HOKEY Key Distribution Exchange (KDE)
draft-ietf-hokey-key-mgm-03.txt

Yoshihiro Ohba

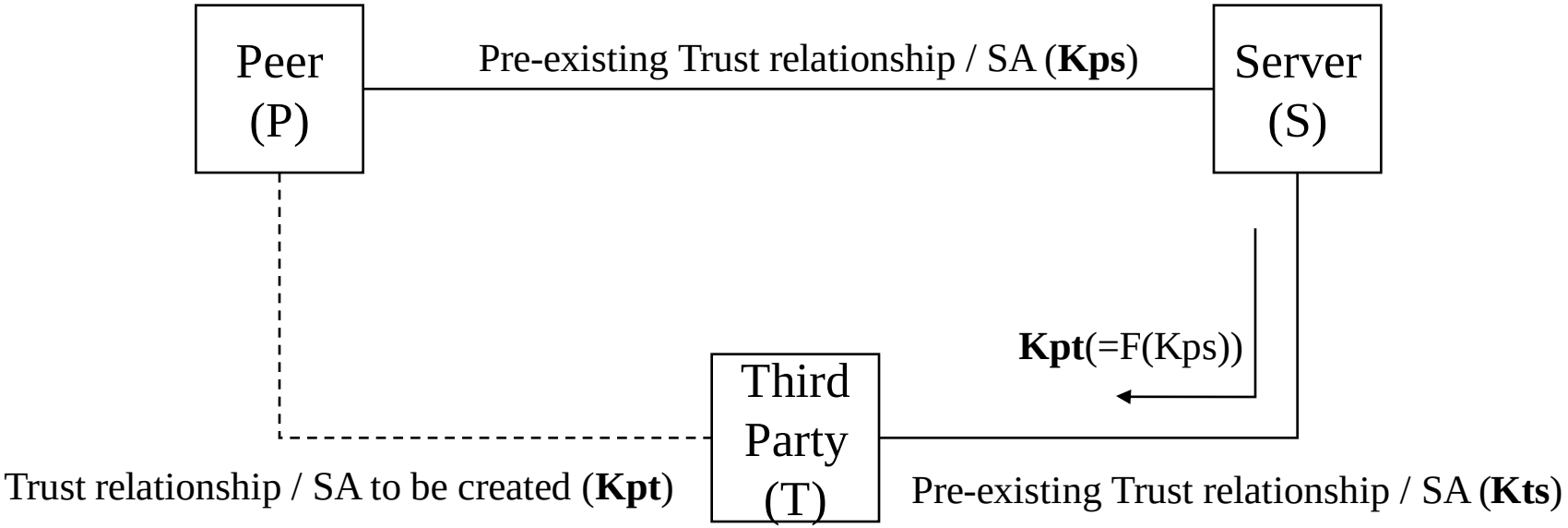
Madjid Nakhjiri

Status of KDE

- Submitted -02
- Two issues are addressed: (i) message format and (ii) hop-by-hop security support
- Other changes
  - Reduced number of use cases (USRK and DSUSRK)
  - Added description on automated key management for KIps and KCps
  - AAA and UDP transport in Appendix
  - No use of timestamp for freshness values
    - sequence number from peer + nonce from third-party
  - Added integrity protection to KDE message 3

  Many thanks to Rafa Marin Lopez and Chunqiang Li
- Additional issue: how KDE is used for bootstrapping ERX
- A new application of KDE : EAP-KDE (draft-ohba-eap-kde-01.txt)
  - A method-based low-latency re-authentication mechanism

Key Distribution Model



Kpt is used for dynamically establishing a trust relationship / SA between P and T

Key Distribution Exchange

Message Name (Parameters) [* means optional message/parameter]	P	T	S
KDE0* (TID,SID,DID*,KT) **(TID, SID, DID,KT) = (Third Party ID, Server ID, Domain ID, Key Type)**	←		
KDE1 (PRT) **PRT(Peer Request Token) =** **Int[KIps,(SEQps, PID, TID, SID, DID*, KT, KN_KIps)]**	→		
KDE2 (TRT) **TRT(Third Party Request Token) =** **Int[KIts, PID, TID, PRT]**			→
KDE3 (TOK) **TOK(Key Token) =** **Int[KIts, (Nt+1, PID, TID, DID*, KT, {Kpt, KN_Kpt, KL_Kpt}KCts, SAT)]**			←
KDE4 (SAT) **SAT(Server Authorization Token) =** **Int[KIps,(SEQps+1, PID, TID, SID, DID*, KN_Kpt, KL_Kpt, KN_KIps)]**	←		

Int [K, X] : X || MIC(K,X)
{X}K: X encrypted with K

SEQps: Sequence Number generated by P
KT: Key Type
KN_X : Key Name for key X
KL_X: Key Lifetime for key X

KIts (or IK): Key Integrity Key
KCts (or CK): Key Encryption Key

Issue 27 - Protocol Format

- The format should be generic enough to be carried in various transport protocols
  - ASN.1 is used
- The default encoding scheme is PER (Packed Encoding Rules)
  - Each transport protocol can specify other encoding scheme

Issue 28 – Encryption optional

- Hop-by-hop security is supported
  - By allowing null encryption and integrity algorithms
  - “In this case, underlying transport protocol security such as IPsec and (D)TLS MUST be used instead.”
  - “The use of hop-by-hop security implies that an intermediary on each hop can access the distributed key material. Hence the use of hop-by-hop security SHOULD be limited to an environment where an intermediary is trusted not to use the distributed key material”

Automated Key Management

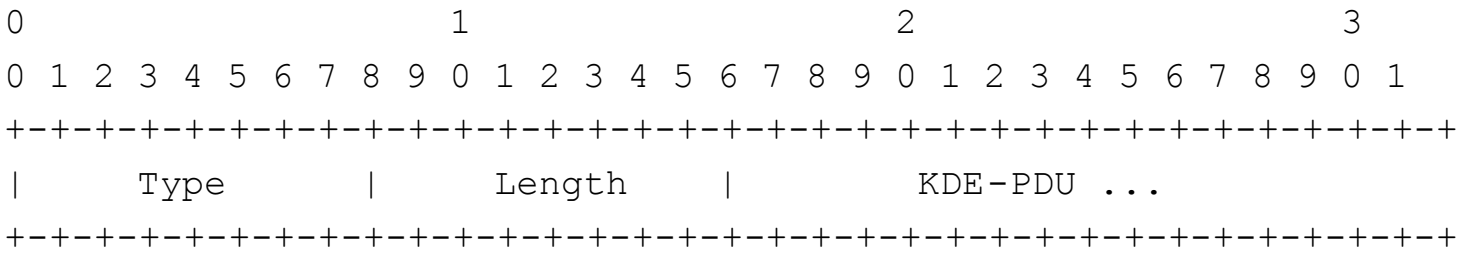
- KIts and KCts require automated key management [RFC4107] because of N^2 keys
  - [key-mgt-03 has incorrect statement to be fixed]
- Kerberos [RFC4120] MAY be used as an automated key management protocol for distributing KIts and KCts.
  - If there is no direct trust relationship between the third-party and the server, then inter-realm Kerberos MAY be used to create a direct trust relationship between the third-party and the server from a chain of trust relationships.

Algorithm changes

- Timestamp is replaced with two freshness values to provide replay protection
  - Reason: timestamp is not easy to maintain
  - Sequence numbers generated by peer and maintained by peer and server
    - provide anti-replay for KDE messages 1, 2 and 4.
  - Nonces generated by third-party
    - provide anti-replay for KDE message 3
- Added integrity protection to KDE message 3 in addition to encryption

UDP and AAA transport for KDE

- UDP transport – new well-known port
- AAA transport – new RADIUS attribute:



How to carry KDE in ERX

- Two alternatives:

Alternative 1: Bootstrapping ERP using KDE over ERP/AAA

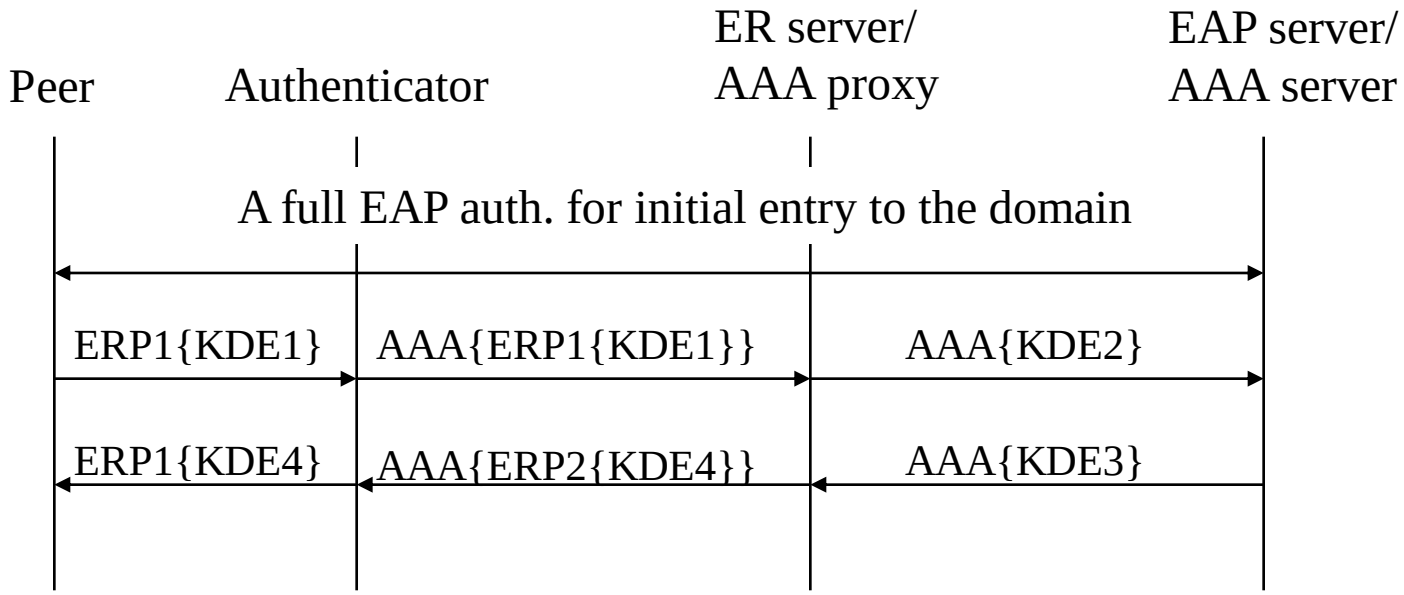
Used with explicit ERP bootstrapping

Alternative 2: Bootstrapping ERP using KDE over UDP

Used with implicit ERP bootstrapping

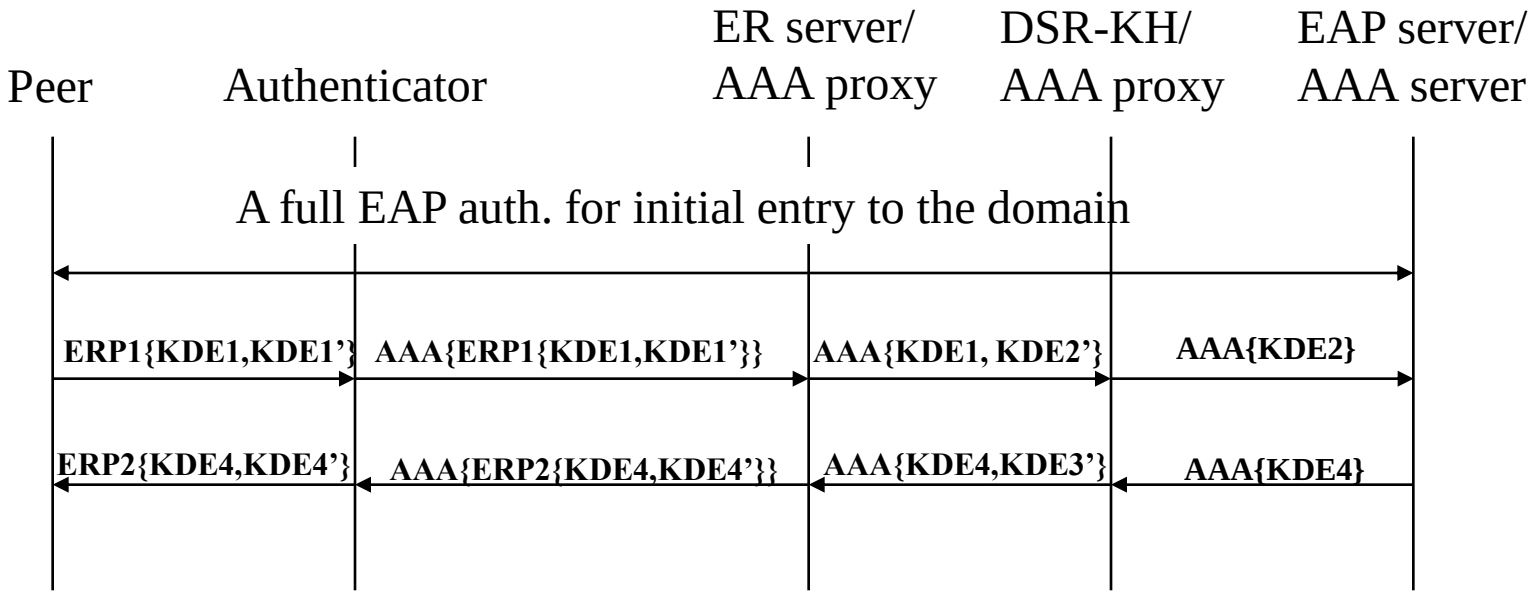
- In both alternatives, when the peer initially enters the visited (or home) domain, it performs a full EAP authentication with the home EAP server through an authenticator
  - Once the peer attaches to the domain through the authenticator, it discovers an ER server in the visited or home domain using DNS or DHCP
- Both alternatives can work with hop-by-hop security

Alternative 1: Bootstrapping ERP using KDE over ERP/AAA (single-key dist.)



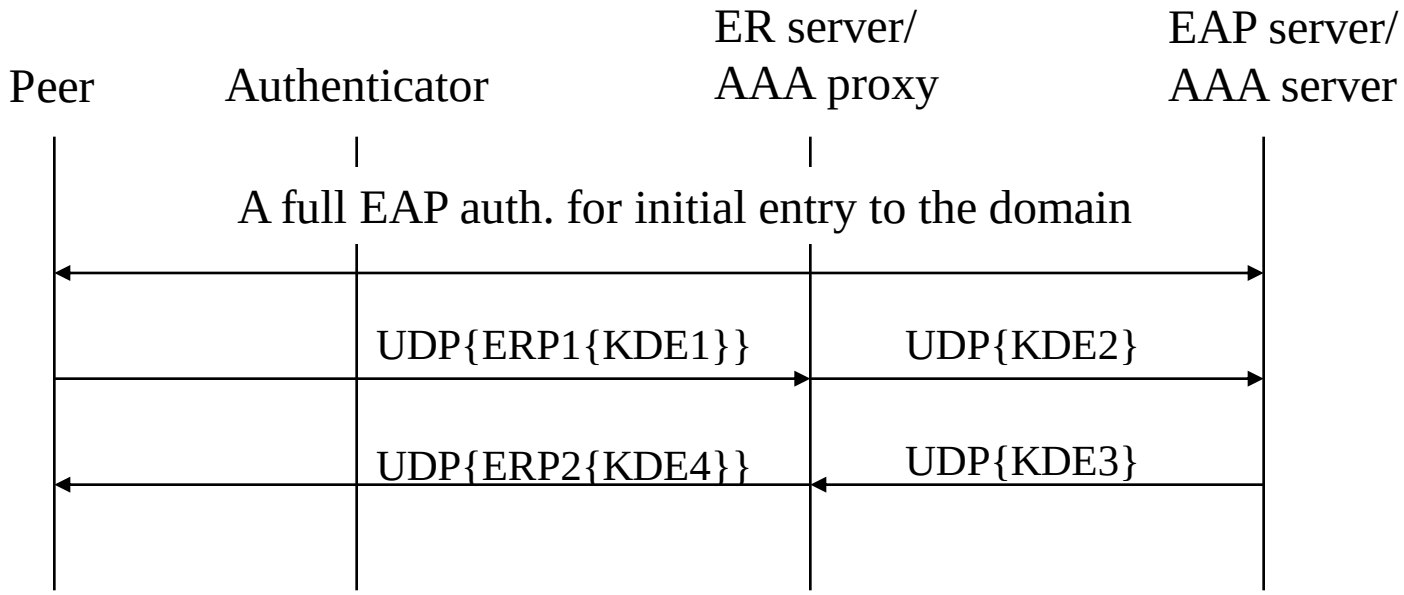
Key Type = 0 (DSRK) [with DSRK && DSR-KH == ER server]
Key Type = 1 (rRK) [without DSRK]

Alternative 1: Bootstrapping ERP using KDE over ERP/AAA (dual-key dist.)



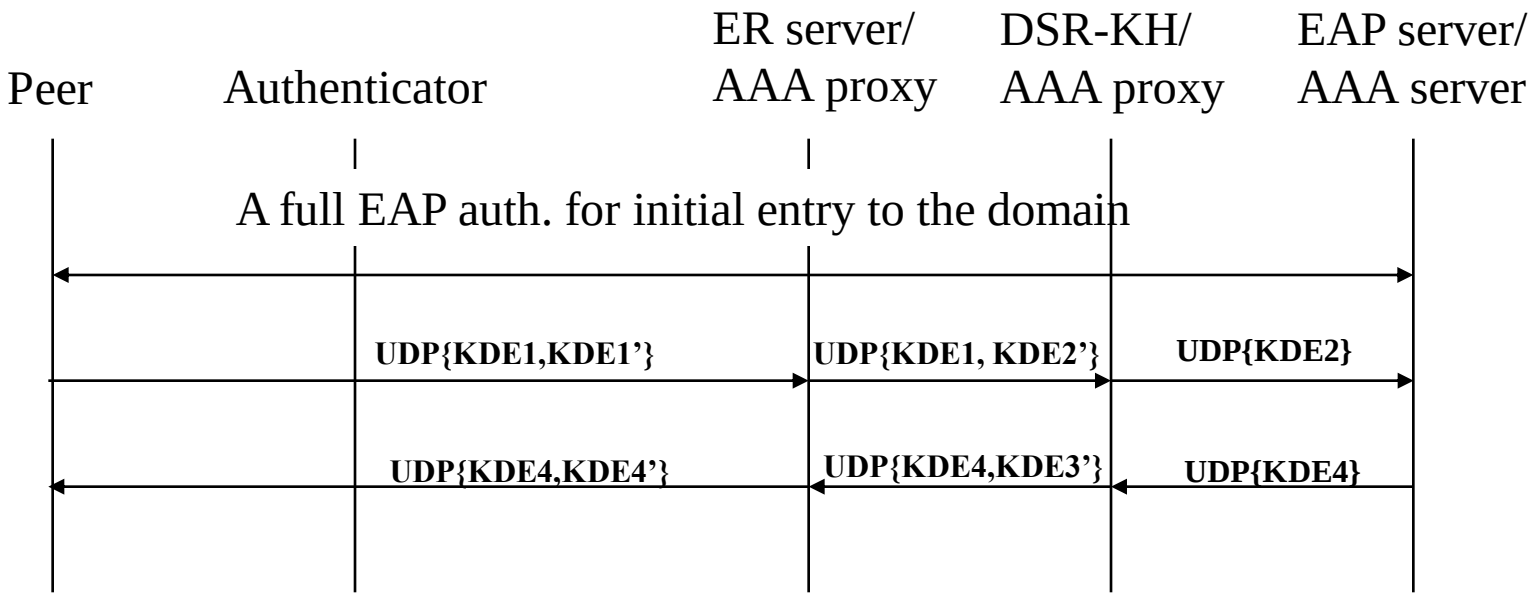
Key Type = 0 (DSRK) for KDE1, KDE2, KDE3 and KDE4
Key Type = 1 (rRK) for KDE1', KDE2', KDE3' and KDE4'

Alternative 2: Bootstrapping ERP using KDE over UDP (single-key dist.)



Key Type = 0 (DSRK) [with DSRK && DSR-KH == ER server]
Key Type = 1 (rRK) [without DSRK]

Alternative 2: Bootstrapping ERP using KDE over UDP (dual-key dist.)



Key Type = 0 (DSRK) for KDE1, KDE2, KDE3 and KDE4
Key Type = 1 (rRK) for KDE1', KDE2', KDE3' and KDE4'