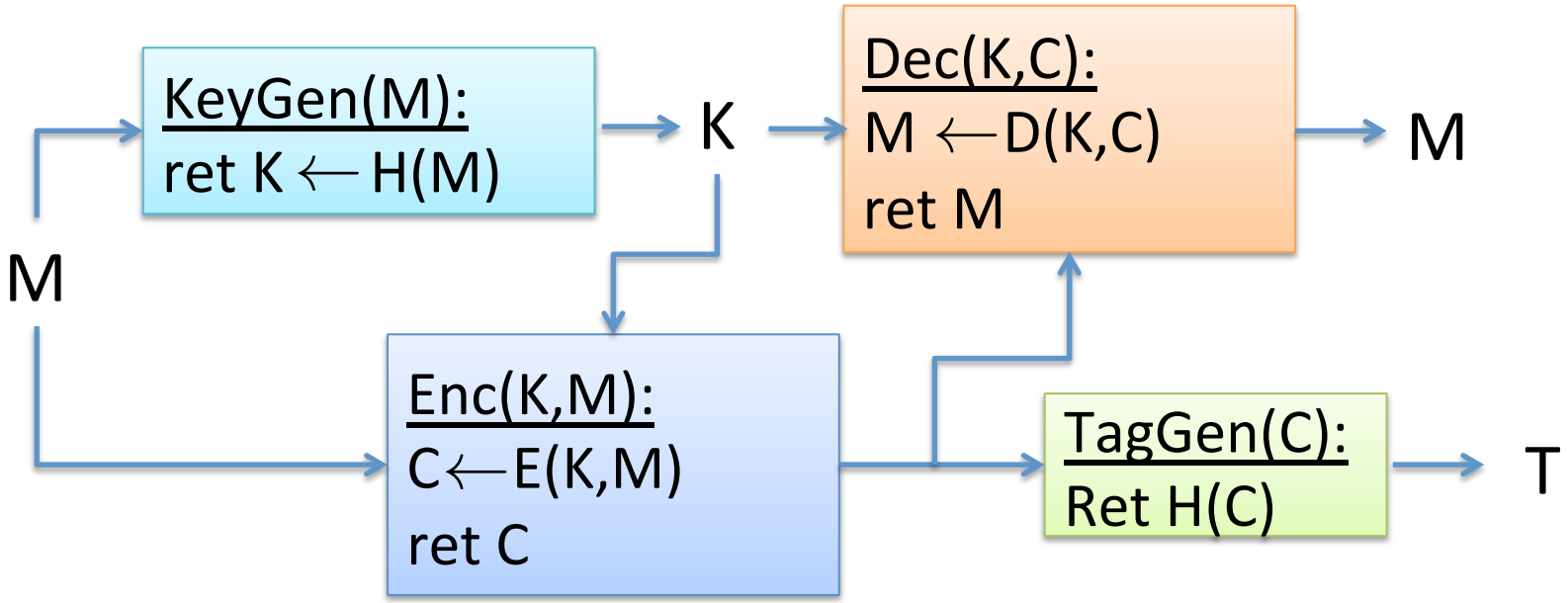
(see paper for formal details)

Convergent encryption

[Douceur et al. 2002]
[Pettitt `96] [Clarke et al. `00]
[Wilcox-O'Hearn `00]

Deterministically encrypt M under cryptographic hash H(M)

CE as an MLE scheme:

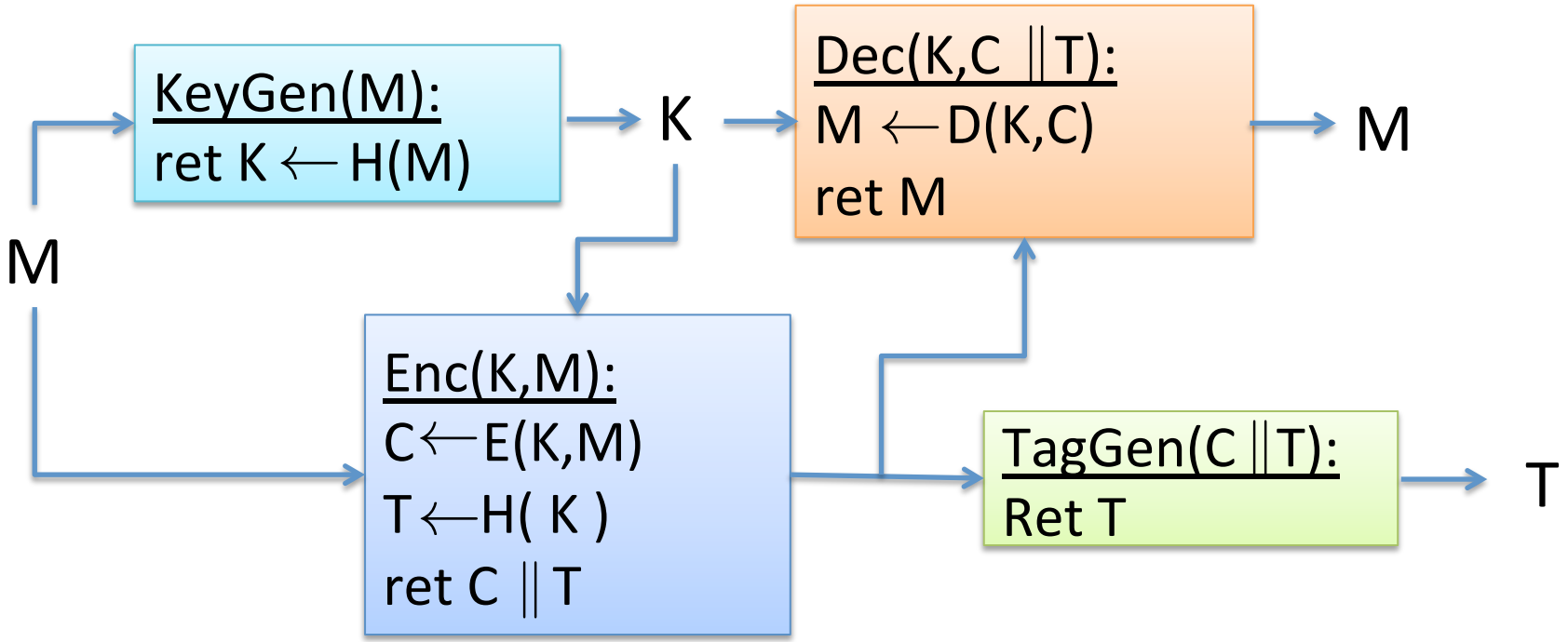


E is deterministic symmetric scheme E (decryption via D)
(e.g., CTR-mode AES with constant IV)

Non-triviality: |K| = 128 bits while M can be arbitrary length

Hash-and-CE (HCE1) scheme

Used in TahoeFS, elsewhere



In paper two new schemes:

- Hash-and-CE 2 with tag check (HCE2)
- Randomized CE (RCE) that achieves single-pass MLE

Why? All three schemes are faster than CE

Message privacy: the bad news

Let $S = \{ M_1 , \ldots , M_m \}$ be known set of possible messages

Let $C \xleftarrow{\$} Enc(KeyGen(M\_i) , M\_i)$ for random i and give adversary C

BruteForce(C):

$T \leftarrow TagGen(C)$

For i = 1 to m do

$K_i \xleftarrow{\$} KeyGen(M_i)$

$C_i \xleftarrow{\$} Enc(K_i , M_i)$

$T_i \leftarrow TagGen(C_i)$

If $T = T_i$ then

Return M_i

Works for any ***tag-correct*** scheme

Runs in time $O(m)$

Observed in [Zooko `08] for CE.

Privacy for MLE schemes only possible for unpredictable messages

Message privacy: the good news

New privacy definitions PRV-CDA PRV$-CDA

Best possible subject to limitation of brute-force attacks

If set of possible messages is too large, no attacker can distinguish between ciphertext and random bit string

Game PRV$-CDA

$b \xleftarrow{\$} \{0,1\}$

$M \xleftarrow{\$} \mathcal{M}$

$K \xleftarrow{\$} \mathrm{KeyGen}(M)$

$C[1] \xleftarrow{\$} \mathrm{Enc}(K, M)$

$C[0] \xleftarrow{\$} \{0,1\}^{|C[1]|}$

$b' \xleftarrow{\$} A(C[b])$

ret $(b = b')$

Weaker PRV-CDA left-or-right indistinguishability notion in paper

Similar to notions for deterministic/searchable PKE [BBO'07,BFOR'08,BFO'08]
hedged PKE [BBNRSS'09]

Analysis of fast MLE schemes

In-use CE and variant HCE

2 new schemes HCE2 and RCE

* Using AES-NI with AES256 for hashing and CTR mode on Intel Core i7-970. CTR mode by itself 1.2 cpb. |M| = 4 KB

Scheme	KeyGen + Enc + TagGen time*	PRV-CDA	PRV$-CDA
CE	11.8 cpb	Yes	Yes
HCE	6.6 cpb	Yes	Yes
HCE2	6.6 cpb	Yes	Yes
RCE	6.5 cpb	Yes	Yes

We provide proofs of security assuming hash is random oracle (RO)

Duplicate faking attacks and MLE integrity

$C_A \xleftarrow{\$}$ FakeCtxt(M,M')

User A

"myFile", C_A

$K_B \xleftarrow{\$}$ Keygen(M)

$C_B \xleftarrow{\$}$ Enc(K_B,M)

User B

"theFile", C_B

Storage server:

T ← TagGen($C2_A$)

T' ← TagGen($C2_B$)

If T = T' then

store C_A

else

store C_A , C_B

M' $\xleftarrow{\$}$ Dec(K_B,C_A)

User B

Get "theFile"

$C1_A$

Service stores just C_A

1) Adversary knows M to be uploaded by B

2) Make fake C_A s.t. T = T', but decrypts to M' != M

3) User B gets back corrupted file later!

Duplicate faking attacks and MLE integrity

$C_A \xleftarrow{\$}$ FakeCtxt(M,M')

User A

"myFile", C_A

Storage server:

T ← TagGen($C2_A$)
T' ← TagGen($C2_B$)
If T = T' then
    store C_A
else
    store C_A , C_B

Service stores just C_A

Attack against HCE:

<u>Encrypt(K,M):</u>
C ← E(K,M)
T ← H(K)
ret C ‖ T

<u>FakeCtxt(M,M'):</u>
K ← H(M)
C_A ← E(K,M')
T ← H(K)
ret C ‖ T

Attack in [Storer et al. `08], but vulnerabilities not realized

Weaker attack would just make decryption fail

Analysis of fast MLE schemes

In-use CE and variant HCE

2 new schemes HCE2 and RCE

* Using AES-NI with AES256 for hashing and CTR mode on Intel Core i7-970. CTR mode by itself 1.2 cpb. |M| = 4 KB

Scheme	KeyGen + Enc + TagGen time*	PRV-CDA	PRV$-CDA
CE	11.8 cpb	Yes	Yes
HCE	6.6 cpb	Yes	Yes
HCE2	6.6 cpb	Yes	Yes
RCE	6.5 cpb	Yes	Yes

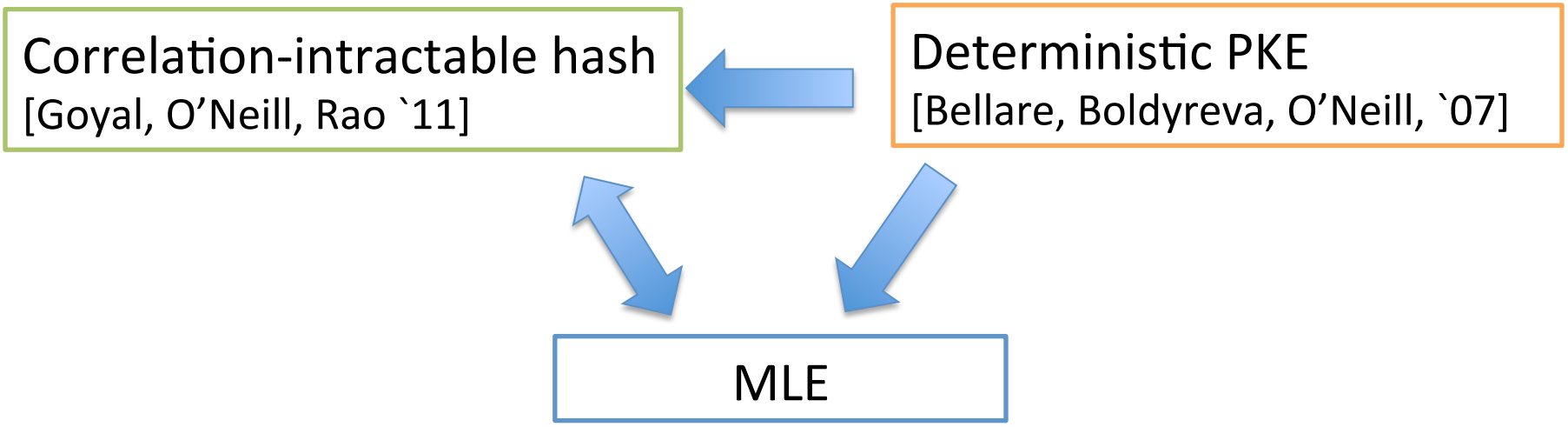
Tag consistency (TC) – prevent replacement, but not failures

Strong tag consistency (STC) – prevent both

HCE2, RCE use new technique, guarded decryption, to get TC

Theory: standard model MLE?

1) Constructions assuming existence of other primitives



2) Constructions for special message spaces

Sample-Extract-Encrypt
Message spaces with high entropy density

A quick digression

Crossover applied-theory research flavors:

New theoretically-sound practical crypto

SA-MLE & DupLESS

Refining old models due to new attacks

MLE

Integrating known theory into applications

Figuring out what people have been trying to do

Formal security analyses of deployed systems

What I talked about last year

A quick digression

Crypto literature can lag behind innovations happening in other academic communities or (gasp) industry

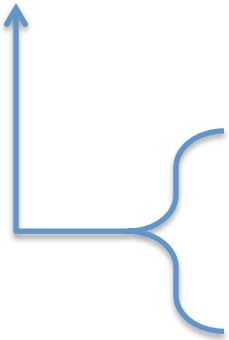
A totally biased corpus of relatively recent examples:

Topic / 1st paper	Who got there first	Lag time	Results of initial crypto work
Keywrap [RS `06]	NIST, IEEE S/MIME	~9 years	Standardized scheme & security defs used more broadly
Deterministic PKE [BBO `07]	Database community, industry (?)	~5 years	New schemes, new definitions used widely, open problems and many follow-up papers
Format-preserving encryption [BRRS `09]	NBS, Security community, Industry	~ 28 years	Widely-deployed standard, swathe of theory papers, applications in stegonagraphy
HSM-friendly AE [BFSW `12]	Industry	~ 1 year	Analyze scheme made public at last year's meeting
Message-locked encryption [BKR `12]	Distributed systems, Industry	~ 11 years	Formalizations, new schemes, new settings, more?

MLE leaks nothing about messages…

if messages are unpredictable

Attacker recovers M given Enc(Keygen(M),M)
in time *O*(m) when m is # of possible messages



1) In some cases m = 2
2) Hard for defenders to determine m!

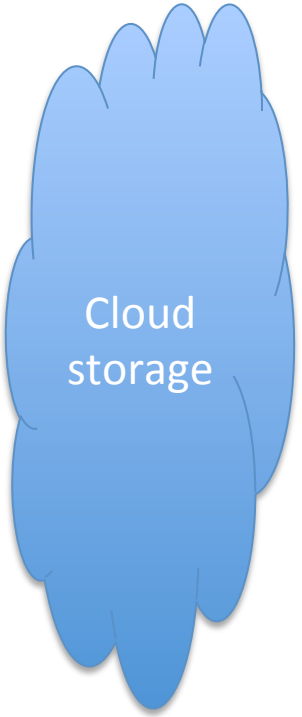
One idea: users share secret key K

[Warner, Perttula / TahoeFS `08]

$K_A \xleftarrow{\$} \text{Keygen}(K \| M)$
$C_B \xleftarrow{\$} \text{Enc}(K_A, M)$

User A

"myFile", C_A



$K_B \xleftarrow{\$} \text{Keygen}(K \| M)$
$C_B \xleftarrow{\$} \text{Enc}(K_B, M)$

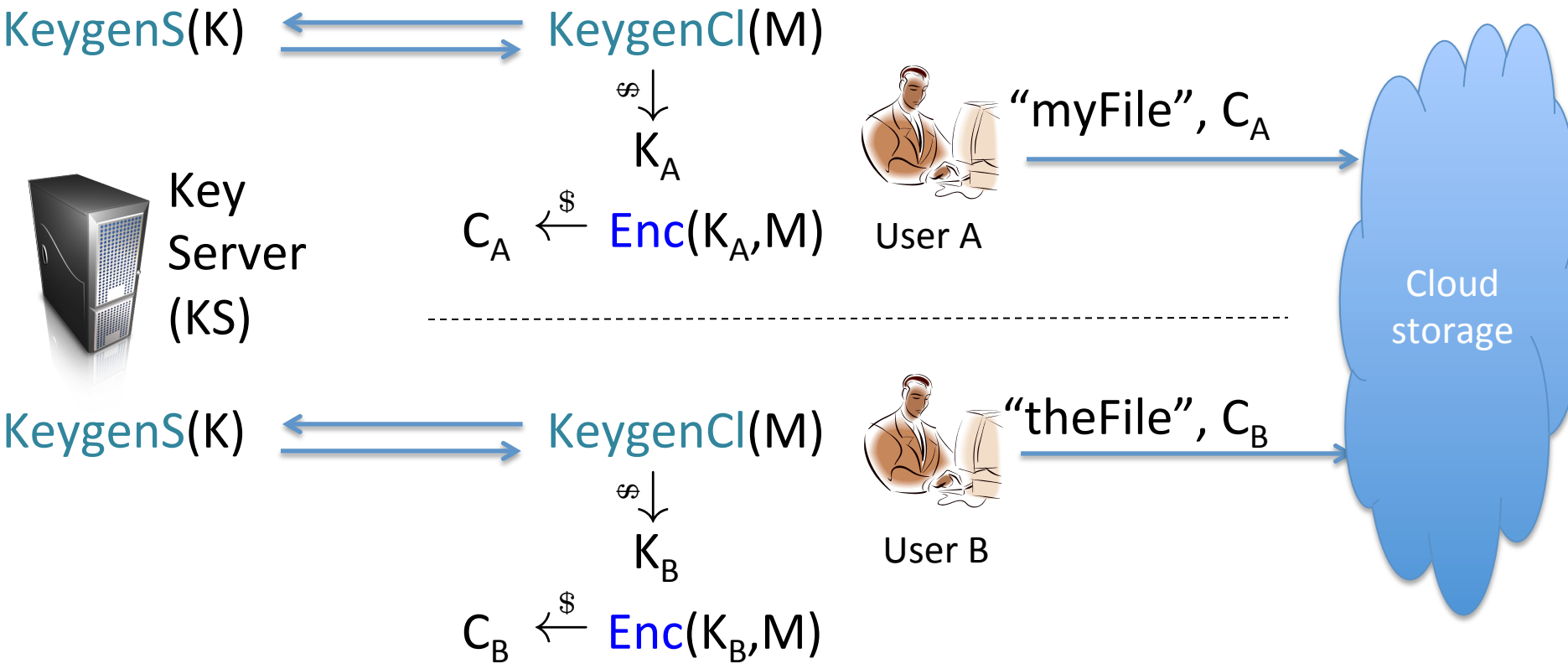
User B

"theFile", C_B

Now brute-force attacks (provably) disappear…

…until K is exposed

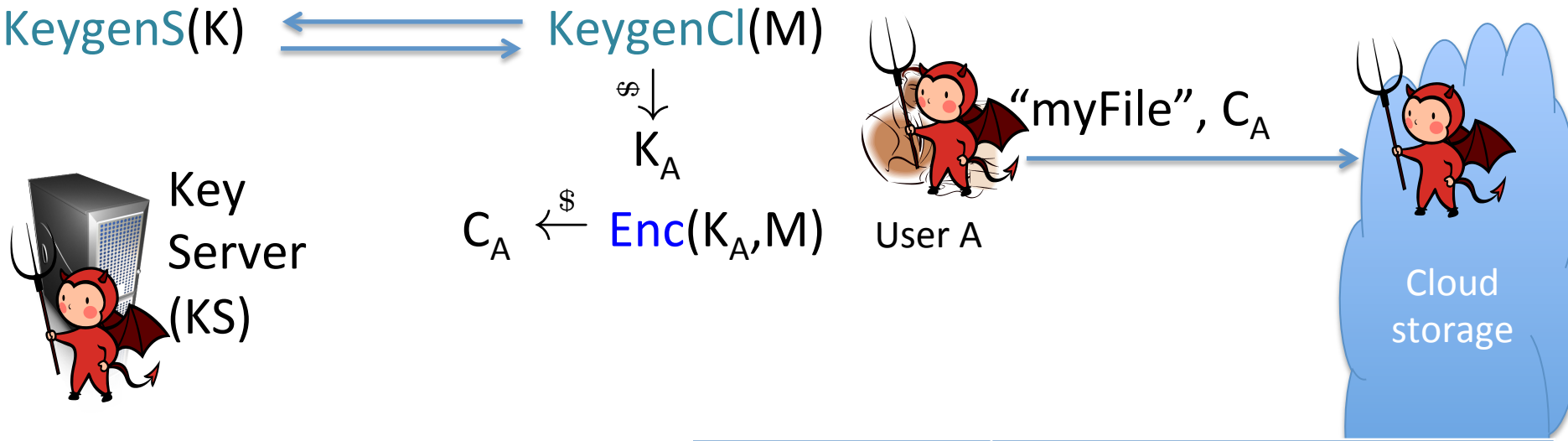
We introduce server-aided MLE (SA-MLE)



Ciphertexts from users of same KS can be deduped

KS can be private (authenticate clients) or possibly public

We introduce server-aided MLE (SA-MLE)



We give concrete scheme using Oblivious PRF protocol (RSA-based blind sigs [Camenisch et al. `07]) for KS and CE for rest

Adversary has compromised	Best privacy attack w/ private KS
KS	No attack exists
Storage	$O(2^{128})$ computations
User + storage	O(m) ***KS queries***
KS + storage	O(m) computations

We provide formal models and analyses

A new expression:
Is it Real-World-Worthy (RWW)?

Who can run the KS?

Key Server (KS)

Is encryption fast enough? KeygenS(sk) ⇄ KeygenCl(M)

Does it work with existing storage systems?

Dropbox

bitcasa INFINITE STORAGE

SPIDEROAK

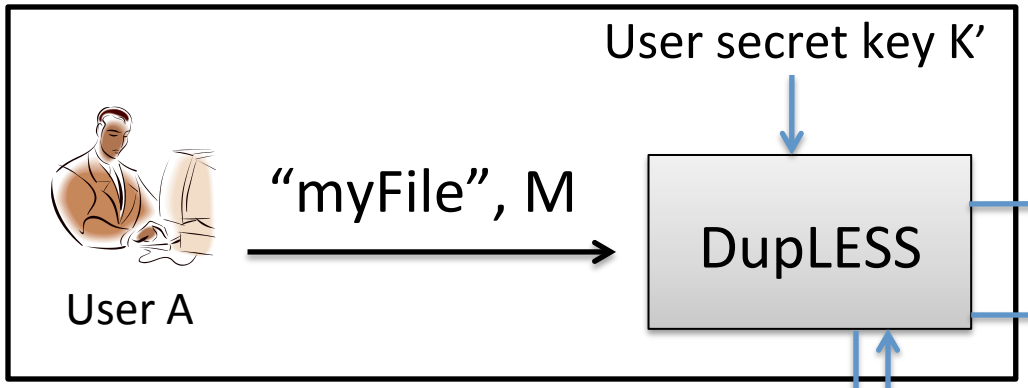
NetApp

mozy

EMC²

DupLESS (DuplicateLess Encryption for Simple Storage)

API-compatible wrappers for storage service plus a KS protocol



Key Server (KS)

C_A can be deduped

DLput(K',F,M)

$K_1, K_2, K_3 \leftarrow KDF(K')$

$K_A \xleftarrow{\$} KeygenCl(M)$

$C_A \xleftarrow{\$} Enc(K_A, M)$

$C_1 \leftarrow DE(K_1, F)$

$C_2 \xleftarrow{\$} SE(K_2, K_A)$

$\sigma \leftarrow HMAC(K_3, C_1 \| C_2 \| C_A)$

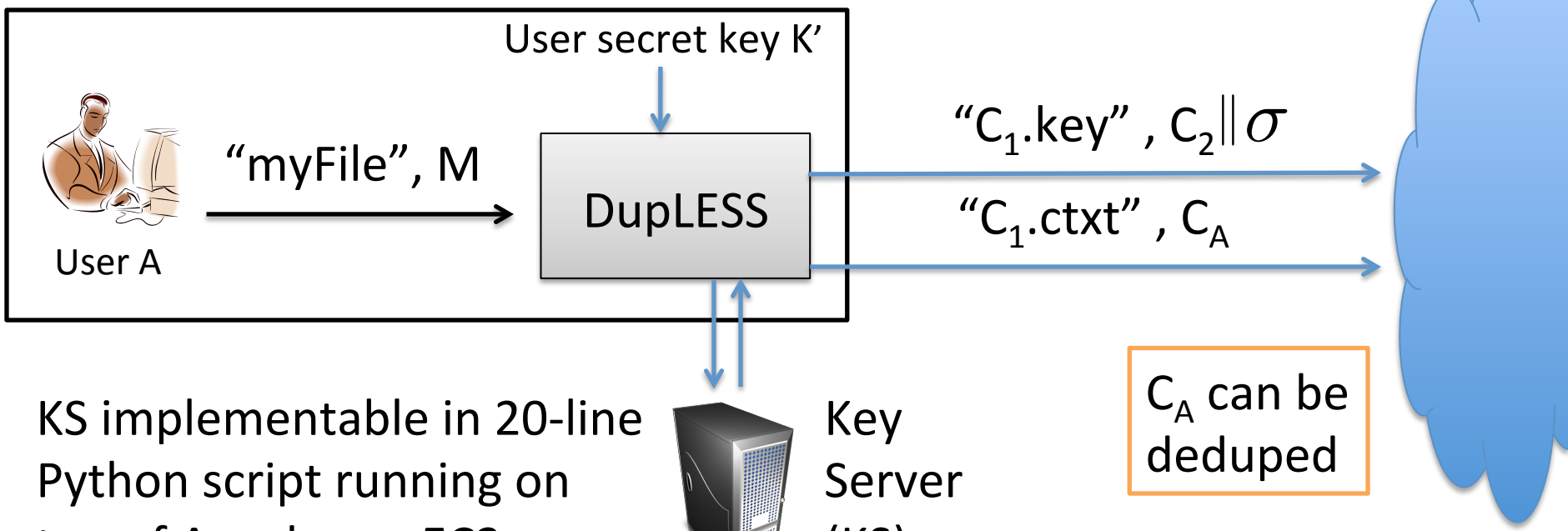
DE is deterministic encryption

SE is randomized CTR mode

Storage overhead: 3L + 120 bytes where L is filename size

DupLESS (DuplicateLess Encryption for Simple Storage)

API-compatible wrappers for storage service plus a KS protocol



KS implementable in 20-line Python script running on top of Apache on EC2.

Key Server (KS)

C_A can be deduped

Simple HTTP(S)-based protocol

KS performs one RSA exponentiation per request

DupLESS (DuplicateLess Encryption for Simple Storage)

Only store requires KS interaction

KS unavailable -> fails safe to conventional encryption

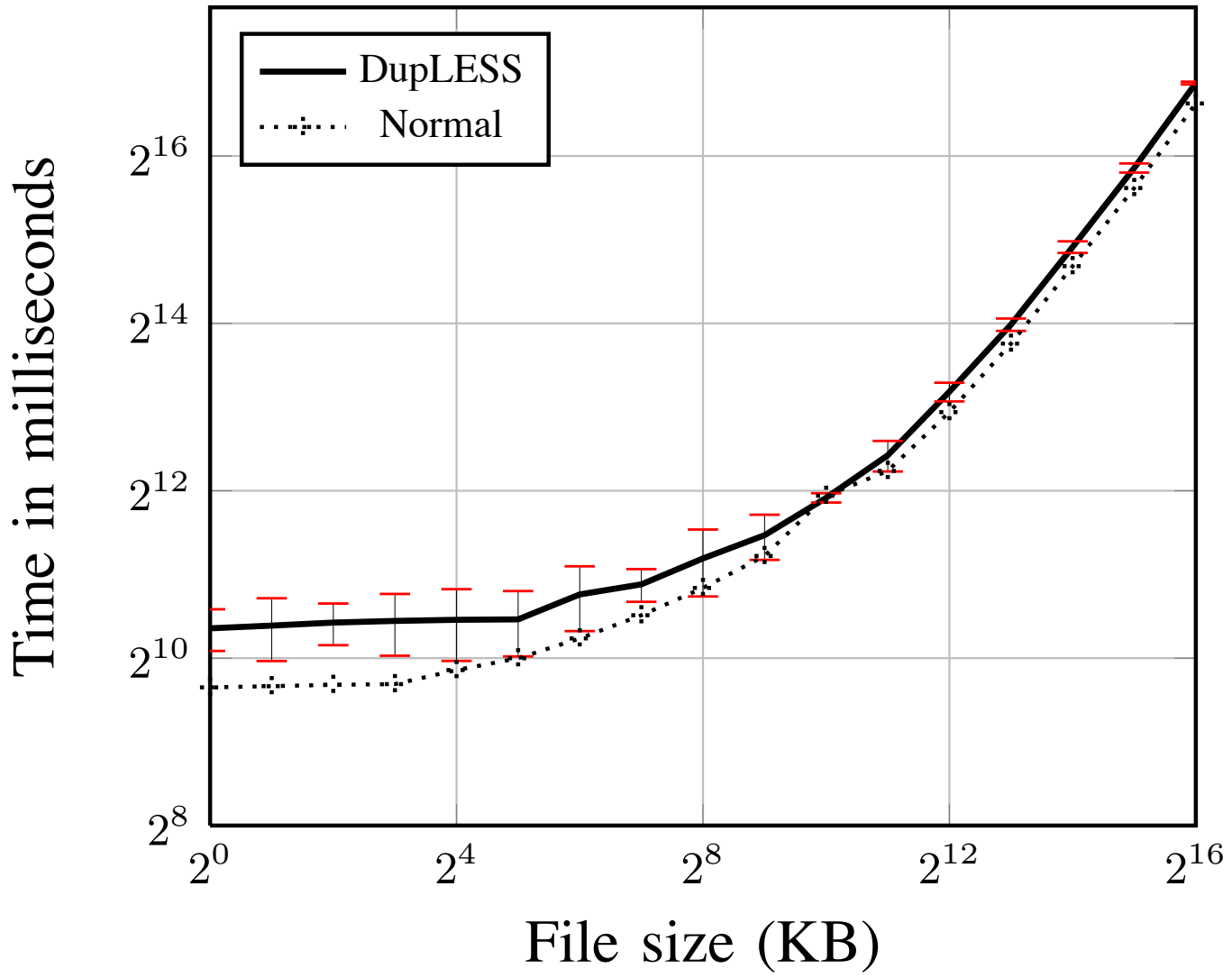
Other API features ***supported/able***

(filename search, listing, sharing, paths, etc.)

Requires ***no changes*** to storage system and no understanding of dedup mechanisms

Optimizations possible (e.g., single storage request)

DupLESS: Performance of put

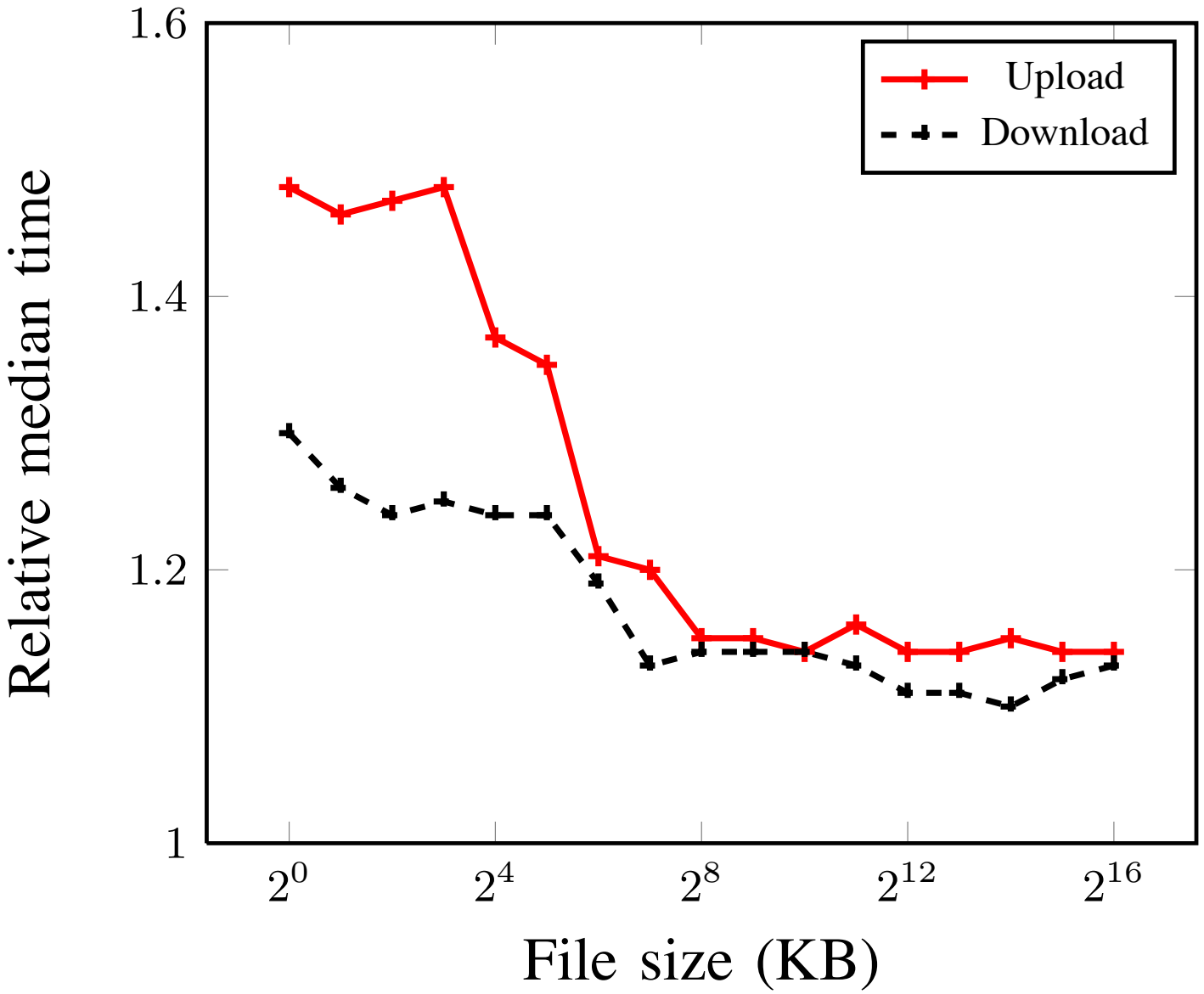


DupLESS: Performance breakdown of put

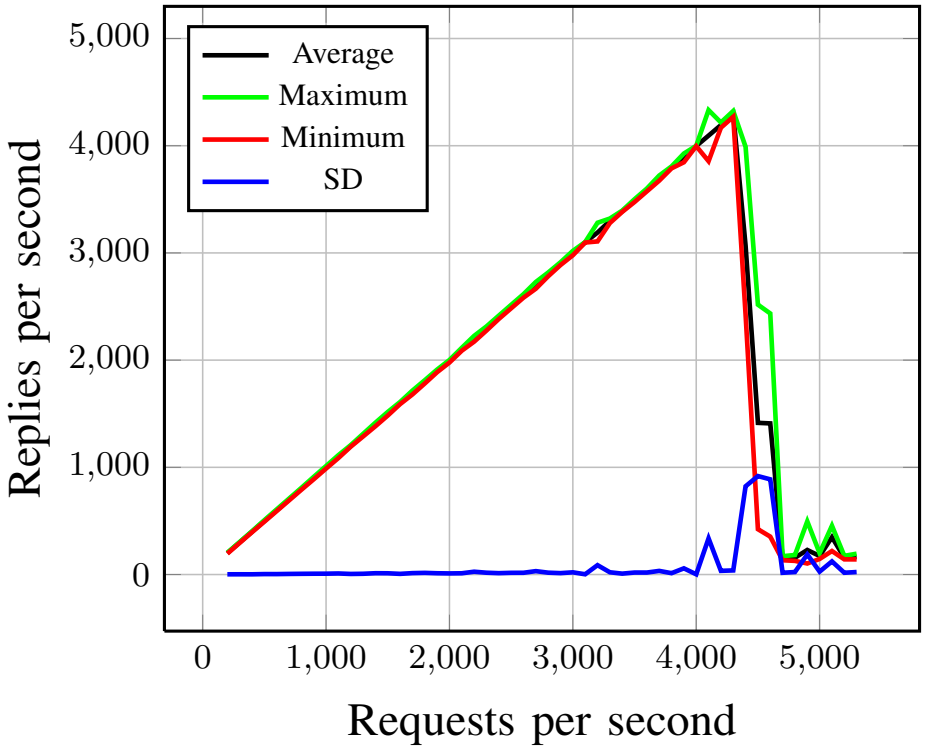
Component	Time in milliseconds		
	Min	Max	Median
Getting the KS Key	371	384	374 (9.1%)
Hashing	24	25	22 (0.5%)
Client-side RSA	6	6	6 (0.1%)
KS interaction	324	322	328 (8.0%)
TLS handshake	240	245	243 (5.9%)
Server-side operations	6	9	7 (0.2%)
Transmission	80	82	81 (0.2%)
Encryption	47	50	49 (1.2%)
Upload (SSput)	3452	4368	3676 (89.7%)
Sum total	3870	4802	4099

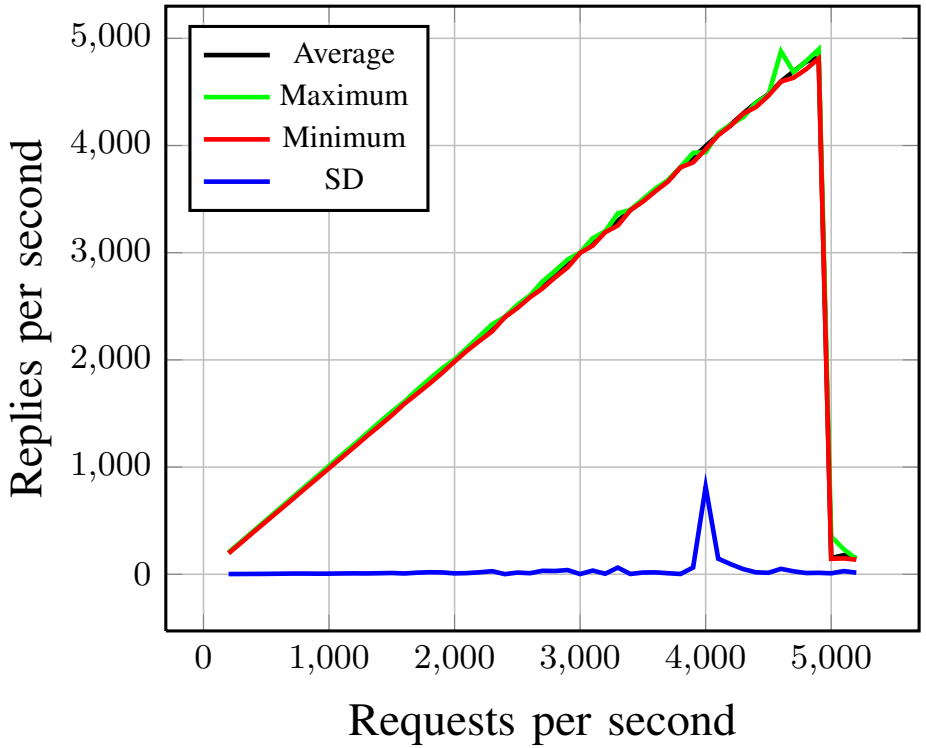
Performed for 1 MB file

DupLESS: Overhead of put and get



KS throughput (on EC2 m1.large instance)





Summary: Encryption that supports dedup

Formal foundations	Message-Locked Encryption (MLE) and new security definitions
Security analyses of CE and variants	Proofs or attacks for existing schemes and new variants
Theoretical foundations	Standard-model constructions (for restricted message spaces) and black-box from other primitives
Preventing offline brute-force attacks	Sever-aided MLE (SA-MLE) and the DupLESS system

Real World Crypto 2013

Debrief

"I have 256-bit stupidity" (paraphrased)
--Adam Langley, RWC 2013

"It's brainless, it's just programming"
--Nigel Smart, RWC 2013

RWC'13 Open Questions

Improving RNG designs and requirements? (Walker / Heninger)
Improved onioning AE schemes for Tor? (Mathewson)
How to deal with overblocking of Tor IP addresses? (Mathewson)
Concrete assumptions that suffice for RO-model schemes? (Bellare)
Revisiting crypto protocols based on many-to-few trust models? (Adida)
Can we make crypto more supportive of counter-cryptanalysis? (Stevens)
Thinking about MPC over real programs (not circuits)? (Smart)
Constant time algorithms and tools for evaluating constant timed-ness? (Langley)
Theory of implementation complexity / ecosystem? (McGrew, Langley)
Side-channel attacks against GCM? (Langley)

RWC'13 Open Questions

Soutions for bad pinning problem? (Perrin)
Designing protocols to be easily updatable in unilateral manner? (Rescorla)
More useful theory for cryptographic agility? (Paterson)
How do we get trust agility and have it be fast? (Applebaum)
…

Send me any interesting ones I missed!

RWC'13 Take-aways

- Implement your algorithms (Rogaway / Langley)
- Kill CBC (Langley and so many others)
- Go beyond the paper (Langley)
- Crypto gives bad demos (Spies)
- Unilateral versus bilateral deployment (Langley / Rescorla / Gueron)

Real World Crypto 2013

Please send slides (PDF) to Dan (dabo@cs.stanford.edu)

Big thanks to Dan for all the local arrangements!

And to Lynda Harris for logistics

And to our sponsors!

Real World Crypto 2014

- Please give us general feedback!
- Venue for next year or subsequent years?
  - Stay in Bay Area? East Coast? Europe?
  - Send us thoughts on this
- Actively looking for more sponsors
  - Talk to any of us!