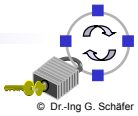
Network Security

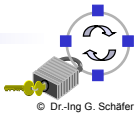
Chapter 8*
Secure Group Communication

- Introduction
- Requirements
- Classification
- Public-key- & Group-key-based Schemes
- Source Authentication



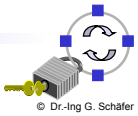
Introduction – Scenarios for Secure Group Communication

- One-to-many communication (broadcast or multicast)
  - E.g. P2P Streaming applications
  - One server providing information
  - Many recipients (Stationary & Mobile)
  - Demands to protocol
    - Low latency
    - Low bandwidth requirements at the server
    - Security management mainly by the server
- Many-to-many communication
  - E.g. Video conferences, P2P file sharing
  - All members send and receive information within the group
  - Groups usually exist only for a certain time
  - Demands to protocol
    - Distributed key agreement



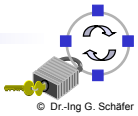
Introduction – Security in Group Communication

- In contrast to unicast the support for confidentiality and authentication of data is more difficult to realize in multicast scenarios
- Special mechanisms required!
- Group Key Management (GKM) usually deployed to provide confidentiality
  - Creation of a common secure context between all participants in group communication by establishing a group-wide known key
  - Agreements
    - Traffic Encryption Key (TEK) to secure the group traffic
    - Key Encryption Key (KEK) for secure TEK distribution
- Source authentication requires some approach with asymmetric knowledge
  - Group members must verify that packets come from the source
  - Must not be able to generate these packets by themselves!



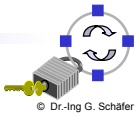
Requirements for global multicast systems

- Scalability
  - When introducing security mechanisms it should still be possible to communicate in large groups (>1.000 participants)
- Moderate resource usage
  - No high processing, storage, communication overhead for participants, which can be limited in their resources
- Ability to cope with churn
  - GKM should sustain bursty behavior of nodes, which is the simultaneously join and leave of nodes in high frequency
- Availability
  - Failure of a single entity of the GKM architecture should not lead to a failure of the whole session



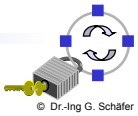
Further security requirements

- Minimal trust in infrastructure
  - GKM should rely only on a minimal number of trusted entities, if possible it leverages existing infrastructure instead of adding new components
- Key independence
  - New group keys should not be dependent on older ones
- No 1-affects-n
  - A single change in the group compound should not lead to a group-wide re-keying
- Backward and forward secrecy
  - No joining member should be able to decrypt the past traffic
  - No leaving member should be able to decrypt the future traffic
- Controlled Access
  - Only legitimated group members should be allowed to take part in group communication



Schemes providing confidentiality

- Public-key-based schemes
- Group-Key-based schemes
  - Centralized Approaches
    - Based on Pairwise keys
    - Tree-based
    - Re-keying based on broadcast
  - Decentralized Approaches
  - Contributory Approaches
    - N-party Diffie-Hellman
      - Ring-based
      - Tree-based



Public-key-based schemes

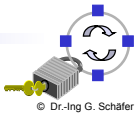
- Public-Key-based Dynamic Conferencing Scheme (PKDC) [ZRM04]
  - Whenever a member m_i wants to send a message *M* to a conference *C*:

$$C = \{m_1,\ldots,m_l\} \cup \{m_i\}$$

1. m_i selects a random session key K
2. It encrypts the message with K and independently with the public keys $P_{i_1},\cdots,P_{i_l}$ of the respective members $m_{i_1},\cdots,m_{i_l}$
3. m_i broadcasts or multicasts the encrypted message along with the encrypted *k* to the group

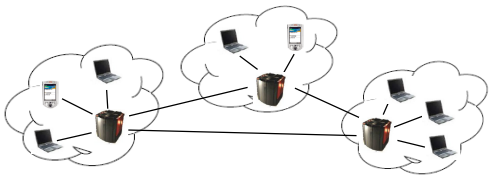
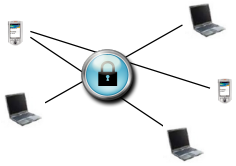
$$m_i \rightarrow *:(\{K\}_{+K_1},\ldots,\{K\}_{+K_l},\{M\}_K)$$

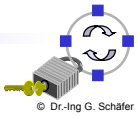
4. Members in conference *C* recover *K*, by using their private keys and decrypt the message afterwards



Group-key based Schemes

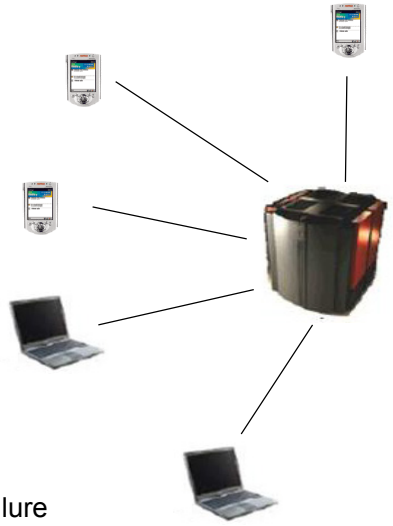
- Centralized
  - Central entity takes over the role of a Group Controller (GC), which is responsible for key-generation and secure distribution
  - Central GC is bottleneck and single-point-of-failure
- Decentralized
  - Several GC take over responsibility for subgroups, respectively
- Contributory
  - Key-generation is performed in distributed manner, in the absence of a central authority
  - Higher message and computation effort compared to other classes

Centralized	Decentralized	Contributory
		



Centralized GKM (1)

- One single key-server, responsible for key-generation and key-distribution, called Group Controller (GC)



- 3 possible approaches:
  - Pairwise keys
    - Group Key Management Protocol (GKMP)
  - Re-keying based on broadcast
    - Secure Locks
  - Hierarchy of keys
    - Logical Key Hierarchy (LKH)
- All centralized protocols have in common:
  - Central entity is bottleneck and single point of failure

Centralized GKM (2) - GKMP

- Group Key Management Protocol (GKMP) [HM97a, HM97b]
  - Central GC maintains pairwise keys to every member
  - The TEK is distributed via pairwise keys
  - To allow for backward and forward secrecy:
    - During member join the new TEK is encrypted and distributed by the old one and once by the joining member‘s key
      - O(1) Encryptions
    - A member leave requires a re-keying via pairwise keys
      - O(n) Encryptions

    - × GC is potential single point of failure and bottleneck
    - × Not scalable (1-affects-n)
    - × Churn poses a problem
    - × No key-independence

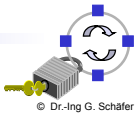
Centralized GKM (3) – Secure Locks

- Secure Locks [CC89]
  - Re-keying based on broadcast
  - An offline server computes r pairwise relatively prime natural numbers

$$N_1,...,N_r \text{ ^ } \gcd(N_i,N_j) = 1 \text{ if } i \neq j$$

  - Each number is assigned to a group member
  - Furthermore, the server shares a secret k_i with every member to compute a *lock* L
  - GC generates a random key K, encrypt it pairwise with every k_i , result is a set of K_i
  - The lock L is the solution of the following equation system, which can be solved by using the Chinese Remainder Theorem

$$L \equiv K_1 \bmod N_1$$
$$\ldots$$
$$L \equiv K_n \bmod N_n$$



TELEMATIK Rechnernetze

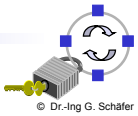
Centralized GKM (3) – Secure Locks

- Out of L, every member, whose N_i and k_i was included in computation, can recover a group key K, by computing

$$K = \{ L \bmod N_i \}_{k_i}$$

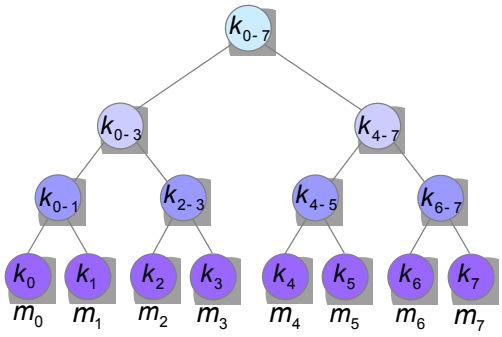
  - ✓ Only one message for a complete re-keying
  - ✓ Relatively low computational expenditure for group members
  - ✗ GC is potential single-point-of-failure and bottleneck
  - ✗ Not scalable
  - ✗ High computation effort
  - ✗ 1-affects-n and susceptible to churn

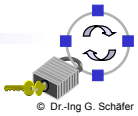
- Dynamic conferencing based on SecureLocks
  - A member willing to communicate with only a specific set of other members in the overall group, includes only the N_i of members contained in the set into computation of the lock L
  - Lock L is broadcasted in the group and only included members are able to retrieve the key
  - For dynamic conferencing no central GC necessary



Centralized GKM (4) – LKH

- Logical Key Hierarchy (LKH)
  - GC shares a pairwise key with every group member and in addition KEKs with subgroups of the group
  - The keys form a logical key tree, in which the TEK is the key of the root node, whereas the pairwise keys are assigned to the leafs
  - The intermediate keys on the paths from the leaf nodes to the root are the subgroup keys and known to all members of that group
  - In case of a member join or leave all keys of the affected nodes from leaf node to the root have to be changed from bottom up to the root key

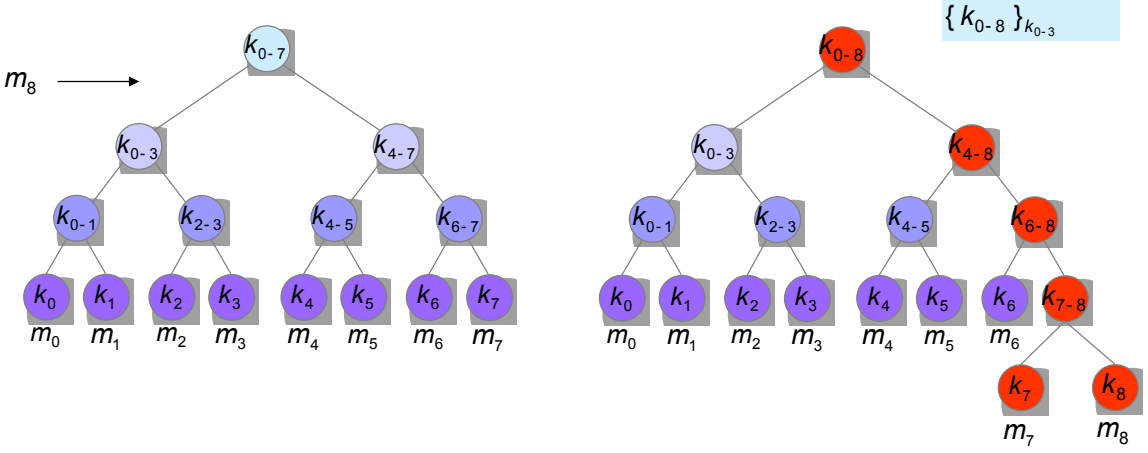


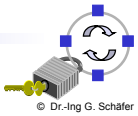


Centralized GKM (6) - LKH

- Member Join
  - Find position for insertion of new members
  - Change keys

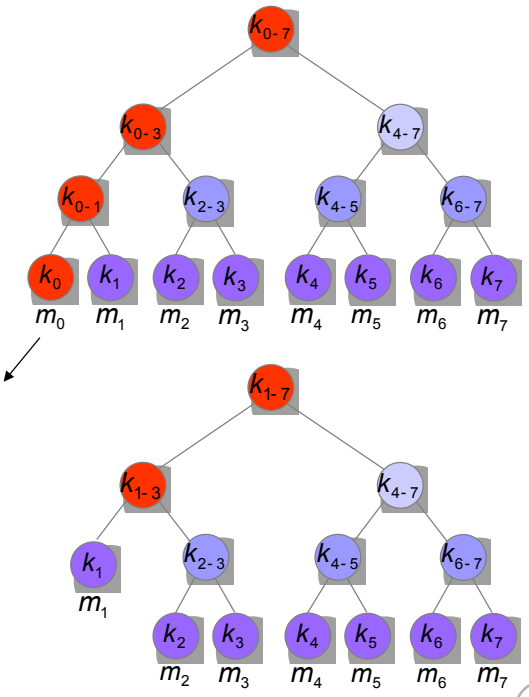
$\{k_{7-8}\}_{k_7} \rightarrow m_7$
$\{k_{7-8}\}_{k_8} \rightarrow m_8$
...
$\{k_{0-8}\}_{k_{0-3}}$



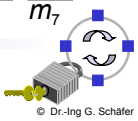


Centralized GKM (5) - LKH

- Member Leave
  - All affected keys on path of leaf node to the root have to be changed
  - GC encrypts altered keys with children keys in the tree and broadcasts them to the group

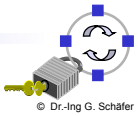


$$\{k'_{1-3}\}_{k_1} \rightarrow m_1$$
$$\{k'_{1-7}\}_{k_1} \rightarrow m_1$$
$$\{k'_{1-3}\}_{k_{2-3}} \rightarrow m_2, m_3$$
$$\{k'_{1-7}\}_{k_{2-3}} \rightarrow m_2, m_3$$
$$\{k'_{1-7}\}_{k_{4-7}} \rightarrow m_4, m_5, m_6, m_7$$



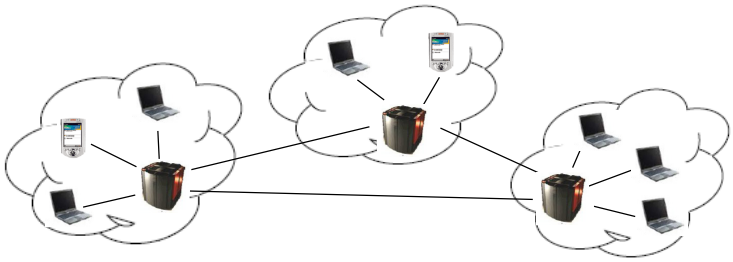
Centralized GKM (7) - LKH

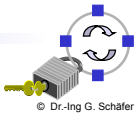
- Logical Key Hierarchy
  - ✓ O(log n) encryptions for complete re-keying
  - ✗ GC is potential single-point-of-failure and bottleneck
  - ✗ Not scalable
  - ✗ 1-affects-n and susceptible to churn



Decentralized GKM (1)

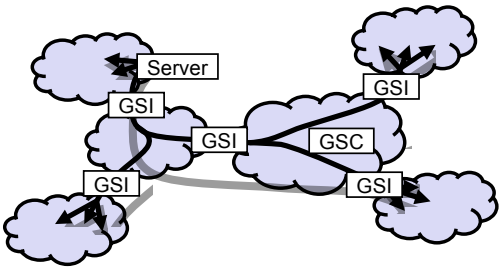
- Partitioning of the group into several subgroups with one subgroup controller (SGC) respectively
- Two possibilities:
  - Independent Subgroup Controllers
    - Example: Iolus
  - Above all Subgroup-Controllers (SGC) one central GC that generates TEKs and the SGCs distribute them to the members
    - Example: IGKMP

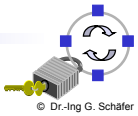




Decentralized GKM (2) - Iolus

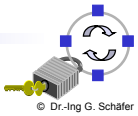
- Independent Sub Group Controllers – Iolus [Mit97]
  - Instead of one TEK for the whole group each subgroup has an own subgroup TEK
  - Subgroups are placed in a logical tree structure
  - Neighbor SGCs in the tree are connected with each other by secure unicast channels and act as proxies for their subgroup members
  - The SGC in the root subgroup is called Group Security Controller (GSC)
  - When a member of one subgroup sends a message to a member in another subgroup, the message has to be re-encrypted at the border of each subgroup, respectively
  - The authors suggest to reduce the overhead by encrypting each message with a random key that is included in the message and re-encrypted at subgroup borders





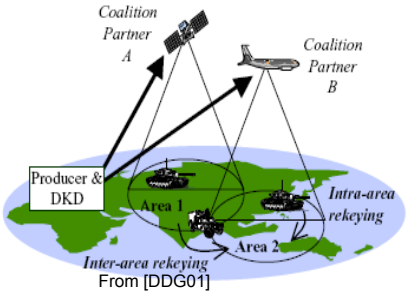
Decentralized GKM (3) - Iolus

- Iolus
  - ✓ No 1-affects-n
  - ✓ System is scalable
  - × Re-encryptions between subgroups
  - × GSC represents is still a potential single-point-of-failure for its group
  - × Data path is affected by failures as data has to be re-encrypted

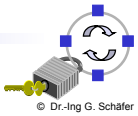


Decentralized GKM (4) – IGKMP

- Inter-Domain Group Key Management Protocol (IGKMP)
  - Domain Key Distributor (DKD) as central entity generates a TEK and distributes it to several Area Key Distributors (AKD) that are responsible for a specific area
  - AKDs maintains subgroups with a subgroup-wide known KEK for secure TEK distribution
  - Within AKDs several re-keying strategies can be deployed including re-keying based on pairwise keys, LKH and other centralized KM approaches
  - ✓ System is scalable
  - × DKD is potential single-point-of-failure
  - × 1-affects-n and susceptible to churn

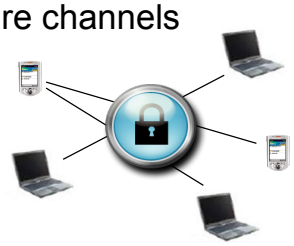


From [DDG01]

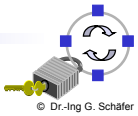


Contributory GKM (1)

- Distributed computation of the TEK
- No central GC, but higher computational effort than in centralized or decentralized schemes
- Two different approaches:
  - Key-generation by one member
    - Centralized Key Distribution by a dedicated member (CKD) and key-distribution via pre-established secure channels
  - N-party Diffie-Hellman key exchange
    - Ring-based: Group Diffie-Hellman (GDH)
    - Tree-Based: Tree-based Group Diffie-Hellman (TGDH) & Fully Decentralized Logical Key Hierarchy (FDLKH)

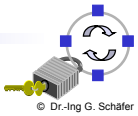


  - Group members with higher computational expenditure than the others are called *sponsors* in the following



Contributory GKM (2) – GDH

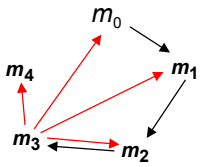
- Group Diffie-Hellman [STW00]
  - Contributory key generation based on N-party Diffie-Hellman key-exchange on a ring structure
  - All members have to agree upon a large prime *p* and a primitive root *g*
  - Each member *m* owns a secret key *s*
  - One member (e.g. the one with the highest index) is sponsor
  - Protocol has 4 stages
  - After completion a common secret is calculated and each member can verify that it contributed to the secret
  - The protocol itself **only** protects against passive attacks!



Contributory GKM (3) – GDH Example for 5 members

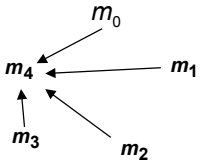
- Stage 1: Upflow

$$m_0 \rightarrow m_1 : g^{s_0} \bmod p$$
$$m_1 \rightarrow m_2 : g^{s_0 s_1} \bmod p$$
$$m_2 \rightarrow m_3 : g^{s_0 s_1 s_2} \bmod p$$



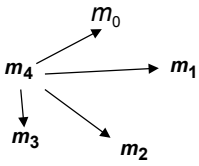
- Stage 2: Broadcast

$$m_3 \rightarrow m_0, m_1, m_2, m_4 : g^{s_0 s_1 s_2 s_3} \bmod p$$



- Stage 3: Response

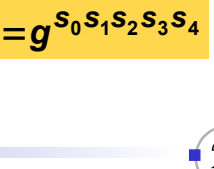
$$m_0 \rightarrow m_4 : g^{s_1 s_2 s_3} \bmod p$$
$$m_1 \rightarrow m_4 : g^{s_0 s_2 s_3} \bmod p$$
$$m_2 \rightarrow m_4 : g^{s_0 s_1 s_3} \bmod p$$
$$m_3 \rightarrow m_4 : g^{s_0 s_1 s_2} \bmod p$$

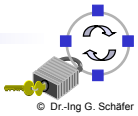


- Stage 4: Broadcast

$$m_4 \rightarrow m_0 : g^{s_1 s_2 s_3 s_4} \bmod p$$
$$m_4 \rightarrow m_1 : g^{s_0 s_2 s_3 s_4} \bmod p$$
$$m_4 \rightarrow m_2 : g^{s_0 s_1 s_3 s_4} \bmod p$$
$$m_4 \rightarrow m_3 : g^{s_0 s_1 s_2 s_4} \bmod p$$

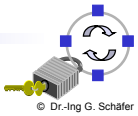
$$\Rightarrow \quad TEK = g^{s_0 s_1 s_2 s_3 s_4}$$





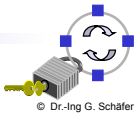
Contributory GKM (4) - GDH

- Member Join
  - A member m_{n+1} joins the group the protocol starts again in stage 2
  - To start the process the member m_n chooses a new secret s_n' and replaces the previous s_n by $s_n s_n$' in the TEK value, which is then sent to the group
- Member Leave
  - A member m_d leaves the group but not m_n:
  - The protocol starts again in stage 4
  - The member m_n broadcasts the messages of stage 3 with a new secret s_n'
  - When m_n is leaving the group:
  - m_{n-1} becomes the new sponsor and blinds the key with a new secret s_n' and the protocol continues as usual with stages 2, 3, and 4



Contributory GKM (5) - GDH

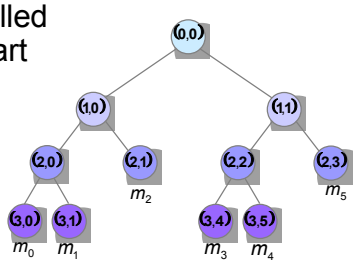
- Group Diffie-Hellman
  - ✓ Contributory key-generation
  - ✗ Setup time is growing linearly with the group size
  - ✗ High computation effort: O(n) exponentiations
  - ✗ 1-affects-n

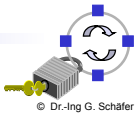


Contributory GKM (6) - TGDH

- Tree-based Group Diffie-Hellman (TGDH) [KPT04]
  - Every member contributes to the creation of a group-wide known key
  - Basic idea is the extension of the Diffie-Hellman key-exchange to a tree structure
  - Members maintain an identical binary key tree
    - All nodes agree on a primitive root *g* and a large prime *p*
    - Leaf nodes are assigned to group members
    - The root node represents the TEK
    - Nodes in between represent intermediate data for TEK computation
    - Every node owns a secret value *s* and a so-called blinded key *bk* that is computed like the first part of a DH key-exchange

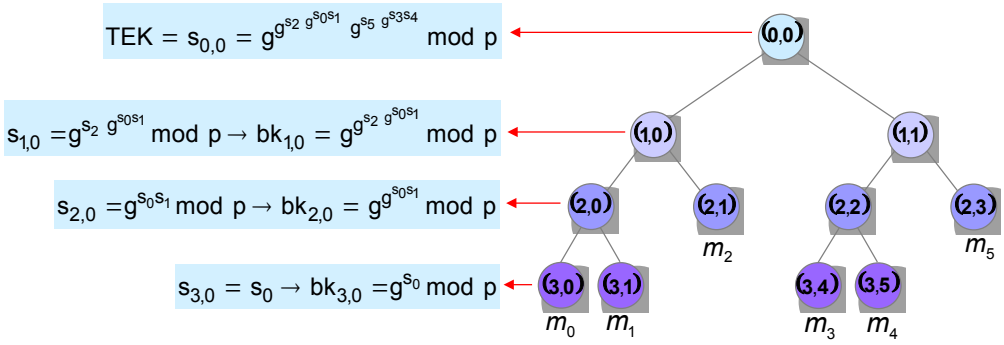
$$bk = g^s \bmod p$$

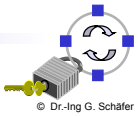




Contributory GKM (7) - TGDH

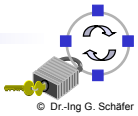
- TEK generation
  1. Every member computes its blinded key and broadcasts it to the group
  2. So-called *sponsors* (chosen according some selection strategy) compute the intermediate keys of the tree and broadcast them to the group
  3. Based on the blinded keys and the own secret values every node recursively computes the secret keys of all nodes from its path to the root node, which comprises the TEK





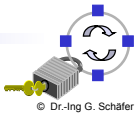
Contributory GKM (8) - TGDH

- Member Join
  - When a new member wants to join the group, it broadcasts its blinded key to the group
  - A sponsor has to be determined
    - Sponsor is a leaf node that gives up its position on level l and gets a new position on level l+1
    - To keep the tree balanced, it should be the rightmost leaf on the lowest possible level
  - On the position of the sponsor a new internal node is inserted, the children of this node become the sponsor and the new inserted member
  - The sponsor computes the new blinded keys on the path from its leaf node to the root, and broadcast them to the group, after this, all members compute the TEK again
- Member Leave
  - Sponsor is in this case the node belonging to the same father node as the leaving member
  - The father node is deleted and the remaining member takes its position
  - The sponsor computes the new blinded keys, broadcast them to the group, and the computation of the TEK starts again



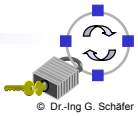
Contributory GKM (9) - TGDH

- Tree-based Group Diffie-Hellman
  - ✓ Contributory key-generation
  - ✓ Scalable
  - ✓ Computational effort: O(log n), for a balanced key-tree
  - ✗ High communication effort (broadcasts of all blinded keys)
  - ✗ All members need the same view on the tree
  - ✗ 1-affects-n



Peer authentication in group communication

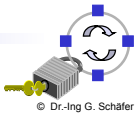
- If it is only to be verified that a message came from any peer within the group:
- Symmetric authentication keys can be derived in the same way as keys for confidentiality, e.g., by a group Diffie-Hellman
  - Security depends on integrity of all group members
  - Probability of security failure increases with group size
- To provide source authentication some kind of asymmetry is required so that
  - Source can generate authentication tags
  - All other peers can verify them, but not generate them
- Naïve method:
  - Source signs each packet with its private key
  - Peers verify each packet with the sources public key
  - Usually prohibitively high overhead in terms of compute power and message size (even if ECC was used!)



Alternative methods to perform source authentication

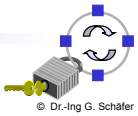
- Use bulk authentication
  - Verify groups of packets
  - Performs less asymmetric cryptographic operations
  - Less packet overhead
  - Complex if packets can get lost

- Use symmetric algorithms to emulate asymmetric behavior
  - By deploying combinations of symmetric MACs
  - By deploying a delayed key disclosure (TESLA)
    - Requires time synchronization
    - Adds time overhead by delayed signature checking

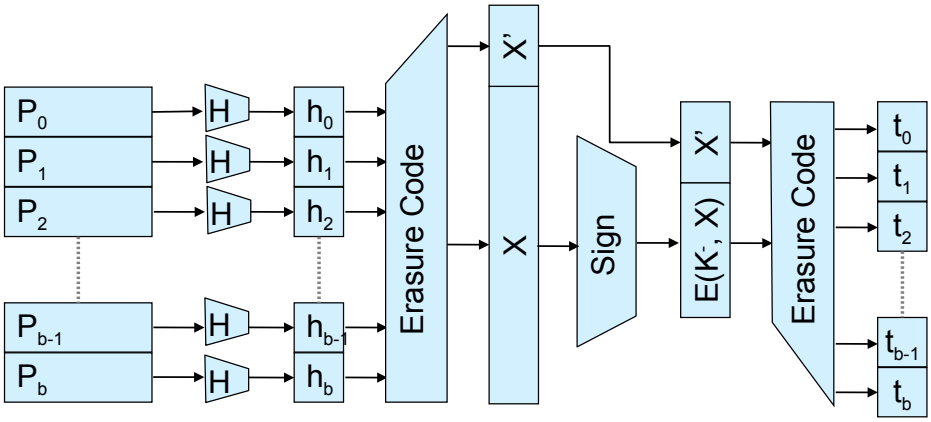


Using bulk authentication

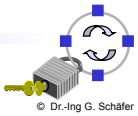
- Idea: Only sign and verify a cryptographic hash over many packets
- Problem: Some packets may get lost so the hash may not be restorable
- Sending data redundantly with the help of erasure codes reduces the problem
- [PaMo03] introduces a clever way to use this
  - Generates tags that are added to every packet
  - The information of the tags combined reveals:
    - The signature over a number of packets
    - Information to restore the signature information of some lost packets (generated by an erasure code)
  - As tags may also get lost a second erasure code is used to spread that information too
  - Robust method, but packets cannot be authenticated until enough of the signature can be restored!
  - May lead to significant delay



Using bulk authentication as described in [PaMo03]



- Plaintext blocks P are hashed and information is spread using an erasure code
- The hashes are signed and signature as well as the redundant information X' is again protected by a second erasure code
- Generated tags t are piggybacked to packets

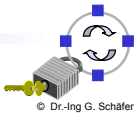


Probabilistic Authentication with symmetric MACs (I)

- Bulk authentication introduces non-negligible delay
- Idea: Create asymmetry by having many MACs [CGI+99]
- Source holds all keys, receivers hold a random subset

- Source generates MACs for all keys (e.g. 760 MACs)
- All MACs are truncated to 1 bit length
- The truncated MACs are attached to the packet and form the signature

- Each receiver verifies that all MACs are correct for the keys it knows (e.g. ⅒ of the pool)
- If MACs are correct: pass the packet to the user process

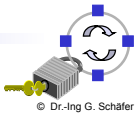


Probabilistic Authentication with symmetric MACs (II)

- Very low delay
- Despite many symmetric operations: significantly lower overhead than RSA and ElGamal at the source
- RSA verification can still be performed faster, if public exponents are chosen in a clever way (e.g. 3 or 65,537)

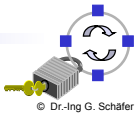
- Attacker has a significant chance of guessing a correct signature, if controlling some multicast receivers

- May be extended to support nodes leaving the system
- Also possible: distribute keys not randomly but by a predefined scheme to guarantee that an attacker has to compromise at least q nodes to generate valid signatures for any other node



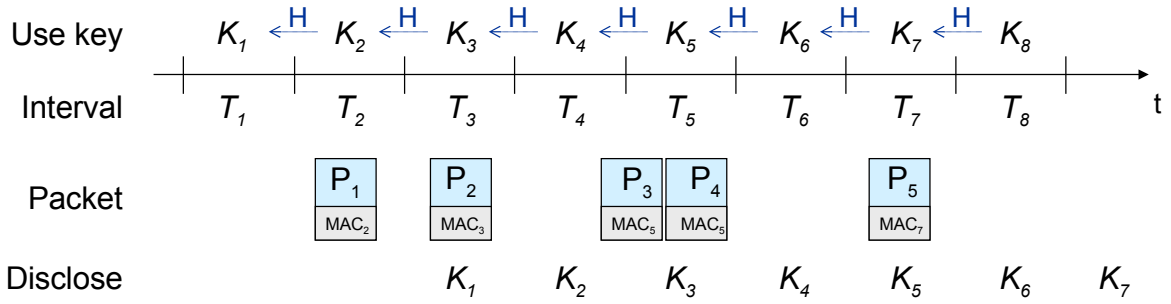
Authenticated Broadcast with μTESLA

- Basic idea for obtaining asymmetry:
  - Delayed key disclosure of a symmetric key
  - Requires more or less loosely synchronized clocks
- Original TESLA [PSC+05] and μTESLA:
  - TESLA stands for *Timed Efficient Stream Loss-tolerant Authentication*
  - Principal idea is inverse use of hash-chains for obtaining integrity keys (basically, a variation of the one-time password idea)
  - μTESLA is a minor variant of TESLA:
    - TESLA uses asymmetric digital signatures to authenticate initial keys, μTESLA uses a protocol based on symmetric cryptography (SNEP protocol)
    - μTESLA discloses the key only once per time interval, and only central base stations authenticate broadcast packets (storage of key chains)



TESLA Functions (1)

- Sender setup:
  - Choose length n of key chain and generate a random K_n
  - Compute and store hash key chain according to $K_{i-1} := H(K_i)$
- Broadcasting authenticated packets:
  - Time is divided in uniform length intervals T_i
  - In time interval T_i the sender authenticates packets with key K_i
  - The key K_i is disclosed in time interval $i + \delta$ (e.g. $\delta = 2$)

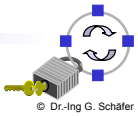


TESLA Functions (2)

- Provision of a new receiver M with an authenticated initial key:
  - M → GC: N_M
  - GC → M: $T_{GC} \mid K_i \mid T_i \mid T_{Int} \mid \delta \mid$ RC5-CBC($IK_{BS,M}$, $N_M \mid T_{BS} \mid K_i \mid T_i \mid T_{Int} \mid \delta$) with T_{Int} denoting the interval length
- Verification of authenticated broadcast packets:
  - Receiver must know current time, maximum clock drift and interval length
  - Packets must be stored with T_i until appropriate key is disclosed
  - Upon disclosure of the appropriate key K_i the authenticity of the packet can be checked
  - It is crucial to discard all packets that have been authenticated with an already disclosed key (requires loose time synchronization with appropriate value for δ)
- Authenticated broadcast by sensor nodes:
  - Sensor node sends a SNEP protected packet to base station
  - Base station sends an authenticated broadcast
  - Main reason: sensor nodes do not have enough memory for key chains

Secure Group Communication - Conclusion

- Group communication can be very difficult to achieve
- Two major reasons:
  - Large group size and dynamics induce tight scalability and efficiency objectives
  - Source authentication creates a non-negligible overhead
  - Boils down to: Three can keep a secret, if two of them are dead. (Benjamin Franklin)
- Choice of protection mechanism depends heavily on scenario
- No one fits all solution



References

[ZRM04] X. Zou, B. Ramamurthy, S. Magliveras. *Secure Group Communications Over Data Networks.* 2004

[RH03] S. Rafaeli and D. Hutchison. *A survey of key management for secure group communication*. ACM Computing Surveys, Volume 35, Issue 3, Pages 309-329. 2003.

[HM97a] H. Harney, C. Muckenhirn. *Group Key Management Protocol (GKMP) Specification*. IETF Request for Comments 2093. 1997.

[HM97b] H. Harney, C. Muckenhirn. *Group Key Management Protocol (GKMP) Architecture*. IETF Request for Comments 2094. 1997.

[WGL00] C. K. Wong, M. Gouda, S. S. Lam. *Secure group communications using key graphs*. IEEE/ACM Transactions on Networking Volume 8, Issue 1, Pages 16-30. 2000.

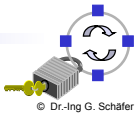
[CC89] G. Chiou, W. Chen. *Secure Broadcasting Using the Secure Lock*. IEEE Transactions on Software Engineering, vol. 15, no. 8, pp. 929-934, August, 1989.

[Mit97] S. Mittra, *Iolus. A Framework for Scalable Secure Multicasting*, Proceedings of ACM SIGCOMM'97, Pages 277-288, 1997.

[DDG01] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D, Towsley, S. Vasudevan, C. Zhang. *Secure group communications for wireless networks.* Military Communications Conference (MILCOM). Pages 113-117. 2001.

[STW00] M. Steiner, G. Tsudik, M. Waidner: Key Agreement in Dynamic Peer Groups. In: *IEEE Transactions on Parallel and Distributed Systems* 11 (2000), S. 769–780

[KPT00] Y. Kim, A. Perrig, G. Tsudik. *Simple and fault-tolerant key agreement for dynamic collaborative groups.* Proceedings of the 7th ACM conference on Computer and communications security. Pages 235-244. 2000.



References

[KPT04] Y. Kim, A. Perrig, G. Tsudik. *Tree-based group key agreement.* ACM Transactions on Information and System Security (TISSEC). Volume 7 Issue 1. Pages 60-96. 2004

[PaMo03] A. Pannetrat, R. Molva. *Efficient Multicast Packet Authentication.* Network and Distributed System Security Symposium (NDSS), 2003.

[CGI+99] R. Canetti, J. Garayt, G. Itkid, D. Micciancios, M. Naore, B. Pinkasll. *Multicast Security: A Taxonomy and Some Efficient Constructions*. IEEE Infocom. 1999.

[PSC+05] Perrig, A. ; Song, D. ; Canetti, R. ; Tygar, J. D. ; Briscoe, B.: *Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction*. 2005. – RFC 4082, IETF, Status: Standard, http://tools.ietf.org/html/rfc4082