PKINIT Freshness Extension

Andrei Popov, Microsoft Corp.

A Client Can Pre-Generate PKINIT Authenticators

- In the Diffie-Hellman exchange, the client uses its private key only to sign the AuthPack specified in Section 3.2.1 of [RFC4556].
- A client can generate requests with future times in the PKAuthenticator, and then send those requests at the future times.
- Unless the time is outside the validity period of the client's certificate, the KDC will validate it and return a TGT the client can use without possessing the private key.
- A client performing PKINIT with the Diffie-Hellman key exchange does not prove current possession of the private key being used for authentication. It proves only prior use of that key.

Freshness Token

- Ensuring that the client has current possession of the private key requires that the signed PKAuthenticator data include a Freshness Token: information that the client cannot forge.
- Freshness Tokens are opaque to the client; only KDCs need to parse and validate them.
- Multiple ways to generate Freshness Tokens exist; Token structure is KDC implementation-specific.
- E.g. a Freshness Token could comprise a random number and a timestamp. The random number would ensure uniqueness of each Token, while the timestamp would bind the Token to a specific period of valid time. The Freshness Token could be guarded against tampering by the use of an HMAC.
- PA-PK-AS-KDCTOKEN structure specifies the Freshness Token:

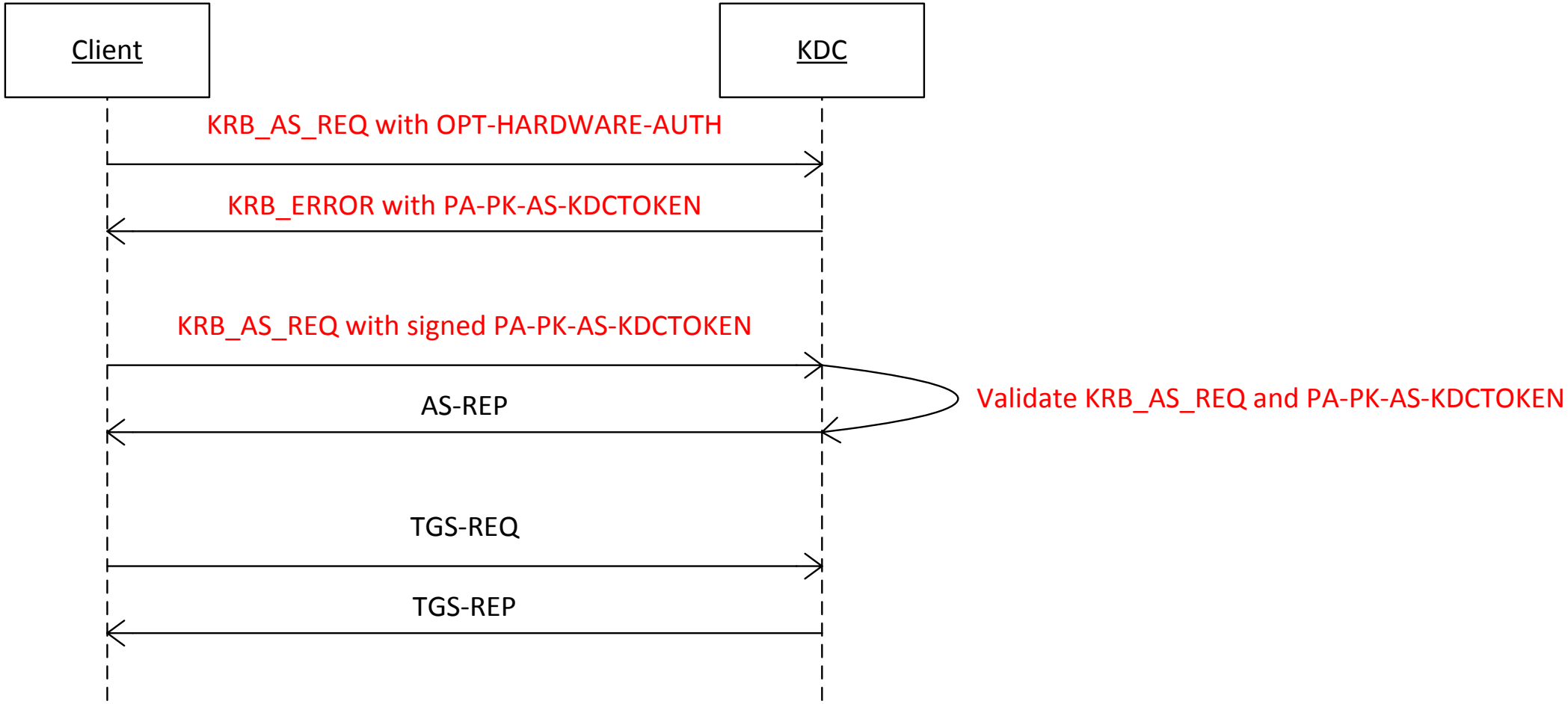
PA-PK-AS-KDCTOKEN ::= OCTET STRING

Extended PKAuthenticator

- The PKAuthenticator structure specified in Section 3.2.1 [RFC4556] is extended to include Freshness Token as follows:

```
PKAuthenticator ::= SEQUENCE {
   cusec        [0] INTEGER (0..999999),
   ctime        [1] KerberosTime,
   nonce         [2] INTEGER (0..4294967295),
   paChecksum   [3] OCTET STRING OPTIONAL,
   kdcToken     [4] PA-PK-AS-KDCTOKEN OPTIONAL,
            -- MUST be present if sent by KDC
   ...
}
```

Message Flow With Freshness Token



Security and Interoperability Considerations

- The Freshness Token MUST be time-bound and MUST include information that the client cannot forge.
- The Freshness Token SHOULD be integrity-protected by the issuing KDC to prevent tampering and to allow a different KDC in the same realm to validate the Token.
- Since the client treats the KDC provided data blob as opaque, changing the contents will not impact existing clients. Thus changes in the Freshness Token structure do not impact client interoperability.

Links And Contact Information

- PKINIT Freshness Extension I-D: http://tools.ietf.org/html/draft-short-pkinit-freshness-00

- Michiko Short michikos@microsoft.com
- Andrei Popov andreipo@microsoft.com