Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 10 August 2025

O. Friel
Cisco
D. Harkins
Hewlett-Packard Enterprise

6 February 2025

Bootstrapped TLS Authentication with Proof of Knowledge (TLS-POK)
draft-ietf-emu-bootstrapped-tls-08

Abstract

This document defines a mechanism that enables a bootstrapping device to establish trust and mutually authenticate against a network. Bootstrapping devices have a public private key pair, and this mechanism enables a network server to prove to the device that it knows the public key, and the device to prove to the server that it knows the private key. The mechanism leverages existing DPP and TLS standards and can be used in an EAP exchange.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 10 August 2025.

Copyright Notice

Table of Contents

1. Introduction

On-boarding of devices with no, or limited, user interface can be difficult. Typically, a credential is needed to access the network, and network connectivity is needed to obtain a credential. This poses a catch-22.

If a device has a public / private keypair, 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 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 on-boarding protocols such as the Device Provisioning Profile [DPP], also referred to as Wi-Fi Easy Connect, address this use case but they have drawbacks. [DPP] for instance does not support wired network access, and does not specify how the device's DPP keypair can be used in a TLS handshake. This document describes an on-boarding protocol that can be used for wired network access, which we refer to as TLS Proof of Knowledge or TLS-POK.

This document does not address the problem of Wi-Fi network discovery, where a bootstrapping device detects multiple different Wi-Fi 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. Thus, the intention is that DPP is the RECOMMENDED mechanism for bootstrapping against Wi-Fi networks, and TLS-POK is the RECOMMENDED mechanism 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

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

* DPP: Device Provisioning Protocol [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]

* PSK: Pre-Shared Key

* TEAP: Tunnel Extensible Authentication Protocol [RFC7170]

1.2. Bootstrapping Overview

A bootstrapping device holds a public / private elliptic curve (EC) key pair which we refer to as a Bootstrap Key (BSK). The private key of the BSK is known only by the device. The public key of the BSK is known by the device, is known by the owner or holder of the device, and is provisioned on the network by the network operator. In order to establish trust and mutually authenticate, the network proves to the device that it knows the public part of the BSK, and the device proves to the network that it knows the private part of the BSK. Once this trust has been established during bootstrapping, the network can provision the device with a credential that it uses for subsequent network access. More details on the BSK are given in Section 2.

1.3. EAP Network Access

Enterprise deployments typically require an [IEEE802.1X]/EAP-based authentication to obtain network access. Protocols like Enrollment over Secure Transport (EST) [RFC7030] can be used to enroll devices into a Certification Authority to allow them to authenticate using 802.1X/EAP. This creates a Catch-22 where a certificate is needed for network access and network access is needed to obtain certificate.

Devices whose BSK public key can be obtained in an out-of-band fashion and provisioned on the network can perform a TLS-based EAP exchange, for instance Tunnel Extensible Authentication Protocol (TEAP) [RFC7170], and authenticate the TLS exchange using the bootstrapping mechanisms defined in Section 3. This network connectivity can then be used to perform an enrollment protocol (such as provided by [RFC7170]) to obtain a credential for subsequent network connectivity and certificate lifecycle maintenance.

1.4. Supported EAP Methods

This document defines a boostrapping mechanism that results in a certificate being provisioned on a device that can be used for subsequent network access. Therefore, an EAP method that supports 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 Section 4 describes how TLS-POK is used with TEAP, including defining a suitable NAI.

If future EAP methods are defined that support 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 on-boarding of devices defined in this document relies on an elliptic curve (EC) bootstrap key (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 on-boarding time by the device. The BSK public key MUST be encoded as the DER representation of an ASN.1 SEQUENCE SubjectPublicKeyInfo from [RFC5280]. 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 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) which 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.

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

Using the process defined herein, the client proves to the 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 network (see [duckling]).

2.1. Alignment with Wi-Fi Alliance Device Provisioning Profile

The definition of the BSK public key aligns with that given in [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 bootstrap key 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 bootstrap public key format could be imported into a BOM into a server that handles devices connecting over both wired and Wi-Fi networks.

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 [RFC8773] Certificate-Based Authentication with an External Pre-Shared Key. The External PSK (EPSK) is derived from the BSK public key as described in Section 3.1, and the EPSK is imported using [RFC9258] Importing External Pre-Shared Keys (PSKs) for TLS 1.3. As the BSK public key is an ASN.1 SEQUENCE SubjectPublicKeyInfo from [RFC5280], and not a full PKI Certificate, the client must use [RFC7250] Using Raw Public Keys in TLS and DTLS in order to present the BSK as raw public key.

The TLS PSK handshake gives the client proof that the 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. External PSK Derivation

An [RFC9258] EPSK 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 [RFC9258] ImportedIdentity structure 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 RFC9258>
```

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 [RFC9258] section 5.1. 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 performance versus storage tradeoff a server can choose is to precompute 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 [RFC7250] "client_certificate_type" extension in the ClientHello and MUST specify type of RawPublicKey.

Upon receipt of the ClientHello, the server looks up the client’s EPSK key 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 found 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 an (EC)DHE 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 client MUST authenticate with a raw public key (its BSK) per [RFC7250]. The BSK is always an elliptic curve 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 has completed at this stage, and the server has proven to the client that is 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 use the [duckling] form of authentication, 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.

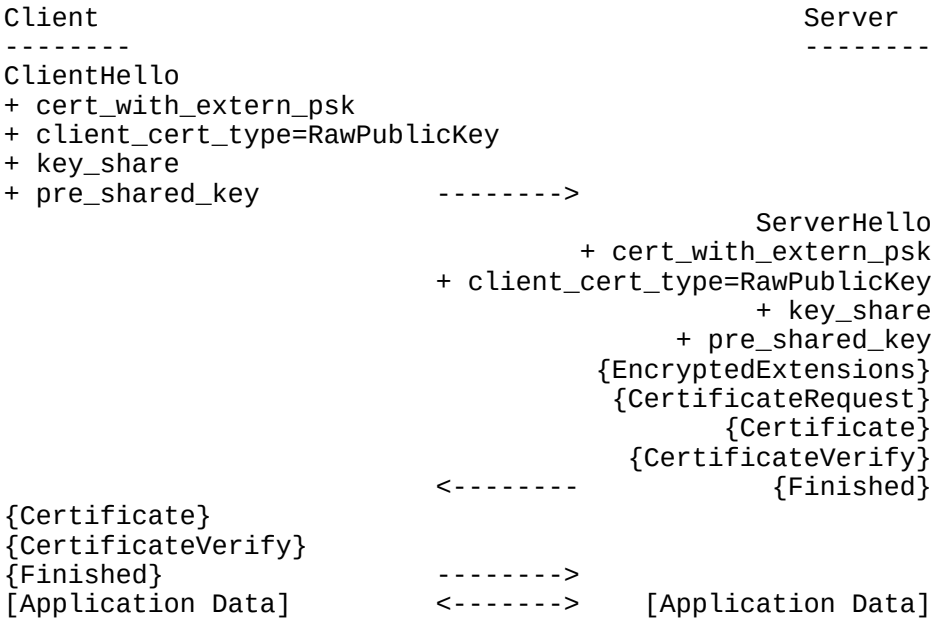
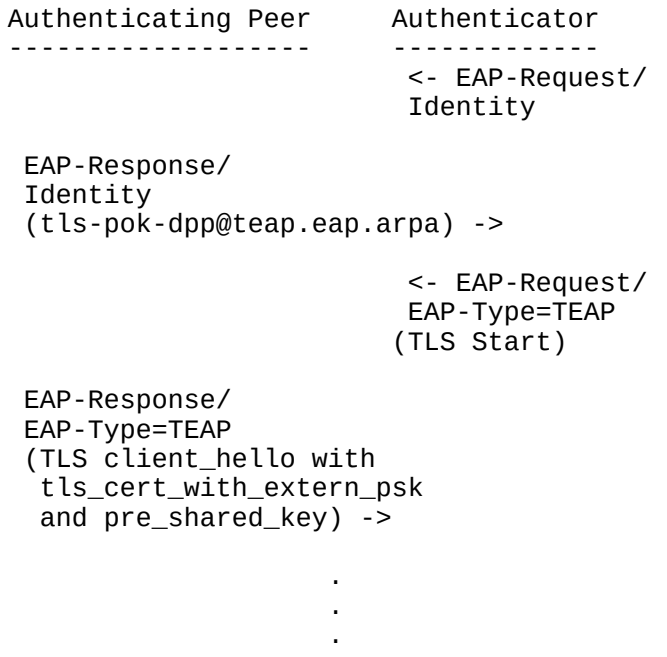


Figure 1: TLS 1.3 TLS-POK Handshake

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 [I-D.ietf-emu-eap-arpa] 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. [I-D.ietf-emu-eap-arpa] recommends how the device should behave if the Authenticator does not support TEAP or TLS-POK.



Both client and server have derived the EPSK and associated [RFC9258] ImportedIdentity 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 received 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 [RFC7170].

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

5. IANA Considerations

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

NAI: tls-pok-dpp@teap.eap.arpa Method Type: TEAP Reference: THIS DOCUMENT

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 External PSK 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 using [RFC5869] to derive the External PSK from the BSK.

7. Security Considerations

Bootstrap and trust establishment by the TLS server is based on proof of knowledge of the client’s bootstrap public key, a non-public datum. The TLS server obtains proof that the client knows its bootstrap public key and, in addition, 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 and therefore any server that proves knowledge of its BSK public key is the appropriate server from which to receive provisioning, for instance via [RFC7170]. [duckling] describes a security model for this type of "imprinting".

An attack on the bootstrapping method which substitutes the public key of a corrupted device for the public key of an honest device can result in the TLS sever on-boarding and trusting the corrupted device.

If an adversary has knowledge of the bootstrap 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 on-boarded.

8. References

8.1. Normative References

[I-D.ietf-emu-eap-arpa]
DeKok, A., "The eap.arpa domain and EAP provisioning", Work in Progress, Internet-Draft, draft-ietf-emu-eap-arpa-06, 29 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-emu-eap-arpa-06>.

[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/rfc/rfc2119>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[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/rfc/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/rfc/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/rfc/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/rfc/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/rfc/rfc9258>.

8.2. Informative References

[DPP] Wi-Fi Alliance, "Device Provisioning Profile", 2020.

[duckling] Stajano, F. and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-Hoc Wireless Networks", 1999, <https://www.cl.cam.ac.uk/~fms27/papers/1999-StajanoAnd-duckling.pdf>.

[IEEE802.1X] IEEE, "Port-Based Network Access Control", 2010.

[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/rfc/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/rfc/rfc7030>.

[RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <https://www.rfc-editor.org/rfc/rfc7170>.

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

[sha2] National Institute of Standards and Technology, "FIPS 180-4 Secure Hash Standard (SHS)", August 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

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