

Current thinking on DKIM2

Richard Clayton

20 March 2026

Message-Instance is a tag value field

- v=n
 - originator uses v=1
 - thereafter use v=2, v=3 etc
- r=base64ofJSON
 - the body and header recipes
 - the base64 encoding simplifies parsing
- h=hashname1:headerhash1:bodyhash1
 - these are the hashes of the header fields & body
 - C an add “,hashname2:headerhash2:bodyhash2” for dexterity

Current thinking on the hashes field

- clayton-08 says hashes should be in a JSON structure
 - and then makes a pigs ear of defining it
- No need for this to be JSON : no parsing issues (thank you Levine)
- So h=algorithm:bodyhash:headerhash
 - then repeat triples if multiple hash functions used
 - this seems like a no brainer
- BTW: there is no versioning scheme for Message-Instance
 - but versioning never works in the real world so do we care ?

Current thinking on signing headers

- draft-clayton says sign every header (except Received, Return-Path, X-*, DKIM-Signature and ARC-*)
- Easy to understand and people cannot make poor choices
- Also keeps header field size down
- Opponents appear to think that it will be hard to keep track of the headers that are added by their systems
- draft-clayton says headers concatenated in natural order (from the top of the message) – alternative is bottom up (like in recipes)

Current thinking on recipes

- clayton-08 defines a JSON structure for body & header recipes
 - and then makes a pigs ear of defining it
- Simplifies parsing to keep as (valid) JSON
- Should use numbers rather than strings for the numbers
 - viz: change the JSON spec in draft-clayton-08
- JSON has UTF8 strings (OK since JSON is then base64 encoded)
- Open question as to whether to number header fields from bottom (makes copy ranges more logical) or from the top (which may be easier to code)

DKIM2-Signature is a tag value field

- i= first signature 1, ascending thereafter
- v= highest Message-Instance signed
- n= a nonce value (local meaning only)
- t= a timestamp
- m=base64ofJSON MAIL FROM / RCPT TO values
- f=donotexplode,donotmodify,exploded,feedback
- d= the signing domain
- s=selector1:algorithm1:signature1 (plus more triples as needed)

NB: we re-use DKIM1 keys and selector system

Current thinking on the signature field

- clayton-08 says signatures should be in a JSON structure
 - and then makes a pigs ear of defining it
- No need for this to be JSON : no parsing issues (thank you Levine)
- So s=selector:algorithm:signature
 - then repeat triples if multiple hash functions used
 - this seems like a no brainer & fixes how to sign issues (where we need a null signature value)
- Note that failing to publish the hash value means we could use ed25519 in a simpler(?) way

Current thinking on what is signed

- draft-clayton-08 says sign concatenation of
 - all Message-Instance header fields
 - then all existing DKIM2-Signature fields
 - finally an incomplete version of the header field being made
- Suggestion to interleave these (in “creation order”)
 - tidy design to do this, but a mess to implement
- draft-clayton-08 fails to sign selector (and algorithm)
 - change to s= field will fix this

Current thinking on the SMTP parameters field

- clayton-08 says put MAIL FROM & RCPT TO values into JSON
 - JSON is poorly specified, but multiple RT is elegantly handled
- Issue is that MF/RT can contain characters that interact with the tag-value format specification (so either need to encode or use an RFC5321 parser to pick out the tag-value fields)
- Some people want “human-readable” form (but your nearest AI system can read base64 for you, even if you don’t write a program)
- Putting the fields into a separate header field (to fix the parsing) seems like a lot of work...

Current thinking on result status values

- DKIM1 results were OK, PERMFAIL, TEMPFAIL
- RFC8601 (“Authentication-Results”) has “none”, “pass”, “fail”, “policy”, “neutral”, “temperror”, “permerror”
- Should we go along with 8601 in the DKIM2 specification ? and should we write it to encourage systems to use consistent text explanations of failures (copying results into 5xx/4xx strings)

Current thinking on DKIM keys

- Although we will re-use DKIM1 keys we will deprecate (and ignore) various exotic fields such as h=, s= and n=
- For those at the back: h= suggests hashes to use
 - which doesn't seem useful, we've done that step
- s= says what you should use the key for
 - so if not s=email you should ignore – which is pointless
- Unclear if deprecating k= is useful or not
- The flag t=y turns out to be almost meaningless (sad for Google)
- t=s is useless (we quietly abolished individual signers)