a framework for evaluating storage system security

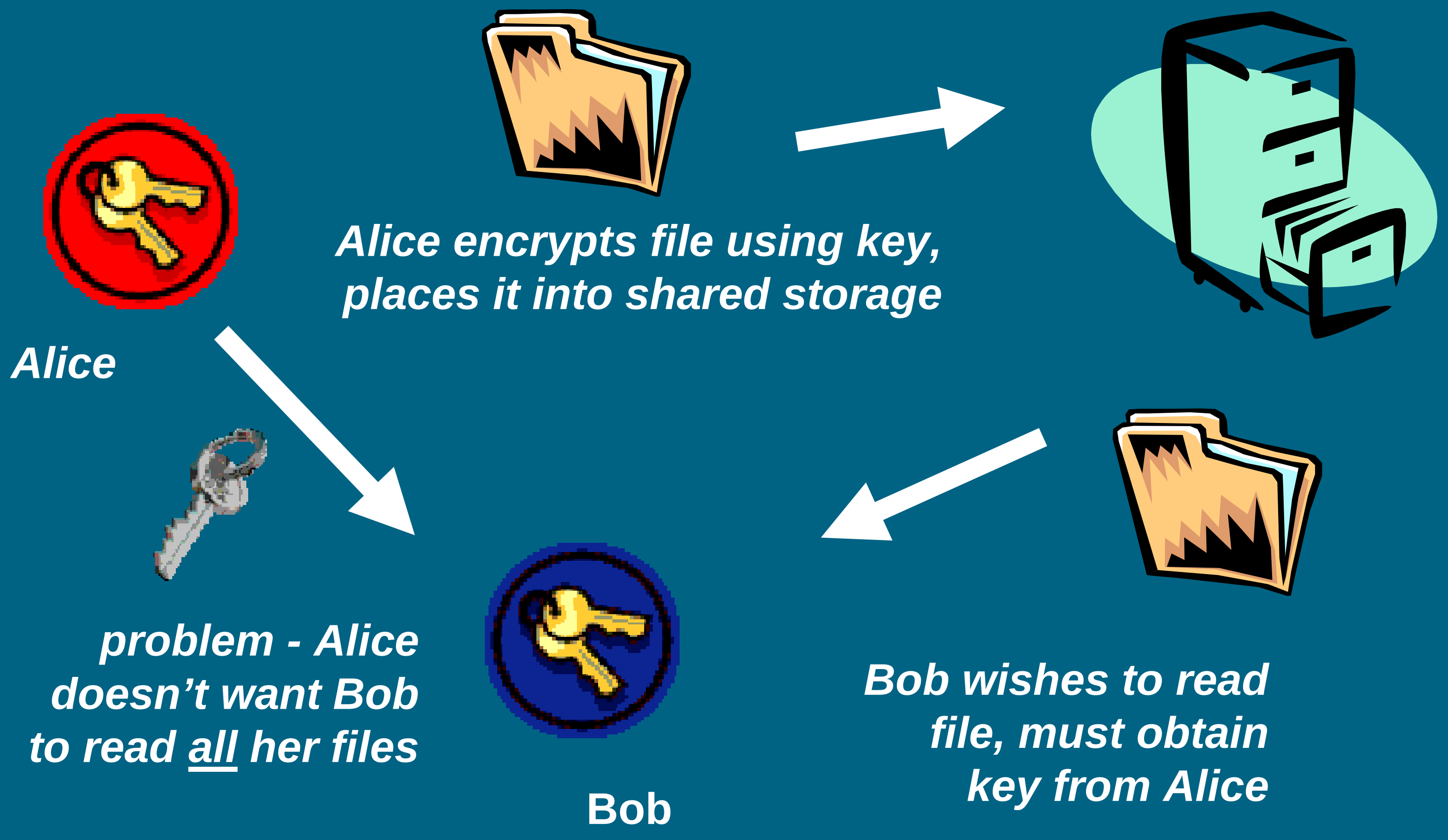
mahesh kallahalla,
erik riedel,
ram swaminathan

hp labs
january 2002

motivation

- storage security is not the same as network security
  - integrity & privacy of *persistent* data
  - secure *sharing* of data over the long term
  - specific optimizations possible for storage
- security work must be (more) quantitative
  - compare systems
  - informed performance, security, and user inconvenience trade-offs

protect and share



core issues

- our context
  - enterprise-scale and global-scale systems
  - large numbers of users
  - many, many data items
- challenges
  - scale is the overriding concern
  - too many keys
  - avoid centralization whenever possible
  - handle revocation as a common case

outline

- framework
  - players
  - attacks
  - existing systems
- design alternatives
- evaluation
- conclusions

framework

players

- *owners*
  - create data
  - determine access to data
- *readers* -- read
- *writers* -- modify
- *storage servers*
  - store / retrieve bits
- *group servers* (many flavors)
  - handle “delegated” keys
- *adversaries*
  - tampers with data
  - may collude w/ others

threats and attacks

attacks, as reported in survey of system managers by CSI/FBI, Spring 2001 *of ~500 responses, 78% had financial losses, only 37% could estimate damage	% surveyed	damage ($ millions)*	msgs		data			revoked user	denial of service
			leak	change	leak	change	destroy		
telecom eavesdropping	10%	1	✓	--	--	--	--	--	--
active wiretap	2%	n/m	--	✓	--	--	--	--	--
system penetration	40%	19	✓	✓	✓	✓	✓	--	--
laptop theft	64%	9	--	--	✓	--	✓	--	--
theft of proprietary info	26%	150	--	--	✓	--	--	✓	--
unauth access by insiders	49%	6	--	--	✓	✓	--	✓	--
sabotage	18%	5	--	--	--	--	✓	--	✓
virus	94%	45	--	--	--	--	✓	--	--
denial of service	36%	4	--	--	--	--	--	--	✓

framework

attacks

- attacks on data
  - leak
  - change
  - destroy
- adversary
  - act alone
  - collude w/ server
  - revoked user
- compromise group server
- denial of service

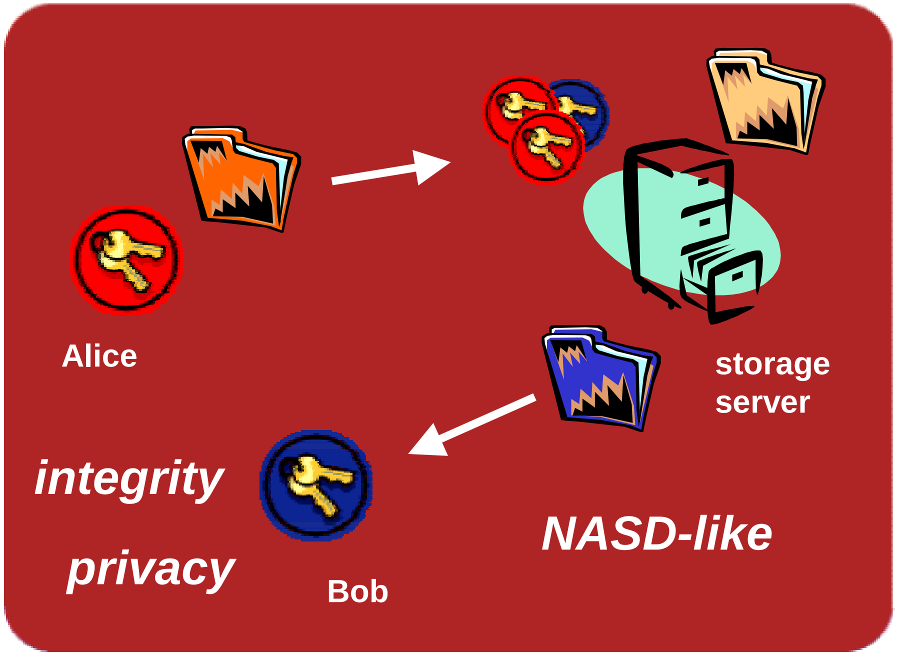
security guarantees - existing systems

system	message attacks	adversary leak	adversary change	adversary destroy	w/ storage srv leak	w/ storage srv change	w/ storage srv destroy	revoked leak	revoked change	subvert group server	denial of service
CFS	--	✓	✓	x	✓	✓	x	--	--	--	x
SFS-RO	✓	✓	✓	x	✓	✓	x	x	--	✓	x
Cepheus	✓	✓	✓	✓	✓	✓	x	✓	✓	x	x
SNAD	✓	✓	✓	✓	✓	✓	x	✓	✓	x	x
NASD	✓	✓	✓	✓	x	x	x	✓	✓	x	x
iSCSI w/ IPsec	✓	✓	x	x	x	x	x	✓	✓	--	x
LUN security	x	x	x	x	x	x	x	x	x	--	x
AFS	✓	✓	✓	✓	x	x	x	✓	✓	✓	x
NFSv4	✓	✓	✓	✓	x	x	x	✓	✓	x	x
PASIS/S4	--	--	--	✓	✓	✓	✓	--	--	--	x
OceanStore	--	✓	✓	✓	✓	x	✓	✓	✓	--	x

outline

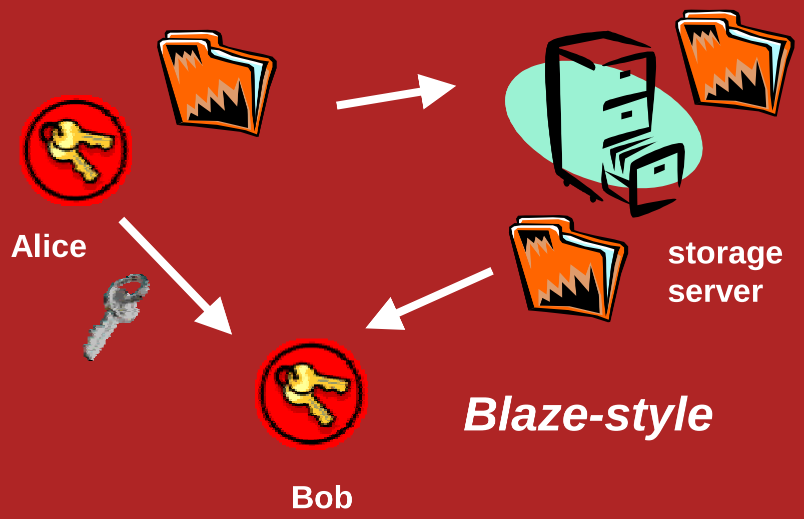
- framework
- design alternatives
  - encrypt-on-wire
  - encrypt-on-disk
- evaluation
- conclusions

encrypt-on-wire systems



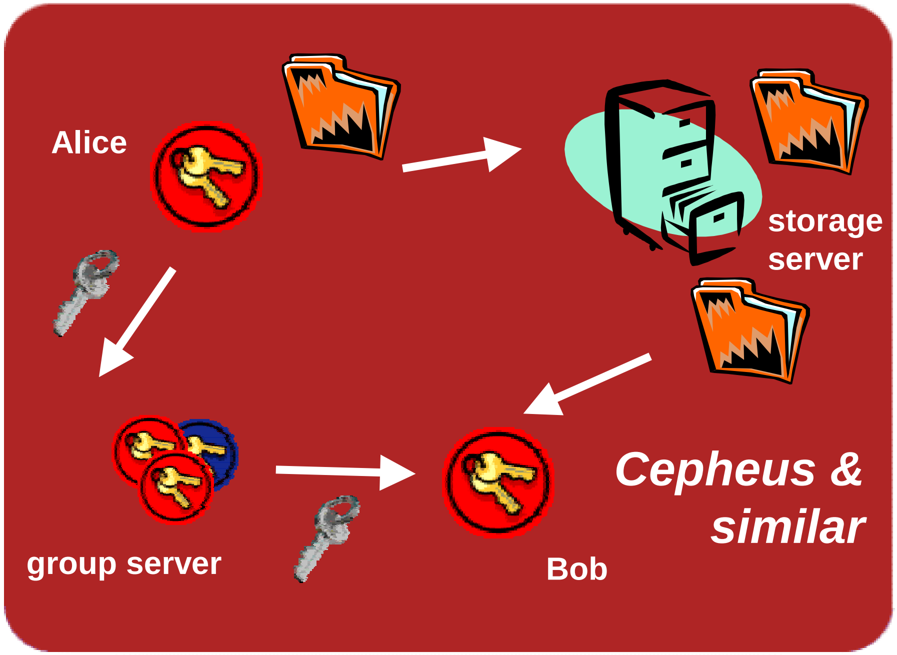
- checksums (integrity)
  - needed by any scheme
  - including signatures
  - session keys
  - pre-computed is a big help
- encryption (privacy)
  - expensive
    - > clients & servers both do encryption work
  - session keys
  - can't do pre-computation
- upside
  - straightforward layering
- downside
  - stored data is unprotected
  - expensive on critical path

Blaze-style encrypt-on-disk



- owners encrypt data
  - place into shared storage system
  - keep the keys
- readers / writers
  - contact owner for key
  - read / write data at will
- per-directory or per-file keys
  - entire sub-trees [Blaze94]
  - extreme is individual files
- upside
  - distributed, owner-managed
- downside
  - lots of keys
  - revocation expensive

Cepheus & similar encrypt-on-disk



- owners encrypt data
  - place into shared storage system
  - keys also stored on a server
- readers / writers
  - get key from group server
  - read / write data at will
- file groups vs. individual file keys
  - use same key for all files with the “same” permissions
  - `rw-r--r-- root bin`
- upside
  - distributed
- downside
  - centralized key server
  - revocation expensive*

outline

- framework
- design alternatives
- evaluation
  - key distribution
  - revocation
- conclusions

key distribution effort

user	per-directory		per-group+	
	dirs owned^	keys distributed*	groups owned^	keys distributed*
wilkes	6,400	640	28	18
alice	1,400	7	13	5
bob	14,000	1,000	17	11
bin	23,000	3,200	33	21
root	26,000	180	130	29
news	11,000	<500	15	5

* number of keys distributed by owners during a 12-hour trace

^ static numbers for the entire system (~500 GB, 4 million files total)

+ group is defined as same <owner>, <group>, <mode> permissions

key distribution effort

user	per-file		per-group+	
	files owned^	keys distributed*	groups owned^	keys distributed*
wilkes	54,500	4,000	28	18
alice	19,400	21	13	5
bob	216,000	3,200	17	11
bin	191,000	8,500	33	21
root	240,000	630	130	29
news	1,570,000	550	15	5

* number of keys distributed by owners during a 12-hour trace

^ static numbers for the entire system (~500 GB, 4 million files total)

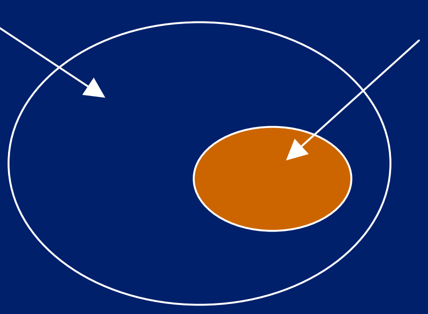
+ group is defined as same <owner>, <group>, <mode> permissions

revocation

- what happens when a user leaves the group or organization?
  - still has keys
  - could have copied data to floppies
- two consequences
  - stop using revoked keys
  - re-encrypt data
- problem
  - amount of re-encryption work for encrypt-on-disk is large

revocation

re-encryption



- lazy re-encryption [Fu99]
  - revoke user
  - change keys
  - mark files for re-encryption
  - only re-encrypt when file is next written
- performance improved at revocation time
- security reduced
  - "hole" closed only slowly

revocation

re-encryption

- quantifying performance
- total encryption work
  - *encrypt-on-disk*
    - > per-file 2 GB
    - > per-group 91 GB
  - *encrypt-on-wire*
    - > per-session 144 GB
- per-group encrypt-on-disk is 2x better performance than per-session encrypt-on-wire
- cost further reduced with lazy re-encryption (another 2x at least)

re-encryption effort

	per-file		per-group+	
	aggressive*	lazy^	aggressive#	lazy
files to be re-encrypted	3,740	469	546,000	121,000

* total number of files accessed by charlie in 10 days

^ total number of these files also accessed by someone else

number of files in all the groups accessed by charlie in 10 days

+ group is defined as same <owner>, <group>, <mode> permissions

re-encryption effort

	per-file		per-group+	
	aggressive*	lazy^	aggressive#	lazy
bytes to be re-encrypted	2 GB	0.5 GB	91 GB	43 GB
bytes encrypted by encrypt-on-wire	144 GB	144 GB	144 GB	144 GB

* total bytes in files accessed by charlie in 10 days

^ total bytes in these files also accessed by someone else

\# all bytes in files in all the groups accessed by charlie in 10 days

\+ group is defined as same <owner>, <group>, <mode> permissions

outline

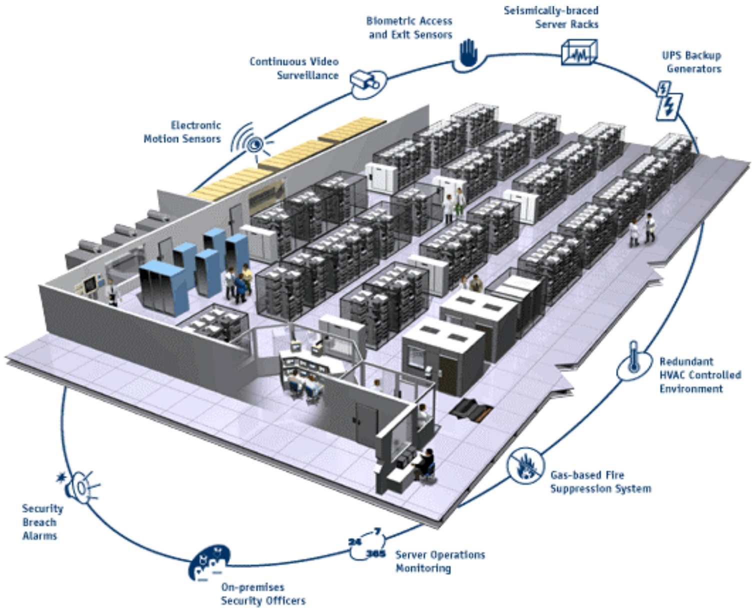
- framework
- design alternatives
- key distribution
- revocation
- conclusions
  - summary
  - future work

summary

- evaluation framework
  - compare trade-offs
- comprehensive solution
  - integrity-on-wire
  - encrypt-on-disk
    - > more efficient & secure
  - key distribution
    - > can be highly scalable
  - revocation
    - > must be treated as a common operation
- security *must* be end-to-end
  - optimize locally
  - best efficiency achieved at individual functions

future work

- design & prototype
  - large scale, shared storage system
  - key management
  - optimized revocation
- security metrics
  - further toward quantitative metrics
- user inconvenience
  - even more difficult to quantify
- denial of service
  - not explored yet



information shadow

wherever you go, your data is always with you

i n v e n t

extra slides

cryptographic operations

operations, basic crypto functions, and which systems bear which costs, data from 10-day cello trace		peak load (one minute)		systems				
		messages (req/s)	bandwidth (MB/s)	NASD	iSCSI w/ IPsec	CFS	SFS	Cepheus
integrity	message signatures	10,200	n/a	✓	✓	n/a	✓	✓
	checksums	10,100	13.9	--	✓	n/a	✓	--
	pre-computed cksum	5,100	5.1	✓	--	--	--	✓

- cost of the various cryptographic functions
  - either bandwidth/cycles required from hosts & devices
  - or bandwidth required from a hardware assist

cryptographic operations

operations, basic crypto functions, and which systems bear which costs, data from 10-day cello trace		peak load (one minute)		systems				
		messages (req/s)	bandwidth (MB/s)	NASD	iSCSI w/ IPsec	CFS	SFS	Cepheus
integrity	message signatures	10,200	n/a	✓	✓	n/a	✓	✓
	checksums	10,100	13.9	--	✓	n/a	✓	--
	pre-computed cksum	5,100	5.1	✓	--	--	--	✓
privacy - server	encryption (reads)	1,100	7.9	✓	✓	--	--	--
	decryptions (writes)	1,700	10.7	✓	✓	--	--	--
privacy - client	encrypt/decrypt	2,700	4.9	✓	✓	✓	✓	✓

additional concerns

- differential cryptanalysis
  - volume of data encrypted with the same key
  - “known plaintext” attacks
- system is only as strong as it’s weakest link
  - authentication (verify who is who)
  - trusted OS (APIs, trust cores / rings)
  - key storage (smart cards, trust cores)
- destruction of data
  - information dispersal
    - > replica management
- denial of service
  - not yet explored