Solutions to Selected Problems

Guide to Internet Cryptography

Companion Material

February 7, 2026

Preface

This document provides solutions to selected problems from the book *Guide to Internet Cryptography: Security Protocols and Real-World Attack Implications*. The material is intended for educational use in courses and self-study.

Book website: https://link.springer.com/book/10.1007/978-3-031-19439-9

1 Chapter 5: Point-to-Point Security

Problem 5.1 PPTPv1: Dictionary Attack

Suppose all PPTP clients use randomly generated strong passwords from the character set `ascii-32-65-123-4`. Can you calculate the size of a complete dictionary for an attack?

Solution

The definition of the set `ascii-32-65-123-4` can be found at http://project-rainbowcrack.com/charset.txt. It includes all the 95 characters available on a standard US keyboard, minus the lowercase characters. Subtracting the 26 lowecase characters from 95, this gives 69 characters. Choosing one character from this set randomly, for each of the 7 positions, results in

$$69^7 \approx 7.4 \cdot 10^{12}$$

possible words in the dictionary. This dictionary can be used on the left and the right DES computation.

Problem 5.2 PPTPv1: Rainbow Tables

Is there a difference in using weak or strong passwords when considering rainbow table-based attacks? Why are rainbow tables so efficient in calculating LMH passwords?

Solution

No, there is NO difference between weak and strong passwords against rainbow table attacks – at least not in the traditional sense of "weak" vs "strong."
Rainbow tables are **pre-computed** hash chains. If a password is in the rainbow table (regardless of strength), it will be cracked instantly. If it is not in the table, it won't be cracked at all.

What matters for rainbow tables:

- **Password length** – longer passwords require exponentially larger tables
- **Character set size** – larger charset = larger tables needed
- **Whether the password was included in the pre-computation**

A "weak" password like `Password123!` and a "strong" random password like `xK9#mQ2$pL7@` are equally vulnerable if both are in the rainbow table, and equally safe if neither is in the table.

Defense against rainbow tables:

Salting – This is the standard defense. Each password gets a unique random salt, making pre-computed tables useless since you would need different tables for each salt value.

Modern Comparison

This is why modern systems use salted hashes (bcrypt, scrypt, Argon2) with key stretching – they make rainbow tables completely impractical regardless of password strength.

Problem 5.3 MS-CHAPv1: DES as a one-way function

A significant problem in the construction of the LMH response is the difference between the size of the DES keys (7 bytes) and the size of the DES output (8 bytes). Would it make sense to exchange the roles of the challenge and the LMH hash value?

Solution

Short Answer: No, this would NOT work and would be even worse.

Security Degradation

If we used: Response = $\text{DES}_{\text{Challenge}}(\text{LMH})$
Problems:

- The challenge is sent **in plaintext** over the network
- The attacker knows the DES key (challenge) completely
- The attacker knows the DES output (response)
- This becomes a **known-key attack**: given key K and ciphertext C, find plaintext P
- The attacker could directly compute: $\text{LMH} = \text{DES}^{-1}_{\text{Challenge}}(\text{Response})$
- **Result:** Instant recovery of the LM hash without any brute-force!

Problem 5.4 MS-CHAPv2: Cryptographic Primitives

In MS-CHAPv2, three cryptographic primitives are used: DES, MD4, and SHA-1. Without extending this set of primitives, and thus without requiring to write new code for new cryptographic primitives: Can you design a *secure* challenge-and-response protocol?

Solution

Just use a mutual authentication protocol (Figure 4.6) with HMAC-SHA1 as the HMAC function. Please note that although collission atatcks on SHA-1 are feasible, it was shown that the HMAC construdtion remains secure (Reference 5 in section 3). To derive a key for subsequent encryption, a symmetric key agreement protocol (Figure 4.7) ca be used, again with HMAC-SHA1 as a buliding block for the PRF.