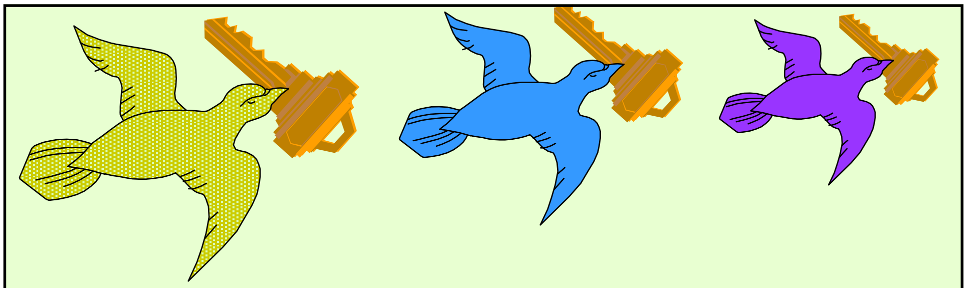
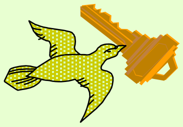


Compact and Unforgeable Key Establishment over an ATM Network

Yuliang Zheng (Monash University, Australia)
Hideki Imai (University of Tokyo, Japan)



Outline of the talk

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

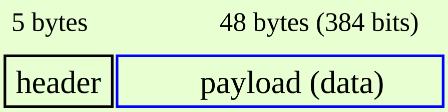
Session Key Establishment

- 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)

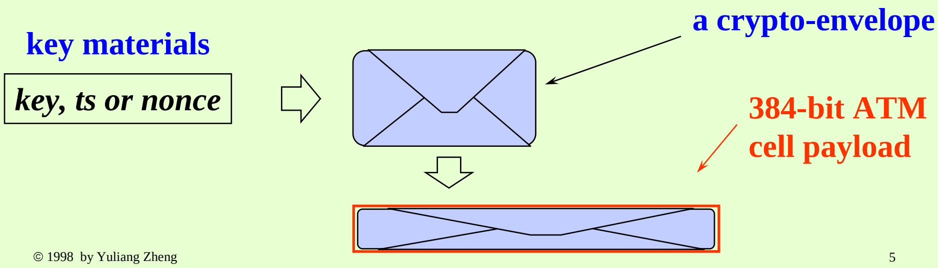
--- Motivation of this Work ---

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



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



key materials

key, ts or nonce

a crypto-envelope

384-bit ATM cell payload

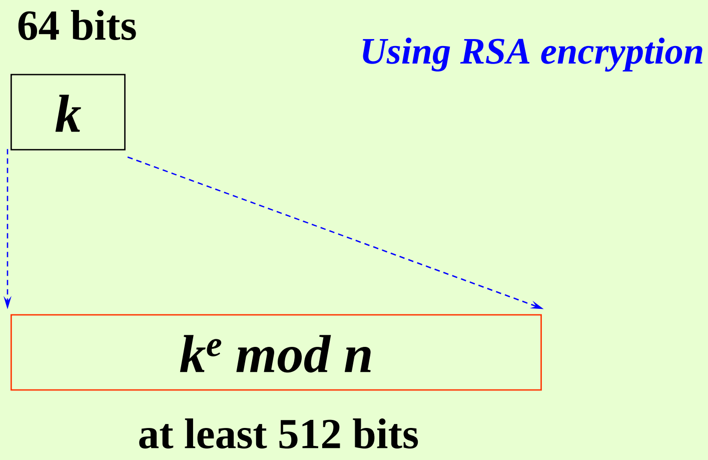
Why using a single ATM cell ?

- 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

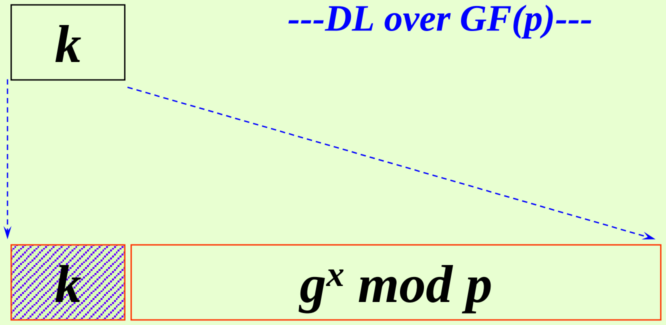
k^e mod n

at least 512 bits

Why ElGamal encryption wouldn't work

64 bits

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

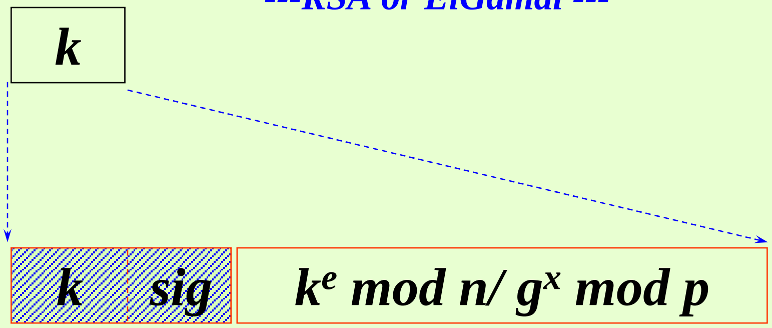


at least 64+512=576 bits

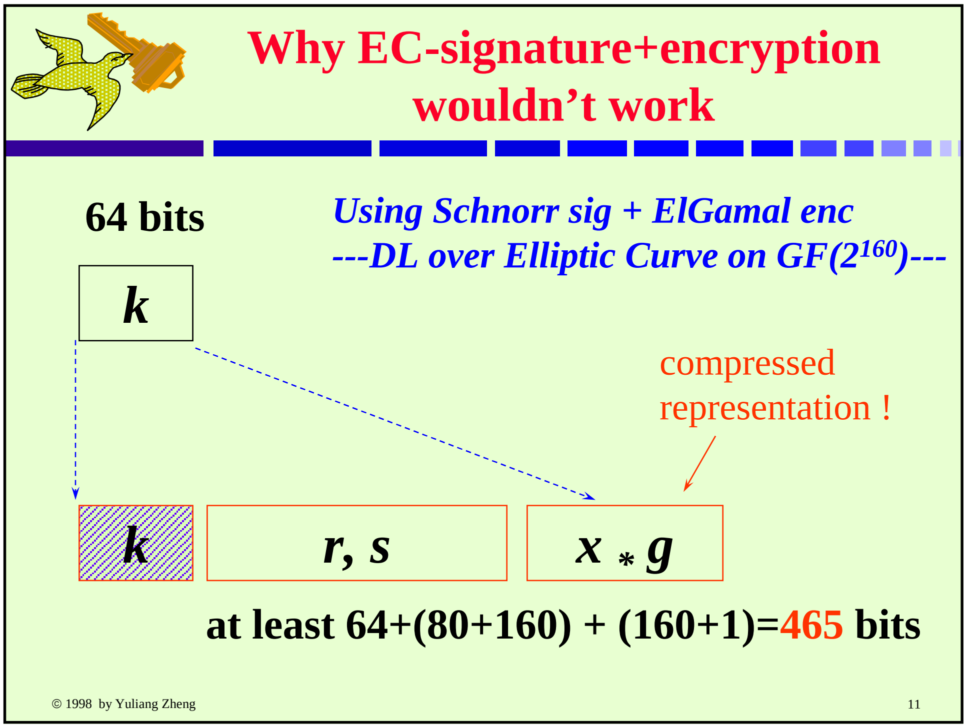
Why public key "signature + encryption" wouldn't work

64 bits

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})---

k

compressed
representation !

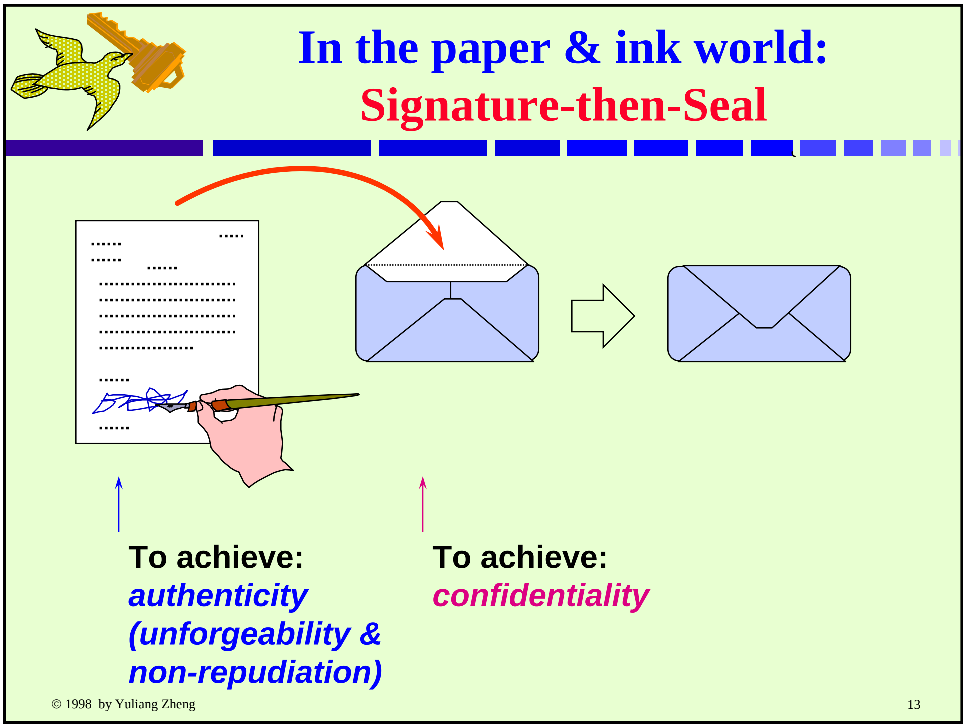
k | ***r, s*** | ***x * g***

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

Signcryption -- a new paradigm

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

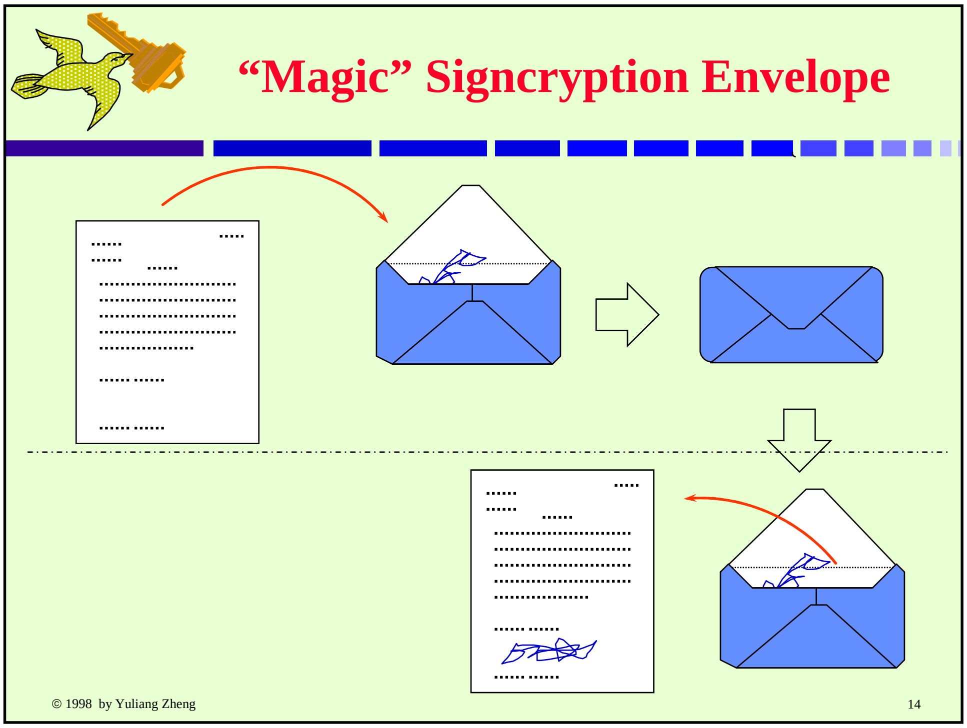
Cost (signcryption) << Cost (signature) + Cost (encryption)



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

To achieve:
authenticity (unforgeability & non-repudiation)

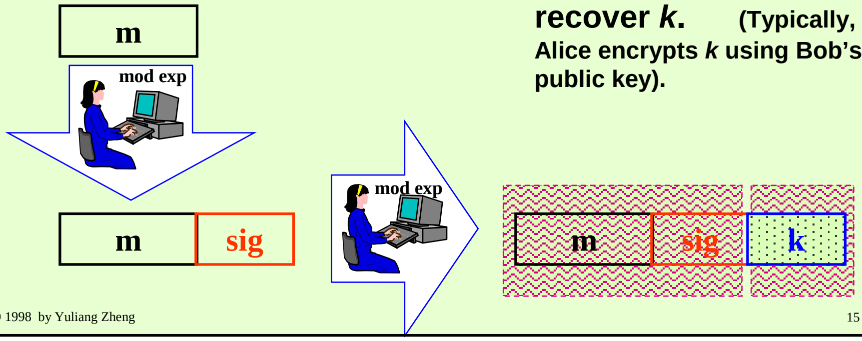
To achieve:
confidentiality



“Magic” Signcryption Envelope

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).

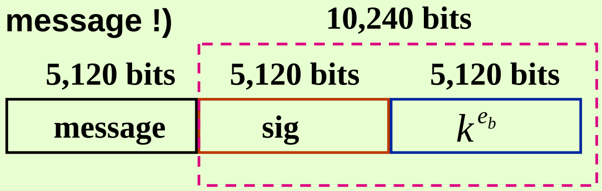


Why signature-then-encryption can be a problem

- 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

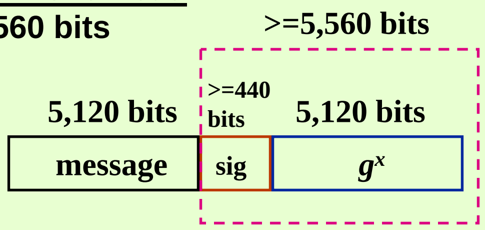
Why signature-then-encryption can be a problem (cnt'd)

- If RSA with a 5120-bit composite is used
  - **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 !)



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



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:** keyed 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 : $y_b = g^{x_b} \bmod p$)

Signcryption -- an example (SCS1)

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

- **Signcrypt by Alice**
  - $k = hash\,(y_b{}^x \bmod p)$ where $x \in_R \{1,\ldots,q-1\}$
  - $$\begin{tikzcd} k \arrow[r] \arrow[dr] & k_1 \\ & k_2 \end{tikzcd}$$
  - $r = KH_{k_2}(m)$
  - $s = \dfrac{x}{r+x_a} \bmod q$

    $c = E_{k_1}(m)$
  - **output** (c,r,s)

- **Unsigncrypt by Bob**
  - $k = hash\,((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$
  - $$\begin{tikzcd} k \arrow[r] \arrow[dr] & k_1 \\ & k_2 \end{tikzcd}$$
  - $m = D_{k_1}(c)$
  - **output**

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

Signcryption -- another example

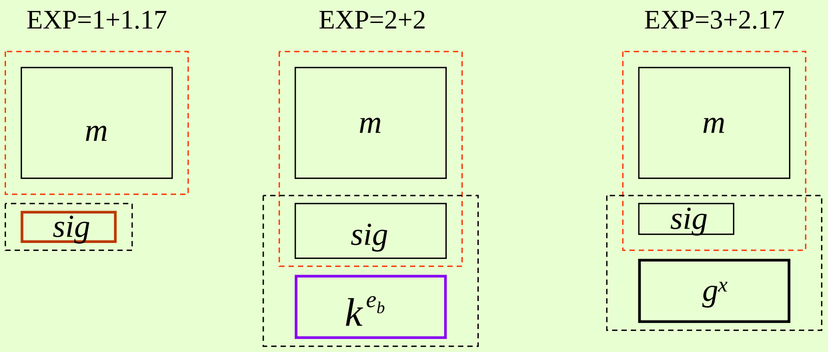
$m \longrightarrow (c,r,s)$ $(c,r,s) \longrightarrow m$

- **Signcrypt by Alice**
  - $k = hash\,(y_b^{\,x} \bmod p)$
    where $x \in_R \{1,\ldots,q-1\}$
  - $$\begin{tikzcd} k \arrow[r] \arrow[dr] & k_1 \\ & k_2 \end{tikzcd}$$
  - $r = KH_{k_2}(m)$
  - $s = (x - r \cdot x_a) \bmod q$
    $c = E_{k_1}(m)$
  - **output** (c,r,s)

- **Unsigncrypt by Bob**
  - $k = hash((g^s \cdot y_a^{\,r})^{x_b} \bmod p)$
  - $$\begin{tikzcd} k \arrow[r] \arrow[dr] & k_1 \\ & k_2 \end{tikzcd}$$
  - $m = D_{k_1}(c)$
  - **output**

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

Signcryption 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:

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|
DL based Signcryption	1 + 1.17 (1 + 2)	|KH| + |q|

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

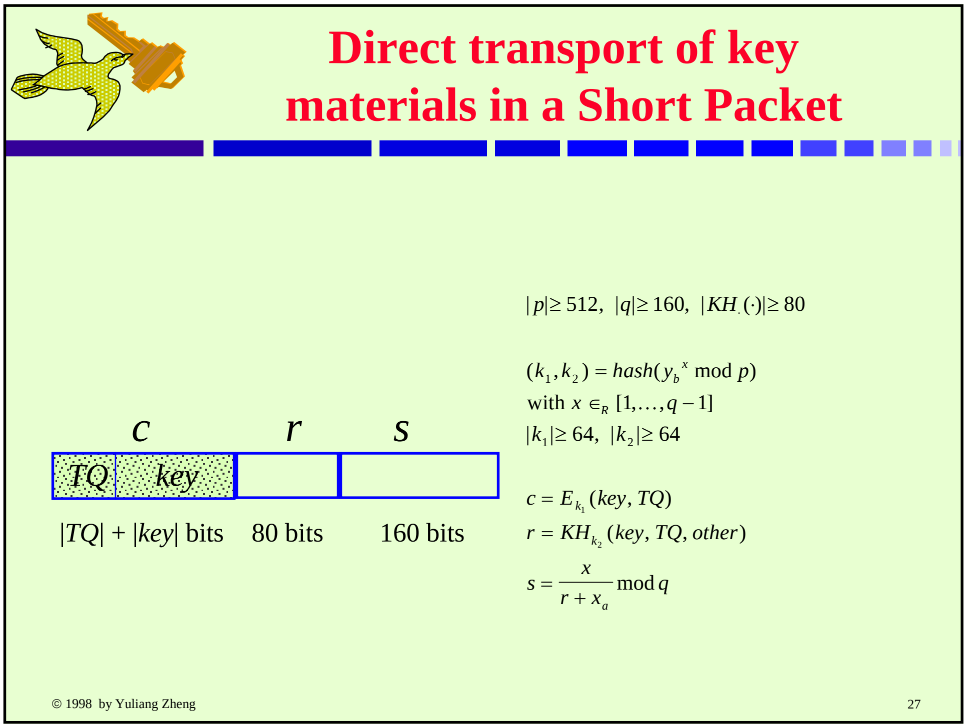
|p|	|q|	|KH|	saving in comp cost	saving in comm overhead
512	144	72	58 %	70.3 %
768	152	80	58 %	76.8 %
1024	160	80	58 %	81.0 %
1536	176	88	58 %	85.3 %
2048	192	96	58 %	87.7 %
3072	224	112	58 %	90.1 %
4096	256	128	58 %	91.0 %
5120	288	144	58 %	92.0 %
8192	320	160	58 %	94.0 %
10240	320	160	58 %	96.0 %

Signcryption v.s. RSA

$\|p\|=\|n_a\|=\|n_b\|$	$\|q\|$	$\|KH\|$	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 %

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**



Direct transport of key materials in a Short Packet

c r s

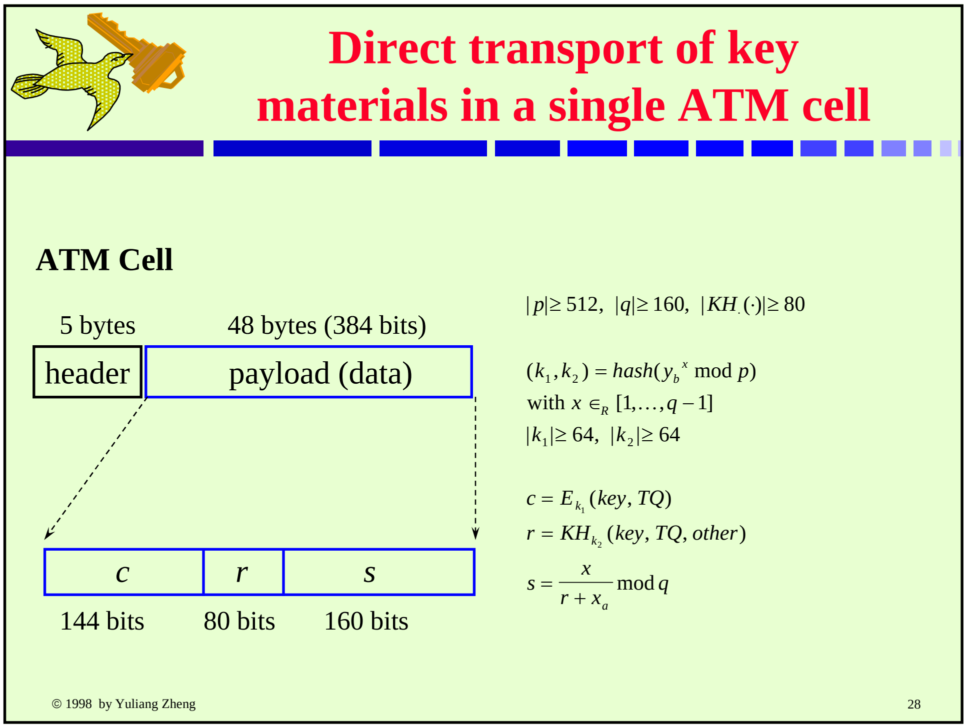
TQ key

$|TQ| + |key|$ bits 80 bits 160 bits

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

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

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



Direct transport of key materials in a single ATM cell

ATM Cell

5 bytes 48 bytes (384 bits)

header payload (data)

c r s

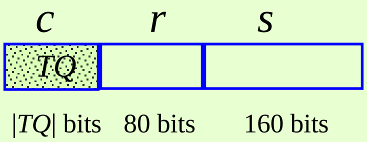
144 bits 80 bits 160 bits

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

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

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

Indirect transport of key materials in a Short Packet



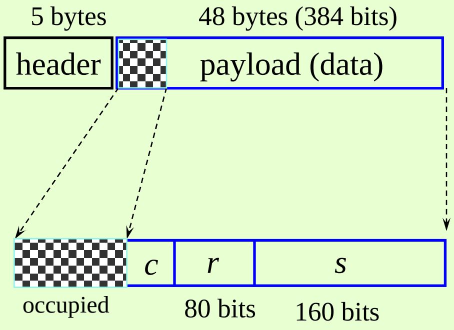
$|p| \ge 512,\ |q| \ge 160,\ |KH_{\cdot}(\cdot)| \ge 80$

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

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

Indirect transport of key materials in a single ATM cell

ATM Cell



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

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

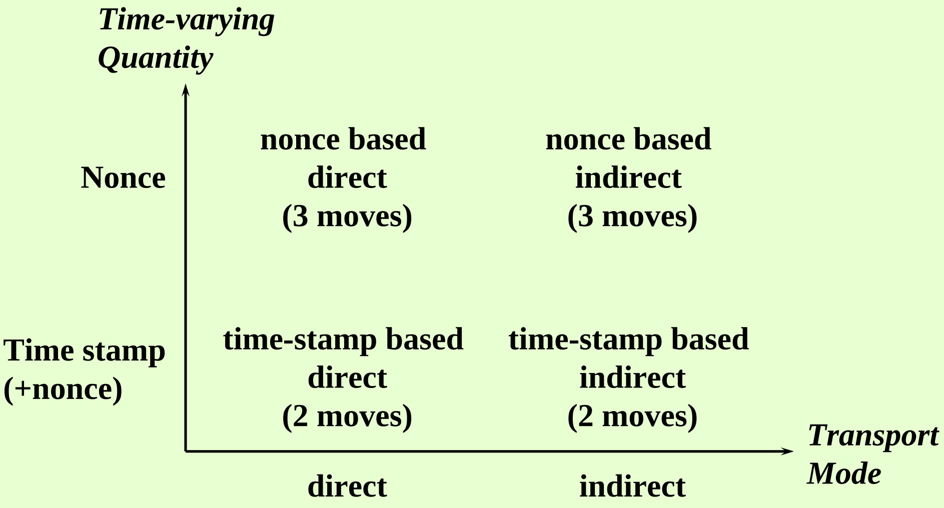
$c = E_{k_1}(TQ)$
$r = KH_{k_2}(TQ, other)$
$s = \dfrac{x}{r + x_a} \bmod q$

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		
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)
Transport Mode	direct	indirect

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)***	$tag = MAC_{key}(NC_b)$

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)***	$tag = MAC_{key}(TS)$

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)***	$key = KH_{k_1,k_2}(TS)$ $tag = MAC_{key}(TS,1)$

How to obtain key exchange protocols

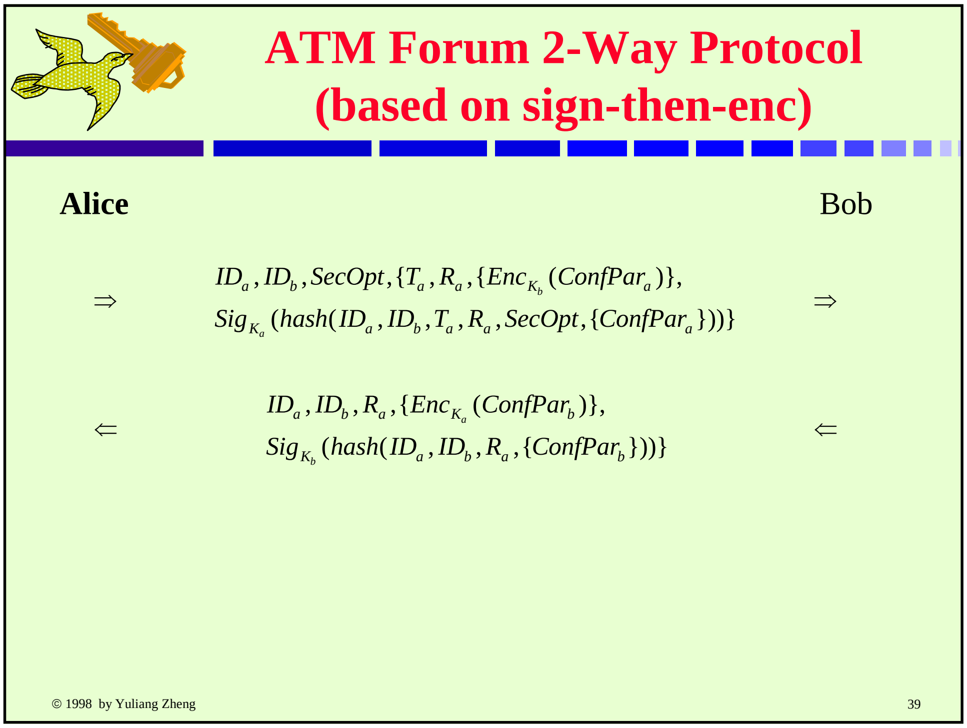
- **Let Bob’s data or ID be involved in the derivation of a session key**
  - **E.g.**
    - **key* = $KH_{key}(NC_b)$**
    - **key* = $KH_{key}(ID_b)$**
    - ***key* = $KH_{key}(NC_b, ID_b)$***
- **Let both Alice & Bob generate key & exchange key materials (which achieves mutual identification).**

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** <=**	$c^* = E_{k^*_1}(key^*)$ $r^* = KH_{k^*_2}(key^*, key, etc)$ $s^* = x^*/(r^* + x_b) \bmod q$

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

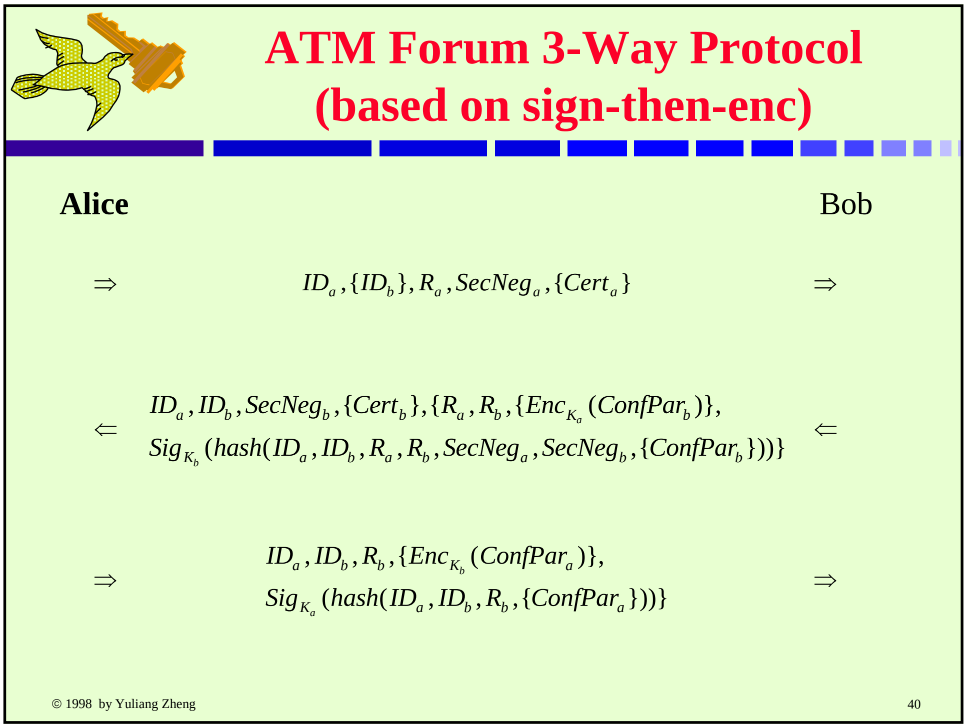


ATM Forum 2-Way Protocol (based on sign-then-enc)

Alice Bob

$$\Rightarrow \quad \begin{array}{l} ID_a, ID_b, SecOpt, \{T_a, R_a, \{Enc_{K_b}(ConfPar_a)\}, \\ Sig_{K_a}(hash(ID_a, ID_b, T_a, R_a, SecOpt, \{ConfPar_a\}))\} \end{array} \quad \Rightarrow$$

$$\Leftarrow \quad \begin{array}{l} ID_a, ID_b, R_a, \{Enc_{K_a}(ConfPar_b)\}, \\ Sig_{K_b}(hash(ID_a, ID_b, R_a, \{ConfPar_b\}))\} \end{array} \quad \Leftarrow$$



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

Alice Bob

$$\Rightarrow \quad ID_a, \{ID_b\}, R_a, SecNeg_a, \{Cert_a\} \quad \Rightarrow$$

$$\Leftarrow \quad \begin{array}{l} ID_a, ID_b, SecNeg_b, \{Cert_b\}, \{R_a, R_b, \{Enc_{K_a}(ConfPar_b)\}, \\ Sig_{K_b}(hash(ID_a, ID_b, R_a, R_b, SecNeg_a, SecNeg_b, \{ConfPar_b\}))\} \end{array} \quad \Leftarrow$$

$$\Rightarrow \quad \begin{array}{l} ID_a, ID_b, R_b, \{Enc_{K_b}(ConfPar_a)\}, \\ Sig_{K_a}(hash(ID_a, ID_b, R_b, \{ConfPar_a\}))\} \end{array} \quad \Rightarrow$$

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”

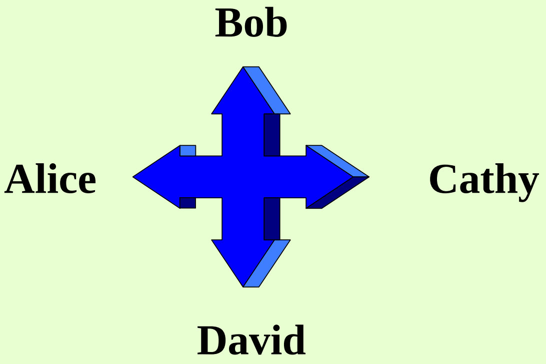
- Forward secrecy w.r.t. a participant
  - compromise of the participant’s long term secret key does NOT result in the exposure of past session keys
  - Beller-Yacobi protocol
    - YES w.r.t. Alice, NO w.r.t. Bob
  - Our protocols
    - NO w.r.t. either Alice or Bob

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

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 $i = 1,..,t$

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

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

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

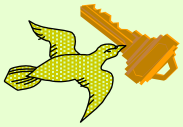
=> ***c***, ***c_1, r_1, s_1***, **……**, ***c_t, r_t, s_t*** =>

Each *R_i, I=1,…,t* finds out (c, *c_i, r_i, s_i*) & unsigncrypt it

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

<= ***tag_1***, **….**, ***tag_t*** ***(optional)*** <=

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



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**

Summary

- addressed the problem of "unforgeable key establishment in small packets s.a. ATM cells"
- solved the problem using signcryption
- Potential applications:
  - high speed networks
  - smart card based security solutions
  - mobile communications, ……