

Signcryption

How to Achieve
Cost(signature & encryption) <<
Cost(signature) + Cost(encryption)

Yuliang Zheng

Calyptix Security Corporation, and
UNC Charlotte
www.signcryption.net

Applications of Signcryption

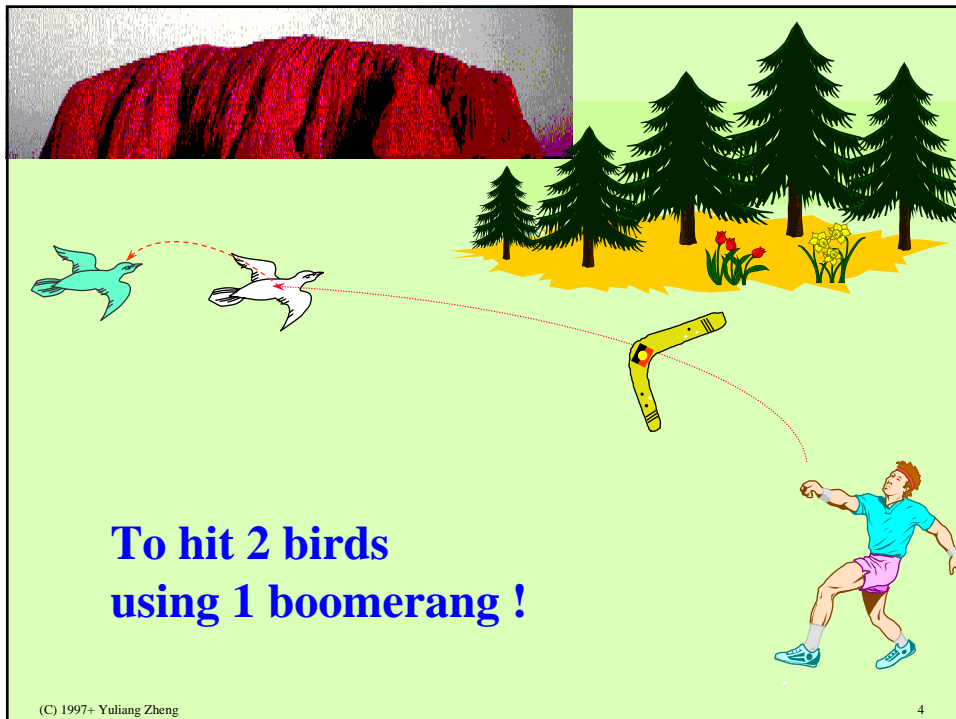
- Bring to society huge savings in comp. & comm. if used widely in
 - ❖ secure & authenticated message delivery / storage
 - ❖ electronic commerce
 - **secure & authenticated transactions !!!**
 - ❖ secure & authenticated multicast (incl. video conference, CSCW etc)
 - ❖ fast, compact, secure, unforgeable & non-repudiated key transport
 - ❖

Goals

- To provide with
 - ❖ both confidentiality,
 - ❖ and authenticity
 - unforgeability &
 - non-repudiation
- but in an efficient way !

(C) 1997+ Yuliang Zheng

3



(C) 1997+ Yuliang Zheng

4

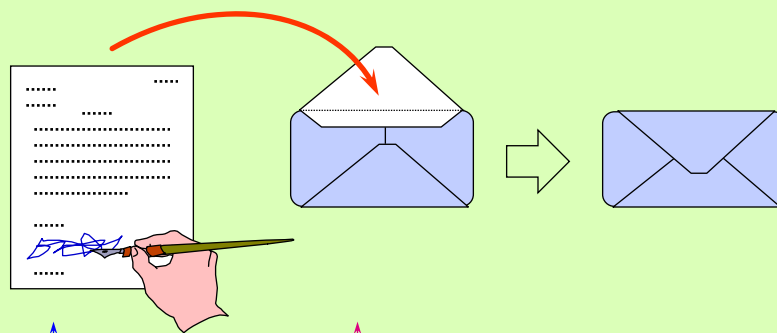
Outline

- problems with sign-then-encrypt
- signcryption -- a new paradigm
- cost-savings of signcryption over sign-then-encrypt
- properties of signcryption
 - ❖ confidentiality, unforgeability and non-repudiation
- signcryption for multiple recipients
- applications

(C) 1997+ Yuliang Zheng

5

In the paper & ink world: Signature-then-Seal

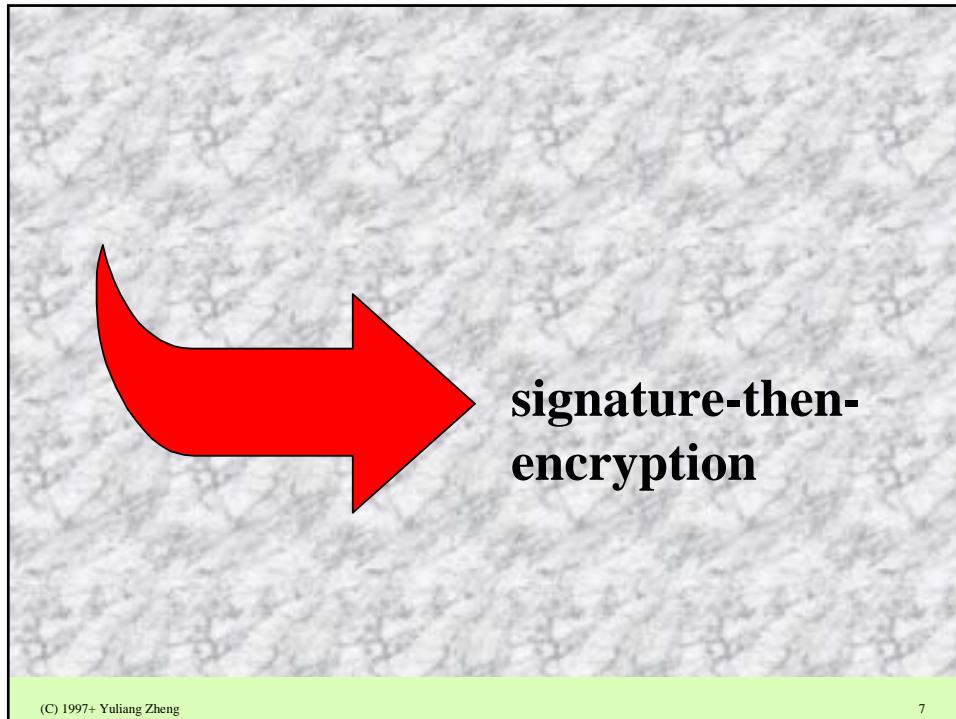


To achieve:
authenticity
(*unforgeability &*
non-repudiation)

To achieve:
confidentiality

(C) 1997+ Yuliang Zheng

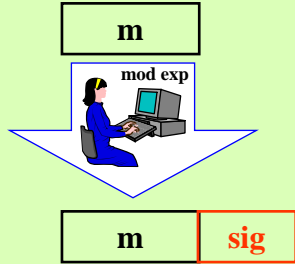
6




In the digital world (Alice to Bob): Signature-then-Encryption


- 1. Signature generation
 - ❖ Alice signs a message m using her secret key, i.e. creating sig on m .
- 2. Encryption
 - ❖ Alice encrypts (m, sig) using DES with k .
 - ❖ Alice creates another data so that Bob can recover k . (Typically, Alice encrypts k using Bob's public key).

m



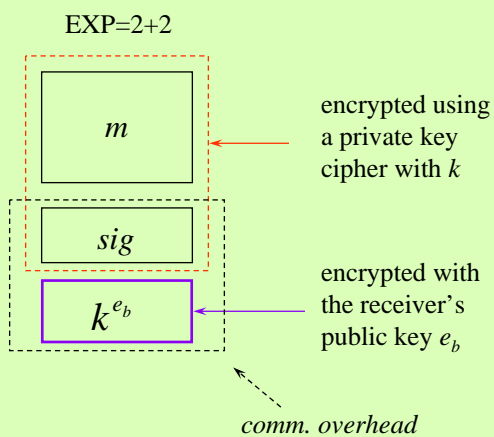
m | sig





(C) 1997+ Yuliang Zheng 8

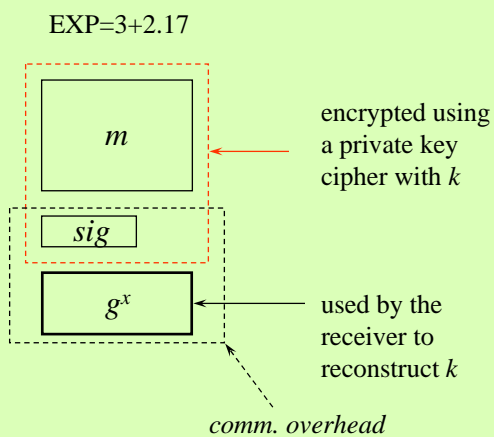
Signature-then-Encryption (based on RSA)



(C) 1997+ Yuliang Zheng

9

Signature-then-Encryption (based on Discrete Logarithm or DL)



(C) 1997+ Yuliang Zheng

10

Cost of Signature-then-Encryption

Cost Schemes	Comp Cost (No. of exp)	Comm Overhead (bits)
RSA based sig-then-enc	2 + 2	$ n_a + n_b $
DL based Schnorr sig + EIGamal enc	3 + 2.17 (3 + 3)	$ hash + q + p $

where *hash* is a 1-way hash function.

(C) 1997+ Yuliang Zheng

11

Why signature-then-encryption can be a problem

- Consider a transaction/message of 5,120 bits (=640 chars, \approx 8 lines) that requires
 - ❖ high level security, or
 - ❖ to be transmitted in 2010
- Very large moduli, say of 5120 bits, have to be used

(C) 1997+ Yuliang Zheng

12

Why signature-then-encryption can be a problem (cnt'd)

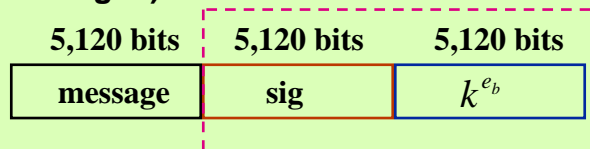
- If RSA with a 5120-bit composite is used

❖ Comp. cost:

2+2=4 exponentiations mod a (very large !)
5120-bit integer

❖ Comm. overhead:

10,240 bits (twice as large as the original
message !)



(C) 1997+ Yuliang Zheng

13

Why signature-then-encryption can be a problem (cnt'd)

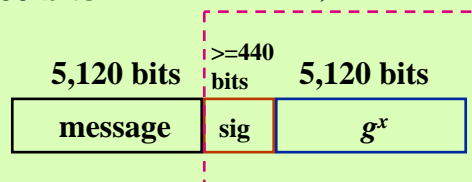
- If Schnorr sig & ElGamal enc with a 5120-bit prime are used

❖ Comp. cost:

3+2.17=5.17 (3+3=6) exponentiations
mod a (very large !)

❖ Comm. overhead:

≥ 5560 bits



(C) 1997+ Yuliang Zheng

14



Signcryption -- a new approach

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

$$\text{Cost (signcryption)} \ll \text{Cost (signature)} + \text{Cost (encryption)}$$

Signcryption -- public & secret parameters

- **Public to all**

- ❖ p : a large prime
- ❖ q : a large prime factor of $p-1$
- ❖ g : $0 < g < p$ & with order $q \bmod p$
- ❖ *hash*: 1-way hash
- ❖ **KH**: key-ed 1-way hash
- ❖ **(E,D)** : private-key encryption & decryption algorithms

- **Alice's keys**

- ❖ x_a : secret key
- ❖ y_a : public key
(note :
 $y_a = g^{x_a} \bmod p$)

- **Bob's keys**

- ❖ x_b : secret key
- ❖ y_b : public key
(note :
 $y_b = g^{x_b} \bmod p$)

Key-ed 1-way hash: examples

- **efficient, but security properties are less understood**

$$KH_k(x) \equiv hash(k, x)$$

where *hash* is a 1-way hash.

Key-ed 1-way hash: examples (cnt'd)

- (slightly) less efficient, but some properties can be proven

- ❖ **NMAC:**

- $KH_{k_1, k_2}(x) \equiv F_{k_1}(F_{k_2}(x))$

- where $F_k(x)$ is the same as *hash*, except that IV used by *hash* is now replaced by k .

- ❖ **HMAC:**

- $KH_k(x) \equiv \text{hash}(k' \oplus \text{opad}, \text{hash}(k' \oplus \text{ipad}, x))$

- where k' is a 0-padded version of k ,
 $\text{opad} = \text{x36...36}$, $\text{ipad} = \text{x5c...5c}$

Signcryption -- 1st example

$$m \longrightarrow (c, r, s)$$

$$(c, r, s) \longrightarrow m$$

- **Signcrypt by Alice**

- ❖ $k = \text{hash}(y_b^x \text{ mod } p)$
where $x \in_R \{1, K, q-1\}$

- ❖ $k \begin{matrix} \longrightarrow & k_1 \\ & \longrightarrow & k_2 \end{matrix}$

- ❖ $r = KH_{k_2}(m)$

- ❖ $s = \frac{x}{r + x_a} \text{ mod } q$

- ❖ $c = E_{k_1}(m)$

- ❖ **output** (c, r, s)

- **Unsigncrypt by Bob**

- ❖ $k = \text{hash}((y_a \cdot g^r)^{s \cdot x_b} \text{ mod } p)$

- ❖ $k \begin{matrix} \longrightarrow & k_1 \\ & \longrightarrow & k_2 \end{matrix}$

- ❖ $m = D_{k_1}(c)$

- ❖ **output**

- $$\begin{cases} m & \text{if } r = KH_{k_2}(m) \\ \text{"invalid"} & \text{if } r \neq KH_{k_2}(m) \end{cases}$$

Signcryption -- 1st example (focusing on unsigncryption)

Let

$$u = s \cdot x_b \pmod{q},$$

$$v = r \cdot u \pmod{q},$$

Then,

$$\begin{aligned} & (y_a \cdot g^r)^{s \cdot x_b} \pmod{p} \\ &= (y_a^{s \cdot x_b} \cdot g^{r \cdot s \cdot x_b}) \pmod{p} \\ &= (y_a^u \cdot g^v) \pmod{p} \end{aligned}$$

This can be done in a smart way, costing only 1.17 exponentiations on average !

D. Knuth,
Seminumerical Algorithms,
Vol. 2 of *The Art of Computer Programming*,
2nd edition, Addison-Wesley,
Exercise 27, Pages 465 & 637.

Signcryption -- 2nd example

$$m \longrightarrow (c, r, s)$$

$$(c, r, s) \longrightarrow m$$

• Signcrypt by Alice

- ❖ $k = \text{hash}(y_b^x \pmod{p})$
where $x \in_R \{1, K, q-1\}$

- ❖ $k \begin{cases} \longrightarrow k_1 \\ \longrightarrow k_2 \end{cases}$

- ❖ $r = KH_{k_2}(m)$

- ❖ $s = \frac{x}{1 + x_a \cdot r} \pmod{q}$

$$c = E_{k_1}(m)$$

- ❖ **output** (c, r, s)

• Unsigncrypt by Bob

- ❖ $k = \text{hash}((g \cdot y_a^r)^{s \cdot x_b} \pmod{p})$

- ❖ $k \begin{cases} \longrightarrow k_1 \\ \longrightarrow k_2 \end{cases}$

- ❖ $m = D_{k_1}(c)$

- ❖ **output**

$$\begin{cases} m & \text{if } r = KH_{k_2}(m) \\ \text{"invalid"} & \text{if } r \neq KH_{k_2}(m) \end{cases}$$

Signcryption -- 3rd example

$m \longrightarrow (c,r,s)$

$(c,r,s) \longrightarrow m$

• Signcrypt by Alice

❖ $k = \text{hash}(y_b^x \bmod p)$
 where $x \in_R \{1, K, q-1\}$

❖ $k \begin{matrix} \longrightarrow & k_1 \\ & \searrow \\ & k_2 \end{matrix}$

❖ $r = KH_{k_2}(m)$

❖ $s = (x - r \cdot x_a) \bmod q$

$c = E_{k_1}(m)$

❖ **output** (c,r,s)

• Unsigncrypt by Bob

❖ $k = \text{hash}((g^s \cdot y_a^r)^{x_b} \bmod p)$

❖ $k \begin{matrix} \longrightarrow & k_1 \\ & \searrow \\ & k_2 \end{matrix}$

❖ $m = D_{k_1}(c)$

❖ **output**

$\begin{cases} m & \text{if } r = KH_{k_2}(m) \\ \text{"invalid"} & \text{if } r \neq KH_{k_2}(m) \end{cases}$

Cost of Signcryption

Alice $m \longrightarrow (c,r,s)$

Bob $(c,r,s) \longrightarrow m$

• Comp. cost

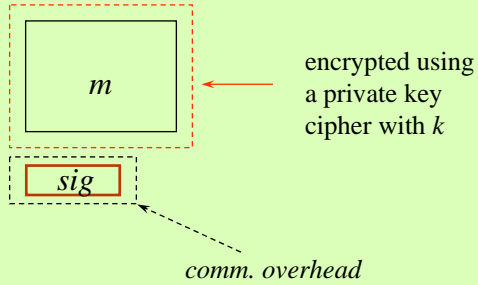
- ❖ by Alice :
1 exponentiation modulo p
- ❖ by Bob :
1.17 exponentiations modulo p (using Shamir's technique)
- ❖ total=2.17 exp mod p

• Comm. overhead

- ❖ $|r| + |s|$ bits
- (note: $|m| = |c|$)

Comp. & Comm. Cost of Signcryption (based on Discrete Logarithm)

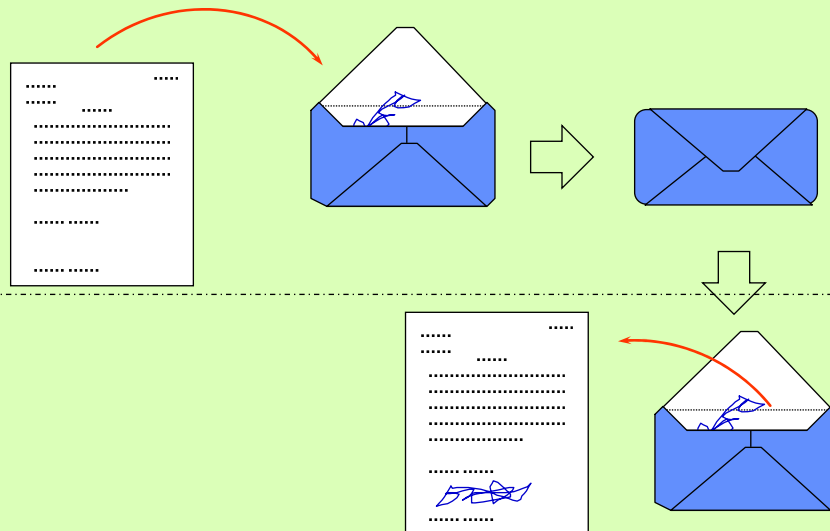
$$EXP=1+1.17$$



(C) 1997+ Yuliang Zheng

25

“Magic” Signcryption Envelope



(C) 1997+ Yuliang Zheng

26

**signature-then-encryption
v.s.
signcryption**

(C) 1997+ Yuliang Zheng 29

Signcryption v.s. Signature-then-Encryption

EXP=1+1.17	EXP=2+2	EXP=3+2.17
(a) Signcryption based on DL	(b) Signature-then-Encryption based on RSA	(c) Signature-then-Encryption based on DL

(C) 1997+ Yuliang Zheng 30

Cost of Signature-then-Encryption v.s. Cost of Signcryption

A simplistic comparison:

Cost Schemes	Comp Cost (No. of exp)	Comm Overhead (bits)
RSA based sig-then-enc	2 + 2	$ n_a + n_b $
DL based Schnorr sig + ElGamal enc	3 + 2.17 (3 + 3)	$ \text{hash} + q + p $
DL based Signcryption	1 + 1.17 (1 + 2)	$ KH + q $

(C) 1997+ Yuliang Zheng

31

A more detailed comparison

- Why do this ?

- ❖ the computing time of $y^x \bmod z$ largely depends on the size of x
- ❖ the sizes of RSA & DL exponents are different
- ❖ DL exponent --- $[1, \dots, q]$
- ❖ RSA public exponent e --- can be small
- ❖ RSA secret exponent d --- $|n|$ bits, BBIIGG !

(C) 1997+ Yuliang Zheng

32

Signcryption v.s. Schnorr Sig + ElGamal Enc

- Saving in comp. cost by signcryption

$$= \frac{(5.17 - 2.17) \text{ modulo exp}}{5.17 \text{ modulo exp}} = 58\%$$

- Saving in comm. overhead by signcryption (assuming $|\text{hash}|=|KH|$)

$$= \frac{|p|}{|KH(\cdot)|+|q|+|p|}$$

Signcryption v.s. Schnorr Sig + ElGamal Enc (cnt'd)

$ p $	$ q $	$ KH $	saving in comp cost	saving in comm overhead
512	144	72	58 %	70.3 %
768	152	80	58 %	76.8 %
1024	160	80	58 %	81.0 %
1536	176	88	58 %	85.3 %
2048	192	96	58 %	87.7 %
3072	224	112	58 %	90.1 %
4096	256	128	58 %	91.0 %
5120	288	144	58 %	92.0 %
8192	320	160	58 %	94.0 %
10240	320	160	58 %	96.0 %

Signcryption v.s. RSA

- Adv. in comp. cost by signcryption

$$= \frac{0.375(|n_a|+|n_b|) - 3.25|q|}{0.375(|n_a|+|n_b|)}$$

(Assuming small RSA public exponents & CRT are used !)

- Adv. in comm. overhead by signcryption

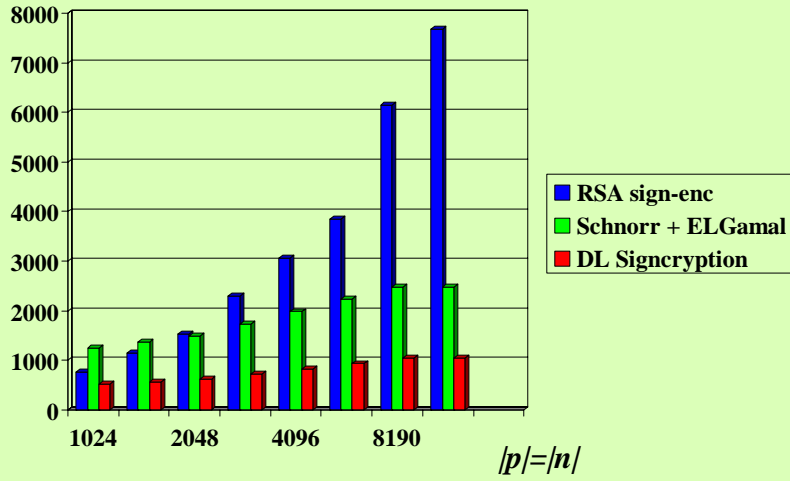
$$= \frac{|n_a|+|n_b| - [KH(\cdot) + |q|]}{|n_a|+|n_b|}$$

Signcryption v.s. RSA

$ p = n_a $ $= n_b $	$ q $	$ KH $	saving in comp cost	saving in comm overhead
512	144	72	0 %	78.9 %
768	152	80	14.2 %	84.9 %
1024	160	80	32.3 %	88.3 %
1536	176	88	50.3 %	91.4 %
2048	192	96	59.4 %	93.0 %
3072	224	112	68.4 %	94.0 %
4096	256	128	72.9 %	95.0 %
5120	288	144	75.6 %	96.0 %
8192	320	160	83.1 %	97.0 %
10240	320	160	86.5 %	98.0 %

DL Signcryption v.s. sign-then-encrypt

of multiplications



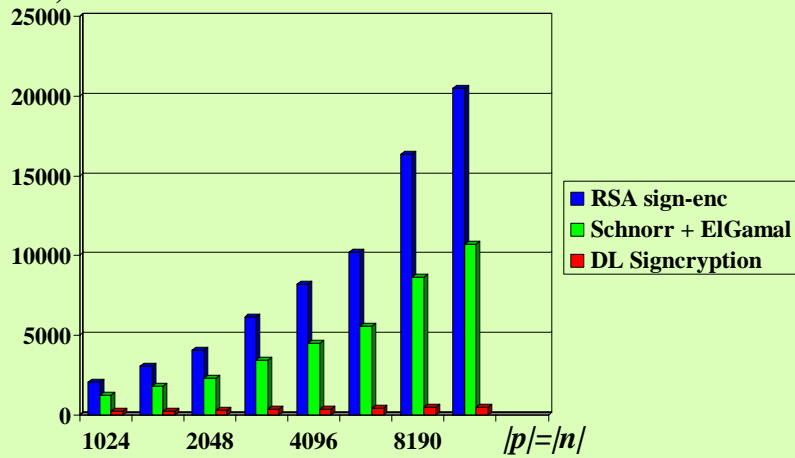
(C) 1997+ Yuliang Zheng

37

DL Signcryption v.s. sign-then-encrypt

comm. overhead

(# of bits)

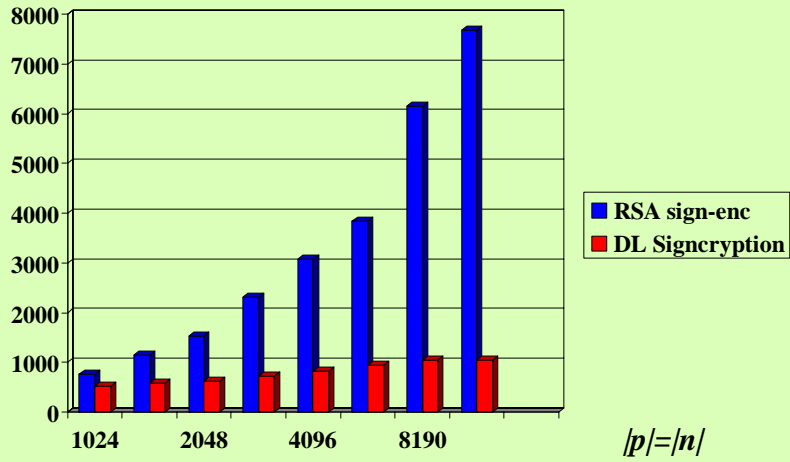


(C) 1997+ Yuliang Zheng

38

DL Signcryption v.s. RSA sign-encrypt

of multiplications

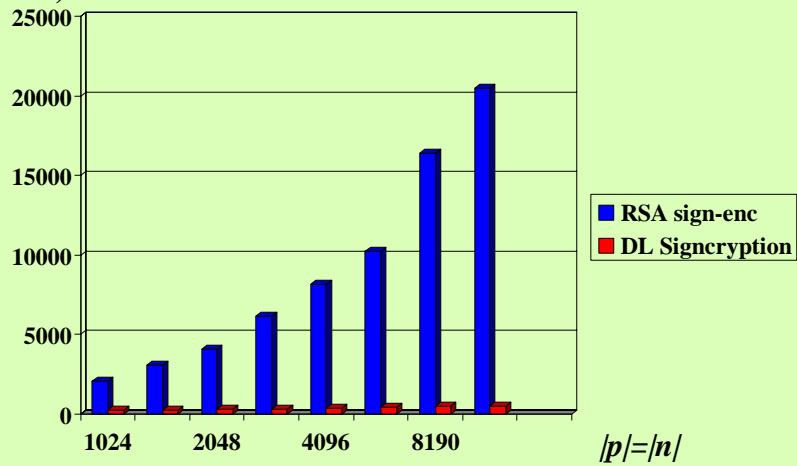


(C) 1997+ Yuliang Zheng

39

DL Signcryption v.s. RSA sign-encrypt

comm. overhead
(# of bits)

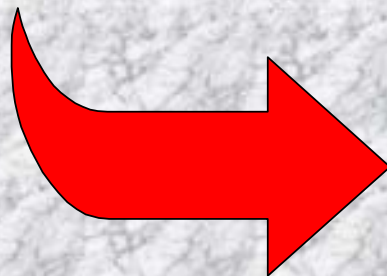


(C) 1997+ Yuliang Zheng

40

Why Can Signcryption Save Cost ?

- For Bob to successfully recover a message, he needs to know $g^x \bmod p$ from which he can compute
$$k = \text{hash}(g^{x \cdot x_b} \bmod p)$$
- With signcryption, Bob can derive $g^x \bmod p$ from r, s, g, p and y_a . That is, there is NO need for Alice to explicitly send Bob $g^x \bmod p$



**a general method
for implementing
signcryption**

Signcryption can be based on any shortened ElGamal-like signatures

How to shorten ElGamal-like signatures:

• Calculation of r

❖ $r = \text{hash}(k, m)$

where

$$k = g^x \text{ mod } p$$

and x is a random number.

• Calculation of s

❖ if $\text{hash}(m)$ is in the original s :

■ $\text{hash}(m) \rightarrow 1$
OR

■ $r \rightarrow 1, \text{hash}(m) \rightarrow r$

❖ otherwise if $\text{hash}(m)$ is not in the original s , go to next step.

❖ if $s = (\dots)/x$, then change it to $s = x/(\dots)$.

Shortened DSS (for Alice)

$m \longrightarrow (m, r, s)$

• Shortened DSS -- type 1

❖ $k = g^x \text{ mod } p$

where $x \in_R \{1, K, q-1\}$

❖ $r = \text{hash}(k, m)$

❖ $s = \frac{x}{r + x_a} \text{ mod } q$

❖ output (m, r, s)

$m \longrightarrow (m, r, s)$

• Shortened DSS -- type 2

❖ $k = g^x \text{ mod } p$

where $x \in_R \{1, K, q-1\}$

❖ $r = \text{hash}(k, m)$

❖ $s = \frac{x}{1 + x_a \cdot r} \text{ mod } q$

❖ output (m, r, s)

A generic signcrypt scheme based on any shortened signature

• Signcrypt by Alice

❖ $k = \text{hash}(y_b^x \bmod p)$
 where $x \in_R \{1, K, q-1\}$

❖ $k \begin{cases} \rightarrow k_1 \\ \rightarrow k_2 \end{cases}$

❖ $r = KH_{k_2}(m)$

❖ $s = s(x, r, x_a, q, K)$

$c = E_{k_1}(m)$

❖ **output** (c, r, s)

• Unsigncrypt by Bob

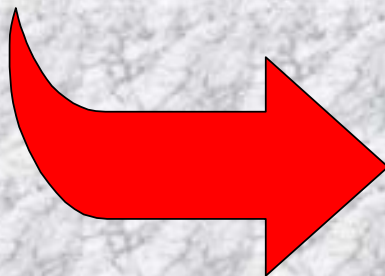
❖ $k = \text{hash}(k(g, r, s, x_b, y_a, p, K))$

❖ $k \begin{cases} \rightarrow k_1 \\ \rightarrow k_2 \end{cases}$

❖ $m = D_{k_1}(c)$

❖ **output**

$$\begin{cases} m & \text{if } r = KH_{k_2}(m) \\ \text{"invalid"} & \text{if } r \neq KH_{k_2}(m) \end{cases}$$



**unforgeability
non-repudiation
&
confidentiality**

Unforgeability of Signcryption

- No body, even Bob the recipient, can forge a valid & signcrypted text from Alice
- How to prove it ?
 - ❖ in the random oracle model
 - ❖ use Pointcheval & Stern's Eurocrypt96 proof technique

Non-repudiation of Signcryption

- With signature-then-encryption, the origin of a decrypted message can be universally verified.
- The origin of an unsigncrypted message, however, can be **directly** verified only by Bob the recipient (using his secret key). It does NOT satisfy “universal verifiability”.

Non-repudiation of Signcryption (cnt'd)

- **“Direct verifiability by the recipient only” is exactly what Alice and Bob want in normal situations !**
- **It is also a main reason (aside from security consideration) why traditionally one uses “signature-then-encryption” rather than “encryption-then-signature” !!**

Non-repudiation of Signcryption (cnt'd)

- **Alice cannot deny the fact that she is the originator of a signcrypted text.**
- **When requested by Bob, a judge can settle a dispute through interactions with Bob.**

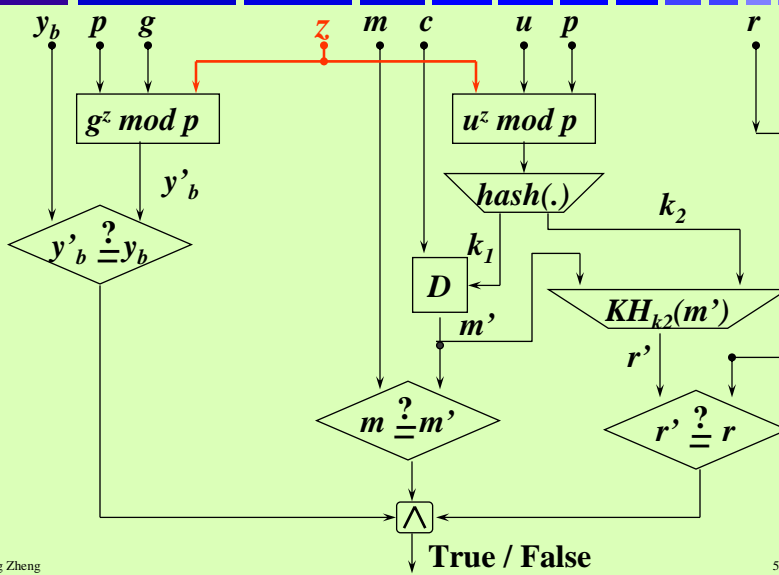
Repudiation Settlement Methods

- Simple if
 - ❖ a trusted tamper-resistant device is used, or
 - ❖ the judge is 100% trusted
- Using a ZK interactive protocol (s.a. Bellare-Jakobsson-Yung Protocol presented at Eurocrypt97) if Bob does not trust the judge
 - Bob “guides” the judge to verify the origin of a message, without revealing his private key x_b

(C) 1997+ Yuliang Zheng

51

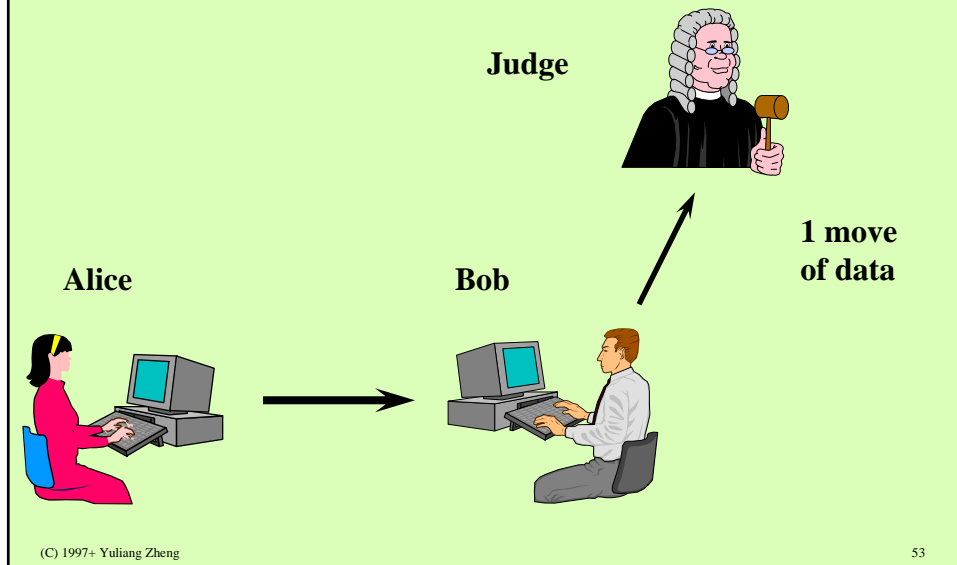
Circuit for ZK Interactive Repudiation Settlement



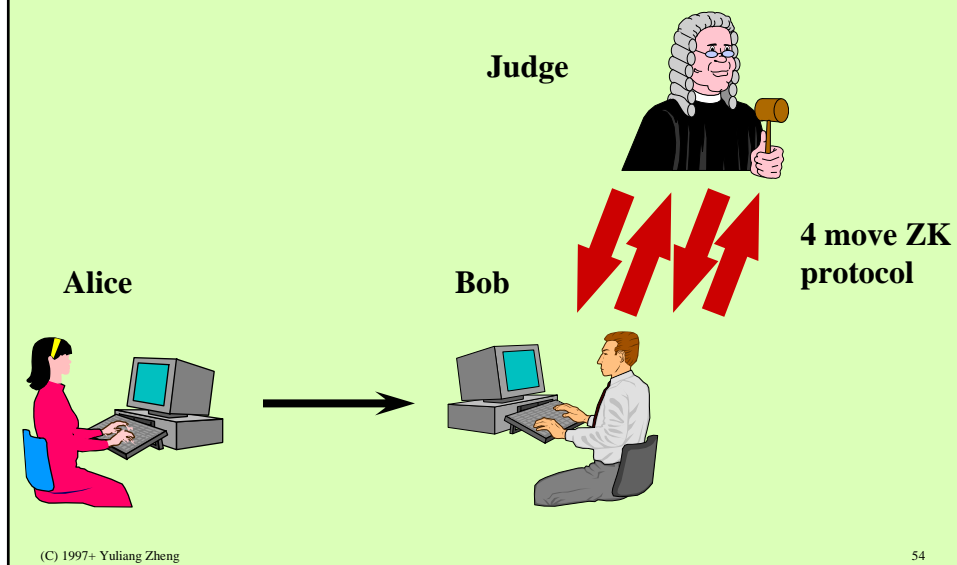
(C) 1997+ Yuliang Zheng

52

Repudiation Settlement for Signature-then-Encryption



Repudiation Settlement for Signcryption



Confidentiality of Signcryption

- A third party cannot obtain information on a message m sealed by a signcryption scheme, if all the underlying primitives are secure (incl: $\langle E, D \rangle$, KH, DH, etc)

Confidentiality of Signcryption (cnt'd)

- How to prove it ?
 - ❖ an attacker for a signcryption scheme

$$m \rightarrow (c, r, s)$$

can be translated into one for a secure encryption scheme defined by

$$m \rightarrow (c, u, r)$$

where

$$c = E_{k_1}(m), u = g^x \text{ mod } p, r = KH_{k_2}(m)$$

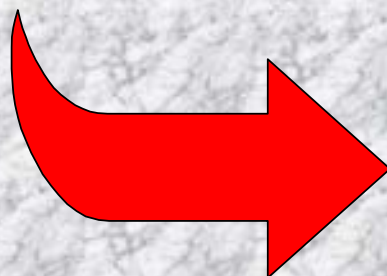
$$k_1 || k_2 = \text{hash}(y_b^x \text{ mod } p)$$

Other aspects of signcryption v.s. sign-then-enc

Attribute paradigm	forward secrecy w.r.t. Alice	past recovery	static key manage.	Repudi. Settle.	“group” orient.
signcryption	no	yes	n/a	Inter- active	yes
sign-then-enc	yes (but, forgeable)	no	n/a	non-inter- active	no
sign-then-enc with a static key	no	yes	distrib/ derivation/ storage	non-inter- active	yes (in most cases)

(C) 1997+ Yuliang Zheng

57

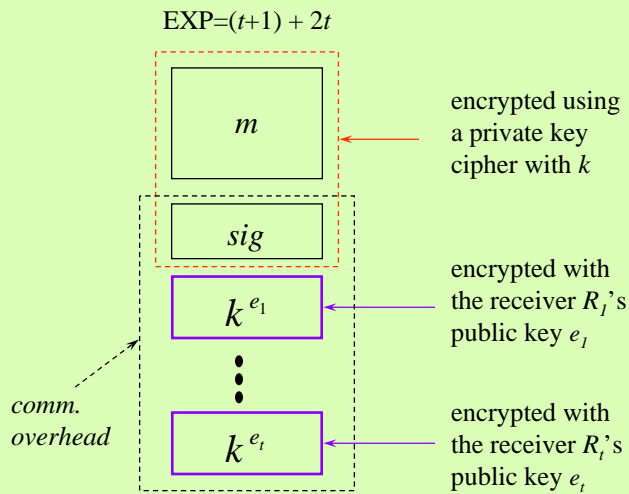


**signcryption
for multiple
recipients**

(C) 1997+ Yuliang Zheng

58

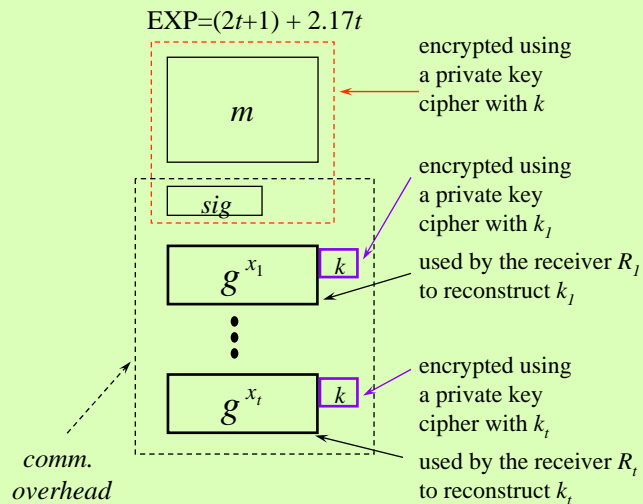
Signature-then-Encryption for Multi-Recipients (RFC1421, RSA)



(C) 1997+ Yuliang Zheng

59

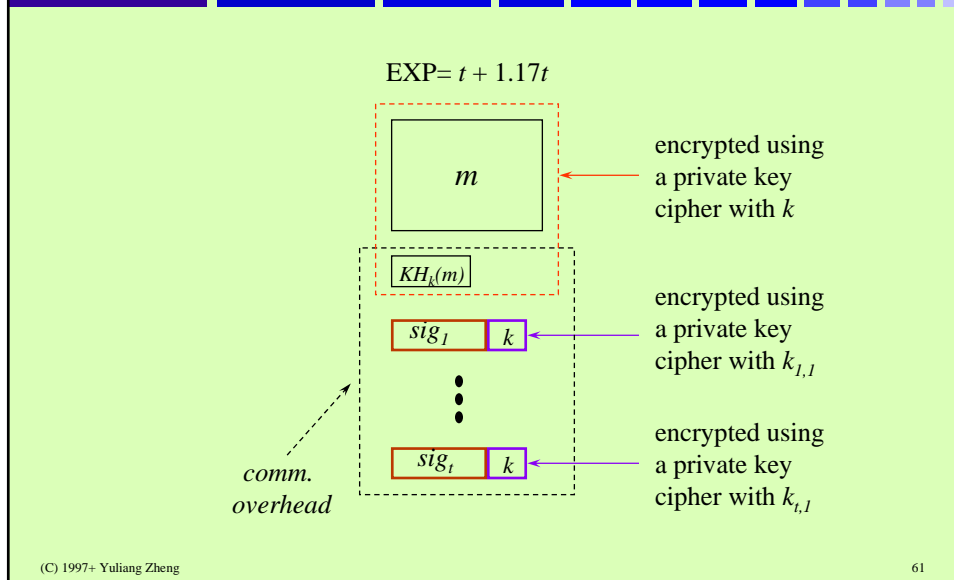
Signature-then-Encryption for Multi-Recipients (based on DL)



(C) 1997+ Yuliang Zheng

60

Signcryption for Multi-Recipients (based on DL)



Signcryption for multiple recipients -- public & secret parameters

- **Public to all**

- ❖ p : a large prime
- ❖ q : a large prime factor of $p-1$
- ❖ g : $0 < g < p$ & with order $q \bmod p$
- ❖ *hash* : 1-way hash
- ❖ *KH* : key-ed 1-way hash
- ❖ (E,D) : private-key encryption & decryption algorithms

- **Alice's keys**

- ❖ x_a : secret key
- ❖ y_a : public key
(note : $y_a = g^{x_a} \bmod p$)

- **Recipient R_i 's keys**

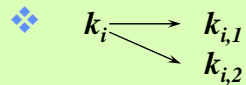
- ❖ x_i : secret key
- ❖ y_i : public key
(note : $y_i = g^{x_i} \bmod p$)

Signcryption by Alice for recipients R_1, \dots, R_t

$c = E_k(m \| h)$, where $h = KH_k(m)$ and k is a random key

- **for $i = 1, \dots, t$**

$k_i = \text{hash}(y_i^{v_i} \bmod p)$ with $v_i \in_R \{1, K, q-1\}$



$c_i = E_{k_{i,1}}(k)$

$r_i = KH_{k_{i,2}}(m, h)$

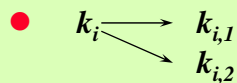
$s_i = v_i / (r_i + x_a) \bmod q$

- **broadcast** $(c, c_1, r_1, s_1, K, c_t, r_t, s_t)$

Unsigncryption by each recipient $R_i, i=1, \dots, t$

- **find out** (c, c_i, r_i, s_i) in $(c, d_1, r_1, s_1, K, c_t, r_t, s_t)$

$k_i = \text{hash}((y_a \cdot g^{r_i})^{s_i \cdot x_i} \bmod p)$

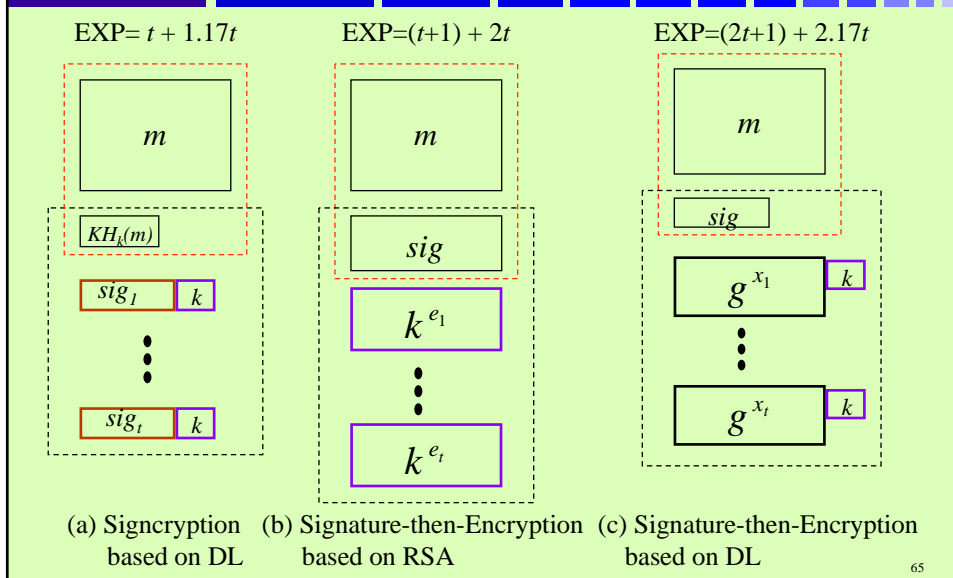


$k = D_{k_{i,1}}(c_i)$

- $w = D_k(c) \quad w \begin{cases} \longrightarrow m \\ \searrow h \end{cases}$

- **output** $\begin{cases} m & \text{if } h = KH_k(m) \text{ and } r_i = KH_{k_{i,2}}(w) \\ \text{"invalid"} & \text{otherwise} \end{cases}$

Signcryption v.s. Signature-then-Encryption for Multi-Recipients



Cost-saving of signcryption for t recipients

Schemes	Cost	comp. Cost (no. of exp.)	comm. overhead (bits)
Schnorr signature + ElGamal encryption		Alice: $2t + 1$ R_i : 2.17	$t \cdot (k + p) + KH + q $
RFC1421 (RSA)		Alice: $t + 1$ R_i : 2	$ n_a + \sum_{i=1, K, t} n_i $
signcryption		Alice: t R_i : 1.17	$t \cdot (k + KH + q) + KH $

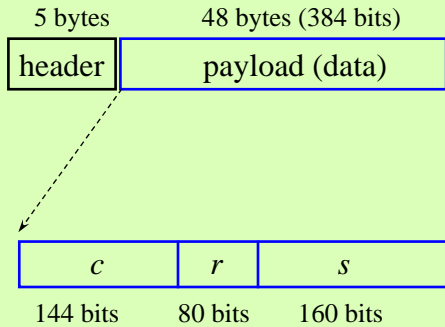


Applications of Signcryption

- **Bring to society huge savings in comp. & comm. if used widely in**
 - ❖ **secure & authenticated message delivery / storage**
 - ❖ **electronic commerce**
 - **secure & authenticated transactions !!!**
 - ❖ **secure & authenticated multicast (incl. video conference, CSCW etc)**
 - ❖ **fast, compact, secure, unforgeable & non-repudiated key transport**

Secure and authenticated key transport in a single ATM cell

ATM Cell



$|p| \geq 512, |q| \geq 160, |KH(\cdot)| \geq 80$

$(k_1, k_2) = \text{hash}(y_b^x \text{ mod } p)$

with $x \in_R [1, K, q-1]$

$|k_1| \geq 64, |k_2| \geq 64$

$c = E_{k_1}(\text{key}, TQ)$

$r = KH_{k_2}(\text{key}, TQ, \text{other})$

$s = \frac{x}{r + x_a} \text{ mod } q$

(C) 1997+ Yuliang Zheng

69

Signcryption cannot be achieved with a shared secret key alone

- Alice and Bob may use a shared secret key, such as $g^{x_a \cdot x_b} \text{ mod } p$ or a KPS key, to carry out secure and efficient communications with content integrity.
- But, without a tamper-proof device and/or a trusted 3rd party, such communications may not achieve unforgeability or non-repudiation.

(C) 1997+ Yuliang Zheng

70

Extensions

- **Signcryption** can be built on other versions of the discrete logarithm, such as those on **elliptic curves**.
- Lenstra's new sub-groups presented at ACISP'97 can also be used in signcryption.

Stop Press, May 2000

- We have recently succeeded in implementing signcryption schemes based on factoring RSA moduli (details are available upon request).
- Thus, signcryption can now be based on
 - ❖ Discrete Log on finite fields
 - ❖ Elliptic curves
 - ❖ Factoring

Summary

- introduced “**digital signcryption**” that achieves
 - ❖ confidentiality
 - ❖ unforgeability & non-repudiation
- proposed concrete **implementations**
- analysed the **significant saving** of signcryption over “signature-then-encryption”