Auxiliary Exchange in IKEv2 Protocol

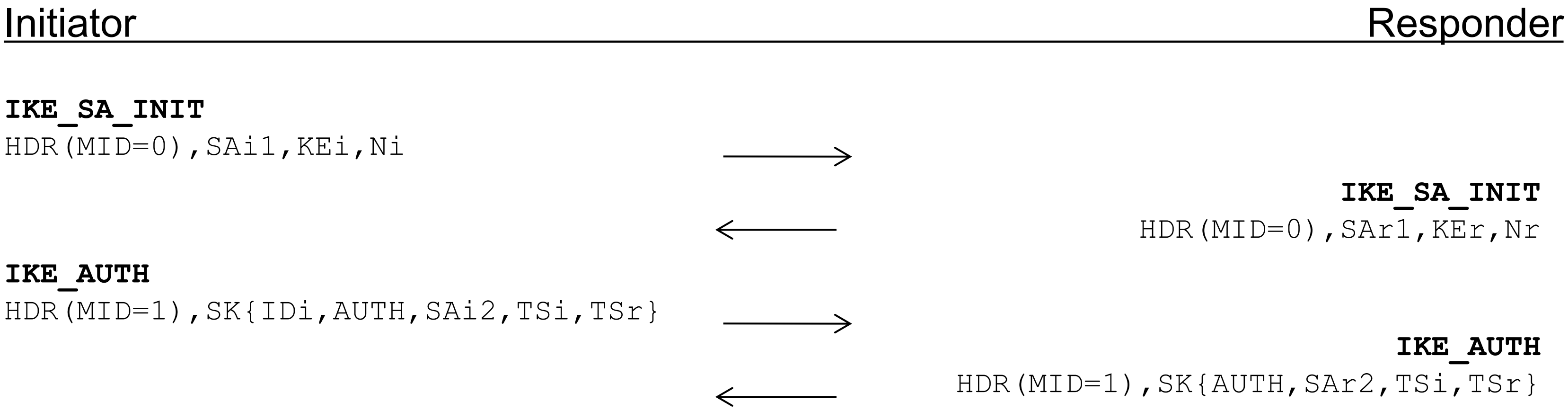
draft-smyslov-ipsecme-ikev2-aux

Valery Smyslov

svan@elvis.ru

IETF 101

Initial IKEv2 Exchanges



Initiator

Responder

IKE_SA_INIT

`HDR(MID=0),SAi1,KEi,Ni` →

IKE_SA_INIT

← `HDR(MID=0),SAr1,KEr,Nr`

IKE_AUTH

`HDR(MID=1),SK{IDi,AUTH,SAi2,TSi,TSr}` →

IKE_AUTH

← `HDR(MID=1),SK{AUTH,SAr2,TSi,TSr}`

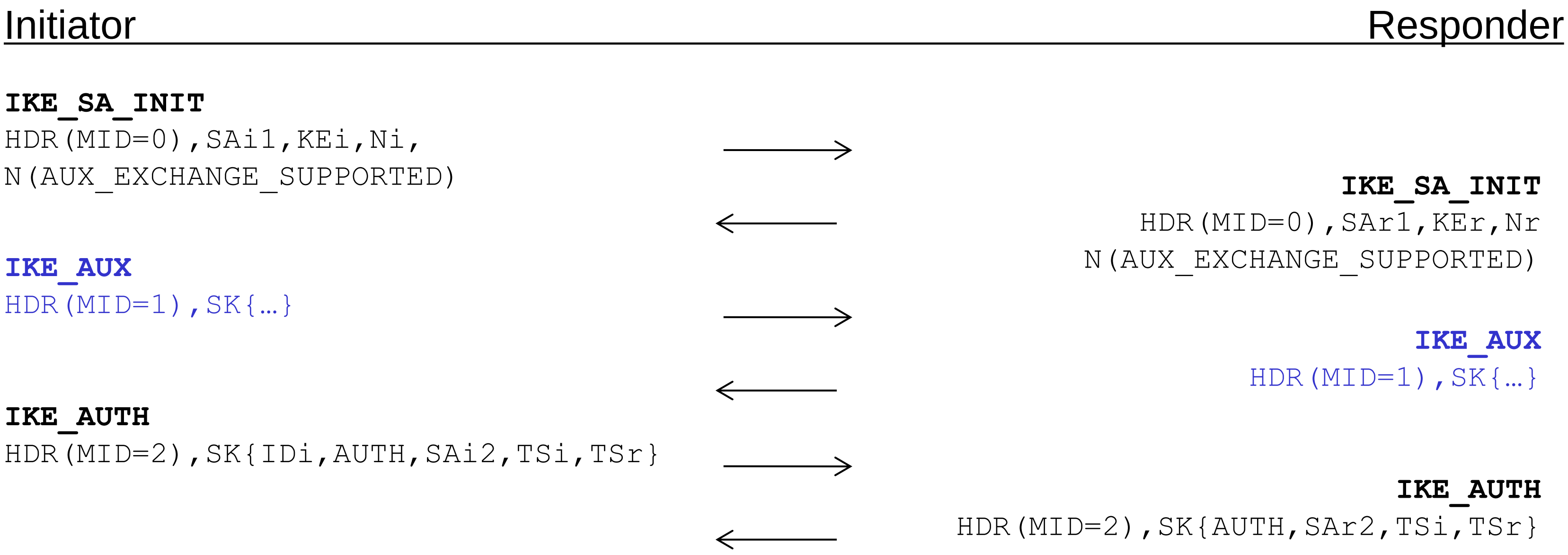
- IKE_SA_INIT messages are usually less then MTU – no IP fragmentation
- IKE_AUTH messages can be large, so IP fragmentation is possible
  - IP fragmentation interacts badly with some middleboxes like NAT and firewalls
- RFC7383 defines a way to avoid IP fragmentation by fragmenting messages in IKE
  - can only be used on encrypted messages, so IKE_SA_INIT is out of scope

The Problem

- Some recent proposals for IKEv2 protocol may lead to the situation when IKE_SA_INIT messages grow above MTU
  - Quantum Safe Key Exchange (QSKE) proposal defines additional Key Exchange payloads to be included into IKE_SA_INIT
  - something else?
- As result IKE_SA_INIT messages become subject for IP fragmentation with all aftermath
- Adding IKE fragmentation to IKE_SA_INIT is cumbersome and may lead to vulnerability to DoS attacks
  - IKE_SA_INIT messages have no protection, so an attacker who is able to see them and to inject bogus fragments can easily mount a reassembly queue poisoning attack

Proposed Solution

New auxiliary (IKE_AUX) exchange is added between IKE_SA_INIT and IKE_AUTH:



```
Initiator                                                  Responder
--------------------------------------------------------------------
IKE_SA_INIT
HDR(MID=0),SAi1,KEi,Ni,
N(AUX_EXCHANGE_SUPPORTED)          -->
                                                      IKE_SA_INIT
                                   <--     HDR(MID=0),SAr1,KEr,Nr
                                        N(AUX_EXCHANGE_SUPPORTED)
IKE_AUX
HDR(MID=1),SK{…}                   -->
                                                          IKE_AUX
                                   <--               HDR(MID=1),SK{…}
IKE_AUTH
HDR(MID=2),SK{IDi,AUTH,SAi2,TSi,TSr}  -->
                                                         IKE_AUTH
                                   <--   HDR(MID=2),SK{AUTH,SAr2,TSi,TSr}
```

IKE_AUX Exchange

- New large payloads are placed in IKE_AUX, keeping IKE_SA_INIT messages small
- IKE_AUX messages are encrypted and MACed, so standard IKE fragmentation can be used
- IKE_AUX messages are authenticated by including their ICVs in signature calculation in IKE_AUTH:

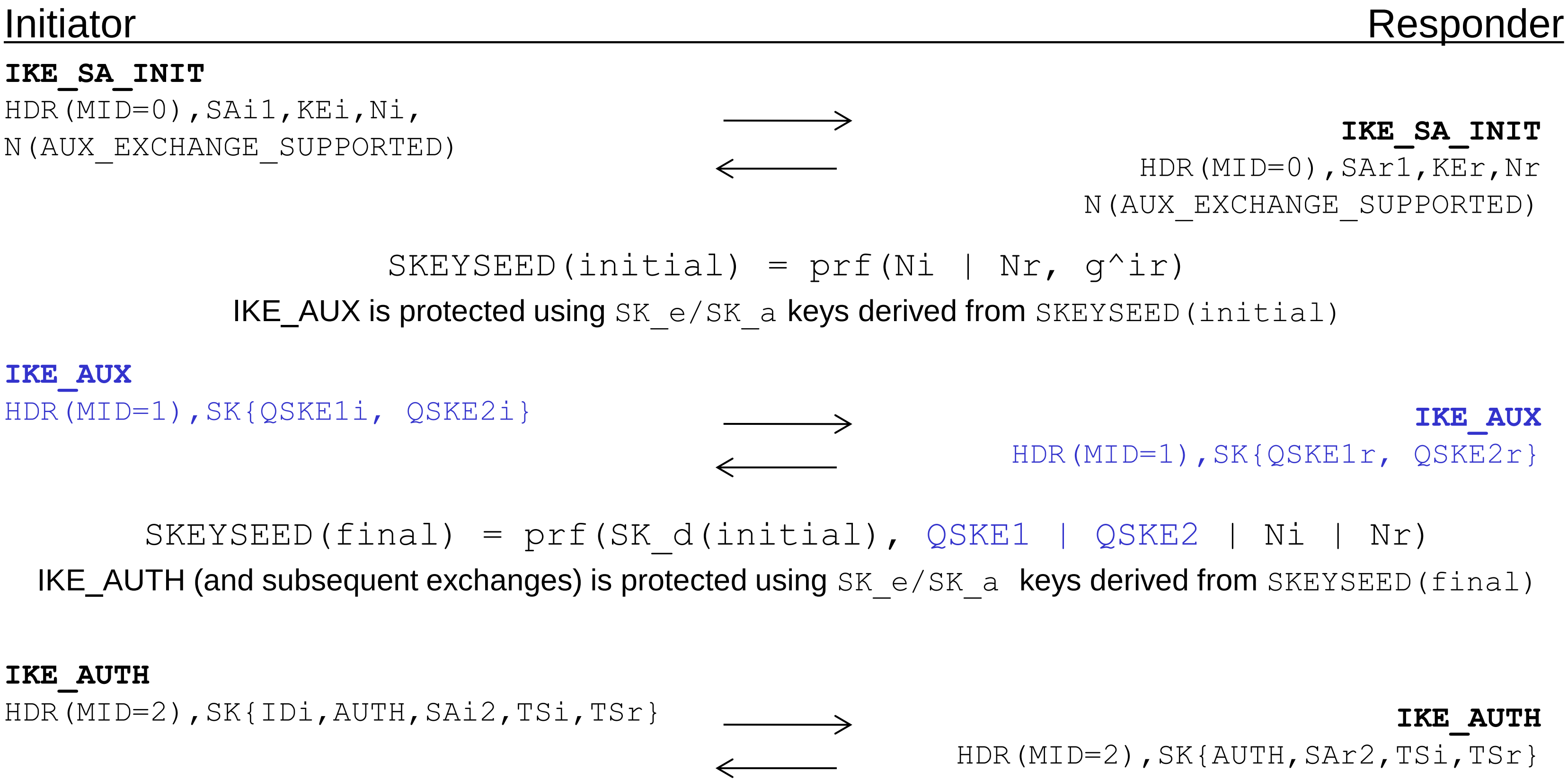
```
InitiatorSignedOctets = RealMessage1 | AUX_I | NonceRData | MACedIDForI
AUX_I = ICV_INIT_1 [ | ICV_INIT_2 [ | ICV_INIT_3 ... ]]
ResponderSignedOctets = RealMessage2 | AUX_R | NonceIData | MACedIDForR
AUX_R = ICV_RESP_1 [ | ICV_RESP_2 [ | ICV_RESP_3 ... ]]
```

Using IKE_AUX with QSKE

- Additional QSKE payload(s) are transferred using IKE_AUX
- IKE_AUX messages are protected using keys derived from key exchange performed in IKE_SA_INIT
  - IKE_SA_INIT messages must always contain KE payload
    - this KE payload may either contain classic (EC)DH public key or public key for some QSKE method, but it must be small enough not to cause IP fragmentation
- Keys for IKE_AUTH and for subsequent exchanges can be calculated as modification of standard IKE SA re-keying:

```
SKEYSEED(final) = prf(SK_d(initial), QSKE1 [| QSKE2 [| QSKE3 …]] | Ni | Nr)
```

Keys in case of QSKE (example)



Initiator Responder

IKE_SA_INIT
HDR(MID=0),SAi1,KEi,Ni,
N(AUX_EXCHANGE_SUPPORTED) →

← **IKE_SA_INIT**
HDR(MID=0),SAr1,KEr,Nr
N(AUX_EXCHANGE_SUPPORTED)

SKEYSEED(initial) = prf(Ni | Nr, g^ir)

IKE_AUX is protected using SK_e/SK_a keys derived from SKEYSEED(initial)

IKE_AUX
HDR(MID=1),SK{QSKE1i, QSKE2i} →

← **IKE_AUX**
HDR(MID=1),SK{QSKE1r, QSKE2r}

SKEYSEED(final) = prf(SK_d(initial), QSKE1 | QSKE2 | Ni | Nr)

IKE_AUTH (and subsequent exchanges) is protected using SK_e/SK_a keys derived from SKEYSEED(final)

IKE_AUTH
HDR(MID=2),SK{IDi,AUTH,SAi2,TSi,TSr} →

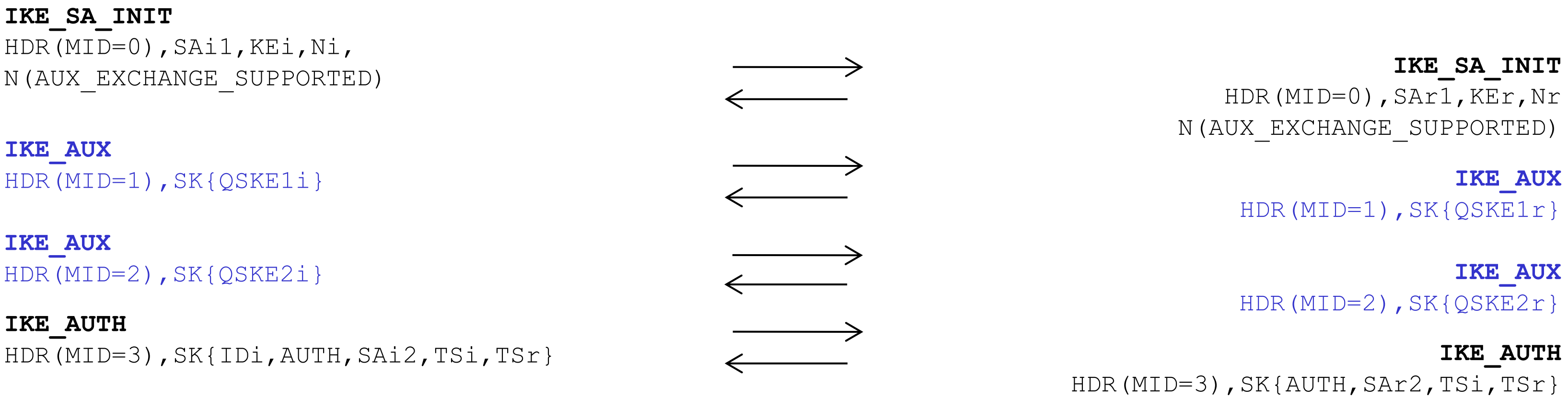
← **IKE_AUTH**
HDR(MID=2),SK{AUTH,SAr2,TSi,TSr}

IKE_AUX Properties

- Complexity
  - a simple standard IKEv2 exchange
  - minimal influence on IKE_SA_INIT and IKE_AUTH
    - IKE_AUTH would start with Message ID > 1
  - uses standard IKEv2 fragmentation
  - some (small) impact on IKE state machine
  - modification of AUTH payload calculation
- Modularity
  - IKE_AUX is not tied to QSKE and can be used in other situations when large amount of data needs to be transferred prior to IKE_AUTH
- Security
  - DoS attacks surface in case of fragmentation is smaller than it would be if fragmentation were done in unprotected IKE_SA_INIT

IKE_AUX Properties (continued)

- Reliability
  - if IKE_AUX is used with QSKE and several QSKE methods are employed, then each QSKE method can optionally be done in a separate IKE_AUX exchange:



IKE_SA_INIT
HDR(MID=0),SAi1,KEi,Ni,
N(AUX_EXCHANGE_SUPPORTED)

IKE_SA_INIT
HDR(MID=0),SAr1,KEr,Nr
N(AUX_EXCHANGE_SUPPORTED)

IKE_AUX
HDR(MID=1),SK{QSKE1i}

IKE_AUX
HDR(MID=1),SK{QSKE1r}

IKE_AUX
HDR(MID=2),SK{QSKE2i}

IKE_AUX
HDR(MID=2),SK{QSKE2r}

IKE_AUTH
HDR(MID=3),SK{IDi,AUTH,SAi2,TSi,TSr}

IKE_AUTH
HDR(MID=3),SK{AUTH,SAr2,TSi,TSr}

  - This would increase probability of IKE SA successful setup on congested or lossy networks in case IKE_AUX messages got fragmented using IKE fragmentation.

- Performance
  - adds extra round trip(s)
  - with QSKE re-calculation of SKEYSEED and derived keys is required

Thanks

- Comments? Questions?
- More details in the draft
- Please review and send feedback to author
- WG adoption?