Padding Oracle Attack on PKCS#1 v1.5: Can Non-standard Implementation Act as a Shelter?

Si Gao, Hua Chen, and Limin Fan

Trusted Computing and Information Assurance Laboratory,
Institute of Software, Chinese Academy of Sciences
{gaosi,chenhua,fanlimin}@tca.iscas.ac.cn

Abstract. In the past decade, Padding Oracle Attacks (POAs) have become a major threat to PKCS#1 v1.5. Although the updated scheme (OAEP) has solved this problem, PKCS#1 v1.5 is still widely deployed in various real-life applications. Among these applications, it is not hard to find that some implementations do not follow PKCS#1 v1.5 step-by-step. Some of these non-standard implementations provide different padding oracles, which causes standard POA to fail. In this paper, we show that although these implementations can avoid the threat of standard POA, they may still be vulnerable to POA in some way. Our study mainly focuses on two cases of non-standard implementations. The first one only performs the "0x00 separator" check in the decryption process; while the other one does not check for the second byte. Although standard POA cannot be directly applied, we can still build efficient padding oracle attacks on these implementations. Moreover, we give the mathematical analysis of the correctness and performance of our attacks. Experiments show that, one of our attacks only takes about 13 000 oracle calls to crack a valid ciphertext under a 1024-bit RSA key, which is even more efficient than attacks on standard PKCS#1 v1.5 implementation. We hope our work could serve as a warning for security engineers: secure implementation requires joint efforts from all participants, rather than simple implementation tricks.

1 Introduction

PKCS#1 is the standard for the implementation of public-key cryptography based on the RSA algorithm. The current version v2.2 [1], published by RSA in 2012, contains two encryption schemes: RSAES_PKCS1_v1.5 and RSAES_OAEP. For simplicity's sake, we denote them as PKCS#1 v1.5 and OAEP respectively. OAEP is required to be supported for new applications, while PKCS#1 v1.5 is included only for compatibility with existing applications.

Padding Oracle Attack. In the past decade, Padding Oracle Attacks (POAs) [2] have become a major threat to PKCS#1 v1.5. Padding Oracle Attack is a type of chosen ciphertext attack, which takes advantage of whether cryptographic operation is successfully executed. Usually, we assume the attacker can trick an honest user to decrypt the ciphertext he chose. In the decryption process, a format check is performed after decryption. Although the attacker does not have access to the decryption result, he can detect whether the ciphertext he chose passes the format check. We call such decryption process a "Padding Oracle" (PO) [3]. By collecting thousands of PO's responses, the original message can be extracted. In the past decade, POAs have drawn major attention from both symmetric and asymmetric cryptography research. In symmetric cryptography, CBC Padding Oracle has been used to build plaintext-recovery attacks on various network protocols [4–9]. In asymmetric cryptography, PKCS#1 v1.5 is the main target. The first POA on PKCS#1 v1.5, published by Bleichenbacher in 1998 [10], took about 1 million PO calls to recover a 1024-bit RSA plaintext. Bleichenbacher's attack has been extensively studied ever since, applied to SSL [11], PIN encryption in EMV [12], USB token [2] and XML encryption [13]. Recently, Bardou, Focardi, Kawamoto, Simionato, Steel and Tsay claim that using their improved version of Bleichenbacher's attack, a wrapped secret key can be recovered from RSA Securid 800 in only 13 minutes [2].

Other Attacks on PKCS#1 v1.5. Other non-POA attacks also exist for PKCS#1 v1.5: Coron, Joye, Naccache and Paillier proposed two brilliant attacks in 2000 [14], which can efficiently recover the plaintext, if the public exponent is small enough, or most message bits are zeros. Bauer, Coron, Naccache, Tibouchi and Vergnaud proposed a broadcast attack [15], which could reveal the identical plaintext when the public exponent is small. However, none of these attacks works for the commonly used public exponent 65537 with a random message. Bauer et al. also proposed a reliable distinguish attack [15]. Using one PO query, it predicts which of two chosen plaintexts corresponds to a challenge ciphertext. Although we only focus on full-plaintext-recovery attacks without requirements on exponent or plaintext, whether POAs can combine with these non-POA attacks may be an interesting topic for further study.

Does PKCS#1 v1.5 Still Matters in Today's Application? To avoid POA, RSA introduced OAEP as the new recommended encoding scheme in PKCS#1 v2.0. However, according to ECRYPT's "Yearly Report on Algorithms and Keysizes", PKCS#1 v1.5 is still widely deployed in today's application ((W)TLS, S/MME, XML, JSON, etc.) [16, 17]. Take USB tokens for instance: most tokens today support PKCS#1 v1.5, while only a few can support OAEP [2]. In software deployment, OAEP is widely supported today; while for backward compatibility reasons, PKCS#1 v1.5 is still mandatory. Jager and Paterson suggest that in such scenario [17], the attacker can trick the honest user to use legacy scheme (PKCS#1 v1.5) , and undermine the security of the up-to-date scheme (OAEP).

Motivation. Despite the fact that detailed implementation instructions are given in [1], implementations do not always follow them. For efficiency or other reasons, they tend to simplify the standard decryption process as long as valid ciphertexts can be decrypted correctly. For instance, in most of Microsoft's Cryptographic Service Providers (CSP), PKCS #1 v1.5 decryption does not check