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

Yuliang Zheng

LINKS – Laboratory for Information and Network Security

Monash University

Email: yuliang.zheng@infotech.monash.edu.au

URL: http://www.netcomp.monash.edu.au/links/

Two important security requirements

- **Authenticity**
  - **Non-repudiation, and**
  - **Unforgeability**
- **Confidentiality**

The Signcryption approach

- Achieves the functions of
  - digital signature
    - unforgeability & non-repudiation
  - encryption
    - confidentiality
- has a significantly smaller comp. & comm. cost

Cost (signcryption) << Cost (signature) + Cost (encryption)

Applications of Signcryption

- Enabler
  - Secure mobile computing & commerce using small devices (PDAs, phones, smart cards)
- Cost saver (both time & bandwidth)
  - "drop-in" replacement of traditional "signature followed by encryption"
  - Secure high speed ATM networks
- Firewall authentication, …

Known variants of signcryption

- based on DL on finite field
  - Zheng, CRYPTO'97
- based on DL on an Elliptic Curve
  - Zheng, CRYPTO'97
  - Zheng & Imai IPL 1998
- based on other sub-groups (e.g. XTR)
  - Lenstra & Verheul, CRYPTO2000
  - Gong & Harn, IEEE-IT 2000
  - Zheng, CRYPTO'97
- based on factoring ?????
  - this paper (PKC2001)

Outline

- High order (power) residuosity problem modulo an RSA integer
- Identification using high order residues
- Signature using high order residues
- Signcryption using high order residues
  - HORSE – High Order Residue based Signcryption Engine
- Performance analysis
- Conclusion

Math notations

- $Z_n = \{0,1,2,\ldots,n-1\}$
- $Z_n^* = \{ i \mid i \in Z_n, \text{ and } \gcd(i, n) = 1\}$

Power residues

- Given 3 integers (r, n, z)
  - z is relatively prime to n, i.e., $\gcd(z,n)=1$
- z is called an
  - r^{th} (power) residue mod n if there exists an integer x s.t. $z = x^r \bmod n$
  - r^{th} nonresidue mod n otherwise.

A good triplet (r, n, h)

- r --- a prime of $\geq$ 120 bits in binary representation
- n --- $n = p\,q$ (an RSA modulus)
  - $p = 2\,r\,p' + 1$, $\gcd(r, p') = 1$
  - $q = 2\,q' + 1$, $\gcd(r, q') = 1$
- h --- an r^{th} nonresidue mod n. I.e.,

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

A useful fact

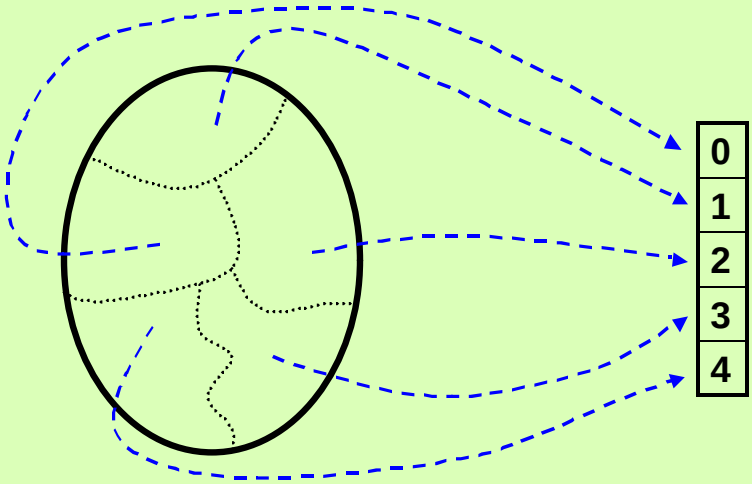
- Given a good triplet (r, n, h), every $x \in Z_n^*$ can be written as

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

for a unique $i = 0, 1, \ldots, r-1$, and a NOT necessarily unique $w \in Z_n^*$.

- the number i is called the call-index of x.

A nice many-to-one mapping



Z_n^*

Z_r={ 0,1,..,r-1}

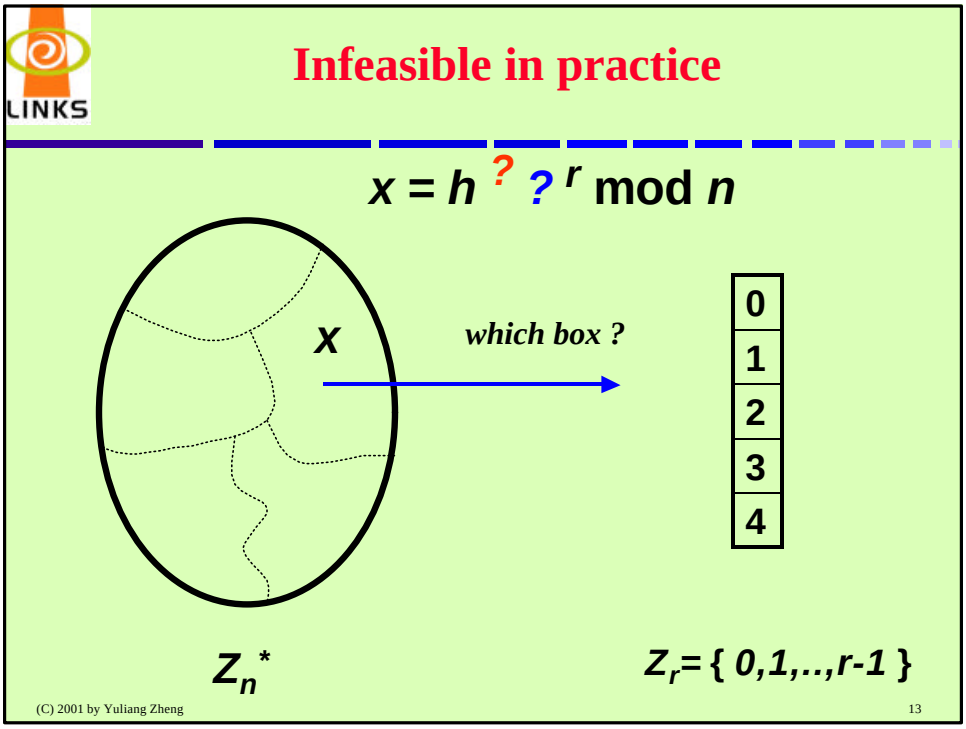
Computational infeasibility of finding the class-index

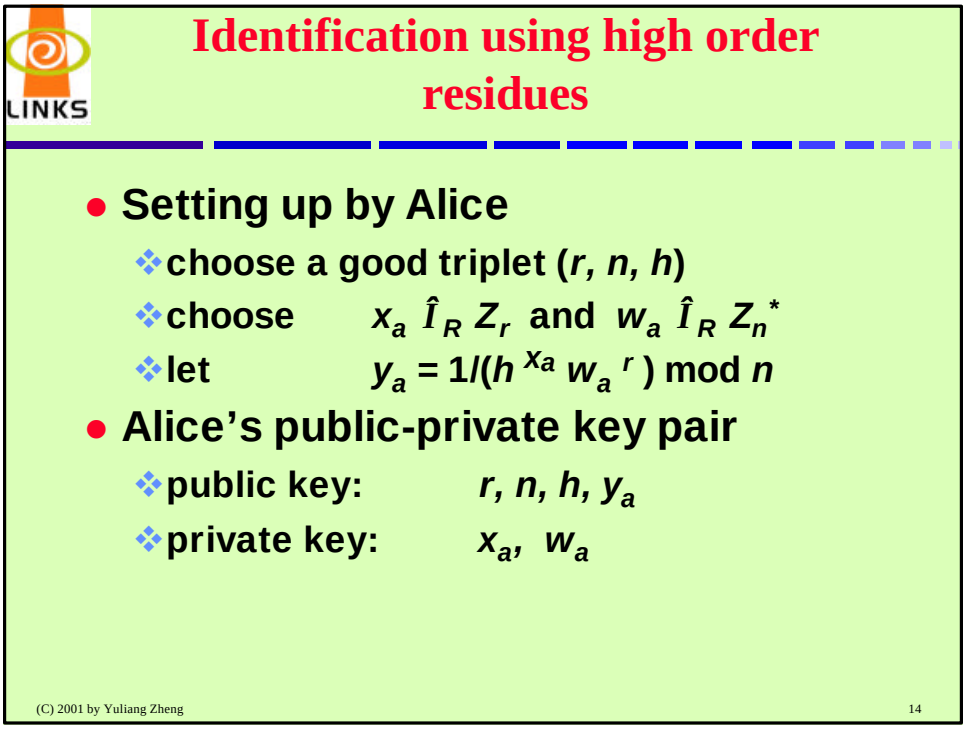
- Given a good triplet (r, n, h), and $x \in Z_n^*$, finding out the class-index i of x is believed to be hard:

(r, n, h) and $x \in Z_n^*$

⇓ Infeasible in practice !

the number i
s.t. $x = h^i w^r \bmod n$


Infeasible in practice
$x = h^{?} ?^{r} \bmod n$
x
which box ?
0
1
2
3
4
Z_n^*
$Z_r = \{ 0,1,..,r-1 \}$
(C) 2001 by Yuliang Zheng
13


Identification using high order residues
Setting up by Alice
choose a good triplet (r, n, h)
choose $x_a \in_R Z_r$ and $w_a \in_R Z_n^*$
let $y_a = 1/(h^{x_a} w_a^{r}) \bmod n$
Alice's public-private key pair
public key: r, n, h, y_a
private key: x_a, w_a
(C) 2001 by Yuliang Zheng
14

Alice identifies herself to Bob

Alice

Bob

$x \in_R Z_r, \quad w \in_R Z_n^*$

$y = h^x w^r \bmod n$

y →

$b \in_R Z_r$

(b acts as a challenge.)

← b

$s = x + b\, x_a$

$v = u\ w_a^{\,b} \bmod n$

s, v →

Accepts Alice only if

$h^s\ v^r\ y_a^{\,b} = y \bmod n$

Techniques for improving efficiency

- choose a small nonresidue h
- use shorter y and b
  - $y = H(h^x u^r) \bmod n$
  - choose a 60-bit random string as b
- choose w_a and u from a smaller range
- generating w_a and u from Hash
  - $w_a = H(x_a, r, n, h)$
  - $u = H(x, r, n, h)$
- Removing w_a and u, but
  - choose x_a from a larger range
  - choose x from a LARGER range

A more efficient way for identification --- setting up

- **Setting up by Alice**
  - **choose a good triplet (*r*, *n*, *h*) and** $\ell \geq |r| + 40$
  - **choose** $x_a \in_R Z_{2^\ell}$
  - **let** $y_a = 1/h^{x_a} \bmod n$
- **Alice's public-private key pair**
  - **public key:** ***n*, *h*, y_a**
  - **private key:** x_a

A more efficient way for identification --- the protocol

Alice		Bob
$x \in_R Z_{2^{1.75\ell}}$ $y = H(h^x \bmod n)$	y →	
		$b \in_R Z_{2^{\ell/2}}$ (b acts as a challenge.)
	← b	
$s = x + b\,x_a$	s, v →	
		Accepts Alice only if $H(h^s y_a^{\,b} \bmod n) = y$

Communication & computation costs of fast identification

- **Communication cost --- small !**
  - **Alice to Bob:** $|H(\bullet)|+2\ell$ bits
    - **e.g. 80+2X160=400 bits**
  - **Bob to Alice:** ℓ bits
- **Computational cost --- small !**
  - **Alice: 1 exponentiation**

    $1.5\times 1.75\ell = 2.625\ \ell$ multiplications
  - **Bob: 1.17 exponentiations**

    $(1+3/4)|s|=1.75^2\ell \approx 3\ \ell$ multiplications

Identification → Signature

- **The basic idea is to let a one-way hash function play the role of Bob, in choosing a random challenge *b***

$b \in_R Z_r$
(b acts as a challenge.)

$b = H(y, m)$
(b acts as a challenge.)

Signature --- a general scheme

Alice's public key:	Alice's signature on *m*:	Signature verification:
r, n, h, y_a	$x \in_R Z_r$, $w \in Z_n^*$	$y = h^s v^r y_a^{\,b} \bmod n$
Alice's private key:		accept only if
x_a, w_a	$b = H(h^x u^r \bmod n, m)$	$H(y, m) = b$
where	$s = x + b\, x_a$	
$x_a \in_R Z_r$, $w_a \in Z_n^*$	$v = u\, w_a^{\,b} \bmod n$	
$y_a = h^{x_a} \bmod n$	$Sig_a(m) = (b, s, v)$	

Length of signature: $2|H(\bullet)| + |r| + |n|$

Signature --- an efficient scheme

Alice's public key:	Alice's signature on *m*:	Signature verification:
n, h, y_a	$x \in_R Z_{2^{1.75\ell}}$	$y = h^s\, y_a^{\,b} \bmod n$
Alice's private key:		accept only if
x_a	$b = H(h^x \bmod n, m)$	$H(y, m) = b$
where	$s = x + b\, x_a$	
$x_a \in_R Z_{2^\ell}$,		
$\ell \geq \lvert r \rvert + 40$	$Sig_a(m) = (b, s)$	
$y_a = h^{x_a} \bmod n$		

Length of signature: $|H(\bullet)| + 1.75\ell$

Communication & computation costs of fast signature

- **Length of signature --- short !**

  $|H(\bullet)|+1.75\ell$ bits

  - **e.g. 80+1.75X160=360 bits**
- **Computational cost --- small !**
  - **signing: 1 exponentiation**

    $1.5\times 1.75\ell = 2.625\ \ell$ multiplications
  - **verifying: 1.17 exponentiations**

    $(1+3/4)|s|=1.75^2\ell \approx 3\ \ell$ multiplications

HORSE (High Order Residue based Signcryption Engine) --- setting up

- **Parameters public to all (generated by TTP)**
  - ***n,h*** **--- part of a good triplet (*r,n,h*)**
  - ℓ **--- security parameter with** $\ell \geq |r|+40$
  - **H --- a 1-way hash**
  - **KH --- a keyed hash**
  - **(E,D) --- a secret key cipher**
- **Users' keys**

Alice's public key: y_a	Bob's public key: y_b
Alice's private key: x_a	Bob's private key: x_b
where $x_a \in_R Z_{2^\ell}$,	where $x_b \in_R Z_{2^\ell}$,
$y_a = h^{x_a} \bmod n$	$y_b = h^{x_b} \bmod n$

HORSE (High Order Residue based Signcryption Engine) --- how it works

• **Alice:** $m \longrightarrow (c,r,s)$	• **Bob:** $(c,d,e) \longrightarrow m$
1. pick $x \in_R Z_{2^{1.75\ell}}$, and let $k = H(y_b^{\,x} \bmod n)$	1. Recover k: $k = H(h^{e x_b} (\frac{1}{y_a})^{d x_b} \bmod n)$
2. k_1 = left-half (k) k_2 = right-half (k)	2. k_1 = left-half (k) k_2 = right-half (k)
3. $c = E_{k_1}(m)$	3. $m = D_{k_1}(c)$
4. $d = KH_{k_2}(m, y_b)$	4. accept m as being valid only if $KH_{k_2}(m, y_b) = d$
5. $e = x + d\, x_a$	
6. send to Bob (c, d, e)	

Communication & computation costs of HORSE

- **Length of signature --- short !**
  $|KH(\bullet)| + 1.75\ell$ bits
  - **e.g. 80+1.75X160=360 bits**
- **Computational cost --- small !**
  - **signcryption: 1 exponentiation**
    $1.5 \times 1.75\ell = 2.625\ell$ multiplications
  - **unsigncryption: 1.17 exponentiations**
    $(1+3/4)2.75\ell \approx 4.8\ell$ multiplications
  - **together:**
    7.4ℓ multiplications

Advantages of HORSE over RSA sign-then-encrypt (small public exponent)

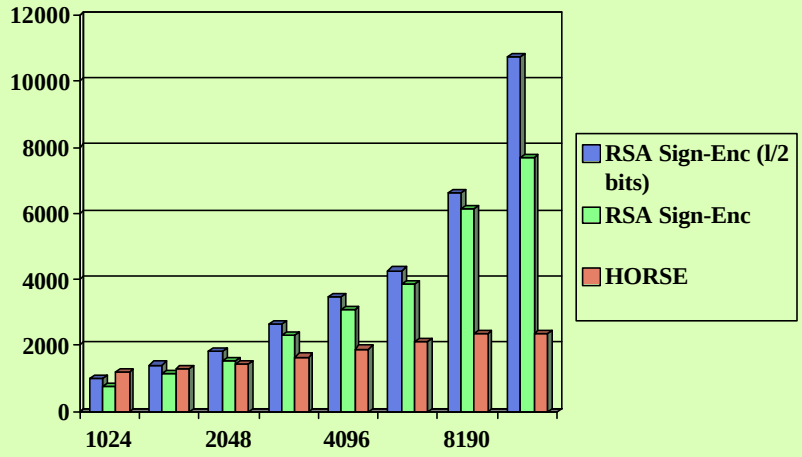
|n|	ℓ	|KH|	saving in comp cost	saving in comm overhead
1024	160	80	-54.1 %	82.4 %
1536	176	88	-35.6 %	84.5 %
2048	192	96	-5.7 %	88.0 %
3072	224	112	28.1 %	89.5 %
4096	256	128	38.3 %	93.0 %
5120	288	144	44.5 %	93.7 %
8192	320	160	61.5 %	95.6 %
10240	320	160	69.2 %	96.5 %

Advantages of HORSE over RSA sign-then-encrypt (ℓ/2 bit public exponent)

|n|	ℓ	|KH|	saving in comp cost	saving in comm overhead
1024	160	80	-17.5 %	82.4 %
1536	176	88	8.0 %	84.5 %
2048	192	96	22.1 %	88.0 %
3072	224	112	37.2 %	89.5 %
4096	256	128	45.2 %	93.0 %
5120	288	144	50.1 %	93.7 %
8192	320	160	64.3 %	95.6 %
10240	320	160	71.0 %	96.5 %

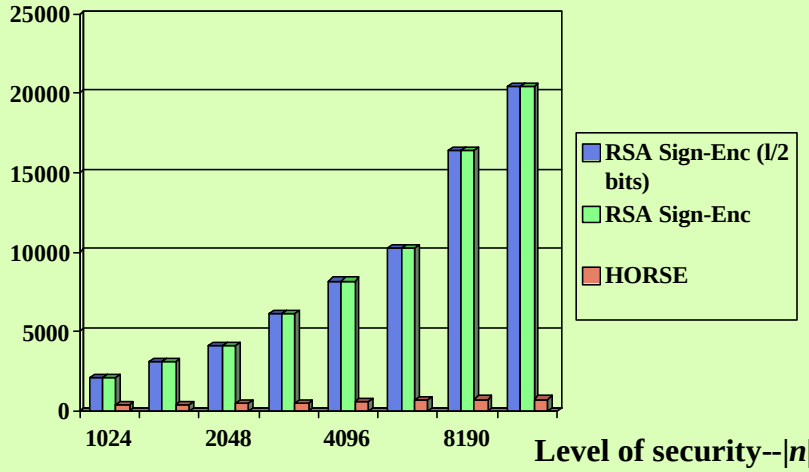
Advantages of HORSE over RSA sign-then-encrypt --- Comp. time ---

Time -- # of multiplications



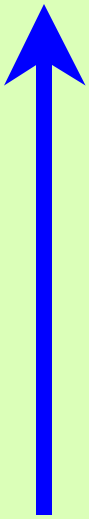
Advantages of HORSE over RSA sign-then-encrypt --- Comm. overhead ---

Comm. overhead -- # of bits



Major variants of signcryption

- based on DL on an Elliptic Curve
  - Zheng, CRYPTO'97
  - Zheng & Imai IPL 1998
- based on other sub-groups (e.g. XTR)
  - Lenstra & Verheul, CRYPTO2000
  - Gong & Harn, IEEE-IT 2000
  - Zheng, CRYPTO'97
- based on DL on finite field
  - Zheng, CRYPTO'97
- based on factoring / residuosity
  - Steinfeld & Zheng, ISW2000
  - Zheng, PKC2001



Finally

Q & A