SMB 3.1.1 security in Windows 10: Multichannel with pre-authentication integrity

Edgar Olougouna

Key takeaways

- End-to-end integrity with SMB 3.1.1
- Mandatory pre-authentication integrity in SMB 3.1.1
- Multichannel key calculation in SMB 3.0.x and SMB 3.1.1

Background

- Windows and chronology of dialects in MS-SMB2
- Security considerations in SMB2 and SMB3
- Message integrity prior to SMB 3.1.1

Dialects in Windows 10 v1703

Client	Server	Dialect
Vista	2008	2.0.2
7	2008 R2	2.1
8	2012	3.0
8.1	2012 R2	3.0.2
10 v1703	2016	3.1.1

- SMB2 Negotiate is un-signed. Problem of MITM downgrading the dialect or tampering with the capabilities.
- SMB 3.0: post-authentication validation of the Negotiate.
- SMB 3.1.1: pre-authentication integrity allows end-to-end message integrity.

Chronology of crypto algorithms in MS-SMB2

Dialect	Key derivation	Signing	Encryption
2.0.2	GSS-API SessionKey	HMAC-SHA256	N/A
2.1	GSS-API SessionKey	HMAC-SHA256	N/A
3.0	GSS-API SessionKey & KDF [SP800-108]	AES-128-CMAC	AES-128-CCM
3.0.2	GSS-API SessionKey & KDF [SP800-108]	AES-128-CMAC	AES-128-CCM
3.1.1	SHA-512 & GSS-API SessionKey & KDF [SP800-108]	AES-128-CMAC	AES-128-CCM and AES-128-GCM

- Ref: GSS-API [RFC2743]; KDF [SP800-108]; SHA-512 [FIPS180-4]; HMAC-SHA256 [FIPS180-2] and [RFC2104]; AES-128-CMAC [RFC4493]; AES-CCM and AES-GCM [RFC5084]

Key consideration: SessionKey

- the starting point of all cryptographic keys used in SMB 2.x and 3.x.
- Unique, Secret, and Genuinely impossible to guess.
- Authentication mechanism needs to provide uniqueness and randomness for session keys.

- SessionKey: first 16 bytes of the key queried from the GSS protocol (e.g. Kerberos [MS-KILE], NTLM [MS-NLMP]) for this authenticated context.
- In SSPI, the session key is extracted by using QueryContextAttributes with SECPKG_ATTR_SESSION_KEY.

NTLM SessionKey

- SessionKey is the ExportedSessionKey based on password hashing, described in MS-NLMP.
- Weak cryptography (MD4, DES, MD5, RC4) but widely-used in non-domain environments.
  - No support for modern cryptographic methods, such as AES or SHA-256.
  - No salt. With the knowledge of password, MITM can easily produce the same key.
  - No mutual authentication between client and server.

- At a minimum:
- Choose password length and complexity carefully to mitigate brute force attacks.
- NTLMv2 should be used
- Auditing should be in place.

- Auditing and restricting NTLM usage guide
- https://technet.microsoft.com/en-us/library/jj865674.aspx

Kerberos SessionKey

- the SessionKey is either the sub-session key, or session key [RFC4120].
- Protected with the strongest encryption type (etype).
- Etype based on configuration of what the KDC enforces and what client and server principals can support (AES256_HMAC_SHA1, AES128_HMAC_SHA1, RC4_HMAC_MD5, DES_CBC_MD5, DES_CBC_CRC).

Message integrity prior to SMB 3.1.1

- Signed session
- Secure dialect negotiation in SMB 3.0.x

Signed session

- Every signed packet includes a signature that the receiver can validate.
- SMB1 signing uses MD5 [RFC1321] as hashing algorithm
- SMB2 uses a better hashing for signing.
- SMB 2.02 and SMB 2.1 use HMAC SHA-256,
- SMB 3.0 upgrades to AES-128-CMAC for its signing.
- SMB 2.x did not have a mechanism to detect MITM downgrading of negotiated capabilities.

Secure dialect negotiation in SMB 3.0.x

- Dialects 3.0 and 3.02 protect against security-downgrade attacks.
- Mandatory signed FSCTL_VALIDATE_NEGOTIATE_INFO request
- Post-authentication validation after the Tree Connect exchange.
- http://blogs.msdn.com/b/openspecification/archive/2012/06/28/smb3-secure-dialect-negotiation.aspx

Pre-authentication integrity in SMB 3.1.1

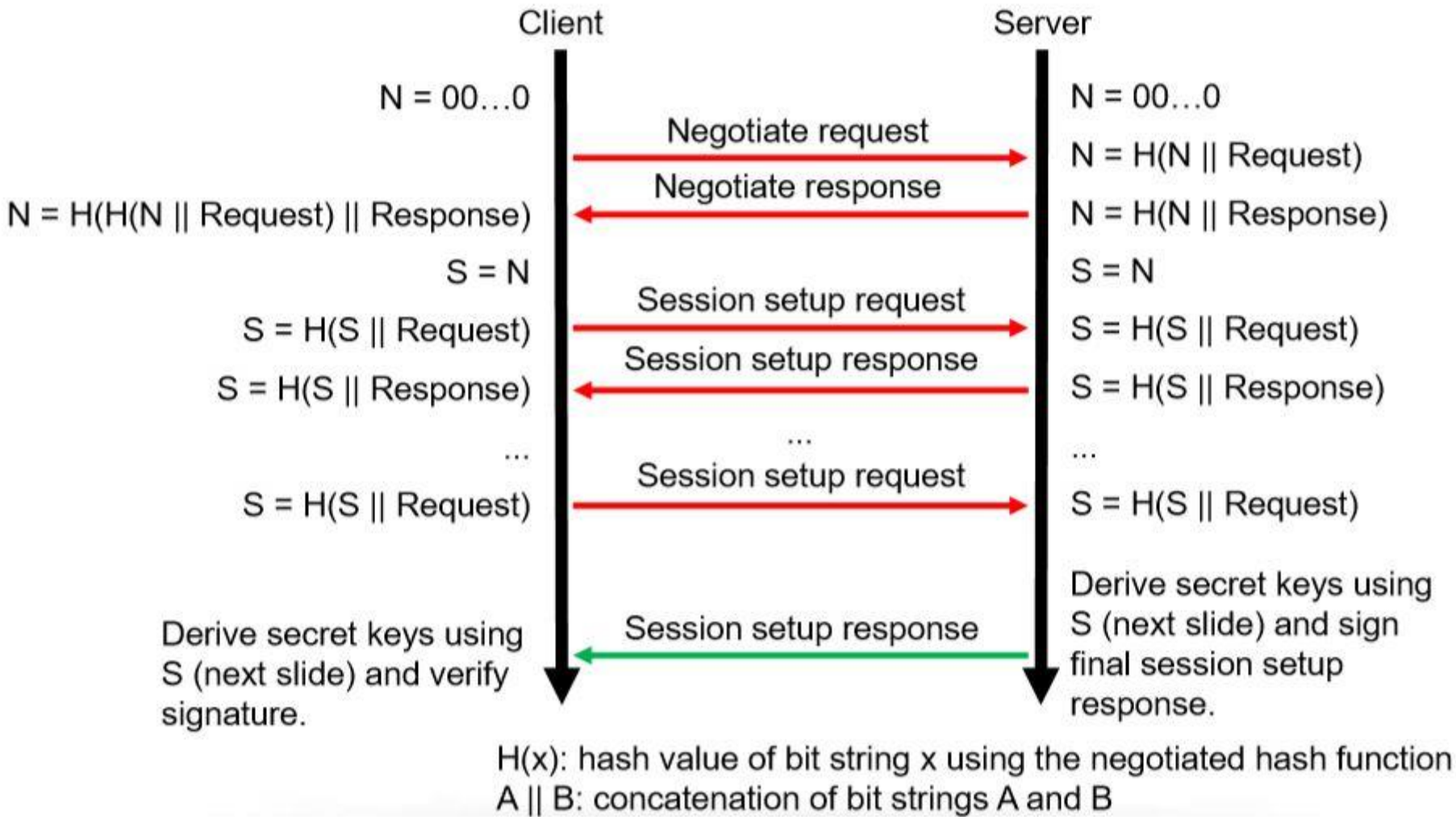
- Pre-auth integrity negotiate context
- Compute the hash value
- Key Derivation
- Multichannel

Pre-authentication integrity overview

- SMB 3.1.1 pre-authentication integrity *supersedes* SMB 3.0.x secure dialect negotiation.

- *End-to-end integrity of pre-authentication messages.*
- Client and server validate signature or decryption of subsequent authenticated messages.
- Final session setup response is always signed.
- Client must sign or encrypt the Tree Connect request.

Compute PreauthIntegrityHashValue



Compute PreauthIntegrityHashValue (cont'd)

- For n-leg GSS-API authentication:
- H (m) = SHA-512 (Concatenate (H (m-1), pre-auth-message (m)))
- m = 1, …, 2n + 1,
- H (0) = 64 zero bytes.
- pre-auth-message (1) = Negotiate request,
- pre-auth-message (2) = Negotiate response,
- pre-auth-message (3) = Session setup request (leg#1),
- . . .
- pre-auth-message (2n + 1) = Session setup request (leg#n).

Compute PreauthIntegrityHashValue (cont'd)

- The hash value used as input for key derivation is:
- Session.PreauthIntegrityHashValue = H (2n + 1)

SMB2_PREAUTH_INTEGRITY_CAPABILITIES negotiate context

- Response context has HashAlgorithmCount = 1
- SHA-512 is the only supported hash function
- Windows 10: Salt is 32 bytes generated with PRNG

SMB2_PREAUTH_INTEGRITY_CAPABILITIES
(Negotiate Context ID: 0x0001)

Byte 0	Byte 1	Byte 2	Byte 3
HashAlgorithmCount		SaltLength	
HashAlgorithms			
…			
Salt			
…			

SMB 3.x key derivation function

- Ko = SMB3KDF (Ki, Label, Context)
- SMB3KDF(): key derivation function (KDF) in Counter Mode, specified in [SP800-108] section 5.1, with 'r' value of 32 and 'L' value of 128, and HMAC-SHA256 as the PRF.
- Ki – The SessionKey is the key derivation key (KDK), used as an input to the KDF, for SMB 3.x.
- Label – the purpose of this derived key, encoded as string and length for SMB 3.x.
- Context – the context information of this derived key, encoded as string and length for SMB 3.x.
- L – An integer that specifies the length of the derived keying material Ko, L is 128 bits for SMB 3.x cryptographic keys. Note that L is a constant since all SMB 3.x keys are 16 bytes in length.
- Ko – Keying material output from the KDF, a binary string of length L, where Ko is the leftmost L bits of KDF result.

Key derivation for dialects 3.0 and 3.0.2

- SMB3KDF (SessionKey, Label, Context)

Derived Key	Label	Context
SigningKey	"SMB2AESCMAC\0"	"Smbign\0"
ApplicationKey	"SMB2APP\0"	"SmbRpc\0"
EncryptionKey (client)	"SMB2AESCCM\0"	"ServerIn \0"
EncryptionKey (server)	"SMB2AESCCM\0"	"ServerOut\0"

- EncryptionKey (client) = DecryptionKey (server)
- EncryptionKey (server) = DecryptionKey (client)

Key derivation for dialects 3.1.1

- SMB3KDF (SessionKey, Label, Context)
  - Context = Session.PreauthIntegrityHashValue)

Derived Key	Label
SigningKey	"SMBSigningKey\0"
ApplicationKey	"SMBAppKey\0"
EncryptionKey (client)	"SMBC2ScipherKey\0"
EncryptionKey (server)	"SMBS2CCipherKey\0"

- EncryptionKey (client) = DecryptionKey (server)
- EncryptionKey (server) = DecryptionKey (client)

SMB 3.x Session binding - Multichannel

- SMB2_SESSION_FLAG_BINDING in Flags of the Session Setup request.
- Server needs to call GSS_Accept_sec_context for each additional channel.
- Get new SessionKey and derive a Channel.SigningKey for the binding channel.
- Session binding request is signed with the master session's SigningKey.
- EncryptionKey, DecryptionKey, and ApplicationKey remain the same from the main channel.

SMB 3.1.1 Session binding and pre-authentication integrity

- Always start with the pre-authentication hash resulting from Negotiate on that connection.

- The client side logic is summarized as follows:
  - Set Session.PreauthIntegrityHashValue to Connection.PreauthIntegrityHashValue.
  - Update Session.PreauthIntegrityHashValue with the SessionSetup request.
  - If it's STATUS_MORE_PROCESSING_REQUIRED, update Session.PreauthIntegrityHashValue with the response, and process subsequent session setup exchange leg.
  - if it's STATUS_SUCCESS, use Session.PreauthIntegrityHashValue to generate cryptographic keys, then verify the signature of the SessionSetup response.

SMB 3.1.1 Session binding and pre-authentication integrity

- For the server side, the logic is similar except that we use:
  - Session.PreauthIntegrityHashValue when SMB2_SESSION_FLAG_BINDING is NOT set, and
  - PreauthSession.PreauthIntegrityHashValue when SMB2_SESSION_FLAG_BINDING is set.
- No guarantee in which order session bindings will be processed. Each binding session setup processing is isolated from the other, but all use the connection's pre-authentication hash as the starting point.

- Steps for SMB 3.x key derivation examples

Key derivation for dialects 3.0 and 3.0.2

- SMB3KDF (SessionKey, Label, Context)

Derived Key	Label	Context
SigningKey	"SMB2AESCMAC\0"	"Smbign\0"
ApplicationKey	"SMB2APP\0"	"SmbRpc\0"
EncryptionKey (client)	"SMB2AESCCM\0"	"ServerIn \0"
EncryptionKey (server)	"SMB2AESCCM\0"	"ServerOut\0"

- EncryptionKey (client) = DecryptionKey (server)
- EncryptionKey (server) = DecryptionKey (client)

Key derivation example for SMB 3.0 multichannel (1/2)

- --- Master session (first channel) ---
- Input cryptographicKey 0x7CD451825D0450D235424E44BA6E78CC
- (queried from GSS authenticated context)
-
- SessionKey 0x7CD451825D0450D235424E44BA6E78CC
- SigningKey 0x0B7E9C5CAC36C0F6EA9AB275298CEDCE
- EncryptionKey 0xFAD27796665B313EBB578F388632B4F7
- DecryptionKey 0xB0F0427F7CEB416D1D9DCC0CD4F99447
- ApplicationKey 0xBB23A4575AA26C721AF525AF15A87B4F

Key derivation example for SMB 3.0 multichannel (2/2)

- --- Second channel (session binding) ---
- Input cryptographicKey 0x4E01A2B313BCF660CC250BEF021AEDE6
- (queried from GSS authenticated context)
- Alternate channel: generate Channel.SigningKey.
- Alternate channel: Copy encryption/decryption keys from main channel
- SessionKey 0x4E01A2B313BCF660CC250BEF021AEDE6
- SigningKey 0xBA1A17DBBFEC349BCA105563D598952F
- EncryptionKey 0xFAD27796665B313EBB578F388632B4F7
- DecryptionKey 0xB0F0427F7CEB416D1D9DCC0CD4F99447
- ApplicationKey 0xBB23A4575AA26C721AF525AF15A87B4F

Key derivation for dialects 3.1.1

- SMB3KDF (SessionKey, Label, Context)
  - Context = Session.PreauthIntegrityHashValue)

Derived Key	Label
SigningKey	"SMBSigningKey\0"
ApplicationKey	"SMBAppKey\0"
EncryptionKey (client)	"SMBC2ScipherKey\0"
EncryptionKey (server)	"SMBS2CCipherKey\0"

- EncryptionKey (client) = DecryptionKey (server)
- EncryptionKey (server) = DecryptionKey (client)

Key derivation example for SMB 3.1.1 multichannel

- Calculate hash value used as input for key derivation:
- Session.PreauthIntegrityHashValue = H (2n + 1)
- Derive the keys

PreauthIntegrityHashValue for each channel

- Master session (first channel)
  - Connection.PreauthIntegrityHashValue
  - PreauthSession.PreauthIntegrityHashValue

- Second channel (session binding)
  - New Connection.PreauthIntegrityHashValue
  - New PreauthSession.PreauthIntegrityHashValue

- Initialization Connection.PreauthIntegrityHashValue

0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000

Negotiate request packet

- FE534D4240000100000000000000080000000000000000000100000000000000FFFE000000000000
- 0000000000000000000000000000000000000000000000000240005000000000003F000000ECD86F32
- 6276024F9F7752B89BB33F3A7000000002000000020210020003020311030000010026000000000
- 010020000100FA49E6578F1F3A9F4CD3E9CC14A67AA884B3D05844E0E5A118225C15887F32FF0000
- 02000600000000000020002000100

Pre-authentication integrity Negotiate context

- 01002600000000000100200001 00FA49E6578F1F3A9F4CD3E9CC14A67A A884B3D05844E0E5A118225C15887F32FF00000200060000000000020002000100
- PreauthIntegrityCaps:
  - HashAlgorithmCount 0x1, SaltLength 0x20, HashAlgorithms 0x0001,
  - Salt FA49E6578F1F3A9F4CD3E9CC14A67AA884B3D05844E0E5A118225C15887F32FF
-
- EncryptionCaps:
  - CipherCount 0x2,
  - Ciphers[0] 0x0002,
  - Ciphers[1] 0x0001

SHA-512 Input Hash Data

- 0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000FE534D4240000100000000000008000000000000000000
  1000000000000FFFE000000000000000000000000000000000000000
  0000000000000000000240005000000000003F000000ECD86F326276024
  F9F7752B89BB33F3A70000000200000002021002000302031103000001
  0026000000000001002000010 0FA49E6578F1F3A9F4CD3E9CC14A67AA8
  84B3D05844E0E5A118225C15887F32FF00000200060000000002000200
  00100

Hash #1

- Connection.PreauthIntegrityHashValue
- DD94EFC5321BB618A2E208BA8920D2F422992526947A409B5037DE1E0FE8C7362B8C47122594CDE0CE26AA9DFC8BCDBDE06219576726233 51A7540F1E54A0426

NEGOTIATE Response

- FE534D4240000100000000000000010001000000000000000100000000000000FFFE000000000000
- 00000000000000000000000000000000000000000000000041000100110302003 9CBCAF329714942
- BDCE5D60F09AB3FB2F0000000000800000008000000008000D8DAE5ADCBAED00109094AB095AED001
- 80004001C00100006082013C06062B0601050502A08201303082012CA01A3018060A2B0601040182
- 3702021E060A2B060104018237020 20AA282010C048201084E45474F455854530100000000000000
- 60000000700000007C7CC0FD06D6362D02DDE1CF343BFE292900F49750B4AA97934D9C4296B26E51
- FD370471B235E15A50DAE15BD5489C8700000000000000006000000001000000000000000000000
- 5C33530DEAF90D4DB2EC4AE3786EC3084E45474F455854530300000001000000400000009800000 0
- 7C7CC0FD06D6362D02DDE1CF343BFE295C33530DEAF90D4DB2EC4AE3786EC3084000000058000000
- 3056A05430523027802530233121301F06035504031318546F6B656E205369676E696E6720507562
- 6C6963204B65793027802530233121301F06035504031318546F6B656E205369676E696E67205075
- 626C6963204B65790100260000000000010020000 10060A3C3B95C3C7CCD51EC536648D9B3AC74C4
- 83CA5B65385A251117BEB30712E500000200040000000000001000200

SHA-512 Input Hash Data

- DD94EFC5321BB618A2E208BA8920D2F422992526947A409B5037DE1E0FE8C7362B8C47122594CDE0
- CE26AA9DFC8BCDBDE062195767262335 1A7540F1E54A0426FE534D42400001000000000000000100
- 0100000000000000010000000000000FFFE0000000000000000000000000000000000000000000
- 00000000000000004100010011030200 39CBCAF329714942BDCE5D60F09AB3FB2F00000000008000
- 00008000000008000D8DAE5ADCBAED00109094AB095AED00180004001C00100006082013C06062B06
- 01050502A08201303082012CA01A3018060A2B06010401823702021E060A2B06010401823702020A
- A282010C048201084E45474F455854530100000000000000600000007000000007C7CC0FD06D6362D
- 02DDE1CF343BFE292900F49750B4AA97934D9C4296B26E51FD370471B235E15A50DAE15BD5489C87
- 00000000000000006000000001000000000000000000000005C33530DEAF90D4DB2EC4AE3786EC308
- 4E45474F455854530300000001000000400000009800000007C7CC0FD06D6362D02DDE1CF343BFE29
- 5C33530DEAF90D4DB2EC4AE3786EC3084000000058000000 3056A05430523027802530233121301F
- 06035504031318546F6B656E205369676E696E67205075626C6963204B65793027802530233121 30
- 1F06035504031318546F6B656E205369676E696E67205075626C6963204B6579010026000000000 0
- 01002000010060A3C3B95C3C7CCD51EC536648D9B3AC74C483CA5B65385A251117BEB30712E50000
- 02000400000000000100020 0

Hash #2

- New Connection.PreauthIntegrityHashValue
- 324BFA92A4F3A190E466EBEA08D9C110DC88BFED758D9846ECC6F541CC1D02AE3C94A79F36011E99
- 7E13F841B91B50957AD07B19C8E2539C0B23FDAE09D2C513

SESSION SETUP Request

- Add NEW SessionId 0x100000000019 to Preauth Integrity hash table with value Connection.PreauthIntegrityHashValue
- PreauthSession.SessionId 0x100000000019
- PreauthSession.PreauthIntegrityHashValue
- 324BFA92A4F3A190E466EBEA08D9C110DC88BFED758D9846ECC6F541CC1D02AE3C94A79F36011E99
- 7E13F841B91B50957AD07B19C8E2539C0B23FDAE09D2C513

- SessionSetup request packet
- FE534D4240000100000000000100800000000000000000000200000000000000FFFE000000000000
- 0000000000000000000000000000000000000000000000190000010100000000000000058004A00
- 0000000000000000604806062B0601050502A03E303CA00E300C060A2B06010401823702020AA22A
- 04284E544C4D535350000100000978208E2000000000000000000000000000000000060380250000
- 000F

SHA-512 Input Hash Data

- 324BFA92A4F3A190E466EBEA08D9C110DC88BFED758D9846ECC6F541CC1D02AE3C94A79F36011E99
- 7E13F841B91B50957AD07B19C8E2539C0B23FDAE09D2C513FE534D4240000100000000001008000
- 0000000000000000020000000000000FFFE0000000000000000000000000000000000000000000
- 0000000000000000190000010100000000000058004A0000000000000000604806062B060105
- 0502A03E303CA00E300C060A2B06010401823702020AA22A04284E544C4D5353500001000009782
- 08E2000000000000000000000000000000000603802500000000F

Hash #3

- PreauthSession.PreauthIntegrityHashValue
- AC0B0F2B9986257700365E416D142A6EDC96DF03594A19E52A15F6BD0D041CD5D432F8ED42C55E33
- 197A50C9EC00F1462B50C592211B1471A04B56088FDFD5F9

SESSION SETUP Response

- STATUS_MORE_PROCESSING_REQUIRED
- PreauthSession.SessionId 0x100000000019

- SessionSetup response packet
- FE534D42400001001600000C001000100010000000000000002000000000000000FFFE000000000000
- 19000000001000000000000000000000000000000000000000090000004800B300A181B03081ADA003
- 0A0101A10C060A2B06010401823702020AA281970481944E544C4D5353500002000000C000C0038
- 00000015828AE20D1D8BA31179D008000000000000000005000500440000000A0092270000000F53
- 0055005400330031003100310002000C0053005500540033003100310001000C0053005500540033003 1
- 00310004000C0053005500540033003100310003000C0053005500540033003100310007000800A1
- A1F5ADCBAED00100000000

SHA-512 Input Hash Data

- AC0B0F2B9986257700365E416D142A6EDC96DF03594A19E52A15F6BD0D041CD5D432F8ED42C55E33
- 197A50C9EC00F1462B50C592211B1471A04B56088FDFD5F9FE534D424000010016000C001000100
- 01000000000000000200000000000000FFFE000000000001900000000100000000000000000000
- 00000000000000000900000004800B300A181B03081ADA0030A0101A10C060A2B06010401823702 02
- 0AA281970481944E544C4D535350000200000000C000C00380000001 5828AE20D1D8BA31179D00800
- 00000000000000500050004400000000A0092270000000F530055005400330031003100020 00C0053
- 0055005400330031003100010 00C0053005500540033003100310004000C0053005500540033 0031
- 00310003000C0053005500540033003100310007000800A1A1F5ADCBAED00100000000

Hash #4

- PreauthSession.PreauthIntegrityHashValue
- 2729E3440DFDDD839E37193F6E8F20C20CEFB3469E453A70CD980EEC06B8835740A7376008563336
- 4C8989895ECE81BF102DEEB14D4B7D48AFA76901A7A38387

SESSION SETUP Request

- FE534D4240000100000000000100800000000000000000000300000000000000FFFE000000000000
- 1900000000100000000000000000000000000000000000000019000001010000000000000005800CF01
- 0000000000000000A18201CB308201C7A0030A0101A28201AA048201A64E544C4D53535000030000
- 00180018009000000EE00EE00A80000000C000C00580000001A001A00640000001200120 07E0000
- 0010001000960100001582 88E2060380250000000FECAC77A5F385A8BF9C38C706EEEDDCD3530055
- 0054003300310031006100 64006D0069006E006900730074007200610074006F0072004400520049
- 0056004500520033003100310000000000000000000000000000000000000000000000000000063078E
- B639FE03E20A231C3AE3BF2308010100000000000000A1A1F5ADCBAED001BC4AD05F223CC90F000000
- 0002000C0053005500540033003100310001000C0053005500540033003100310004000C00530055
- 0054003300310031000300 0C0053005500540033003100310007000800A1A1F5ADCBAED001060004
- 0002000000080030003000000000000000000000000000300000B61FEFCAA857EA57BF1EDCEBF8974B
- 8E0EBA5A6DFD9D07A31D11B548F8C9D0CC0A0010000000000000000000000000000000000090016
- 0063006900660073002F0053005500540033003100310000000000000000000000000003B9BDFF38F
- 5EE8F9663F11A0F4C03A78A3120410010000006 3775A9A5FD97F0600000000

SHA-512 Input Hash Data

- 2729E3440DFDDD839E37193F6E8F20C20CEFB3469E453A70CD980EEC06B8835740A7376008563336 4C8989895ECE81BF102DEEB14D4B7D48AFA76901A7A38387FE534D424000010000000000010080000000000000000000000300000000000000FFFE000000000000190000000010000000000000000000000000000000000000000190000010100000000000005800CF01000000000000000A18201CB308201C7A0030A0101A28201AA048201A64E544C4D535350000300000018001800900000000EE00EE00A80000000C000C00580000001A001A00640000001200120 07E0000001000100096010000158288E206038025000000FECAC77A5F385A8BF9C38C706EEEDDCD3530055005400330031003100610064006D0069006E0069007300740072006100740 06F0072004400520049005600450052003300310031000000000000000000000000000000000000000000000000000063078EB639FE03E20A231C3AE3BF2308010100000000000A1A1F5ADCBAED001BC4AD05F223CC90F0000000002000C0053005500540033003100310001000C0053005500540033003100310004000C0053005500540033003100310003000C0053005500540033003100310007000800A1A1F5ADCBAED001060004000200000008003000300000000000000000000000300000B61FEFCAA857EA57BF1EDCEBF8974B8E0EBA5A6DFD9D07A31D11B548F8C9D0CC0A0010000000000000000000000000000000900160063006900660073002F0053005500540033003100310000000000000000000000003B9BDFF38F5EE8F9663F11A0F4C03A78A3120410010000063775A9A5FD97F0600000000

Hash #5

- PreauthSession.PreauthIntegrityHashValue
- 0DD13628CC3ED218EF9DF9772D436D0887AB9814BFAE63A80AA845F36909DB7928622DDDAD522D9751640A459762C5A9D6BB084CBB3CE6BDADEF5D5BCE3C6C01

SESSION SETUP Response - COMPLETED

- SessionSetup response packet
- FE534D4240000100000000000100800009000000000000000300000000000000FFFE000000000000
- 190000000100000EBE146DA120BA25FC3376A49DFE31BC10900000048001D00A11B3019A0030A01
- 00A312041001000003B453CDC352416420000000

- SessionSetup response header signature 0xEBE146DA120BA25FC3376A49DFE31BC1

- PreauthSession.PreauthIntegrityHashValue
- 0DD13628CC3ED218EF9DF9772D436D0887AB9814BFAE63A80AA845F36909DB7928622DDDAD522D9751640A459762C5A9D6BB084CBB3CE6BDADEF5D5BCE3C6C01

Keys for the master session (first channel)

- Input cryptographicKey (GSS SessionKey) 0x270E1BA896585EEB7AF3472D3B4C75A7
- preauthIntegrityHashValue
- 0DD13628CC3ED218EF9DF9772D436D0887AB9814BFAE63A80AA845F36909DB7928622DDDAD522D9751640A459762C5A9D6BB084CBB3CE6BDADEF5D5BCE3C6C01

- Call KDF and derives the keys:
- SessionKey 0x270E1BA896585EEB7AF3472D3B4C75A7
- SigningKey 0x73FE7A9A77BEF0BDE49C650D8CCB5F76
- EncryptionKey 0x629BCBC54422A0F572B97F45989B6073
- DecryptionKey 0xE2AF0DCEFAC68DA71A0DFBD0D1350D74
- ApplicationKey 0x6D7AD7954E9EC61E907B4D473DC178FF

Second channel (session binding)

- New Connection.PreauthIntegrityHashValue
- New PreauthSession.PreauthIntegrityHashValue

- Initialization Connection.PreauthIntegrityHashValue

0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000

Negotiate Request - SHA-512 Hash Input Data

- PreauthIntegrityCaps.Salt
- 3E38B8B81A9AFB172278056F4898FD09A1B9976A49B2D771161FCD4D40DF7D7D

- SHA-512 Input Hash Data
- 0000000000000000000000000000000000000000000000000000000000000000000000000000000000
- 0000000000000000000000000000000000000000000000FE534D42400001000000000000008000
- 0000000000000000010000000000000FFFE0000000000000000000000000000000000000000000
- 0000000000000000240005000100000003F000000ECD86F326276024F9F7752B89BB33F3A70000000
- 0200000002021002000302031103000001002600000000000100200001003E38B8B81A9AFB172278
- 056F4898FD09A1B9976A49B2D771161FCD4D40DF7D7D0000020006000000000002000200100

Hash #1

- New Connection.PreauthIntegrityHashValue
- F035C2B2BAB116E0DCF6A74E26670604D1BF6DDA065913AF7C30E93C1F025AC3CE2DD44D4DE2652 4A785E5D8E06AF0BE1C74296FEF05B045C3793A12B32C49DF

NEGOTIATE Response - SHA-512 Input Hash Data

- F035C2B2BAB116E0DCF6A74E26670604D1BF6DDA065913AF7C30E93C1F025AC3CE2DD44D4DE26524
- A785E5D8E06AF0BE1C74296FEF05B045C3793A12B32C49DFFE534D4240000100000000000000100
- 0100000000000000010000000000000FFFE0000000000000000000000000000000000000000000
- 00000000000000004100010011030200 39CBCAF329714942BDCE5D60F09AB3FB2F00000000008000
- 00008000000008000A76533AECBAED00109094AB095AED00180004001C00100006082013C06062B06
- 01050502A08201303082012CA01A3018060A2B06010401823702021E060A2B06010401823702020A
- A282010C048201084E45474F455854530100000000000000600000007000000007E7CC0FD06D6362D
- 02DDE1CF343BFE29172BA8482F8B7BE4DBA9F69A8DEBBA8336E453F3EE58983492F66354ED01A5BE
- 00000000000000006000000001000000000000000000000005C33530DEAF90D4DB2EC4AE3786EC308
- 4E45474F455854530300000001000000400000009800000007E7CC0FD06D6362D02DDE1CF343BFE29
- 5C33530DEAF90D4DB2EC4AE3786EC3084000000058000000 3056A05430523027802530233121301F
- 06035504031318546F6B656E205369676E696E67205075626C6963204B65793027802530233121 30
- 1F06035504031318546F6B656E205369676E696E67205075626C6963204B657901002600000000000
- 0100200001002521FE7288989F816EE6D3D6C55921FBEF2CC239789E83B8A0EC29E8D84EBAB80000
- 020004000000000001000200

Hash #2

- Connection.PreauthIntegrityHashValue
- E267AB1AA0403082AA2A9FEB0224AF3EA92E53CAA50A893A9635F0659F93591F81391737E68DB0C9
- AD878C56449C36A6895EBCF435A7D97072C7B596B8AF3817

SESSION SETUP Request - SHA-512 Input Hash Data

- E267AB1AA0403082AA2A9FEB0224AF3EA92E53CAA50A893A9635F0659F93591F81391737E68DB0C9
- AD878C56449C36A6895EBCF435A7D97072C7B596B8AF3817FE534D42400001000000000001008000
- 08000000000000000200000000000000FFFE000000000001900000000010000068BD0C58F613CE13
- 34B1EB51C68D39EA190001010100000000000000058004A000000000000000000604806062B060105
- 0502A03E303CA00E300C060A2B06010401823702020AA22A04284E544C4D5353500001000000009782
- 08E200000000000000000000000000000000000603802500000000F

Hash #3

- PreauthSession.PreauthIntegrityHashValue
- 8346469934A59E951A3F2DA7FA4C2C29F0F6B13A6B0951D4CD5279F8D40FD84FF98157937613C6BE
- 9514582E44344B1710DD5BFCE3BB023D28C6EA512E0ADEBD

SESSION SETUP Response - SHA-512 Input Hash Data

- STATUS_MORE_PROCESSING_REQUIRED
- SHA-512 Input Hash Data
- 8346469934A59E951A3F2DA7FA4C2C29F0F6B13A6B0951D4CD5279F8D40FD84FF98157937613C6BE
- 9514582E44344B1710DD5BFCE3BB023D28C6EA512E0ADEBDFE534D424000010016000C001000100
- 09000000000000000200000000000000FFFE000000000000190000000010000014A71CAE724995E
- 612430E2BE87578C090000004800B300A181B03081ADA0030A0101A10C060A2B06010401823702020
- 0AA281970481944E544C4D535350000200000000C000C00380000015828AE23823DA68E32D2FE100
- 00000000000000500050004400000000A0092270000000F530055005400330031003100310002000C0053
- 00550054003300310031000100 0C00530055005400330031003100040 00C00530055005400330031
- 00310003000C00530055005400330031003100070008004 13B42AECBAED00100000000

Hash #4

- PreauthSession.PreauthIntegrityHashValue
- 6DAD1BA61CAF5FDFBB46D995463FF5780F7248D692E70CE87D8B58B2FBEFD438937E1BCBEC3676F2
- 6F7EE374E169F8AFB17671FB9A47AB88EE2C079DB2B2C7D3

SESSION SETUP Request - SHA-512 Input Hash Data

- 6DAD1BA61CAF5FDFBB46D995463FF5780F7248D692E70CE87D8B58B2FBEFD438937E1BCBEC3676F2
- 6F7EE374E169F8AFB17671FB9A47AB88EE2C079DB2B2C7D3FE534D424000010000000000001008000
- 08000000000000000300000000000000FFFE000000000000190000000001000003CA64529ACAE57B2
- 6DCEC17B14E477D7190001010100000000000000005800CF010000000000000000A18201CB308201C7
- A0030A0101A28201AA048201A64E544C4D5353500003000000180018009000000000EE00EE00A80000
- 000C000C00580000001A001A006400000012001200 7E0000001000100096010000158288E2060380
- 250000000FED00E0F3D901F7B77E98071474F83009530055005400330031003100610064006D0069
- 006E006900730074007200610074006F007200440052004900560045005200330031003100000000
- 0000000000000000000000000000000000000000000A46794DA5D2033965D5339E1A675B79B010100
- 0000000000413B42AECBAED0018E4D343661BA6A75000000000200 0C00530055005400330031003 1
- 0001000C00530055005400330031003100310004000C00530055005400330031003100310003000C00530055
- 00540033003100310007000800413B42AECBAED001060004000200000008003000300000000000000
- 0000000000000300000B61FEFCAA857EA57BF1EDCEBF8974B8E0EBA5A6DFD9D07A31D11B548F8C9D0
- CC0A001000000000000000000000000000000000000000900160063006900660073002F005300550054
- 00330031003100000000000000000000000000000C8C61D67CE69769389133691765 5DDD3A312041001
- 0000005122F070A7314A9B00000000

Hash #5

- PreauthSession.PreauthIntegrityHashValue
- EA3BF912B11CBFEC5B1889E8209614218687F82FA5294521AD3063425E49E88A10BD022124CE2512
- 3BC9111F52D9566BA88BF46344E6063DC5E3FF0389026F6C

SESSION SETUP Response - COMPLETED

- SessionSetup response packet
- FE534D4240000100000000000100800009000000000000000300000000000000FFFE000000000000
- 19000000001000008D604217FFBACF40635BF9D872992150090000048001D00A11B3019A0030A01
- 00A31204100100000CFB046FC5D368E2200000000

- SessionSetup response header signature 0x8D604217FFBACF40635BF9D872992150

- PreauthSession.PreauthIntegrityHashValue
- EA3BF912B11CBFEC5B1889E8209614218687F82FA5294521AD3063425E49E88A10BD022124CE2512
- 3BC9111F52D9566BA88BF46344E6063DC5E3FF0389026F6C

Keys for the second channel

- Input cryptographicKey (GSS SessionKey) 0x84B9DBB730116A8FA6E9889555C265F9
- Alternate channel: generate **Channel.SigningKey.**
- Alternate channel: Copy encryption/decryption keys from main channel
- preauthIntegrityHashValue
- EA3BF912B11CBFEC5B1889E8209614218687F82FA5294521AD3063425E49E88A10BD022124CE2512
- 3BC9111F52D9566BA88BF46344E6063DC5E3FF0389026F6C

- Call KDF and derives the keys:
- SessionKey 0x84B9DBB730116A8FA6E9889555C265F9
- **SigningKey** 0xC962BCA1A9DD1697B030644199705431
- EncryptionKey 0x629BCBC54422A0F572B97F45989B6073
- DecryptionKey 0xE2AF0DCEFAC68DA71A0DFBD0D1350D74
- ApplicationKey 0x6D7AD7954E9EC61E907B4D473DC178FF

Blog white papers

- The main body of my presentation is based on these articles:
-
- SMB 2 and SMB 3 security in Windows 10: the anatomy of signing and cryptographic keys
- https://blogs.msdn.microsoft.com/openspecification/2017/05/26/smb-2-and-smb-3-security-in-windows-10-the-anatomy-of-signing-and-cryptographic-keys/
-
- SMB 3.1.1 Pre-authentication integrity in Windows 10
- https://blogs.msdn.microsoft.com/openspecification/2015/08/11/smb-3-1-1-pre-authentication-integrity-in-windows-10/

Thank You!

Questions?