Stream:	Internet Engineering Task Force (IETF)
RFC:	9966
Category:	Standards Track
Published:	May 2026
ISSN:	2070-1721

Authors:	O. Friel	D. Harkins
	Cisco	*Hewlett-Packard Enterprise*

RFC 9966
Bootstrapped TLS Authentication with Proof of Knowledge

Abstract

This document defines a mechanism that enables a bootstrapping device to establish trust and mutually authenticate against a TLS server. Bootstrapping devices have a public/private key pair; this mechanism enables a TLS server to prove to the device that it knows the public key and enables the device to prove to the TLS server that it knows the private key. The mechanism leverages existing Device Provisioning Profile (DPP) and TLS standards and can be used in an Extensible Authentication Protocol (EAP) exchange with an EAP server.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Onboarding devices with no, or limited, user interface can be difficult. Sometimes a credential is needed to access a network based on [IEEE8802.1x] or EAP, and network connectivity is needed to obtain a credential. This poses a challenge for onboarding devices.

If a device has a public/private key pair, and trust in the integrity of a device's public key can be obtained in an out-of-band fashion, a device can be authenticated and provisioned with a usable credential for [IEEE8802.1x] / EAP-based network access. While this authentication can be strong, the device's authentication of the network is somewhat weaker. [DUCKLING] presents a functional security model to address this asymmetry.

Device onboarding protocols such as the Device Provisioning Profile [DPP], also referred to as Wi-Fi Easy Connect, address this use case, but they have drawbacks. For instance, [DPP] does not support wired network access and does not specify how the device's DPP key pair can be used in a TLS handshake. This document describes an authentication mechanism that a device can use to mutually authenticate against a TLS server and describes how that authentication protocol can be used in an EAP exchange for wired network access as described in [IEEE8802.1x]. This protocol is called "TLS Proof of Knowledge" or "TLS-POK".

This document does not address the problem of wireless network discovery, where a bootstrapping device detects multiple different wireless networks and needs a more robust and scalable mechanism than simple round-robin to determine the correct network to attach to. DPP addresses this issue for Wi-Fi, but DPP's discovery will not work on a wired [IEEE8802.1x] ethernet port, but TLS-POK will. Therefore, TLS-POK **SHOULD NOT** be used for bootstrapping against wireless networks and **SHOULD** be used for bootstrapping against wired networks.

1.1. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terminology is used throughout this document.

802.1X: IEEE Port-Based Network Access Control [IEEE8802.1x]

BSK: Bootstrap Key, which is an elliptic curve public/private key pair from a cryptosystem suitable for doing ECDSA

DPP: Device Provisioning Profile [DPP]

EAP: Extensible Authentication Protocol [RFC3748]

EC: Elliptic Curve

ECDSA: Elliptic Curve Digital Signature Algorithm

EPSK: External Pre-Shared Key

EST: Enrollment over Secure Transport [RFC7030]

NAI: Network Access Identifier

PSK: Pre-Shared Key

TEAP: Tunnel Extensible Authentication Protocol [RFC9930]

1.2. Bootstrapping Overview

A bootstrapping device holds a public/private Elliptic Curve (EC) key pair, which this document refers to as a "Bootstrap Key" (or "BSK"). The private key of the BSK is known only by the device. The public key of the BSK is:

- known by the device,
- known by the owner or holder of the device, and
- provisioned on the TLS server by the TLS server operator.

In order to establish trust and mutually authenticate, the TLS server proves to the device that it knows the public part of the BSK, and the device proves to the TLS server that it knows the private part of the BSK. More details on the BSK are given in Section 2.

The TLS server could be an EAP server for 802.1X network access or could, for example, be an Enrollment over Secure Transport (EST) [RFC7030] server. In the case of authentication against an EAP server, the EAP server **SHOULD** provision the device with a credential that it uses for subsequent EAP authentication.

1.3. EAP Network Access

Enterprise deployments typically require an 802.1X / EAP-based authentication to obtain network access. Protocols like Enrollment over Secure Transport (EST) [RFC7030] can be used to enroll devices with a Certification Authority to allow them to authenticate using 802.1X / EAP. This creates a problem for bootstrapping devices where a certificate is needed for EAP authentication and 802.1X network access is needed to obtain a certificate.

Devices whose BSK public key can be obtained in an out-of-band fashion and provisioned on the EAP server can authenticate with the EAP server using the mechanism defined in Sections 3 and 4. This network connectivity can then be used to perform an enrollment protocol (such as provided by [RFC9930]) to obtain a credential for subsequent EAP authentication. Certificate lifecycle management may also be performed in subsequent TEAP transactions.

1.4. Supported EAP Methods

This document defines a bootstrapping mechanism that results in a certificate being provisioned on a device that can be used for subsequent EAP authentication. Therefore, an EAP method supporting the provisioning of a certificate on a device is required. The only EAP method that currently supports provisioning of a certificate on a device is TEAP; therefore, this document assumes that TEAP is the only supported EAP method. Section 4 describes how TLS-POK is used with TEAP, including defining a suitable NAI.

If future EAP methods are defined supporting certificate provisioning, then TLS-POK could potentially be used with those methods. Defining how this would work is out of scope of this document.

2. Bootstrap Key

The mechanism for device onboarding defined in this document relies on an EC BSK. This BSK **MUST** be from a cryptosystem suitable for doing ECDSA. A bootstrapping client device has an associated EC BSK. The BSK may be static and baked into device firmware at manufacturing time or may be dynamic and generated at onboarding time by the device. The BSK public key **MUST** be encoded as the DER representation of an ASN.1 SEQUENCE SubjectPublicKeyInfo from [RFC5480]. The subjectPublicKey **MUST** be the compressed format of the public key. Note that the BSK public key encoding **MUST** include the ASN.1 AlgorithmIdentifier in addition to the subjectPublicKey. If the BSK public key can be shared in a trustworthy manner with a TLS server, a form of entity authentication (the step from which all subsequent authentication proceeds) can be obtained.

The exact mechanism by which the TLS server gains knowledge of the BSK public key is out of scope of this specification, but possible mechanisms include scanning a QR code to obtain a base64 encoding of the DER representation of the ASN.1 SubjectPublicKeyInfo or uploading of a Bill of Materials (BOM) that includes this information. More information on QR encoding is given in Section 2.1. If the QR code is physically attached to the client device, or the BOM is associated with the device, the assumption is that the BSK public key obtained in this bootstrapping method belongs to the client. In this model, physical possession of the device implies legitimate ownership of the device.

The TLS server may have knowledge of multiple BSK public keys corresponding to multiple devices, and existing TLS mechanisms are leveraged that enable the server to identify a specific BSK public key corresponding to a specific device.

Using the process defined herein, the client proves to the TLS server that it has possession of the private key of its BSK. Provided that the mechanism in which the server obtained the BSK public key is trustworthy, a commensurate amount of authenticity of the resulting connection can be obtained. The server also proves that it knows the client's BSK public key, which, if the client does not gratuitously expose its public key, can be used to obtain a modicum of correctness, that the client is connecting to the correct server (see [DUCKLING]).

2.1. Alignment with Wi-Fi Alliance Device Provisioning Profile

The definition of the BSK public key aligns with [DPP]. This, for example, enables the QR code format as defined in [DPP] to be reused for TLS-POK. Therefore, a device that supports both wired LAN and Wi-Fi LAN connections can have a single QR code printed on its label, or dynamically display a single QR code on a display, and the BSK can be used for DPP if the device bootstraps against a Wi-Fi network or TLS-POK if the device bootstraps against a wired network. Similarly, a common BSK public key format could be imported into a BOM into a server that handles devices connecting over both wired and Wi-Fi networks.

[DPP], and therefore TLS-POK, does not support the use of RSA or postquantum crypto systems due to the size of public key and its unsuitableness to be represented in a QR code. If [DPP] is modified in the future to support postquantum crypto systems, this document will be updated to track support.

Any bootstrapping method defined for, or used by, [DPP] is compatible with TLS-POK.

3. Bootstrapping in TLS 1.3

Bootstrapping in TLS 1.3 leverages Certificate-Based Authentication (as per [RFC8773]) with an EPSK. The EPSK is derived from the BSK public key as described in Section 3.1, and the EPSK is imported using "Importing External Pre-Shared Keys (PSKs) for TLS 1.3" [RFC9258]. As the BSK public key is an ASN.1 SEQUENCE SubjectPublicKeyInfo from [RFC5480], and not a full PKI Certificate. The client must present the BSK as a raw public key as described in [RFC7250] and use ECDSA as defined in [NIST.FIPS.186-5] for authentication.

The TLS PSK handshake gives the client proof that the TLS server knows the BSK public key. Certificate-based authentication of the client to the server using the BSK gives the server proof that the client knows the BSK private key. This satisfies the proof of ownership requirements outlined in Section 1.

3.1. EPSK Derivation

An EPSK [RFC9258] is made up of the tuple of (Base Key, External Identity, Hash). The Base Key is the DER-encoded ASN.1 subjectPublicKeyInfo representation of the BSK public key. Zero-byte padding **MUST NOT** be added to the DER-encoded representation of the BSK public key.

The External Identity is derived from the DER-encoded representation of the BSK public key using [RFC5869] with the SHA-256 hash algorithm [SHA2] as follows:

```
epskid = HKDF-Expand(HKDF-Extract(<>, Base Key),
                        "tls13-bspsk-identity", L)
where:
  - epskid is the EPSK External Identity
  - Base Key is the DER-encoded ASN.1 subjectPublicKeyInfo
    representation of the BSK public key
  - L equals 32, the length in octets of the SHA-256 output
  - <> is a NULL salt which is a string of L zeros
```

SHA-256 **MUST** be used when deriving epskid using [RFC5869].

The ImportedIdentity structure [RFC9258] is defined as:

```
struct {
   opaque  external_identity<1...2^16-1>;
   opaque  context<0..2^16-1>;
   uint16  target_protocol;
   uint16  target_kdf;
} ImportedIdentity;
```

and is created using the following values:

```
external_identity = epskid
context = "tls13-bsk"
target_protocol = TLS1.3(0x0304)
target_kdf = <as per RFC 9258>
```

The ImportedIdentity context value **MUST** be "tls13-bsk". This informs the server that the mechanisms specified in this document for deriving the EPSK and executing the TLS handshake **MUST** be used. The EPSK and ImportedIdentity are used in the TLS handshake as specified in [RFC9258]. Multiple ImportedIdentity values may be imported as per Section 5.1 of [RFC9258]. The target_kdf follows [RFC9258] and aligns with the cipher suite hash algorithms advertised in the TLS 1.3 handshake between the device and the server.

A server can choose a trade-off between performance and storage by precomputing the identity of every bootstrapped key with every hash algorithm that it uses in TLS and use that to quickly lookup the bootstrap key and generate the PSK. Servers that choose not to employ this optimization will have to do a runtime check with every bootstrap key it holds against the identity the client provides.

Test vectors for derivation of an EPSK External Identity from a BSK are given in the appendix.

3.2. TLS 1.3 Handshake Details

The client includes the "tls_cert_with_extern_psk" extension in the ClientHello, per [RFC8773]. The client identifies the BSK public key by inserting the serialized content of ImportedIdentity into the PskIdentity.identity in the PSK extension, per [RFC9258]. The client **MUST** also include the "client_certificate_type" extension per [RFC7250] in the ClientHello and **MUST** specify type of RawPublicKey.

Upon receipt of the ClientHello, the server looks up the client's EPSK in its database using the mechanisms documented in [RFC9258]. If no match is found, the server **MUST** terminate the TLS handshake with an alert. If the server finds the matching BSK public key, it includes the "tls_cert_with_extern_psk" extension in the ServerHello message and the corresponding EPSK identity in the "pre_shared_key" extension. When these extensions have been successfully negotiated, the TLS 1.3 key schedule **MUST** include both the EPSK in the Early Secret derivation and a Diffie-Hellman Ephemeral (DHE) or ECDHE shared secret value in the Handshake Secret derivation.

After successful negotiation of these extensions, the full TLS 1.3 handshake is performed with the additional caveat that the server **MUST** send a CertificateRequest message and the client **MUST** authenticate with a raw public key (its BSK) per [RFC7250]. The BSK is always an EC key pair; therefore, the type of the client's Certificate **MUST** be ECDSA and **MUST** contain the client's BSK public key as a DER-encoded ASN.1 subjectPublicKeyInfo SEQUENCE.

Note that the client **MUST NOT** share its BSK public key with the server until after the client has completed processing of the ServerHello and verified the TLS key schedule. The PSK proof is completed at this stage, and the server has proven to the client that it knows the BSK public key, and it is therefore safe for the client to send the BSK public key to the server in the Certificate message. If the PSK verification step fails when processing the ServerHello, the client terminates the TLS handshake and the BSK public key **MUST NOT** be shared with the server.

When the server processes the client's Certificate, it **MUST** ensure that it is identical to the BSK public key that it used to generate the EPSK and ImportedIdentity for this handshake.

When clients are configured to trust the first network that proves possession of their public key (as in [DUCKLING]), they **MAY** forgo the checking of the server's certificate in the CertificateVerify and rely on the integrity of the bootstrapping method employed to distribute its key in order to validate trust in the authenticated TLS connection.

The handshake is shown in Figure 1.

```
Client                                              Server
------                                              ------
ClientHello
 +cert_with_extern_psk
 +client_cert_type=RawPublicKey
 +key_share
 +pre_shared_key              -------->
                                               ServerHello
                                    + cert_with_extern_psk
                            + client_cert_type=RawPublicKey
                                               + key_share
                                          + pre_shared_key
                                     {EncryptedExtensions}
                                      {CertificateRequest}
                                             {Certificate}
                                       {CertificateVerify}
                              <--------           {Finished}
{Certificate}
{CertificateVerify}
{Finished}                    -------->
```

Figure 1: TLS 1.3 TLS-POK Handshake

4. Using TLS Bootstrapping in EAP

Upon "link up", an Authenticator on an 802.1X-protected port will issue an EAP Identity request to the newly connected peer. For unprovisioned devices that desire to take advantage of TLS-POK, there is no initial realm in which to construct an NAI (see [RFC7542]). This document uses the NAI mechanisms defined in [RFC9965] and defines the EAP username "tls-pok-dpp" for use with the TEAP realm "teap.eap.arpa". The username "tls-pok-dpp" **MUST** be included, yielding an initial identity of "tls-pok-dpp@teap.eap.arpa". This identifier **MUST** be included in the EAP Identity response in order to indicate to the Authenticator that TEAP is the desired EAP method. [RFC9965] recommends how the device should behave if the Authenticator does not support TEAP or TLS-POK.

```
EAP Peer                      EAP Server
--------                      ----------
                              <- EAP-Request/
                              Identity

EAP-Response/
Identity
(tls-pok-dpp@teap.eap.arpa) ->

                              <- EAP-Request/
                              EAP-Type=TEAP
                             (TLS Start)

EAP-Response/
EAP-Type=TEAP
(TLS client_hello with
 tls_cert_with_extern_psk
 and pre_shared_key) ->

                         .
                         .
                         .
```

Figure 2: EAP Exchange Using TLS-POK

Both client and server have derived the EPSK and associated ImportedIdentity [RFC9258] from the BSK public key as described in Section 3.1. When the client starts the TLS exchange in the EAP transaction, it includes the ImportedIdentity structure in the pre_shared_key extension in the ClientHello. When the server receives the ClientHello, it extracts the ImportedIdentity and looks up the EPSK and BSK public key. As previously mentioned in Section 2, the exact mechanism by which the server has gained knowledge of or been provisioned with the BSK public key is outside the scope of this document.

The server continues with the TLS handshake and uses the EPSK to prove that it knows the BSK public key. When the client replies with its Certificate, CertificateVerify, and Finished messages, the server **MUST** ensure that the public key in the Certificate message matches the BSK public key.

Once the TLS handshake completes, the client and server have established mutual trust. The server can then proceed to provision a credential onto the client using, for example, the mechanisms outlined in [RFC9930].

The client can then use this provisioned credential for subsequent EAP authentication. The BSK is only used during bootstrap and is not used for any subsequent EAP authentication.

5. IANA Considerations

This document adds the following entry to the "EAP Provisioning Identifiers" registry in the "Extensible Authentication Protocol (EAP) Registry" registry group.

NAI: tls-pok-dpp@teap.eap.arpa

Method Type: TEAP

Reference: RFC 9966

6. Implementation Considerations

Three key points are documented above and are repeated here.

- The subjectPublicKey contained in the ASN.1 SEQUENCE SubjectPublicKeyInfo **MUST** be the compressed format of the public key.
- When deriving the EPSK from the BSK, zero-byte padding **MUST NOT** be added to the DER-encoded representation of the BSK public key.
- SHA-256 **MUST** be used when following [RFC5869] to derive the EPSK from the BSK.
- The BSK public key **MUST NOT** be freely available on the network.

7. Security Considerations

Bootstrap and trust establishment by the TLS server are based on proof of knowledge of the client's BSK public key, a non-public datum. The TLS server obtains proof that the client knows its BSK public key and also possesses its corresponding private key.

Trust on the part of the client is based on successful completion of the TLS 1.3 handshake using the EPSK derived from the BSK. This proves to the client that the server knows its BSK public key. In addition, the client assumes that knowledge of its BSK public key is not widely disseminated; therefore, any server that proves knowledge of its BSK public key is the appropriate server from which to receive provisioning, for instance via [RFC9930]. [DUCKLING] describes a security model for this type of "imprinting".

An attack on the bootstrapping method, which substitutes the public key of a rogue device for the public key of an honest device, can result in the TLS server onboarding and trusting the rogue device.

If an adversary has knowledge of the BSK public key, the adversary may be able to make the client bootstrap against the adversary's network. For example, if an adversary intercepts and scans QR labels on clients, and the adversary can force the client to connect to its server, then the adversary can complete the TLS-POK handshake with the client and the client will connect to the adversary's server. Since physical possession implies ownership, there is nothing to prevent a stolen device from being onboarded.

The BSK key pair used for ECDSA **MUST** be generated and validated according to Section 6.2 of [NIST.FIPS.186-5].

Manufacturers **SHOULD** use a unique BSK for every single device they manufacture. If multiple devices share the same BSK, then network operators cannot differentiate between these devices and cannot ensure that only specific authorized devices are allowed connect to their networks.

8. References

8.1. Normative References

[NIST.FIPS.186-5] NIST, "Digital Signature Standard (DSS)", NIST FIPS 186-5, DOI 10.6028/NIST.FIPS.186-5, February 2023, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8773] Housley, R., "TLS 1.3 Extension for Certificate-Based Authentication with an External Pre-Shared Key", RFC 8773, DOI 10.17487/RFC8773, March 2020, <https://www.rfc-editor.org/info/rfc8773>.

[RFC9258] Benjamin, D. and C. A. Wood, "Importing External Pre-Shared Keys (PSKs) for TLS 1.3", RFC 9258, DOI 10.17487/RFC9258, July 2022, <https://www.rfc-editor.org/info/rfc9258>.

[RFC9965] DeKok, A., "The eap.arpa. Domain and Extensible Authentication Protocol (EAP) Provisioning", RFC 9965, DOI 10.17487/RFC9965, May 2026, <https://www.rfc-editor.org/info/rfc9965>.

8.2. Informative References

[DPP] Wi-Fi Alliance, "Wi-Fi Easy Connect(TM) Specification", Version 3.0, 2022, <https://www.wi-fi.org/system/files/Wi-Fi_Easy_Connect_Specification_v3.0.pdf>.

[DUCKLING] Stajano, F. and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-Hoc Wireless Networks", Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science, vol. 1796, pp. 172-182, DOI 10.1007/10720107_24, 1999, <https://www.cl.cam.ac.uk/~fms27/papers/1999-StajanoAnd-duckling.pdf>.

[IEEE8802.1x] IEEE/ISO/IEC, "IEEE/ISO/IEC International Standard-Telecommunications and exchange between information technology systems--Requirements for local and metropolitan area networks-/-Part 1X:Port-based network access control", ISO/IEC/IEEE 8802-1X:2021, DOI 10.1109/IEEESTD.2021.9650828, 2021, <https://ieeexplore.ieee.org/document/9650828>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/info/rfc7030>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/info/rfc7542>.

[RFC9930] DeKok, A., Ed., "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 9930, DOI 10.17487/RFC9930, February 2026, <https://www.rfc-editor.org/info/rfc9930>.

[SHA2] NIST, "Secure Hash Standard", NIST FIPS 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

Appendix A. Test Vectors

A.1. Test Vector 1: prime256v1

Base64 encoding of BSK:


```
MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACMvLyoOykj8sFJxSoZfzafuVEvM+kNYCxp
EC6KITLb9g=
```


Base64 encoding of epskid:

`Bd+lLlg/ERdtYacfzDfh1LjdL0+QWJQHdYXoS7JDSkA=`

A.2. Test Vector 2: secp384r1

Base64 encoding of BSK:


```
MEYwEAYHKoZIzj0CAQYFK4EEACIDMgACwDXKQ1pytcR1WbfqPaNGaXQ0RJnijJG1em8ZK
ilryZRDfNioq7+EPquT6l9laRvw
```


Base64 encoding of epskid:


```
yMWK26ec3klVFewg2znKntQgVoRcRRjW81n677GL+8w=
```


A.3. Test Vector 3: secp521r1

Base64 encoding of BSK:


```
MFgwEAYHKoZIzj0CAQYFK4EEACMDRAADAIiHIAOXdPVuI8khCnJQHT1j53rQRnFCcY3CZ
UvxdXKJR9KW5RVB3HDQfmkoQWHEz4XngXUeFyDXliEo3eF6vhqDMFgwEAYHKoZIzj0CAQ
YFK4EEACMDRAADAIiHIAOXdPVuI8khCnJQHT1j53rQRnFCcY3CZUvxdXKJR9KW5RVB3HD
QfmkoQWHEz4XngXUeFyDXliEo3eF6vhqD
```


Base64 encoding of epskid:

`D+s3Ex81A8N36ECI3AdXwBzrOXuonZUMdhhHXVINhg8=`

A.4. Test Vector 4: brainpoolP256r1

Base64 encoding of BSK:

```
MDowFAYHKoZIzj0CAQYJKyQDAwIIAQEHAyIAA3fyUWqiV8NC9DAC88JzmVqnoT/
reuCvq8lHowtwWNOZ
```


Base64 encoding of epskid:


```
j2TLWcXtrTej+f3q7EZrhp5SmP31uk1ZB23dfcR93EY=
```


Authors' Addresses

Owen Friel
Cisco
Email: ofriel@cisco.com

Dan Harkins
Hewlett-Packard Enterprise
Email: daniel.harkins@hpe.com