A Surfeit of SSH Cipher Suites

Martin R. Albrecht, Jean Paul Degabriele, Torben B. Hansen and Kenneth G. Paterson

ACM CCS - 27/10/2016

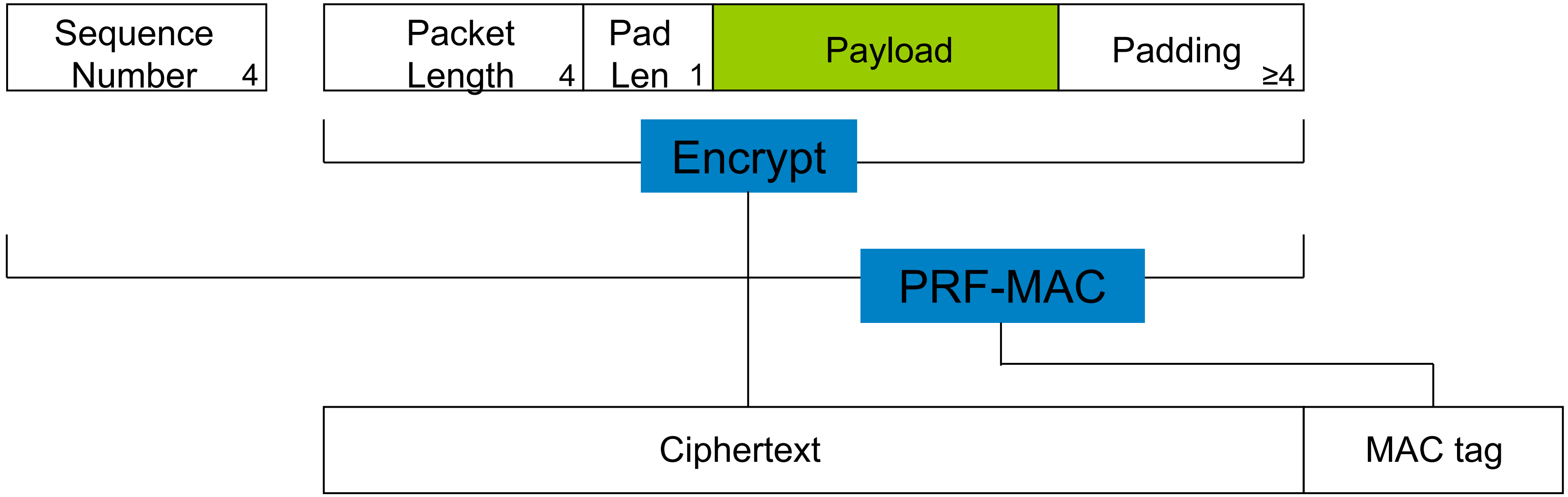
ROYAL HOLLOWAY UNIVERSITY OF LONDON

Outline of this talk

- Overview of SSH and related work.
- SSH deployment statistics.
- A new attack on CBC-mode in OpenSSH.
- Security analysis of ‘new’ OpenSSH AE modes.

Overview of SSH and Related Work

The SSH Binary Packet Protocol (RFC 4253)



- **Encode-then-Encrypt&MAC** construction, stateful because of inclusion of 4-byte sequence number.
- Packet length field measures the size of the packet: |PadLen|+ |Payload| + |Padding|.
- RFC 4253 (2006): various block ciphers in **CBC mode (with chained IV)** and **RC4**.
- RFC 4344 (2006): added **Counter mode** for the corresponding block ciphers.

Timeline of related work on SSH-BPP

2002.

- Formal security analysis of SSH-BPP by Bellare, Kohno and Namprempre [BKN02]. They introduced an **extended security model** and proved **SSH-CTR** and **SSH-CBC variants** (w/o IV chaining) secure.

2009.

- Albrecht, Paterson and Watson [APW09] found a plaintext-recovery attack against **SSH in CBC mode**.
- The leading implementation was OpenSSH (reported 80% of servers), and they released a **patch** in version 5.2 to stop this specific attack on CBC mode.
- The attack exploited **fragmented delivery in TCP/IP**, and worked on **all CBC variants** considered in [BKN02].

Timeline of related work on SSH-BPP

2010.

- The [APW09] attack highlighted a deficiency in the [BKN02] security model.
- Paterson and Watson [PW10] prove SSH-CTR secure in an extended model that captures fragmented delivery of ciphertexts.

2012.

- Boldyreva, Degabriele, Paterson and Stam [BDPS12] study ciphertext fragmentation more generally, addressing limitations in the [PW10] model.
- Furthermore they consider **boundary hiding** and resistance to a special type of **denial of service** attack as additional security requirements.
- Both aspects are inherently related to ciphertext fragmentation and correspond to the SSH design choices of **encrypting** the length field and **validating** its contents.

SSH Deployment Today

SSH deployment today

- We performed a measurement study of SSH deployment.
- We conducted two IPv4 address space scans in Nov/Dec 2015 and Jan 2016 using ZGrab/ZMap.
- Grabbing banners and SSH servers' preferred algorithms.
  - Actual cipher used in a given SSH connection depends on client and server preferences.
- Roughly 2^{24} servers found in each scan.
- Nmap fingerprinting suggests mostly embedded routers and firewalls.

The state of SSH today: SSH versions

software	scan 2015–12		scan 2016–01	
dropbear_2014.66	7,229,491	(42.0%)	8,334,758	(47.0%)
OpenSSH_5.3	2,108,738	(12.3%)	2,133,772	(12.0%)
OpenSSH_6.6.1p1	1,198,987	(7.0%)	1,124,914	(6.3%)
OpenSSH_6.0p1	554,295	(3.2%)	573,634	(3.2%)
OpenSSH_5.9p1	467,899	(2.7%)	500,975	(2.8%)
dropbear_2014.63	422,764	(2.5%)	197,353	(1.1%)
dropbear_0.51	403,923	(2.3%)	434,839	(2.5%)
dropbear_2011.54	383,575	(2.2%)	64,666	(0.4%)
ROSSSH	345,916	(2.0%)	333,992	(1.9%)
OpenSSH_6.6.1	338,787	(2.0%)	252,856	(1.4%)
dropbear_0.46	301,913	(1.8%)	335,425	(1.9%)
OpenSSH_5.5p1	262,367	(1.5%)	272,990	(1.5%)
OpenSSH_6.7p1	261,867	(1.5%)	213,843	(1.2%)
OpenSSH_6.2	255,088	(1.5%)	288,710	(1.6%)
dropbear_2013.58	236,409	(1.4%)	249,284	(1.4%)
dropbear_0.53	217,970	(1.3%)	213,670	(1.2%)
dropbear_0.52	132,668	(0.8%)	136,196	(0.8%)
OpenSSH	110,602	(0.6%)	108,520	(0.6%)
OpenSSH_5.8	88,258	(0.5%)	[illegible]	[illegible]
OpenSSH_5.1	86,338	(0.5%)	[illegible]	[illegible]
OpenSSH_5.3p1	84,559	(0.5%)	0	
OpenSSH_7.1	83,793	(0.5%)	0	

Mostly OpenSSH and dropbear; others less than 5%.

The state of SSH today: SSH versions

software	scan 2015–12		scan 2016–01	
dropbear_2014.66	7,229,491	(42.0%)	8,334,758	(47.0%)
OpenSSH_5.3	2,108,738	(12.3%)	2,133,772	(12.0%)
OpenSSH_6.6.1p1	1,198,987	(7.0%)	1,124,914	(6.3%)
OpenSSH_6.0p1	554,295	(3.2%)	573,634	(3.2%)
OpenSSH_5.9p1	467,899	(2.7%)	500,975	(2.8%)
dropbear_2014.63	422,764	(2.5%)	197,353	(1.1%)
dropbear_0.51	403,923	(2.3%)	434,839	(2.5%)
dropbear_2011.54	383,575	(2.2%)	[illegible]	[illegible]
ROSSSH	345,916	(2.0%)	[illegible]	[illegible]
OpenSSH_6.6.1	338,787	(2.0%)	[illegible]	[illegible]
dropbear_0.46	301,913	(1.8%)	335,425	[illegible]
OpenSSH_5.5p1	262,367	(1.5%)	272,990	[illegible]
OpenSSH_6.7p1	261,867	(1.5%)	213,843	[illegible]
OpenSSH_6.2	255,088	(1.5%)	288,710	[illegible]
dropbear_2013.58	236,409	(1.4%)	249,284	(1.4%)
dropbear_0.53	217,970	(1.3%)	213,670	(1.2%)
dropbear_0.52	132,668	(0.8%)	136,196	(0.8%)
OpenSSH	110,602	(0.6%)	108,520	(0.6%)
OpenSSH_5.8	88,258	(0.5%)	89,144	(0.5%)
OpenSSH_5.1	86,338	(0.5%)	44,170	(0.2%)
OpenSSH_5.3p1	84,559	(0.5%)	0	(0.0%)
OpenSSH_7.1	83,793	(0.5%)	0	(0.0%)

Dropbear at 56-58%. 886k older than version 0.52, so vulnerable to variant of 2009 CBC-mode attack!

The state of SSH today: SSH versions

software	scan 2015–12		scan 2016–01	
dropbear_2014.66	7,229,491	(42.0%)	8,334,758	(47.0%)
OpenSSH_5.3	2,108,738	(12.3%)	2,133,772	(12.0%)
OpenSSH_6.6.1p1	1,198,987	(7.0%)	1,124,914	(6.3%)
OpenSSH_6.0p1	554,295	(3.2%)	573,634	(3.2%)
OpenSSH_5.9p1	467,899	(2.7%)	500,975	(2.8%)
dropbear_2014.63	422,764	(2.5%)	197,353	(1.1%)
dropbear_0.51	403,923	(2.3%)	434,839	(2.5%)
dropbear_2011.54	383,575	(2.2%)	64,666	(0.4%)
ROSSSH	345,916	(2.0%)	333,992	(1.9%)
OpenSSH_6.6.1	338,787	(2.0%)	252,856	(1.4%)
dropbear_0.46	301,913	(1.8%)	335,425	(1.9%)
OpenSSH_5.5p1	262,367	(1.5%)	272,990	(1.5%)
OpenSSH_6.7p1	261,867	(1.5%)	213,843	(1.2%)
OpenSSH_6.2	255,088	(1.5%)	288,710	(1.6%)
dropbear_2013.58	236,409	(1.4%)	249,2[illegible]	[illegible]
dropbear_0.53	217,970	(1.3%)	213,[illegible]	[illegible]
dropbear_0.52	132,668	(0.8%)	136,1[illegible]	[illegible]
OpenSSH	110,602	(0.6%)	108,5[illegible]	[illegible]
OpenSSH_5.8	88,258	(0.5%)	89,1[illegible]	[illegible]
OpenSSH_5.1	86,338	[illegible]	[illegible]	[illegible]
OpenSSH_5.3p1	84,559	(0.5%)	[illegible]	[illegible]
OpenSSH_7.1	83,793	(0.5%)	[illegible]	[illegible]

OpenSSH at 37-39%.
130-166k older than version 5.2 and prefer CBC mode, so vulnerable to 2009 attack!

The state of SSH today: preferred algorithms

encryption and mac algorithm	count	
aes128-ctr + hmac-md5	3,877,790	(57.65%)
aes128-ctr + hmac-md5-etm@	2,010,936	(29.90%)
aes128-ctr + umac-64-etm@	331,014	(4.92%)
aes128-cbc + hmac-md5	161,624	(2.40%)
chacha20-poly1305@	115,526	(1.72%)
aes128-ctr + hmac-sha1	68,027	(1.01%)
des + hmac-md5	40,418	(0.60%)
aes256-gcm@	28,019	(0.42%)
aes256-ctr + hmac-sha2-512	17,897	(0.27%)
aes128-cbc + hmac-sha1	11,082	(0.16%)
aes128-ctr + hmac-ripemd160	10,621	(0.16%)

OpenSSH preferred algorithms (@ stands for @openssh.com)

- Lots of diversity (155 combinations).
- CTR dominates, followed by CBC, surprising amount of EtM.
- ChaCha20-Poly1305 on the rise? (became default in OpenSSH 6.9).
- Small amount of GCM.

The state of SSH today: preferred algorithms

encryption and mac algorithm	count	
aes128-ctr + hmac-sha1-96	8,724,863	(90.44%)
aes128-cbc + hmac-sha1-96	478,181	(4.96%)
3des-cbc + hmac-sha1	321,492	(3.33%)
aes128-ctr + hmac-sha1	62,465	(0.65%)
aes128-ctr + hmac-sha2-256	36,150	(0.37%)
aes128-cbc + hmac-sha1	14,477	(0.15%)

Dropbear preferred algorithms

- Less diversity than OpenSSH.
- CTR also dominates, followed by CBC.
- No "exotic" options.

An Attack on Patched OpenSSH with CBC

The [APW09] Attack (simplified)

- Decryption in OpenSSH:
  - The first block of a packet to be received is decrypted and the length field LF is extracted.
  - It is then checked that $5 \leq LF \leq 2^{18}$, and if not an error is sent.
  - If the test passes, it waits until LF bytes are received and then verifies the MAC.
- The number of bytes sent until a “MAC invalid” error is observed leaks the value of LF.
- Any intercepted ciphertext block can be sent as the first block, if successful the attack will recover its first 4 bytes.

The OpenSSH 5.2 patch

- Basic idea: make errors independent of LF.
  - If the length check fails, do not send an error message, but wait until 2^{18} bytes have arrived, then check the MAC.
  - If the length checks pass, but the MAC check eventually fails, then wait until 2^{18} bytes have arrived, then check the MAC.
- No error message is ever sent until 2^{18} bytes of ciphertext have arrived.
- Can no longer count bytes to see how many are required to trigger MAC failure.

However an attack is still possible…

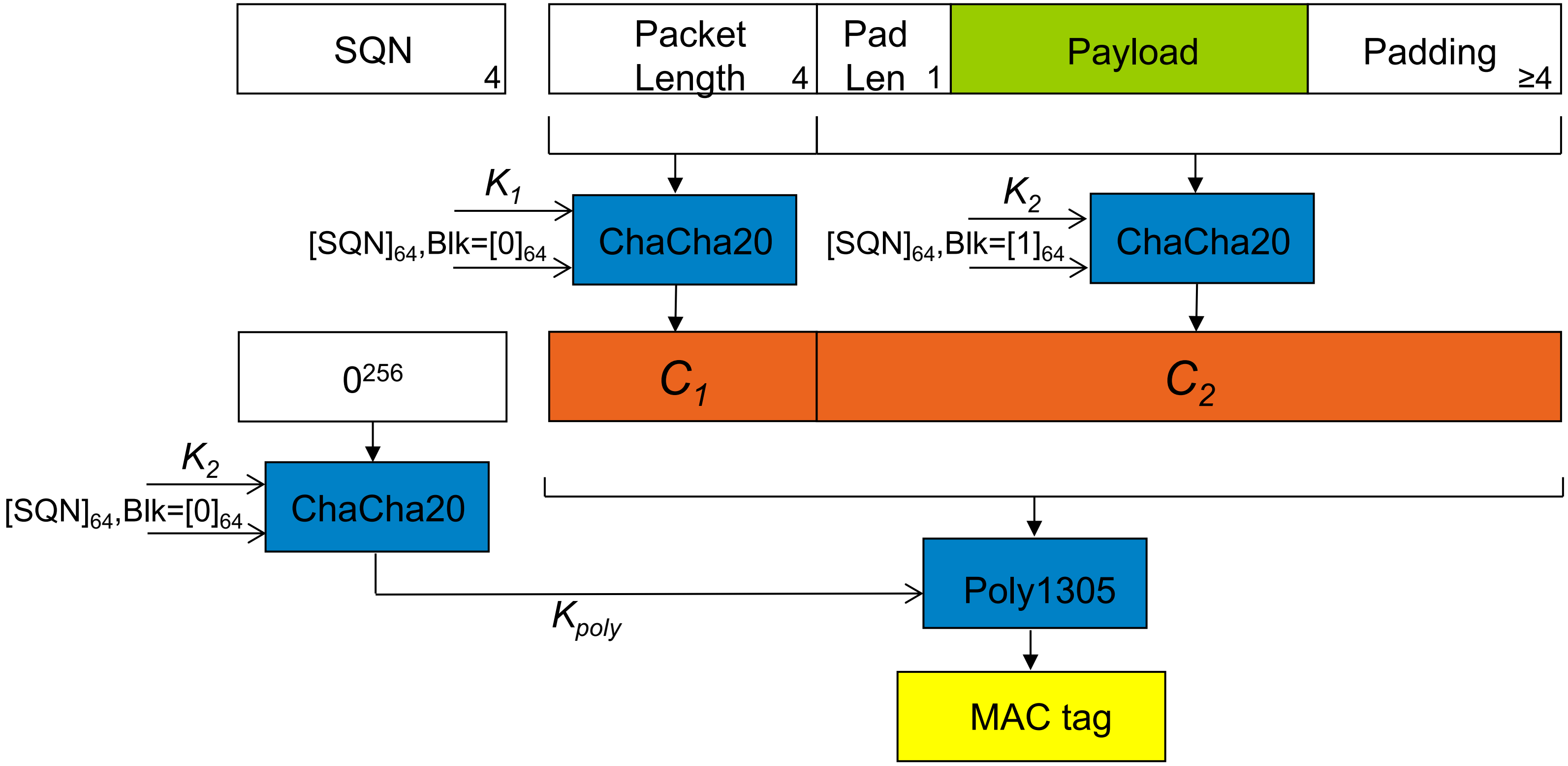
- One MAC check is done if length check fails: on 2^{18} bytes.
- Two MAC checks are done if length checks pass: one on roughly LF bytes, the other on 2^{18} bytes.
- This leads to a timing attack which verifiably recovers 18 bits with success probability 2^{-18}.
- Up to 30 bits may be recovered with more fine-grained timing information.
- Version 5.2 + CBC mode preferred by roughly 20k OpenSSH servers.

Security Analysis of OpenSSH AE Modes

OpenSSH authenticated encryption modes

- Since [APW09] a number of new schemes have been introduced in OpenSSH.
- AES-GCM: since v6.2; **length field is not encrypted** but is instead treated as associated data.
- generic Encrypt-then-MAC (gEtM): since v6.2; overrides native E&M processing; length field also not encrypted but covered by the MAC.
- ChaCha20-Poly1305@openssh.com: since v6.5 and promoted to default in v6.9; **reintroduces encryption of the length field**.

ChaCha20-Poly1305@openssh.com



Security analysis in the presence of fragmentation

- We used the **framework of [BDPS12]** to analyse the security of these schemes.
- We identified and fixed a **technical issue** in the IND-sfCFA confidentiality definition.
- Introduced a matching notion of **ciphertext integrity**, INT-sfCTXT, which was not considered in [BDPS12].
- We made an effort to reflect closely the OpenSSH code.
- **Issue in gEtM**: retrofitted in legacy E&M code - the MAC is computed once the ciphertext has arrived but is not compared to received MAC until *after* decryption!

Security analysis of ChaCha20-Poly1305 in OpenSSH

	IND-sfCFA	INT-sfCTF	BH-CPA	BH-sfCFA	n-DOS-sfCFA
CBC	✗	✓	✓	✗	✗
fixed-CBC	✗	✓	✓	✗	✗
CTR	✓	✓	✓	✗	✗
fgEtM	✓	✓	✗	✗	✗
AES-GCM	✓	✓	✗	✗	✗
ChaCha20-Poly1305	✓	✓	✓	✗	✗

Security comparison of SSH AE modes

- BH-CPA (passive adversary), BH-sfCFA (active adversary).
- n-DOS-sfCFA: inability to produce n-bit sequence of fragments that produces no output (w/o limiting max packet size to n).

Concluding Remarks

Concluding Remarks

- We notified the OpenSSH team of our new attack on CBC and the problem in generic EtM.
- Both issues were addressed in OpenSSH v7.3, released in August 2016.
- None of the schemes in use possesses all security properties that one may consider desirable for SSH.
- Yet such schemes do exist, e.g. InterMAC from [BDPS12].

The End – Thank You