A Chosen Text Attack on The Modified Cryptographic Checksum Algorithm of Cohen and Huang

Bart Preneel[1], *Antoon Bosselaers, René Govaerts and Joos Vandewalle*

ESAT Laboratories, Katholieke Universiteit Leuven

Abstract. *A critical analysis of the modified cryptographic checksum algorithm of Cohen and Huang points out some weaknesses in the scheme. We show how to exploit these weaknesses with a chosen text attack to derive the first bits of the key. This information suffices to manipulate blocks with a negligible chance of detection.*

1. INTRODUCTION

The protection of the integrity of data stored in computers and of messages in a communication network is becoming a problem of paramount importance. Data integrity can only be achieved through adding controlled redundancy under protection of a secret key. Two major approaches can be distinguished [9]. The first solution is a compression of the plaintext with a non-keyed hash function or Manipulation Detection Code (MDC) followed by an encryption of the plaintext and/or the result of the hash function. The other option is the compression of the plaintext under control of a secret key. In this case, the compression function is called a Message Authentication Code (MAC).

The first approach has the advantage that the authentication and encryption can be separated and results in a simplified key management (only one key). Moreover the design of a fast collision free hash function is in general less complicated than the

[1]NFWO aspirant navorser, sponsored by the National Science Foundation of Belgium.

design of a good MAC. However, the security of the scheme is completely dependent on the subsequent encryption. More details on the requirements for non-keyed hash functions can be found in [4, 10]. In case when no secrecy of the data is required, a MAC can offer a secure and economical solution. The requirements that are imposed on a MAC are the following:

1. The description of the MAC(.,.) must be *publicly known* and the only secret information lies in the key (extension of Kerckhoff's principle).
2. The data X can be of *arbitrary* length and the MAC has a fixed length n.
3. Given X and K, the computation of MAC(K,X) must be *"easy"*.
4. Given X, it is computationally infeasible to determine MAC(K,X) with a probability of success significantly higher than $1/2^n$. Even when a large set of pairs $\{X_i, MAC(K,X_i)\}$ are known, it is computationally infeasible for an opponent to determine the key K or to compute MAC(K,X') for any $X' \neq X_i$ with a probability of success significantly higher than $1/2^n$.
5. The MAC must be *collision free:* this means that it is computationally infeasible to find two distinct messages which hash to the same result without knowing the secret key K.

However, a good MAC is only a first step in obtaining integrity protection. A secure system should also provide a serial number, time information, a specification of the number of blocks and provide a procedure to deal with plaintexts whose length is not an integer multiple of the blocklength.

2. THE SCHEME OF COHEN AND HUANG

The design of a fast and secure MAC or MDC based on blockciphers can certainly not be considered as a trivial task. One only has to look at the long list of proposals coming from reputed cryptographers that were broken [1, 9, 11, 12]. The scheme proposed in [3] and modified in [8] is based on modular exponentiation (RSA, [13]). The factorization of the modulus is discarded so that the built-in trapdoor cannot be opened.

The plaintext consists of l blocks X_0 through X_{l-1} where X_0 is the filename or the first plaintext block. The logical EXOR operation is denoted with $\oplus$.

Algorithm 1 – A Revised checksum algorithm.

```
Key              Select an RSA key $(K_e, N)$ and a seed $K$.
Initialization   Set $Y_0 = \mathbf{RSA}(X_0 \oplus K)$
Main loop        For $i = 1$ to $l - 1$ :
                 $Y_i = \mathbf{RSA}\,(1 + [(X_i \oplus K) \bmod (Y_{i-1} - 1)])$
Result           The result of the checksum is $Y_{l-1}$
```

Here $\mathbf{RSA}(X)$ denotes the RSA encryption of the plaintext X with the key (K_e, N). As indicated in [3], there is no reason to keep the pair (K_e, N) secret, so the actual key of the cryptographic checksum algorithm is the seed K. The use of the RSA algorithm implies that certain parts of the algorithm are cryptographically strong. However, the computation of modular exponentiations is very slow, even with fast dedicated hardware available [2, 7] (17–20 kbit/s for a modulus and an exponent of 512 bits). The performance can be improved by means of a preliminary compression of the plaintext. This can be looked at as a tradeoff between two extremes: in the first case, there is no compression which results in a secure but slow compression algorithm ; on the other hand, the message could be compressed with an MDC and then the result could be encrypted with the RSA algorithm. In the last case, the RSA does not improve directly the cryptographical strength of the MDC. Our attack is worked out under the assumption that there is no preliminary compression such that the scheme takes maximal profit of the strength of the RSA. Because the compression $C(X_i)$ was not specified in [3, 8] we are obliged to omit it. A last remark concerns the length of the result. We believe that there is no reason to keep the full 512 bits of the result. Without decreasing the security level significantly, the result can be reduced to 128 bits by EXORing lower and higher parts.

3. WEAKNESS OF THE MODULO REDUCTION

The coupling of the blocks by a modulo reduction results in specific weaknesses like non-uniform distributions of the intermediate variables. When x and y are independent random integers between 0 and $N - 1$, the probability that $x \bmod y$ equals i is given by[2]:

$$P\left[(x \bmod y) = i\right] = \frac{1}{N^2} \sum_{k=i+1}^{N} \left\lfloor \frac{N-1+k-i}{k} \right\rfloor$$

[2]we define $x \bmod 0 = x \bmod N = x$.

where $\lfloor z \rfloor$ denotes the largest integer smaller than or equal to z. For $i = N-1$ this probability equals $\frac{1}{N^2}$ instead of $\frac{1}{N}$ for a uniform distribution. For $i=0$ the sum can be worked out to [6]

$$\frac{1}{N}\left[\ln(N-1) + 2\gamma + O(\frac{1}{\sqrt{N}})\right],$$

where $\gamma = 0.5772156649\ldots$ is Euler's constant.

Because K is a uniform random variable, the same holds for $X_i \oplus K$. The good randomness properties of the RSA cause the result Y_i of the RSA operation also to be random. As a consequence,

$$P\left[X_i \oplus K < Y_{i-1} - 1\right] = \frac{1}{2} - \frac{2}{N}.$$

In that case, Y_i is independent of all previous blocks.

Under the assumption that the plaintext blocks are independent, uniformly distributed random variables, one can easily prove the following theorem.

Theorem 1 *If the first t bits of Y_{l-k} $(2 \leq k \leq l)$ are equal to* 1*, the probability that Y_{l-1} is independent of Y_{l-k} — and thus of the data blocks X_0 to X_{l-k} — is approximately equal to* $1 - 1/2^{k+t-1}$.

This opens the door to tamper with messages by changing individual blocks and combining blocks of different plaintexts in one new plaintext with a low probability of detection. There is especially a very small dependence of the checksum result on the first plaintext blocks. This clearly violates the imposed requirements. One can wonder how an attacker can obtain the intermediate result, but this is very easy when he can compute the checksum for a shorter version of the plaintext. The error probability of an attack could be significantly lowered when he would know K or at least the first bits of K. In the following paragraph we will show how the first s bits of K can be derived by means of an adaptive chosen plaintext attack.

4. DERIVING THE FIRST s BITS OF THE KEY K

We assume that an attacker can compute the checksum for messages consisting of one block X_0 and of two blocks $[X_0, X_1]$. Later it will be shown that the attack also works for longer messages. The corresponding checksums are given by the following equations:

$$\begin{aligned} Y_0 &= \mathbf{RSA}(X_0 \oplus K) \\ Y_1 &= \mathbf{RSA}\,(1 + [(X_1 \oplus K) \bmod (Y_0 - 1)]) \end{aligned}$$

Because the modular exponentiation is bijective, we can extract information on the most significant bits of K by comparing Y_0 and Y_1. For a given X_0, we are looking for a X_1 such that Y_0 equals Y_1.

$$Y_0 = Y_1 \iff X_0 \oplus K = 1 + [(X_1 \oplus K) \bmod (Y_0 - 1)]$$

If $(X_1 \oplus K) < Y_0 - 1$ the modulo operation has no influence and thus

$$X_0 \oplus K = 1 + (X_1 \oplus K).$$

The fact that K is unknown cannot prevent us from solving it for X_1 if $X_1 \oplus K < Y_0 - 1$.

We will denote the components of a vector A with

$$[A(n), A(n-1), \ldots, A(1)].$$

The vector E_i is defined as follows:

$$E_i(j) = \begin{cases} 0 & \text{for } n \geq j \geq i+1 \\ 1 & \text{for } i \geq j \geq 1 \end{cases} \qquad (1)$$

Algorithm 2 - Solving for X_1.

```
$i = 0$
compute $Y_0$
**repeat**   $i = i + 1$
             $X_1 = X_0 \oplus E_i$
             compute $Y_1$
             **until** $(Y_0 = Y_1)$ or $(i \geq j)$
```

The expected number of trials for this algorithm is 2. It would be possible to try all n possibilities, but in order to speed up the algorithm, we limit the number of trials to j. The error probability equals then $1/2^j$.

We are now able to describe our attack. We will show first how to determine the most significant bit of K and then it will be indicated how the attack can be extended to the higher order bits.

4.1 Deriving the most significant bit of K

results in a special Y_0 and in step 2 we look for a corresponding value of X_1 that results in equality between Y_0 and Y_1.

Step 1

Choose X_0 and compute the corresponding value of $Y_0 = \mathbf{RSA}(X_0 \oplus K)$ until

$$\begin{cases} Y_0(n) &= 1 \\ Y_0(i) &= 0, \quad i = n-1, n-2, \ldots, n-k+1 \\ Y_0(1) &= 0. \end{cases}$$

This will require on the average 2^{k+1} RSA encryptions. In case $Y_0 = 0$ we are very lucky, because this implies $K = X_0$. In the following, we will thus assume that $K \neq X_0$.

Step 2

Use Algorithm 2 to find a X_1 such that $Y_0 = Y_1$. We have to consider two cases:

1. $X_0(n) \oplus K(n) = 0$ (probability $= \frac{1}{2}$).

   The construction of X_1 implies that $(X_1 \oplus K) < Y_0 - 1$ and thus Algorithm 2 will give a solution after on the average 2 RSA encryptions.

2. $X_0(n) \oplus K(n) = 1$ (probability $= \frac{1}{2}$).

   (a) $X_1 \oplus K = Y_0 - 1$ (probability $= \frac{i-1}{2^n}$): in this case we are very lucky again because Y_1 will be 1 and K can be easily computed.

   (b) $X_1 \oplus K < Y_0 - 1$ (probability $\simeq \frac{1}{2^k}$): Algorithm 2 will find a solution as in case 1.

   (c) $X_1 \oplus K > Y_0 - 1$ (probability $\simeq 1 - \frac{1}{2^k}$): because $Y_0(n) = 1$, the modulo operation can be replaced with a subtraction:

   $$1 + [(X_1 \oplus K) \bmod (Y_0 - 1)] = 1 + (X_1 \oplus K) - Y_0 + 1.$$

   Equality of Y_0 and Y_1 can be obtained if

   $$X_0 \oplus K = (X_1 \oplus K) + 2 - Y_0.$$

For the least significant bit, this results in following equation:

$$X_0(1) \oplus K(1) = X_0(1) \oplus E_i(1) \oplus K(1) \oplus Y_0(1).$$

The fact that $E_i(1) = 1$ results in $Y_0(1) = 1$ which contradicts our previous assumption that $Y_0(1) = 0$. However, even when this would not be the case, it is very unlikely that Algorithm 2 would yield a solution.

It is easily seen that the above procedure allows to determine the most significant bit of K: if Algorithm 2 succeeds, we decide that $K(n) = X_0(n)$, else we put $K(n) = \overline{X_0(n)}$. There are two cases in which the algorithm fails. The fact that only j steps are applied in Algorithm 2 implies that it is wrongly decided with a probability of $1/2^j$ that there is no solution, but every additional RSA computation divides this error probability by 2. A more serious problem is that if $K(n) = \overline{X_0(n)}$, Algorithm 2 will succeed with a probability $1/2^k$. A halving of this error probability requires on the average a doubling of the amount of precomputation in step 1. This leads us to the conclusion that these errors will occur more frequently. The results are summarized in the following table.

probability	# RSA calculations	result
$1/2$	$2^{k+1} + 2$	$K(n)$
$1/2^{k+1}$	$2^{k+1} + 2$	$\overline{K(n)}$
$1/2\left(1 - (1/2^k)\right)$	$2^{k+1} + j$	$K(n)$

4.2 Deriving the s most significant bits of K

The same attack can be extended to derive the first s bits of K. It is not feasible to compute *all* bits of K because the amount of computation doubles for each bit. However, an attacker does not need to know all bits of K to improve his odds significantly. When he knows the first s bits of K, he can force the first s bits of $X_i \oplus K$ to zero, which implies that $X_i \oplus K$ is smaller than Y_{i-1} with a probability of $1 - (1/2^s)$, when Y_{i-1} is uniformly distributed. On the other hand, in case Y_{i-1} is known, it is possible to find a X_i such that $X_i \oplus K < Y_{i-1}$ for $2^n - 2^{n-s}$ values of Y_{i-1}.

The following variation on our attack will compute the sth bit of K under the assumption that the first $s - 1$ bits of K are already known.

Step 1

Choose X_0 and compute the corresponding value of $Y_0 = \mathbf{RSA}(X_0 \oplus K)$ until

$$\begin{cases} Y_0(i) \oplus K(i) \oplus X_0(i) & = & 0 \quad i = n, n-1, \ldots, n-s+2 \\ Y_0(n-s+1) & = & 1 \\ Y_0(i) & = & 0, \quad i = n-s, n-s-1, \ldots, n-s-k+2 \\ Y_0(1) & = & 0. \end{cases}$$

This will require on the average 2^{k+s} RSA encryptions.

Step 2

As for the first bit, use Algorithm 2 to find a X_1 such that $Y_0 = Y_1$.

To derive the first s bits of K, the total number of modular exponentiations can be shown to be approximately

$$s \cdot \left(1 + \frac{j}{2}\right) + 2^{k+1} \cdot (2^s - 1).$$

When $j \gg k$, the probability that these s bits are correct equals

$$\left(1 - \frac{1}{2^{k+1}}\right)^s.$$

4.3 Further extensions

We assumed for reasons of simplicity that the length of the chosen plaintext was only two blocks. The attack can however be extended to longer plaintexts. It suffices to look for a plaintext that results in a very large checksum Y_{l-1}. We can then add two blocks X_l, X_{l+1} to the text and with a probability $\frac{Y_{l-1}}{N}$ we can write

$$\begin{aligned} Y_l &= \mathbf{RSA}\left(1 + [(X_l \oplus K) \bmod (Y_{l-1} - 1)]\right) \\ &= \mathbf{RSA}\left(1 + (X_l \oplus K)\right) \\ Y_{l+1} &= \mathbf{RSA}\left(1 + [(X_{l+1} \oplus K) \bmod (Y_l - 1)]\right) \end{aligned}$$

In this case we can repeat the previous attack. The only difference is that the addition of 1 appears also in the first equation and thus Algorithm 2 is no longer necessary. On the other hand, we need a plaintext with a large checksum.

5. SUMMARY

We have shown that the modified version of the cryptographic checksum algorithm proposed by Cohen and Huang is insecure. The result of the checksum is insensitive to changes in the initial part of the plaintext and thus several manipulations are possible. Moreover, an attacker can compute the first bits of the key using an adaptive chosen text attack. Knowledge of these bits reduces significantly the chances on detecting a fraud.

References

[1] S.G. Akl, "On the Security of Compressed Encodings", *Advances in Cryptology, Proc. Crypto 83*, Plenum Press, New York, p. 209–230.

[2] E.F. Brickell, "A Survey of Hardware Implementations of RSA", *Advances in Cryptology, Proc. Crypto '89.*

[3] F. Cohen, "A Cryptographic Checksum for Integrity Protection", *Computers & Security*, Vol. 6, p. 505-510, 1987.

[4] I.B. Damgård, "Design principles for hash functions", *Advances in Cryptology, Proc. Crypto '89.*

[5] M. Girault, "Hash-functions Using Modulo-n Operations", *Advances in Cryptology, Proc. Crypto 86*, Springer Verlag, p. 217–226.

[6] G.H. Hardy and E.M. Wright, *"An introduction to the theory of numbers. 5th edition."*, Oxford University Press, 1979.

[7] F. Hoornaert, M. Decroos, J. Vandewalle and R. Govaerts, "Fast RSA-Hardware: Dream or Reality ?", *Advances in Cryptology, Proc. Eurocrypt 88*, Springer Verlag, p. 257–264.

[8] Y.J. Huang and F. Cohen, "Some Weak Points of One Fast Cryptographic Checksum Algorithm and its Improvement", *Computers & Security*, Vol. 7, p. 503-505, 1988.

[9] R.R. Jueneman, "A High Speed Manipulation Detection Code", *Advances in Cryptology, Proc. Crypto 86*, Springer Verlag, p. 327–347.

[10] R.C. Merkle, "One way hash functions and DES", *Advances in Cryptology, Proc. Crypto '89.*

[11] S.F. Mjølsnes, "A Hash Of Some One–Way Hash functions and Birthdays", preprint.

[12] B. Preneel, R. Govaerts and J. Vandewalle, "Cryptographically Secure Hash Functions: an Overview", *Internal Report, ESAT Laboratories K.U.Leuven*, 1989.

[13] R.L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Comm. ACM*, Vol. 21, No. 2, p. 120–126, 1978.

[14] G.J. Simmons,"A Survey of Information Authentication", *Proc. IEEE*, Vol. 76, No. 5, p. 603–620, 1988.