# A New Public Key Cryptosystem Based on Higher Residues


**David Naccache**
Gemplus Card International
34 rue Guynemer
Issy-les-Moulineaux CEDEX, 92447, France
naccache@compuserve.com

**Jacques Stern**
Ecole Normale Supérieure
45 rue d'Ulm
Paris CEDEX 5, 75230, France
jacques.stern@ens.fr



**Abstract**

This paper describes a new public-key cryptosystem based on the hardness of computing higher residues modulo a composite RSA integer. We introduce two versions of our scheme, one deterministic and the other probabilistic. The deterministic version is practically oriented: encryption amounts to a single exponentiation w.r.t. a modulus with at least 768 bits and a 160-bit exponent. Decryption can be suitably optimized so as to become less demanding than a couple RSA decryptions. Although slower than RSA, the new scheme is still reasonably competitive and has several specific applications. The probabilistic version exhibits an homomorphic encryption scheme whose expansion rate is much better than previously proposed such systems. Furthermore, it has semantic security, relative to the hardness of computing higher residues for suitable moduli.


## 1 Introduction

It is striking to observe that two decades after the discovery of public-key cryptography, the cryptographer's toolbox still contains very few asymmetric encryption schemes. Consequently, the search for new public-key mechanisms remains a major challenge. The quest appears sometimes hopeless as new schemes are immediately broken or, if they survive, are compared with RSA, which is obviously elegant, simple and efficient.

Similar investigations have been relatively successful in the related setting of identification, where a user attempts to convince another entity of his identity by means of an on-line communication. For example, there have been several attempts to build identification protocols based on simple operations (see [33, 35, 36, 26]). Although the question of devising new public-key cryptosystems appears much more difficult (since it deals with trapdoor functions rather than simple one-way functions), we feel that research in this direction is still in order: simple yet efficient constructions may have been overlooked.

The scheme that we propose in the present paper uses an RSA integer $n$ which is a product of two primes $p$ and $q$, as usual. However, it is quite different from RSA in many respects:

1. it encrypts messages by exponentiating them with respect to a fixed base rather than by raising them to a fixed power
2. it uses a different "trapdoor" for decryption
3. its strength is not directly related to the strength of RSA
4. it exhibits further "algebraic" properties that may prove useful in some applications.

We briefly comment on those differences. The first one may offer a competitive advantage in environments where a large amount of memory is available: such environments allow impressive speed-ups in exponentiations that do not have analogous counterparts in RSA-like operations. The second is of obvious interest in view of the fact quoted above that there are very few public-key cryptosystems available. Without going into technical details at this point, let us simply mention that the new trapdoor is obtained by injecting small prime factors in $p-1$ and $q-1$. In order to understand what the third difference is, we note that, if the modulus $n$ can be factored, then both RSA and the proposed cryptosystem are broken. However, it is an open problem whether or not RSA is "equivalent" to factoring, which would mean that breaking RSA allows to factor. For this reason, the hypothesis that RSA is secure has become an assumption of its own, formally stronger than factoring. Our cryptosystem is related to another hypothesis, also formally stronger than factoring and known as the higher residuosity assumption. This may help to understand how these various hypotheses are related. Finally, we will explain the algebraic property of our scheme (called the *homomorphic property*) by means of an example: suppose that one wishes to withdraw a small amount $u$ from the balance $m$ of some account; assume further that the balance is given in encrypted form $E(m)$ and that the clerk performing the operation does not have access to decryption. The cryptosystem that we propose simply solves the problem by computing $E(m)/E(u) \bmod n$, which turns out to be the encryption of the new balance $m-u$.

The ability to perform algebraic operations such as additions or substractions by playing only with the cryptograms has potential applications in several contexts. We quote a few:

1. in election schemes, it provides a tool to obtain the tally without decrypting the individual votes (see [4])

2. in the area of watermarking, it allows to add a mark to previously encrypted data (as explained in [25]).

Still, in these contexts, it is often needed to encrypt data taken from a small set $S$ (e.g. 0/1 votes) and it is well known that deterministic cryptosystems, such as RSA, fail here: in order to decrypt $E(a)$, one can simply compare the ciphertext with the encryptions of all members of $S$ and thus find the correct value of $a$. In order to overcome the difficulty, one has to use probabilistic encryption, where each plaintext has many corresponding ciphertexts, depending on some additional random parameter chosen at encryption time. Such a scheme should make it impossible to distinguish encryptions of distinct values, even if these are restricted to range over a set with only two elements. This very strong requirement has been termed *semantic security* ([12]). As a further difference with RSA, the cryptosystem introduced in this paper, has a very natural probabilistic version, with proven semantic security.

The probabilistic homomorphic encryption schemes proposed so far suffer from a serious drawback: they have very poor bandwith. Typically, they need something like one kilobit to encrypt just a few bits, which is a quite severe expansion rate. This may be acceptable for election schemes but definitely hampers other applications. The main achievement of the present paper is to reach a significant bandwith, while keeping the other properties, including semantic security.

Before we turn to the more technical developments of our paper, it is in order to compare it with earlier work: it is indeed the case that the question of finding trapdoors for the discrete logarithm problem has been the subject of many papers. At this point, it is fair to mention that the probabilistic cryptosystem that we propose is actually quite close to the most general case of the homomorphic encryption schemes introduced by Benaloh in his Ph-D thesis [4]. Still, both in this thesis and in the related work ([5, 6, 7]), the security and potential applications are only investigated in a setting where the bandwith remains small. A more recent paper by Park and Won (see [24])describes a related probabilistic cryptosystem using a trapdoor based on injecting a single power of a small odd integer into $p-1$ or $q-1$ and proves its security with respect to an *ad hoc* statement. Thus, our paper offers the first thorough discussion of the security of a probabilistic homomorphic encryption scheme with significant bandwith. After the completion of the present work, we have been informed that another homomorphic probabilistic encryption scheme, using moduli $n$ of the form $p^2q$, where $p$ and $q$ are primes, had been found by Okamoto and Uchiyama (see [22]), achieving an expansion rate similar to ours. Finally, it should be emphasized that the deterministic version of our scheme is not simply a twist that fixes the random string in the probabilistic version: considering its practicality, we believe that, even if it is not intended to be a direct competitor to RSA, it enters the very limited list of efficient public-key cryptosystems.

The paper is organized as follows: in the next two sections, we successively describe the deterministic and the probabilistic version of our scheme, the former with a practical approach, the latter in a more complexity-theoretic spirit. We then discuss applications and end up with a challenge for the research community.

## 2 The deterministic version

As was just mentioned, our approach to the deterministic scheme is practically oriented: we discuss system set-up and key-generation, encryption and decryption, with performances in mind. We also carry on a security analysis at the informal level and we derive minimal sugested parameters.

### 2.1 System set-up and key generation

The scheme that we propose in the present paper can be described as follows: let $\sigma$ be a squarefree odd $B$-smooth integer, where $B$ is small integer and let $n = pq$ be an RSA modulus such that $\sigma$ divides $\phi(n)$ and is prime to $\phi(n)/\sigma$. Typically, we think of $B$ as being a 10 bit integer and we consider $n$ to be at least 768 bits long. Let $g$ be an element whose multiplicative order modulo $n$ is a large multiple of $\sigma$. Publish $n$, $g$ and keep $p$, $q$ and optionally $\sigma$ secret. A message $m$ smaller than $\sigma$ is encrypted by $g^m \bmod n$; decryption is performed using the prime factors of $\sigma$ as will be seen in the next subsection.

Generation of the modulus appears rather straightforward: pick a family $p_i$ of $k$ small odd distinct primes, with $k$ even. Set $u = \prod_{i=1}^{k/2} p_i$, $v = \prod_{k/2+1}^{k} p_i$ and $\sigma = uv = \prod_{i=1}^{k} p_i$. Pick two large primes $a$ and $b$ such that both $p = 2au + 1$ and $q = 2bv + 1$ are prime and let $n = pq$.

However, this generation is lengthy especially when the size of the modulus grows: $a$ has to be chosen in the appropriate range and tested for primality as well as $p = 2au + 1$ until both tests succeed simultaneously. This might be a bit time-consuming. Instead, we suggest to generate $a$, $b$, $u$ and $v$ first (independently of any primality requirements on $p$ and $q$) and use a couple of 24-bit "tuning primes" $p'$ and $q'$ (not used in the encryption process) such that $p = 2aup' + 1$ and $q = 2bvq' + 1$ are primes. To avoid interferences with the encryption mechanics, we recommend to make sure that $\gcd(p'q', \sigma) = 1$ and $p' \neq q'$. In practice, such an approach is only 9% slower than equivalent-size RSA key-generation.

To select $g$, one can choose it at random and check whether or it has order $\phi(n)/4$. The main point is to ensure that $g$ is not a $p_i$-th power, for each $i \leq k$ by testing that $g^{\frac{\phi(n)}{p_i}} \neq 1 \bmod n$. The success probability is :

$$\pi = \prod_{i=1}^{k} (1 - \frac{1}{p_i}), \text{ whose logarithm is : } \ln(\pi) \simeq -\sum_{i=1}^{k} \frac{1}{p_i}$$

If the $p_i$s are the first $k$ primes, this in turn can be estimated as $-\ln\ln k$ and results in the quite acceptable overall probability of $\pi \cong 1/\ln k$. Another method consists in choosing, for each index $i \leq k$, a random $g_i$ until it is not a $p_i$-th power. With overwhelming probability $g = \prod_{i=1}^{k} g_i^{\sigma/p_i}$ has order $\geq \phi(n)/4$.

### 2.2 Encryption and Decryption

Encryption consists in a single modular exponentiation: a message $m$ smaller that $\sigma$ is encrypted by $g^m \bmod n$. Note that it does not require knowledge of $\sigma$. A lower bound (preferably a power of two) is enough but it is unclear how important for the security of the scheme is keeping $\sigma$ secret. However, if one chooses to keep $\sigma$ secret, necessary precautions (similar to these applied to Rabin's scheme [31] or Shamir's RSA for paranoids [34]) should be enforced for not being used as an oracle[1].

[1] For example, an attacker having access to a decryption box can decrypt $g^m \bmod n$ for some $m > \sigma$ and get $m \bmod \sigma$. This discloses (by subtraction) a multiple of $\sigma$ and $\sigma$ can then be found by a few re-

Also, there is actually no reason why the $p_i$s should be prime. Everything goes through, *mutatis mutandis*, as soon as the $p_i$s are mutually prime. Thus, for example, they can be chosen as prime powers, which is a way to increase the variability of the scheme.

Decryption is based on the chinese remainder theorem. Let $p_i$, $1 \le i \le k$, be the prime factors of $\sigma$. The algorithm computes the value $m_i$ of $m$ modulo each $p_i$ and gets the result by chinese remaindering, following an idea which goes back to the Pohlig-Hellman paper [27]. In order to find $m_i$, given the ciphertext $c = g^m \bmod n$, the algorithm computes $c_i = c^{\frac{\phi(n)}{p_i}} \bmod n$, which is exactly $g^{\frac{m_i\phi(n)}{p_i}} \bmod n$. This follows from the following easy computations, where $y_i$ stands for $\frac{m-m_i}{p_i}$:

$$c_i = c^{\frac{\phi(n)}{p_i}} = g^{\frac{m\phi(n)}{p_i}} = g^{\frac{(m_i+y_ip_i)\phi(n)}{p_i}}$$

$$= g^{\frac{m_i\phi(n)}{p_i}} g^{y_i\phi(n)} = g^{\frac{m_i\phi(n)}{p_i}} \bmod n$$

By comparing this result with all possible powers $g^{\frac{j\phi(n)}{p_i}}$, it finds out the correct value of $m_i$. In other words, one loops for $j = 0$ to $p_i - 1$ until $c_i = g^{\frac{j\phi(n)}{p_i}} \bmod n$.

The cleartext $m$ can therefore be computed by the following procedure :

```
for i = 1 to k
{
  let c_i = c^{φ(n)/p_i} mod n
  for j = 0 to p_i - 1
 {if c_i == g^{jφ(n)/p_i} mod n let m_i = j}
}
x = ChineseRemainder({m_i}, {p_i})
```

The basic operation used by this (non-optimized) algorithm is a modular exponentiation of complexity $\log^3(n)$, repeated less than :

$$k\, p_k < \log(n)\, p_k \cong \log(n)\, k\, \log(k) < \log^2(n)\, \log\log(n)$$

times. Decryption therefore takes $\log^5(n)\log\log(n)$ bit operations.

This is clearly worse than the $\log^3(n)$ complexity of RSA but encryption can be optimized if a table stores all possible values of $t[i,j] = g^{\frac{j\phi(n)}{p_i}}$, for $1 \le i \le k$ and $1 \le j \le i$: the value $m_i$ of the cleartext $m$ modulo $p_i$ is found by table look-up, once $c^{\frac{\phi(n)}{p_i}} \bmod n$ has been computed. It is not really necessary to store all $g^{\frac{j\phi(n)}{p_i}}$. Any hash function that distinguishes $g^{\frac{j\phi(n)}{p_i}}$ from $g^{\frac{j'\phi(n)}{p_i}}$, for $j \ne j'$ will do and, in practical terms, a few bytes will be enough, for example approximately $2|p_i|$ bits from each $t[i,j]$. It is even possible to use hash functions that do not discriminate values of $g^{\frac{j\phi(n)}{p_i}}$: the proper one is spotted by considering, by table look-up hashes of $g^{\frac{2^\ell j\phi(n)}{p_i}}$, for $\ell = 1, 2, \cdots$ until there is no ambiguity. This can be very efficiently implemented by storing hash values in increasing order w.r.t. $\ell$ and one single bit might be enough.

peated trials and gcds. To prevent such an action, the decryption box cannot only re-encrypt and check against the ciphertext received, as this allows a search by dichotomy. It should first check that the cleartext is in the appropriate range, e.g. $< 2^t$ with $2^t < m$, re-encrypt it and then check that it matches up with the original ciphertext before letting anything out.

### 2.3 A toy example

• **key generation for** $k = 6$
$p$ $21211 = 2 \times 101 \times 3 \times 5 \times 7 + 1$,

$q = 928643 = 2 \times 191 \times 11 \times 13 \times 17 + 1$,

$n = 21211 \times 928643 = 19697446673$ and $g = 131$ yield the table:

| | $i=1$ | $i=2$ | $i=3$ | $i=4$ | $i=5$ | $i=6$ |
|---|---|---|---|---|---|---|
| $j=0$ | 0001 | 0001 | 0001 | 0001 | 0001 | 0001 |
| $j=1$ | 1966 | 6544 | 1967 | 6273 | 6043 | 0372 |
| $j=2$ | 9560 | 3339 | 4968 | 7876 | 4792 | 7757 |
| $j=3$ | | 9400 | 1765 | 8720 | 0262 | 3397 |
| $j=4$ | | 5479 | 6701 | 7994 | 0136 | 0702 |
| $j=5$ | | | 6488 | 8651 | 6291 | 4586 |
| $j=6$ | | | 2782 | 4691 | 0677 | 8135 |
| $j=7$ | | | | 9489 | 1890 | 3902 |
| $j=8$ | | | | 8537 | 6878 | 5930 |
| $j=9$ | | | | 2312 | 2571 | 6399 |
| $j=10$ | | | | 7707 | 7180 | 6592 |
| $j=11$ | | | | | 8291 | 9771 |
| $j=12$ | | | | | 0678 | 0609 |
| $j=13$ | | | | | | 7337 |
| $j=14$ | | | | | | 6892 |
| $j=15$ | | | | | | 3370 |
| $j=16$ | | | | | | 3489 |

where entry $\{i,j\}$ contains $g^{j\phi(n)/p_i} \bmod n \bmod 10000$.
• **encryption of** $m = 202$

$$c = g^m \bmod n = 131^{202} \bmod 19697446673 = 519690214$$

• **decryption**
by exponentiation, we retrieve :

$$c^{\frac{\phi(n)}{p_1}} \bmod n \bmod 10000 = 1966$$
$$c^{\frac{\phi(n)}{p_2}} \bmod n \bmod 10000 = 3339$$
$$c^{\frac{\phi(n)}{p_3}} \bmod n \bmod 10000 = 2782$$
$$c^{\frac{\phi(n)}{p_4}} \bmod n \bmod 10000 = 7994$$
$$c^{\frac{\phi(n)}{p_5}} \bmod n \bmod 10000 = 1890$$
$$c^{\frac{\phi(n)}{p_6}} \bmod n \bmod 10000 = 3370$$

wherefrom, by table lookup :

$$m \bmod 3 = \texttt{table(1966)} = 1$$
$$m \bmod 5 = \texttt{table(3339)} = 2$$
$$m \bmod 7 = \texttt{table(2782)} = 6$$
$$m \bmod 11 = \texttt{table(7994)} = 4$$
$$m \bmod 13 = \texttt{table(1890)} = 7$$
$$m \bmod 17 = \texttt{table(3370)} = 15$$

and by Chinese remaindering : $m = 202$.

### 2.4 Suggested parameters and security analysis

We suggest to take $\sigma > 2^{160}$ and we consider $|n| = 768$ bits as a minimum size for the modulus.

If the factorization of $n$ is found, then $a$ and $b$ become known as well as $\phi(n)$. The scheme is therefore broken. However, the scheme does not appear to be provably equivalent to factoring. Rather, it is related to the question of having oracles that decide whether or not a random number $x$ is a $p_i$-th power modulo $n$, for $i = 1, \dots, k$. This is known as the higher residuosity problem and is currently considered unfeasible. Formal equivalence of this problem and the probabilistic version of our encryption scheme will be proved in the next session. Considering the basic deterministic version, we have no formal proof but we haven't found any plausible line of attack either. Also, the efficient factoring methods such as the quadratic sieve (QS) or the number field sieve (NFS) do not appear to take any advantage from the side information that $u$ (resp. $v$) divides $p-1$ (resp. $q-1$). The same is true of simpler methods like Pollard's $p-1$ since we have ensured that neither $p-1$ nor $q-1$ is smooth. Finally, elliptic curve weaponry [18] will not pull-out factors of $n$ in the range considered. Note that the requested size of $n$ (768 bits or more) makes factoring $n$ a very hard task anyway.

We now turn the size of $\sigma$. In order to avoid the computation of discrete logarithms by the baby step-giant step method, we have to make $\sigma$ large enough. As already stated, $2^{160}$ is a minimum. This can be achieved for example by making $\sigma$ a permutation of the first 30 odd primes, which yields $\sigma \simeq 2^{160.45}$. Alternatively, one one can choose a sequence of 16 primes with 10 bits. Since there are 75 such primes, this leads to a $\cong$ 53-bit entropy. Adding prime powers, as stated above, will further increase these figures.

There is a further difficulty, when $\sigma$ is known. Note that

$$4ab = \frac{\phi(n)}{\prod_{i=1}^{k} p_i} = \frac{n-p-q+1}{\sigma}$$

hence $4ab$ differs from $\frac{n}{\sigma}$ only by $\epsilon = -\frac{p+q-1}{\sigma}$. The numerator is of size $|n|/2$, hence, if it does not exceed the denominator by a fairly large number of bits, the value of $ab$ is basically known and decryption can be performed.

When the exact splitting of the factors of $\sigma$ into $u$ and $v$ are known as well, the previous analysis can be pushed further. Reducing the relation $n = (2au+1)(2bv+1)$ modulo $u$, we find that $n = 2bv + 1 \bmod u$ and we can calculate $d = b \bmod u$. Similarly, we learn $c = a \bmod v$. We let $a = rv + c$ and $b = su + d$, with $r$, $s$ unknown and, using the fact that $\sigma = uv$, we obtain:

$$n = (2rvu + 2cu + 1)(2suv + 2dv + 1) =$$

$$4rs\sigma^2 + 2\sigma[r(2dv+1) + s(2cu+1)] + (2cu+1)(2dv+1)$$

which is of the form

$$n = 4rs\sigma^2 + 2\sigma(\alpha r + \beta s) + \gamma$$

with known $\alpha$, $\beta$ and $\gamma$. Reducing modulo $\sigma^2$, this provides the value $\delta$ of $\alpha r + \beta s \bmod \sigma$. At this point, our analysis becomes quite technical and the reader may skip the following and jump to the conclusion that $n >> \sigma^4$.

For the interested reader, we note that the pair $(r, s)$ lies in the two-dimensional lattice $L$ defined by

$$L = \{(x, y) | \alpha x + \beta y = \delta \bmod \sigma\}$$

This lattice has determinant $\sigma$. Also, it is easily seen that $\alpha$ and $\beta$ are bounded by $2\sigma$ and $\gamma$ by $4\sigma^2$. From this we get

$$rs \le \frac{n}{4\sigma^2} \le rs + r + s + 1 = (r+1)(s+1)$$

Thus, the pair $(r, s)$ is very close to the boundary of the curve $C$ with equation $xy = \frac{n}{4\sigma^2}$. More precisely, the distance between the pair $(r, s)$ and the curve does not exceed $\sqrt{2}$. This defines a geometric area $A$ that includes $(r, s)$. Now, key generation usually induces constraints that limit the possible range of the parameters. For this reason, it is appropriate to replace $C$ by the line $x + y = \frac{\sqrt{n}}{2\sigma}$ in order to estimate the size of $A$. This leads to an approximation which is $O(\frac{\sqrt{n}}{\sigma})$. The number of lattice points from $L$ in this area is, in turn, measured by the ratio between the size of $A$ and the determinant, which is $\frac{\sqrt{n}}{\sigma^2}$. It is safe to ensure that this set is beyond exhaustive search, which we express by $n >> \sigma^4$.

Note that the ratio $|n|/|\sigma|$ is the expansion rate of the encryption, where $|n|$ denotes, as usual, the size of $n$ in bits. It is of course desirable to make this rate as low as possible. On the other hand, as a consequence of the above remarks, we see that $\frac{|n|}{4} - |\sigma|$ should be large. Asymptotically, this is achieved as soon as we fix an expansion rate which is $> 4$. For real-size parameters, we suggest to respect the heuristic bound $\frac{|n|}{4} - \sigma \ge 128$, which is consistent with our minimal parameters. Larger parameters allow a slightly better expansion rate.

### 2.5 Performances

Despite its expansion rate, the new cryptosystem is quite efficient: encryption requires the elevation of a constant 768-bit number to a 160-bit power. Several batch ([21, 23]) and pre-processing ([2]) techniques can speed-up such computations, which might be a small advantage over RSA.

Decryption is slightly more awkward since $k$ exponentiations are needed. But this number can be reduced in a few ways :

Firstly, while computing $c^{\phi(n)/p_i} \bmod n$ for each $i$, it is possible to first store $c' = c^{4ab} \bmod n$ and raise $c'$ to the successive powers $\sigma/p_i$ so that (besides the first one), the remaining exponentiations involve 160-bit powers. One can further, in the square-and-multiply algorithm, share the "square" part of the various exponentiations. A careful bookkeeping of the number of modular multiplications obtained by setting $|n| = 768$ and choosing sixteen 10-bit primes $p_i$, shows that the total number of modular multiplications decreases to 2352: 912 for the computation of $c'$ and 1440 for the rest Actually, the "multiply" part can be somehow amortized as well: we refer to [21] for a proper description of such an optimized exponentiation strategy. The resulting computing load is less than what is needed for a couple of RSA decryptions with a similar modulus.

Unfortunately, there is a drawback in reducing the value of $k$: in the 30-prime variant it is necessary to store 1718 different $t[i, j]$ hash values. Hashing on two bytes seems enough and results in an overall memory requirement of four kilobytes. In the 16-prime variant, hash values of 3 bytes are necessary and the table size becomes $\cong$ 100 kilobytes. As observed at the end of section 2.2, the hash table can be drastically reduced at the cost of a minute computation overhead.

Another speed-up can be obtained by separately performing decryption modulo $p$ and $q$ so as to take advantage of smaller operand sizes. This alone, divides the decryption workfactor by four.

Finally, decryption is inherently parallel and naturally adapted to array processors since each $m_i$ can be computed independently of all the others.

### 2.6 Implementation

The new scheme (768-bit $n$, $k = 30$) was actually implemented on a 68HC05-based ST16CF54 smart-card (4,096 EEPROM bytes, 16,384 ROM bytes and 352 RAM bytes). The public key is only 96-byte long and as in most smart-card implementations, $n$'s storage is avoided by a command that re-computes the modulus from its factors upon request (re-computation and transmission take 10 ms). For further space optimization $g$'s first 91 bytes are the byte-reversed binary complement of $n$'s last 91 bytes. Decryption (a 4,119-byte routine) takes 3,912 ms. Benchmarks were done with a 5 MHz oscillator and ISO 7816-3 T=0 transmission at 115,200 bauds.

## 3 The probabilistic version

### 3.1 The setting

We now turn to the probabilistic version of the scheme. As already explained, we adopt a more complexity-oriented approach and, for example, we view $B$ as bounded by a polynomial in $\log n$. The probabilistic version replaces the ciphertext $g^m \bmod n$ by $c = x^\sigma g^m \bmod n$, where $x$ is chosen at random among positive integers $< n$. Decryption remains identical. This is due to the fact that the effect of multiplying by $x^\sigma$ is cancelled by raising the ciphertext to the various powers $\frac{\phi(n)}{p_i}$, as performed by the decryption algorithm. Note that this version requires $\sigma$ to be public.

The resulting scheme is *homomorphic*, which means that $E(m + m' \bmod \sigma) = E(m)E(m') \bmod n$. Probabilistic homomorphic encryption has received a lot of applications, both practically and theoretically oriented. To name a few, we quote the early work of Benaloh on election schemes ([4]) and the area of zero-knowledge proofs for NP (see [13, 3]). Known such schemes are the Quadratic Residuosity schemes of Goldwasser and Micali ([12]) which encrypts only one bit and its extensions to higher residues modulo a *single* prime (see [4]), which encrypts a few bits. As already explained in section 1, these schemes suffer from a serious drawback: a complexity theoretic analysis has to view the cleartext as logarithmic in the size of of ciphertext. In other words, the expansion rate, i.e. the ratio between the length of the ciphertext and the length of the cleartext is huge. In our proposal, this ratio is exactly $\frac{|n|}{|\sigma|}$. Note that that our assumption that $\sigma$ is $B$-smooth, for some small $B$, does not preclude a linear ratio. The maximum size of $\sigma$ is $\sum_{p<B} \log p$, where p ranges over primes and it is known that $\theta(B) = \sum_{p<B} \ln p \simeq B$. Thus, even if $B$ is logarithmic in $n$, there are enough primes to make $|\sigma|$ a linear proportion of $|n|$. This is a definite improvement over previous homomorphic schemes. Note however that, following the comments in section 2.4, it is safe to take $\frac{|\sigma|}{|n|} < 1/4$.

### 3.2 A complexity theoretic approach

We already observed that the security of our proposal is related to the question of distinguishing higher residues modulo $n$, that is integers of the form $x^p \bmod n$, when $p$ is a prime divisor of $\phi(n)$. In the rest of this section, we want to clarify this relationship in the asymptotic setting of complexity theory. In view of the remarks just made, we find it convenient to assume that the ratio $\frac{|\sigma|}{|n|}$ has a fixed value $\alpha < 1/4$. We also fix a polynomial $B$ in $\log n$. The parameters which are of interest to us are pairs $(n, \sigma)$ such that $\sigma$ is squarefree, odd and $B$-smooth, $n$ is a product of two primes $p$, $q$, $\sigma$ is a divisor of $\phi(n)$ prime to $\phi(n)/\sigma$ and $\frac{|\sigma|}{|n|} = \alpha$. We call any integer $n$ that appears as first coordinate of such a pair $(B, \alpha)$-dense. Distinguishing higher residues is usually considered difficult (see [4]). We conjecture that this remains true when $n$ varies over $(B, \alpha)$-dense integers. Towards a more precise statement, let $R_p(y, n)$ be one if $y$ is a $p$-th residue modulo $n$ and zero otherwise. Define a higher residue oracle to be a probabilistic polynomial time algorithm $A$ which takes as input a triple $(n, y, p)$ and returns a bit $A(n, y, p)$ such that the following holds:
*There exists a polynomial $Q$ in $|n|$ such that, for infinitely many values of $|n|$, one can find a prime $p(|n|) < B$, with:*

$$Pr\{A(n, y, p) = R_p(y, n)\} \geq 1 - \frac{1}{p} + \frac{1}{Q}$$

where the probability is taken over the random tosses of $A$ and its inputs, conditionnally to the event that $n$ is $(B, \alpha)$-dense and $p$ is a divisor of $\phi(n)$.

Our **Intractability Hypothesis** is that there is no higher residue oracle. The constant $1 - \frac{1}{p}$ comes from the obvious stategy for approximating $R_p$ which consists in constantly outputting zero. This strategy is successful for a proportion $1 - \frac{1}{p}$ of the inputs.

### 3.3 A security proof

The security of probabilistic encryption scheme has been investigated in [12]. In this paper, the authors introduced the notion of *semantic security*: given two messages $m_0$ and $m_1$, a message distinguisher is a probabilistic polynomial time algorithm $D$, which distinguishes encryptions of $m_0$ from encryptions of $m_1$. More, accurately, it outputs a bit $D(n, \sigma, g, y)$ in such a way that, setting

$$\theta_i = Pr\{D(n, \sigma, g, y) = 1 | y \in E(m_i)\}$$

where $E(m_i)$ is the set of encryptions of $m_i$, the following holds:
*There exists a polynomial $Q$ in $|n|$ such that, for infinitely many values of $|n|$, $|\theta_0 - \theta_1| \geq \frac{1}{Q}$*

Semantic security is the assertion that there is no pair of polynomial time algorithms $F$, $D$ such that $F$ produces two messages for which $D$ is a message distinguisher.

**Theorem 1** *Assume that no higher residue oracle exists. Then, the probabilitic version of the encryption scheme has semantic security.*

The proof of this result uses the *hybrid technique* for which we refer to [11]. It is technical in character and we have chosen to only include a sketch it in an appendix to the present paper.

## 4 Applications and variants

Even if we do not expect large scale replacement of RSA by our scheme, we feel that the latter is worth some academic interest. Especially, we believe that it opens up new applications. We have not yet fully investigated those potential applications but we give some suggestions below.

### 4.1 Traceability

Our proposal could offer some help in the management of key escrowing services. Consider the variant of the Diffie-Hellman key exchange protocol, where a composite modulus $n$ is used. Such a variant has been studied by various researchers including Mc Curley in [20], where it is shown that some specific choices lead to a scheme that is at least as difficult as factoring. Assume further that the modulus $n$ and the base for exponentiations $g$ are chosen as described in section 1. It has been proposed (see e.g [14]) that $g$ and $n$ could be defined by some kind of TTP (Trusted Third Party). Now, the user's public key $y$ and his secret key $x$ are related by $y = g^x \bmod n$. It is conceivable to leave the choice of $x$ to the user with the provision that $x \bmod \sigma = ID$, where $ID$ is the identity of the user. This can be checked by the TTP upon registration of the key. Thus, we have reached a situation where the identity is embedded in the public key through a trapdoor, although the actual key is not. One should not however overestimate the resulting functionality. It could be useful in scenarios where traceability is made possible via escrowing but where confidentiality cannot be broken even with the help of the escrowing services. Alternatively, it might be used to split traceability and secret key recovery between key escrows. Note that the above proposal requests that $\sigma$ is made public: as already observed, this does not seem to endanger the scheme.

### 4.2 Variants of the scheme

As is often the case, one can design numerous variants of the basic scheme. We will mention two because of their potential applications.

**Use of moduli with three prime factors** As for RSA, it is possible to embed three prime factors $p$, $q$, $r$ in the modulus in place of two. The construction is straightforward: the small odd primes $p_i$ are split into three groups thus yielding, by multiplication, three integers $u$, $v$, $w$. The three primes are then sought among integers of the form $2au + 1$ (resp. $2bv + 1$, resp. $2cw + 1$). It seems possible to keep the minimum size of $n$ to 768 bits, which allows $a$, $b$, $c$ to be around 200 bits. Following an idea of Maurer and Yacobi ([19]), we can then have a complete trapdoor for the discrete logarithm with base $g$: once the $\sigma$ part has been computed, there remains to compute the logarithm modulo $a$, $b$ and $c$, which is not immediate but well within the reach of current technology, since these numbers are 200 bit integers. Again, the variant could prove useful in key escrowing scenarios of, say, Diffie-Hellman keys, where it might be desirable to have a lengthy recovery of the secret key for consumer's protection.

**Multiplicative encryption** In this variant, $\sigma$ is made public and encryption applies to messages of length $k$, $m = \sum_{i=1}^{k} m_i 2^{i-1}$. In order to encrypt $m$, one computes $e = \prod_{i=1}^{k} p_i^{m_i}$ and apply probabilistic encryption to $e$. Of course, the bandwith of this variant is very low: using a 768 bit modulus $n$ and choosing the first 30 odd primes for $p_i$s, we obtain a 30 bit input and a 768 bit output. Allowing a larger input has drastic consequences in terms of the size of $n$. The value of $\sigma$ is close to $2^{560}$ when the first $k$ primes are used with $k = 80$ but reaches $2^{998.4}$ for $k = 128$ and $2^{1309}$ for $k = 160$. Using the heuristic bound mentioned in section 2.4, we get for the length of $n$ something beyond 5000 bits if $k$ is 160. This goes down to 2400 bits when $k = 80$.

As a result, the variant just decribed is not really practical and there is little chance that it can ever be adopted as an actual encryption scheme. On the other hand, the ciphertext $c(m)$ can be used in an encryption scheme à la El Gamal. The modulus is not prime since it is an RSA modulus, but it makes no difference on the user's size. From $h = c(m)$, he can manufacture a public key $y$ with a corresponding matching secret key $x$ of his choice $y = h^x \bmod n$ The resulting cryptosystem allows ciphertext traceability in the sense of Desmedt (see [9]). Our proposal enables to trace ciphertexts by a technique similar to the one used by Desmedt, but decreases the size of the modulus from something like 10000 bits to 2500 bits. The tracing algorithm goes as follows: extract from an El Gamal encryption the part $u = h^r \bmod n$ and apply the decryption algorithm, treating $u$ as a ciphertext. The decryption algorithm will basically find the original message $m$, which provides the identity of the user and from which $h$ was built. Several errors may occur due to the fact that $r$ might have some of the $p_i$s as divisors: the corresponding decrypted values of $m_i$ will be set to 1, regardless of their original values. The correct value can be found if a sample of ciphertexts are available or, alternatively, if an error-correction capacity has been added to $m$. Such an error-correction mechanism is highly advisable anyway in view of the attacks against software key escrow reported in [15].

Note that, one can further reduce the size of the exponent. This is because 40 bits may be considered enough for tracing purposes. The value of $\sigma$ goes down to approximately $2^{233}$ and 1088 bits becomes an acceptable minimum length for the modulus.

## 5 Challenge

It is a tradition in the cryptographic community to offer cash rewards for successful cryptanalysis. More than a simple motivation means, such rewards also express the designers' confidence in their own schemes. As an incentive to the analysis of the new scheme, we therefore offer \$ $|n|$ to whoever will decrypt :

$$c = \texttt{13370fe62d81fde356d1842fd7e5fc1ae5b9b449}$$
$$\texttt{bdd00866597e61af4fb0d939283b04d3bb73f91f}$$
$$\texttt{0d9d61eb0014690e567ab89aa8df4a9164cd4c6e}$$
$$\texttt{6df80806c7cdceda5cfda97bf7c42cc702512a49}$$
$$\texttt{dd196c8746c0e2ef36ca2aee21d4a36a}_{16}$$

$$g = \texttt{0b9cf6a789959ed4f36b701a5065154f7f4f1517}$$
$$\texttt{6d731b4897875d26a9e24415e111479050894ba7}$$
$$\texttt{c532ada1903c63a84ef7edc29c208a8ddd3fb5f7}$$
$$\texttt{d43727b730f20d8e12c17cd5cf9ab4358147cb62}$$
$$\texttt{a9fb8878bf15204e444ba6ade6132743}_{16}$$

$$n = \texttt{1459b9617b8a9df6bd54341307f1256dafa241bd}$$
$$\texttt{65b96ed14078e80dc6116001b83c5f88c7bbcb0b}$$
$$\texttt{db237daac2e76df5b415d089baa0fd078516e60e}$$
$$\texttt{2cdda7c26b858777604c5fbd19f0711bc75ce00a}$$
$$\texttt{5c37e2790b0d9d0ff9625c5ab9c7511d}_{16}$$

where $k = 30$ ($p_i$ is the $i$-th odd prime) and the message is ASCII-encoded. The challenger should be the first to decrypt at least 50% of $c$ and publish the cryptanalysis method but the authors are ready to carefully evaluate *ad valorem* any feedback they get.

## Acknowledgements

The paper grew out of a previous version which did not include the probabilistic case of our scheme. We wish to thank Julien Stern for suggesting us this alternative mode of encryption. We also want to thank J. Benaloh for help in clarifying our respective contributions in the definition of the probabilistic case. Finally, we are grateful to Adi Shamir, for helpful comments including the improved decryption algorithm mentioned in section 2.2 and also to one of the anonymous referees for pointing out the clever trick that yields the improved security analysis included at the end of section 2.4.


## References


[1] R. Anderson, *Robustness principles for public-key protocols*, Advances in Cryptology Crypto'95, Santa Barbara, Lectures Notes in Computer Science 963, pp. 236–247, Springer-Verlag, 1995.

[2] E. Brickell, D. Gordon, K. McCurley and D. Wilson, *Fast Exponentiation with Precomputation*, Advances in Cryptology Eurocrypt'92, Balatonfüred, Lectures Notes in Computer Science 658, pp. 200–207, Springer-Verlag, 1993.

[3] G. Brassard, D. Chaum and C. Crépeau, *Minimum Disclosure Proofs of Knowledge*, JCSS, Vol. 37(2), Oct. 1988, pp. 156–189.

[4] J. D. Cohen Benaloh, *Verifiable Secret-Ballot Elections*, Ph-D thesis, Yale University, 1988.

[5] J. D. Cohen and M. J. Fischer, (1985), *A robust and verifiable cryptographically secure election scheme*, Proc. of 26th Symp. on Foundation of Computer Science, 1985, 372–382.

[6] J. D. Cohen Benaloh, *Cryptographic Capsules: A Disjunctive Primitive for Interactive Protocols*, Advances in Cryptology Crypto'86, Santa Barbara, Lectures Notes in Computer Science , pp. 213–222, Springer-Verlag, 1986.

[7] J. D. Cohen Benaloh and M. Yung, *Distributing the Power of a Government to Enhance the Privacy of Voters*, Proc. of 5h Symp. on Principles of Distributed Computing, 1986, 52–62.

[8] D. Denning (Robling), *Cryptography and data security*, Addison-Wesley Publishing Company, pp. 148, 1983.

[9] Y. Desmedt, *Securing traceability of ciphertexts – Towards a secure software key escrow system*, Advances in Cryptology Eurocrypt'95, Saint-Malo, Lectures Notes in Computer Science 921, pp. 417–157, Springer-Verlag, 1995.

[10] W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, vol. IT-22-6, pp. 644–654, 1976.

[11] O. Goldreich, *Foundations of cryptography (Fragments of a book).* Weizmann Institut of Science, 1995.

[12] S. Goldwasser and S. Micali, *Probabilistic Encryption*, JCSS, 28(2), April 1984, pp. 270–299.

[13] O. Goldreich, S. Micali and A. Wigderson, *Proofs that Yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design*, Proc. of 27th Symp. on Foundation of Computer Science, 1986, pp.174–187.

[14] N. Jefferies, C. Mitchell and M. Walker, *A proposed architecture for trusted third party services*, Cryptography Policy and Algorithms, Queensland, Lecture Notes in Computer Science 1029, pp. 98–114, Springer-Verlag, 1996.

[15] L. Knudsen and T. Pedersen, *On the difficulty of software key escrow*, Advances in Cryptology Eurocrypt'96, Saragossa, Lectures Notes in Computer Science 1070, pp. 237–244, Springer-Verlag, 1996.

[16] P. Kocher, *Timing attacks in implementations of Diffie-Hellman, RSA, DSS and other systems*, Advances in Cryptology Crypto'96, Santa Barbara, Lectures Notes in Computer Science , pp. 104-113, Springer-Verlag, 1996.

[17] Kaoru Kurosawa, Yutaka Katayama, Wakaha Ogata and Shigeo Tsujii, *General public key residue cryptosystems and mental poker protocols*, Advances in Cryptology Eurocrypt'90, Aarhus, Lectures Notes in Computer Science 473, pp. 374–388, Springer-Verlag, 1996.

[18] H. Lenstra Jr., *Factoring integers with elliptic curves*, Annals of Mathematics, 126, pp. 649–673, 1991.

[19] U. Maurer and Y. Yacobi, *Non-interactive public key cryptography*, Advances in Cryptology Eurocrypt'91, Brighton, Lectures Notes in Computer Science 547, pp. 498–507, Springer-Verlag, 1991.

[20] K. McCurley, *A key distribution system equivalent to factoring*, Journal of Cryptology, vol. 1, pp. 85–105, 1988.

[21] D. M'Raïhi and D. Naccache, *Batch exponentiation - A fast DLP-based signature generation strategy*, Proceedings of the third ACM conference on Computer and Communications Security, New Delhi, pp. 58–61 , 1996.

[22] T. Okamoto and S. Uchiyama, *A new public-key cryptosystem as secure as factoring*, Advances in Cryptology Eurocrypt'98, Helsinki, Lectures Notes in Computer Science , pp. to appear, Springer-Verlag, 1998.

[23] D. Naccache and J. Stern, *A new public-key cryptosystem*, Advances in Cryptology Eurocrypt'97, Constance, Lectures Notes in Computer Science 1233, pp. 27–36, Springer-Verlag, 1997.

[24] Sung-Jun Park and Dong-Ho Won, *A generalization of public key residue cryptosystem*, In Proc. of 1993 KOREA-JAPAN joint workshop on information security and cryptology, 202–206.

[25] B. Pfitzmann and M. Schunter, *Asymmetric fingerprinting*, Advances in Cryptology Eurocrypt'96, Saragossa, Lectures Notes in Computer Science 1070, pp. 84–95, Springer-Verlag, 1996.

[26] D. Pointcheval, *A new identification scheme based on the perceptrons problem*, Advances in Cryptology Eurocrypt'94, Perugia, Lectures Notes in Computer Science 950, pp. 318–328, Springer-Verlag, 1995.

[27] S. C. Pohlig and M. E. Hellman, *An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance* IEEE Transactions on Information Theory, vol. IT-24-1, pp. 106–110, 1978.

[28] J. Pollard, *Theorems on factorization and primality testing*, Proceedings of the Cambridge Philosophical Society, vol. 76, pp. 521-528, 1974.

[29] J. Pollard, *Factoring with cubic integers*, A. Lenstra and H. Lenstra Jr., The development of the number field sieve, vol. 1554, LNM, 4–10, Springer-Verlag, 1993.

[30] C. Pomerance, *Analysis and comparison of some integer factoring algorithms*, printed in H. Lenstra Jr. and R. Tijdeman, Computational Methods in Number Theory I, Mathematisch Centum Tract 154, Amsterdam, pp. 89–139, 1982.

[31] M. Rabin, *Digitalized signatures and public-key functions as intractable as factorization*, MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[32] R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, vol. 21-2, pp. 120-126, 1978.

[33] A. Shamir, *An efficient identification scheme based on permuted kernels*, Advances in Cryptology Crypto'89, Santa Barbara, Lectures Notes in Computer Science 435, pp. 606–609, Springer-Verlag, 1990.

[34] A. Shamir, *RSA for paranoids*, CryptoBytes, vol. 1-3, pp. 1–4, 1995.

[35] J. Stern, *A new identification scheme based on syndrome decoding*, Advances in Cryptology Crypto'93, Santa Barbara, Lectures Notes in Computer Science 773, pp. 13–21, Springer-Verlag, 1994 .

[36] J. Stern, *Designing identification schemes with keys of short size*, Advances in Cryptology Crypto'94, Santa Barbara, Lectures Notes in Computer Science 839, pp. 164–173, Springer-Verlag, 1995.

## Appendix: Sketch of the Security Proof.

We show that any message distinguisher can be turned into an algorithm that recognizes higher residues. We let $D$ be a distinguisher for two messages $m_0$ and $m_1$ and start from the fact that, keeping the above notations, $\theta_0$ and $\theta_1$ are significantly distinct. We next use the *hybrid technique* for which we refer to [11], pp.91–93. Hybrids consist of a sequence of random variables $Y_i$, $0 \leq i \leq k$, such that

1. Extreme hybrids collide with $E(m_0)$ and $E(m_1)$ respectively.
2. Random values of each hybrid can be produced by a probabilistic polynomial time algorithm.
3. There are only polynomially many hybrids.

In such a situation, [11] shows that $D$ distinguishes two neighbouring hybrids. Our hybrids are formed by considering a message $\mu_i$, such that

$$\mu_i = m_0 \bmod p_j \quad \text{for} \quad j > i \;\; \text{and}$$

$$\mu_i = m_1 \bmod p_j \quad \text{for} \quad j \leq i$$

and letting $Y_i$ to be uniformly distributed over the set $E(\mu_i)$ of encryptions of $\mu_i$. It is easily seen that conditions 1, 2 and 3 are satisfied. Thus, for some index $i$, $D$ significantly distinguishes $Y_i$ and $Y_{i-1}$. Set $\mu = \mu_i$, $p = p_i$ and let $\mu^j$, $1 \leq j \leq p$, be the unique message such that

$$\mu^j = \mu \bmod p_\ell \quad \text{for} \quad \ell \neq i \quad \text{and} \quad \mu^j = j \bmod p$$

We note that, both $m_i$ and $m_{i-1}$ appear among the $\mu^j$s and we show that $D$ cannot distinguish encryptions of any two of the $\mu^j$s. This will yield the desired contradiction.

Let

$$\pi_j = Pr\{D(n, \sigma, g, y) = 1 | y \in E(\mu_j)\}$$

and assume that some $\pi_i$ significantly exceeds the other ones. In other words, $\pi_i \geq \sup_{j \neq i} \pi_j + \frac{1}{Q}$ for some polynomial $Q$ and infinitely many values of $|n|$. We show how to predict $p$-th residuosity: given $z$, we run $D$ over a large sample $N$ of inputs $(n, \sigma, y)$ where $y = x^\sigma z^{\ell\sigma/p} g^{\mu_i}$, with $x > n$ and $\ell \leq p$ chosen at random, and we average the outputs. Now, if $z$ is a $p$-th residue, then $y$ simply varies over $E(\mu_i)$, whereas, if $z$ is not a $p$-th residue, $y$ randomly varies over the union of all $E(\mu_j)$s. Thus, in the first case, the average is close to $\pi_i$, whereas, in the second case, it is approximately $\frac{\sum_{j=1}^p \pi_j}{p}$. It is easily seen that the difference is bounded from below by $\frac{p-1}{p}\frac{1}{Q}$. Using the law of large numbers, this is enough to make the proper decision on the $p$-th residuosity, with probability as close to 1 as we wish, by using only polynomially large samples. This finishes the proof.

**Remarks.**
1. Turning the previous sketch into a complete proof involves a technical but rather long write-up: especially, a precise version of the law of large numbers has to be made explicit, e.g. by using the Chebishev inequality. Also, the values of $\pi_i$ and $\frac{\sum_{j=1}^p \pi_j}{p}$ are not known *a priori* and should be approximated as well using the law of large numbers. We urge the interested reader to consult [11] for similar proofs.
2. The higher residuosity oracle that was built in the proof for the sake of contradiction uses inputs $\sigma$ and $g$ on top of $n$, $y$ and $p$. Actually, one can check that everything goes through, *mutatis mutandis*, if $\sigma$ is replaced by $\overline{\sigma} = \prod_{p<B} p$. Thus $\sigma$ is not really needed. As for $g$, as seen in section 2.1, it can be chosen at random: a proper choice will be spotted by sampling the corresponding oracle and checking its correctness.