Revised Cookie Processing in IKEv2

`draft-smyslov-ipsecme-ikev2-cookie-revised`

Valery Smyslov

svan@elvis.ru

IETF 115

Using Cookies in IKEv2

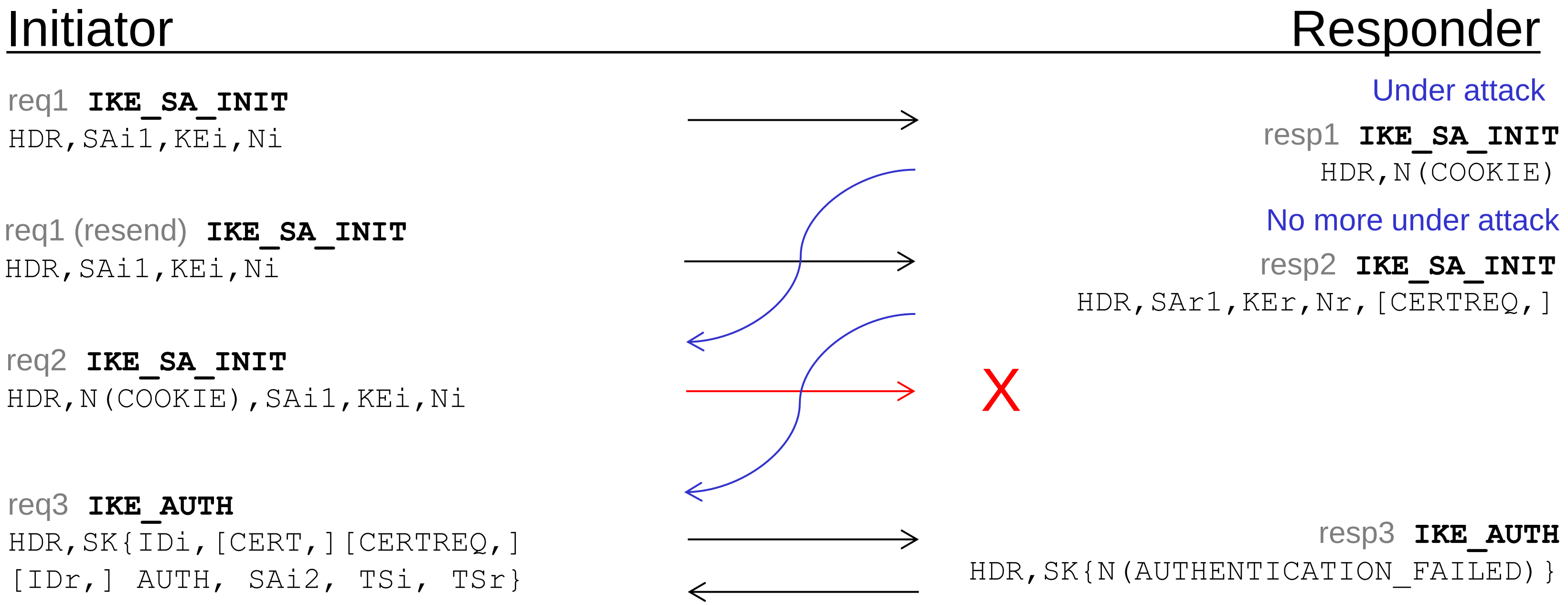
```
Initiator                                                      Responder
------------------------------------------------------------------------
req1  IKE_SA_INIT
HDR,SAi1,KEi,Ni                  -------->
                                                      resp1  IKE_SA_INIT
                                 <--------                 HDR,N(COOKIE)

req2  IKE_SA_INIT
HDR,N(COOKIE),SAi1,KEi,Ni        -------->
                                                      resp2  IKE_SA_INIT
                                 <--------  HDR,SAr1,KEr,Nr,[CERTREQ,]

req3  IKE_AUTH
HDR,SK{IDi,[CERT,][CERTREQ,]     -------->
[IDr,] AUTH, SAi2, TSi, TSr}                             resp3  IKE_AUTH
                                 <--------           HDR,SK{IDr,[CERT,]
                                                   AUTH, SAi2, TSi, TSr}
```

According to RFC 7296, the most recent `IKE_SA_INIT` request is included in the `AUTH` payload calculation in the `IKE_AUTH` exchange. In this example it is **`req2`** for both the initiator and the responder.

Problem Scenario 1



Initiator — Responder

req1 **IKE_SA_INIT**
`HDR,SAi1,KEi,Ni`

Under attack

resp1 **IKE_SA_INIT**
`HDR,N(COOKIE)`

req1 (resend) **IKE_SA_INIT**
`HDR,SAi1,KEi,Ni`

No more under attack

resp2 **IKE_SA_INIT**
`HDR,SAr1,KEr,Nr,[CERTREQ,]`

req2 **IKE_SA_INIT**
`HDR,N(COOKIE),SAi1,KEi,Ni`

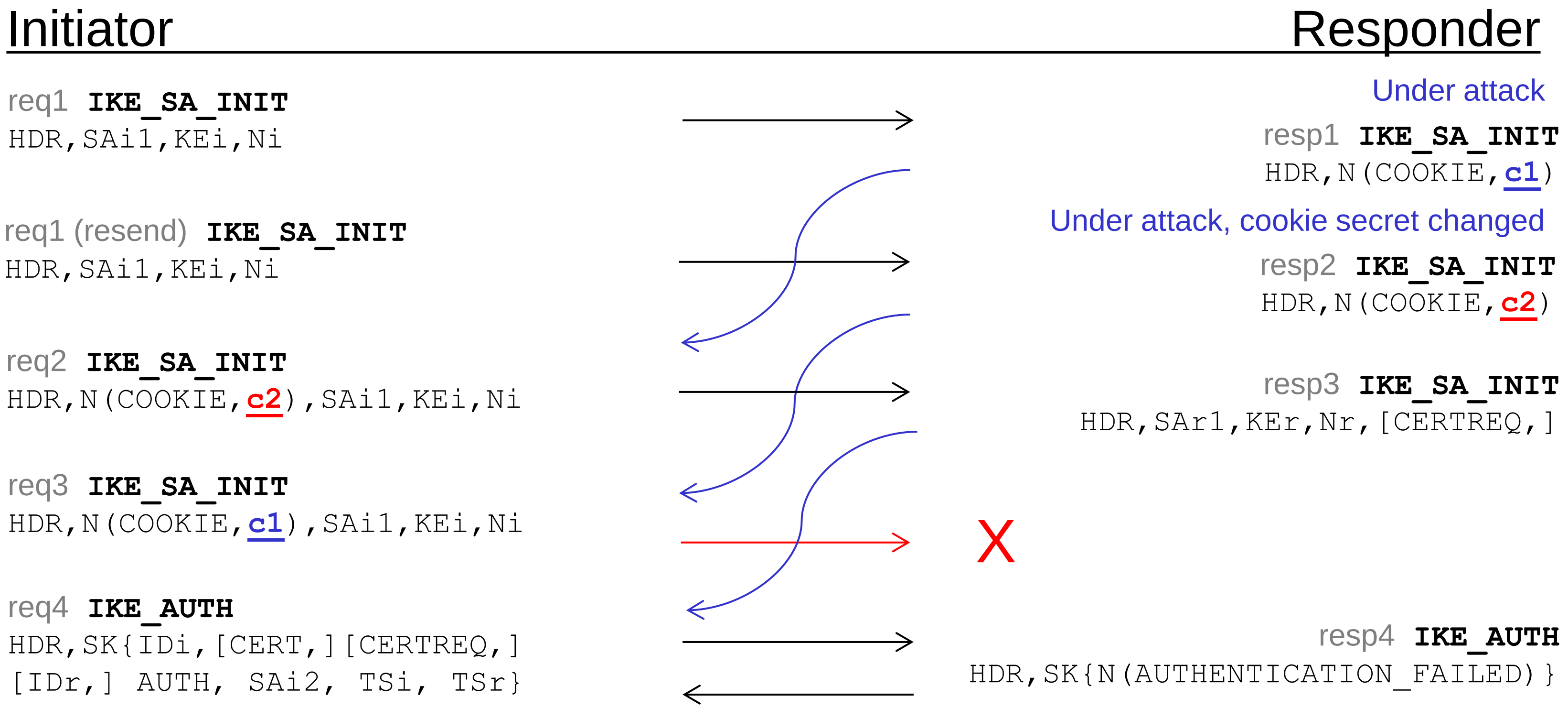
X

req3 **IKE_AUTH**
`HDR,SK{IDi,[CERT,][CERTREQ,]`
`[IDr,] AUTH, SAi2, TSi, TSr}`

resp3 **IKE_AUTH**
`HDR,SK{N(AUTHENTICATION_FAILED)}`

The most recent `IKE_SA_INIT` request sent by the initiator is **`req2`**, while the responder only received **`req1`**, so authentication would fail.

Problem Scenario 2



Initiator | Responder

Under attack

req1 **`IKE_SA_INIT`**
`HDR,SAi1,KEi,Ni`

resp1 **`IKE_SA_INIT`**
`HDR,N(COOKIE,`**`c1`**`)`

Under attack, cookie secret changed

req1 (resend) **`IKE_SA_INIT`**
`HDR,SAi1,KEi,Ni`

resp2 **`IKE_SA_INIT`**
`HDR,N(COOKIE,`**`c2`**`)`

req2 **`IKE_SA_INIT`**
`HDR,N(COOKIE,`**`c2`**`),SAi1,KEi,Ni`

resp3 **`IKE_SA_INIT`**
`HDR,SAr1,KEr,Nr,[CERTREQ,]`

req3 **`IKE_SA_INIT`**
`HDR,N(COOKIE,`**`c1`**`),SAi1,KEi,Ni`

X

req4 **`IKE_AUTH`**
`HDR,SK{IDi,[CERT,][CERTREQ,]`
`[IDr,] AUTH, SAi2, TSi, TSr}`

resp4 **`IKE_AUTH`**
`HDR,SK{N(AUTHENTICATION_FAILED)}`

The most recent `IKE_SA_INIT` request sent by the initiator is **`req3`**, while the responder only received **`req2`**, so authentication would fail.

Source of the Problem

- The `IKE_SA_INIT` request can be sent several times with different content depending on the responder state
- If there is high probability of packets loss and reordering, then peers may complete the `IKE_SA_INIT` exchange having different views on what was the most recently sent `IKE_SA_INIT` request
- This request message is used in calculation of the `AUTH` payload. If peers use different messages for the calculation, the authentication would erroneously fail

Severity of the Problem

- There are some preconditions for this problem to become noticeable
  - network with high probability of packet loss and delay
  - relatively frequent change of responder state (either changing cookie generation secret or changing responder’s mind whether it is under attack)
- It might be extremely rare in normal conditions, but in stress tests we observed that up to 5% of SAs failed due to this problem
  - customers wonder why authentication sometimes fails with proper credentials
- This is a protocol flaw

Proposed Solution Overview

- Revise cookie processing by excluding Notify payload containing cookie (if present) from the `IKE_SA_INIT` request message when calculating the `AUTH` payload content
  - the cookie is already verified by the responder, no need to include it into the data to be authenticated
- For backward compatibility make the revised processing negotiable

Negotiation

Initiator | Responder

req1 **IKE_SA_INIT**
HDR,SAi1,KEi,Ni $\longrightarrow$

$\longleftarrow$ resp1 **IKE_SA_INIT**
HDR,N(COOKIE,c),N(REVISED_COOKIE)

req2 **IKE_SA_INIT**
HDR,N(REVISED_COOKIE,c),SAi1,KEi,Ni $\longrightarrow$

$\longleftarrow$ resp2 **IKE_SA_INIT**
HDR,SAr1,KEr,Nr,[CERTREQ,]

req3 **IKE_AUTH**
HDR,SK{IDi,[CERT,][CERTREQ,]
[IDr,] AUTH, SAi2, TSi, TSr} $\longrightarrow$

$\longleftarrow$ resp3 **IKE_AUTH**
HDR,SK{IDr,[CERT,]
AUTH, SAi2, TSi, TSr}

Responder includes a new notification REVISED_COOKIE in the message containing COOKIE notification. If initiator also supports this extension, it returns cookie in this notification instead of COOKIE notification

Revised Cookie Processing

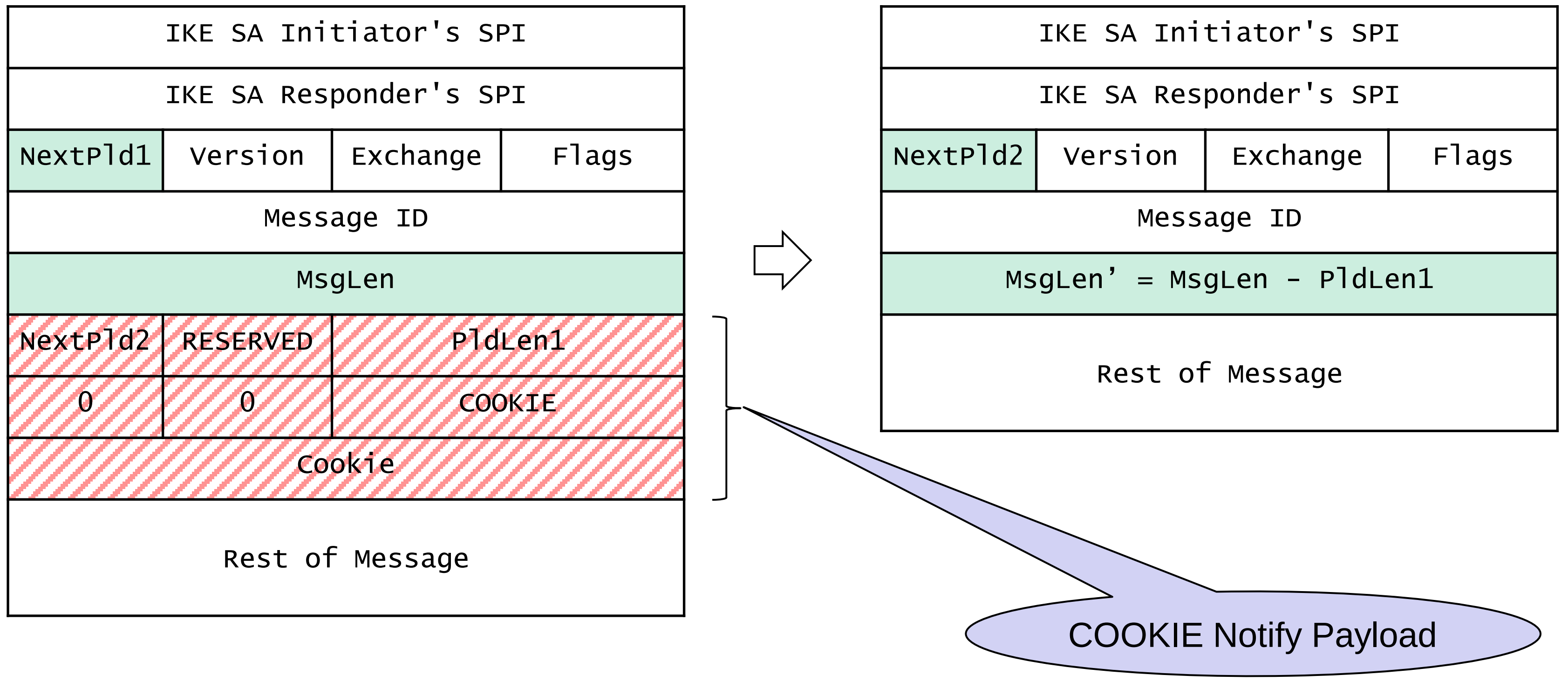
- If peers agreed upon using this extension then the cookie processing is changed
  - no changes in cookie anti-clogging function – responder still sends stateless cookie and when it is returned back by initiator it MUST be verified before message is processed

  According to RFC7296 initiator's `AUTH` payload is calculated by signing (or MAC'ing) the blob:

```
InitiatorSignedOctets = RealMessage1 | NonceRData | MACedIDForI
```

  - if `COOKIE` Notify payload is present in RealMessage1 (i.e. in `IKE_SA_INIT` request message), then for the purpose of `AUTH` payload calculation the message is modified as if it contained no this payload

Adjusting IKE_SA_INIT Request for AUTH Payload Calculation



Thanks

- Comments? Questions?
- Is this problem worth to address?
- Is the suggested approach reasonable?
- WG adoption?