Journal of Information Hiding and Multimedia Signal Processing ©2012 ISSN 2073-4212
Ubiquitous International Volume **3**, Number **2**, April **2012**

Number Theoretic SETUPs for RSA Like Factoring Based Algorithms

Constantinos Patsakis

Department of Computer Engineering and Mathematics
Universitat Rovira i Virgili
UNESCO Chair in Data Privacy
Av. Països Catalans 26 43007 Tarragona, Catalonia, Spain
kpatsak@gmail.com

Received September 2010; revised April 2012

Abstract. *For many years there was a very common misbelief that having embedded mechanisms in hardware, constitute the optimal solution not only in the usage and the performance for the encryption algorithms, but for their security as well. The idea that lead to this common belief was that by this way an attacker could not tamper the encryption algorithms, specially regarding the asymmetric ones. M. Young and Yung introduced kleptography, showing that there are more risks than the ones that are expected to be faced, when using cryptographic systems.*

This work summarizes the scope of kleptography and analyzes two new number theoretic SETUPs for factoring encryption algorithms, one based on Lagrange's four square theorem and one based on Coppersmith's theorem. Both SETUPs target public key encryption algorithms, whose key generation depends on the product of two primes, providing a robust and secure solution.

Keywords: RSA, ElGamal, Secretly Embedded Trapdoor with Universal Protection, backdoor, public key cryptography, Coppersmith's theorem, factoring, cryptovirology, kleptography, four square theorem.

1. **Introduction.** Cryptography has been developing for centuries. One of its main objectives is to restrict access on some texts, data or knowledge, allowing only authorized entities with proper credentials. Throughout these centuries we notice that there is constant confrontation between the creators of the algorithms and the cryptanalyzers, the ones who try to break the algorithms. The past decades, this confrontation lead us to a certain point, where we can talk about provable security. We are now able to discuss freely about the properties of certain encryption algorithms, without the need to hide their structure in order to make them more secure, getting us closer to Augustine Kerckhoffs's principle as well. According to this principle, the strength of an encryption algorithm should rest only on the lack of key knowledge and not on the hidden properties or structure of the algorithm. If this principle is violated, then when either properties or structure is disclosed wholly or partially, the algorithm and the protected information become insecure.

Up to the 70's, all encryption algorithms belonged to the category of ciphers, that we now call symmetric or private key, where the same key is being used both for the encryption and the decryption. Rivest, Shamir and Adleman changed everything that we thought about cryptography and its uses, marking a new era, with the introduction of public or asymmetric key cryptography. The introduction of RSA and the wide use of

number theory in cryptography lead to new applications and wider adoption of cryptography, setting new security standards.

But what happens with Malory? In public key cryptography, people falsely believed that if we have a mechanism that provides us with the private and the public key that meet certain criteria and statistical properties, then we can use them to securely encrypt and decrypt our data. But could Malory create these keys? Could this be made undetected? Why should we trust the key creator and when should we trust him? What are the risks of doing so?

The questions above were answered many years later. Anderson was one of the firsts to study how one could tamper the key creation procedure [2]. Even though his scheme wasn't secure for Malory, as it was later shown by Kalinksi [20], the first cracks had been made. Young and Yung, studied the aforementioned questions and they were lead to the introduction of SETUPs proposing some algorithms that could be used. The new attack scenario sets Malory in another place in the attack scheme. Malory becomes the key generator and attacks those that use the tampered keys. We should note here a very important difference between private and public key cryptography. In the case of private key cryptography, each entity can create its own key without external help, since it is just a bitstream of certain length that has to be memorized or securely stored. In public key cryptography, things are quite different. The needed algebraic structure coupled with the needed key size, make it impossible for someone to pick the key pair on his own. In the case of RSA, it is impossible for someone to create primes of 300 decimal digits by hand and most programming languages are not able to handle big integers, e.g. integers of 1024 bits length. Therefore, special cryptographic libraries or encryption devices have to be used. People used to believe and some still do, that if the key creation mechanism is imposed to conform to certain standards, to produce keys with certain statistical properties and this mechanism is not able to communicate with another entity, then the produced keys are safe to be used. This scheme is shown in figure1.

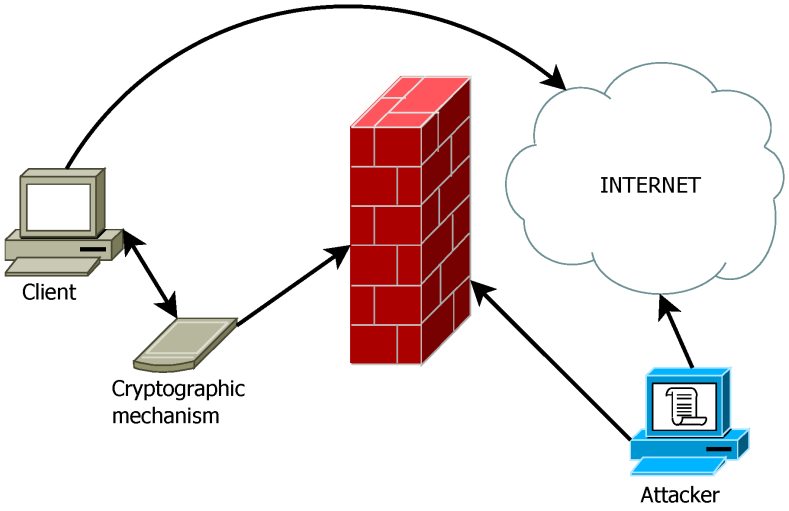

FIGURE 1. The misbelieved model.
The attacker may provide a client with a cryptographic mechanism (API, device etc). The misbelief is that since this mechanism cannot access the Internet the attacker cannot launch an attack on the produced keys.

However, all these depend on the direct communication of the encryption mechanism with another entity. Young and Yung, studied the possibility of having a covert channel,

leaking information to Malory. The attack scenario that they introduced is the following. Malory creates a trapdoor in a key generator. Malory tricks Alice into using her key generator to create public and private keys. Alice creates the key pair and publishes his public key on the internet. Malory gets Alice public key and using the trapdoor finds the private key. Malory can now decrypt every encrypted message that she intercepts from Alice. The trapdoor creates a covert channel between Malory and the private key, through Alice's public key.

The scenario above indicates that Malory could be the manufacturer of the encryption device, the developer of a cryptographic library or even an attacker that finds his way through the key generator. This type of attacks were named by Young and Yung SETUP (Secretly Embedded Trapdoor with Universal Protection). A simple model of a SETUP can be seen in figure2.

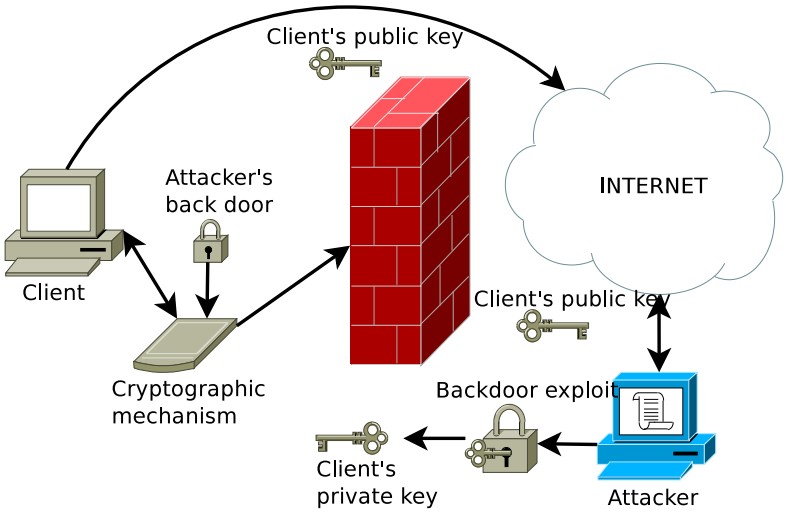

FIGURE 2. The SETUP model.

The attacker provides a cryptographic mechanism to the client with an embedded trapdoor. The client creates his public and private key pair with the provided mechanism, which acts as a black box to him. By getting the public key, the attacker is able to use it to exploit the embedded trapdoor and gain access to client's private key.

One of the main aspects that kleptography points out in cryptography, is the fact that the security of an encryption device or application, does not only rest upon the used algorithms or the safe implementation, but also on the trust to the manufacturer. The reason for studying SETUPs is that by figuring out how such implementations work, it might be possible to detect and recognize them, using reverse engineering techniques or timing attacks.

In this work the main focus is on the creation of SETUPs for factoring based encryption algorithms [26, 25], like RSA [28] or Rabin [27] or even less used algorithms, like Blum-Goldwasser [4] and Goldwasser-Micali [15]. The structure of this work is as follows, firstly the reader is introduced to some needed mathematical background. The following paragraph gives a brief description of some public key encryption algorithms, followed by a section on the previous work on SETUPs. The next two sections the proposed SETUPs are being analyzed. The following section discusses some experimental results. Finally, this work concludes pointing out some considerations and discusses some issues that arise from this research.

2. **Needed mathematical background.** In order to analyze the SETUPs it is essential to review some mathematical theorems and identities.

For the first SETUP, the main idea is based on Lagrange's four square theorem [12, 18] and [16].

Theorem 2.1. *Every positive integer n can be written as a sum of the squares of four integers.*

Example 2.1. $15 = 3^2 + 2^2 + 1^2 + 1^2$, $5413654687 = 73099^2 + 8378^2 + 1^2 + 1^2$

A very central role will play the Euler's identity, that is:

Theorem 2.2. *For every* $x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4$ *we have:*

$$(x_1^2 + x_2^2 + x_3^2 + x_4^2)(y_1^2 + y_2^2 + y_3^2 + y_4^2) = a^2 + b^2 + c^2 + d^2$$

where:

$$a = x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4$$

$$b = x_1y_2 - x_2y_1 + x_3y_4 - x_4y_3$$

$$c = x_1y_3 - x_3y_1 + x_4y_2 - x_2y_4$$

$$d = x_1y_4 - x_4y_1 + x_2y_3 - x_3y_2$$

If $n = 2^s(2x+1)$ and with $r_4(n)$ we denote the number of representations of an integer n as a sum of four squares, then

$$r_4(n) = \begin{cases} 8f(n) & for\ s = 0 \\ 24f(2x+1) & for\ s \neq 0 \end{cases}$$

where

$$f(x) = \sum_{d|x} d$$

Now denoting by $r_3(n)$ the number of representations of an integer n as a sum of three squares and by using a Hardy's conjecture [17], proved by Bateman [3] and later improved by Landau [21, 22], we have that:

Theorem 2.3.

$$\sum_{n \leq x} r_3(n) \cong \frac{4}{3}\pi x^{\frac{3}{2}}$$

From the above theorem it can easily be proved that

Theorem 2.4.

$$r_3(n) \cong 2\pi n^{\frac{1}{2}}$$

The proof is quite straight forward. We will use the previous theorem and try to express $r_3(n)$ as the difference of two sums. So, if we set $F(x) = \sum_{n \leq x} r_3(n)$ then obviously:

$$F(x+1) - F(x) = \sum_{n \leq x+1} r_3(n) - \sum_{n \leq x} r_3(n) = r_3(x+1)$$

By replacing now $F(x)$ and $F(x+1)$ we have that

$$F(x+1)-F(x)=\sum_{n\leq x+1} r_3(n)-\sum_{n\leq x} r_3(n)$$

$$\simeq \frac{4}{3}\pi(x+1)^{\frac{3}{2}}-\frac{4}{3}\pi x^{\frac{3}{2}}=\frac{4}{3}\pi((x+1)^{\frac{3}{2}}-x^{\frac{3}{2}})$$

$$=\frac{4}{3}\pi(\frac{(x+1)^{\frac{3}{2}}-x^{\frac{3}{2}}}{(x+1)^{\frac{3}{2}}+x^{\frac{3}{2}}}((x+1)^{\frac{3}{2}}+x^{\frac{3}{2}}))$$

$$=\frac{4}{3}\pi\frac{(x+1)^3-x^3}{(x+1)^{\frac{3}{2}}+x^{\frac{3}{2}}}$$

$$=\frac{4}{3}\pi\frac{x^3+3x^2+3x+1-x^3}{(x+1)^{\frac{3}{2}}+x^{\frac{3}{2}}}$$

$$=\frac{4}{3}\pi\frac{3x^2+3x+1}{(x+1)^{\frac{3}{2}}+x^{\frac{3}{2}}}$$

If x is big enough, then we have that

$$3x^2+3x+1\cong 3(x+1)^2$$

thus

$$(x+1)^{\frac{3}{2}}+x^{\frac{3}{2}}\cong 2(x+1)^{\frac{3}{2}}$$

So,

$$F(x+1)-F(x)\cong\frac{4}{3}\pi\frac{3(x+1)^2}{2(x+1)^{\frac{3}{2}}}$$

$$=2\pi(x+1)^{\frac{1}{2}}=r_3(x+1)$$

Finally, we have that

$$r_3(n)\cong 2\pi n^{\frac{1}{2}}$$

The reader may find more results regarding sums of squares of integers in [24].

Coppersmith's theorem [9] have been widely used in cryptography and cryptanalysis and will enable us to extract the private key from the public one, in the second SETUP. The theorem is the following.

Theorem 2.5. *Let N be an integer and $f\in\mathbb{Z}[x]$ a monic polynomial of degree d. Set $X=N^{\frac{1}{d}-\epsilon}$ for some $\epsilon\geq 0$. Given N and f, one can efficiently find all integers x_0, with $|x_0|<X$, satisfying*

$$f(x_0)\equiv 0 \ mod\ N$$

The running time is dominated by the time it takes the LLL algorithm on a lattice of dimension $O(w)$ with $w=min(\frac{1}{\epsilon},logN)$.

Coppermith later generalized it to the following [8] theorem for the case of two variables:

Theorem 2.6. *Let $p(x,y)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let X and Y be upper bounds on the desired integer solution (x_0,y_0), and let*

$$W=max_{i,j}|p_{ij}|X^iY^j$$

If $XY<W^{2/(3\delta)}$, then in time polynomial in $(logW,2^\delta)$, one can find all integer pairs (x_0,y_0) such that $p(x_0,y_0)=0,\ |x_0|\leq X$, and $|y_0|\leq Y$.

Several attacks on public key cryptography have been based on these two theorems like [6, 5, 7, 23], as these theorems allow the attacker to launch an attack from partial knowledge of key bits.

3. **Public key encryption algorithms.** In this section there's a review of two public key encryption algorithms, more precisely, RSA and ElGamal. The security of the first algorithm depends greatly on the lack of efficient integer factorization algorithms, yet to be more precise the security of RSA is based on the so called RSA problem. Given an integer c, a big integer n, which is a product of two primes and an integer e, co-prime to $\phi(n)$, there is no deterministic polynomial time algorithm to compute the solution x to the following equation

$$c = x^e \ mod \ n$$

Some results on this subject can be found in [10, 1, 19].

The second algorithm, ElGamal, bases its security in the lack of efficient algorithms for the calculation of discrete logarithm over finite fields. A more formal definition of the problem is the following: given a prime p and $a,\ b \in \mathbb{Z}_p^*$, there is no deterministic polynomial time algorithm to compute the solution x of the following equation

$$b = a^x \ mod \ p$$

The aim of this review is to clarify the place that the trapdoor is set for the proposed SETUPs.

The most widely used public key algorithm is RSA [28]. It was developed by Rivest, Shamir and Adleman in 1978 and remains secure to many attacks that have been developed against it. The algorithm has a very clear number theoretic outline and can be described with elementary number theory. RSA, as every public key algorithm has three different stages, initialization, encryption and decryption. During the initialization stage, each entity creates a pair of public and private keys. The procedure is the following

1. Create two large prime numbers p and q.
2. Calculate values $n = pq$ and $\phi(n) = (p-1)(q-1)$.
3. Select a random number e co-prime to $\phi(n)$. In most applications this is set as $2^{16}+1$.
4. Calculate d, the inverse of $e \ mod \ \phi(n)$, that is $ed \ mod \ \phi(n) \equiv 1$.

Then the public key is the pair (e, n) and is published, while d is the private key. The encryption and decryption processes are very similar. If Alice wants to send Bob message m, then all Alice has to do is find Bob's public key (e, n) and send him the message c, where

$$c \equiv m^e \ mod \ n$$

To decrypt the message c, Bob uses his private key d and calculates

$$c^d \ mod \ n$$

Since the order of multiplication group $mod\, n$ is $\phi(n)$ and

$$ed \ mod \ \phi(n) \equiv 1$$

we have that:

$$c^d \ mod \ n \equiv (m^e)^d mod \ n \equiv m^{ed} \ mod \ n \equiv m^1 \ mod \ n \equiv m$$

ElGamal is a public key encryption algorithm, developed by Taher El Gamal [13] in 1985 and its security is based on the difficulty of calculating discrete logarithm over finite fields. In the initialization of the algorithm, we first pick a big random prime p and set G the cyclic multiplicative group $(\mathbb{Z}_p^*, \times)$ and a generator g of this group. We pick a random

$x \in \{2, 3, ..., p-1\}$ and calculate $h = g^x mod\ p$. The private key is x, while the public key is the triple (p, g, h). The encryption process is the following. If entity B wants to send message m to entity A, then it obtains A's public key and picks a random $r \in (\mathbb{Z}_p^*, \times)$. Afterwords, B calculates values c_1 and c_2, where

$$c_1 = g^y mod\ p$$

$$c_2 = mh^y mod\ p$$

and sends (c_1, c_2) to A.

For the decryption process, A on receiving (c_1, c_2), decrypts the message, by calculating the value $c_2 c_1^{-x} mod\ p$ to obtain message m. This works because:

$$\begin{aligned} c_2 c_1^{-x} &\equiv m h^y (g^y)^{-x} mod\ p \\ &\equiv m (g^x)^y (g^y)^{-x} mod\ p \\ &\equiv m g^{xy} g^{-xy} mod\ p \equiv m \end{aligned}$$

4. **Previous work.** We start this section with a definition of SETUPs. A. Young and M. Yung, have the following definition [34, 32]:

Definition 4.1. *Let C be an honest black box cryptosystem that conforms to a public specification. Let C' be a dishonest version of C that contains a publicly known cryptotrojan algorithm, that was implemented by an attacker A, and that may contain secret seeding information that is not publicly known. Cryptosystem C' constitutes a SETUP version of C if the following properties hold:*

1. C and C' run in polynomial time.
2. The outputs of C and C' are indistinguishable to all efficient probabilistic algorithms, except for the attacker who can always distinguish and ...
3. The outputs of C are confidential to all efficient probabilistic algorithms and do not compromise the cryptosystem that C implements.
4. The outputs of C' are confidential to all efficient probabilistic algorithms except for the attacker A and do not compromise the cryptosystem that C' implements.
5. With overwhelming probability the attacker A can decrypt, forge or otherwise cryptanalyze at least one private output of C' given sufficient number of public outputs of C'.

The above definition is not as insightful as to what a SETUP is realy like, so it is more than appropriate to give an elementary example. Let's suppose that we want to create a SETUP for RSA. The attacker should be able to find the private key d efficiently, when all that he can get from the client is his public key pair (e, n). Let's suppose that the attacker has a secure keyed hash function H, then he can alter the key generation of the private key pair as follows.

1. Create two large prime numbers p and q.
2. Calculate values $n = pq$ and $\phi(n) = (p-1)(q-1)$.
3. Set $d = H(k, n)$, where k is the attacker's key. While $gcd(d, \phi(n)) \neq 1$
   (a) $n = n + 1$,$d = H(k, n)$
4. Calculate e, the inverse of $d\ mod\ \phi(n)$, that is $ed\ mod\ \phi(n) \equiv 1$.

It is more than obvious from their definition, that e, d may switch places. The key generation procedure above creates a random n of arbitrary length and two random looking exponents e and d, with $ed\ mod\ \phi(n) \equiv 1$. The attacker can now easily find the private key d, by calculating $H(k, n)$. If this is not the case, he tries a few more times with $H(k, n+1)$, $H(k, n+2)$ etc.

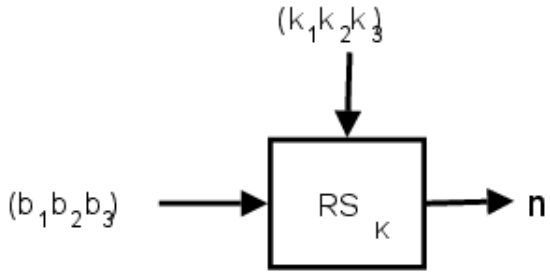

FIGURE 3

This SETUP, despite its simplicity, has an inherent flaw that does not allow it to be used. The SETUP can only work with e and d that it creates and not with fixed e, as it happens in many applications.

It is clear, that this hashing technique can easily be ported to ElGamal encryption algorithm, creating a secure SETUP. In this case, we have the same key generation procedure with the original one, with only one alternation. Instead of picking a random x, the attacker uses his secure hash function H to pick $x = H(p||g)$, where $||$ denotes the concatenation operator.

Several SETUPs have been proposed for the public key algorithms. The beginning was made with Anderson [2] in 1993, which Kaliski proved to be prone to attacks later the same year [20]. Young and Yung, apart from setting formal definitions and setting the new grounds for the foundations of kleptography, introduced improved methods for SETUP with PAP (Pretty-Awful-Privacy) [32, 33]. A simpler method has been proposed by Crepeau and Slakmon [11]. A revised SETUP of Young and Yung is given in [30]. For more on the applications of kleptography and generaly of malicious cryptography the reader may refer to [32, 31, 35, 14].

The two new SETUPs that are being presented in this work show the high dependability of SETUPs on number theory. Moreover, by studying possible attacks, we are able to detect and defend ourselves from other similar attacks.

5. **The four square method.** The SETUP that is presented in this section tries to take advantage of the Euler's identity we stated in a previous section. According to it, we have an identity which may lead us to a factorization of a big integer, under certain restrictions. The restriction is to bound the values of certain variables in the Euler's identity, up to a certain value. Then if we decompose the product properly, its prime factorization will be easily found.

Before presenting this SETUP, it is necessary to make a reference to Rabin and Shallit, who propose in [29] two randomized algorithms for decomposing integers as sum of squares. The first algorithm aims to the decomposition of integers as sum of two squares and has complexity of $O(log^2 n)$. The second one has complexity $O(log^2 n\ loglogn)$ and it is made for decomposing integers as sum of four squares. Both of them are randomized algorithms, hence we shall use three keyed pseudo-random number generators namely Gen_1, Gen_2 and Gen_3, with keys k_1, k_2 and k_3 respectively. This way, we create a keyed algorithm for the decomposition of an integer as sum of four squares which we call RS_K, Figure 5. This means that each key triplet $K = (k_1, k_2, k_3)$ decomposes differently an integer.

The SETUP is the following. Let's suppose that we have a mechanism M that honestly creates primes. M provides us with p and q, that their product n conforms to every

standard of RSA encryption referred above. From Lagrange's theorem we have that n can be written as a sum of squares of four integers. We apply RS_K to n to find a, b, c and d, thus:

$$n = a^2 + b^2 + c^2 + d^2$$

We can now form the following system of equations:

$$(x_1^2 + x_2^2 + x_3^2 + x_4^2)(y_1^2 + y_2^2 + y_3^2 + y_4^2) = a^2 + b^2 + c^2 + d^2$$

$$a = x_1y_1 + x_1y_1 + x_1y_1 + x_1y_1$$

$$b = x_1y_2 - x_2y_1 + x_3y_4 - x_4y_3$$

$$c = x_1y_3 - x_3y_1 + x_4y_2 + x_2y_4$$

$$d = x_1y_4 - x_4y_1 + x_2y_3 - x_3y_2$$

$$p = x_1^2 + x_2^2 + x_3^2 + x_4^2$$

$$q = y_1^2 + y_2^2 + y_3^2 + y_4^2$$

Of course this is a non-linear system of 7 equations and 8 variables. In order to simplify it, we set a base $B = (b_1, b_2, b_3)$ which determines the maximum values of variables $x_1,\ x_2$ and $y_1$respectively. By using every possible value of $x_1,\ x_2$ and y_1 from base B, we can solve the remaining equations for $x_3,\ x_4,\ y_2,\ y_3$ and y_4 efficiently. For the sake of simplicity, the equations to be checked have been omitted, yet they can easily be produced using a mathematical application like Mathematica or Matlab. If the system of equations is not solvable, a new pair of p and q is generated from M and the procedure is repeated until a proper pair is found.

It may seem that setting a threshold to values $x_1,\ x_2$ and y_1 would eliminate many decompositions, yet there are plenty such forms that meet our constrains. Let's suppose that we have a big prime q and set $0 \leq y_1 \leq 100$ then $q - 100^2 \cong q$ and we want to count how many representations as sum of three squares $q - 100^2$ has. From the proved theorem, there are about $2\pi\sqrt{q}$ such representations. So totally we have about $200\pi\sqrt{q}$ representations for which

$$q = y_1^2 + y_2^2 + y_3^2 + y_4^2$$

$$0 \leq y_1 \leq 100$$

and we have $n = pq$.

A base $B = (100, 1000, 100)$ suggests that when prime p is expressed as a sum of four squares, then the square of an integer of at most 100 and a square of an integer of at most 1000 appear. On the same time, q in its sum of squares representation has the square of an integer of at most 100. The restrictions of this base can be easily fulfilled, applying in worst case scenario $100 \times 1000 \times 100 = 10,000,000$ calculations for solving the system of equations, which can be thought in many cases an affordable cost.

The attack is quite obvious again, the attacker finds the client's public key and decomposes it with RS_K. Knowing that for this decomposition, the restrictions of B are met,

Algorithm 1 The pseudo-code of the proposed algorithm.

- while p is not prime
  - $r_1 = random\ integer\ in\ (0, \sqrt[3]{N-1})$
  - $p = (a_1 + r_1)^E\ mod\ N || r_1$
- while q is not prime
  - $r_2 = random\ integer\ in\ (0, \sqrt[3]{N-1})$
  - while $r_1 r_2 > \sqrt[3]{N-1}$
    - $r_2 = random(0, \sqrt[3]{N-1})$
  - $q = (a_2 + r_2)^E\ mod\ N || r_2$
- return (p, q)

he tries to solve the system of equations for every possible triplet of the bounded values $x_1,\ x_2$ and y_1.

6. **SETUP based on Coppersmith's theorem.** The second SETUP that is presented in this work uses Coppersmith's theorem in order to hide its trapdoor. This theorem enables an attacker to solve efficiently non-linear systems of equations, under certain restrictions. In the proposed SETUP scheme, the generation of the primes p and q is tampered to contain a random value and the attacker's key. The SETUP creates a product $n = pq$, whose first bits follow a certain non-detectable pattern.

Coppersmith's theorem allows an attacker knowing the upper half bits of the two prime factors to efficiently factor n. The presented SETUP needs the creation of the attacker key, which is composed by a secret RSA key (E, D, N) and two integer values a_1 and a_2.

To embed the backdoor in the client's RSA key, the attacker tries to find primes p and q of the following form:

$$p = (a_1 + r_1)^E\ mod\ N || r_1$$

$$q = (a_2 + r_2)^E\ mod\ N || r_2$$

where r_1 and r_2 are two random integers, whose magnitude will be determined later on and $||$ represents the concatenation operator. The outline of the algorithm is illustrated in algorithm 1.

This scheme works because if the attacker wants to recover the prime factorization, he may use Coppersmith's theorem to solve one binomial equation. To create the equatiom we first substitute p and q in the equality $n = pq$. So we have that:

$$n = pq = [(a_1 + r_1)^E\ mod\ N || r_1] * [(a_2 + r_2)^E\ mod\ N || r_2]$$

If we set r_1 and r_2 to be k bits long, then we can rewrite it as:

$$n = [(a_1 + r_1)^E\ mod\ N * 10^k + r_1] * [(a_2 + r_2)^E\ mod\ N * 10^k + r_2]$$

Now, applying $mod\ N$ to both sides, we have that:

$$n\ mod\ N \equiv [(a_1 + r_1)^E * 10^k + r_1] * [(a_2 + r_2)^E * 10^k + r_2]$$

which is an equation on two variables r_1 and r_2 of degree $2E$. For the sake of simplicity, we set $x = r_1$ and $y = r_2$, with their upper bounds, being X and Y respectively. In order to be able to have as many possible r_1 and r_2, while being able to solve the equation using Coppersmith's theorem we require $E = 3$ so that

$$XY < M^{2/2 \cdot 3} = \sqrt[3]{M}$$

where $M = max_{i,j}|p_{ij}|X^iY^j$. Making the substitutions, the equation to be solved is the following:

$$10^{2k}a_1^3a_2^3 - n + 10^k a_2^3 x + 3\cdot 10^{2k}a_1^2a_2^3 x + 3\cdot 10^{2k}a_1a_2^3x^2$$

$$+10^{2k}a_2^3x^3 + 10^k a_1^3 y + 3\cdot 10^{2k}a_1^3a_2^2 y + xy + 3\cdot 10^k a_1^2 xy$$

$$+3\cdot 10^k a_2^2 xy + 9\cdot 10^{2k}a_1^2a_2^2 xy + 3\cdot 10^k a_1 x^2 y + 9\cdot 10^{2k}a_1a_2^2x^2y$$

$$+10^k x^3 y + 3\cdot 10^{2k}a_2^2x^3y + 3\cdot 10^{2k}a_1^3a_2y^2 + 3\cdot 10^k a_2xy^2$$

$$+9\cdot 10^{2k}a_1^2a_2xy^2 + 9\cdot 10^{2k}a_1a_2x^2y^2 + 3\cdot 10^{2k}a_2x^3y^2$$

$$+10^{2k}a_1^3y^3 + 10^k xy^3 + 3\cdot 10^{2k}a_1^2xy^3 + 3\cdot 10^{2k}a_1x^2y^3$$

$$+10^{2k}x^3y^3 \equiv 0 \ mod\ N$$

Since we do not know the coefficients after their reduction modulo N, we may assume that $M = N - 1$, thus

$$XY < \sqrt[3]{N-1}$$

If we want X and Y to be of the same magnitude we finally have that

$$r_1, r_2 < \sqrt[6]{N-1}$$

which means that if the public key of the attacker is small, let's say that it has 512 bits length, then for each r_1 and r_2 we have about 2^{85} possible values to select. The wide range of random seeds leads to a very wide selection of possible primes. Moreover, since both the upper and the lower part, is randomized, it is impossible without the knowledge of the attacker's key to recover or even detect the SETUP. As it is obvious from the test results of the SETUP, that will be illustrated in the following section, the effective randomization that it provides, enables the quick creation of possible primes.

7. **Experimental results.** In this work, only the experimental results of the second algorithm are included. The reason for excluding experimental results of the first algorithm is that the process to find proper primes p and q is significant and may reveal the existence of the SETUP. However, the same does not apply in the case of the second algorithm. The key generation tests were made on a system with Intel Core2 Quad CPU Q9400 at 2.66GHz with 4 GB RAM, running on Ubuntu 10.04, with kernel 2.6.32-24 and using Wolfram Mathematica 7. Figures 4 and 5 present the time it takes to create RSA keys of 1024 bits length with an honest algorithm and the SETUP. The average time for the honest key creation is about 0.156 secs and of course is comparable to 0.17 secs of the presented SETUP. The same applies to their standard deviation, which is 0.03 secs and 0.11 secs respectively.

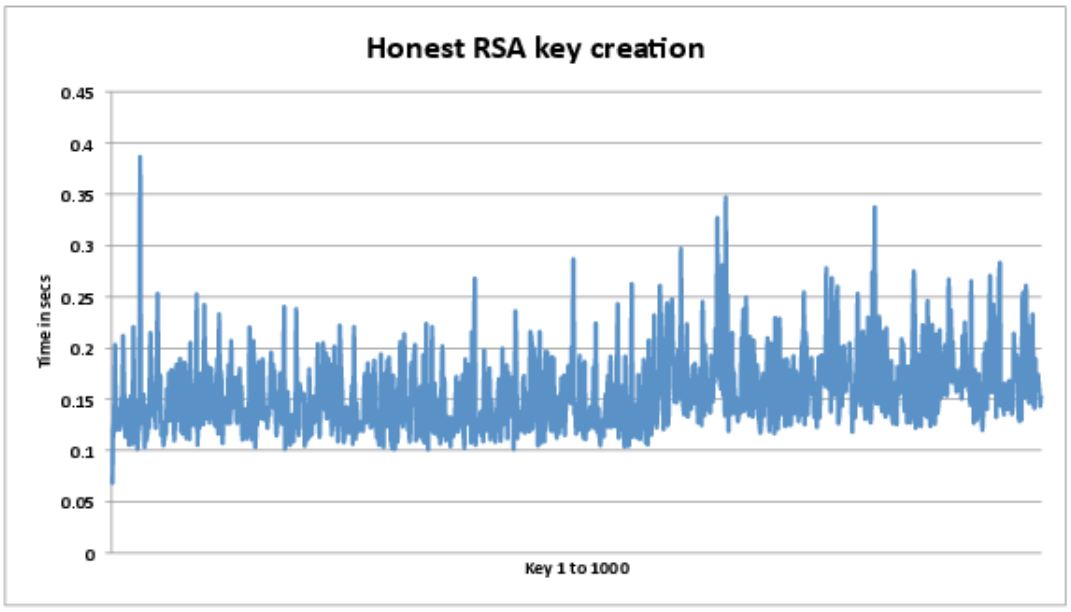

FIGURE 4. Honest RSA key creation time for 1000 keys.

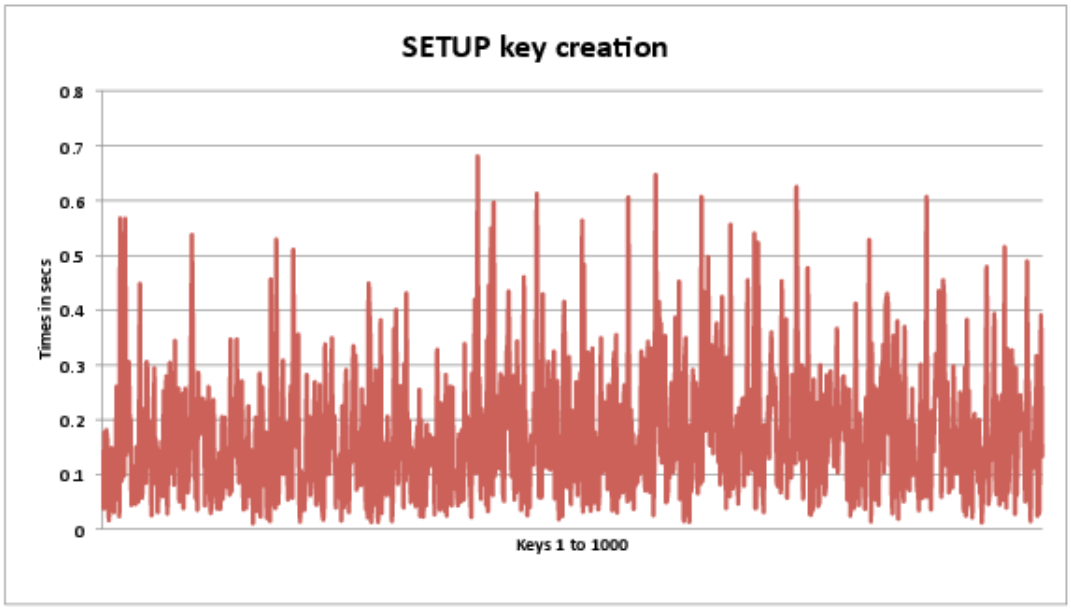

FIGURE 5. RSA keys creation time with SETUP.

8. **Conclusions.** Creating and studding SETUPs might of course seem like working against security, on the side of the attackers, and many arguments can be said about that. In any case the reality is that in order for someone to understand how to defend against attackers, he must study how they are attacking or how they might attack. Kleptography exposes the dark side of cryptography. It tries to identify and exploit the risks that a user is exposed to by the use of encryption algorithms, mechanisms and devices, where companies or developers are hidding the code being used. Therefore, when it comes to using encryption algorithms, specially if their implementation remains a blackbox to the user, then the user must trust the provider of this implementation.

The two presented SETUPs can be applied to all RSA like factoring based encryption algorithms. The first algorithm might be more time consuming, yet the second one is comparable to the honest one, as it provides not only good key creation times, but a wide range of primes to be selected thus makes it untraceable. The use of random numbers, in conjunction with modular exponentiation, circumvents the problem of embedding secret information in the key and on the same time creating primes with no identifiable pattern

Both algorithms, for key extraction, due to their number theoretic roots, require the solution of some modular equations and can be implemented efficiently. It should be noted that he backdoor embedding process in both algorithms, does not undermine the strength of the encryption algorithms from outside attacks, therefore the user remains secure from others. Not only these SETUPs are untraceable, but even if someone tries to exploit the possibility of their existence, hewill fail, unless of course if he can guess the used key.

Disclaimer and acknowledgments. The author is with the UNESCO Chair in Data Privacy, but he is solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization. This work was partly supported by the Spanish Government through project CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES".

REFERENCES

[1] D. Aggarwal and U. Maurer, Breaking RSA generically is equivalent to factoring, *Proc. of 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 5479, pp. 36-53, 2009.

[2] R. J. Anderson, A practical RSA trapdoor, *Journal of Electronics Letters*, vol. 29, no.11, pp. 995, 1993.

[3] P. T. Bateman, On the representations of a number as the sum of three squares, *Trans. American Mathematical Society*, vol. 71, no. 1, pp. 71-101, 1951.

[4] M. Blum and S. Goldwasser, An efficient probabilistic public key encryption scheme which hides all partial information, *Proc. of 4th International Cryptology Conference*, pp. 289-299, 1985.

[5] D. Boneh and G. Durfee, Cryptanalysis of RSA with private key d less than $N^{0.292}$, *IEEE Trans. Information Theory*, vol. 46, no. 4, pp. 1339-1349, 2000.

[6] J. S. Coron, A. Joux, I. Kizhvatov, D. Naccache and P. Paillier, Fault attacks on RSA signatures with partially unknown messages, *Proc. of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 5747, pp. 444-456, 2009.

[7] D. Coppersmith, M. Franklin, J. Patarin and M. Reiter, Low-exponent RSA with related messages, *Proc. of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, pp. 1-9, 1996.

[8] D. Coppersmith, Finding a small root of a bivariate integer equation; factoring with high bits known, *Proc. of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, pp. 178-189, 1996.

[9] D. Coppersmith, Finding a small root of a univariate modular equation, *Proc. of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, pp. 155-165, 1996.

[10] J. S. Coron and A. May, Deterministic polynomial-time equivalence of computing the RSA secret key and factoring, *Journal of Cryptology*, vol. 20, no. 1, pp. 39-50, 2007.
[11] A. S. C. Crepeau, Simple backdoors for RSA key generation, *Proc. of Cryptographers' Track at the RSA Conference*, pp. 403-416, 2003.
[12] L. E. Dickson, *History of the Theory of Numbers, Volume II: Diophantine Analysis*, Dover Books on Mathematics, 2005.
[13] T. Elgamal, A public-key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Information Theory*, vol. 31, no. 4, pp. 469-472, 1985.
[14] E. Filiol, Anti-forensic techniques based on malicious cryptography, *Proc. of the 9th European Conference on Information Warfare and Security*, pp. 63-70, 2010.
[15] S. Goldwasser and S. Micali, Probabilistic encryption and how to play mental poker keeping secret all partial information, *Proc. of the 14th Annual ACM Symposium on Theory of Computing*, pp. 365-377, 1982.
[16] E. Grosswald, *Representations of Integers as Sums of Squares*, Springer-Verlag, 1985.
[17] G. H. Hardy, On the expression of a number as the sum of two squares, *Quarterly Journal of Mathematics*, vol. 46, pp. 263-283, 1915.
[18] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, Sixth Edition. Oxford University Press, 2008.
[19] A. Joux, D. Naccache and E. Thomé, When e-th roots become easier than factoring, *Proc. of the 13rd International Conference on the Theory and Application of Cryptology and Information Security*, pp. 13-28, 2007.
[20] B. S. K. Jun, Anderson's RSA trapdoor can be broken, *Journal of Electronics Letters*, vol. 29, no. 15, pp. 1387-1388, 1993.
[21] E. Landau, *Elementary Number Theory*, USA, Chelsea Publishing Company, 1958.
[22] E. Landau, *Collected Works*, Thales-Verlag, 1986.
[23] A. May, Cryptanalysis of unbalanced RSA with small CRT-exponent, *Proc. of 22nd Annual International Cryptology Conference*, pp. 242-256, 2002.
[24] C. J. Moreno and S. S. Wagstaff Jr., *Sums of Squares of Integers*, FL, USA, Chapman & Hall/CRC, 2005.
[25] C. Patsakis and N. Alexandris, A new SETUP for factoring based algorithms, *Proc. of 6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 200-203, 2010.
[26] C. Patsakis and E. Fountas, Creating RSA trapdoors using Lagrange four Square theorem, *Proc. of 5th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 779-782, 2009.
[27] M. O. Rabin, Digitalized signatures and public-key functions as intractable as factorization, *Massachusetts Institute of Technology Cambridge*, MA, USA, 1979.
[28] R. L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Journal of Communications of the ACM*, vol. 21, pp. 120-126, 1978.
[29] J. O. Shallit and M. O. Rabin, Randomized algorithms in number theory, *Proc. of the Symposium on Frontiers of the Mathematical Sciences*, vol. 39, pp. S239-S256, 1985.
[30] A. Young, M. Yung, Malicious cryptography: kleptographic aspects, *Proc. of the Cryptographers Track at the RSA Conference*, pp. 7-18, 2005.
[31] A. Young, and M. Yung, Kleptography from standard assumptions and applications, *Proc. of the 7th International Conference on Security and Cryptography for Networks*, pp. 271-290, 2010.
[32] A. Young, and M. Yung, *Malicious Cryptography: Exposing Cryptovirology*, Manhattan, USA, John Wiley and Sons, 2004.
[33] A. Young, and M. Yung, A space efficient backdoor in RSA and its applications, *Journal of Selected Areas in Cryptography*, pp. 128-143, 2005.
[34] A. Young, and M. Yung, The dark side of "Black-Box" cryptography or: should we trust capstone?, *Proc. 16th Annual International Cryptology Conference*, pp. 89-103, 1996.
[35] A. Young, and M. Yung, Kleptography: Using Cryptography Against Cryptography, *Proc. of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, pp. 62-74, 1997.