1. Shortened Digital Signature Schemes
2. Signcryption Schemes
3. Compact and Unforgeable Key Agreement Schemes

Yuliang Zheng

Monash University, Australia

August 1998

http://www.pscit.monash.edu.au/~yuliang/

MONASH AUSTRALIA'S INTERNATIONAL UNIVERSITY

3 Separate Parts

- **3 separate contributions**
  - **2 shortened digital signature schemes**
  - **signcryption schemes**
  - **compact & unforgeable key agreement schemes**
- **Technically related, but each can serve as an independent contribution.**

 2

Shortened digital signature

Digital Signature Standard (DSS)

- **Alice's keys**
  - ***p*** **: a large prime**
  - ***q*** **: a large prime factor of *p-1***
  - ***g*** **: *0<g<p* & with order *q* mod *p***
  - ***hash*: 1-way hash**
  - x_a **: secret key**
  - y_a **: public key (note: $y_a = g^{x_a} \bmod p$)**

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

- $x \in_R \{1, \ldots, q-1\}$
- $r = (g^x \bmod p) \bmod q$
- $s = \dfrac{hash(m) + x_a \cdot r}{x} \bmod q$
- **output**

  (m,r,s)

Shortened ElGamal-like signatures

How to shorten ElGamal-like signatures:

- **Calculation of *r***
  - $r = hash(k, m)$

    where

    $k = g^x \bmod p$

    and *x* is a random number.

- **Calculation of *s***
  - if *hash(m)* is in the original *s* :
    - *hash(m)* --> 1
      OR
    - r --> 1, *hash(m)* --> *r*
  - otherwise if *hash(m)* is not in the original *s*, go to next step.
  - if *s = (…)/x*, then change it to *s = x/(…).*

Shortened DSS (2 mthods)

Method 1 $m \longrightarrow (m,r,s)$

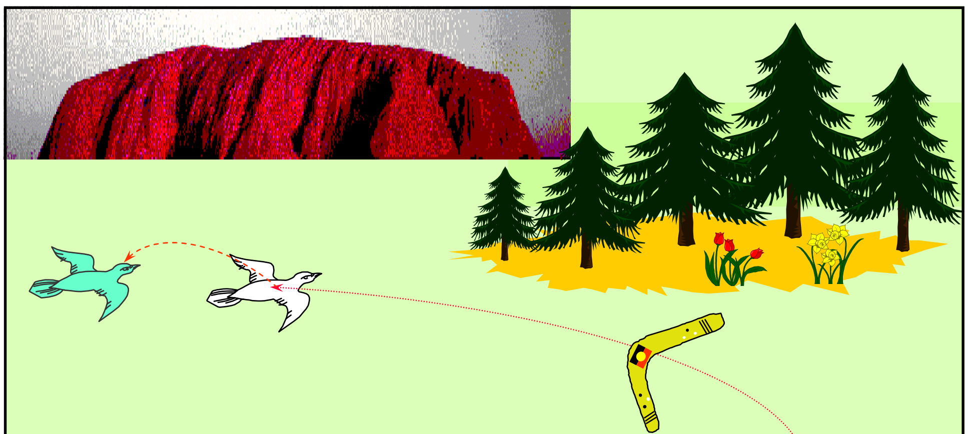
- **Shortened DSS -- type 1**
  - $k = g^x \bmod p$
    where $x \in_R \{1,\ldots,q-1\}$
  - $r = hash(k, m)$
  - $s = \frac{x}{r + x_a} \bmod q$
  - output (m,r,s)

Method 2 $m \longrightarrow (m,r,s)$

- **Shortened DSS -- type 2**
  - $k = g^x \bmod p$
    where $x \in_R \{1,\ldots,q-1\}$
  - $r = hash(k, m)$
  - $s = \frac{x}{1 + x_a \cdot r} \bmod q$
  - output (m,r,s)

Advantages of Shortened DSS

- The *r* part, and hence the signature, is shorter
- No need to do inversion in verification
- Admits “provable” security, using Pointchavel-Stern proof technique, assuming *hash* is an random oracle

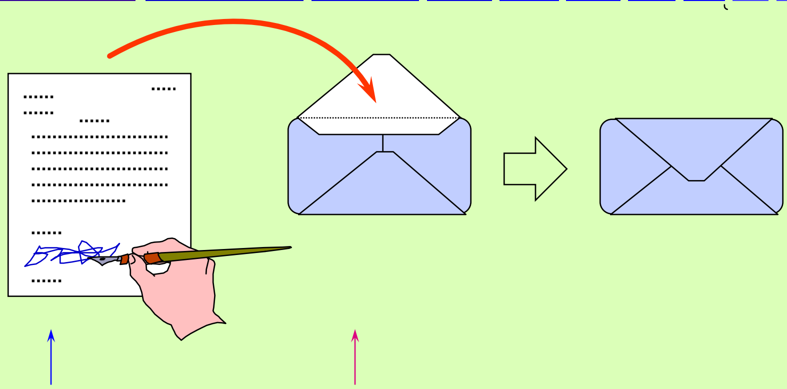


Signcryption or
how o hit 2 birds
using 1 boomerang !

Goals

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

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



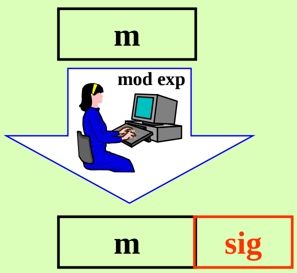
To achieve: *authenticity (unforgeability & non-repudiation)*

To achieve: *confidentiality*

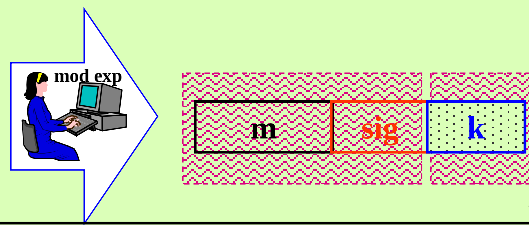
MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

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*.**

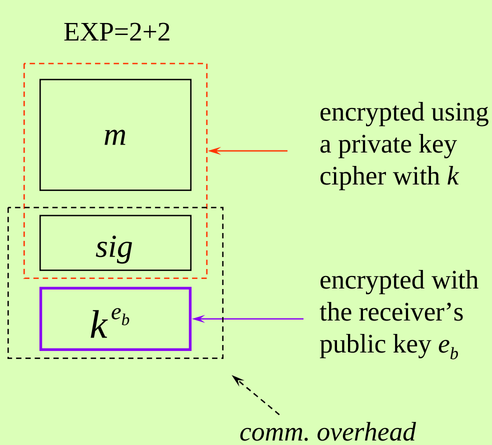


- **2. Encryption**
  - **Alice encrypts (*m*,*sig*) using DES with *k*.**
  - **Alice creates another data so that Bob can recover *k*.** **(Typically, Alice encrypts *k* using Bob's public key).**



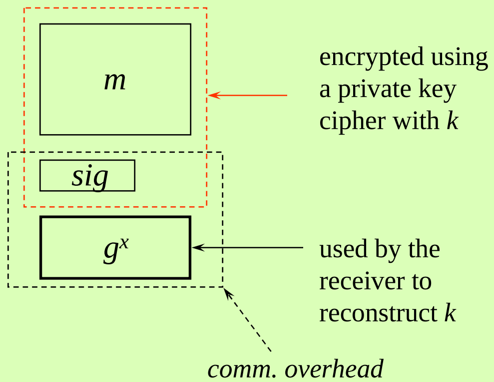
MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

Signature-then-Encryption (based on RSA)



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

EXP=3+2.17



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 + ElGamal enc	3 + 2.17 (3 + 3)	$\|hash\| + \|q\| + \|p\|$

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

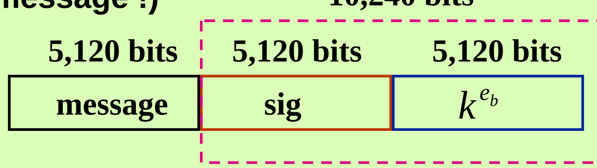
Why signature-then-encryption can be a problem

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

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

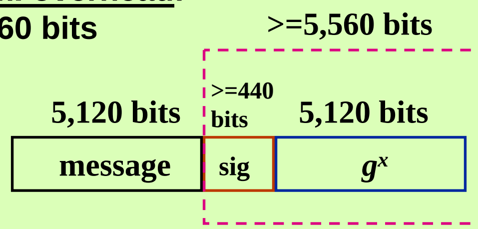
- If RSA with a 5120-bit composite is used
  - <u>Comp. cost</u>:
    2+2=4 exponentiations mod a (very large !) 5120-bit integer
  - <u>Comm. overhead</u>:
    10,240 bits (twice as large as the original message !)

10,240 bits



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 !) 5120-bit integer
  - **Comm. overhead:** >= 5560 bits



(C) 1997-98 by Yuliang Zheng 17

(C) 1997-98 by Yuliang Zheng 18

Signcryption -- a new approach

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

Cost (signcryption)	<<	Cost (signature) + 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* mod *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:**
    - $KH_k(x) \equiv hash(k' \oplus opad, hash(k' \oplus ipad, x))$
      where *k'* is a 0-padded version of *k*, *opad*=x36…36, *ipad*=x5c…5c

Signcryption -- 1st example

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

- **Signcrypt by Alice**
  - $k = hash\ (y_b{}^x \bmod p)$
    where $x \in_R \{1,\ldots,q-1\}$
  - $k \to k_1,\ k_2$
  - $r = KH_{k_2}(m, y_b, etc)$
  - $s = \dfrac{x}{r + x_a} \bmod q$
    $c = E_{k_1}(m)$
  - **output** (c,r,s)

- **Unsigncrypt by Bob**
  - $k = hash\,((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$
  - $k \to k_1,\ k_2$
  - $m = D_{k_1}(c)$
  - **output**

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

Signcryption -- 2nd example

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

- **Signcrypt by Alice**
  - $k = hash\ (y_b{}^x \bmod p)$
    where $x \in_R \{1,\ldots,q-1\}$
  - $k \to k_1,\ k_2$
  - $r = KH_{k_2}(m, y_b, etc)$
  - $s = \dfrac{x}{1 + x_a \cdot r} \bmod q$
    $c = E_{k_1}(m)$
  - **output** (c,r,s)

- **Unsigncrypt by Bob**
  - $k = hash((g \cdot y_a{}^r)^{s \cdot x_b} \bmod p)$
  - $k \to k_1,\ k_2$
  - $m = D_{k_1}(c)$
  - **output**

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

Signcryption -- 3rd example

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

- **Signcrypt by Alice**
  - $k = hash\,(y_b{}^x \bmod p)$
    where $x \in_R \{1,\ldots,q-1\}$
  - $k \to k_1, k_2$
  - $r = KH_{k_2}(m, y_b, etc)$
  - $s = (x - r \cdot x_a) \bmod q$
    $c = E_{k_1}(m)$
  - output (c,r,s)

- **Unsigncrypt by Bob**
  - $k = hash((g^s \cdot y_a{}^r)^{x_b} \bmod p)$
  - $k \to k_1, k_2$
  - $m = D_{k_1}(c)$
  - output

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

 25

Cost of Signcryption

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

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

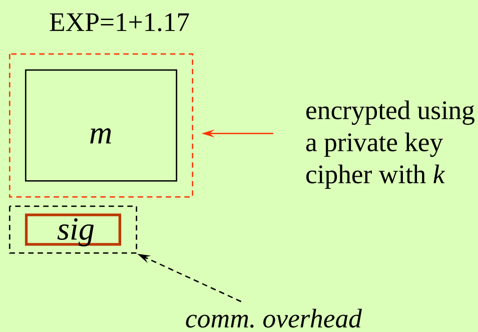
- **Comp. cost**
  - by Alice :
    1 exponentiation modulo p
  - by Bob :
    1.17 exponentiations modulo p (using Shamir's technique)
  - total=2.17 exp mod p

- **Comm. overhead**
  - | r | + | s | bits

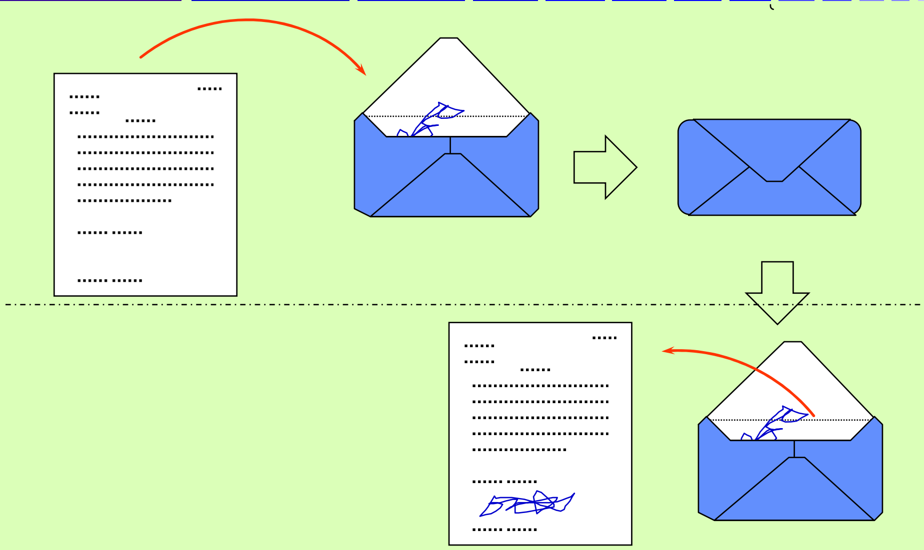
  (note: |m| = |c|)

 26

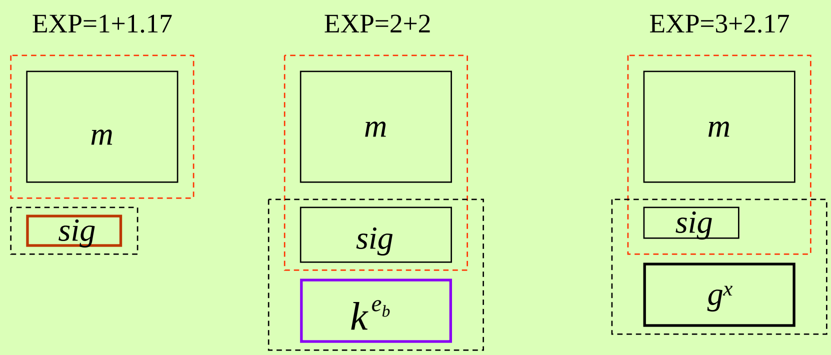
Comp. & Comm. Cost of Signcryption (based on Discrete Logarithm)



“Magic” Signcryption Envelope



Signcryption v.s. Signature-then-Encryption



(a) Signcryption based on DL

(b) Signature-then-Encryption based on RSA

(c) Signature-then-Encryption based on DL

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

A simplistic comparison:

Schemes \ Cost	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)	|hash| + |q| + |p|
DL based Signcryption	1 + 1.17 (1 + 2)	|KH| + |q|

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

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,…,*q*]**
  - **RSA public exponent *e* --- can be small**
  - **RSA secret exponent *d* --- |*n*| bits, BBIIGG !**

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

Signcryption v.s. Schnorr Sig + ElGamal Enc

- **Saving in comp. cost by signcryption**

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

- **Saving in comm. overhead by signcryption (assuming |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 %

Unforgeability of Signcryption

- No body, even Bob the recipient, can forge a valid & signcryted 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" !!

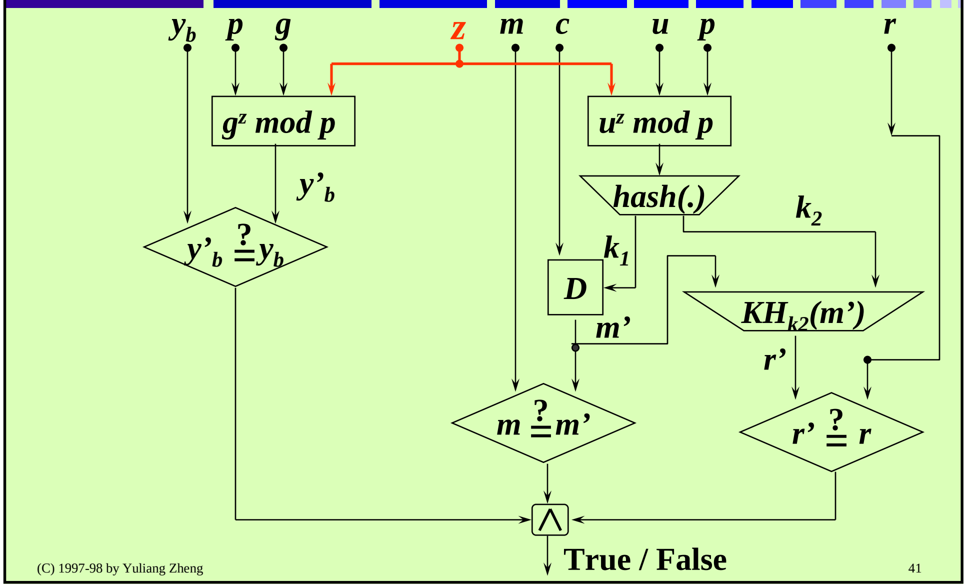
Non-repudiation of Signcryption (cnt'd)

- 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.

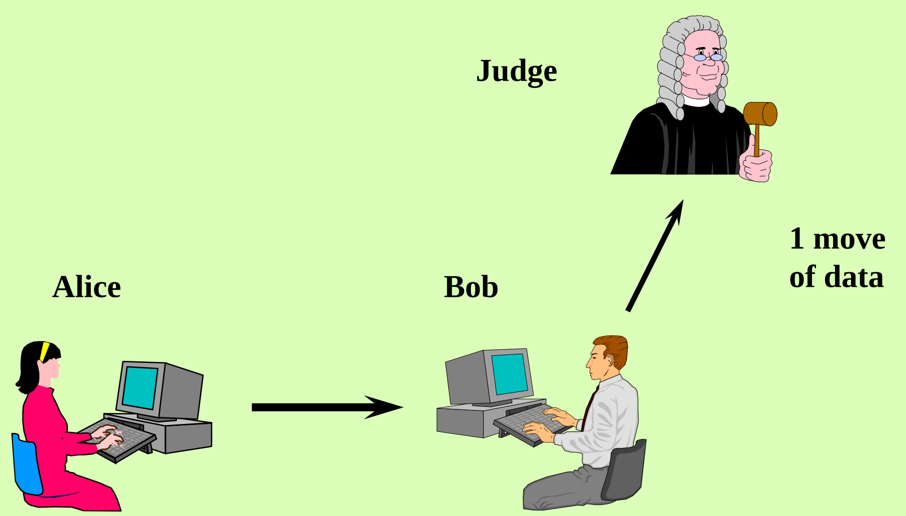
Repudiation Settlement Methods

- 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

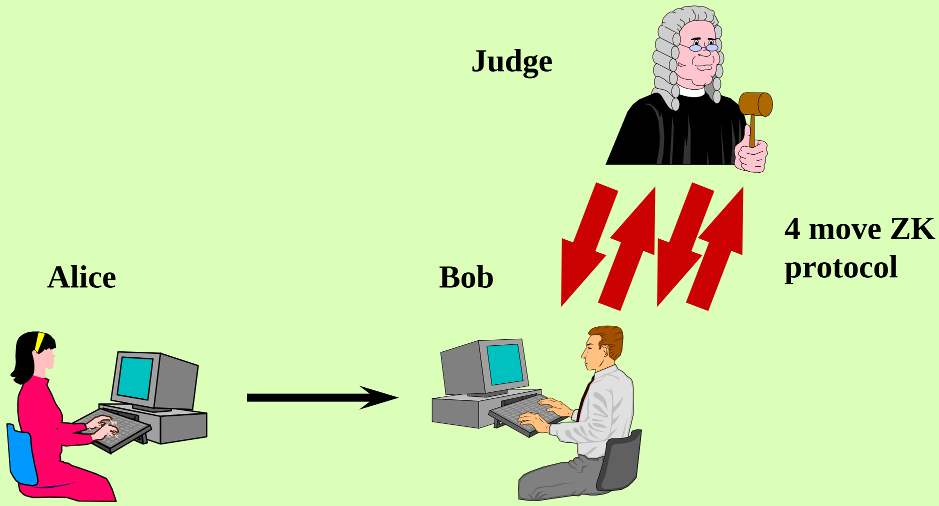
Circuit for ZK Interactive Repudiation Settlement



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: <E,D>, 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

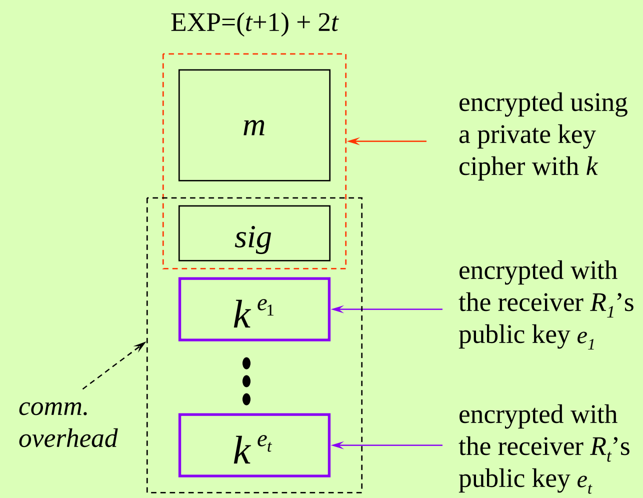
$$c = E_{k_1}(m),\ \ u = g^x \bmod p,\ \ r = KH_{k_2}(m, y_b, etc),$$

$$k_1 \| k_2 = hash(y_b^x \bmod p)$$

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

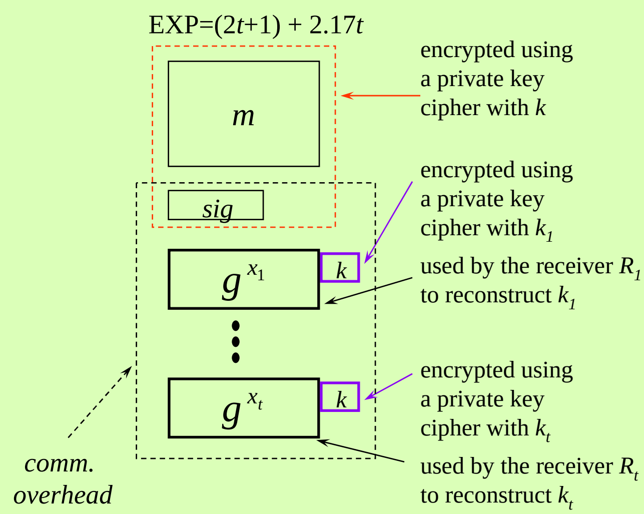
Attribute / paradigm	forward secrecy w.r.t. Alice	past recovery	static key manage.	Repudi. Settle.	"group" orient.
signcryption	no	yes	n/a	Inter-active	yes
sign-then-enc	yes (but, forgeable)	no	n/a	non-inter-active	no
sign-then-enc *with a static key*	no	yes	distrib/ derivation/ storage	non-inter-active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

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



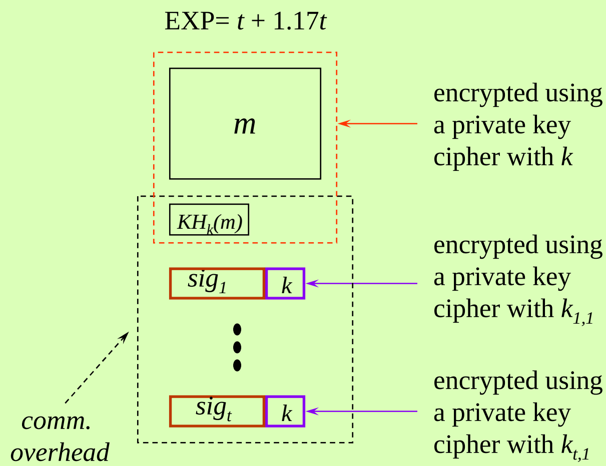
 47

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



 48

Signcryption for Multi-Recipients (based on DL)



 49

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* mod *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 : $y_i = g^{x_i} \bmod p$)**

 50

MONASH AUSTRALIA'S INTERNATIONAL UNIVERSITY

Signcryption by Alice for recipients $R_1,\ldots,R_t$

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

- **for *i* = 1,…,t**

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

  - $k_i \rightarrow k_{i,1}$, $k_i \rightarrow k_{i,2}$

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

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

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

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

MONASH AUSTRALIA'S INTERNATIONAL UNIVERSITY

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

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

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

- $k_i \rightarrow k_{i,1}$, $k_i \rightarrow k_{i,2}$

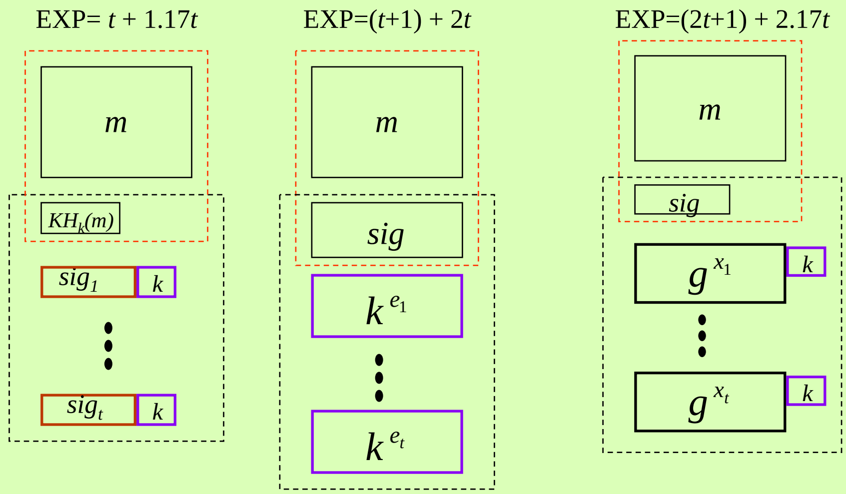
  $k = D_{k_{i,1}}(c_i)$

- $w = D_k(c)$ $w \rightarrow m$, $w \rightarrow h$

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

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

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

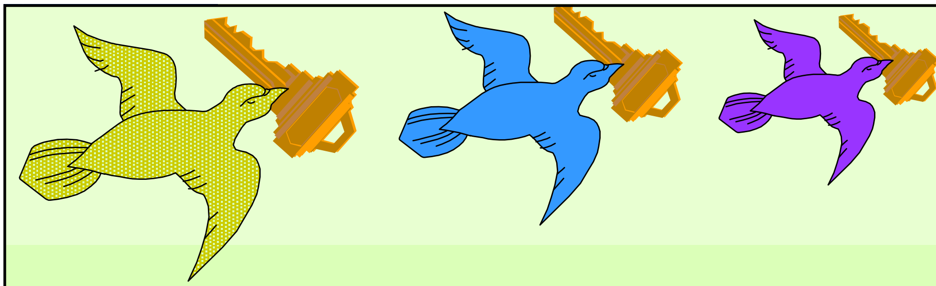


(a) Signcryption based on DL (b) Signature-then-Encryption based on RSA (c) Signature-then-Encryption based on DL

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

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	$\|n_a\|+\sum_{i=1,\ldots,t}\|n_i\|$
signcryption	Alice: t R_i: 1.17	$t \cdot (\|k\|+\|KH\|+\|q\|)+\|KH\|$



Compact and *Unforgeable* Key Establishment

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

Outline of the talk

- **Motivation of this research**
- **Introduction to signcryption**
- **Key materials transport using signcryption**

Session Key Establishment

- **A process for two participants to agree upon a freshly shared key**
- **Dimensions**
  - **security against various attacks**
  - **authenticity v.s. identification**
  - **unforgeability & non-repudiation**
  - **transport v.s. exchange**
  - **secret v.s. public key crypto**
  - **key distrib. center v.s. cert. authority**
  - **efficiency (msg length, # of moves, comp cost)**

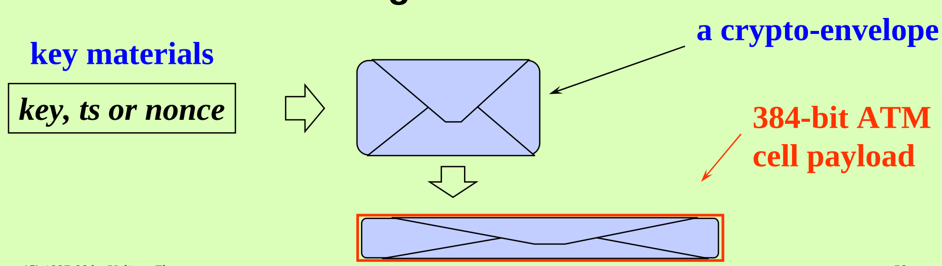
Asynchronous Transfer Mode (ATM) --- Motivation of this Work ---

- **Cell switching**
  - **Data are placed into cells of fixed-size (53 bytes), and then**
  - **transported over virtual circuits**
- **ATM cell structure**

5 bytes	48 bytes (384 bits)
header	payload (data)

Problem to be solved

- To transport encrypted key materials
  - using a single ATM cell
  - with a low computational cost
  - in a secure and unforgeable way
  - without using a KDC



Why using a single ATM cell ?

- If the encrypted version of key materials exceeds 384 bits, problems would occur :
  - splitting data
  - buffering
  - re-assembling data

Why focusing on public key cryptosystems

- The problem CAN be solved using using secret key or types of cryptosystems
- However, with such a solution
  - unforgeability cannot be achieved without a TTP/tamper-proof devices
  - Key management is an issue
    - Distribution
    - Derivation, and/or
    - Secure Storage

Why RSA encryption wouldn't work

64 bits

Using RSA encryption

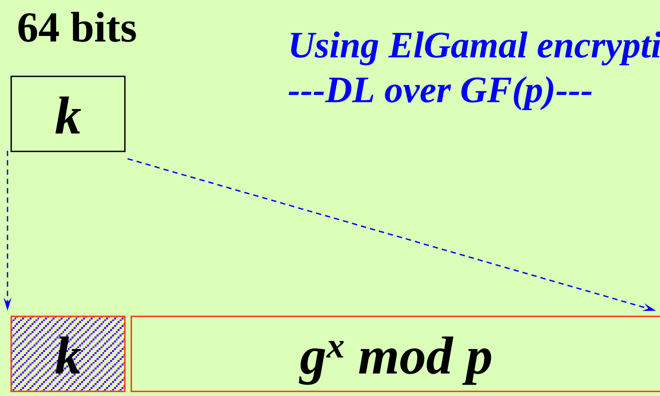
k

$k^e \bmod n$

at least 512 bits

Why ElGamal encryption wouldn't work

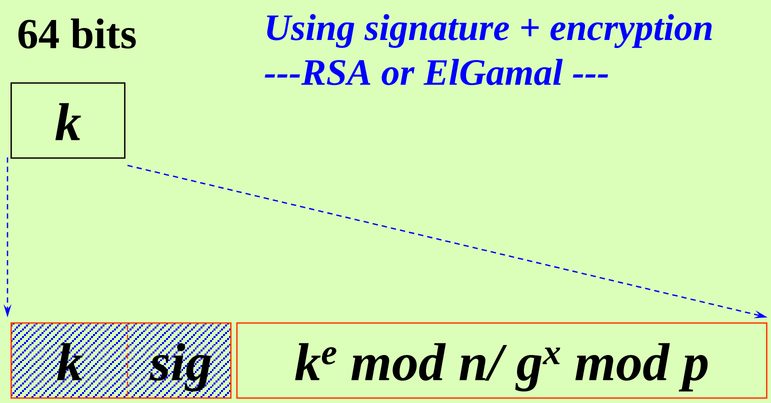
Using ElGamal encryption
---DL over GF(p)---



at least 64+512=576 bits

Why public key "signature + encryption" wouldn't work

Using signature + encryption
---RSA or ElGamal ---

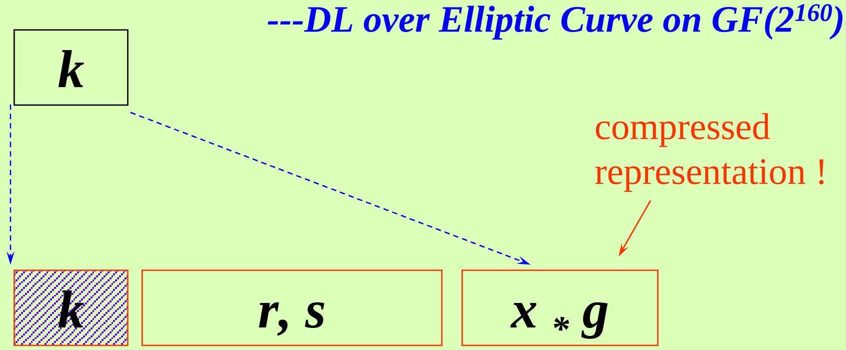


> 512 bits

Why EC-signature+encryption wouldn't work

64 bits

Using Schnorr sig + ElGamal enc
---DL over Elliptic Curve on GF(2^{160})---



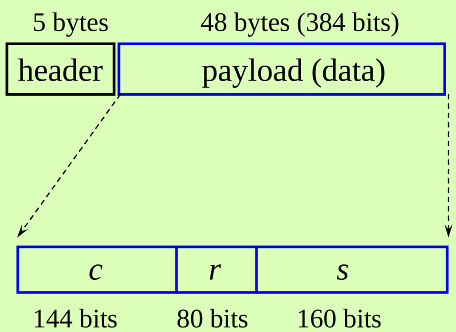
at least 64+(80+160) + (160+1)=465 bits

New approach

- **Using signcryption to transport key materials in a small data packet, s.a. an ATM cell.**

Direct transport of key materials in a small packet

ATM Cell



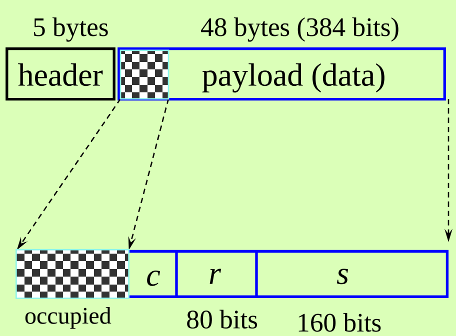
$|p| \geq 512, \ |q| \geq 160, \ |KH_{\cdot}(\cdot)| \geq 80$

$(k_1, k_2) = hash(y_b{}^x \bmod p)$
with $x \in_R [1, \ldots, q-1]$
$|k_1| \geq 64, \ |k_2| \geq 64$

$c = E_{k_1}(key, TQ)$
$r = KH_{k_2}(key, TQ, other)$
$s = \frac{x}{r + x_a} \bmod q$

Indirect transport of key materials in a small packet

ATM Cell



$|p| \geq 512, \ |q| \geq 160, \ |KH_{\cdot}(\cdot)| \geq 80$

$(k_1, k_2) = hash(y_b{}^x \bmod p)$
with $x \in_R [1, \ldots, q-1]$
$|k_1| \geq 64, \ |k_2| \geq 64$

$c = E_{k_1}(TQ)$
$r = KH_{k_2}(TQ, other)$
$s = \frac{x}{r + x_a} \bmod q$

2 Dimensions to be considered

- Direct v.s. Indirect key transport
  - Direct key material transport
    - a random session key is explicitly included in key materials
  - Indirect key material transport
    - a random session key is to be derived from key materials
- Ensuring Freshness using
  - a time-stamp, or
  - a nonce

4 Types of Key Transport Protocols

Time-varying Quantity ↑ / *Transport Mode* →	direct	indirect
Nonce	nonce based direct (3 moves)	nonce based indirect (3 moves)
Time stamp (+nonce)	time-stamp based direct (2 moves)	time-stamp based indirect (2 moves)

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

Direct key transport using a nonce (for unicast)

Alice		Bob
	<= NC_b <=	Pick a nonce NC_b
$c = E_{k_1}(key)$ $r = KH_{k_2}(key, NC_b, etc)$ $s = x/(r + x_a) \bmod q$	=> *c, r, s* =>	unsigncrypt
verify *tag*	<= *tag* <= *(optional)*	$tag = MAC_{key}(NC_b)$

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

Direct key transport using a time-stamp (for unicast)

Alice		Bob
$c = E_{k_1}(key, TS)$ $r = KH_{k_2}(key, TS, etc)$ $s = x/(r + x_a) \bmod q$	=> *c, r, s* =>	unsigncrypt, and check the freshness of TS
verify *tag*	<= *tag* <= *(optional)*	$tag = MAC_{key}(TS)$

MONASH AUSTRALIA'S INTERNATIONAL UNIVERSITY

Indirect key transport using a time-stamp (2 moves)

Alice		Bob
$c = E_{k_1}(TS)$ $r = KH_{k_2}(TS, etc)$ $s = x/(r + x_a) \bmod q$	=> *c, r, s* =>	unsigncrypt, and check the freshness of TS
$key = KH_{k_1,k_2}(TS)$ verify *tag*	<= *tag* <= *(optional)*	$key = KH_{k_1,k_2}(TS)$ $tag = MAC_{key}(TS,1)$

MONASH AUSTRALIA'S INTERNATIONAL UNIVERSITY

How to obtain key exchange protocols

- Let Bob's data or ID be involved in the derivation of a session key
  - E.g.
    - $key^* = KH_{key}(NC_b)$
    - $key^* = KH_{key}(ID_b)$
    - $key^* = KH_{key}(NC_b, ID_b)$
- Let both Alice & Bob generate key & exchange key materials (which achieves mutual identification).

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

Direct key exchange using a nonce (for unicast)

Alice		Bob
	<= NC_b <=	Pick a nonce NC_b
$c = E_{k_1}(key)$ $r = KH_{k_2}(key, NC_b, etc)$ $s = x/(r + x_a) \bmod q$	=> c, r, s =>	unsigncrypt
unsigncrypt	<= c*, r*, s* <=	$c^* = E_{k^*_1}(key^*)$ $r^* = KH_{k^*_2}(key^*, key, etc)$ $s^* = x^*/(r^* + x_b) \bmod q$

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

ATM Forum Proposals

- Two protocols, both based on X.509
  - 2-way protocol
  - 3-way protocol
- Correspondence
  - ATM 2-way <=> direct key exchange using a time-stamp
    ATM 3-way <=> direct key exchange using a nonce

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

ATM Forum 2-Way Protocol (based on sign-then-enc)

Alice		Bob
$\Rightarrow$	$ID_a, ID_b, SecOpt, \{T_a, R_a, \{Enc_{K_b}(ConfPar_a)\},$ $Sig_{K_a}(hash(ID_a, ID_b, T_a, R_a, SecOpt, \{ConfPar_a\}))\}$	$\Rightarrow$
$\Leftarrow$	$ID_a, ID_b, R_a, \{Enc_{K_a}(ConfPar_b)\},$ $Sig_{K_b}(hash(ID_a, ID_b, R_a, \{ConfPar_b\}))\}$	$\Leftarrow$

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

ATM Forum 3-Way Protocol (based on sign-then-enc)

Alice		Bob
$\Rightarrow$	$ID_a, \{ID_b\}, R_a, SecNeg_a, \{Cert_a\}$	$\Rightarrow$
$\Leftarrow$	$ID_a, ID_b, SecNeg_b, \{Cert_b\}, \{R_a, R_b, \{Enc_{K_a}(ConfPar_b)\},$ $Sig_{K_b}(hash(ID_a, ID_b, R_a, R_b, SecNeg_a, SecNeg_b, \{ConfPar_b\}))\}$	$\Leftarrow$
$\Rightarrow$	$ID_a, ID_b, R_b, \{Enc_{K_b}(ConfPar_a)\},$ $Sig_{K_a}(hash(ID_a, ID_b, R_b, \{ConfPar_a\}))\}$	$\Rightarrow$

Advantages of Our Signcryption based Protocols over ATM Forum's

- Significant savings in
  - computational time and
  - communication overhead

Comparison with Beller-Yacobi protocol

Attributes / protocols	Comp. Cost (# of exp)	Longest Msg	Pre comp.
Beller-Yacobi	1 + 2.25 (1 + 4)	>= 512 bits	Yes
Our protocols	1 + 1.17 (1 + 2)	< = 384 bits	Yes*

* Only when Alice knows whom to communicate with

About "Forward Secrecy"

- **Forward secrecy w.r.t. a participant**
  - compromise of the participant's long term secret key does NOT result in the exposure of past session keys
  - Beller-Yacobi protocol
    - YES w.r.t. Alice, NO w.r.t. Bob
  - Our protocols
    - NO w.r.t. either Alice or Bob

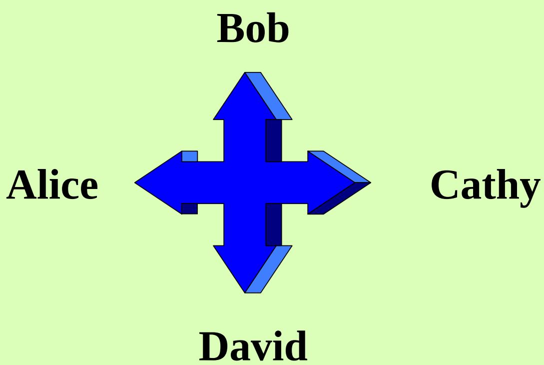
About Forward Secrecy (cnt'd)

- **Forward secrecy w.r.t. Alice CAN be obtained in our proposals**
  - by making a Alice's long term secret key x_a hard to compromise
  - E.g.
    secret sharing,
    mathematically and/or physically

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

Extensions

- the proposed protocols can be extended to "multi-cast" conference key establishment



Direct multicast key transport using a nonce

Alice & each R_i, $I=1,\ldots,t$ $NC = NC_1 + \ldots + NC_t$	NC_1 <= ….. <= NC_t	**Each R_i, $I=1,\ldots,t$** **Pick a nonce NC_b**
Alice: $key \in_R \{0,1\}^{l_1}$, $k \in_R \{0,1\}^{l_2}$ $h = KH_k(key, NC, etc)$ $c = E_k(key, h)$ for each $i = 1,..,t$ $v_i \in_R [1,\ldots,q-1]$ $(k_{i,1}, k_{i,2}) = hash(y_i^{v_i} \bmod p)$ $c_i = E_{k_{i,1}}(k)$ $r_i = KH_{k_{i,2}}(h, etc_i)$ $s_i = \frac{v_i}{r_i + x_a} \bmod q$	c c_1, r_1, s_1 => …... => c_t, r_t, s_t	**Each R_i, $I=1,\ldots,t$** **finds out (c, c_i, r_i, s_i)** **& unsigncrypt it**
Alice & each R_i, $I=1,\ldots,t$ **verify $tag_{1,..,}tag_t$**	tag_1 <= …. <= tag_t *(optional)*	**Each R_i, $I=1,\ldots,t$** $tag_i = MAC_{key}(NC_i)$

Direct multicast key transport using a time-stamp

Alice:

for each $i = 1, .., t$

$v_i \in_R [1, \ldots, q-1]$

$(k_{i,1}, k_{i,2}) = hash(y_i^{v_i} \bmod p)$

$key \in_R \{0,1\}^{l_1}$, $k \in_R \{0,1\}^{l_2}$

get time − stamp TS

$h = KH_k(key, TS, etc)$

$c = E_k(key, TS, h)$

for each $i = 1, .., t$

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

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

$s_i = \dfrac{v_i}{r_i + x_a} \bmod q$

=> **c, c_1, r_1, s_1 …… c_t, r_t, s_t** =>

Each R_i, $I=1,\ldots,t$ finds out (c, c_i, r_i, s_i) & unsigncrypt it

Each R_i, $I=1,\ldots,t$
$tag_i = MAC_{key}(TS, ID_i)$

<= **tag_1 …. tag_t *(optional)*** <=

Alice & each R_i, $I=1,\ldots,t$ verify $tag_{1,..}, tag_t$

Speeding-up through Randomization

- **R_i may decide, in a probabilistic fashion**
  - **whether or not generating NC_i**
  - **whether or not multicasting tag_i**
- **Similarly, Alice and each R_i may randomly choose a subset of tags received for verification**

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

Summary

- 2 shortened DSS schemes
- Signcryption schemes
- Compact and unforgeable key agreement protocols

MONASH
AUSTRALIA'S INTERNATIONAL UNIVERSITY

Q & A