Identification, Signature and Signcryption Using High Order Residues Modulo an RSA Composite

Yuliang Zheng

LINKS - Laboratory for Information and Network Security
Monash University, McMahons Road, Frankston, VIC 3199, Australia
yuliang.zheng@infotech.monash.edu.au
www.netcomp.monash.edu.au/links/

Abstract. Signcryption is a public key cryptographic primitive that fulfills the functions of digital signature and public key encryption concurrently, with a cost smaller than that required by the traditional signature followed by encryption method. The concept of signcryption, together with an implementation based on the discrete logarithm problem, was proposed in 1996. In this work, we demonstrate how to implement efficient signcryption using high order (power) residues modulo an RSA composite. This contributes to the research of extending computational underpinnings of signcryption schemes to problems related to integer factorization. In the course of achieving our goal, we also show efficient protocols for user identification, and fast and compact digital signature schemes.

Keywords: High Order Residues, Public Key Encryption, RSA, Signature, Signcryption.

1 Introduction

The idea of using (power) residues in public key cryptography first appeared in [5] where Goldwasser and Micali showed how to use quadratic residues in randomized encryption in a bit-by-bit fashion. This early work was followed by Benaloh and Yung's paper [3] where it was proposed to use r^{th} residues, where r was a *small* [1] *prime*, to construct a more efficient randomized public key encryption scheme. In [21] Zheng, Matsumoto and Imai proved that the requirement of r being a *small prime* could be relaxed to a small odd integer. This was further relaxed in [9] where Kurosawa and co-workers showed that r could take the form of a *small even* integer. In [2] (see also [14]), Benaloh observed that one could employ the Chinese Remainder Theorem in decryption, which further relaxed requirements of the number r — it could be a *large odd* integer, provided that it contains only *small and distinct* prime factors. More recently, Pailier [15]

[1] By "small" one generally means that the relevant parameter is bounded from above by a *poly-logarithmic* function of a security parameter. Likewise, a "large" parameter is one bounded from above by a *polynomial* function of a security parameter.

K. Kim (Ed.): PKC 2001, LNCS 1992, pp. 48–63, 2001.

discussed how to construct probabilistic public key encryption schemes involving n^{th} residues modulo n^2, where n is an RSA composite.

In all the public key encryption schemes presented in these successive papers, except those special schemes proposed in [15], decryption involves exhaustive search over a space whose size is dictated by a prime factor of r. This explains why these randomized encryption schemes do not work when r has a *large* prime factor.

While all the prior work on the use of r^{th} residues had been mainly limited to the construction of randomized public key encryption (requiring the number r contains only small prime factors), the present work demonstrates applications of r^{th} residues, with r being a *large* prime, in constructing efficient user identification, digital signature and more important, signcryption schemes. [2]

Signcryption was first proposed in [19] as an efficient public key cryptographic primitive that achieves both message confidentiality and non-repudiation with a much smaller cost than that required by digital signature followed by public key encryption. The first implementation of signcryption was based on the discrete logarithm problem over a finite field, which admitted a natural analogue on an elliptic curve over a finite field [20]. The same observation applies also to other sub-group based public key cryptosystems such as the XTR [12]. This early effort left as an interesting research topic to find a signcryption scheme that relies for its security on other computationally hard problems such as factoring large integers. Progress in this line of research has been made recently in [17]. Results presented in this paper represent yet another approach to building signcryption schemes on the integer factorization problem.

Section 2 provides background knowledge on high order (power) residues. Section 3 shows how to construct user identification protocols (also called passport protocols) that are based on high order residues modulo an RSA composite. This is followed by Section 4 where the identification protocols are converted into efficient digital signature schemes. The usefulness of high order residues is highlighted in Section 5 where a signcryption scheme called HORSE is presented. Efficiency of the identification, signature and signcryption schemes, both in terms of computation costs and expanded bits, is analyzed in Section 6. Finally the paper is closed with a summary in Section 7.

2 High Order (Power) Residues

The intension of this section is to summarize some of the core mathematical background that is required in understanding the identification, signature and signcryption schemes to be presented in this paper. Some useful further information on higher order residuosity can be found in [21]

[2] Identification and signature schemes proposed in [10,16] rely on properties of a subgroup related to an RSA modulus, and hence appear to be technically different from the present work. Furthermore, schemes in [10] work only when r is "small", as they require in their setting up stage the extraction of r^{th} roots and search over a space of r elements.

Let r and n be positive integers, and z be an integer relatively prime to n (i.e., $\gcd(z, n) = 1$). If there exists an integer x such that $z = x^r \bmod n$, z is said to be an r^{th} (power) residue modulo n. Otherwise z is said to be an r^{th} nonresidue modulo n.

The set of integers $[0, 1, \ldots, n-1]$ is denoted by $\mathbb{Z}_n$, and the set of integers in $\mathbb{Z}_n$ that are relatively prime to n is denoted by $\mathbb{Z}_n^*$.

We are interested in the case where r is a prime of at least 120 bits in size or length in binary representation, and n is an RSA modulus, i.e., $n = pq$, where both p and q are large (> 250 bits) primes. We further require that the three primes r, p and q be related by

$$\gcd(r, p-1) = r, \quad \gcd(r, q-1) = 1.$$

In practice, one may choose p and q in such a way that they take the form of

$$p = 2rp' + 1, \quad q = 2q' + 1$$

where both p' and q' are primes that are different from r.

For r and n of the above forms, an element $z \in \mathbb{Z}_n^*$ is necessarily an r^{th} residue modulo q, and it is an r^{th} residue modulo p if and only if $z^{(p-1)/r} = 1 \bmod p$. Thus z is an r^{th} nonresidue modulo n if and only if

$$z^{(p-1)/r} \neq 1 \bmod p.$$

As a consequence, when the factors p and q are known, one can quickly verify whether or not $z \in \mathbb{Z}_n^*$ is an r^{th} residue modulo n.

Note that $1/r$ of the elements in $z \in \mathbb{Z}_n^*$ are r^{th} residues modulo n, and the remaining $(r-1)/r$ of the elements are all r^{th} nonresidues modulo n. This makes easy the task of finding an r^{th} nonresidue modulo n.

Definition 1. *We say that three integers (r, n, h) are a* good triplet *if they fulfill the following requirements:*

1. *r is a prime whose size (length in binary representation) is at least 120 bits.*
2. *$n = pq$ is an RSA modulus of at least 512 bits, satisfying $\gcd(r, p-1) = r$ and $\gcd(r, q-1) = 1$.*
3. *h is an r^{th} nonresidue modulo n, or equivalently, $h^{(p-1)/r} \neq 1 \bmod p$.*

It should be pointed out that there is a slightly more general version of a good triplet (r, n, h) in which r is defined as an odd integer with distinct prime factors r_1, r_2, ..., r_t. One of the prime factors of r must be large, say of 120 bits in binary representation. n and r are related in the same way as in the above definition. The number h is an r_i^{th} nonresidue modulo p for every factor r_i, i.e., $h^{(p-1)/r_i} \neq 1 \bmod p$ for all $i = 1, 2, \ldots, t$. The identification, signature and signcryption schemes to be proposed in the forthcoming sections will all work with respect to such a more general version of a good triplet, although our discussions will be focused on the case where r is a large prime.

Fact 1 *For a good triplet (r, n, h), every element $x \in \mathbb{Z}_n^*$ can be presented as*

$$x = h^i \cdot w^r \bmod n$$

for a unique integer $i \in \mathbb{Z}_r$, where $\mathbb{Z}_n = [0, 1, \ldots, r-1]$ and a not necessarily unique $w \in \mathbb{Z}_n^*$. The number i is called the class-index of x.

Finding the class-index of x with respect to a good triplet appears to be infeasible, *even if one has the knowledge of the factors of n.* Currently known methods for solving the problem require the knowledge of p and q, and involve search over $\mathbb{Z}_r$. The average computation time required by such an algorithm is in the order of $r/2$, which renders the algorithm ineffective when r is a large prime. It should be pointed out that two of the classical methods for solving the discrete logarithm problem in a group, namely Shank's baby-step-giant-step method and Pollard's rho method (see [13]), do not appear to be applicable to the class-index problem under consideration, although both methods run faster than exhaustive search.

Another fact of importance is that the degree of difficulty in solving the class-index finding problem is not effected in any way by the choice of h, as the problem falls into a class of problems that share an interesting property called random self-reducibility (with respect to h). (See [1] for more discussions on random self-reducibility.) This fact will be used later in designing efficient schemes.

These observations form the computational basis of our new identification, digital signature and signcryption schemes to be presented in the coming sections.

3 Identification Using High Order Residues

3.1 Basic Protocol

At the setting up stage, a user Alice first chooses a good triplet (r, n, h). In addition, she also chooses at random x_a from $\mathbb{Z}_r = [0, 1, \ldots, r-1]$ and w_a from $\mathbb{Z}_n^*$. Namely $x_a \in_R \mathbb{Z}_r$ and $w_a \in_R \mathbb{Z}_n^*$, where $\in_R$ indicates an element is chosen uniformly at random from a set.

Alice then forms

$$y_a = \frac{1}{h^{x_a} \cdot w_a^r} \bmod n$$

She keeps x_a and w_a as her important private key, and publishes the triplet (r, n, h) and y_a as her public key.

Later when Alice wishes to prove to another user Bob that she is indeed Alice, she first forwards to Bob her (certified) public key. Bob verifies the authenticity of Alice's public key, and if he is satisfied with the verification, the two users then engage in the protocol specified in Table 1. At the end of an execution of the protocol, Bob would accept Alice's claim if and only if the verification at Step 4 is successful.

Table 1. Identification Using High Order Residues

Step	Alice	Channel	Bob
	Public key: r, n, h, y_a Private key: x_a, w_a		
1	Alice chooses $x \in_R \mathbb{Z}_r$ and $u \in_R \mathbb{Z}_n^*$. She then forms $y = h^x \cdot u^r \bmod n$ and sends y to Bob.	$\Rightarrow y \Rightarrow$	
2		$\Leftarrow b \Leftarrow$	Bob chooses $b \in_R \mathbb{Z}_r$ and forwards it to Alice as a challenge.
3	Alice forms $s = x + b \cdot x_a$ $v = u \cdot w_a^b \bmod n$ She then passes s and v over to Bob. Note that no modular operation is involved in the calculation of s.	$\Rightarrow s, v \Rightarrow$	
4			Bob verifies whether or not $h^s \cdot v^r \cdot y_a^b = y \bmod n$ Bob accepts Alice if and only if the equation passes the test.

3.2 Efficient Variants

A number of methods can be considered to improve the efficiency of the basic identification protocol.

A Small *h*. As was discussed earlier, the computational difficulty of the class-index finding problem is not dependent on how h, an r^{th} nonresidue modulo n, is chosen. Thus a small h may be selected so that it uses less memory and helps speed-up computations involving h. For a large r, an overwhelming majority $(\frac{r-1}{r})$ of elements in $\mathbb{Z}_n^*$ are r^{th} nonresidues. Hence the smallest r^{th} nonresidue

h can be easily identified by verifying whether

$$h^{(p-1)/r} \neq 1 \bmod p$$

for $h = 2, 3, 4, \ldots$, where p is a factor of n.

Shorter y and b. In Step 1, Alice may choose to send a hashed value of $h^x \cdot u^r \bmod n$ to Bob, that is

$$y = \mathcal{H}(h^x \cdot u^r \bmod n)$$

where $\mathcal{H}$ is a one-way hashing function. Accordingly, the verification by Bob in Step 4 should be modified to

$$\mathcal{H}(h^s \cdot v^r \cdot y_a^b \bmod n) = y$$

In Step 2, Bob may send Alice a shorter b, say of 60 bits, as a challenge. These improvements will reduce the bandwidth of messages exchanged between Alice and Bob.

Shorter w_a and u. As the generation of secure random bits may consume substantial computational resources, Alice may choose to generate w_a and u from a smaller range, say $\mathbb{Z}_{2^{80}} = [0, 1, \ldots, 2^{80} - 1]$.

Generating w_a and u deterministically. Alice may choose to generate w_a and u in the following way:

$$w_a = \mathcal{H}(x_a, r, n, h)$$
$$u = \mathcal{H}(x, r, n, h)$$

where $\mathcal{H}$ is a one-way hash function. This will completely eliminate the need of generating random bits for these two values.

Removing w_a and u altogether. A more efficient variant of the protocol is to fix w_a to 1, while choosing x_a from a range greater than $\mathbb{Z}_r$. More specifically, Alice can choose

$$x_a \in_R \mathbb{Z}_{2^\ell} = [0, 1, \ldots, 2^\ell - 1]$$

where ℓ may be at least 40 bits longer than the size of r. Namely $\ell \geq |r| + 40$, where $|\cdot|$ indicates the number of bits in the binary representation of an integer. Interestingly, with this modification, the number r will be used only in the setting up stage, but not in an identification process afterwards. Thus r no longer needs to be made public.

At the setting up stage, Alice first chooses a good triplet (r, n, h). She then chooses $x_a \in_R \mathbb{Z}_{2^\ell}$, and forms

$$y_a = \frac{1}{h^{x_a}} \bmod n$$

Alice then keeps x_a as her important private key, and publishes n, h and y_a as her public key. (Note that the use of the number r is now limited to the generation of h.)

When Alice wishes to prove to another user Bob that she is indeed Alice, she first forwards to Bob her (certified) public key. Bob verifies the authenticity of Alice's public key, and if he is satisfied with the verification, the two users then engage in the protocol described in Table 2.

Table 2. Identification Using High Order Residues — A More Efficient Version

Step	Alice	Channel	Bob
	Public key: n, h, y_a Private key: x_a		
1	Alice chooses $x \in_R \mathbb{Z}_{2^{1.75\ell}}$ and forms $y = \mathcal{H}(h^x \bmod n)$ She then sends y to Bob.	$\Rightarrow y \Rightarrow$	
2		$\Leftarrow b \Leftarrow$	Bob chooses $b \in_R \mathbb{Z}_{2^{\ell/2}}$ and forwards it to Alice as a challenge.
3	Alice forms $s = x + b \cdot x_a$ and sends it to Bob. Note that no modular operation is involved in the calculation of s.	$\Rightarrow s \Rightarrow$	
4			Bob verifies whether or not $\mathcal{H}(h^s \cdot y_a^b \bmod n) = y$ Bob accepts Alice if and only if the equation passes the test.

As $x_a \in \mathbb{Z}_{2^\ell}$ and $\ell \geq |r| + 40$, x_a can be expressed as $x_a = x'_a + f \cdot r$ for some $0 \leq x'_a < r$ and f. ¿From this it follows that

$$h^{x_a} \bmod n = h^{x'_a} \cdot (h^f)^r \bmod n.$$

Thus the efficient protocol in Table 2 can be viewed as one obtained from the protocol in Table 1 by letting $h^f \bmod n$ play the role of $w_a \in \mathbb{Z}_n^*$.

Note that the protocol in Table 2 has also incorporated other ideas discussed in this section, especially on shortening y and b. Also note that since the x chosen in Step 1 essentially plays the role of "masking" the secret key x_a in Step 3, it should be sufficiently long, say $|x| \geq |x_a| + |b| + 40$. Assuming that $|b| \approx \ell/2$ and $\ell \geq 160$, it would be adequate to have $|x| = 1.75\ell$.

4 Digital Signature Using High Order Residues

4.1 A General Signature Scheme

The identification protocol described in Table 1 can be converted to a digital signature scheme by substituting the role of Bob with an one-way hash function.

Alice sets up all the required parameters (including both public and private keys) in the same way as described in Section 3.1. Alice's public key is composed of y_a and a good triplet (r, n, h). Her private key is a pair of numbers x_a and w_a which are chosen, uniformly at random, from $\mathbb{Z}_r = [0, 1, \ldots, r-1]$ and $\mathbb{Z}_n^*$ respectively. The public and private keys are related by $y_a = \frac{1}{h^{x_a} \cdot w_a^r} \bmod n$.

To sign a message m, Alice first chooses at random x from $\mathbb{Z}_r$ and u from $\mathbb{Z}_n^*$, She then generates three numbers (b, j, v) as her signature on the message m as follows:

$$
\begin{aligned}
b &= \mathcal{H}(h^x \cdot u^r \bmod n, m) \\
s &= x + b \cdot x_a \\
v &= u \cdot w_a^b \bmod n
\end{aligned}
$$

Here $\mathcal{H}$ is an one-way hash function, and the calculation of s does not involve a modular operation.

Given m and (b, s, v), one uses Alice's public key to verify the authenticity of the signature by checking

$$\mathcal{H}(h^s \cdot v^r \cdot y_a^b \bmod n, m) = b$$

The signature is deemed authentic only if the equation holds.

We note that while our signature scheme bears some similarities to a scheme proposed in [7], there is an important technical difference. Namely, the scheme in [7] requires that $p-1$ and $q-1$, where p and q are the factors of an RSA composite n, share a large common divisor f which is needed in the generation of a signature.

4.2 A More Efficient Signature Scheme

Techniques for improving the efficiency of the identification protocol that have been discussed in Section 3.2 can be employed to make the signature scheme more efficient. To highlight the improvements, in the following we specify the

digital signature scheme that corresponds to the efficient identification protocol presented in Section 3.2.

As in Section 3.2, Alice's private key is $x_a \in_R \mathbb{Z}_{2^\ell}$, where $\ell \geq |r| + 40$, and her public key consists of three numbers n, h and y_a. The public and private keys are related by $y_a = \frac{1}{h^{x_a}} \bmod n$. There is no need to publish the number r, hence it can be erased at the completion of the setting up stage.

To sign a message m, Alice first chooses $x \in_R \mathbb{Z}_{2^{1.75\ell}}$. She then forms her signature on the message m, which is composed of two numbers (b, s), as follows:

$$b = \mathcal{H}(h^x \bmod n, m)$$
$$s = x + b \cdot x_a$$

The authenticity of Alice's signature can be confirmed by verifying

$$\mathcal{H}(h^s \cdot y_a^b \bmod n, m) = b$$

Table 3 summarizes the two signature schemes. Note that with the efficient version of the signature schemes, it is important that $|x|$ is sufficiently large, namely, $|x| \geq |x_a| + |b| + 40$. Once again assuming that $|b| \approx \ell/2$ and $\ell \geq 160$, it would suffice to have $|x| = 1.75\ell$.

Table 3. Signature Schemes Using High Order (Power) Residues

Signature scheme	Generation of signature	Verification of signature	Length of signature
General scheme Public key: r, n, h, y_a Private key: x_a, w_a	$m \to (m, b, s, v)$: $x \in_R \mathbb{Z}_r$ $u \in_R \mathbb{Z}_n^*$ $b = \mathcal{H}(h^x \cdot u^r \bmod n, m)$ $s = x + b \cdot x_a$ $v = u \cdot w_a^b \bmod n$	(m, b, s, v) : $y = h^s \cdot v^r \cdot y_a^b \bmod n$ accept only when $\mathcal{H}(y, m) = b$	$2\|\mathcal{H}(\cdot)\| + \|r\| + \|n\|$
Efficient scheme Public key: n, h, y_a Private key: x_a	$m \to (m, b, s)$: $x \in_R \mathbb{Z}_{2^{1.75\ell}}$ $b = \mathcal{H}(h^x \bmod n, m)$ $s = x + b \cdot x_a$	(m, b, s) : $y = h^s \cdot y_a^b \bmod n$ accept only when $\mathcal{H}(y, m) = b$	$\|\mathcal{H}(\cdot)\| + 1.75\ell$

5 HORSE — An Efficient Hight Order Residue Signcryption Engine

Using the technique for constructing signcryption schemes that was first developed in [19], the efficient signature scheme described in Table 3 can be used to design a new signcryption scheme whose security is related to the hardness of factoring large RSA moduli.

Like the signcryption schemes in [19], some of the parameters for HORSE are required to be shared among all users. The only difference with [19] is that with the present scheme, these shared parameters must be generated either by trusted authorities, possibly in a distributed manner, or by a "black-box" computer mimicking the function of trusted authorities.

To be more specific, the trusted authorities choose, on behalf of all users, a good triplet (r, n, h). The authorities may also choose an integer ℓ so that it is at least 40 bits longer than the size of r (in binary representation). Once (r, n, h) and ℓ are chosen, the authorities publish h and n, as well as ℓ. They then make the prime factors of n, i.e., p and q, and the number r inaccessible to users. Typically, this is done by erasing all the traces about p, q and r.

Alice must first set up her own pair of public and private keys y_a and x_a. This is done by

$$\begin{aligned} x_a &\in_R \mathbb{Z}_{2^\ell}, \\ y_a &= h^{x_a} \bmod n \end{aligned}$$

Alice publishes y_a in a public key directory, while keeping x_a as her matching private key.

Likewise Bob must also set up his pair of public and private keys y_b and x_b:

$$\begin{aligned} x_b &\in_R \mathbb{Z}_{2^\ell}, \\ y_b &= h^{x_b} \bmod n \end{aligned}$$

Table 4 summarizes the setting up of signcryption.

For Alice to signcrypt a message m to be sent to Bob, she carries out the signcryption operations detailed in Table 5. On receiving a signcrypted message from Alice, Bob can extract the original message by following the unsigncryption steps indicated in the same table. Note that in describing the signcryption scheme, it is assumed that $|KH.(\cdot)| \approx \ell/2$ and $\ell \geq 160$. This results in the choice of $|x| = 1.75\ell$, ensuring that $|x| \geq |x_a| + |b| + 40$.

To close this section, we point out that the way public and private keys are set up in the HORSE signcryption scheme also admits a system reminiscent to the ElGamal public key encryption scheme [4].

When a user Cathy wishes to send to Bob a message m in a secure way, she first chooses $x \in_R \mathbb{Z}_{2^\ell}$, and computes $k = \mathcal{H}(y_b^x \bmod n)$. Cathy then forms $c_1 = E_k(m)$, and $c_2 = h^x \bmod n$, and forwards to Bob the pair (c_1, c_2) as a ciphertext of m. Note that there is no need for Cathy to set up her public and private keys.

Table 4. Setting up for the Signcryption Scheme HORSE

Parameters public to all:
n — a large RSA modulus (chosen by trusted authorities)
h — an r^{th} nonresidue modulo n (chosen by trusted authorities)
ℓ — size of a secret key (may be chosen by trusted authorities)
$\mathcal{H}$ — a one-way hash function with $\|\mathcal{H}(\cdot)\| \geq 128$
KH — a keyed one-way hash function with $\|KH.(\cdot)\| \geq 80$
(E, D) — the encryption and decryption algorithms of a private key cipher
Alice's keys:
x_a — private key ($x_a \in_R \mathbb{Z}_{2^\ell}$)
y_a — public key ($y_a = h^{x_a} \bmod n$)
Bob's keys:
x_b — private key ($x_b \in_R \mathbb{Z}_{2^\ell}$)
y_b — public key ($y_b = h^{x_b} \bmod n$)

On receiving (c_1, c_2) from Cathy, Bob can recover k by involving his private key x_b in the computation of $k = c_2^{x_b} \bmod n$. He can then proceed to extract the original message m from c_1 by $m = D_k(c_1)$.

6 Efficiency of the Schemes

We examine the efficiency of the identification, signature and signcryption schemes in terms of computational efforts invested and communication overhead required. With a protocol or algorithm employing public key cryptography, the dominant computation is modular exponentiations involving large integers. When computing the product of several modular exponentiations, we can use a very effective technique that was discussed in Knuth's book (see Exercise 27, Pages 465 and 637 of [8]; see also [18]). The same technique was later re-discovered by Shamir (see the last part of [4]).

6.1 Efficiency of Identification

We focus on the more efficient protocol specified in Table 2. Messages communicated between Alice and Bob are very compact: $|\mathcal{H}(\cdot)| + 2\ell$ bits from Alice to Bob and ℓ bits from Bob to Alice.

Alice needs to perform one modular exponentiation which can be pre-computed well before the start of the protocol. Using the classical "square-and-multiply" method, on average the exponentiation takes $1.5 \cdot 1.75\ell = 2.625\ell$ modular multiplications.

Bob needs to compute the product of two modular exponentiations. The size (or length) of the longer exponent s has 1.75ℓ bits. Using the fast method discussed in Knuth's book, Bob can complete, once again on average, the computation in $(1 + 3/4)|s| = 1.75^2\ell \approx 3\ell$ modular multiplications.

Table 5. The Signcryption Scheme HORSE

Signcryption by Alice the Sender:
$m \rightarrow (c, d, e)$

1. Pick $x \in_R \mathbb{Z}_{2^{1.75\ell}}$, and let $k = \mathcal{H}(y_b^x \bmod n)$.
2. Split k into k_1 and k_2 of appropriate size.
3. $c = E_{k_1}(m)$.
4. $d = KH_{k_2}(m, \textit{bind\_info})$,
   where *bind_info* may contain, among other data, the public key (certificate) of the recipient, and optionally the public key (certificate) of the sender.
5. $e = x + d \cdot x_a$.
6. Send to Bob the signcrypted text (c, d, e).

Unsigncryption by Bob the Recipient:
$(c, d, e) \rightarrow m$

1. Recover k from d, e, h, n, y_a and x_b:
   $k = \mathcal{H}((h^e \cdot (\frac{1}{y_a})^d)^{x_b} \bmod n)$.
2. Split k into k_1 and k_2.
3. $m = D_{k_1}(c)$.
4. Accept m as a valid message originated from Alice only if $KH_{k_2}(m, \textit{bind\_info})$ is identical to d.

6.2 Efficiency of Signature

With the fast signature scheme (the second scheme) in Table 3, its signature is significantly shorter than the RSA signature scheme. More specifically, the size of our signature is $|\mathcal{H}(\cdot)| + 1.75\ell$ bits. Assuming that $|\mathcal{H}(\cdot)| = 80$ and $\ell = 200$, the signature has only 430 bits.

The signing procedure requires one exponentiation, or $1.5 \cdot 1.75\ell = 2.625\ell$ modular multiplications on average. This is much faster than the generation of an RSA signature which involves a full length exponent.

The verification of a signature will take more time than the RSA signature scheme with a small public key, as it requires the computation of the product of two exponentiations, with s being the longer exponent. On average, the product takes $(1 + 3/4)|s| = 1.75^2\ell \approx 3\ell$ modular multiplications.

6.3 Efficiency of Signcryption

The communication overhead, measured in bits, of the signcryption scheme HORSE specified in Table 5, is

$$|d| + |e| = |KH.(\cdot)| + 1.75\ell$$

Recall that the communication overhead of the traditional RSA signature followed by RSA encryption is

$$|n_a| + |n_b|$$

where n_a is Alice's RSA modulus and n_b Bob's. Clearly HORSE represents an significant improvement over RSA.

The computational cost for signcryption is

$$1.5 \cdot 1.75\ell = 2.625\ell$$

modular multiplications on average. The unsigncryption operation involves the computation of the product of two exponentiations. The two exponents are $e \cdot x_b$ and $d \cdot x_b$. It is important to note that as $\phi(n)$, the Euler's ϕ-function, is not known to Bob, the size of the exponents cannot be reduced! Clearly, the longer exponent is $e \cdot x_b$ which has 2.75ℓ bits. Thus on average unsigncryption takes

$$(1 + 3/4) \cdot 2.75\ell \approx 4.8\ell$$

modular multiplications.

Together, signcryption and unsigncryption take

$$7.4\ell$$

modular multiplications.

T compare HORSE with RSA signature followed by RSA encryption, we assume that $|n| = |n_a| = |n_b|$, that the size of r and the size of an output of the key-ed hash function KH are related by $|r| = 1.5|KH.(\cdot)|$, and that $\ell = |r| + 0.5|KH.(\cdot)| = 2|KH.(\cdot)|$.

We further assume that the Chinese Remainder Theorem is used in RSA decryption and signature generation, achieving the theoretically maximum speedup. Namely we assume that the average computational cost for RSA signature generation is $\frac{1.5}{4}|n_a| = 0.375|n_a|$ modular multiplications, and for RSA decryption it is $\frac{1.5}{4}|n_b| = 0.375|n_b|$ modular multiplications. With RSA encryption and signature verification, to simplify our discussions we consider two cases, although there are numerous other possible combinations for one to choose from in practice. These two cases are: (1) small public exponents (say of 10 bits or less), and (2) $\ell/2$-bit public exponents,

In order to examine how signcryption outperforms the signature-then-encryption approach, we define the advantages of signcryption as $(1 - C_{sc}/C_{s+e})$, where C_{sc} indicates the cost of signcryption, while C_{s+e} the cost of signature-then-encryption. More specifically, we have

$$\begin{aligned}
&\text{advantage in} \quad \text{average computational cost} \\
&\qquad = \begin{cases} 1 - \frac{7.4\ell}{0.375(|n_a|+|n_b|)}, & \text{for small public exponents} \\ 1 - \frac{7.4\ell}{0.375(|n_a|+|n_b|)+1.5\ell}, & \text{for } \ell/2\text{-bit public exponents} \end{cases} \\
&\text{advantage in} \quad \text{communication overhead} \\
&\qquad = 1 - \frac{|KH.(\cdot)| + 1.75\ell}{|n_a| + |n_b|}
\end{aligned}$$

Table 6 demonstrates the advantages with respect to various key sizes. While the selection of parameter sizes in Table 6 is admittedly somewhat arbitrary, we note that it is still more conservative than a table suggested in [11].

Table 6. Advantage of Signcryption Scheme HORSE over RSA based Signature-Then-Encryption with *Small Public Exponents*

security parameters				advantage in average computational cost		advantage in communication overhead
$\|n\|$ ($\|n_a\|, \|n_b\|$)	ℓ	$\|KH.(\cdot)\|$	$[\|r\|]$	small public exponent	$\ell/2$-bit public exponent	
1024	160	80	[120]	-54.1%	-17.5%	82.4%
1280	176	88	[132]	-35.6%	-6.4%	84.5%
1536	176	88	[132]	-13.0%	8.0%	87.1%
1792	192	96	[144]	-5.7%	13.0%	88.0%
2048	192	96	[144]	7.6%	22.1%	89.5%
2560	208	104	[156]	19.8%	31.0%	90.9%
3072	224	112	[168]	28.1%	37.2%	91.8%
4096	256	128	[192]	38.3%	45.2%	93.0%
5120	288	144	[216]	44.5%	50.1%	93.7%
8192	320	160	[240]	61.5%	64.3%	95.6%
10240	320	160	[240]	69.2%	71.0%	96.5%

In some applications, one may wish to choose RSA public exponents that are longer than $\ell/2$, or even of full size, while in some other applications the Chinese Remainder Theorem may not be used in RSA decryption or signature generation. Furthermore, one may choose to select key sizes by following the suggestions in [11]. In all these situations, the signcryption scheme HORSE will demonstrate even greater savings in computation time and communication overhead.

7 Concluding Remarks

We have demonstrated applications of r^{th} power residues modulo an RSA composite in constructing efficient identification protocols, digital signature and signcryption schemes. A major difference between this work and prior research is that here r is a *large* prime, or more generally an odd integer containing a *large* prime factor. Efficiency of our schemes is analyzed and compared with some existing solutions. Of particular interest to a practitioner in public key cryptography is the fact that the signcryption scheme HORSE is significantly more advantageous over the traditional "signature followed by encryption" approach using the RSA signature and encryption schemes, both in terms of computational and communication overhead. A formal analysis of the security of the protocols and schemes presented in this paper remains a challenging topic for future research.

To close this paper, we summarize in Table 7 the main variants of signcryption known currently.

Table 7. Currently Known Variants of Signcryption

	Computational Foundation	Reference
1	discrete logarithm on a finite field	Zheng, CRYPTO'97 [19]
2	discrete logarithm on an elliptic curve	Zheng, CRYPTO'97 [19], Zheng & Imai, IPL (1998) [20]
3	factoring / residuosity	Steinfeld & Zheng, ISW2000 [17], Zheng, PKC'01
4	other sub-groups (e.g., XTR)	Gong & Harn, IEEE-IT (2000) [6], Lenstra & Verheul, CRYPTO2000 [12], Zheng, CRYPTO'97 [19]

Acknowledgment

Thanks to Ron Steinfeld for various discussions on selecting security parameters for the schemes. Thanks also to PKC2001 Program Committee members for their helpful comments.

References

1. M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. *Journal of Computer and System Sciences*, 39:21–50, 1989.
2. J. Benaloh. Dense probabilistic encryption. In *Workshop on Selected Areas in Cryptography (SAC'94)*, pages 120–128, Ontario, Canada, 1994. Queen's University.
3. J. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of voters. In *Proceedings of the 5-th ACM Symposium on Principles of Distributed Computing*, pages 52–62, 1986.
4. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.
5. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
6. G. Gong and L. Harn. Public-key cryptosystems based on cubic finite field extensions. *IEEE Transactions on Information Theory*, 45(7):2601–2605, 2000.
7. S. J. Kim, S. J. Park, and D. H. Won. Convertible group signatures. In *Advances in Cryptology - ASIACRYPT'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 311–321, Berlin, New York, Tokyo, 1996. Springer-Verlag.
8. D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, 2 edition, 1981.
9. K. Kurosawa, Y. Katayama, W. Ogata, and S. Tsujii. General public key residue cryptosystems and mental poker protocols. In *Advances in Cryptology - EUROCRYPT'90*, volume 473 of *Lecture Notes in Computer Science*, pages 374–388, Berlin, New York, Tokyo, 1990. Springer-Verlag.

10. B. Lee, S. Kim, and D. Won. ID-based multisignature scheme based on the high residuosity problem. In *Proceedings of 1997 Joint Workshop on Information Security and Cryptography (JW-ISC'97)*, pages 227–230, Seoul, Korea, 1997. KIISC (Korea).
11. A. Lenstra and E. Verheul. Selecting cryptographic key sizes. In *Public Key Cryptography — The Third International Workshop on Practice and Theory in Public Key Cryptography (PKC2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 446–465, Berlin, New York, Tokyo, 2000. Springer-Verlag.
12. A. Lenstra and E. Verheul. The XTR public key system. In *Advances in Cryptology - CRYPTO2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 1–19, Berlin, New York, Tokyo, 2000. Springer-Verlag.
13. A. K. Lenstra and H. W. Lenstra. *Algorithms in Number Theory*, volume A of *Handbook in Theoretical Computer Science*, chapter 12, pages 673–715. Elsevier and the MIT Press, 1990.
14. D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 59–66. ACM Press, 1998.
15. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 399–416, Berlin, New York, Tokyo, 1999. Springer-Verlag.
16. D. Pointcheval. The composite discrete logarithm and secure authentication. In *Public Key Cryptography — The Third International Workshop on Practice and Theory in Public Key Cryptography (PKC2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 113–128, Berlin, New York, Tokyo, 2000. Springer-Verlag.
17. R. Steinfeld and Y. Zheng. A signcryption scheme based on integer factorization. In *Information Security — Proceedings of 2000 Information Security Workshop (ISW2000)*, Lecture Notes in Computer Science, Berlin, New York, Tokyo, 2000. Springer-Verlag. (to appear).
18. S.-M. Yen, C.-S. Laih, and A. K. Lenstra. Multi-exponentiation. *IEE Proceedings - Computers and Digital Techniques*, 141(6):325–326, 1994.
19. Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In *Advances in Cryptology - CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179, Berlin, New York, Tokyo, 1997. Springer-Verlag.
20. Y. Zheng and H. Imai. How to construct efficient signcryption schemes on elliptic curves. *Information Processing Letters*, 68:227–233, 1998.
21. Y. Zheng, T. Matsumoto, and H. Imai. Residuosity problem and its applications to cryptography. *Transactions of IEICE*, E71(8):759–767, August 1988.