Background: Basic Cryptography

- Symmetric Key System
  - a shared symmetric key
  - Examples: DES, IDEA, RC4, AES
- Asymmetric Key System
  - a pair of private and public keys
  - Examples: RSA, DSA, ElGamal, Rabin, FFS

Background: Authentication Services

- Needham-Schroeder Protocols (*CACM*, 1978)
  - Kerberos (MIT, 1988) – part of project Athena (1983-1991) to develop campus-wide distributed computing environment
  - ...
- Secure sockets layers
  - SNP (U. Texas at Austin, 1993)
    - offshoot from authentication protocol verification work sponsored by NSA
    - to secure Internet applications that use TCP (or UDP)
    - published in *Proceedings USENIX*, June 1994
  - SSL (Netscape, 1995, 1996)
  - TLS (1999)

Motivation (circa 1997)

- Traditional network applications
  - message-oriented unicast,
    e.g., email, file transfer, client-server
- Emerging network applications
  - flow-oriented, e.g., audio, video, stock quotes
  - multicast, e.g., teleconference, software distribution
- Problem 1: Secure group communications - scalability
- Problem 2: How to sign efficiently?

Secure Group Communications Using Key Graphs

by Chung Kei Wong, Mohamed Gouda, and Simon S. Lam
in *Proc. ACM SIGCOMM '98*

Secure group communications

- Applications
  - teleconference
  - information services
  - collaborative work
  - virtual private networks
- Group members share a symmetric key to
  - encrypt/decrypt communications

    providing confidentiality, integrity, and authenticity of messages delivered between group members
  - access resources

Group key management

- A group session may persist for a long time
- Secure rekeying
  - after each join
  - after each leave
  - periodically -> batch rekeying (another paper)
- Scalable server and protocols
  - for large groups with frequent joins and leaves
- Scalable and reliable transport (Zhang, Lam, Lee, Yang, 2003)

Assumptions

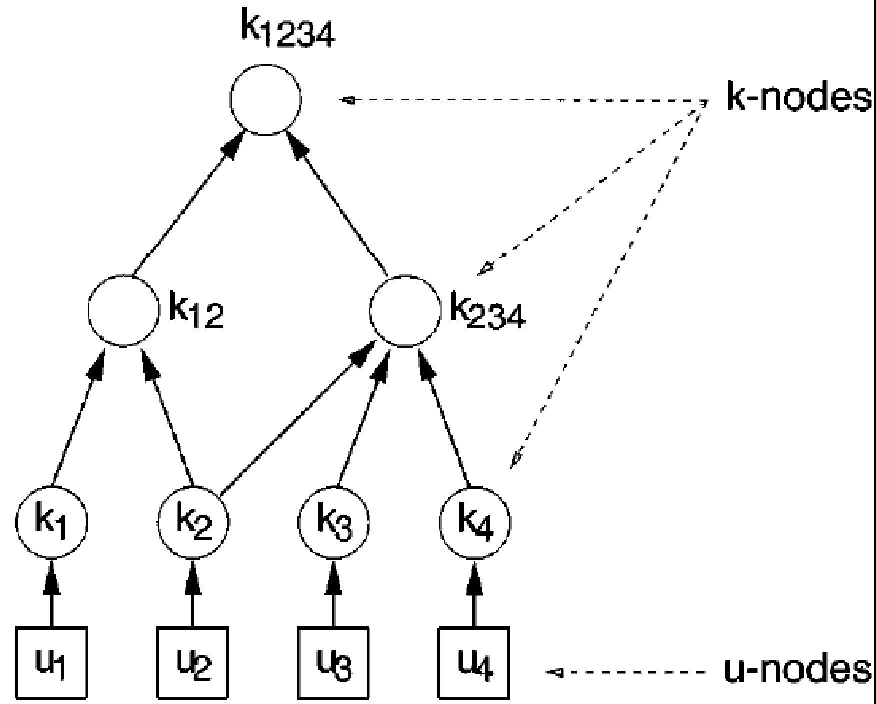
- Key server is trusted and secure (may be replicated)
- An authentication service
  - for example, SSL
  - mutual authentication of server and joining user
  - distribution of a key shared by server and joining user (individual key)
- Access control by key server or by an authorization service (e.g., a set of registrars)

Group rekeying

- Non problem after a join
  - new group key encrypted by old group key
  - one encryption/rekey msg for all existing users
- After a leave has occurred
  - new group key encrypted by individual key of each user
  - *n*-1 encryptions/rekey messages for group size *n*
  - *not scalable*

Key graph

- A directed acyclic graph with *u*-nodes and *k*-nodes
  - *u*-node – no incoming edge
  - root – a k-node with no outgoing edge
  - user *u* has key *k* if and only if there is a directed path from node u to node k
  - one or more roots (e.g., for multiple groups)



- userset(k) is set of users that hold k
- keyset(u) is set of keys held by u

Key covering problem

- When a user u' leaves a secure group, every key k' that has been held by u' and shared by other users should be changed
- To minimize the work of rekeying, the server would like to find a minimum size subset K' of keys and securely send new keys to affected users i.e., userset(K') is the subset of users who need new keys
- This problem is NP-hard in general

Special cases of key graph

n users, 1 key server manages key graph

- Star
- Tree - assumed to be full and balanced with height *h*, degree *d*
- Complete - a key for every nonempty subset of users (there are $2^n - 1$)

	Star	Tree	Complete
Total # of keys	$n+1$	$\frac{d}{d-1}n$	$2^n - 1$
# of keys per user	2	h	2^{n-1}

Key star

Group of *n* users, one group key, *n* individual keys

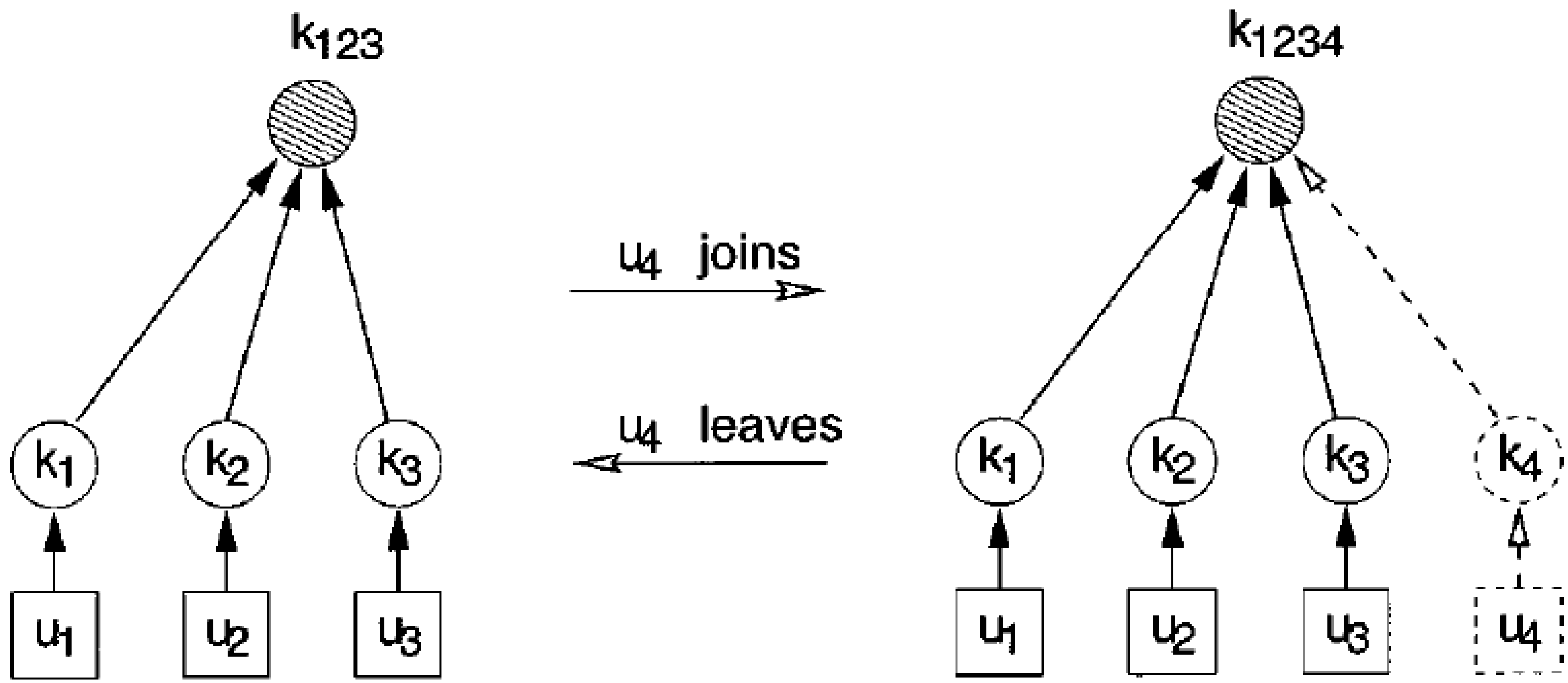


Fig. 3. Star key graphs before and after a join (leave).

Join Protocol

- Protocol

$u_4 \rightarrow s$: join request

$s \leftrightarrow u_4$: mutual authentication, distribute k_4

s : generate k_{1234}

$s \rightarrow u_4 : \{k_{1234}\}_{k_4}$

$s \rightarrow \{u_1, u_2, u_3\} : \{k_{1234}\}_{k_{123}}$

- Encryption cost: 2

Leave Protocol

- Protocol

  $u_4 \rightarrow s$: {leave request} $_{k4}$

  $s \rightarrow u_4$: {leave granted} $_{k4}$

  s: generate k_{123}

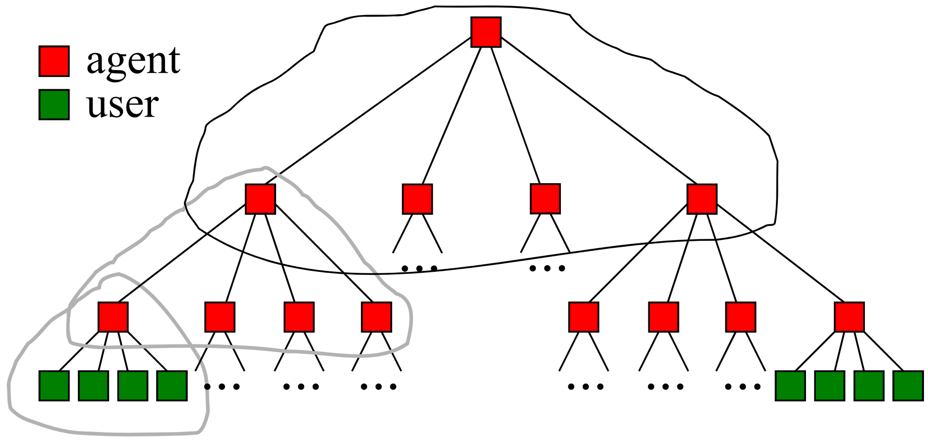
  $s \rightarrow \{u_1\}: \{k_{123}\}_{k1}$

  $s \rightarrow \{u_2\}: \{k_{123}\}_{k2}$

  $s \rightarrow \{u_3\}: \{k_{123}\}_{k3}$

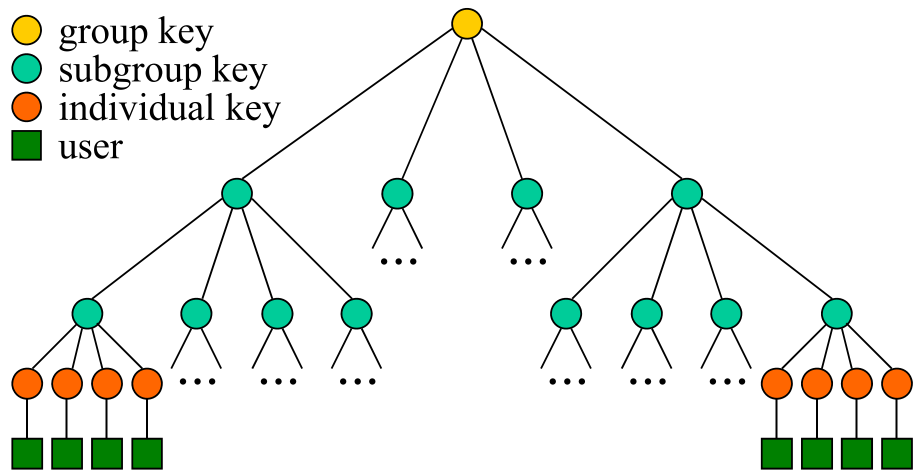
- Encryption cost: *n*-1 for group size *n*
- O(*n*) cost is not scalable

Iolus approach [Mittra 1997]



- A hierarchy of security agents
- No globally shared group key
  - join/leave affects local subgroup only
- Agents forward message key
  - decrypting and re-encrypting it with subgroup keys
- Requirement: many trusted agents

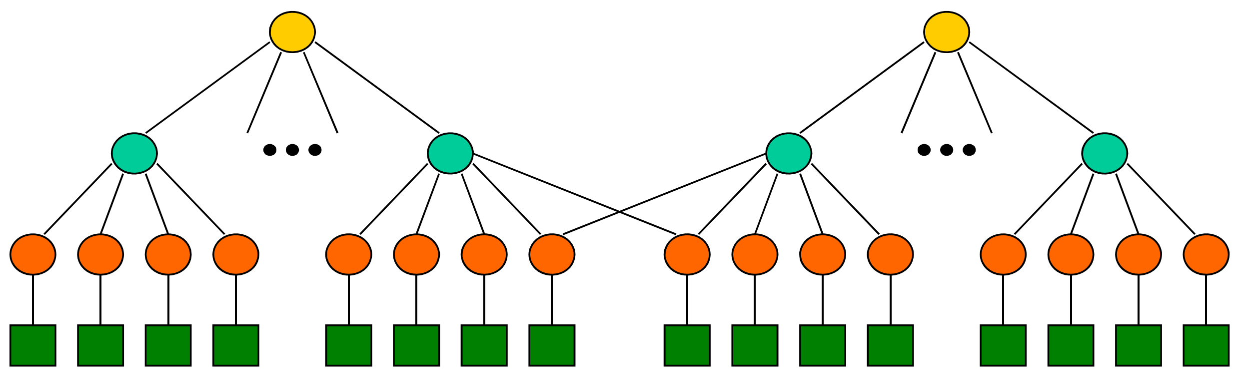
Our approach



- A hierarchy of keys
- Multiple keys for each user
  - user has every key along path to root
- A single trusted key server is sufficient (may be replicated for reliability)

Key graph

- Data structure maintained by key server
- For a single secure group
  - key tree sufficient for scalability
- Multiple secure groups
  - merging multiple trees into a graph



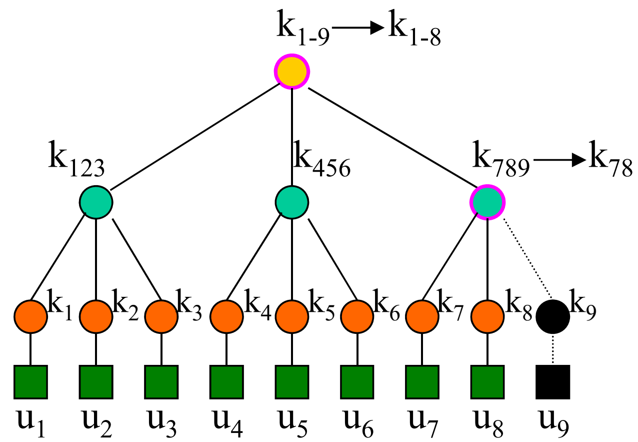
Rekeying strategies

How to compose and deliver rekey messages

- user-oriented
- key-oriented
- group-oriented

User-oriented rekeying

- Select new keys needed by a user or subset of users, form a rekey message and encrypt it
- (d-1)(h-1) rekey messages - sent by unicast or subgroup multicast
- Most work on server, least work on user



Leaving

$$s \rightarrow \{u_1, u_2, u_3\} \quad : \quad \{k_{1-8}\}_{k_{123}}$$

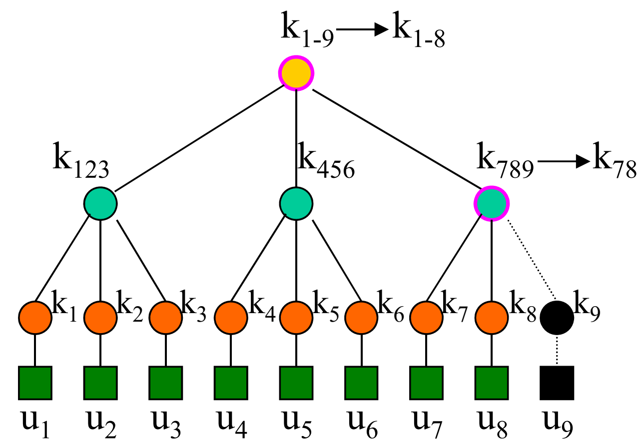
$$s \rightarrow \{u_4, u_5, u_6\} \quad : \quad \{k_{1-8}\}_{k_{456}}$$

$$s \rightarrow u_7 \quad : \quad \{k_{1-8}, k_{78}\}_{k_7}$$

$$s \rightarrow u_8 \quad : \quad \{k_{1-8}, k_{78}\}_{k_8}$$

Key-oriented rekeying

- Encrypt each new key, then compose rekey messages - encryption cost d(h-1) -1
- (d-1)(h-1) rekey messages – sent by unicast or subgroup multicast
- Less work on server than user-oriented



Leaving

$$s \to \{u_1, u_2, u_3\} \quad : \quad \{k_{1-8}\}_{k_{123}}$$

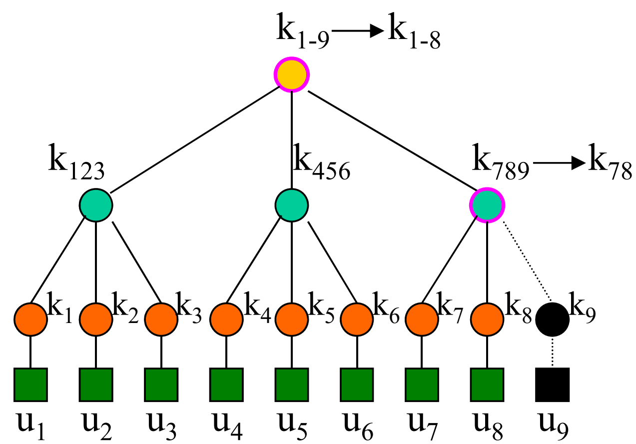
$$s \to \{u_4, u_5, u_6\} \quad : \quad \{k_{1-8}\}_{k_{456}}$$

$$s \to u_7 \quad : \quad \{k_{1-8}\}_{k_{78}}, \{k_{78}\}_{k_7}$$

$$s \to u_8 \quad : \quad \{k_{1-8}\}_{k_{78}}, \{k_{78}\}_{k_8}$$

Group-oriented rekeying

- One rekey message containing all encrypted new keys – sent by multicast
- Message size $O(\log n)$
- Each user decrypts what it needs
- Least work on server, most work on user
- *A user cannot decrypt any key that does not belong to the user*



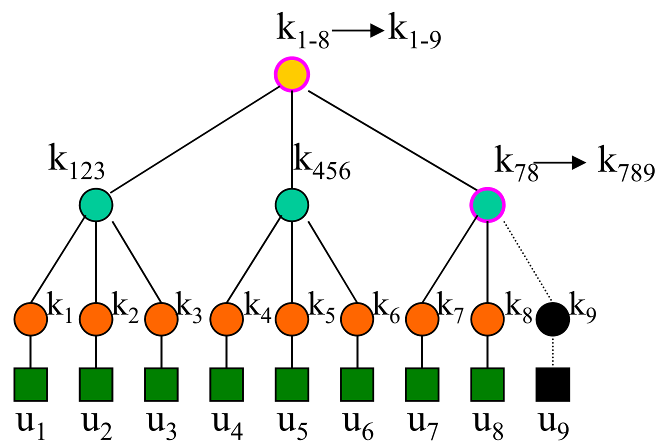
Leaving

$$s \to \{u_1,\ldots,u_8\} \quad :$$

$$\{k_{78}\}_{k_7}, \{k_{78}\}_{k_8},$$
$$\{k_{1-8}\}_{k_{123}}, \{k_{1-8}\}_{k_{456}},$$
$$\{k_{1-8}\}_{k_{78}}$$

Join: group-oriented rekeying

- Encryption cost: 2(h-1)
- Key tree incurs a larger cost than key star



Joining of u_9:

$s \rightarrow \{u_1, \ldots, u_8\}$: $\{k_{1-9}\}_{k_{1-8}}, \{k_{789}\}_{k_{78}}$

$s \rightarrow u_9$: $\{k_{1-9}, k_{789}\}_{k_9}$

Ave. encryption/decryption cost of a request

(a)	the requesting user		
	Star	Tree	Complete
join	1	$h-1$	2^n -1
leave	0	0	0

(b)	a non-requesting user		
	Star	Tree	Complete
join	1	$d/(d-1)$	2^{n-1}
leave	1	$d/(d-1)$	0

(c)	the server		
	Star	Tree	Complete
join	2	$2(h-1)$	2^{n+1}-2
leave	$n-1$	$d(h-1)-1$	0

Average encryption/decryption cost of a request (join or leave)

	Star	Tree	Complete
cost of the server	$n/2$	$(d+2)(h-1)/2$	2^n
cost of a user	1	$d/(d-1)$	2^n

- For a full and balanced tree, $h-1=\log_d(n)$
- For a key tree (instead of key star), server does less work, but user does slightly more work
- Optimal key tree degree is 4

Experiments

- Two SGI machines connected by 100 Mbps Ethernet
  - server on one, users on the other
- Rekey messages sent as UDP packets
- DES, MD5, RSA from CryptoLib
- *n* joins, then 1000 randomly generated join/leave requests

Server processing time per join/leave request

includes:

- time to parse a request, traverses key tree to determine which keys to change, generates new keys, updates key tree
- time to encrypt new keys and construct rekey messages,
- time to compute message digest of rekey messages and digital signatures,
- time to send out rekey messages using socket system calls

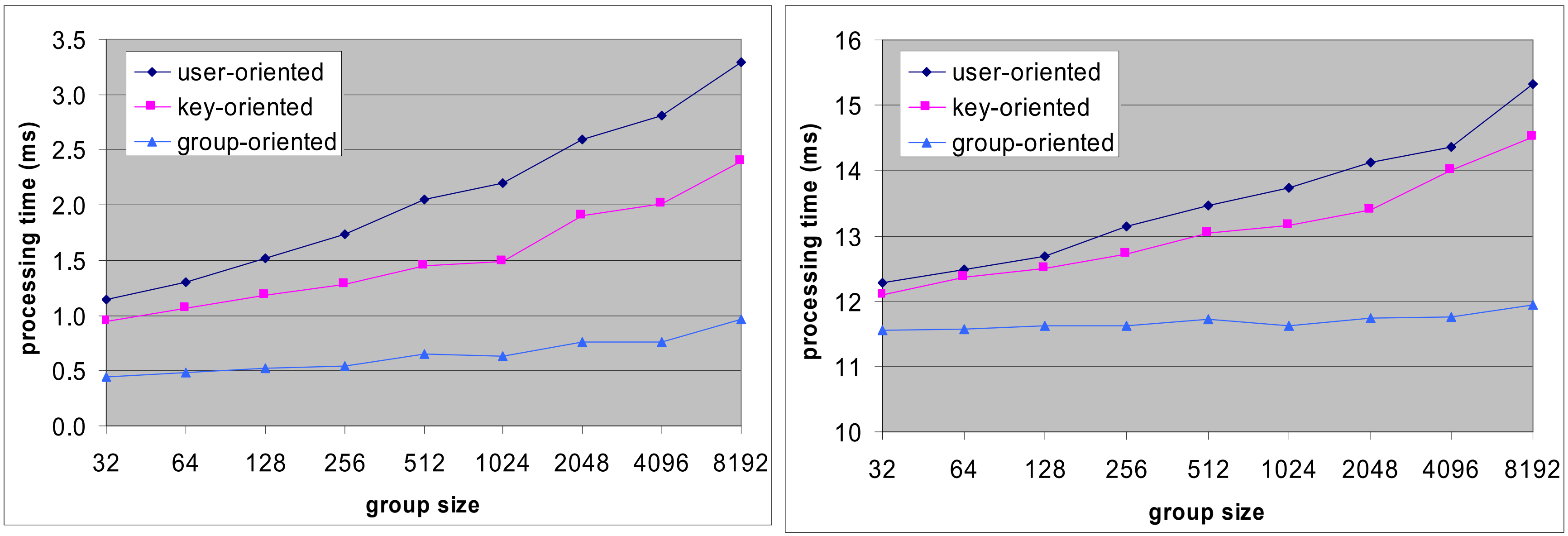
Technique for signing rekey messages

	one signature per rekey msg				
	msg size (byte)		proc time (msec)		
	join	leave	join	leave	ave
user-oriented	263.1	233.8	76.7	204.6	140.6
key-oriented	303.0	270.9	76.3	203.8	140.1
group-oriented	525.5	1005.7	11.9	12.0	11.9

	one signature for all rekey msgs				
	msg size (byte)		proc time (msec)		
	join	leave	join	leave	ave
user-oriented	312.8	306.9	13.6	17.1	15.3
key-oriented	352.8	344.0	13.1	15.9	14.5
group-oriented	525.5	1005.7	11.9	12.0	11.9

key tree degree 4, initial group size 8192, encryption and signature

Server processing time (per request) versus group size

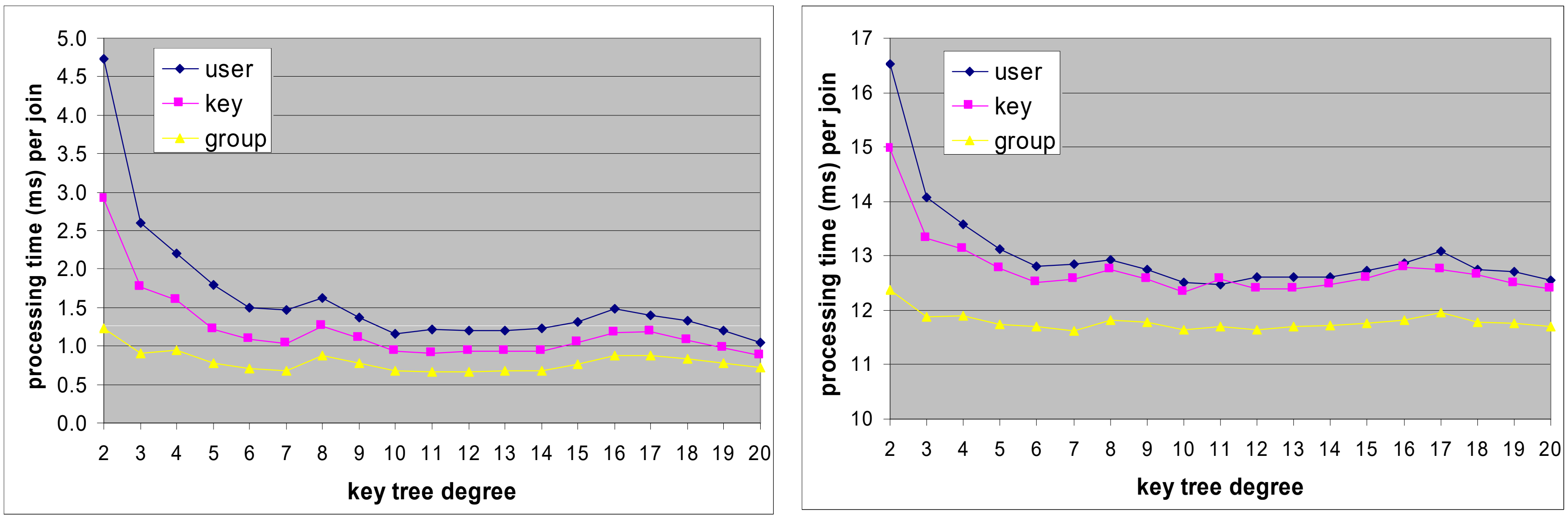


encryption only **encryption and signature**

- Increases linearly with logarithm of group size

Server processing time versus key tree degree (per join)

Initial group size 8192

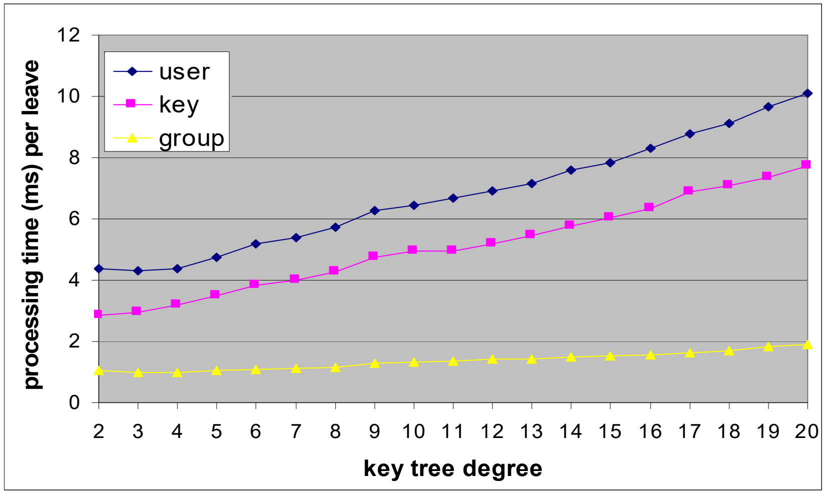


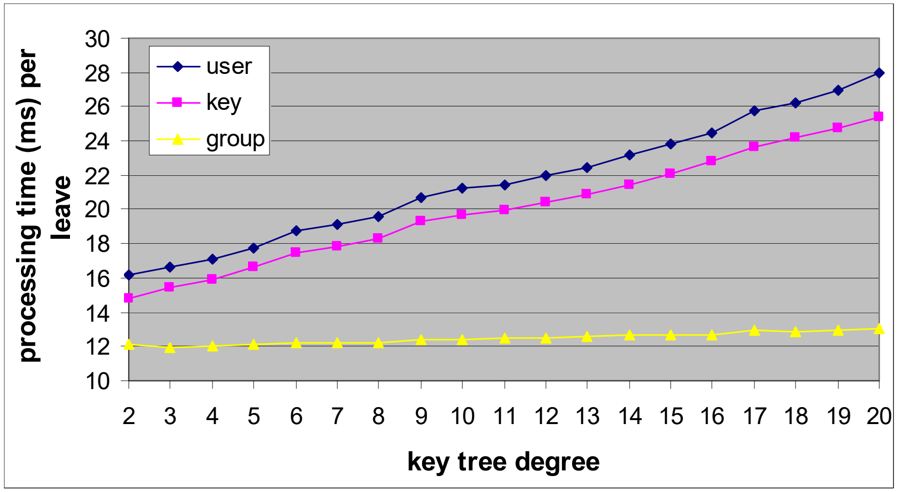
encryption only **encryption and signature**

Cost is proportional to 2(h-1)

Server processing time versus key tree degree (per leave)

Initial group size 8192



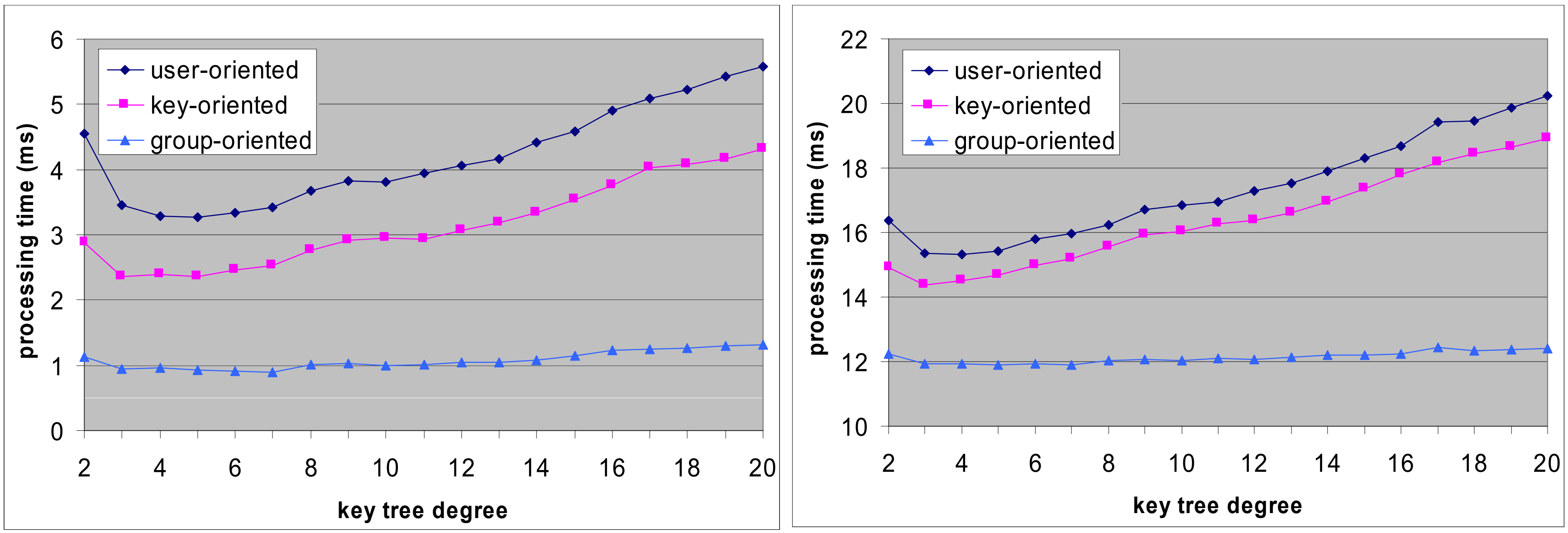


encryption only

encryption and signature

Cost is proportional to d(h-1)

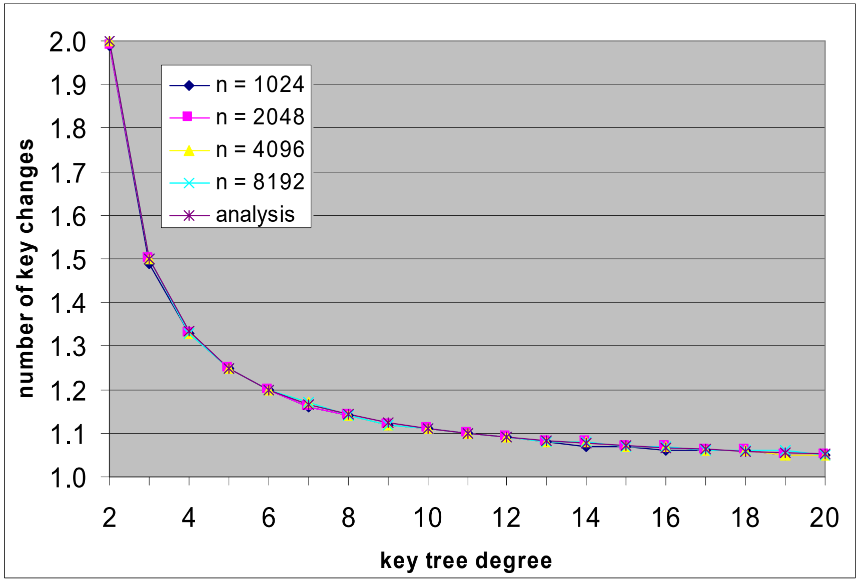
Server processing time versus key tree degree (per request)

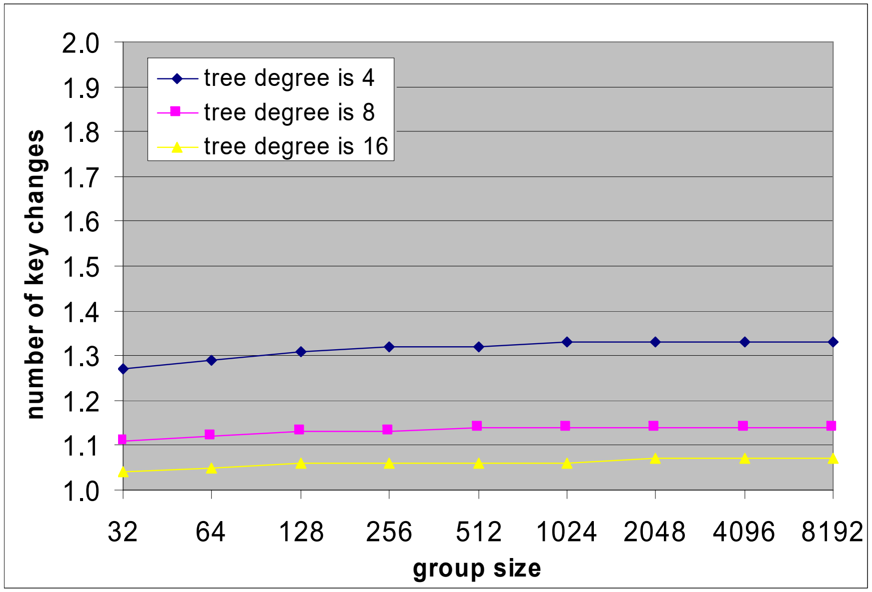


encryption only **encryption and signature**

- Initial group size 8192
- 4 is optimal degree (analytic result)

Number of key changes by a user (per request)





- Very close to analytic result, $d / (d - 1)$

Rekey messages sent by server

- With encryption and signature (initial group size 8192, key tree degree 4)

	Ave. rekey message size (bytes)		Ave. number of rekey messages	
	per join	per leave	per join	per leave
User-oriented	312.8	306.9	7.00	19.02
Key-oriented	352.8	344.0	7.00	19.02
Group-oriented	525.5	1005.7	1	1

- Total number of bytes sent is much smaller for group-oriented rekeying than the others

Rekey messages received by user

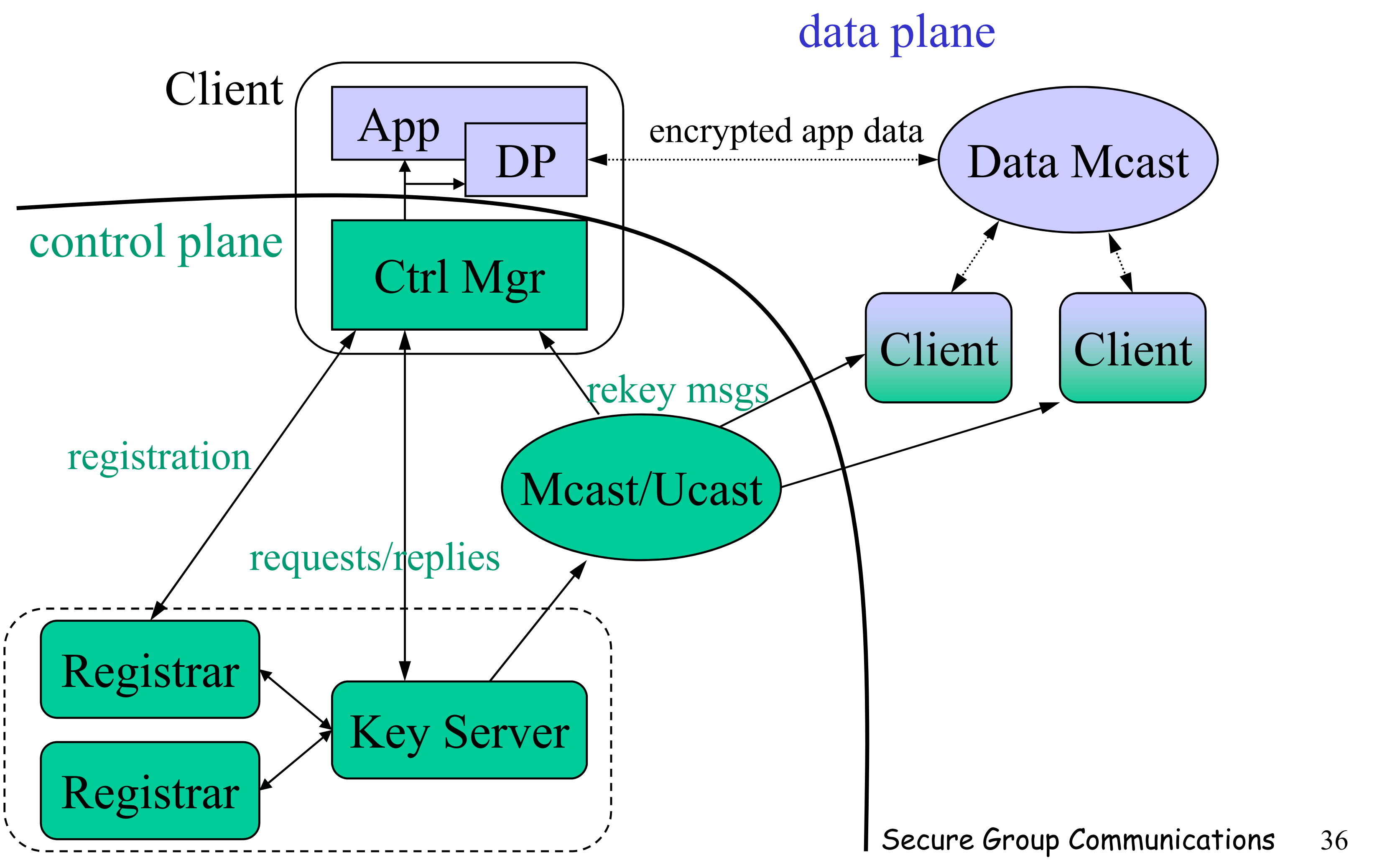
- With encryption and signature (initial group size 8192, key tree degree 4)

	Ave. rekey message size (bytes)		Ave. number of rekey messages	
	per join	per leave	per join	per leave
User-oriented	209.3	237.4	1	1
Key-oriented	227.9	256.0	1	1
Group-oriented	525.5	1005.7	1	1

Conclusions

- Scalable server performance demonstrated experimentally and analytically
  - Group-oriented rekeying requires smallest processing time and transmission bandwidth of server (signing is also easier), but requires each user to do more work
  - Hybrid approach with use of user- or key-oriented rekeying for users with limited capabilities
- Solution to just the most obvious problem of scalable server processing
  - Many more papers to follow

Keystone system architecture [C. K. Wong and S. S. Lam, 2000]



Extensions

- Batch rekeying
- Reliable and scalable communications [Zhang, Lam, Lee, Yang, 2003]
  - Proactive FEC with unicast recovery – this works well because each client needs only a small fraction of new keys
  - Adaptive FEC
  - Key identification, FEC block id estimation, etc.
- Not done
  - Replicated servers and registrars
  - Multiple groups - access control of resources

End