Discovery for BRSKI

draft-eckert-anima-brski-discovery-03

ANIMA WG IETF120

Toerless Eckert, Futurewei USA (tte@cs.fau.de)

Esko Dijk (esko.dijk@iotconsultancy.nl)

06/26/2024

Changes sind IETF118 (not on agenda for 119)

- Editorial: Tables too wide for txt, but good for html, discussed with rfced etc.. Will keep until rfc Editor to avoid manual reformatting for every draft version.

- Intermedia ANIMA Meeting April 2024 introduced/reconfirmed choice of CORE-LF encoding
  - Single big open item

- Updated doc accordingly
  - Unified use of “Variation String” across all discovery mechanism

- TODO:
  - Run through text, try to simplify, maybe move examples into appendix so core text is shorter
  - Check update constrained-voucher/constrained-proxy to be in sync (constrained voucher seems OK.)
  - Goal to ask for WGLC in Dublin.

Goal Reminder

- Enables BRSKI “client” to discover instances for a particular BRSKI-service
- Client wants to find an instance supporting a variation of BRSKI supported by Client.
  - Pledge looking for Registrar, which:
    - Speaks RFC8995 (instead of BRSKI-PRM)
    - Uses CoAP instead of TLS (aka: “constrained-voucher” version of BRSKI)
    - Uses CMP enrollment (instead of EST)
- Deployment (Client and Servers) want to use a particular discovery protocol/mechanisms – independent of which variation is needed
  - Currently: GRASP, DNS-SD or CORE-LF (Core Link Format)
- Discoverable variations should be extensible purely through IANA Registries (not document updates)
  - Registry to be applicable across any current/future discovery mechanisms
  - Discoverable BRSKI-service also extensible
- Discovery needs to work across stateful/stateless BRSKI proxies
  - Proxy enabled to proxy service instance announcements for any variation – without understanding the variation. Note: Proxies are transport-proxies, they never need to understand variations, which all are about payload format/options.

Proxy functionalities

- Discovery wants to make it possible that proxies will be able to proxy any current/future BRSKI variations – and announce accordingly to pledges.

- RFC8995 BRSKI discovery with GRASP
  - Registrar announces itself with objective name AN_join_registrar
    - Proxies look for this objective name
  - Proxy announces itself with objective name AN_Proxy
    - Pledges look for this objective name

- Clarification / explanation in BRSKI discovery
  - **Registrars** that support “stateful” mode (such as in RFC8995), **can always act as Proxies**
    - without additional BRSKI code
    - Registrars see no difference whether Proxy or Pledge connects
  - But Registrars then need to announce themselves as stateful proxy – AN_Proxy in addition to announcing themselves as Registrars to Proxis with. – AN_join_registrar

- The same logic can be applied to constrained BRSKI

Constrained BRSKI considerations

- CORE-LF – Core Link Format:
  Discovery mechanism using CoAP – primary for cBRSKI
    - GRASP: AN_Proxy ~ CORE-LF: rt=brski.jp
    - GRASP: AN_join_registrar ~ CORE-LF: rt=brski.rs
    - GRASP: — ~ CORE-LF: rt=brski.rjpy
- draft-ietf-anima-constrained-voucher
    - Introduces cBRSKI even without proxy
    - Needs to only introduce rt=brski.jp
- draft-ietf-anima-constrained-proxy
    - Introduces proxies.
    - Needs to introduce rt=brski.rs indicating registrars to be discovered by proxies
    - New (not in RFC8995)
        - Stateless proxy mode Registrar->Proxy: rt=brski.rjpy
    - Need to also update constrained-proxy text for this

BRSKI Discovery Registry elements

Contexts

Variation Type / Choices

Variation Strings

BRSKI Discovery part 1: Definition of contexts

Context	Applicable Variation Types	Discovery Mechanism	Service Name(s) / Transport	Reference(s)
BRSKI	mode vformat enroll	GRASP	"AN_join_registrar" / "AN_Proxy" with IPPROTO_TCP	[RFC8995]
		DNS-SD	"brski-registrar" / "brski-proxy" with TCP	[RFC8995]
cBRSKI	mode vformat enroll	CORE-LF	rt=brski.jp with coaps	[I-D.ietf-anima-constrained-voucher]
			rt=brski.rs / rt=brski.rjpy with coaps	[I-D.ietf-anima-constrained-join-proxy]
		DNS-SD	"brski-proxy" / "brski-registrar" / "brski-registrar-rpy" with UDP	[THIS-RFC]
		GRASP	"AN_join_registrar" / "AN_join_registrar_rjp" / "AN_Proxy" with IPPROTO_UDP	[THIS-RFC]
BRSKI-PLEDGE	mode vformat enroll	DNS-SD	"brski-pledge" with TCP	[THIS-RFC]

Variations, Types and Choices

- **Variation Type**: Some aspect of a BRSKI protocol session for which there are different alternatives
  - **mode**, **vformat**, **enroll**
- **Variation Type Choice**: value for a variation type:
  - **mode** High-level operation of the BRSKI connection
    - **rrm**: registrar responder mode – as in RFC8995
    - **prm**: pledge responder – as in BRSKI-PRM
  - **vformat** Voucher Format used
    - **cms**: as in RFC8995
    - **cose**: as in cBRSKI
    - **jose**: as in BRSKI-PRM
  - **enroll** Enrollment protocol
    - **est**: as in RFC8995 or RFC9148 (CoAP-EST)
    - **cmp**: as speced in BRSKI-AE (lightweight CMP)
    - **scep**: reserved for RFC8894

- **Variation**: combination of one variation type choice for each variation type
  - This describes one supported complete BRSKI session supported by the responder (e.g.: Registrar)
  - RFC8995: **rrm**, **cms**, **est**
- **Extensible**: IANA registry will allow to add new types and values easily.

BRSKI Variation Type and Values – by context Excerpt

Context	Variation Type	Variation Type Choice	Reference	Flags	Note(s)
BRSKI, cBRSKI	mode	rrm	[RFC8995] ThisRFC	Dflt	Registrar Responder Mode the mode specified in [RFC8995]
		prm	ThisRFC		Pledge Responder Mode [I-D.ietf-anima-brski-prm]
BRSKI	vformat	cms	[RFC8368] ThisRFC	Dflt	CMS-signed JSON Voucher
		cose	ThisRFC		CBOR with COSE signature
cBRSKI		cose	ThisRFC	Dflt	CBOR with COSE signature [I-D.ietf-anima-constrained-voucher]
		cms	[RFC8368] ThisRFC		CMS-signed JSON Voucher

How to bring Variations into “arbitrary” Service Discovery mechanisms ?

- Capabilities of different service discovery mechanisms vary widely
  - DNS-SD, GRASP, CORE-LF – and we do not want to redo our approach when another one is needed
- Parsing of encoding may be more/less complex
  - Especially pledges are a target that requires “as easy as possible” protocols
  - Slim/limited firmware impossible to fix easily when only rare problems occur.

- Solution: Variation String
  - Variation is expressed ACROSS ALL SERVICE DISCOVERY MECHANISMS as a simple string, concatenating the variation-type-values for all (known) variation types.
    - RFC8995 BRSKI: rrm-cms-est
    - Just need to define for each service discovery mechanism a place where to put this variations string
  - But future extensibility.
    - Assume we would have started without considering enrollment options (“Nobody every needs alternatives to EST”).
    - If we later added enrol = est, cms, scep, … Old server implementations would not know to have “-est” in variation
  - Solution: Variation String not do include DEFAULT (first defined) Variation Type Choices
    - RFC8995 BRSKI: “” (all defaults)
  - Reliable Approach:
    - Variation String could be unambiguously be figured out by ever implementer … But mistakes would be problematic
    - Much easer to put the Variation Strings ALSO into the Registry – even though they are “derived” information from prior Registry information.

BRSKI Variation Strings

Context	Reference	Variation String	Variations	Explanations / Notes
BRSKI	[RFC8995]	"" / "EST-TLS"	rrm cms est	Note 1
	[I-D.ietf-anima-brski-ae]	cmp	rrm cms cmp	
	[I-D.ietf-anima-brski-prm]	prm-jose	prm jose est	
cBRSKI	[I-D.ietf-anima-constrained-voucher]	"" / "rrm-cose"	rrm cose est	

- Note 1: The Variation String "EST-TLS" is equivalent to the Variation String "" and is required and only permitted for the AN_join_registrar objective value in GRASP for backward compatibility with RFC8995, where it is used for this variation. Note that AN_proxy uses "".

- Only registering combinations actually used by reference specs!
  ... Future entries may not require RFC but could use other specs
  ... Up for us when deciding on Registry criterias. Currently
  spec required + expert review

BRSKI Discovery
Encoding examples

GRASP Encoding

```
[M_FLOOD, 12340815, h'fe8000000000000000000000000001', 180000,
    [["AN_Proxy", 4, 1, "", RFC8995 Variation
     [O_IPv6_LOCATOR,
     h'fe8000000000000000000000000001', IPPROTO_TCP, 4443],
     ["AN_Proxy", 4, 1, "", cBRSKI rrm,cose,est  Variation
     [O_IPv6_LOCATOR,
     h'fe8000000000000000000000000001', IPPROTO_UDP, 4684]]
]

[M_FLOOD, 42310815, h'fe8000000000000000000000000001', 180000,
    [["AN_Proxy", 4, 1, "prm",
     [O_IPv6_LOCATOR,
     h'fe8000000000000000000000000001', IPPROTO_TCP, 44000]]
]
```

Figure 5: GRASP example with two different processes

pty DNS-SD TXT Strings may be problematic, so we use the replacement strings “EST-TLS” and “rrm-cose”

DNS-SD: variation strings as “Keys” for TXT RR

```
                _brski-registrar._tcp.local
                IN PTR  0200:0000:7400._brski-registrar._tcp.local
0200:0000:7400._brski-registrar._tcp.local
                 IN SRV  1 2 4555 0200:0000:7400.local
0200:0000:7400._brski-registrar._tcp.local IN TXT "EST-TLS" "prm-jose" "cmp"
0200:0000:7400.local
                 IN AAAA  fda3:79a6:f6ee:0000::0200:0000:6400:0001

                _brski-registrar._udp.local
                 IN PTR  0200:0000:7400._brski-registrar._udp.local
0200:0000:7400._brski-registrar._udp.local
                 IN SRV  1 2 5684 0200:0000:7400.local
0200:0000:7400._brski-registrar._udp.local IN TXT "rrm-cose"
```

Figure 2: DNS-SD for a simple BRSKI and cBRSKI registrar

pty DNS-SD TXT Strings may be problematic, so we use the replacement strings “EST-TLS” and “rrm-cose”

CORE-LF: “bv=<variation-string>

```
Template:

REQ: GET /.well-known/core?rt=brski.r*

RES: 2.05 Content
     <scheme://[Responder_IP_unicast_address]:join-port>;\
       rt=brski.r*(;brv=brski-variation-string)

Example:

REQ: GET /.well-known/core?rt=brski.r*

RES: 2.05 Content
     <coaps://[2001:db8:0:abcd::52]:7634/b>;rt=brski.rs,
     <coaps+jpy://[2001:db8:0:abcd::52]:7633/b>;rt=brski.rjpy,
     <https://[2001:db8:0:abcd::52]:7634/b>;rt=brski.rs;bv=,
     <https://[2001:db8:0:abcd::52]:7634/b>;rt=brski.rs;bv=cmp,
     <https://[2001:db8:0:abcd::52]:7634/b>;rt=brski.rs;bv=prm-jose,
```