Limits of Proofs: Models / Plaintext Recovery Attacks Against SSH

Advanced Topics in ~~Cybersecurity~~Cryptography (7CCSMATC)

Martin R. Albrecht

Outline

Introduction

Programme

To prove that some construction achieves a targeted security notion, we make models.
What happens if these models are wrong?

Main Reference

Martin R. Albrecht, Kenneth G. Paterson, and Gaven J. Watson. Plaintext Recovery Attacks against SSH. In: *2009 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2009, pp. 16–26. DOI: 10.1109/SP.2009.5

SSH

SSH

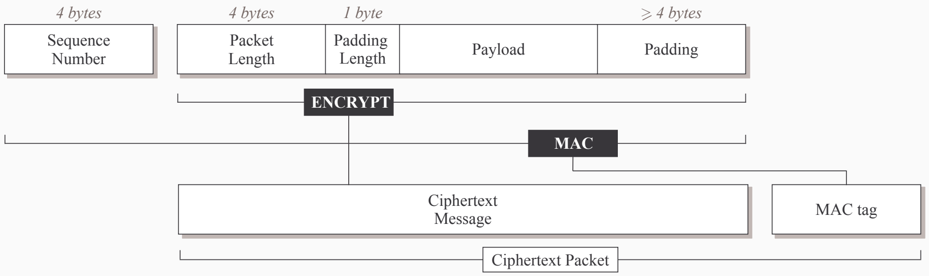
"The Secure Shell Protocol (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Its most notable applications are remote login and command-line execution." — Wikipedia

SSH RFCs

- SSHv1 had several design flaws.
- SSHv2 standardised in 2006 by the IETF in RFCs 4251-4254.
- RFC 4253 specifies the SSH Binary Packet Protocol (BPP).
    - Symmetric encryption and integrity protection for SSH packets, using keys agreed in an earlier exchange.
- SSHv2 is widely regarded as being secure.[1]
- A minor flaw in the BPP that allows distinguishing attacks.[2]

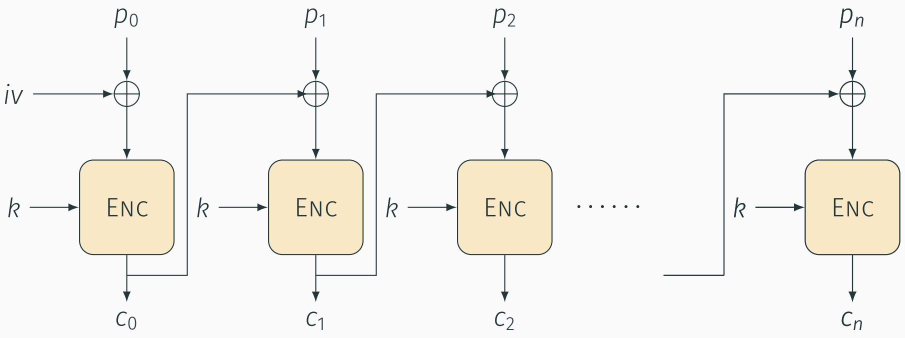
[1] It now is secure, everything discussed here is ancient.
[2] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated Encryption in SSH: Provably Fixing The SSH Binary Packet Protocol. In: *ACM CCS 2002*. Ed. by Vijayalakshmi Atluri. ACM Press, Nov. 2002, pp. 1–11. DOI: `10.1145/586110.586112`.

SSH Binary Packet Protocol



- `Encode-then-Encrypt&MAC` construction.
- Packet length field measures total size of packet on the wire in bytes and is encrypted to hide true length of SSH packets.
- RFC 4253 mandates 3DES-CBC and recommends AES-CBC.

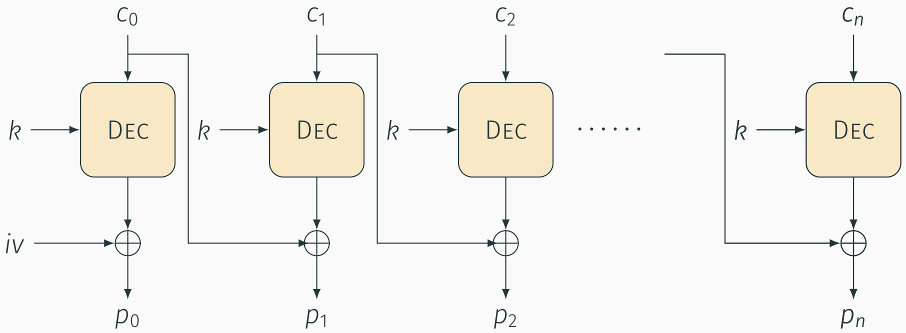
CBC Mode – Encryption



$$c_i := E_K(p_i \oplus c_{i-1})$$

- SSH uses a chained IV in CBC mode.
- IV for this packet is the last ciphertext block from the previous packet.

CBC Mode – Decryption



$$p_i := E_k^{-1}(c_i) \oplus c_{i-1}$$

Attacking SSH

What Makes the Attack Possible

The attacks exploit an interaction of the following SSH features:

- The packet length field encodes how much data needs to be received before the MAC can be checked; this field is encrypted.
- The attacker can send data on an SSH connection in small chunks (TCP).
- CBC mode is likely to be used because support is mandated.
- A MAC failure is visible on the network.

The Attack

$$\text{Client} \;\Big|\; \longrightarrow \qquad c_1^* \ldots c_{i-1}^* c_i^* \ldots c_n^* \qquad \longrightarrow \;\Big|\; \text{Server}$$

The Attack

$$\begin{array}{r|ccc|c} \text{Client} & \longrightarrow & c_1^* \ldots c_{i-1}^* c_i^* \ldots c_n^* & \longrightarrow & \text{Server} \\ & & \text{pick target block } c_i^* & & \end{array}$$

The Attack

Client	$\longrightarrow$	$c_1^* \ldots c_{i-1}^* c_i^* \ldots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'

The Attack

Client	$\longrightarrow$	$c_1^* \ldots c_{i-1}^* c_i^* \ldots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'
				retrieve length field

The Attack

Client	$\longrightarrow$	$c_1^* \dots c_{i-1}^* c_i^* \dots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'
				retrieve length field
		r	$\longrightarrow$	wait state

The Attack

Client	$\longrightarrow$	$c_1^* \dots c_{i-1}^* c_i^* \dots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'
				retrieve length field
		r	$\longrightarrow$	wait state
		$\vdots$	$\vdots$	wait state
		$\vdots$	$\vdots$	wait state

The Attack

Client	$\longrightarrow$	$c_1^* \dots c_{i-1}^* c_i^* \dots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'
				retrieve length field
		r	$\longrightarrow$	wait state
		$\vdots$	$\vdots$	wait state
		$\vdots$	$\vdots$	wait state
		r	$\longrightarrow$	MAC check

The Attack

Client	$\longrightarrow$	$c_1^* \dots c_{i-1}^* c_i^* \dots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'
				retrieve length field
		r	$\longrightarrow$	wait state
		$\vdots$	$\vdots$	wait state
		$\vdots$	$\vdots$	wait state
		r	$\longrightarrow$	MAC check
	$\longleftarrow$	$\perp$	$\longleftarrow$	MAC failure

The Attack

Client	$\longrightarrow$	$c_1^* \ldots c_{i-1}^* c_i^* \ldots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'
				retrieve length field
		r	$\longrightarrow$	wait state
		$\vdots$	$\vdots$	wait state
		$\vdots$	$\vdots$	wait state
		r	$\longrightarrow$	MAC check
	$\longleftarrow$	$\perp$	$\longleftarrow$	MAC failure

The number of random blocks R sent before a MAC error occurred allows us to deduce the packet length field (the first 32 bits) of p'.

The Attack

Client	$\longrightarrow$	$c_1^* \ldots c_{i-1}^* c_i^* \ldots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'
				retrieve length field
		r	$\longrightarrow$	wait state
		$\vdots$	$\vdots$	wait state
		$\vdots$	$\vdots$	wait state
		r	$\longrightarrow$	MAC check
	$\longleftarrow$	$\perp$	$\longleftarrow$	MAC failure

The number of random blocks R sent before a MAC error occurred allows us to deduce the packet length field (the first 32 bits) of p'.

$$p_i^* = E_k^{-1}(c_i^*) \oplus c_{i-1}^*$$

The Attack

Client	$\longrightarrow$	$c_1^* \ldots c_{i-1}^* c_i^* \ldots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'
				retrieve length field
		r	$\longrightarrow$	wait state
		$\vdots$	$\vdots$	wait state
		$\vdots$	$\vdots$	wait state
		r	$\longrightarrow$	MAC check
	$\longleftarrow$	$\perp$	$\longleftarrow$	MAC failure

The number of random blocks R sent before a MAC error occurred allows us to deduce the packet length field (the first 32 bits) of p'.

$$p_i^* = E_k^{-1}(c_i^*) \oplus c_{i-1}^* \qquad p' = E_k^{-1}(c_i^*) \oplus iv$$

The Attack

Client	$\longrightarrow$	$c_1^* \dots c_{i-1}^* c_i^* \dots c_n^*$	$\longrightarrow$	Server
		pick target block c_i^*		
		c_i^*	$\longrightarrow$	new packet p'
				retrieve length field
		r	$\longrightarrow$	wait state
		$\vdots$	$\vdots$	wait state
		$\vdots$	$\vdots$	wait state
		r	$\longrightarrow$	MAC check
	$\longleftarrow$	$\perp$	$\longleftarrow$	MAC failure

The number of random blocks R sent before a MAC error occurred allows us to deduce the packet length field (the first 32 bits) of p'.

$$p_i^* = E_k^{-1}(c_i^*) \oplus c_{i-1}^* \qquad p' = E_k^{-1}(c_i^*) \oplus iv \qquad p_i^* = p' \oplus iv \oplus c_{i-1}^*$$

Performance of Attack

- The attack recovers 32 bits of plaintext with probability 1.
    - But requires the injection of about 2^{27} random blocks to trigger the MAC check.
    - It would also lead to an SSH connection tear-down.
- In practice, things are a bit more complicated ...

Length Checks

According to RFC 4253:

- "Note that the '`packet_length`' field is also encrypted, and processing it requires special care when sending or receiving packets."
- "Implementations SHOULD decrypt the length after receiving the first 8 (or cipher block size, whichever is larger) bytes of a packet."
- "...implementations SHOULD check that the packet length is reasonable"

Attacking OpenSSH

OpenSSH

- OpenSSH is the most popular implementations of the SSH RFCs.
    - OpenSSH reported that OpenSSH accounts for more than 80% of all deployed SSH servers, at the time of the attack[3]
- The attacks apply up to OpenSSH Version 5.1.
    - Version 5.2 was released 23 Feb 2009

[3] www.openssh.org/usage/index.html

Length Checks in OpenSSH

- OpenSSH performs two separate length checks
    - Packet length check
    - Block length check
- If both these checks pass then the MAC will be checked.
- If any of these three checks fails, the SSH connection is terminated in subtly different ways.

Packet Length Check

```
if (packet_length < 1 + 4 || packet_length > 256 * 1024) {
    buffer_dump(&incoming_packet);
    packet_disconnect("Bad packet length  %d.",packet_length);
 }
```

- Maximum possible length field allowed by OpenSSH is 2^{18}.
    - Recall that the length field is 32 bits in length.
    - So 14 most significant bits of length field must equal 0.
    - Chances for random 32-bit length field: roughly 2^{-14}.
- `packet_disconnect` function sends an error message then tears down the SSH connection.

Block Length Check

```
if (need % block_size != 0)
    fatal("padding error:%need %d block %d mod %d", need, block_size, need % block_size);
```

Here, `need = 4 + packet_length - block_size`

- This test checks that the amount of data we expect to receive is a multiple of the blocksize.
- If the check passes, then the 4 least significant bits of length field equal 12 (128-bit block cipher).
- Chances for random 32-bit length field: 2^{-4}.
- `fatal` function tears down SSH connection without sending an error message.

Wait State

- If the two length checks pass then OpenSSH 5.1 enters a wait state.
- It accepts incoming blocks into a buffer until enough blocks have arrived.

```
if (buffer_len(&input) < need + maclen)
   return SSH_MSG_NONE;
```

- Once enough data has arrived, MAC is checked.
- MAC failure results in a call to the `packet_disconnect` function, which results in an error message and a session tear down.

Attacks Against OpenSSH

These length checks give us two attacks:

- With probability 2^{-14} can verifiably retrieve 14 bits of plaintext.
    - This attack works due to the differing behaviour of the packet length check and block length check.
- With probability 2^{-18} we can verifiably retrieve 32 bits of plaintext.
    - This attack requires both checks to pass.
    - The system is then in a wait state and we can proceed with our attack as originally described.

Vulnerability Announcement and Reactions

CPNI Vulnerability Announcement

- We worked with the UK Centre for Protection of National Infrastructure (CPNI)[4] to disclose the attacks.
- In this announcement CPNI took up our suggestion to recommend a switch to counter mode (CTR) encryption.

[4]This role is now part of NCSC.

Vendor Reactions

SunSSH	increased the version number because of a security vulnerability “for the first time”. However, it seems they only addressed the 2^{-14} attack.
SSH.com	acknowledged that their products were vulnerable and claim to have addressed the issue.
Bitvise	acknowledged that their WinSSHD product was vulnerable and issued an update which successfully prevents the attacks: randomisation of length field after failure of sanity checking.
Dropbear	added support for counter mode.

Overview

Partial list of affected vendors/products: `http://www.kb.cert.org/vuls/id/958563`

OpenSSH Reactions

- OpenSSH published a statement and committed a first fix (21/11/2008):
  - Both the statement and the bugfix address only the 2^{-14} attack.
- Then OpenSSH released OpenSSH 5.2 (23/02/2009).
  - Changes order of ciphers to place AES in counter mode and `arcfour256` stream cipher ahead of CBC mode block ciphers.

Reflection on Disclosure

“We recommend avoiding any intermediaries in the disclosure process, since they may impose their own policies with negative consequences. For example, our disclosure of the previously mentioned plaintext recovery attack against (Open)SSH [APW09] was hampered by us going via the UK’s Centre for the Protection of National Infrastructure, resulting in the following notification on the OpenSSH project webpages:

The OpenSSH team has been made aware of an attack against the SSH protocol version 2 by researchers at the University of London. Unfortunately, due to the report lacking any detailed technical description of the attack and CPNI’s unwillingness to share necessary information, we are unable to properly assess its impact.“

Martin R. Albrecht and Kenneth G. Paterson. Analysing Cryptography in the Wild - A Retrospective. Cryptology ePrint Archive, Report 2024/532. 2024. URL: `https://eprint.iacr.org/2024/532`

Security Models

Formal Security Analysis

A formal security analysis of SSH was performed in Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated Encryption in SSH: Provably Fixing The SSH Binary Packet Protocol. In: *ACM CCS 2002*. Ed. by Vijayalakshmi Atluri. ACM Press, Nov. 2002, pp. 1–11. DOI: `10.1145/586110.586112`

- Their work begins by describing attacks against SSH as it was described in RFCs.
- They proposed variants of the SSH BPP and proved them to be IND-CCA secure.
- But the attacks described here still applied to one of the provably secure alternatives.

Provably Secure SSH Alternatives

- SSH-$NPC is SSH with a randomised, per-packet IV with a random padding field.
    - Attacks would still apply to the obvious implementation of this variant.
    - An iterated attack version would be more efficient since we now have control over the IV.
- [BKN02] also propose SSH-CTR, SSH-CTRIV-CBC and SSH-EIV-CBC variants of the BPP, and prove them all secure.
    - Our attacks do not apply to any of these variants.
    - This does not constitute a proof of security against attacks of the type discussed here.

Security Model vs Implementations

The security model in [BKN02] only allows for a single type of error output on decryption.

- The 2^{-14} attack against OpenSSH exploits the fact that the errors are distinguishable.
- Even allowing only one error type, there is a very simple chosen plaintext distinguishing attack having probability one against SSH-\$NPC!

Security Model vs the Standard

- The security model of [BKN02] assumes that ciphertexts and plaintexts are handled as "atomic" strings.
- Recall the following excerpt from the standard:
    - "Implementations SHOULD decrypt the length after receiving the first 8 (or cipher block size, whichever is larger) bytes of a packet."
- As we have seen the packet length field is used during decryption and its treatment is critical to the security of the system.
- A correct security model therefore needs to take into account that the amount of data needed to complete the decryption process is governed by data produced during the decryption process.

Closing the Gaps

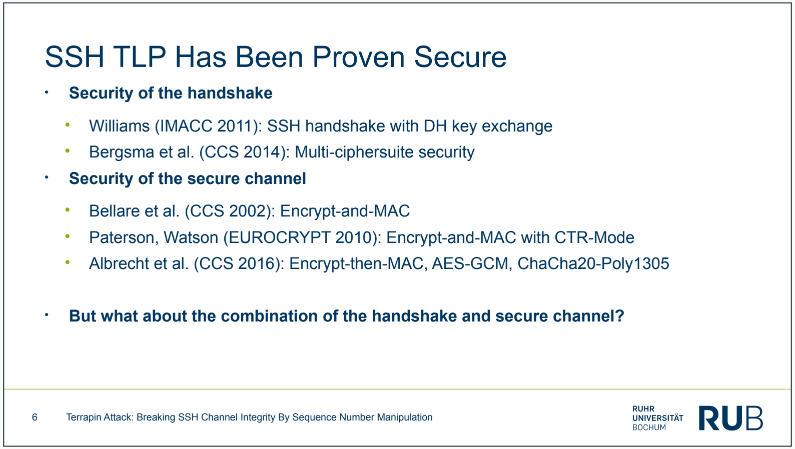
- Our attack on SSH illustrates the gap between security models and:
  - What the specifications actually require a protocol to do.
  - How a protocol is actually implemented.
- This problem is not unique to SSH:
  - Krawczyk gave a proof of security for the SSL/TLS Record Layer [Kra01]
  - Padding is not considered in his model but is required by the specifications.
  - A padding oracle attack against OpenSSL is given in [CHVV03]

Follow Ups

- [PW10] analysed SSH's use of CTR-mode, showing that it achieves security in a prototypical model supporting ciphertext fragmentation.
- The paper inspired the more general and mature treatment of symmetric encryption supporting fragmented decryption in [BDPS12].
  - That work introduced general notions formalising confidentiality against fragmentation attacks.
  - It also introduced a scheme (InterMAC) meeting additional security goals in a model supporting ciphertext fragmentation: DOS resistance and length hiding.
- [ADHP16] proved popular modes of SSH (`ChaCha20-Poly1305`, `gEtM` and `AES-GCM`) secure, also a model supporting ciphertext fragmentation.[5]
- [AHP19] implemented and benchmarked InterMAC.

[5]That attack also reported an attack against the fixed CBC implementation in OpenSSH.

Close … But no Cigar I



SSH TLP Has Been Proven Secure

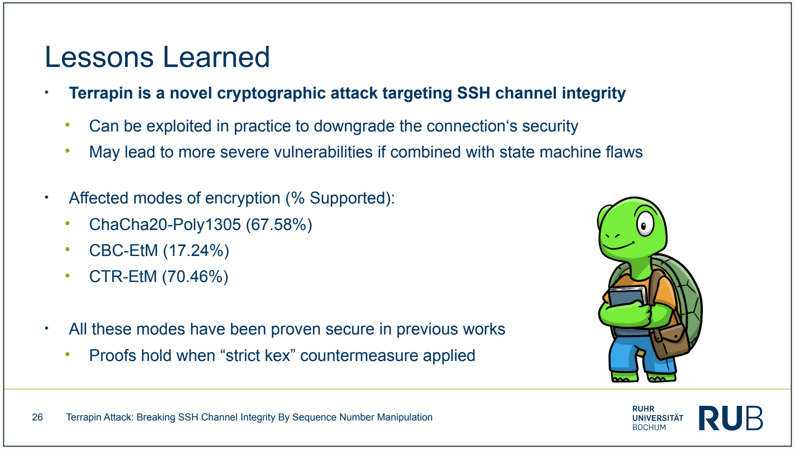
- **Security of the handshake**
  - Williams (IMACC 2011): SSH handshake with DH key exchange
  - Bergsma et al. (CCS 2014): Multi-ciphersuite security
- **Security of the secure channel**
  - Bellare et al. (CCS 2002): Encrypt-and-MAC
  - Paterson, Watson (EUROCRYPT 2010): Encrypt-and-MAC with CTR-Mode
  - Albrecht et al. (CCS 2016): Encrypt-then-MAC, AES-GCM, ChaCha20-Poly1305
- **But what about the combination of the handshake and secure channel?**

6 Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation

RUHR UNIVERSITÄT BOCHUM RUB

Fabian Bäumer, Marcus Brinkmann, and Jörg Schwenk. Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation. In: *USENIX Security 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, Aug. 2024

Close ... But no Cigar II



Lessons Learned

- **Terrapin is a novel cryptographic attack targeting SSH channel integrity**
  - Can be exploited in practice to downgrade the connection‘s security
  - May lead to more severe vulnerabilities if combined with state machine flaws
- Affected modes of encryption (% Supported):
  - ChaCha20-Poly1305 (67.58%)
  - CBC-EtM (17.24%)
  - CTR-EtM (70.46%)
- All these modes have been proven secure in previous works
  - Proofs hold when “strict kex” countermeasure applied

26 Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation

RUHR UNIVERSITÄT BOCHUM RUB

Fabian Bäumer, Marcus Brinkmann, and Jörg Schwenk. Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation. In: *USENIX Security 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, Aug. 2024

Fin

... A PROOF IN ”THE STANDARD MODEL”

IS STILL A PROOF IN A MODEL

References I

[ADHP16] Martin R. Albrecht, Jean Paul Degabriele, Torben Brandt Hansen, and Kenneth G. Paterson. A Surfeit of SSH Cipher Suites. In: *ACM CCS 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM Press, Oct. 2016, pp. 1480–1491. DOI: `10.1145/2976749.2978364`.

[AHP19] Martin R. Albrecht, Torben Brandt Hansen, and Kenneth G. Paterson. libInterMAC: Beyond Confidentiality and Integrity in Practice. In: *IACR Trans. Symm. Cryptol.* 2019.1 (2019), pp. 46–83. ISSN: 2519-173X. DOI: `10.13154/tosc.v2019.i1.46-83`.

[AP24] Martin R. Albrecht and Kenneth G. Paterson. Analysing Cryptography in the Wild - A Retrospective. Cryptology ePrint Archive, Report 2024/532. 2024. URL: `https://eprint.iacr.org/2024/532`.

References II

[APW09] Martin R. Albrecht, Kenneth G. Paterson, and Gaven J. Watson. **Plaintext Recovery Attacks against SSH**. In: *2009 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2009, pp. 16–26. DOI: `10.1109/SP.2009.5`.

[BBS24] Fabian Bäumer, Marcus Brinkmann, and Jörg Schwenk. **Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation**. In: *USENIX Security 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, Aug. 2024.

[BDPS12] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. **Security of Symmetric Encryption in the Presence of Ciphertext Fragmentation**. In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Berlin, Heidelberg, Apr. 2012, pp. 682–699. DOI: `10.1007/978-3-642-29011-4_40`.

References III

[BKN02] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated Encryption in SSH: Provably Fixing The SSH Binary Packet Protocol. In: *ACM CCS 2002*. Ed. by Vijayalakshmi Atluri. ACM Press, Nov. 2002, pp. 1–11. DOI: 10.1145/586110.586112.

[CHVV03] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password Interception in a SSL/TLS Channel. In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Berlin, Heidelberg, Aug. 2003, pp. 583–599. DOI: 10.1007/978-3-540-45146-4_34.

[Kra01] Hugo Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?) In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Berlin, Heidelberg, Aug. 2001, pp. 310–331. DOI: 10.1007/3-540-44647-8_19.

References IV

[PW10] Kenneth G. Paterson and Gaven J. Watson. Plaintext-Dependent Decryption: A Formal Security Treatment of SSH-CTR. In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Berlin, Heidelberg, 2010, pp. 345–361. DOI: `10.1007/978-3-642-13190-5_18`.