Generating Minimum Entropy Affine Multiplicative Relations in $\mathbb{Z}_N$

(Submission to IMACC 2011)

Abstract. Affine RSA signatures consist in signing $\omega \cdot m + a$ instead of the message m for some fixed constants ω, a. A thread of publications progressively reduced the size of m for which affine signatures can be forged in polynomial time. The current bound is $\log m \sim \frac{1}{3} \log N$ where N is the public RSA modulus.

In this paper we consider a slightly different problem: instead of minimizing m's size we try to minimize its *entropy*. We show that affine signatures on $\frac{1}{4} \log N$ entropy-bit messages can be forged in polynomial time. This problem has no specific cryptographic impact but allows to better understand how malleable the RSA function is.

We also exhibit a second sub-exponential time technique (faster than factoring) for creating affine modular relations between integers containing three messages of size $\frac{1}{4} \log N$ and a fourth message of size $\frac{3}{8} \log N$.

Keywords: cryptography; RSA signatures; message padding; affine redundancy.

1 Introduction

To prevent forgers from exploiting RSA's multiplicative homomorphism [6], it is a common practice not to sign a message m directly but to first apply to it a *padding function* $\mu(m)$.

This paper considers one of the simplest padding functions called affine padding (or fixed-pattern padding):

$$\sigma = \mu(m)^d \mod N = (\omega \cdot m + a)^d \mod N$$

Here d denotes the RSA private exponent and N the public modulus. We denote by $|X|$ the bit-size of X. Following [1], we use the following notations:

$$\mu(m) = \omega \cdot m + a \quad \text{where} \begin{cases} \omega \text{ is the multiplicative redundancy} \\ a \text{ is the additive redundancy} \end{cases} \tag{1}$$

Since no proof of security is known for RSA signatures using such a μ, those signatures should not be used in practice. Nonetheless, the study of such simple padding formats is useful for understanding how malleable the RSA function is.

In 1985, [2] exhibited forgeries for $\omega = 1$ when $m \sim \sqrt[3]{N^2}$. This attack was extended by [3] in 1997 to any values of ω, a and for $m \sim \sqrt{N}$. Finally, [1] exhibited in 2001 forgeries when $m \sim \sqrt[3]{N}$. This remained the best polynomial-time result for a decade. Relaxing the polynomial time constraint [4] showed that smaller message sizes can be tackled in complexity lower than that of all currently known factorization algorithms.

Our contribution. This paper does not further improve the bound but considers a slightly different problem: Rather than minimizing $\log m$ we try to minimize m's *entropy*. Using a variant of [1], we show how to forge signatures with a message entropy of $|N|/4$, instead of $|N|/3$. This problem has no specific cryptographic impact but allows to better understand how malleable the RSA function is.

2 Brier-Clavier-Coron-Naccache's Algorithm

In this section we briefly recall Brier *et al.*'s attack [1] using a slightly different exposition.

The goal is to find four distinct messages $x, y, z, t \in \mathbb{Z}$ each of size one third of the modulus size, such that:

$$(\omega \cdot x + a) \cdot (\omega \cdot y + a) = (\omega \cdot z + a) \cdot (\omega \cdot t + a) \mod N \tag{2}$$

which enables to forge the signature of x using:

$$(\omega \cdot x + a)^d = \frac{(\omega \cdot z + a)^d \cdot (\omega \cdot t + a)^d}{(\omega \cdot y + a)^d} \mod N$$

Denoting $P = a/\omega \mod N$, from (2) it is sufficient to solve the following equation:

$$(P + x)(P + y) = (P + z)(P + t) \mod N$$

Theorem 1 (Brier *et al.*). *Given N, P in $\mathbb{Z}$ with $P \neq 1$, the equation:*

$$(P + x)(P + y) = (P + z)(P + t) \mod N$$

has a solution $x, y, z, t \in \mathbb{Z}$ computable in polynomial time, with $0 \leq x, y, z, t \leq 8 \cdot \sqrt[3]{N}$ and with $y \neq z$ and $y \neq t$.

Proof. The previous equation gives:

$$P \cdot (x + y - z - t) = z \cdot t - x \cdot y \mod N$$

By developing P/N as a continued fraction, we find $U, V \in \mathbb{Z}$ such that $P \cdot U = V \mod N$ where $-\sqrt[3]{N} \leq U < \sqrt[3]{N}$ and $0 < V < 2 \cdot N^{2/3}$ and $\gcd(U, V) = 1$. Therefore it suffices to solve the following system:

$$\begin{cases} x \cdot y - z \cdot t &= V \\ x + y - z - t &= -U \end{cases} \tag{3}$$

A solution can be found using the following lemma.

Lemma 1. *Let $A, B, C \in \mathbb{Z}$ with $\gcd(B, C) = 1$. The system of equations:*

$$\begin{cases} x \cdot y - z \cdot t = A \\ \quad x - z \quad\;\; = B \\ \quad y - t \quad\;\; = C \end{cases}$$

has a solution given by

$$\begin{cases} t = y - C & = \frac{A - C \cdot (A \cdot C^{-1} \mod B)}{B} - C \\ x = B + z & = B + (A \cdot C^{-1} \mod B) \\ y = \frac{A - C \cdot z}{B} & = \frac{A - C \cdot (A \cdot C^{-1} \mod B)}{B} \\ z = A \cdot C^{-1} \mod B & \end{cases}$$

Proof. Letting $x = B + z$ and $t = y - C$, the first equation can be replaced by:

$$(B + z) \cdot y - z \cdot (y - C) = A$$

which gives:

$$B \cdot y + C \cdot z = A \tag{4}$$

Since $\gcd(B, C) = 1$ we get:

$$z = A \cdot C^{-1} \mod B$$

Moreover from equation (4) we obtain:

$$y = \frac{A - C \cdot z}{B} \tag{5}$$

which is an integer since $A - C \cdot z = 0 \mod B$. This concludes the proof. □

We come back to the proof of Theorem 1. Let $A = V$ and choose B such that $\sqrt[3]{N} < B < 2 \cdot \sqrt[3]{N}$ and $\gcd(B, U) = 1$. Let $C = -U - B$ which gives $B + C = -U$; therefore from system (3) it suffices to solve the system:

$$\begin{cases} x \cdot y - z \cdot t = A \\ x - z \quad = B \\ y - t \quad = C \end{cases}$$

Since $\gcd(B, C) = \gcd(B, -U - B) = \gcd(B, U) = 1$, the previous system can be solved using Lemma 1. Moreover we have $0 \leq z < B \leq 2 \cdot \sqrt[3]{N}$ and $0 \leq x \leq 3 \cdot \sqrt[3]{N}$. From $C = -U - B$ we have:

$$1 \leq -C \leq 3 \cdot \sqrt[3]{N}$$

which gives $0 \leq y \leq 5 \cdot \sqrt[3]{N}$, and eventually $0 \leq t \leq 8 \cdot \sqrt[3]{N}$.

Finally since $C \neq 0$ we have $y \neq t$. Moreover if $y = z$ from equation (5) we get $A = (B + C) \cdot z = -U \cdot z = -V$ which gives $V = U \cdot z$; if $z \neq 1$ this gives $\gcd(U, V) \neq 1$, a contradiction; if $z = 1$ this gives $U = V$ and therefore $P = 1$, a contradiction; therefore $y \neq z$. This concludes the proof of the Theorem. $\square$

An example of such a forgery is given in Appendix A.

The idea lends itself to many variants. For instance $5|N|/4$ entropy bit relations can be found by solving the following equation (modulo N):

$$(P+tK+x)(P+tK+y) = (P+tK+z)(P+tK+w) \Rightarrow P+tK = \frac{xy - wz}{w - x - y + z}$$

Letting $P = A/B \bmod N$ with $|A| = 3|N|/4$ and $|B| = |N|/4$ we get $P + tK = (A + tK \cdot B)/B$ for any t. Picking $t = -\lfloor A/(KB)$, this gives $P + tK = C/B$ for some $|C| = |N|/2$. We can hence solve this equation using Lemma 1 by identifying:

$$\begin{cases} C = xy - wz \\ B = w - x - y + z \end{cases}$$

3 Minimal Entropy Forgeries

We now consider a slightly different equation. Let $k = |N|$, define $K = 2^{\lfloor k/4 \rfloor}$ and consider the equation:

$$(P + x) \cdot (P + K \cdot y) = (P + z) \cdot (P + K \cdot t) \mod N \tag{6}$$

The following explains how to find in polynomial time four distinct solutions x, y, z, t of size $\sim \frac{1}{4} \log N$. Therefore this gives a relation between four messages $m_1 = x$, $m_2 = K \cdot y$, $m_3 = z$ and $m_4 = K \cdot t$ of size $\sim \frac{1}{2} \log N$ but an entropy of $\sim \frac{1}{4} \log N$ bits only.

By expanding equation (6) we get:

$$P\left(x - z + K \cdot (y - t)\right) = K \cdot (z \cdot t - x \cdot y) \mod N$$

As previously, by developing P'/N as a continued fraction with $P' = P/K \mod N$, we can find U, V such that $P \cdot U = K \cdot V \mod N$ where this time we take $-\sqrt{N} \le U \le \sqrt{N}$ and $0 \le V \le 2\sqrt{N}$. Then it suffices to solve the system:

$$\begin{cases} x - z + K \cdot (y - t) = -U \\ \quad x \cdot y - z \cdot t \quad\quad = V \end{cases}$$

Using Euclidean division we write $U = H \cdot K + L$ with $0 \le L < K$; then it suffices to solve the system:

$$\begin{cases} x \cdot y - z \cdot t = V \\ \quad x - z \quad = -L \\ \quad y - t \quad = -H \end{cases}$$

which can be solved thanks to Lemma 1. Since $V \sim \sqrt{N}$ and the size of both L and H is roughly $\frac{1}{4}|N|$, one obtains four solutions x, y, z and t of size $\frac{1}{4}|N|$ in polynomial time.

However, for the lemma to apply, we must assume that $\gcd(L, H) = 1$; this makes the algorithm heuristic. If we assume that L and H are uniformly distributed, we have:

$$\Pr\left[\gcd(L, H) = 1\right] \approx \frac{6}{\pi^2}$$

We illustrate the process in the appendix using RSA Laboratories' official 1024-bit challenge modulus RSA-309.

Note that one can improve the algorithm's success probability by considering

$$(L', H') = (L + r \cdot K, H - r)$$

instead of (L, H), for small values of $r \in \mathbb{Z}$. Assuming independent probabilities, after ℓ trials the failure probability drops to $(1 - 6/\pi^2)^\ell$, which is negligible even for small values of ℓ (and experiments suggest that in practice failure probability decreases even faster than this rough estimate).

3.1 Message Entropy

We now define more precisely message entropy in the context of affine RSA forgery.

Let P be a fixed pattern and N a random variable denoting the RSA modulus. Let $\mathcal{F}$ be a forging algorithm making ℓ signatures queries for messages $m_1, \ldots, m_\ell$ and producing a forgery $m_{\ell+1}$. We regard the messages $m_1, \ldots, m_\ell$ and $m_{\ell+1}$ as random variables induced by N (and possibly by the random tape of $\mathcal{F}$). We consider the entropy of individual messages separately and take the maximum entropy over all messages:

$$H_P := \max\{H(m_i) \mid (m_1, \ldots, m_\ell, m_{\ell+1}) \leftarrow \mathcal{F}(N, e, P), (N, e) \leftarrow \mathsf{GenKey}(1^k)\}$$

We define the forgery's entropy as the maximum over all possible values of the pattern P:

$$H := \max\{H_P \mid P \in \mathbb{Z}\}$$

We see that in the described algorithm, the message entropy is roughly $|N|/4$, whereas it was $|N|/3$ in Brier *et al.*'s attack.

Note that in the previous definition we consider the maximum entropy of *all* messages required for the forgery and not only the entropy of the message whose signature is forged. For example [3] is selective, which means that the attacker can for a signature for a message of his choosing; therefore the forged message can have zero entropy; however the remaining messages in [3] are half the size of N and their entropy is roughly $|N|/2$.

4 Sub-Exponential Strategies

We start by noting that $\frac{|N|}{4}$ forgeries of the form

$$(P+x)(P+y) = (P+z)(P+w) \bmod N \quad \Rightarrow \quad P = \frac{z+w-x-y}{xy-wz} \bmod N$$

This means that P can be written as a modular ratio of two integers (namely $z+w-x-y$ and $xy-wz$) that are respectively $|N|/4$ and $|N|/2$ bits long. As this is expected to occur with probability $\sim 2^{-|N|/4}$ we infer that for arbitrary P values, such forgeries shouldn't exist in general.

Consider a forgery of the form:

$$(P+x)(P+y) = P(P+x+y+z) \bmod N$$

Hence, if x, y, z exist, they are such that:

$$P = \frac{xy}{z} \bmod N$$

Write:

$$P = \frac{A}{B} \bmod N \quad \text{where } |A| = \frac{5|N|}{8} \text{ and } |B| = \frac{3|N|}{8}$$

Then for any u, $P + u = (A + uB)/B \bmod N$. Thus, if we fix $u' = -\lfloor A/B \rfloor$, we get:

$$P + u' = \frac{A + u'B}{B} = \frac{A'}{B} \bmod N$$

where $|A'| = |B| = 3|N|/8$. We can attempt to factor A' as a product $x \times y$ of two factors smaller than $|N|/4$ bits each. If this factorization does not succeed add one to u' and start over.

The result is a $\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{3}{8}$ forgery such as the one given in Appendix B.

Again, the idea lends itself to a number of variants. For instance, relations of the form

$$(P + tx)(P + ty) = (P + tz)(P + tw) \Rightarrow P = \frac{t(xy - wz)}{w - x - y + z}$$

can be found by writing $P = \frac{A}{B} \bmod N$ with $|A| = |B| = |N|/2$, factoring A to find t and continuing with Lemma 1.

5 Further Research

While computing $\frac{1}{4}$ forgeries remains an open problem, neighboring problems may lead to surprising algorithms. We give here two such variants as departure points to future research.

5.1 The Case of Two Interchangeable Padding Patterns

Let P and P' be two independently generated padding patterns and assume that the signer can sign messages using either P or P'. We have:

$$(P+x)(P'+y) = (P+z)(P'+t) \bmod N \ \Rightarrow \ P(y-t)+P'(x-z)+xy-zt = 0 \bmod N$$

Find A, B, C of respective sizes $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$ such that $PC + P'B + A = 0 \bmod N$ and solve the system

$$\begin{cases} x \cdot y - z \cdot t = A \\ \quad x - z \quad\;\; = B \\ \quad y - t \quad\;\; = C \end{cases}$$

as before. Note that this yields a $\frac{1}{4}$ forgery only if P and P' are "independent". If $P = P' + \alpha$ for a small α then a $\frac{1}{3}$ forgery is found.

5.2 Allowing The Attacker to Influence N

Assume that the attacker can select the most significant half of N (*e.g.* [5] and [7] who report that such a practice does not seem to weaken N). Let P be an arbitrary padding pattern and:

$$(P+x)(P+y) = (P+x')(P+y') \bmod N \;\;\Rightarrow\;\; P(x+y-x'-y')+xy-x'y' = 0 \bmod N$$

where x, y, x', y' are all of size $\frac{1}{4}$. This is solved by writing:

$$\begin{cases} P \cdot a + b & = 0 \bmod N \\ x + y - x' - y' & = a \\ xy - x'y' & = b \end{cases}$$

Hence, for a given P we need to find an N for which a and b are of respective sizes $\frac{1}{4}$ and $\frac{1}{2}$. We then find x, y, x', y' exactly as previously but of size $\frac{1}{4}$ (to do so define $x - x' = \alpha$ for an arbitrary α and solve the two equations). Write $Pa + b = kN$ as we can select the most significant bits of N, let $N = N_1 + N_0$ where N_1 (of size 1) is chosen by the attacker and where N_0 (of size $\frac{1}{2}$) is not under the attacker's control.

This boils down to $Pa + b = k(N_1 + N_0)$. Selecting $N_1 = 2P$ the attacker gets $Pa + b = k(2P + N_0)$. Hence $k = 1$, $a = 2$ and $b = N_0$ is a satisfactory choice for which a and b are of respective sizes $\frac{1}{4}$ and $\frac{1}{2}$ (as a matter of fact a is much smaller but this is not an issue).

SAGE code for the attack is given in Appendix C.

References

1. E. Brier, C. Clavier, J.S. Coron and D. Naccache, *Cryptanalysis of RSA signatures with fixed pattern padding*. Proceedings of CRYPTO'01.

2. W. De Jonge and D. Chaum, *Attacks on some RSA signatures*. Proceedings of Crypto'85, LNCS vol. 218, Springer-Verlag, 1986, pp. 18-27.
3. M. Girault and J.-F. Misarksy, *Selective forgery of RSA signatures using redundancy*, Proceedings of Eurocrypt'97, LNCS vol. 1233, Springer-Verlag, 1997, pp. 495–507.
4. A. Joux, D. Naccache, E. Thomé, *When e-th Roots Become Easier Than Factoring*. ASIACRYPT 2007: 13-28
5. A.K. Lenstra, *Generating RSA moduli with a predetermined portion*, Advances in Cryptology ASIACRYPT '98, LNCS vol. 1514, Springer, pages 1-10, 1998.
6. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, CACM 21, 1978.
7. A. Shamir, *RSA for paranoids*, CryptoBytes (The Technical Newsletter of RSA Laboratories) 1 (3) (1995) l, 3, and 4.
8. `http://sites.google.com/site/bbuhrow/home`

A Minimum Entropy Forgery

$\mu(m_1) \cdot \mu(m_2) = \mu(m_3) \cdot \mu(m_4) \mod N_{309}$ with $\omega = 1$ and $a = P = 2^{1023} - 2^{516}$. N_{309} is RSA Laboratories' unfactored challenge modulus RSA-309.

The entropy of messages m_1, m_2, m_3 and m_4 is $\cong \frac{|N_{309}|}{4}$.

N_{309} = RSA-309

```
= bdd14965 645e9e42 e7f658c6 fc3e4c73 c69dc246 451c714e b182305b 0fd6ed47
  d84bc9a6 10172fb5 6dae2f89 fa40e7c9 521ec3f9 7ea12ff7 c3248181 ceba33b5
  5212378b 579ae662 7bcc0821 30955234 e5b26a3e 425bc125 4326173d 5f4e25a6
  d2e172fe 62d81ced 2c9f362b 982f3065 0881ce46 b7d52f14 885eecf9 03076ca5
```

$\mu(m_1)$ =

```
7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffff0
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001
0a22096c f25655f4 104b2971 bc8b4f4f 817e6f4c d0ca8b25 ac5e8377 819e9d23
```

$\mu(m_2)$ =

```
7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffff0
e1bdb579 4ad9e45a 7b17ee62 bf736d1c 8d897862 ce2c3349 72600b8b 44a8d4fb
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

$\mu(m_3)$ =

```
7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffff0
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
113271f6 527c0815 fbf55d24 4883207b 827b9fb9 dd3ba8a6 2af1b776 d550a12d
```

$\mu(m_4)$ =

```
7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffff1
0c9ca82a 80d6e82a e84d12a7 6415e27e d6d909da c2331285 aca27f4e 632d1556
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

B Fast Sub-Exponential Forgery

$\mu(m'_1)\cdot\mu(m'_2) = \mu(m'_3)\cdot\mu(m'_4) \mod N_{309}$. Factorization for obtaining this relation was done with YAFU [8] using fast MPQS and SIQS implementations for Core2 processors.

```
P′ = 7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
     ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000008 ← note the 8
     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

μ(m′1) = 7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
         ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
         00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000008
         78dfd16f afa9c95b 2fecb797 21eae4a7 5217f260 0a9b852a 01dee0cf 315aea20

μ(m′2) = 7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
         ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
         00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000008
         78dfd16f afaa4c53 011c40cf ce5ff1d9 f9d2f822 3bf3b3ad c770bdd4 4644e869

μ(m′3) = 7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
         ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
         00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000008
         78dfd16f afa9c95b 2fefcd95 a1f55dd7 1f55b73a b29e0570 f72a86d2 940f34d1

μ(m′4) = 7fffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
         ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
         00000000 00000000 00000000 00000000 98fb10e4 ef8f2456 b2ab14a2 236dadea
         6b28b31b 1bfb2493 8f74d85e eceb7450 2d848353 fe9f1dec b40f041b bbb2a1f8
```

C Allowing The Attacker to Influence N (SAGE code)

```
def hex(x):
  s=x.digits(base=16,digits='0123456789abcdef')
  s.reverse()
  return "".join(s)

def invmod(a,b):
  g,c,d=xgcd(a,b)
  return c

def testattack(n=512):
  P=ZZ.random_element(2^n)
  N1=2*P
  q=random_prime(2^(n//2))
  p=N1//q
```

```
NN=p*q
print "N=",NN
print "N/2=",NN//2
print "P=",P

a=2
b=NN-N1

al=ZZ.random_element(2^(n//4))
while gcd(al,a)!=1:
  al=ZZ.random_element(2^(n//4))

xp=ZZ(mod(b*invmod(a-al,al),al))
y=(b-xp*(a-al))/al
x=xp+al
yp=y-(a-al)

print "x=",x
print "y=",y
print "x\'=",xp
print "y\'=",yp

print "(P+x)(P+y)=(P+x\')(P+y\') mod N ",mod((P+x)*(P+y)-(P+xp)*(P+yp),NN)==0
```