Usage of SP800-56A in Industry Standard Protocols

Overview and Discussion

Michael Powers
Cryptographic & Security Testing Laboratory (CSTL)
6841 Benjamin Franklin Drive
Columbia, MD 21046
NVLAP Lab Code: 200427-0

leidos

Who am I?

- **Michael Powers**
- Currently working for the Leidos AT&E labs as the CSTL Technical Director
- Responsible for FIPS 140-2 and SCAP-related activities
- Previous work experience includes penetration testing, reverse engineering, server administration, and product development
- B.S in Mathematics and a Minor in Computer Science from UMBC

Agenda

- Motivations behind this talk:
  - Upcoming SP800-56A Key Agreement Transitions
- Introductory material:
  - Brief Overview of the **History** of SP800-56A
  - Overview of the SP800-56A **Schemes**
- Discussion:
  - Discussion of how the SP800-56A schemes fall into the **TLS** protocol
  - Discussion of how the SP800-56A schemes fall into the **SSH** protocol
  - Discussion of how the SP800-56A schemes fall into the **IPsec** protocol
  - Brief Discussion on Required **Self-Tests** for SP800-56A

leidos

Upcoming SP800-56A Key Agreement Transitions

Upcoming SP800-56A Key Agreement Transitions

- Non-approved Key Agreement Schemes are disallowed after **December 31, 2017**
  - **Source:** SP800-131A Revision 1
- Implications:
  - Certificates that have currently-"allowed" listings like the ones below will no longer be allowed to use them in FIPS mode:
    - **Diffie-Hellman (key agreement; key establishment methodology provides 112 bits of encryption strength)**
    - **EC Diffie-Hellman (shared secret computation provides 192 bits of encryption strength)**
  - In order to use these Key Agreement Schemes after the transition date, they must be either (1) **Certified** through the CAVP (e.g.: via a **KAS** certificate) or (2) **Vendor-affirmed**. See IG **D.1-Rev2** for more information on the vendor-affirmation requirements.

leidos

Brief Overview of the **History** of SP800-56A

leidos

Brief Overview of the History of SP800-56A – Revision 1

- The original version of SP800-56A was published on **May 3, 2006**
- The first revision of SP800-56A was published on **March 8, 2007**
- **Changes Introduced (very minor):**
  - The standard was updated to allow for the **dual use** of keys in one context – **Certificate Requests**. Specifically, a private key that is intended for usage in key establishment can also be used to sign the initial certificate request for the associated public key certificate.
- With the early revisions of SP800-56A, I believe the hope was that major protocols (such as TLS) would adopt the KDFs proposed in SP800-56A
  - This, however, did not occur as hoped – the Working Group Chair reviewed the SP800-56A KDFs and determined that they were **not a good fit for TLS** *(source **[5]**)*

Brief Overview of the History of SP800-56A – Revision 2

- The second revision of SP800-56A was published in **May 2013**
- **Major Changes Introduced:**
  - Added CMAC as an approved MAC
  - Changed the procedures for both FFC and ECC key-pair generation
  - Added the option of **Application-Specific Key-Derivation Methods**
  - HMAC with an approved hash function is now approved for one-step key derivation
  - SP800-56C “extraction-then-expansion” method is now approved for two-step key derivation
  - **Many** other changes…

Brief Overview of the History of SP800-56A – Outstanding Issues and "Gotchas"

- As of the time of writing this talk, there are a few pending issues I see with SP800-56A:
  - **(#1)** For Diffie-Hellman, the current FIPS standards/IGs do **not allow for many common groups** after the December 31, 2017 transition date, such as the MODP groups defined in RFC 3426 (for IKEv2) or the Oakley Groups defined in RFC 2409 (for IKE).
    - **The groups in RFC 5114** appear to be compliant to SP800-56A, in particular:
      - 2048-bit MODP Group with 224-bit Prime Order Subgroup
      - 2048-bit MODP Group with 256-bit Prime Order Subgroup

leidos

Brief Overview of the History of SP800-56A – Outstanding Issues and "Gotchas" (Cont.)

- **(#2)** For Diffie-Hellman, you're restricted to schemes that use a **2048-bit** modulus. This seems inconsistent with SP800-131A, as well as FIPS 186-4 DSA (which allows up to **3072-bit**) and RSA (which actually allows any arbitrary size **>= 2048-bits**).
- **(#3)** For EC Diffie-Hellman, the current FIPS standards/IGs do **not allow for non-NIST curves** after the December 31, 2017 transition date, such as Curve25519.
  - **FIPS 140-2 IG A.2** outlines that non-Approved ECDSA curves can be used in the FIPS Approved mode of operation (with some requirements), but the IG only appears to be relevant to **ECDSA**, and **<u>not ECDH</u>.**

leidos

Overview of the SP800-56A **Schemes**

leidos

Overview of the SP800-56A Schemes – Acronyms I’ll be using

- **FFC** – Finite Field Cryptography (e.g.: what is used in Diffie-Hellman and MQV)
- **ECC** – Elliptic Curve Cryptography (e.g.: what is used in EC Diffie-Hellman and EC MQV)
- **DH** – Diffie-Hellman
- **ECDH** – Elliptic Curve Diffie-Hellman
- **MQV** – Menezes-Qu-Vanstone (Another key agreement scheme based on DH/ECDH)
- **TLS –** Transport Layer Security
- **SSH –** Secure Shell
- **IPsec –** Internet Protocol Security
- **IKE –** Internet Key Exchange

Overview of the SP800-56A Schemes

- **“2e, 2s” schemes (dhHybrid1, Full Unified Model, Full MQV) -** Each party generates an ephemeral key pair and uses a static key pair. The shared secret is derived from a combination of the result of the ephemeral and static exchanges.
- **“2e, 0s” schemes (dhEphem, Ephemeral Unified Model)** – Each party generates just a single ephemeral key pair to arrive at a shared secret. This is (more or less) “vanilla” DH/ECDH.
- **“1e, 2s” schemes (dhHybridOneFlow, One-Pass Unified Model, MQV1)** – One party generates an ephemeral key pair and both parties use a static key pair. The shared secret is derived from a combination of the result of the ephemeral/static and static exchanges.

leidos

Overview of the SP800-56A Schemes (Continued)

- **“1e, 1s” schemes (dhOneFlow, One-Pass Diffie-Hellman) -** One party generates an ephemeral key pair, and one party has a static key pair. The shared secret is derived from a single exchange involving the ephemeral/static key pairs.
- **“0e, 2s” schemes (dhStatic, Static Unified Model) –** Both parties use only static key pairs. A nonce is exchanged and the shared secret is derived from a single exchange involving just the static key pairs.

Discussion of how the SP800-56A schemes fall into the **TLS** protocol

leidos

Discussion of how the SP800-56A schemes fall into the TLS protocol

- **TLS_DH –** Scheme rarely seen in the wild. TLS_DH utilizes the '*dhStatic*' or '*dhOneFlow*' SP800-56A Revision 2 scheme with the TLS KDF depending on whether or not the client utilizes a static or ephemeral key.
- **TLS_DHE –** Common scheme. TLS_DHE utilizes the '*dhEphem*' SP800-56A Revision 2 scheme with the TLS KDF.
- **TLS_ECDH –** Scheme rarely seen in the wild. TLS_ECDH utilizes the '*Static Unified Model*' or the '*One-Pass Diffie-Hellman*' SP800-56A Revision 2 scheme with the TLS KDF depending on whether or not the client utilizes a static or ephemeral key.
- **TLS_ECDHE –** Common scheme. TLS_ECDHE utilizes the '*Ephemeral Unified Model*' SP800-56A Revision 2 scheme with the TLS KDF.
- ***TLS_DH_anon/TLS_ECDH_anon –*** *Similar to TLS_DH/TLS_ECDH except there is no authentication of the server. As such, these do not fall under any of the SP800-56A schemes.*
- ***TLS_PSK –*** *A pre-shared key is used, bypassing any need for DH/ECDH. This does not fall under an SP800-56A scheme.*
- ***TLS_SRP –*** *A password is used, bypassing any need for DH/ECDH. This does not fall under an SP800-56A scheme.*

leidos

Discussion of how the SP800-56A schemes fall into the TLS protocol (Continued)

- One quick way to determine which ciphers that your TLS implementation supports would be to utilize the **SSL Server Test** provided by Qualys (source **[4]**)
- Using this, you will be able to generate a list of supported cipher suites, and look at the prefixes, matching them to what I outlined on the previous slide:
  - **TLS_ECDHE**_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)[1]
  - **TLS_ECDHE**_RSA_WITH_AES_128_CBC_SHA256 (0xc027)[1]
  - **TLS_DHE**_RSA_WITH_AES_128_GCM_SHA256 (0x9e) **DH 2048 bits**[2]
  - **TLS_DHE**_RSA_WITH_AES_256_GCM_SHA384 (0x9f) **DH 2048 bits**[2]
  - *TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)*
  - *TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)*
- [1]**Important Note:** The Diffie-Hellman schemes MUST use 2048-bit ‘p’ values and 224 or 256-bit ‘q’ values. See next slide for more detail.
- [2]**Important Note:** The implementation must support at least one of the NIST curves. See next slide for more detail.

Discussion of how the SP800-56A schemes fall into the TLS protocol (Continued)

- **Important Note:** The Diffie-Hellman schemes **MUST** use 2048-bit ‘p’ values and 224 or 256-bit ‘q’ values:
  - With **Apache** (2.4.8 and newer), you can set custom DH parameters via the configuration ‘*SSLOpenSSLConfCmd DHParameters “<path>”*’
  - With **nginx**, you can set custom DH parameters via the configuration ‘*ssh_dhparam <path>*’
- **Important Note:** The implementation must support **at least one** of the NIST curves:
  - On the Qualys SSL Server Test results page, you should see a line like what is depicted below. If you see any subset of the curves I’ve listed, then you are good to go!

Supported EC Named Curves	sect571k1, sect571r1, secp521r1, sect409k1, sect409r1, secp384r1, sect283k1, sect283r1, secp256r1, sect233k1, sect233r1, secp224r1 (server preferred order)

leidos

Discussion of how the SP800-56A schemes fall into the **SSH** protocol

leidos

Discussion of how the SP800-56A schemes fall into the SSH protocol

- The SSH protocol supports many Key Exchange algorithms. The following are some common ones that ***can be*** compliant with SP800-56A, with some caveats:
  - **diffie-hellman-group-exchange-sha256 -** This key exchange utilizes the '*dhEphem*' SP800-56A Revision 2 scheme with the SSH KDF.
  - **ecdh-sha2-nistp256 –** This key exchange utilizes the '*Ephemeral Unified Model*' SP800-56A Revision 2 scheme with the SSH KDF.
  - **ecdh-sha2-nistp384 -** This key exchange utilizes the '*Ephemeral Unified Model*' SP800-56A Revision 2 scheme with the SSH KDF.
  - **ecdh-sha2-nistp521 -** This key exchange utilizes the '*Ephemeral Unified Model*' SP800-56A Revision 2 scheme with the SSH KDF.

Discussion of how the SP800-56A schemes fall into the SSH protocol (Continued)

- One quick way to determine which ciphers that your SSH implementation supports would be to connect to the SSH server using OpenSSH, with the following command: ‘ssh –vvv <host>’
  - In the debugging output, you should see a line that reads something like *“debug2: kex_parse_kexinit: curve25519-sha256@libssh.org,**ecdh-sha2-nistp256**,**ecdh-sha2-nistp384**,**ecdh-sha2-nistp521**,**diffie-hellman-group-exchange-sha256***,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1”*
- [1]**Important Note:** The Diffie-Hellman schemes MUST use 2048-bit ‘p’ values and 224 or 256-bit ‘q’ values as per SP800-56A. See next slide for more detail.

Discussion of how the SP800-56A schemes fall into the SSH protocol (Continued)

- **Important Note:** The Diffie-Hellman schemes MUST use 2048-bit ‘p’ values and 224 or 256-bit ‘q’ values as per SP800-56A:
  - With OpenSSH you can set custom Diffie-Hellman parameters by using the ‘**moduli**’ (or ‘**primes**’ on older systems) file contained in **/etc/ssh/**
  - **Note** that the ‘**ssh-keygen**’ appears to produce “Safe” primes (e.g.: p = 2q + 1) by default, which are **not compliant** to SP800-56A; so using the standard methods of generating primes is not an option.
    - One would need to use either (1) publicly-published values (such as those published in **RFC 5114**) or (2) custom-generated (e.g.: not using ssh-keygen) values in order to be compliant with SP800-56A.
    - *I have personally confirmed that manually inputting the two 2048-bit primes defined in RFC 5114 works with OpenSSH with no apparent issues, despite them not being “Safe” primes.*

leidos

Discussion of how the SP800-56A schemes fall into the **IPsec** protocol

leidos

Discussion of how the SP800-56A schemes fall into the IPsec protocol

- The IPsec protocol uses IKEv1/IKEv2 for key exchange. This is where Diffie-Hellman and EC Diffie-Hellman are used within the IPsec protocol.
  - *For DH, the '**dhEphem**' SP800-56A scheme is used, and for ECDH the '**Ephemeral Unified Model**' SP800-56A scheme is used.*
- **IKE has support for many DH groups, such as:**
  - ***RFC 2409** defines two 'Oakley Groups' for usage in IKE – Also referred to as groups 1 and 2*
  - ***RFC 3526** defines six 'MODP Groups' for usage in IKE – Also referred to as groups 5, 14, 15, 16, 17 and 18*
  - **RFC 5114** defines three 'MODP Groups with Prime Order Subgroup' for usage in IKE. These are also referred to as groups **22**, **23** and **24**
- **IKE has support for many ECDH curves as well, such as:**
  - ***RFC 2409** defines two 'Oakley Groups' for usage in IKE – Also referred to as groups 3 and 4*
  - **RFC 5114** defines five 'Random ECP Groups' for usage in IKE, which are just six of the NIST Curves (P-192, P-224, P-256, P-384, P-521) These are also referred to as groups **25**, **26**, **19**, **20** and **21** respectively.
  - ***RFC 6932** defines four 'Brainpool Elliptic Curves' for usage in IKE – also referred to as groups 27, 28, 29 and 30.*

Discussion of how the SP800-56A schemes fall into the IPsec protocol (Continued)

- There is no particularly '*simple*' universal way to determine which Key exchange protocols that a particular IKE implementation supports. I recommend Google:
  - **Microsoft** supports DH groups 1, 2 and 14 as well as ECDH with P-256 and P-384 (source **[10]**)
  - **StrongSwan** supports DH groups 1, 2, 5, 14, 15, 16, 17, 18, 22, 23 and 24 as well as ECDH with P-192, P-224, P-256, P-384, P-521, Brainpool curves, and Curve25519 (sources **[11]** and **[12]**)
- **Important Note:** It looks like for SP800-56A Revision 2, the groups 19, 20, 21, 22, 23, 24 and 26 are the only standardized groups that would be compliant.

leidos

Brief Discussion on Required **Self-Tests** for SP800-56A

leidos

Brief Discussion on Required Self-Tests for SP800-56A

- In order to claim an implementation as compliant to SP800-56A, it also needs to meet the **self-test requirements** as outlined in **FIPS 140-2 IG 9.6**:
  - **Primitive "Z" Computation KAT** – E.g.: Perform a simulated DH / ECDH exchange, and verify that the resulting shared secret is correct.
  - **Key Derivation Function (KDF) KAT –** The underlying SHA function used in the KDF needs to be tested via a Known Answer Test.
  - **KATs on Prerequisite Algorithms** – DSA, ECDSA, SHS and DRBG can all be prerequisites for the SP800-56A schemes. They all need to have the required self-tests implemented.
  - **Conditional Tests for Assurances**
    - **5.5.2 –** Verifying the correctness of the domain parameters (e.g.: DSA PQG(ver) or just using a NIST-defined or RFC-defined set of parameters)
    - **5.6.2 –** Verifying correctness of keys (e.g.: Certifying DSA/ECDSA Key Generation, ECDSA PKV, pairwise consistency)
    - **5.6.3** – Keys can't be associated with multiple sets of domain parameters, keys must be generated using Approved methods, keys must be protected from unauthorized access, disclosure, modification and substitution.
  - **Conditional Tests on Prerequisite Algorithms -** Pair-wise conditional test (e.g.: pair-wise consistency test) must be performed on every key pair generated by the module.

Questions?

Leidos CSTL Contact Information

- **Michael Powers** – CSTL Technical Director
  - Michael.C.Powers@leidos.com
  - +1 (443) 367-7422
- **Jason Tseng** – CSTL Laboratory Manager
  - Jason.K.Tseng@leidos.com
  - +1 (443) 367-7808
- **Amit Sharma** – AT&E Laboratory Director
  - Amit.Sharma@leidos.com
  - +1 (443) 367-7733
- www.leidos.com/infosec/testing-accreditation

Sources

- [1] SP800-131A Revision 1
  - http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf
- [2] SP800-56A Revision 1
  - http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf
- [3] SP800-56A Revision 2
  - http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf
- [4] Qualys SSL Server Test
  - https://www.ssllabs.com/ssltest/
- [5] TLS 1.2 and NIST SP 800-56A
  - https://www.ietf.org/proceedings/67/slides/tls-2/tls-2.ppt
- [6] RFC 2409, [7] 3526, [8] 5114 and [9] 6932
  - https://tools.ietf.org/html/rfc2409
  - https://tools.ietf.org/html/rfc3526
  - https://tools.ietf.org/html/rfc5114
  - https://tools.ietf.org/html/rfc6932
- [10] IPsec Algorithms and Method Supported in Windows
  - https://technet.microsoft.com/en-us/library/dd125380(v=ws.10).aspx
- [11] IKEv1 Cipher Suites and [12] IKEv2 Cipher Suites
  - https://wiki.strongswan.org/projects/strongswan/wiki/IKEv1CipherSuites
  - https://wiki.strongswan.org/projects/strongswan/wiki/IKEv2CipherSuites