Examining Misimplemented RSA and Strengthened Authentication for Variations of the Cryptovirological Information Extortion Attack

Justin Troutman[†]

April 1, 2005

1 The Counter-Attack

(Note: The following scenario, although a variation, retains the assumptions of the original information extortion attack in [1], such that the victim, V, does not extract IV and K_s, while briefly in RAM.)

1.1 Deduction Algorithm

To initiate the counter-attack, we need to propose a deduction algorithm, for deducing the necessary information; let's consider the following notation, from [2]. Let RSA public exponent, e, equal 5, and RSA public key, n, consists of 2048 bits. If $m < \sqrt[5]{n}$, then $m^e = m^5 < n$. As such, no modular reduction has occurred.

In our particular scenario, m is a 256-bit integer, given as $\{ChkSm, IV, K_s\}$; thus, public-key encrypted m' is given as $\{ChkSm, IV, K_s\}K_f$. We'll call this 256-bit integer our necessary information. The encrypted necessary information, using the RSA public key, given the parameters of the deduction algorithm, is less than $2^{256\times 5} = 2^{1280}$, which is less than the n of 2048 bits.

By using the above notation, from [2], m is deduced by computing $\sqrt[5]{m^5}$, which reveals $\{ChkSm, IV, K_s\}$. Therefore, as aforementioned, no modular reduction has occurred, and this necessary information is deduced by computing the fifth root of the encrypted necessary information. For the sake of clarity, note the slight, but, necessary adaptations between the notation in this paper and the notation in [2].

1.2 Cheating the Transaction Successfully Via the Lack of A Priori Information

The victim, V, generates a bogus H'. V computes $ChkSm'$ on H', using $\{H', IV, K_s\}$, with a block cipher in CFB mode. V forms altered m_a, which consists of $\{ChkSm',$

IV, $K_s\}$. V encrypts m_a with W's K_f to render m'_a, given as $\{ChkSm', IV, K_s\}K_f$. V uploads m'_a and H' and cheats the system, successfully, whilst having recovered the original critical data, D.

The adversary, W, has no *a priori* knowledge of the desired data, H, or its *ChkSm*, therefore, the authentication mechanism has failed; data confidentiality and integrity are not preserved.

2 Preserving Data Integrity

2.1 Formal Authentication

In the above counter-attack, it is shown that a confidentiality failure leads to an integrity failure, since the adversary, W, has no *a priori* knowledge of information related to the desired data, H, he is attempting to authenticate. Therefore, given IV and K_s, the integrity is compromised; given K_f, the transaction can be successfully completed, thus allowing the victim, V, to successfully cheat.

It is rather trivial to instantiate a scheme for including formal authentication, which also protects confidentiality, which uses a priori information as a means of properly verifying the integrity of the desired data, H. Therefore, by making this *a priori* information known to the adversary, W, but unknown to the victim, V, the victim, V, cannot successfully divulge or manipulate the transaction.

2.2 Introducing a MAC and Fixed Keys

To preserve data integrity, we introduce a function for producing a *MAC*, or *Message Authentication Code*; as such, we also introduce an additional assumption. The assumption is that two fixed symmetric keys reside in the cryptovirus, which are not extracted by the victim, V, before they are overwritten in RAM, after use; these fixed keys are known by the adversary, W.

Let's assume that the two fixed keys, K_a and K_e, were derived from larger, master key material, and are made distinct by a key separation technique, in the form of, for example, PRF(shared secret, "encryption key") and PRF(shared secret, "authentication key"). We'll also assume that any IVs used are known, to the adversary, W, and fixed, having been generated beforehand, by the adversary, W, using a cryptographically-secure pseudo-random number generator.

2.3 Strengthened Attack Via HMAC

For this particular example, let's assume the adversary, W, is using authentication, then encryption, in that order, or, as commonly denoted, "*AtE*." For this scheme, a hash function is used; let's assume SHA-256, in the HMAC construction. One of the aforementioned fixed keys that will be used for authentication is denoted as K_a.

Using K_a, and the notation and notions of security for HMAC, as discussed in [2, 3, 4], the following is computed, specifically for the attack: $h(K_a \oplus opad \mid\mid h(K_a \oplus ipad \mid\mid H))$, where *opad* and *ipad* are specified constants, h is the hash function, and H is, of course, the desired message data.

After HMAC has completed, K_a is deleted, and the MAC is encrypted using a block cipher in CFB mode, as originally specified in [1]; K_e is the fixed key used for encryption; this key, being symmetric, will also be used to decrypt. As such, we'll denote this use as K_d. Also, let's back-track, and assume that critical data, D, has been encrypted with a fixed IV and K_e.

The MAC is encrypted with K_e, such that $\{MAC\}K_e$; we'll denote $\{MAC\}K_e$ as m, thus revising the original information extortion attack. At this point, K_e is also deleted. To further revise the original information extortion attack, m' is formed by computing $\{(((MAC)\ K_e)\ K_f)\}$.

2.4 Completing the Transaction Successfully Via the Inclusion of A Priori Information

At this point, when the adversary, W, instructs the victim, V, to send information, the information being sent is m', which is $\{(((MAC)\ K_e)\ K_f)\}$, and the desired data, H. The adversary, W, decrypts m' with K_w, thus revealing m, which is then decrypted with K_d ($K_d = K_e$), thus deducing the MAC value; after such, he authenticates the desired data, H, by computing a MAC, using HMAC with K_a, then compares the result to the MAC deduced from m.

This formally authenticates H, since the adversary, W, has *a priori* information that is unknown to the victim, V; this *a priori* information is K_a and K_e. To complete the transaction, in a fair manner, the adversary, W, would send the victim, V, $\{IV, K_d\}$.

3 Conclusion

The counter-attack, although successful, relies completely on the insecure implementation of RSA, under specific conditions, where dangerous structure is not addressed using proper encoding functions, most often dubbed as "padding."

It is reasonable to assume that a clever adversary will take this into consideration. If so, the original attack, as specified in [1], works quite well. If not, the revised attack exploits the indirect failures in confidentiality and integrity, as a result of the insecure implementation of RSA.

However, RSA should be handled with meticulous care, thus suggesting that this issue should not exist, where the adversary is being responsible. The purpose of this research is to remind one of the fallacies associated with poor implementations of textbook RSA, as well as suggest a formal mechanism for providing authentication, and confidentiality, where RSA is misimplemented.

† Cryptographer, Charlotte-Metro, North Carolina

References

[1] A. Young, M. Yung, "*Cryptovirology: Extortion-Based Security Threats and Countermeasures*," IEEE Symposium on Security & Privacy, pages 129-141, May 6-8, 1996.

[2] N. Ferguson, B. Schneier, "*Practical Cryptography*," pages 94, 101-104, and 108, Wiley Publishing, Inc., 2003.

[3] M. Bellare, R. Canetti, H. Krawczyk, "*Keying Hash Functions for Message Authentication*," In (Neal Koblitz, editor), Advances in Cryptology - CRYPTO '96, volume 1109 of Lecture Notes in Computer Science, pages 1-15, Springer-Verlag, 1996.

[4] H. Krawczyk, M. Bellare, R. Canetti, "*HMAC: Keyed-Hashing for Message Authentication*," RFC 2104, February 1997.